

Fondamentaux du Web

HTML, CSS, JavaScript

Lab 6 — Tableaux et objets

`forEach`, `map`, `filter`

Licence 1 DUT1-Informatique — 2025–2026

Informations pratiques

Où sommes-nous ? Au Lab 5, vous avez écrit vos premières fonctions. Mais une application manipule rarement *une* donnée : elle jongle avec des **collections** (54 pays, 10 questions, une liste d'étudiants). Aujourd'hui, on apprend à traiter des collections proprement, et on crée le fichier de données du projet.

Avant de commencer :

- Votre `script.js` du Lab 5 fonctionne (tag `v0.5` posé).
- Vous maîtrisez les fonctions et `for...of` du Lab 5.
- La console (F12) est votre terrain d'essai.

Validation de fin : le fichier `data.js` existe, contient cinq pays et une fonction « au hasard », la console affiche vos résultats sans erreur, et le tag `v0.6` est poussé avec un rapport.

Objectifs du Lab 6

À la fin, vous saurez :

1. Manipuler un tableau (`length`, `push`, `pop`, accès par index).
2. Agir sur chaque élément avec `forEach`.
3. Transformer un tableau avec `map`.
4. Filtrer un tableau avec `filter`.
5. Créer et lire des **objets** (des fiches d'identité).
6. Manipuler des **tableaux d'objets** et les afficher avec `console.table`.
7. Écrire le fichier `data.js` : cinq pays africains + une fonction « au hasard ».

Table des matières

Rappel express du Lab 5	2
1 Les tableaux : rappel et compléments	2
2 <code>forEach</code> : agir sur chaque élément	2
3 <code>map</code> : transformer chaque élément	3

4	filter : ne garder que certains éléments	3
4.1	map ou filter ? Le réflexe à retenir	3
5	Les objets : des fiches d'identité	4
6	Tableaux d'objets : le pattern le plus courant	4
7	Choisir un élément au hasard	4
8	On crée le fichier de données : data.js	5
9	Exercices d'application	5
10	Aide-mémoire du Lab 6	6
11	Sauvegarder mon travail (routine Git)	6
12	Résumé et transition	7

Rappel express du Lab 5

Vous savez écrire une fonction qui renvoie un résultat, décider avec `if`/ternaire, et parcourir un tableau avec `for...of`. On part de là.

1 Les tableaux : rappel et compléments

Un **tableau** est une liste ordonnée de valeurs. *Analogie* : une liste de courses, où chaque article a une position.

```
const villes = ["Dakar", "Thiès", "Saint-Louis", "Ziguinchor"];

console.log(villes[0]);      // "Dakar" -> on compte à partir de 0 !
console.log(villes[2]);      // "Saint-Louis"
console.log(villes.length);  // 4 (le nombre d'éléments)

villes.push("Touba");        // ajoute à la fin
console.log(villes.length);  // 5
villes.pop();                // retire le dernier
console.log(villes.length);  // 4
```

Le piège du débutant : on compte à partir de 0

Le **premier** élément est à l'index 0, pas 1. Donc dans un tableau de 4 éléments, les index vont de 0 à 3. `villes[4]` n'existe pas et renvoie `undefined`.

2 forEach : agir sur chaque élément

On a déjà fait ça avec `for...of`. `forEach` est une autre manière, très lisible, qui veut dire « pour chaque élément, fais ceci ». *Analogie* : l'enseignant fait l'appel et coche chaque nom, l'un après l'autre.

```
const villes = ["Dakar", "Thiès", "Saint-Louis"];

villes.forEach((ville) => {
  console.log(`Ville : ${ville}`);
});
```

Rappel CM3 : la flèche =>

Le `(ville) => { ... }` est une **fonction fléchée**, vue au CM3 : c'est juste une fonction écrite en plus court, qu'on donne à `forEach`. Pour aujourd'hui, recopiez ce modèle tel quel. `forEach` appelle cette fonction *une fois pour chaque élément*, en lui passant l'élément courant.

3 map : transformer chaque élément

`map` sert à **fabriquer un nouveau tableau** en transformant chaque élément de l'ancien. *Analogie* : on passe une pile de feuilles à la photocopieuse réglée « agrandir 200 % » : on obtient une *nouvelle* pile, de *même taille*, mais chaque feuille est transformée.

```
const prixFcfa = [656, 1312, 3280]; // des prix en FCFA

const prixEuro = prixFcfa.map((p) => p / 656);

console.log(prixEuro); // [1, 2, 5]
console.log(prixFcfa); // [656, 1312, 3280] -> l'original n'a pas changé !
```

Point clé : `map` renvoie un **nouveau** tableau de **même longueur** ; il ne modifie pas l'original.

4 filter : ne garder que certains éléments

`filter` sert à **garder seulement** les éléments qui passent un test. *Analogie* : au marché, on trie un cageot de mangues pour ne garder que les bonnes ; le cageot ressort plus petit.

```
const notes = [12, 7, 15, 9, 18, 10];

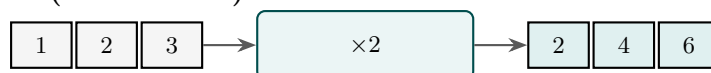
const admis = notes.filter((note) => note >= 10);

console.log(admis); // [12, 15, 18, 10] -> seulement les notes >= 10
```

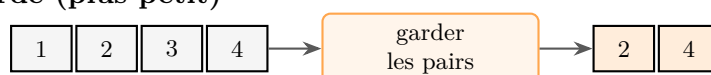
Point clé : `filter` renvoie un tableau **plus petit ou égal** : on a gardé un sous-ensemble.

4.1 map ou filter ? Le réflexe à retenir

`map` — transforme (même taille)



`filter` — garde (plus petit)



map garde le même nombre d'éléments en les transformant ; *filter* en retire certains.

5 Les objets : des fiches d'identité

Un tableau range des valeurs *par position*. Un **objet** range des informations *par étiquette* (qu'on appelle une *propriété*). *Analogie* : une carte d'identité, avec des champs nommés « nom », « âge », « ville ».

```
const etudiant = {
  nom: "Fatou",
  age: 20,
  ville: "Dakar"
};

console.log(etudiant.nom);      // "Fatou" -> accès par le point
console.log(etudiant.age);      // 20
console.log(etudiant["ville"]); // "Dakar" -> autre écriture, avec crochets
```

Les deux écritures `etudiant.nom` et `etudiant["nom"]` donnent la même chose. On préfère le **point** car c'est plus court et plus lisible.

6 Tableaux d'objets : le pattern le plus courant

La combinaison gagnante : un **tableau** dont chaque élément est un **objet**. C'est ainsi qu'on représente une liste de pays, d'étudiants, de produits...

```
const etudiants = [
  { nom: "Fatou", note: 12 },
  { nom: "Moussa", note: 8 },
  { nom: "Awa", note: 15 }
];

// Parcourir et afficher chaque étudiant
etudiants.forEach((e) => {
  console.log(`${e.nom} a eu ${e.note}/20`);
});

// Vue en tableau dans la console (très pratique pour vérifier)
console.table(etudiants);
```

Vérification

`console.table` affiche un *vrai tableau* avec colonnes et lignes dans la console. Essayez : c'est l'outil idéal pour inspecter une liste d'objets.

7 Choisir un élément au hasard

Un quiz a besoin de tirer une question au hasard. Deux outils :

- `Math.random()` : un nombre décimal au hasard entre 0 et 1.
- `Math.floor(x)` : arrondit *vers le bas* (`Math.floor(3.9)` vaut 3).

En combinant les deux, on obtient un index au hasard dans un tableau :

```
function choisirAuHasard(tableau) {
  const index = Math.floor(Math.random() * tableau.length);
  return tableau[index];
}
```

```
const villes = ["Dakar", "Thiès", "Saint-Louis", "Ziguinchor"];
console.log(choisirAuHasard(villes)); // une ville différente à chaque appel
```

Comment ça marche ? `Math.random() * tableau.length` donne un nombre entre 0 et la taille du tableau ; `Math.floor` le transforme en un index entier valide (de 0 à `taille-1`).

8 On crée le fichier de données : `data.js`

Il est temps de poser la **matière première** de Jàng Afrig. Créez un fichier `data.js` à la racine du projet, et branchez-le *avant* `script.js` dans le `<head>` :

```
<script src="data.js" defer></script>
<script src="script.js" defer></script>
```

Dans `data.js`, on met cinq pays africains (sous forme de tableau d'objets) et la fonction « au hasard » :

```
// data.js -- les données de Jàng Afrig

const pays = [
  { cca3: "SEN", nom: "Sénégal", capitale: "Dakar" },
  { cca3: "MLI", nom: "Mali", capitale: "Bamako" },
  { cca3: "GHA", nom: "Ghana", capitale: "Accra" },
  { cca3: "KEN", nom: "Kenya", capitale: "Nairobi" },
  { cca3: "MAR", nom: "Maroc", capitale: "Rabat" }
];

function choisirAuHasard(tableau) {
  const index = Math.floor(Math.random() * tableau.length);
  return tableau[index];
}
```

Vérification

Dans `script.js`, ajoutez `console.table(pays);` puis `console.log(choisirAuHasard(pays));`. La console doit montrer le tableau des cinq pays, puis un pays tiré au hasard. **Ces deux fichiers parlent déjà entre eux** : `script.js` utilise des choses définies dans `data.js`, parce que `data.js` est chargé en premier.

9 Exercices d'application

Exercice 1 — Moyenne d'une classe (`forEach`)

Créez un tableau de 10 étudiants DUT1-Info (objets `{ nom, note }`). Avec `forEach`, calculez la **moyenne de la classe**.

Indice : une variable `somme` (avec `let`), on y ajoute chaque `e.note`, puis on divise par `etudiants.length`.

Exercice 2 — Prix en euros (`map`)

À partir d'un tableau de prix en FCFA, fabriquez avec `map` le tableau des mêmes prix en euros (diviser par 656).

Exercice 3 — Les admis (filter)

À partir du tableau des 10 étudiants de l'exercice 1, fabriquez avec `filter` le tableau des **seuls étudiants admis** (note ≥ 10), puis affichez-le avec `console.table`.

Exercice 4 — Les capitales (map)

À partir du tableau `pays` de `data.js`, fabriquez avec `map` un tableau qui ne contient que les **noms des capitales**.

Indice : `pays.map((p) => p.capitale)`.

10 Aide-mémoire du Lab 6

Besoin	Code
Accéder à un élément	<code>tableau[index]</code> (index commence à 0)
Taille	<code>tableau.length</code>
Ajouter / retirer à la fin	<code>tableau.push(x)</code> / <code>tableau.pop()</code>
Agir sur chaque élément	<code>tableau.forEach((e) => { ... })</code>
Transformer (même taille)	<code>tableau.map((e) => ...)</code>
Filtrer (plus petit)	<code>tableau.filter((e) => condition)</code>
Lire une propriété d'objet	<code>objet.propriete</code>
Visualiser une liste d'objets	<code>console.table(tableau)</code>
Élément au hasard	<code>choisirAuHasard(tableau)</code> (voir la section 7)

11 Sauvegarder mon travail (routine Git)**Routine Git de fin de Lab — tag v0.6**

```
git status
git add .
git commit -m "Lab 6 : data.js et methodes de tableaux"
git tag v0.6
git push
git push --tags
```

Vérifiez sur GitHub que le commit *et* le tag v0.6 apparaissent.

Rapport à produire

Déposez `rapport-lab-6-NOM-Prenom.pdf` dans le dossier `rapports/`, avec quatre sections :

1. **Ce que j'ai fait** : `data.js`, et les exercices `forEach/map/filter`.
2. **Difficultés rencontrées** : vos erreurs (avec les messages copiés).
3. **Comment je les ai résolues**.
4. **Lien vers mon repo** avec le tag v0.6.

12 Résumé et transition

Vous savez maintenant manipuler des **collections** : parcourir (**forEach**), transformer (**map**), filtrer (**filter**), et organiser des données en **tableaux d'objets**. Surtout, vous avez créé **data.js**, le réservoir de données de Jàng Afrig.

Tout ça vit encore dans la console. Au **Lab 7**, on accomplit le grand saut : on apprend à **lire et modifier la page visible** (le DOM) et à **réagir aux clics** (les événements). Résultat : ****un premier quiz Capitales jouable****, avec un score, du début à la fin — toujours sur nos cinq pays, sans Internet pour l'instant.

Les données sont prêtes. Au Lab 7, Jàng Afrig prend enfin vie à l'écran.

Dr. El Hadji Bassirou TOURÉ — UCAD
Licence 1 DUT1-Informatique — Fondamentaux du Web — 2025-2026