

# Introduction à JavaScript Modernes

**KELLY** Abdoulaye

*Mali\_Code*

mlcode223@gmail.com

# Plan de l'Introduction à ES6+

- 1 Introduction à ES6+
- 2 Promesses et gestion des erreurs
- 3 Async/Await
- 4 Requêtes HTTP avec Fetch
- 5 Résumé du Jour 4

- ES6 (ECMAScript 2015) a introduit de nouvelles fonctionnalités pour JavaScript.
- Fonctionnalités clés :
  - Fonctions fléchées.
  - Template literals.
  - Déstructuration.
  - Classes.

# Fonctions fléchées

```
// Fonction traditionnelle  
function addition(a, b) {  
    return a + b;  
}  
  
// Fonction fléchée  
const addition = (a, b) => a + b;  
  
console.log(addition(5, 3)); // 8
```

# Template literals

```
const nom = "Alice";  
const age = 25;
```

```
// Utilisation de template literals
```

```
const message = `Bonjour, je m'appelle ${nom} et j'ai ${age} ans.`;  
console.log(message); // "Bonjour, je m'appelle Alice et j'ai 25 ans."
```

# Déstructuration

```
const personne = { nom: "Alice", age: 25 };
```

```
// Déstructuration d'un objet
```

```
const { nom, age } = personne;
```

```
console.log(nom); // "Alice"
```

```
console.log(age); // 25
```

```
// Déstructuration d'un tableau
```

```
const nombres = [1, 2, 3];
```

```
const [premier, deuxieme] = nombres;
```

```
console.log(premier); // 1
```

# Promesses en JavaScript

- Une promesse représente une valeur qui peut être disponible maintenant, plus tard ou jamais.
- États d'une promesse : 'pending', 'fulfilled', 'rejected'.
- Méthodes : `then`, `catch`, `finally`.

# Exemple de promesse

```
const maPromesse = new Promise((resolve, reject) => {
  setTimeout(() => {
    const succes = true;
    if (succes) {
      resolve("Opération réussie !");
    } else {
      reject("Erreur !");
    }
  }, 2000);
});

maPromesse
  .then(resultat => console.log(resultat)) // "Opération réussie !"
  .catch(erreur => console.error(erreur)); // "Erreur !"
```



- `async` et `await` simplifient l'utilisation des promesses.
- `async` déclare une fonction asynchrone.
- `await` attend la résolution d'une promesse.

# Exemple de async/await

```
async function fetchData() {  
  try {  
    const reponse = await fetch("https://api.example.com/data");  
    const data = await reponse.json();  
    console.log(data);  
  } catch (erreur) {  
    console.error("Erreur :", erreur);  
  }  
}  
  
fetchData();
```

- L'API 'fetch' permet de faire des requêtes HTTP.
- Syntaxe simple pour récupérer des données depuis une API.
- Retourne une promesse.

# Exemple de fetch

```
fetch("https://api.example.com/data")  
  .then(reponse => reponse.json())  
  .then(data => console.log(data))  
  .catch(erreur => console.error("Erreur :", erreur));
```

- Fonctionnalités ES6+ : fonctions fléchées, template literals, déstructuration.
- Promesses : gestion des opérations asynchrones.
- Async/Await : simplification des promesses.
- Requêtes HTTP avec 'fetch'.

Des questions sur le contenu de l'in ?