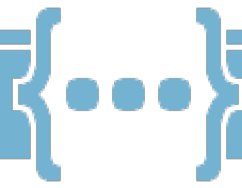


# Cours 7

## Tableaux et constantes

Intro. à la programmation - Aut. 2021



- ❖ Révision
- ❖ Tableaux
- ❖ Boucles et tableaux
- ❖ Constantes



## ❖ DOM

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

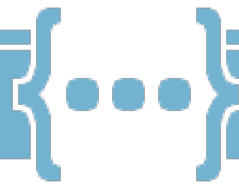
### ◆ Classes

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

### ◆ Attributs

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

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



## ❖ Boucles

### ◆ Syntaxe :

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

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



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

```
function modifierRenard(){  
    → let idRenard = document.getElementById("renard");  
  
    idRenard.textContent = "Blablala";  
    idRenard.style.color = "orange";  
    idRenard.style.borderWidth = "5px";  
    idRenard.classList.add("animal");  
    idRenard.setAttribute("style", "fontSize:13px");  
}
```



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

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 : "Benard")

```
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
```

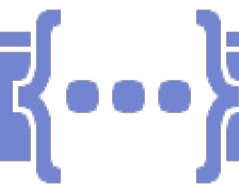
```
let message = "Le tableau ages contient " + ages.length + " éléments.";  
// message contient "Le tableau ages contient 7 éléments."
```



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

```
let prix = [4.5, 3.99, 7.2, 8.4];  
let qte = [2, 3, 4, 3];  
  
let valeurTotaleArticle1 = prix[0] * qte[0]; // Vaut 4.5 * 2 (Donc 9)
```

```
let couleurs = ["Bleu", "Rouge", "Violet", "Rose"];  
let idMessage = document.getElementById(elementId: "message");  
  
idMessage.textContent = "Mes couleurs préférées sont " + couleurs[0] + " et " + couleurs[2];  
  
// "Mes couleurs préférées sont Bleu et Violet"
```



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



◆ Pourquoi ? 🤔

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

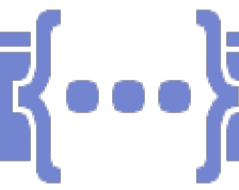
```
let prix = [5.5, 1, 3, 4.5, 7, 1, 3, 8, 2, 5];  
  
let prixTotal = prix[0] + prix[1] + prix[2] + prix[3] + prix[4] +  
                prix[5] + prix[6] + prix[7] + prix[8] + prix[9];
```

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

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

```
let prix = [5.5, 1, 3, 4.5, 7, 1, 3, 8, 2, 5];  
let prixTotal = 0;  
  
for(let index = 0; index < prix.length; index++){  
    prixTotal += prix[index];  
}
```

Sans 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.

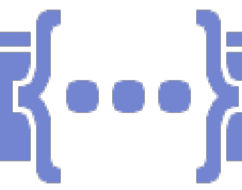
### ◆ Une **boucle** qui sert à parcourir un **tableau** ressemblera toujours à ceci

L'**index** commence à **0**,  
comme l'index d'un **tableau**

La **boucle** s'arrête quand l'**index**  
atteint la **taille du tableau**

L'**index** augmente de **1** par itération  
pour parcourir **tous** les éléments

```
for(let index = 0; index < monTableau.length; index++){  
    // Instructions ...  
}
```



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

```
for(let index = 0; index < monTableau.length; index++){  
    // Instructions avec monTableau[index]  
}
```

```
let monTableau = [1, 3, 3, 1, 2];  
                  ↑           ↑  
            monTableau[0]   monTableau[4]  
monTableau.length vaut 5
```

L'**index** commence à **0**, comme l'index d'un **tableau**

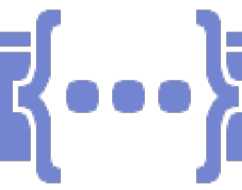
Si l'**index** commençait à **1**, on sauterait le **premier élément** du tableau qui se situe à l'**index 0**.

La **boucle** s'arrête quand l'**index** atteint **monTableau.length**

Si l'**index** atteint la valeur **5**, qui est la **taille du tableau**, on tenterait d'accéder à l'élément **monTableau[5]** ... qui n'existe pas !

L'**index** augmente de **1** par itération pour parcourir **tous** les éléments

Très important pour consulter les éléments **monTableau[0]**, puis **monTableau[1]**, puis **monTableau[2]**, etc...



## ❖ Exemple 1

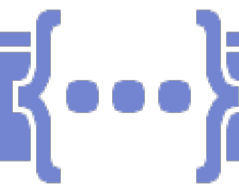
### ◆ Afficher tous les éléments

- On **concatène** tous les noms des étudiants dans le **contenu textuel** de l'élément **#liste**

```
let etudiants = ["Camille", "Pascal", "Akim", "Josée",  
                "Timothé", "Becky", "Ronald", "Prim"];  
let idListe = document.getElementById( elementId: "liste");  
  
for(let index = 0; index < etudiants.length; index++){  
    idListe.textContent += etudiants[index] + " ";  
}
```

```
// Camille Pascal Akim Josée Timothé Becky Ronald Prim
```





## ❖ Exemple 2

### ◆ Somme des éléments d'un tableau

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

```
let tabPrix = [4.5, 2, 3, 1.5, 16, 7.5, 9, 10];  
let prixTotal = 0;  
  
for(let index = 0; index < tabPrix.length; index++){  
    prixTotal = prixTotal + tabPrix[index];  
}
```

Notons que **tabPrix.length** vaut **8**

Itération #1 (**index** vaut **0**)

On ajoute **tabPrix[0]** (donc **4.5**) au **prixTotal**.  
(0 + 4.5)

Itération #2 (**index** vaut **1**)

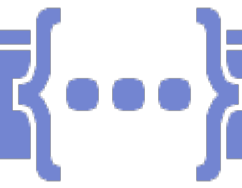
On ajoute **tabPrix[1]** (donc **2**) au **prixTotal**.  
(0 + 4.5 + 2)

Itération #8 (**index** vaut **7**)

On ajoute **tabPrix[7]** (donc **10**) au **prixTotal**.  
(0 + 4.5 + 2 + 3 + 1.5 + 16 + 7.5 + 9 + 10)

La valeur finale est donc **53.5**





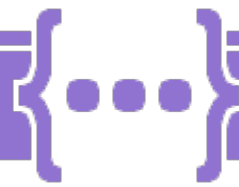
## ❖ Exemple 3

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

```
let animaux = ["Chien", "Oiseau", "Grenouille", "Chat"];

for(let index = 0; index < animaux.length; index++){
    if(animaux[index] == "Chat") {
        alert("Il y a un chat à l'index " + index + " !");
    }
}
```

```
// "Il y a un chat à l'index 3 !"
```



## ❖ Constantes

- ◆ Nous avons vu comment déclarer des **variables** ...

```
let nom = "Simone";  
let lettres = ["a", "b", "c", "d"];
```

- ◆ Nous pouvons également déclarer des **variables constantes**

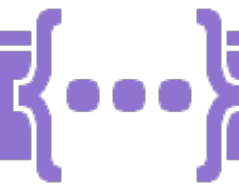
- Il suffit de remplacer le mot-clé **let** par « **const** »

```
const PI = 3.1415;  
const USD_EN_CAD = 0.79;
```

**Convention** : Les variables constantes sont nommées en **MAJUSCULES** pour les reconnaître facilement !

- Ces **variables constantes** peuvent être utilisées comme n'importe quelle autre

```
let circonference = PI * 1.65;
```



## ❖ Constantes

- ◆ **Particularité** : Leur valeur **ne peut pas** être **modifiée**. (Cela provoque une erreur dans le programme)

```
const PI = 3.1415;
```



```
PI = 3.14;
```



```
PI = PI + 1;
```

```
let PI = 3.1415;
```



```
PI = 3.14;
```



```
PI = PI + 1;
```

## ◆ Pourquoi les utiliser ?

- Lorsqu'on sait qu'une **variable** n'est jamais censée changer de valeur, il est préférable d'en faire une **constante**. Cela évite des **comportements inattendus** dans le code, comme changer la valeur d'une variable par erreur alors qu'il ne faut pas.



## ❖ Constantes

### ◆ Globales ou locales

- Tout comme les variables déclarées avec « **let** », une variable **constante** peut être **globale** ou **locale**, selon l'endroit où on l'a déclarée.

```
const gUSD_EN_CAD = 0.79;  
  
function aireCercle(){  
    const gPI = 3.1415;  
    // ...  
}
```

- **gUSD\_EN\_CAD** : Déclarée hors fonction, donc **globale**. Peut être utilisée n'importe où dans le code.
- **gPI** : Déclarée dans une fonction, donc **locale**. Peut seulement être utilisée dans la fonction où elle a été déclarée.
  - À éviter : en général on veut que nos constantes soient globales.