**1. Principes Généraux du JavaScript**

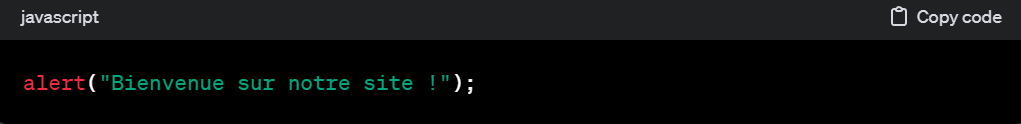
**Origines du JavaScript : Navigateurs et Serveurs**

JavaScript (JS) a été créé en **1995** par **Brendan Eich** alors qu'il travaillait pour Netscape Communications.

Initialement conçu pour améliorer l'expérience utilisateur dans les navigateurs web, JS est rapidement devenu un outil essentiel pour le développement **front-end**.

Avec l'émergence de **Node.js** en **2009**, JavaScript a étendu son domaine d'application aux serveurs, permettant aux développeurs de construire des applications **full-stack** avec un seul langage de

* Par exemple, un script simple pour afficher une alerte :



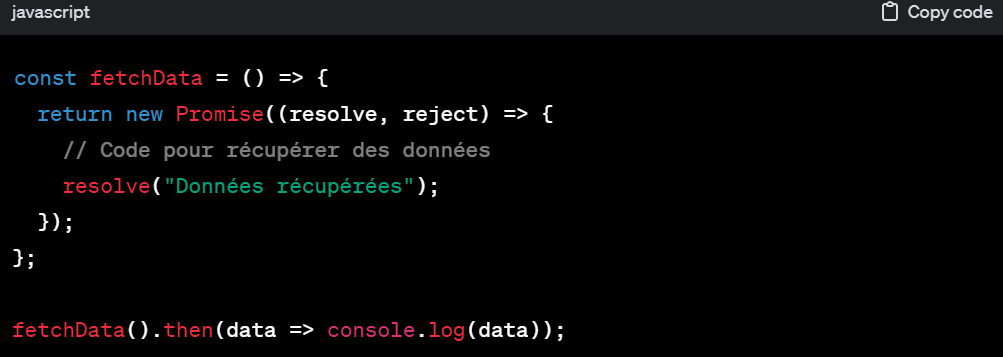
**Evolution du Langage**

JS a évolué à travers plusieurs versions, avec ECMAScript (ES) comme spécification standard.

ES6, sorti en 2015, a introduit des fonctionnalités majeures comme les classes, les modules, les promesses, et la syntaxe de déstructuration.

Depuis ES6, de nouvelles fonctionnalités sont ajoutées annuellement pour améliorer le langage et répondre aux besoins des développeurs modernes.

Par exemple :



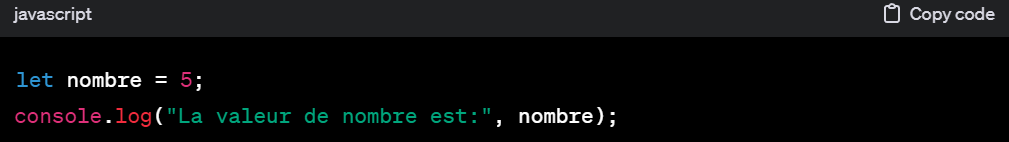
L'objet **Promise** (pour « promesse ») est utilisé pour réaliser des traitements de façon asynchrone. Une promesse représente une valeur qui peut être disponible maintenant, dans le futur voire jamais

**Environnement de Développement**

**Programmation** : JS peut être écrit dans n'importe quel éditeur de texte, mais les IDE tels que Visual Studio Code offrent des fonctionnalités avancées (suggestions de code, débogage intégré).

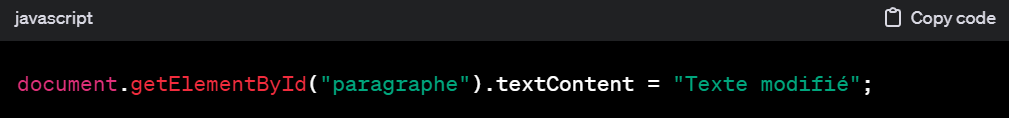
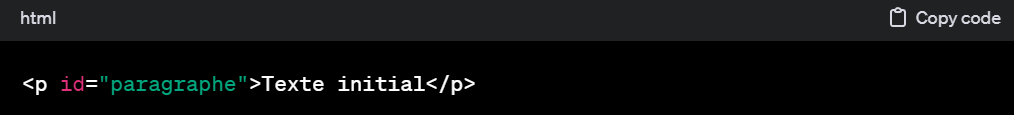
**Exécution** dans un navigateur ou via Node.js.

**Débogage** : Utilisation de console.log pour tracer les valeurs.



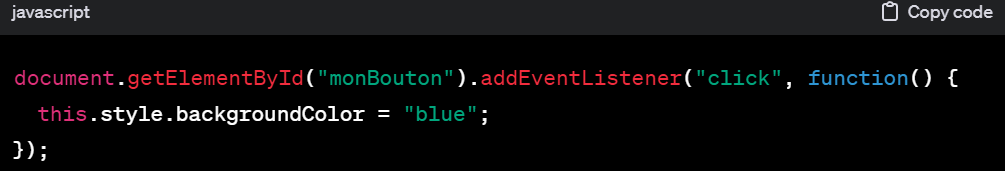
**Rappels HTML : DOM, Mise en Forme et Composants**

Le **Document Object Model (DOM)** représente la structure du document HTML. JS peut manipuler le DOM pour modifier dynamiquement le contenu et la structure des pages web.



**Mise en forme** : JS interagit avec les éléments HTML pour modifier leur apparence ou comportement.

**Composants :** JS permet de créer des composants interactifs comme des menus déroulants, des modaux, des sliders.



Rappels CSS : Style et Positionnement

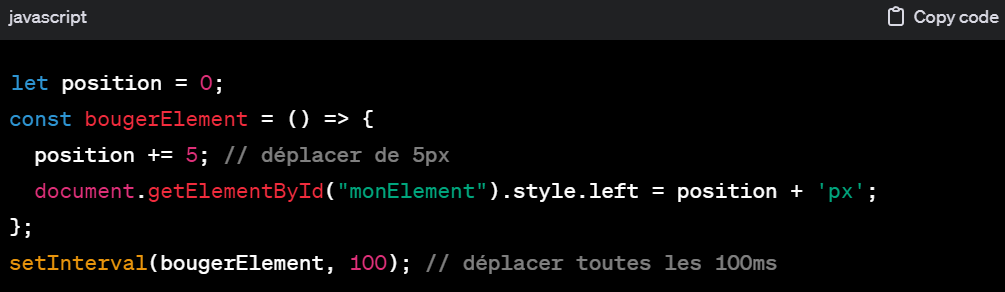
**Style :** JS peut modifier les styles CSS d'éléments pour des interactions dynamiques (ex: changer de couleurs, tailles).

Exemple modifier la **taille (font-size en css)** du texte **« Cliquez-moi »** qui se trouve sur le **boutton**



**Positionnement :** JS peut être utilisé pour manipuler le positionnement des éléments, comme pour des animations ou des dispositions interactives.

Exemple pour une animation simple pour déplacer de 5 px par la gauche un texte toutes les 0.1 secondes



**setInterval()** appelle à plusieurs reprises une fonction ou exécute un extrait de code, avec un **délai fixe** entre chaque appel.

**Conclusion :**

Ce cours a couvert les **fondamentaux de JavaScript**, un langage clé dans le développement web, en se concentrant sur ses origines, son évolution, son environnement de développement, et son interaction avec le HTML et le CSS.

* **Origines et Extension** : JavaScript 1995 a commencé comme un outil pour les **navigateurs** et s'est étendu aux **serveurs** avec Node.js 2009, démontrant sa polyvalence.
* **Évolution** : Avec des mises à jour comme **ES6 2015**, JavaScript a introduit des concepts modernes comme les **fonctions fléchées** et les **promesses**, améliorant ainsi la clarté et la puissance du langage.
* **Environnement de Développement** : Les développeurs utilisent des IDE pour la **programmation**, des navigateurs ou Node.js pour l'**exécution**, et des outils comme **console.log** pour le **débogage**.
* **Interaction avec HTML/CSS** : JavaScript manipule le **DOM** pour changer le contenu et la structure des pages web. Il utilise également le CSS pour modifier **styles** et **positionnements**, rendant les sites web plus interactifs et dynamiques.

En résumé, **JavaScript est un pilier** du développement web moderne. Sa compréhension approfondie est essentielle pour créer des expériences utilisateur riches et interactives. Les exemples pratiques fournis illustrent comment ces concepts s'appliquent dans des scénarios réels.