Bachelor CDWFS
Applications web dynamiques et interactives
2025-2026

Javascript

Les tableaux

Les boucles

Imaginons que nous voulions un programme qui écrive dans la console tous les nombre de 1 à 5. On pourrait, bien sûr, l'écrire ainsi :

```
console.log(1);
console.log(2);
console.log(3);
console.log(4);
console.log(5);
```

Mais, si ce n'est pas 5 chiffres que l'on doit écrire dans la console mais 10 ou 100 ou même 1000, ce type d'écriture ne serait vraiment pas pertinent. On pourrait alors utiliser des boucles.
Il y a deux types de boucles, les boucles while et les boucles for

Boucle while

Exemple
En utilisant une boucle while, le programme ci-dessus s'écrirait :

```
let nombre = 1;
while (nombre <= 5){
      console.log(nombre);
      nombre++;
}
```

La syntaxe de l'instruction while est la suivante :

```
while (condition){
      // instructions exécutées tant que la condition est vérifiée
}
```

Avant chaque tour de boucle, la condition associée au while est évaluée.
Si elle est vraie, les instructions du bloc de code associé au while sont exécutées. Ensuite, l'exécution revient au niveau du while et la condition est à nouveau vérifiée.

Si elle est fausse, les instructions du bloc ne sont pas exécutées et le programme continue juste après le bloc while.

Exercice : contrôle d'accès

Reprendre l'algorithme qui contrôle l'accès des utilisateurs :
L'utilisateur doit saisir un login et un mot de passe. Si l'utilisateur tape le login « campus » et le mot de passe « web », le programme affiche « Accès autorisé » ; sinon, le programme affiche « Accès refusé».

1. **Modifier l'algorithme pour qu'on redemande un login et un mot de passe tant que l'utilisateur ne saisit pas le bon login et le bon mot de passe.**

```
let login = prompt(« Login ? »)
let pwd = prompt(« Password?)
while( login != "digital" || pwd != "web"){
      console.log(«  Accès refusé»)
      login = prompt(« Login ? »)
      pwd = prompt(« Password?)
}
console.log(«  Accès autorisé»)
```

2. **Modifier l'algorithme pour que l'utilisateur ne puisse pas faire plus de 3 tentatives.**

```
let login = prompt(« Login ? »)
let pwd = prompt(« Password?)
let attempts = 0
while( (login != "digital" || pwd != "web") && attemts < 3){

      login = prompt(« Login ? »)
      pwd = prompt(« Password?)
      attempts++
}
if(attempts == 3){
      console.log(«  Accès refusé»)
}else{
      console.log(«  Accès autorisé»)
}
```

Boucle for

On a fréquemment besoin d'écrire des boucles dont la condition est basée sur la valeur d'une variable qui est modifiée dans le corps de la boucle. Pour répondre à ce besoin, JavaScript dispose de la boucle « for ».

Voici notre programme d'exemple réécrit avec une boucle for .

```
let compteur;
for (compteur = 1; compteur <= 5; compteur++){
      console.log(compteur);
}
```

La syntaxe de l'instruction for est la suivante :

```
for (initialisation; condition; étape){
      // instructions exécutées tant que la condition est vérifiée
}
```

Son fonctionnement est un peu plus complexe que celui d'un while :

- L' initialisation se produit une seule fois, au début de l'exécution.
- La condition est évaluée avant chaque tour de boucle. Si elle est vraie, un nouveau tour de
- boucle est effectué. Sinon, la boucle est terminée.
- L' étape est réalisée après chaque tour de boucle.

Execution pas à pas

Utiliser les outils de développement du navigateur pour contrôler l'éxecution d'une boucle. Se rendre dans l'onglet Source/Debugger :

- Créer une boucle while
- Placer un point d'arrêt dans le navigateur / Enlever un point d'arrêt
- Lancer l'execution pas à pas. Voir les variables.
- Bouton play pour avancer jusqu'au prochain point d'arrêt
- Expression espionne : "watch"

Debugger

Placer un point d'arrêt dans le code Js avec le mot-clef "debugger".

Cela provoque

- l'arrêt du programme
- l'ouverture de l'onglet **Onglet Source/Debugger** dans les outils de développement du navigateur.

Les valeur des variables utilisées par le programme sont clairement visibles.

Utiliser le bouton « Play » pour relancer le programme.

Les tableaux

Imaginez que vous souhaitiez informatiser la liste de tous les films que vous avez vus cette année.
Une première solution serait de créer une variable par film, comme dans l'exemple suivant.

```
let film1 = "Le loup de Wall Street";
let film2 = "Vice-Versa";
let film3 = "Babysitting";
// ...
```

Si vous êtes cinéphile, vous risquez rapidement de vous retrouver avec un grand nombre de variables dans votre programme. De plus, toutes ces variables sont entièrement indépendantes. Il n'existe aucun moyen pour, par exemple, afficher la liste complète des films ou rechercher un titre dans la liste.Il faudrait trouver une solution pour mémoriser ensemble plusieurs éléments de manière fiable et **facilement accessible.**

Cette solution existe : ce sont les tableaux.

Un tableau est un type de donnée qui permet de stocker un ensemble d'éléments.
Documentation
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array

Créer un tableau

On déclare un tableau à l'aide d'une paire de crochets [ ]
Voici comment créer notre liste de films sous la forme d'un tableau :

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
```

Tout ce qui se trouve entre les crochets correspond au contenu du tableau. Les différents éléments stockés sont séparés par des virgules.
Puisqu'un tableau est destiné à contenir plusieurs éléments, une bonne pratique consiste à donner aux variables tableaux des noms exprimant le pluriel, comme par exemple films, tabFilms ou encore lesFilms.

Obtenir la taille d'un tableau

Le nombre d'éléments d'un tableau s'obtient en lui appliquant la propriété .**length**

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
console.log(films.length); // Affiche 3
```

Bien entendu, cette propriété renvoie 0 dans le cas d'un tableau vide (sans aucun élément).

```
let tableauVide = []; // Création d'un tableau vide
console.log(tableauVide.length); // Affiche 0
```

Accéder à un élément d'un tableau

Chaque élément présent dans un tableau est identifié par un numéro, appelé son indice (index en anglais). On peut représenter graphiquement un tableau comme un ensemble de cases, chacune stockant une valeur spécifique et étant associée à un indice. Voici comment on pourrait représenter le tableau films déclaré ci-dessus :

Indice	0	1	2
Valeur	Le loup de wall street	Vice-Versa	Baby-sitting

L'accès à un élément s'effectue en plaçant cet indice entre crochets, comme dans l'exemple ci-dessous :

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
console.log(films[0]); // Affiche "Le loup de Wall Street"
console.log(films[1]); // Affiche "Vice-Versa"
console.log(films[2]); / Affiche "Babysitting"
```

Remarques :

- L'indice du premier élément d'un tableau est 0 et non 1 comme on aurait pu s'y attendre.
- Le plus grand indice utilisable est égal à la taille du tableau - 1.
- Utiliser un indice invalide pour accéder à un élément d'un tableau JavaScript renvoie la valeur undefined

```
console.log(films[3]); // Affiche undefined
```

Méthode .at()

```
console.log(films.at(0); // Affiche "Le loup de Wall Street"
```

Parcourir un tableau

L'es exemples ci-dessous permet d'afficher la liste des films présents dans le tableau.

Avec une boucle while

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
let i = 0 ;
while (i < films.length){
      console.log(films[i]);
      i = i + 1 ;
}
```

Avec une boucle for

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
for (let i = 0; i < films.length; i++){
      console.log(films[i]);
}
```

Avec la boucle for, on fait varier l'indice du tableau de 0 (indice du premier élément) à taille du tableau - 1 (indice du dernier) pour accéder aux éléments les uns après les autres.

Avec une boucle for ... of

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
for (let film of films){
    console.log(film);
}
```

★ **Avec la fonction forEach**

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
let i = 0 ;
films.forEach(function(item){
  console.log(item)
})
```

Cette syntaxe utilise les fonctions de rappel (callback function).

Ajouter un élément dans un tableau

Vous voulez ajouter le film « Le labyrinthe » à la liste. Vous pouvez le faire avec deux possibilités :

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
// Méthode n°1
films[3] = "Le labyrinthe" ;
// Plus généralement :
films[films.length] = "Dune" ;
// Méthode n°2
films.push("Le labyrinthe");
console.log(films[3]); // Affiche "Le labyrinthe"
```

Autres méthodes :
https://howtocreateapps.com/ways-to-insert-elements-to-an-array-in-javascript/

Autres opérations sur les tableaux

Tester la présence d'un élément

Includes()

Cette méthode retourne true ou false

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
let found = films.includes("Vice-Versa") ;
```

Récupérer la position d'un élément dans un tableau

indexOf(element)

Cette méthode retourne la position d'un élément dans le tableau

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
let position = films.indexOf("Vice-Versa") ;
```

Pour des cas plus complexe, utiliser **findIndex()**

```
const index = items.findIndex(item => item.id === id);
```

Supprimer un élément dans un tableau

Méthode 1 : avec filter

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
films = films.filter(name => name !== "Vice-Versa");
```

Méthode 2 : d'abord identifier la position de l'élément, puis utiliser la méthode **splice** :

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
let position = films.indexOf("Vice-Versa") ;
films.splice(position, 1);
```

Supprimer plusieurs éléments

```
const myfruits = ["Banana", "Orange", "A", "B", "C",  "Apple", "Mango"];
let letters =  myfruits.splice(2, 3);
console.log(myfruits)
```

Premier paramètre : l'index du premier élément à supprimer. Second paramètre : le nombre d'éléments à supprimer. Valeur de retour : les éléments supprimés.
La méthode splice permet également d'insérer des éléments.

★ Filtrer un tableau

Exemple : pour trouver tous les films qui contiennent le mot « loup » :

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
const selection = films.filter((film) => film.includes("loup"));
```

La fonction **filter** utilise une fonction de rappel (callback function). Elle retourne un tableau contenant les éléments filtrés.
Filter est fréquement utilisée pour supprimer des éléments (cf ci-dessus)

Trier un tableau

Par ordre alphabétique croissant

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
let sortedFilms = films.sort() ;
```

★ Par ordre numérique croissant

```
let nombres = [4, 2, 5, 1, 3];
nombres.sort((a, b) => a - b);
```

La fonction **sort** utilise une fonction de rappel (callback function).

Convertir un tableau en chaîne et inversement

join()

```
let films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
let liste = films.join(",") ;
```

Cette méthode transforme un tableau en chaîne de caractère.
Inversement, la méthode split(), appliquée à une chaîne, retourne un tableau

```
let liste = "Le loup de Wall Street, Vice-Versa, Babysitting"];
let films = list.split(",") ;
```

Reduire à une seule valeur avec reduce

```
const values = [45, 4, 9, 16, 25];
let sum = values.reduce((total, value) => {
    return total + value;
} )
console.log(sum)
```

★ Fusion de 2 tableaux

```
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];
// concat
const myChildren = myGirls.concat(myBoys);
// spread opérator
const children = [...myGirls, ...myBoys]
```

★ Extraire

```
const myfruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = myfruits.slice(1, 3);
```

La méthode slice comporte 2 paramètres : l'index du début et l'index de fin (exclu). Elle ne modifie pas le tableau original

★ Créer un tableau à partir d'un autre tableau

```
const numbers1 = [45, 4, 9, 16, 25];
const numbers2 = numbers1.map((value, index) => {
    return value * 2;
})
console.log(numbers2)
```

Aller plus loin

Map

Une structure similaire aux tableaux qui permet de stocker des valeurs uniques

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Map