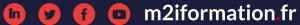


INTRODUCTION











HISTORIQUE









CRÉATION DE JAVASCRIPT

JavaScript a été créé en 1995 par Brendan Eich, alors employé chez Netscape Communications.











ÉVOLUTION ET VERSIONS

JavaScript a évolué au fil des années avec de nombreuses versions. La version actuelle est ECMAScript 2021 (ES12).









UTILISATION











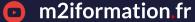
NAVIGATEURS WEB

JavaScript est principalement utilisé pour ajouter des fonctionnalités dynamiques aux pages web.











SERVEURS (NODE.JS)

Node.js permet d'utiliser **JavaScript** côté serveur pour créer des applications web, des API, etc.









APPLICATIONS MOBILES ET DE BUREAU

JavaScript est également utilisé pour développer des applications mobiles et de bureau grâce à des frameworks tels que React Native et Electron.









PRÉSENTATION DE JAVASCRIPT: VARIABLES



INTRODUCTION AUX VARIABLES

Les variables sont utilisées pour stocker des **données** et les utiliser dans le programme.

En JavaScript, le mot-clé var, let ou const est utilisé pour déclarer une variable.

Mot-clé	Description
var	Variables ayant une portée de fonction
let	Variables ayant une portée de bloc
const	Variables ayant une portée de bloc et immuables

```
var x = 10;
let y = 20;
const PI = 3.14;
```











MATHÉMATIQUES









ADDITION

Pour **additionner** deux nombres en JavaScript, utilisez l'opérateur +.











SOUSTRACTION

Pour soustraire deux **nombres** en JavaScript, utilisez l'opérateur **=**.

var difference = 5 - 3; // difference vaut 2











MULTIPLICATION

Pour **multiplier** deux nombres en **JavaScript**, utilisez l'opérateur *.











DIVISION

Pour diviser deux nombres en **JavaScript**, utilisez l'opérateur /.









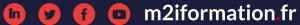


COMPARAISON











LOGIQUES









ET LOGIQUE

L'opérateur **ET logique** (&&) renvoie true si les deux opérandes sont vrais, sinon il renvoie false.











OU LOGIQUE

L'opérateur **OU logique** (||) renvoie true si au moins un des opérandes est vrai, sinon il renvoie false.

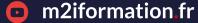
```
let resultat = a || b; // true
```













PRÉSENTATION TOTALE DEJAVASCRIPT





ASSIGNATION









ASSIGNATION SIMPLE

En JavaScript, l'opérateur d'affectation (=) est utilisé pour attribuer une valeur à une **variable**.









ADDITION ET ASSIGNATION

L'opérateur d'affectation **addition** (+=) ajoute une valeur à une variable existante et affecte le résultat à cette variable.













SOUSTRACTION ET ASSIGNATION

L'opérateur d'affectation soustraction (-=) soustrait une valeur d'une variable existante et affecte le résultat à cette variable.









MULTIPLICATION ET ASSIGNATION

L'opérateur d'affectation multiplication ($\star=$) multiplie une valeur par une variable existante et affecte le résultat à cette variable.









DIVISION ET ASSIGNATION

L'opérateur d'affectation division (/=) divise une variable existante par une valeur et affecte le résultat à cette variable.









PRÉSENTATION TOTALE DE JAVASCRIPT : FONCTIONS



INTRODUCTION AUX FONCTIONS

Les **fonctions** sont des blocs de code **réutilisables** pour effectuer une action spécifique.

```
function nomDeLaFonction(parametre1, parametre2) {
  return resultat;
nomDeLaFonction(arg1, arg2);
```











DÉCLARATION D'UNE FONCTION











SYNTAXE

Pour déclarer une **fonction**, utilisez le mot-clé <u>function</u> suivi du nom de la fonction, d'une paire de parenthèses et d'un bloc de code entre accolades :

```
function maFonction() {
```











PARAMÈTRES

Les paramètres sont des variables passées à une fonction. Ils sont définis entre les parenthèses lors de la déclaration de la fonction :

```
function maFonction(param1, param2) {
```









RETOUR DE VALEUR

Utilisez le mot-clé **return** pour renvoyer une valeur depuis une fonction :

```
function maFonction(a, b) {
```









INVOCATION D'UNE FONCTION









SYNTAXE

Pour appeler une fonction, utilisez son nom suivi d'une paire de parenthèses :

maFonction();

Note : Assurez-vous que la fonction est définie avant de l'appeler, sinon vous obtiendrez une erreur.









ARGUMENTS

Les **arguments** sont des valeurs passées aux **paramètres** d'une fonction lors de son **invocation** :

maFonction(arg1, arg2);











OBJETS INTÉGRÉS









OBJETS INTÉGRÉS

JavaScript propose des objets intégrés pour faciliter certaines opérations courantes.

- Math: opérations mathématiques
- **Date** : manipulation de dates
- **Array** : manipulations d'arrays
- **String** : manipulations de chaînes de caractères
- **Object**: manipulations d'objets

```
let nombre = Math.random() * 10;
let aujourd'hui = new Date();
let liste = [1, 2, 3];
let texte = "Hello, world!";
let objet = { nom: "John", age: 30 };
```









DATE

L'objet **Date** permet de gérer les **dates** et **heures**.

• Créer un objet Date :

```
let dateActuelle = new Date();
```

• Obtenir une partie spécifique de la date (jour, mois, année, etc.) :

```
let jour = dateActuelle.getDate();
let mois = dateActuelle.getMonth() + 1; // Les mois sont indexés de 0 à 11
let annee = dateActuelle.getFullYear();
```











CRÉER UNE DATE

const now = new Date(); // Date actuelle const specificDate = new Date("2022-01-01"); // Date spécifique

Méthode	Description
new Date()	Crée une date avec l'heure actuelle
new Date(string)	Crée une date avec une chaîne de caractères









MÉTHODES DE L'OBJET Date

```
now.getFullYear(); // Retourne l'année
now.getMonth(); // Retourne le mois (0 - 11)
now.getDate(); // Retourne le jour du mois (1 - 31)
```













MATH

L'objet **Math** fournit des **fonctions mathématiques**.

Fonction	Description
Math.abs(x)	Renvoie la valeur absolue de x
Math.max(x, y, z,, n)	Renvoie le nombre le plus grand de la liste
Math.min(x, y, z,, n)	Renvoie le nombre le plus petit de la liste
Math.random()	Renvoie un nombre aléatoire entre 0 et 1
Math.round(x)	Arrondit x à l'entier le plus proche
Math.sqrt(x)	Renvoie la racine carrée de x









EXEMPLES D'UTILISATION

```
Math.random(); // Retourne un **nombre aléatoire** entre 0 et 1
Math.floor(3.8); // Arrondit à l'**entier inférieur** (3)
Math.ceil(3.2); // Arrondit à l'**entier supérieur** (4)
```











ARRAY

L'objet **Array** permet de gérer des **listes d'éléments**.

Création d'une array:

```
let liste = ['élément1', 'élément2', 'élément3'];
```

Accéder aux éléments:

```
console.log(liste[0]); // 'élément1'
console.log(liste[1]); // 'élément2'
```

Ajouter et enlever des éléments:

```
liste.push('élément4'); // Ajoute un élément à la fin
liste.pop(); // Enlève le dernier élément
```













CRÉER UN TABLEAU

```
const array2 = new Array(1, 2, 3);
```

Syntaxe courte	Syntaxe longue
const array = [1, 2, 3];	const array2 = new Array $(1, 2, 3)$;











MÉTHODES DE L'OBJET Array

```
array.push(4); // Ajoute un élément à la fin
array.pop(); // Supprime le dernier élément
array.shift(); // Supprime le premier élément
array.unshift(0); // Ajoute un élément au début
```











STRING

L'objet String permet de gérer des chaînes de caractères.

- Concaténation: str1 + str2
- Longueur:str.length
- Accès à un caractère : str[index]
- Méthodes utiles :
 - indexOf()
 - slice()
 - replace()
 - split()











CRÉER UNE CHAÎNE DE CARACTÈRES









MÉTHODES DE L'OBJET String

```
str.length; // Retourne la longueur de la chaîne
str.toUpperCase(); // Convertit la chaîne en majuscules
str.toLowerCase(); // Convertit la chaîne en minuscules
str.indexOf("world"); // Retourne l'index de la sous-chaîne (7)
```

















