

Semaine 8

Tableaux

Intro. à la programmation



- ❖ Révision
- ❖ Tableaux
- ❖ Boucles et tableaux
- ❖ Tableaux d'éléments HTML



❖ DOM

```
<div class="maClasse"> ... </div>
```

◆ Classes

- Ajouter : `document.querySelector(".classe").classList.add("nouvelle_classe")`
- Retirer : `document.querySelector(".classe").classList.remove("ancienne_classe")`
- Basculer : `document.querySelector(".classe").classList.toggle("classe")`
 - Retire la classe si elle est déjà présente. Ajoute la classe si elle n'était pas présente.
- Contient ? : `document.querySelector(".classe").classList.contains("nom_classe")`
 - Résulte en un booléen (true ou false)


◆ Attributs

- Ajouter / Modifier : `document.querySelector(".classe").nomAttribut = "valeur";`
- Obtenir : `let valeur = document.querySelector(".classe").nomAttribut;`
 - Nous donne la valeur associée à cet attribut.

```
<div class="titre" title="Bannière">Deux attributs</div>
```



- ❖ Astuce pour éviter de réécrire `document.querySelector(...)` pour le même élément plusieurs fois



```
let elementMario = document.querySelector(".mario");  
  
elementMario.style.borderColor = "red";  
elementMario.textContent = "Mario brosse 🍷";  
elementMario.title = "Plombier italien";  
elementMario.style.color = "crimson";  
elementMario.classList.add("peach");
```



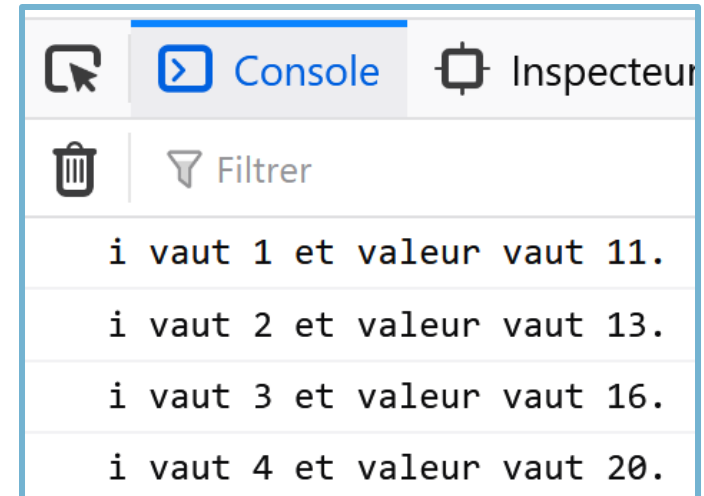
❖ Boucles

◆ Syntaxe :

```
while(condition d'exécution) {  
    // Code à répéter  
}
```

```
let valeur = 10;  
let i = 0;  
  
while(i < 4){  
    i += 1;  
    valeur += i;  
    console.log(`i vaut ${i} et valeur vaut ${valeur}`);  
}
```

La **valeur** finale est : 10 + 1 + 2 + 3 + 4 (Donc 20)





❖ Stocker plusieurs **données similaires**...

◆ Exemple : On souhaite stocker le **nom** et l'**âge** de 7 étudiant(e)s

```
let nomEtudiant1 = "Madeleine";    let age1 = 17;
let nomEtudiant2 = "Omar";          let age2 = 19;
let nomEtudiant3 = "Conrad";        let age3 = 20;
let nomEtudiant4 = "Marie-Gisèle";  let age4 = 104;
let nomEtudiant5 = "Jean-Jérémie";  let age5 = 21;
let nomEtudiant6 = "Sandra";        let age6 = 18;
let nomEtudiant7 = "Bartolomé";     let age7 = 22;
```

- Oulala... ça fait beaucoup de **variables similaires**. On peut se perdre rapidement. 😞



❖ Solution : Mettre les données dans un **tableau**

◆ Les **tableaux** permettent de ranger des données similaires

- Syntaxe pour **créer un tableau** :

```
let couleurs = ["Bleu", "Rouge", "Jaune", "Vert"];
```

Données (Séparées par des **virgules**)

Peuvent être des **nombres**, des **chaînes de caractères**, des **booléens**, etc.

- Exemples :

```
let nomsEtudiants = ["Madeleine", "Omar", "Conrad",  
                    "Marie-Gisèle", "Jean-Jérémie",  
                    "Sandra", "Bartolomé"];
```

```
let ages = [17, 19, 20, 104, 21, 18, 22];
```



❖ Accéder aux données (aux « éléments ») d'un **tableau**

◆ Syntaxe :

`nomTableau[index]`

Nombre de **0** à « **taille du tableau - 1** »

◆ Exemple :

```
let couleurs = ["Bleu", "Rouge", "Jaune", "Vert", "Violet"];
```

- Accéder à la donnée **"Bleu"** et la stocker dans une variable :
-> `let a = couleurs[0];` // a vaut "Bleu"
- Accéder à la donnée **"Violet"** :
-> `couleurs[4]`
- Accéder à la donnée **"Rouge"** :
-> `couleurs[1]`

Index	Donnée
0	"Bleu"
1	"Rouge"
2	"Jaune"
3	"Vert"
4	"Violet"

Donc pour un tableau avec **5 éléments**, l'index va de **0** à **4** !



❖ Modifier une donnée dans un **tableau**

◆ Syntaxe

```
nomTableau[index] = "nouvelle valeur";
```

◆ Exemple :

```
let personnages = ["Mario", "Luigi", "Peach", "Bernard"];
```

- On veut modifier l'élément à l'**index 3** (C'est-à-dire la quatrième donnée : "**Bernard**")

```
personnages[3] = "Yoshi";
```

- Résultat :

```
["Mario", "Luigi", "Peach", "Yoshi"]
```





❖ Obtenir la **taille d'un tableau**

◆ Syntaxe

`nomTableau.length`

◆ Exemple :

```
let ages = [17, 19, 20, 104, 21, 18, 22];  
let longueur = ages.length; // longueur contient la valeur 7
```

```
let message = `Le tableau ages contient ${ages.length} éléments`;  
// message contient "Le tableau ages contient 7 éléments"
```



❖ Utiliser les données d'un **tableau**

```
let prix = [4.5, 3.99, 7.2, 8.4];  
let quantites = [2, 3, 4, 3];  
  
let valeurTotaleArticle1 = prix[0] * quantites[0];  
//      ↑ Vaut 4.5 * 2, donc 9
```

```
let couleurs = ["Bleu", "Rouge", "Violet", "Rose"];  
let elementMessage = document.querySelector(".message");  
  
elementMessage.textContent = `Mes couleurs préférées sont ${couleurs[0]} et ${couleurs[2]}`;  
// "Mes couleurs préférées sont Bleu et Violet";
```



❖ `push()` +

- ◆ Permet d'ajouter un élément à la fin d'un tableau

```
let couleurs = ["Bleu", "Rouge", "Jaune", "Vert"];
```

```
couleurs.push("Violet");
```

```
// couleurs vaut ["Bleu", "Rouge", "Jaune", "Vert", "Violet"]
```

```
// couleurs.length vaut maintenant 5
```

```
["Bleu", "Rouge", "Jaune", "Vert"]
```



```
["Bleu", "Rouge", "Jaune", "Vert", "Violet"]
```



❖ pop() 🗑️

- ◆ Permet entre autres* de retirer un élément à la fin du tableau

```
let notes = [68, 71, 93, 78];
```

```
notes.pop();
```

```
// notes vaut [68, 71, 93]
```

[68, 71, 93, 78]



[68, 71, 93]

*pop() permet aussi de récupérer l'élément retiré dans une variable, mais nous l'utiliserons seulement pour en retirer dans ce cours.



❖ splice()

- ◆ Permet entre autres* de retirer des éléments dans un **tableau**
 - Pas juste à la fin comme **pop()** !
- ◆ Syntaxe pour **retirer** des éléments

monTableau.splice(index, nbRetirer)

Premier élément à retirer

Combien d'élément on retire au total ?

◆ Exemple :

```
let couleurs = ["Bleu", "Rouge", "Jaune", "Vert", "Orange", "Violet"];
couleurs.splice(0, 2);
// couleurs vaut maintenant ["Jaune", "Vert", "Orange", "Violet"]
```

- L'élément à l'**index 0** est le premier à être retiré et est inclus dans le nombre total d'éléments à retirer.

*splice() permet aussi d'ajouter des éléments, mais nous l'utiliserons seulement pour en retirer dans ce cours.



❖ splice()

◆ Autre Exemple :

```
let couleurs = ["Bleu", "Rouge", "Jaune", "Vert", "Orange", "Violet"];  
couleurs.splice(2, 1);  
// couleurs vaut maintenant ["Bleu", "Rouge", "Vert", "Orange", "Violet"]
```

- ◆ "Jaune" était l'élément à l'index 2 et seulement 1 élément devait être retiré, au total.



❖ Les **boucles** et les **tableaux** sont « meilleurs amis ».



◆ Pourquoi ? 🤔

◆ Tentons de calculer la somme de **tous les éléments** d'un **tableau** sans boucle :

```
let prix = [5.49, 1.99, 99.99, 8.49, 7.72];  
let totalPrix = prix[0] + prix[1] + prix[2] + prix[3] + prix[4];
```

Imaginez s'il y avait eu **50** prix à additionner !

◆ Tentons à nouveau, mais avec une **boucle**

```
let prix = [5.49, 1.99, 99.99, 8.49, 7.72];  
let prixTotal = 0;  
let i = 0;  
  
while(i < prix.length){  
    prixTotal += prix[i];  
    i += 1;  
}
```

Avant même d'entrer dans les détails, on dirait déjà qu'il y a beaucoup moins de code répétitif !



❖ Parcourir un **tableau** à l'aide d'une **boucle**

◆ Dans de nombreuses situations, il faut **parcourir un tableau en entier**...

- Afficher tous les éléments
- Calculer la somme ou la moyenne
- Trouver le maximum / minimum
- Trier les éléments par ordre croissant / alphabétique
- etc.

◆ Une **boucle** qui sert à parcourir un **tableau** ressemblera souvent à ceci :

monTableau est une variable qui contient un tableau

i commence à 0 ... et ira jusqu'à **monTableau.length** ...

```
let i = 0;

while(i < monTableau.length){
    // ... Code qui contient monTableau[i] ...
    i += 1;
}
```



❖ Parcourir un **tableau** à l'aide d'une **boucle**

```
let i = 0;

while(i < monTableau.length){
    // ... Code qui contient monTableau[i] ...
    i += 1;
}
```

◆ **monTableau** est le nom de la variable qui contient un tableau.

- ex. `let monTableau = ["a", "b", "c"];`

◆ L'objectif est de parcourir `monTableau[0]`, `monTableau[1]` et `monTableau[2]`

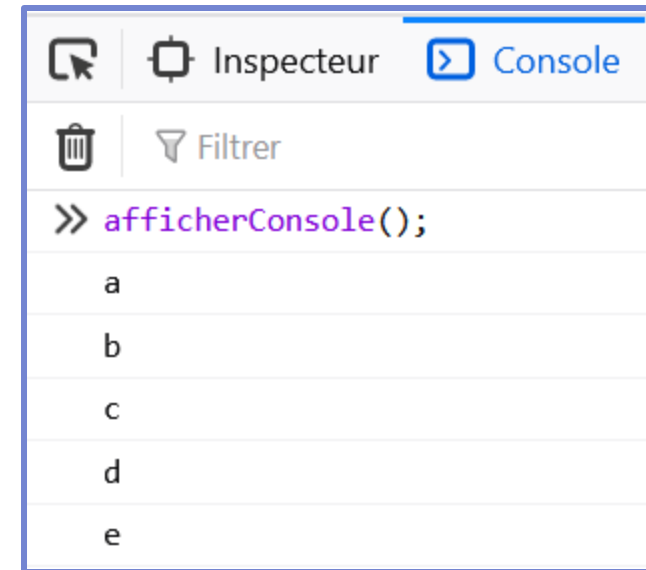
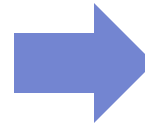
- On a donc besoin que `i` commence à `0`, se rende jusqu'à `2` et augmente de `1` à chaque répétition de boucle.
- Donc `i` vaudra `0` ... puis `1` ... puis `2` ... puis **fin de la boucle** !



❖ Exemple 1

- ◆ Afficher toutes les données d'un tableau dans la console
 - À chaque itération, on affiche la valeur de `lettres[i]` dans la **console**.

```
function afficherConsole(){  
  
    let lettres = ["a", "b", "c", "d", "e"];  
  
    let i = 0;  
    while(i < lettres.length){  
        console.log(lettres[i]);  
        i += 1;  
    }  
}
```



Pourquoi on met `i < lettres.length` plutôt que `i < 5` ? Car `lettres.length` vaut **5** ! Pas besoin de compter les **index** du tableau manuellement.



❖ Exemple 2

◆ Somme des éléments d'un tableau

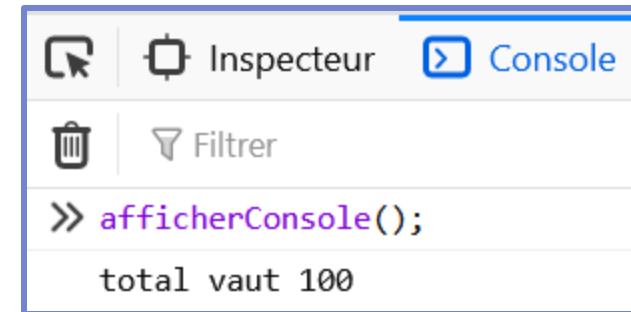
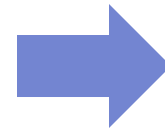
- **total** : Variable pour stocker la **somme** des nombres. On l'initialise à **zéro**.

```
let nombres = [10, 20, 30, 40];

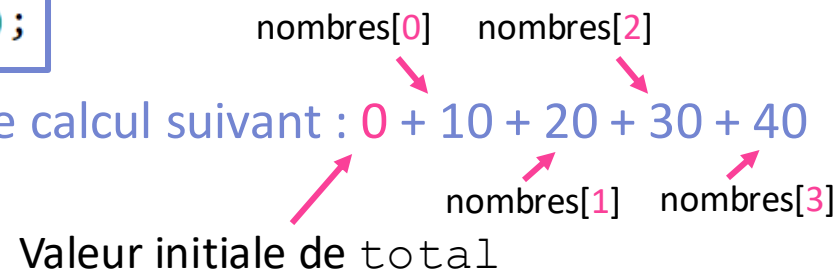
let i = 0;
let total = 0;

while(i < nombres.length){
  total += nombres[i];
  i += 1;
}

console.log(`total vaut ${total}`);
```



- La boucle, avec 4 itérations, fait le calcul suivant : $0 + 10 + 20 + 30 + 40$





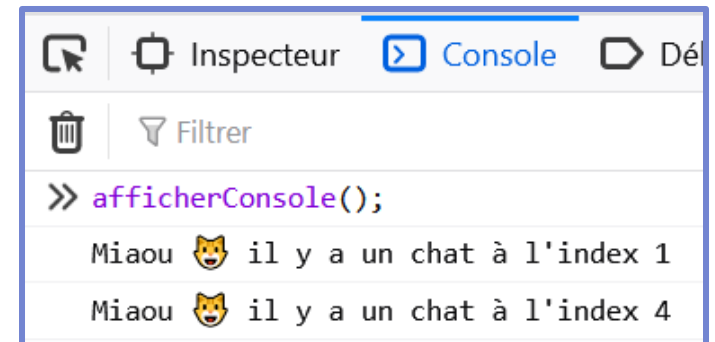
❖ Exemple 3

- ◆ Chercher le mot "**chat**" dans un tableau et mettre un message s'il y en a un.

```
let animaux = ["chien", "chat", "grenouille", "poisson", "chat"];

let i = 0;

while(i < animaux.length){
  if(animaux[i] == "chat"){
    console.log(`Miaou  il y a un chat à l'index ${i}`);
  }
  i += 1;
}
```





❖ Exemple 4

◆ Augmenter de 5 toutes les valeurs dans le tableau

- Comme *i* vaudra 0, puis 1, puis 2, puis 3, c'est comme si on faisait :

nombres[0] += 5, puis
nombres[1] += 5, puis
nombres[2] += 5, puis
nombres[3] += 5.

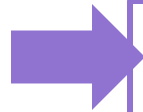
```
let nombres = [1, 2, 3, 4];  
  
let i = 0;  
while(i < nombres.length){  
    nombres[i] += 5;  
    i += 1;  
}
```

Résultat :

```
>> nombres  
← ► Array(4) [ 6, 7, 8, 9 ]
```



❖ Stocker un élément HTML dans une variable



```
let elementMario = document.querySelector(".mario");  
  
elementMario.style.borderColor = "red";  
elementMario.textContent = "Mario brosse 🍷";  
elementMario.title = "Plombier italien";  
elementMario.style.color = "crimson";  
elementMario.classList.add("peach");
```



- ❖ Tableaux avec plusieurs éléments qui ont la même classe
 - ◆ Disons qu'on souhaite modifier plusieurs éléments avec la **même classe** :

```
<p class="texte">Allo</p>  
<p class="texte">Salut</p>  
<p class="texte">Bonjour</p>
```

- ◆ **Solution** : Ranger tous les éléments avec la **même classe** dans un tableau à l'aide de **document.querySelectorAll** :

```
let elements = document.querySelectorAll(".texte");
```

```
<p class="texte">Allo</p>  
<p class="texte">Salut</p>  
<p class="texte">Bonjour</p>
```

```
let elements = [ , , ];
```




❖ Tableaux avec plusieurs éléments qui ont la même classe

- ◆ Une fois qu'on a notre **tableau d'éléments**, on peut les modifier en utilisant les **index []** :

```
<div class="texte">fr fr no cap</div>  
<div class="texte">sick imho</div>  
<div class="texte">Poggers</div>
```

```
let elements = document.querySelectorAll(".texte");  
  
elements[0].style.color = "blue";  
elements[1].style.color = "red";  
elements[2].style.color = "goldenrod";
```

fr fr no cap
sick imho
Poggers



fr fr no cap
sick imho
Poggers



❖ Tableaux avec plusieurs éléments qui ont la même classe

- ◆ On peut aussi procéder avec une boucle qui modifie les éléments de l'**index 0** jusqu'à **tableau.length** :

```
<div class="texte">fr fr no cap</div>
<div class="texte">sick imho</div>
<div class="texte">Poggers</div>
```

```
let elements = document.querySelectorAll(".texte");
let i = 0;

while(i < elements.length){
  elements[i].style.color = "limegreen";
  elements[i].textContent += ` index ${i}`;
  i += 1;
}
```

fr fr no cap
sick imho
Poggers



fr fr no cap index 0
sick imho index 1
Poggers index 2



❖ `querySelector()` vs `querySelectorAll()`

- ◆ `querySelector()` permet d'obtenir un seul élément.
 - On s'en sert pour obtenir le premier élément avec la **classe** demandée.

```
let element = document.querySelector(".mario");
element.textContent = "It's a me !";
element.style.color = "crimson";
```

- ◆ `querySelectorAll()` permet d'obtenir un tableau de plusieurs éléments.
 - On s'en sert pour obtenir plusieurs éléments avec une même **classe**.

```
let elements = document.querySelectorAll(".petiteImage");
let i = 0;

while(i < elements.length){
    elements[i].classList.add("cadreBleu");
    elements[i].src = "images/balai.png";
    i += 1;
}
```