Chapitre 3 - JavaScript : Des Fondamentaux aux Concepts Avancés

Public cible : Débutants en JS (mais ayant des bases en HTML/CSS).
Objectifs :

- Comprendre la syntaxe et les concepts de base.
- Manipuler le DOM pour créer des pages interactives.
- Découvrir des concepts avancés (async, fetch API, closures).
- Appliquer les connaissances via des exercices et un mini-projet.

📌 Structure Détaillée

(Avec timing, théorie, exemples et exercices)

1️⃣ Introduction (20 min)

A. Qu'est-ce que JavaScript ?

- **Rôle** : Ajouter de l'interactivité aux pages web (ex: formulaires, animations).
- **Différence HTML/CSS/JS** :
  - HTML = Structure, CSS = Style, **JS = Comportement**.

B. Intégration dans une page

- **Méthode 1** : Balise `<script>` interne.

```
<script>
  console.log("Hello World!");
</script>
```

- **Méthode 2** : Fichier externe (`script.js`).

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

(`defer` charge le script après le HTML)

Exercice Icebreaker :

- Affichez "Bonjour [prénom]" dans la console.
  Corrigé :

```
const prenom = prompt("Quel est votre prénom ?");
console.log(`Bonjour ${prenom} !`);
```

2️⃣ Les Bases

A. Variables et Types de Données

- **`let`** (modifiable), **`const`** (constant).
- **Types** : `string`, `number`, `boolean`, `array`, `object`.

```
let age = 25; // number
const pays = "France"; // string
```

Exercice :

- Créez un tableau `fruits` et ajoutez-y 3 fruits. Affichez le 2ème fruit.
  Corrigé :

```
const fruits = ["pomme", "banane", "kiwi"];
console.log(fruits[1]); // Affiche "banane" (index 1)
```

B. Opérateurs et Conditions

- **Comparaisons** : `===` (strictement égal), `!==`, `>`, `<=`.

- **Conditions** : `if/else`, `switch`.

```
if (age >= 18) {
  console.log("Majeur");
} else {
  console.log("Mineur");
}
```

Exercice :

- Vérifiez si un nombre est pair ou impair.
  Corrigé :

```
const nombre = 10;
if (nombre % 2 === 0) {
  console.log("Pair");
} else {
  console.log("Impair");
}
```

3️⃣ Fonctions et Boucles

A. Fonctions

- **Déclaration** :

```
function direBonjour(nom) {
  return `Bonjour ${nom} !`;
}
console.log(direBonjour("Alice")); // "Bonjour Alice !"
```

- **Fonction fléchée** (ES6) :

```
const addition = (a, b) => a + b;
```

Exercice :

- Écrivez une fonction qui calcule la surface d'un rectangle.
  Corrigé :

```
const surface = (longueur, largeur) => longueur * largeur;
console.log(surface(5, 3)); // 15
```

B. Boucles

- **`for`** :

```
for (let i = 0; i < 5; i++) {
  console.log(i); // 0, 1, 2, 3, 4
}
```

- **`forEach`** (pour les tableaux) :

```
fruits.forEach(fruit => console.log(fruit));
```

Exercice :

- Affichez les nombres de 1 à 10 avec `for`, puis avec `forEach` sur un tableau.
  Corrigé :

```
// Avec for
for (let i = 1; i <= 10; i++) console.log(i);

// Avec forEach
const nombres = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
nombres.forEach(n => console.log(n));
```

4️⃣ Manipulation du DOM

A. Sélection et Modification d'Éléments

- **Sélection** :

```
const titre = document.querySelector("h1"); // Premier <h1>
const boutons = document.querySelectorAll(".btn"); // Tous les éléments avec class="btn"
```

- **Modification** :

```
titre.textContent = "Nouveau titre";
boutons[0].style.backgroundColor = "red";
```

Exercice :

- Changez la couleur de fond d'un `<div>` au clic sur un bouton.
  Corrigé :

```
<button id="btn">Changer la couleur</button>
<div id="box" style="width: 100px; height: 100px;"></div>
<script>
  document.getElementById("btn").addEventListener("click", () => {
    document.getElementById("box").style.background = "blue";
  });
</script>
```

B. Événements

- **Types** : `click`, `mouseover`, `keydown`, `submit`.

```
bouton.addEventListener("click", () => {
  alert("Bouton cliqué !");
});
```

Exercice :

- Créez un compteur de clics sur un bouton.
  Corrigé :

```
let compteur = 0;
bouton.addEventListener("click", () => {
  compteur++;
  console.log(`Clics : ${compteur}`);
});
```

5️⃣ Concepts Avancés

A. Fetch API (Requêtes HTTP)

- **Récupérer des données** (ex: API JSONPlaceholder) :

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => response.json())
  .then(data => console.log(data));
```

Exercice :

- Affichez les noms des utilisateurs dans la console.
  Corrigé :

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(res => res.json())
  .then(users => users.forEach(user => console.log(user.name)));
```

B. Async/Await

- **Version moderne de Fetch** :

```
async function fetchUsers() {
  const response = await fetch("https://jsonplaceholder.typicode.com/users");
  const users = await response.json();
  console.log(users);
}
fetchUsers();
```

Exercice :

- Affichez les emails des utilisateurs avec async/await.
  Corrigé :

```
async function fetchEmails() {
  const res = await fetch("https://jsonplaceholder.typicode.com/users");
  const users = await res.json();
  users.forEach(user => console.log(user.email));
}
```

6️⃣ Projet Final

Consigne :

- Créez une **liste de tâches** (To-Do List) avec :
  - Un champ input pour ajouter une tâche.
  - Un bouton pour valider.
  - Affichage des tâches sous forme de liste.
  - Bouton de suppression pour chaque tâche.

Exemple de code de départ :

```
<input type="text" id="taskInput" placeholder="Nouvelle tâche">
<button id="addTask">Ajouter</button>
<ul id="taskList"></ul>
```

Corrigé partiel :

```
document.getElementById("addTask").addEventListener("click", () => {
  const task = document.getElementById("taskInput").value;
  const li = document.createElement("li");
  li.textContent = task;
  document.getElementById("taskList").appendChild(li);
});
```

📚 Ressources & Bonnes Pratiques

- **Outils** :
  - JSFiddle pour tester du code rapidement.
  - MDN JavaScript pour la documentation.
- **Bonnes pratiques** :
  - Utilisez `const` par défaut, `let` si nécessaire.
  - Évitez les `var` (obsolète).
  - Nommez les variables de manière descriptive (`userAge` au lieu de `x`).