

# Introduction à JavaScript

Fidel

# Introduction : Qu'est-ce que JavaScript ?

## Le langage du web dynamique

JavaScript est le langage de script qui apporte l'interactivité aux pages web. Dans l'architecture web moderne :

- **HTML** structure le contenu
- **CSS** définit la présentation
- **JavaScript** gère l'interactivité et la logique métier côté client

## Cas d'usage typiques :

- Manipulation du DOM en temps réel
- Gestion d'événements utilisateur
- Communication asynchrone avec des APIs
- Validation de formulaires côté client

## Environnement d'exécution

JavaScript s'exécute principalement côté client dans le navigateur, mais également côté serveur avec Node.js.

# Environnement de Développement

## Outils requis

- **Éditeur de code** : Visual Studio Code, WebStorm, Sublime Text
- **Navigateur moderne** avec outils de développement intégrés
- **Console développeur** pour le débogage et les tests

## Intégration dans une page web

```
1 <script src="script.js"></script>
2 <!-- ou -->
3 <script>
4   // Code JavaScript inline
5 </script>
```

.


# Partie 1 : Syntaxe et Types Fondamentaux

# Déclaration de Variables

JavaScript propose trois mots-clés pour déclarer des variables, chacun avec sa portée et ses règles de réassignation.

## Portée et mutabilité

```
1 let message = "Bonjour le monde !"; // Variable réassignable, portée de bloc
2 const anneeActuelle = 2025; // Constante, portée de bloc
3 var ancienneVariable = "legacy"; // Portée de fonction (deprecated)
```



- **let** : Variable réassignable avec portée de bloc
- **const** : Référence immuable avec portée de bloc
- **var** : Ancienne syntaxe, portée de fonction, éviter en JavaScript moderne

# Système de Types

JavaScript est un langage à typage dynamique avec coercion automatique.

## Types primitifs

```
1 // String - chaînes de caractères
2 let nom = "Alice";
3 let template = `Utilisateur: ${nom}`; // Template literals
4
5 // Number - nombres en virgule flottante IEEE 754
6 let age = 30;
7 let prix = 19.99;
8
9 // Boolean - valeurs logiques
10 let estConnecte = true;
11 let estAdmin = false;
```

## Types complexes

```
1 // Array - collections ordonnées
2 let fruits = ["pomme", "banane", "fraise"];
3
4 // Object - collections de propriétés clé-valeur
5 let utilisateur = {
6   nom: "Alice",
7   age: 30,
8   roles: ["admin", "user"]
9 };
```

# Opérateurs et Expressions

## Opérateurs arithmétiques et logiques

```
1  let a = 10, b = 5;
2
3  // Arithmétiques
4  console.log(a + b, a - b, a * b, a / b, a % b); // 15, 5, 50, 2, 0
5
6  // Comparaisons strictes (recommandées)
7  console.log(a === b); // false - égalité stricte
8  console.log(a !== b); // true - inégalité stricte
9  console.log(a > b);    // true
10
11 // Opérateurs logiques
12 console.log(true && false); // false - ET logique
13 console.log(true || false); // true - OU logique
14 console.log(!true);         // false - NON logique
```



.

# Partie 2 : Structures de Contrôle

# Instructions Conditionnelles

Exécution conditionnelle de blocs de code basée sur des expressions booléennes.

```
1  let ageUtilisateur = 18;
2
3  if (ageUtilisateur >= 18) {
4      console.log("Accès autorisé");
5  } else {
6      console.log("Accès refusé");
7  }
8
9  // Conditions multiples
10 let score = 85;
11 if (score >= 90) {
12     console.log("Excellence");
13 } else if (score >= 70) {
14     console.log("Satisfaisant");
15 } else {
16     console.log("Insuffisant");
17 }
```

# Structures Itératives

Répétition contrôlée d'instructions pour le traitement de collections ou l'exécution répétitive.

## Boucle for classique

```
1 let fruits = ["pomme", "banane", "fraise"];
2
3 // Itération avec index
4 for (let i = 0; i < fruits.length; i++) {
5   console.log(`${i}: ${fruits[i]}`);
6 }
7
8 // for...of pour itérer sur les valeurs
9 for (let fruit of fruits) {
10  console.log(fruit);
11 }
12
13 // for...in pour itérer sur les propriétés
14 for (let index in fruits) {
15   console.log(index, fruits[index]);
16 }
```

.

# Partie 3 : Fonctions et Modularité

# Définition et Invocation de Fonctions

Les fonctions encapsulent la logique réutilisable et constituent l'unité de base de l'organisation du code.

## Syntaxes de déclaration

```
1 // Déclaration de fonction (hoistée)
2 function saluer(nom) {
3   return "Bonjour, " + nom + " !";
4 }
5
6 // Expression de fonction
7 const calculer = function(a, b) {
8   return a * b;
9 };
10
11 // Fonction fléchée (ES6+)
12 const doubler = (nombre) => nombre * 2;
13 const additionner = (a, b) => a + b;
```

## Invocation et valeur de retour

```
1 let message = saluer("Marc");
2 let resultat = calculer(5, 3);
3 console.log(message); // "Bonjour, Marc !"
4 console.log(resultat); // 15
```





## Partie 4 : Manipulation du DOM

Le Document Object Model (DOM) est l'interface de programmation pour interagir avec les documents HTML.

# Sélection d'Éléments

Accès aux éléments HTML via les méthodes de sélection du DOM.

```
1 // Sélecteurs CSS
2 const titre = document.querySelector('h1');
3 const boutons = document.querySelectorAll('.btn');
4
5 // Méthodes traditionnelles
6 const elementById = document.getElementById('mon-id');
7 const elementsByClass = document.getElementsByClassName('ma-classe');
```

## Structure HTML de référence

```
1 <!DOCTYPE html>
2 <html>
3 <head><title>Application</title></head>
4 <body>
5   <h1 id="titre">Titre principal</h1>
6   <button class="btn">Action</button>
7 </body>
8 </html>
```

# Modification de Contenu et Propriétés

Altération dynamique des éléments DOM pour créer des interfaces réactives.

```
1  const titre = document.querySelector('h1');
2
3  // Modification du contenu textuel
4  titre.textContent = "Nouveau titre";
5
6  // Modification du HTML interne
7  titre.innerHTML = "Titre avec <em>emphase</em>";
8
9  // Manipulation des attributs
10 titre.setAttribute('data-status', 'active');
11 titre.classList.add('highlight');
12 titre.classList.toggle('visible');
```

# Gestion d'Événements

Programmation événementielle pour répondre aux interactions utilisateur.

```
1  const bouton = document.querySelector('button');
2
3  // Gestionnaire d'événement
4  bouton.addEventListener('click', function(event) {
5      console.log('Événement click capturé');
6      event.preventDefault(); // Prévenir le comportement par défaut
7  });
8
9  // Avec fonction fléchée
10 bouton.addEventListener('click', (event) => {
11     alert("Interaction utilisateur détectée");
12 });
13
14 // Événements clavier
15 document.addEventListener('keydown', (event) => {
16     console.log(`Touche pressée: ${event.key}`);
17 });
```

.

# Perspectives et Évolution

# Récapitulatif Technique

Cette introduction a couvert :

- **Syntaxe moderne** : let/const, arrow functions, template literals
- **Types et structures** : primitifs, objets, tableaux
- **Logique de contrôle** : conditions, boucles, fonctions
- **Interface DOM** : sélection, modification, événements

# Projets Pratiques Recommandés

## Applications de démonstration

1. **Interface de calcul** - Opérations arithmétiques avec interface graphique
2. **Gestionnaire de tâches** - CRUD avec persistance locale
3. **Interface dynamique** - Manipulation de styles et animations CSS

## Progression suggérée

- Maîtrise des APIs natives du navigateur
- Concepts avancés : closures, prototypes, asynchrone
- Frameworks modernes : React, Vue.js, Angular



# Écosystème et Technologies Connexes

## Stack de développement moderne

- **Frontend** : React, Vue.js, Angular, Svelte
- **Backend** : Node.js, Express, API REST/GraphQL
- **Tooling** : Webpack, Vite, ESLint, TypeScript
- **Testing** : Jest, Cypress, Testing Library

## Domaines d'application

- Applications web single-page (SPA)
- Applications mobiles hybrides
- Applications desktop (Electron)
- Services backend et APIs

# Questions et Approfondissement

## Ressources de référence

- **MDN Web Docs** - Documentation technique de référence
- **ECMAScript Specification** - Standard du langage
- **Node.js Documentation** - Runtime serveur

## Communautés et veille technologique

- Stack Overflow pour la résolution de problèmes
- GitHub pour l'exploration de projets open source
- Conférences : JSConf, React Conf, Vue Conf

Speaker notes