# Les tableaux

```
var tableau = ["Bob", "Julien", "Roger"];
alert(tableau); // affiche « Bob,Julien,Roger »

// Parcourir un tableau à la main

alert(tableau[0]); // affiche « Bob »
alert(tableau[1]); // affiche « Julien »
alert(tableau[2]); // affiche « Roger »

// Parcourir un tableau automatiquement
var i ;
for (i = 0; i < tableau.length; i++) {
    alert(tableau[i]);
}

// Supprimer le dernier élément

alert(tableau.pop()); // affiche « Roger »
alert(tableau); // affiche « Bob, Julien »

// Ajouter un élément à la fin

tableau.push("Toto");
alert(tableau); // affiche « Bob, Julien, Toto »
```

# Exercice d'application

## Étape 1 : afficher un tableau sous forme de liste

Construire une liste à l'aide d'un tableau. Dans le HTML, j'ai un `<ul id="liste"></ul>`, dans le JavaScript, j'ai un tableau ; le but est que tous les éléments du tableau se retrouve dans des `<li>` dans le `<ul>`.


## Étape 2 : mettre le code dans une fonction

Autant il est fréquent d'avoir la déclaration des variables et les onclick en dehors des fonctions, autant il est préférable de mettre le reste dans des fonctions, pour bien segmenter le code en « morceaux logiques ». Créer une fonction `afficherListe()` qui contient notre boucle. Puis l'appeler pour qu'elle s'exécute.

**Conseil :** que se passe-t-il si l'on appelle plusieurs fois la fonction ? Assurez-vous que les éléments ne se répètent pas dans la liste.

## Étape 3 : ajout d'un élément

En continuant sur le code précédent, ajouter un champ texte `<input id="texte">` et un bouton `<button id="ajouter">Ajouter</button>`. Faire en sorte que lorsque l'on clique sur le bouton, le texte saisi soit ajouté au tableau et à la liste. On doit associer une fonction à chaque `onclick`. On va donc avoir une 2e fonction, `ajouterElement()`.

**Rappel :** normalement le mot « innerHTML » ne doit figurer qu'une seule fois dans votre code.

## Étape 4 : bouton « annuler »

Créer un bouton « Annuler » qui supprime le dernier élément ajouté.