Objectif du module

À la fin de ce module, l'étudiant sera capable de :

- Comprendre à quoi sert une boucle dans une interface
- Utiliser un tableau pour stocker une liste (catégories, produits, messages, images…)
- Générer des éléments HTML à partir d'un tableau
- Afficher dynamiquement une liste d'éléments sur une page
- Filtrer ou ajouter des éléments à une liste simple

Table des matières

1. Pourquoi les boucles et tableaux sont utiles en UI ?

Sans boucle :

- tu écris 10 fois le même HTML (copier-coller)
- tu dois modifier 10 endroits si ça change

Avec boucle + tableau :

- tu stockes les données une fois
- tu génères l'interface automatiquement

Exemples UI typiques :

- liste de produits
- menu de navigation
- tags/badges
- galerie d'images
- liste de messages / commentaires

2. Les tableaux : une "liste" de valeurs

Déclarer un tableau

```
const categories = ["Accueil", "Produits", "Contact"];
```

Accéder à un élément

```
console.log(categories[0]); // "Accueil"
```

Ajouter un élément

```
categories.push("À propos");
```

Un tableau = une liste

L'index commence à 0

3. Les boucles : répéter une action

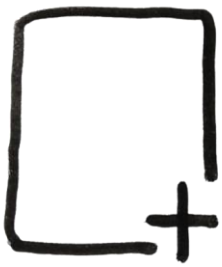
Boucle for

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

length = taille du tableau

on parcourt tous les éléments

Comment manipuler le DOM en 3 étapes :

 Créer	 Remplir	 attacher

Exemple UI : générer une liste `<ul>`

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Panier</title>
  <!-- Inclure Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.css">
</head>
<body class="p-4">
    <h2>Catégories</h2>
    <ul id="liste"></ul>


<script>
const categories = ["Maison", "Mode", "Sport", "Tech"];
const liste = document.querySelector("#liste");

for (let i = 0; i < categories.length; i++) {
  const li = document.createElement("li");
  li.textContent = categories[i];
  liste.appendChild(li);
}
</script>
</body>
</html>
```

La méthode appendChild() attache un nœud (élément) à la fin d'une liste d'enfants d'un nœud parent spécifié.

Exemple UI : badges Bootstrap

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Panier</title>
```

```
  <!-- Inclure Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.css">
</head>
<body class="p-4">
    <div id="tags"></div>
<script>
const tags = ["Promo", "Nouveau", "Best-seller"];
const container = document.querySelector("#tags");

for (let i = 0; i < tags.length; i++) {
  const span = document.createElement("span");
  span.textContent = tags[i];
  span.classList.add("badge", "bg-primary", "me-2");
  container.appendChild(span);
}

</script>
</body>
</html>
```

Exemple : filtrer une liste

Afficher seulement les éléments qui contiennent une lettre.

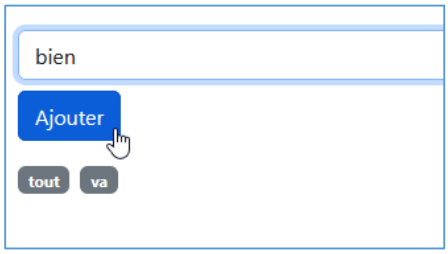
```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>ajouttag</title>
  <!-- Inclure Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.css">
</head>
<body class="p-4">
<script>

const categories = ["Maison", "Mode", "Sport", "Tech", "Jeux"];
const filtre = "o"; // on veut afficher celles qui contiennent "o"

for (let i = 0; i < categories.length; i++) {
  if (categories[i].toLowerCase().includes(filtre)) {
    console.log(categories[i]);
  }
}
</script>
</body>
</html>
```

Exercice – flitreliste.html

Modifier l'exemple pour afficher les résultats dans la page au lieu de la console.

Exercice – ajouttag.html

Prévoir un formulaire où l'utilisateur ajoute un tag, et il apparaît directement :



Créer une interface contenant :

1. Un champ texte
2. Un bouton "Ajouter"
3. Une zone d'affichage des tags

Contraintes :

- Les tags doivent être stockés dans un tableau JavaScript.
- Lorsqu'on clique sur "Ajouter" :
    - Le tag est ajouté au tableau.
    - La liste affichée est reconstruite dynamiquement.
- Si le champ est vide → ne rien ajouter.

Objectif :

Comprendre comment :

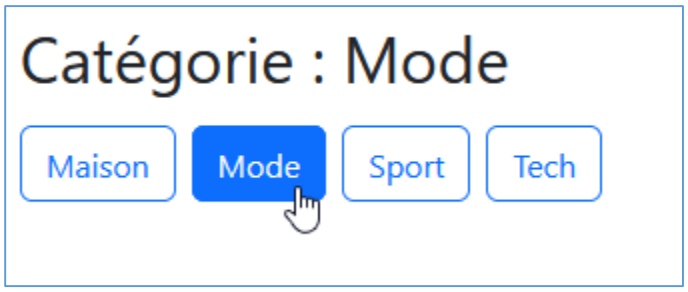
- utiliser `push()`
- parcourir un tableau avec une boucle
- générer dynamiquement des éléments HTML

Exercice : galerie de boutons – galerieboutons.html

Créer une liste de catégories à partir d'un un tableau

> générer automatiquement des boutons

Quand on clique un bouton, le titre de la page change



Créer une page contenant :

1. Un titre :

   "Choisissez une catégorie"
2. Une zone vide destinée à recevoir les boutons

Étapes :

1. Créer un tableau JavaScript contenant au moins 4 catégories.
2. Générer automatiquement un bouton pour chaque catégorie.
3. Lorsque l'on clique sur un bouton :

- Le titre de la page devient :

  "Catégorie : [nom]"

Bonus (facultatif) :

- Mettre en évidence le bouton actif (changer sa classe Bootstrap).

Objectif :

- Utiliser une boucle `for`
- Utiliser `createElement()`
- Utiliser `appendChild()`
- Ajouter un événement `click`

Exercice — Listevilles.html — Pays → villes (liste dynamique)

Consigne :

Créer une page contenant :

- Une liste déroulante `Pays` (`<select id="pays">`)
- Une liste déroulante `Villes` (`<select id="villes">`)

Données :

Créer **2 tableaux** (version simple) :

```
const pays = ["Belgique", "France"];
const villesBelgique = ["Wavre", "Bruxelles","Ottignies","Mons","Anvers" ];
const villesFrance = ["Paris", "Lille","Marseille","Lyon"];
```

- Quand l'utilisateur change de pays :
  1. La liste des villes est vidée
  2. On ajoute les villes correspondantes dans le `<select>` "villes"

Contraintes :

- Utiliser un événement `change`
- Utiliser une boucle `for` pour remplir les options
- Utiliser `createElement("option")` + `appendChild`
- Utiliser `innerHTML = ""` uniquement pour vider la liste des villes

Livrable attendu :

- Les villes changent automatiquement quand on change de pays.