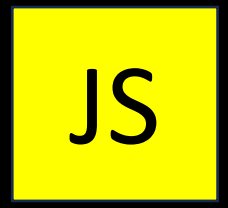

JAVASCRIPT

Version 0.1
Rédigé et mis en page par
Romain Fournier.

TABLE DES MATIERES

1. Présentation
2. Pourquoi choisir JavaScript ?
3. Introduction aux spécificités
4. Définitions & mécanismes

Version 0.1
Rédigé et mis en page par
Romain Fournier.

JS

1. Présentation

Un langage crée en **1995** par **Brendan Eich**, Ingénieur chez Netscape.

- Le nom **Javascript** a été choisi pour surfer sur la vague de popularité de Java.

En **1997**, standardisation par **ECMA** international :

- Les nouvelles versions suivent la norme **ECMAScript** (par ex. **ES6**).
- Le chiffre indique le numéro de la version.

Version majeure : **ES6** (sortie en **2015**).

- La base recommandée pour coder.
- En 2025, version actuelle : ES15. (prochaine révision en Juin)

Forme la trinité du web avec **HTML** et **CSS**.

- Il dynamise les sites web.
- Il réagit aux interactions de l'utilisateur.

Version 0.1
Rédigé et mis en page par
Romain Fournier.

JS

2. Pourquoi choisir JavaScript ?

- **Portabilité** : Exécuté nativement dans tous les navigateurs sans installation.
- **Polyvalence** : Développement full-stack, mobile, ou desktop avec des frameworks comme React, Vue, ou Electron.
- **Communauté massive** : Soutenu par un écosystème riche en bibliothèques et outils.
- **Facilité d'accès** : Parfait pour débuter et pour des projets avancés.

JS

3. Introduction aux spécificités

Les slides suivantes vont détailler les spécificités de ce langage de programmation.

Le JS est un langage :

- **De haut niveau** : abstrait et proche de la logique humaine.
- **Interprété** : exécuté directement, sans compilation préalable.
- **Faiblement typé** : les types de données peuvent changer dynamiquement.
- **Orienté prototype** : basé sur des objets et leurs prototypes.
- **Multi-paradigme** : supporte différents styles de programmation (fonctionnelle, impérative, etc.).

Version 0.1
Rédigé et mis en page par
Romain Fournier.

JS

4. Définitions & mécanismes → Langage de haut niveau

Langage :

- **→ de haut niveau ←**

- Plus proche du **langage humain** que du langage machine.
- **Plus facile à lire** et à comprendre.
- **Compatible avec différents systèmes** grâce à une abstraction qui réduit les configurations nécessaires.
- **Cache les détails complexes** au niveau matériel.

JS

4. Définitions & mécanismes → Langage interprété

Langage :

- ~~de haut niveau~~
- **→ Interprété ←**

Il existe 2 catégories de langages de programmation :

- **Interprété**
- **Compilé**

Interprété :

- Le code est **lu et exécuté ligne par ligne** par un programme : l'**interpréteur**.
- Exemple : En JavaScript, l'**interpréteur est le navigateur** (côté front).

Compilé :

- Le code est **traduit intégralement** avant son exécution par un programme : le **compilateur**.
- Exemple : Les langages comme **C ou Java** génèrent un fichier exécutable après compilation.

Version 0.1
Rédigé et mis en page par
Romain Fournier.

JS 4. Définitions & mécanismes → Langage faiblement typé

Langage :

- ~~de haut niveau~~
- ~~Interprété~~
- **→ faiblement typé ←**

- **Pas de déclaration explicite de type** : Le type de la variable n'est pas défini à l'avance.
- **Typage dynamique** : Le type de donnée est déterminé lors de l'affectation ou de l'initialisation.
- **Conversions implicites** :
  - Lors d'une **concaténation**, les valeurs sont converties en chaînes de caractères :

    Exemple : '4' + 4 = '44' (et non 8).
  - Lors d'une opération mathématique, les valeurs sont converties si nécessaire :

    Exemple : '4' - 2 = 2.

Version 0.1
Rédigé et mis en page par
Romain Fournier.

JS

4. Définitions & mécanismes → Langage orienté prototype

Langage :

- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- **→ orienté prototype ←**

- Le **prototypage** est un concept de la programmation orientée objet.
- En JavaScript, **tout objet possède un prototype**.
  - Un prototype est un objet dont un autre objet **hérite les propriétés et méthodes**.

JS

4. Définitions & mécanismes → Langage multi-paradigme

Langage :

- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- **→ multi-paradigme ←**

- **Impérative** → Série d'instructions simples.
- **Procédurale** → Structuré en chaîne de fonctions.
- **Fonctionnelle** → Fonctions 1ère classe, pures, closures.
- **POO Prototypée** → Héritage via prototype.
- **Événementielle** → Actions liées aux utilisateurs.
- **Asynchrone** → Non bloquant, promesses, async/await.

Note :

Ces paradigmes peuvent être combinés dans un même projet.

*Automatic Semicolon Insertion

JS

4. Définitions & mécanismes → Les instructions

Langage :

- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- ~~multi-paradigme~~

→ Les instructions. ←

- Une instruction représente **une ligne de code**.
- Une instruction peut être :
  - Une **déclaration de variable**, initialisée ou non.
  - Une **fonction**.
  - Une **structure de contrôle** (ex. if, boucle for/while…), qui peut regrouper plusieurs instructions.

La fin d'une instruction est marquée par un **point-virgule.**

→ Bien que **facultatifs**, il est recommandé de toujours les utiliser.

L'utilisation de la mécanique ASI* peut provoquer des erreurs, notamment lorsqu'on termine et commence une instruction par des [] ou ().

Version 0.1
Rédigé et mis en page par
Romain Fournier.

JS

4. Définitions & mécanismes → Les types de données

Langage :
- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- ~~multi-paradigme~~

~~Les instructions.~~

→ Les types de données. ←

Comme tout langage, JavaScript utilise des types de données qui dictent le comportement des variables et des opérations.

Il existe 8 types de données.

Types primitifs (7):
- String
- Number
- BigInt
- Boolean
- Symbol
- Null *(special)*
- Undefined (*special*)

Type complexe (1):
- Object (reference)
  - Object
  - Array
  - Function
  - Date
  - …

JS 4. Définitions & mécanismes → Les types de données -> String

Langage :
- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- ~~multi-paradigme~~

~~Les instructions.~~

→ Les types de données. ←

String = Chaîne de caractères.
- Peut contenir : lettres, nombres, symboles, mots, phrases.
- Délimitée par un symbole spécial :
  - ‘ : Apostrophe simple
  - " : Apostrophe double
  - ` : Accent grave (permets l’interpolation d’expressions)

Exemples :

L’apostrophe simple
```
let username = 'Roger est au PMU !’;
```
L’apostrophe double
```
let username = "Roger va à la boulangerie.";
```
L’accent grave
```
let username = `Roger rentre chez lui.`;
```

JS 4. Définitions & mécanismes → Les types de données -> Number

Langage :
- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- ~~multi-paradigme~~

~~Les instructions.~~

→ Les types de données. ←

Number = Nombre (entier et décimaux).
- Représente toutes les valeurs numériques.

Exemples :
```
let age = 42;         // Nombre entier
let price = 9.49;     // Nombre décimal
let temp = -10;       // Nombre négatif
```

Valeurs spéciales :
- **Infinity** : résultat de divisions par 0.
- **NaN** : retour d'une opération illogique

4. Définitions & mécanismes → Les types de données -> Boolean

Langage :
- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- ~~multi-paradigme~~

~~Les instructions.~~

→ Les types de données. ←

Boolean = Valeur logique.
- Deux états possibles : `true` ou `false`.
- Très pratique pour les conditions !

Exemples :

```
let isLogged = true;

if (isLogged) {
    console.log("Welcome !");
}
```

Version 0.1
Rédigé et mis en page par
Romain Fournier.

JS 4. Définitions & mécanismes → Les types de données -> null / undefined

Langage :
- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- ~~multi-paradigme~~

~~Les instructions.~~

→ Les types de données. ←

null = Absence volontaire de valeur.
- Indique qu'une variable est explicitement vide.

Exemples :

```
let user = null;
```

undefined = non défini.
- Indique qu'une variable a été déclarée mais n'a pas de valeur.

Exemples :

```
let user;
console.log(user); // undefined
```

JS

4. Définitions & mécanismes → Les variables

Langage :
- ~~de haut niveau~~
- ~~Interprété~~
- ~~faiblement typé~~
- ~~orienté prototype~~
- ~~multi-paradigme~~

~~Les instructions.~~
~~Les types de données.~~
→ Les variables. ←

Une variable stocke une donnée dans la mémoire de l'ordinateur qui exécute le script.
Cette donnée peut être manipulée pour effectuer des actions.

Pour initialiser une variable, il y a plusieurs étapes :
- Un mot-clé (`let`, `const`, `var`).
- Un nom.
- Un opérateur d'affectation (=).
- Une valeur (de n'importe quel type : nombre, tableau, fonction…).

```
let username = "darkJuju_94";
// "username" est le nom de la variable de type String,
"darkJuju_94" est la valeur
```