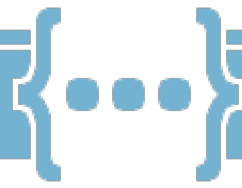


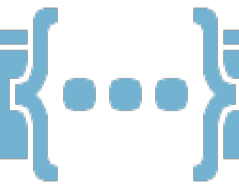
# Cours 8

## Tableaux

Intro. à la programmation



- ❖ Révision
- ❖ Tableaux
- ❖ Boucles et tableau



## ❖ DOM

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

### ◆ Classes

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

### ◆ Attributs

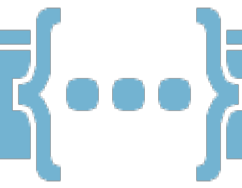
- Ajouter / Modifier : `document.querySelector("#id").setAttribute("nomAttribut", "valeur")`
- Retirer : `document.querySelector("#id").removeAttribute("nomAttribut");`
- Obtenir : `document.querySelector("#id").getAttribute("nomAttribut")`
  - Nous donne la valeur associée à cet attribut.

```
<p id="banniere" title="Bannière">Deux attributs</p>
```



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

```
function modifierMario(){  
    let elementMario = document.querySelector("#mario");  
  
    elementMario.textContent = "Mario brosse 🍷";  
    elementMario.style.color = "red";  
    elementMario.style.borderWidth = "5px";  
    elementMario.classList.add("goomba");  
    elementMario.classList.toggle("mushroom");  
    elementMario.setAttribute("title", "Plombier");  
}
```



## ❖ Boucles

### ◆ Syntaxe :

```
for(initialisation; condition d'exécution; incrémentation) {  
    // Code à répéter  
}
```

Initialisation	Condition d'exécution	Incrémentation
for(let index = 1;	index < 3;	index += 1){
	// Code à répéter	
}		



## ❖ 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
<b>0</b>	"Bleu"
<b>1</b>	"Rouge"
<b>2</b>	"Jaune"
<b>3</b>	"Vert"
<b>4</b>	"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 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]



## ❖ splice()

- ◆ Permet entre autre\* 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.

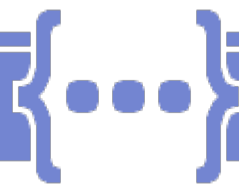


## ❖ splice()

### ◆ 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];
```

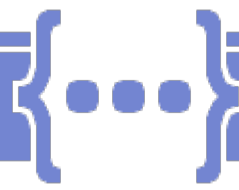
Imaginez s'il y avait eu **50** prix à additionner !

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

```
let prix = [5.49, 1.99, 99.99, 8.49, 7.72];  
  
let totalPrix = 0;  
  
for(let p of prix){  
    totalPrix += p;  
}
```

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.

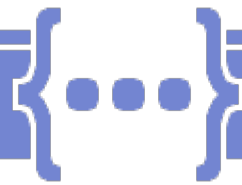
### ◆ Une **boucle** qui sert à parcourir un **tableau** ressemblera toujours à ceci

- Ça ressemble à une boucle **for**, mais c'est un autre format adapté aux tableaux.

**donnee** représente chacune des données du tableau

**monTableau** est une variable qui contient un tableau

```
for(let donnee of monTableau){  
  // Du code ...  
}
```



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

```
for(let donnee of monTableau){  
    // Du code ...  
}
```

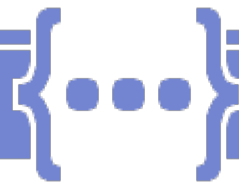
◆ **monTableau** est le nom de la variable qui contient un tableau.

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

◆ **donnee** est une variable locale spéciale qui représente chaque donnée du tableau, une à la fois, à travers les itérations.

- ex.

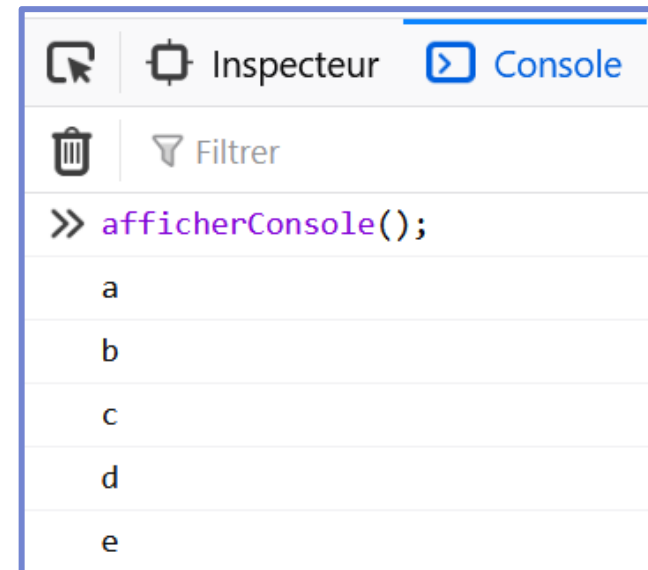
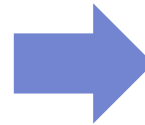
- Première itération : **donnee** vaut "a".
- Deuxième itération : **donnee** vaut "b".
- Troisième itération : **donnee** vaut "c".

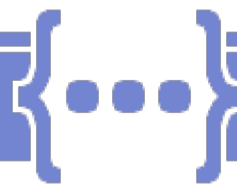


## ❖ Exemple 1

- ◆ Afficher toutes les données d'un tableau dans la console
  - À chaque itération, on affiche la valeur de **donnee** dans la **console**.

```
function afficherConsole(){  
  
    let lettres = ["a", "b", "c", "d", "e"];  
  
    for(let donnee of lettres){  
        console.log(donnee);  
    }  
}
```



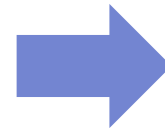


## ❖ Exemple 2

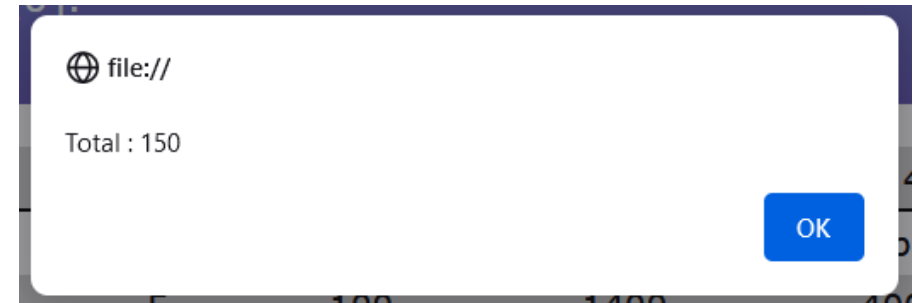
### ◆ Somme des éléments d'un tableau

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

```
let nombres = [10, 20, 30, 40, 50];  
let total = 0;  
  
for(let n of nombres){  
    total += n;  
}  
  
alert("Total : " + total);
```



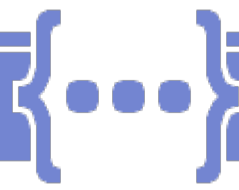
Alerte effectuée :



- La boucle, avec 5 itérations, fait le calcul suivant : 0 + 10 + 20 + 30 + 40 + 50

Valeur initiale de total





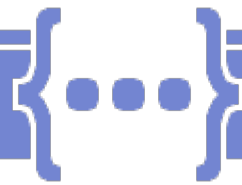
## ❖ Exemple 3

- ◆ Chercher le mot **"Chat"** dans un tableau et lancer une **alerte** s'il y en a un.

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

for(let a of animaux){
  if(a == "chat"){
    alert("Miaou ");
  }
}
```



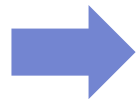


## ❖ Tableaux et #ids

### ◆ Modifier plusieurs éléments avec des ids différents

- Il faut commencer par créer un **tableau** qui contient les **#ids** des éléments pour lesquels nous souhaitons faire une opération commune :

```
<h2 id="titre">Semaine 8</h2>
<p id="texte1">Allo</p>
<p id="texte2">Ça va ?</p>
```

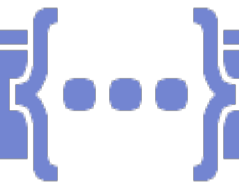


```
let ids = ["#titre", "#texte1", "#texte2"];
```

- Ensuite, on fait une **boucle** qui parcourt le **tableau** :

```
let ids = ["#titre", "#texte1", "#texte2"];

for(let i of ids){
  document.querySelector(i).classList.add("sobre");
}
```



### ❖ Tableaux avec plusieurs éléments qui ont la même classe

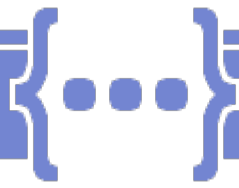
- ◆ Disons qu'on souhaite modifier plusieurs éléments avec la **même classe**, mais qui n'ont pas d'id :

```
<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");
```

- Attention : on met un **.** plutôt qu'un **#** car c'est une **classe** et non un id.



## ❖ Tableaux avec plusieurs éléments qui ont la même classe

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


- ◆ Une fois qu'on a notre **tableau d'éléments**, on peut les modifier à l'aide d'une boucle qui parcourt le tableau :

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

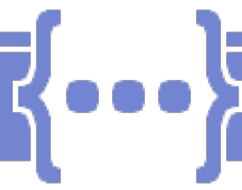
for(let e of elements){
  e.style.color = "red";
}
```

- ◆ Attention : on met juste **e** (pas `document.querySelector(e)`)
  - Notre tableau contient des éléments, pas des ids. C'est un peu l'équivalent de ceci :

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







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

### ◆ `querySelector()` permet d'obtenir un seul élément.

- On s'en sert pour obtenir un élément avec un `id` précis.

```
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");

for(let e of elements){
  ...e.classList.add("imageCommentaire");
  ...e.style.borderColor = "black";
}
```