Introduction à JavaScript

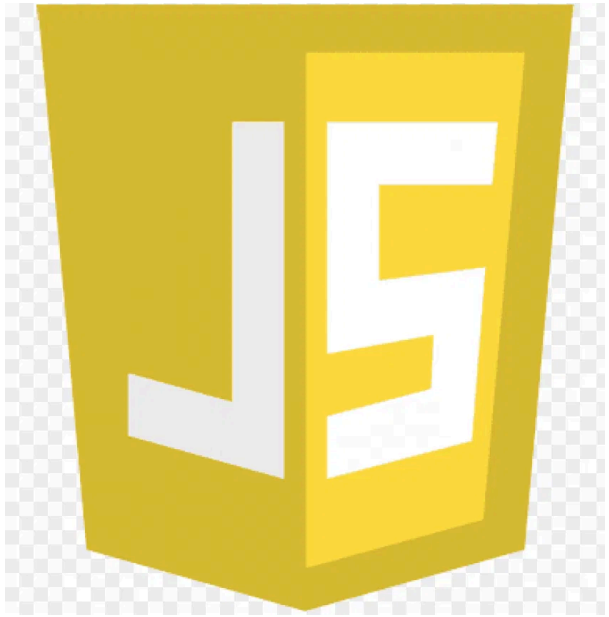
Les Bases du Web Interactif

Pour les étudiants de 1ère année Supinfo

Au Programme

1. L'Histoire de JavaScript
2. Variables et Types de Données
3. Fonctions et Déclarations
4. Conditions et Opérateurs
5. Boucles et Itération

L'Histoire de JavaScript



CRÉATEUR & CONTEXTE

Développé en **1995 en 10 jours** par **Brendan Eich** chez Netscape pour rendre le web plus interactif.

OBJECTIFS INITIAUX

- Exécution côté client (dans le navigateur)
- Facile à apprendre pour les développeurs
- Complémentaire à Java (d'où le nom)

ÉVOLUTION

Initialement nommé **LiveScript**, renommé en **JavaScript** en 1996. Standardisé en 1997 sous le nom **ECMAScript**.

Les Versions Importantes d'ECMAScript

ES5

Publié en 2009

- ✓ **Manipulation de Tableaux** - Méthodes améliorées pour travailler avec les arrays.
- ✓ **Support JSON** - Parsing et stringify des données JSON.
- ✓ **Mode Strict** - Meilleure détection des erreurs de programmation.

ES6 / ES2015

Publié en 2015 - La Grande Révolution

- ✓ **Classes** - Syntaxe orientée objet plus familière.
- ✓ **Modules** - import/export pour mieux organiser le code.
- ✓ **Promesses** - Gestion de l'asynchrone simplifiée.
- ✓ **Fonctions Fléchées** - Syntaxe concise et moderne.

📅 **Après ES6 :** De nouveaux standards sont publiés chaque année (ES2016, ES2017, etc.) avec des améliorations progressives et des nouvelles fonctionnalités.

Intégrer JavaScript dans une Page HTML

LA BALISE <SCRIPT>

JavaScript s'intègre dans une page HTML avec la balise **`<script>`**. Elle peut contenir du code JavaScript directement ou charger un fichier externe.

```
<script src="mon-fichier.js"></script>
```

PLACEMENT : HEAD VS BODY

- **`Dans le <head>`** : Le script est chargé avant le rendu de la page. Peut ralentir le chargement.
- **`En fin de <body>`** : Le script est chargé après le rendu. Meilleure performance.

ATTRIBUTS : DEFER ET ASYNC

- **`defer`** : Charge le script en arrière-plan sans bloquer le rendu. Exécution après le chargement de la page.
- **`async`** : Charge et exécute le script dès qu'il est disponible. Peut bloquer le rendu.

```
<script src="mon-fichier.js" defer></script>
```

Variables et Conventions

CONVENTIONS DE NOMMAGE

1. **camelCase**

   Première lettre minuscule, puis première lettre de chaque mot en majuscule. Ex: **uneVariable**, **monNom**, **ageUtilisateur**

2. **Case-Sensitive**

   JavaScript distingue les majuscules des minuscules. **maVariable** ≠ **mavariable**. Attention à la casse !

3. **Noms Descriptifs**

   Utilisez des noms clairs qui expliquent le contenu. Ex: **nombreUtilisateurs** au lieu de **n**

TROIS FAÇONS DE DÉCLARER DES VARIABLES

`const`

Immutable. Ne peut pas être réassigné après initialisation.

✓ Recommandé par défaut

`let`

Peut être réassigné. Limité au bloc courant (meilleur que var).

✓ Utiliser si réassignation nécessaire

`var`

Ancien. Susceptible au hoisting et à des comportements imprévisibles.

✗ À éviter absolument

Les Types de Données Primitives

Nombres

JavaScript ne distingue pas les entiers des décimaux. Inclut aussi **NaN** (Not A Number) pour les opérations invalides.

Chaînes de Caractères

Texte entouré de guillemets simples `'...'`, doubles `"..."`, ou backticks `` `...` `` pour les template literals.

Booléens

Valeurs logiques : **true** ou **false**. Utilisés pour les conditions et les tests logiques.

Undefined

Valeur par défaut d'une variable déclarée mais non initialisée. Indique l'absence de valeur.

Null

Valeur intentionnellement vide, assignée par le développeur pour indiquer quelque chose de vide ou d'absent.

Symbols (ES6)

Valeurs uniques générées à chaque appel. Utiles pour créer des clés univoques dans les objets.

💡 **Important**

En JavaScript, **tout est un objet**, même les fonctions ! Les types primitives ont des méthodes et des propriétés grâce à l'auto-boxing (conversion automatique en objet).

Déclarer et Utiliser des Fonctions

Déclaration Classique

La façon traditionnelle de déclarer une fonction avec le mot-clé **function**.

```
function add(a, b) {
return a + b;
}
```

Expression de Fonction

Assigner une fonction à une variable. Peut être anonyme ou nommée.

```
const add = function(a, b) {
return a + b;
};
```

Fonctions Fléchées (ES6)

Syntaxe concise avec **=>**. Return implicite si une seule ligne.

```
const add = (a, b) => a + b;
```

Appel de Fonction

Exécuter une fonction en utilisant son nom suivi de parenthèses avec les arguments.

```
const result = add(5, 3);
console.log(result); // 8
```

CONCEPTS AVANCÉS

- Une fonction peut **retourner une autre fonction** (fonctions d'ordre supérieur)
- Une fonction peut **recevoir une autre fonction en paramètre** (callbacks)

Conditions et Opérateurs

OPÉRATEURS DE COMPARAISON

`===`	Égalité stricte (valeur + type)	`!==`	Inégalité stricte
`==`	Égalité (valeur seulement)	`!=`	Inégalité
`>`	Plus grand que	`<`	Moins que
`>=`	Plus grand ou égal	`<=`	Moins ou égal

OPÉRATEURS LOGIQUES

`&&`

ET logique : vrai si les deux conditions sont vraies

`||`

OU logique : vrai si une condition est vraie

`!`

NON logique : inverse la condition

⚠️ **Important**

Préférez toujours **===** à **==** ! L'égalité stricte évite les conversions de type implicites qui peuvent causer des bugs.

Structures Conditionnelles

if / else if / else

Exécute un bloc de code si une condition est vraie. Permet de tester plusieurs conditions avec **else if**.

```
if (age >= 18) {
console.log("Adulte");
} else if (age >= 13) {
console.log("Ado");
} else {
console.log("Enfant");
}
```

switch

Compare une expression à plusieurs valeurs. Évite les forêts de if/else if quand on teste une seule variable.

```
switch (jour) {
case 1:
console.log("Lundi");
break;
case 2:
console.log("Mardi");
break;
default:
console.log("Autre");
}
```

💡 **Conseil**

Utilisez **if/else** pour des conditions complexes avec plusieurs variables. Utilisez **switch** pour comparer une seule variable à plusieurs valeurs. N'oubliez pas le **break** dans le switch pour éviter les "fall-through" !

Boucles : Répéter du Code

while

Répète un bloc de code tant que la condition est vraie. Vérifie la condition avant chaque itération.

```
let i = 0;
while (i < 5) {
console.log(i);
i++;
}
```

do...while

Exécute le bloc une fois, puis répète tant que la condition est vraie. Garantit au moins une exécution.

```
let i = 0;
do {
console.log(i);
i++;
} while (i < 5);
```

for

Répète un bloc un nombre défini de fois. Idéal quand on connaît le nombre d'itérations.

```
for (let i = 0; i < 5; i++) {
console.log(i);
}
```

💡 Quand utiliser quoi ?

Utilisez **for** quand vous connaissez le nombre d'itérations (ex: parcourir un tableau). Utilisez **while** ou **do...while** quand la condition dépend d'une variable externe. Attention aux boucles infinies !

Bonnes Pratiques pour Débuter

1 Utilisez const par Défaut

Déclarez vos variables avec **const**. Utilisez **let** seulement si vous devez réassigner. Jamais **var**.

2 Préférez === à ==

Utilisez toujours **===** pour comparer. Cela évite les conversions de type implicites qui causent des bugs.

3 Nommez Vos Variables Clairement

Utilisez des noms descriptifs en **camelCase**. Ex: **nombreUtilisateurs** au lieu de **n** ou **num**.

4 Commentez Votre Code

Expliquez la logique complexe avec des commentaires. Cela aide les autres (et vous-même) à comprendre le code plus tard.

5 Testez en Console (F12)

Ouvrez la console du navigateur (F12) pour tester votre code et déboguer. Utilisez **console.log()** pour afficher des valeurs.

6 Évitez les Variables Globales

Encapsulez votre code dans des fonctions. Les variables globales peuvent causer des conflits et des bugs difficiles à trouver.

Conclusion : JavaScript est Partout

1 **Le Langage du Web Côté Client**

JavaScript est **le seul langage nativement supporté par les navigateurs**. Il permet de créer des expériences web interactives et dynamiques.

2 **ES6 (2015) a Révolutionné le Langage**

Avec **const, let, classes, modules, promesses, et fonctions fléchées**, ES6 a modernisé JavaScript et l'a rendu plus facile à utiliser.

3 **Trois Concepts Clés à Maîtriser**

Variables (const, let), **Fonctions** (déclaration, expression, fléchées), et **Conditions/Boucles** (if/else, switch, while, for). Ces trois piliers sont la base de tout programme JavaScript.

4 **La Pratique est la Clé**

Écrivez du code, testez-le, déboguez-le. La compréhension vient avec la pratique régulière. Utilisez la console du navigateur (F12) pour expérimenter.

Questions ?

N'hésitez pas à poser vos questions ! La maîtrise de **JavaScript** est essentielle pour devenir un développeur web compétent.

POUR APPROFONDIR

📚 **MDN Web Docs** - Documentation officielle et complète

🔗 **JavaScript.info** - Tutoriels interactifs et détaillés

💻 **Eloquent JavaScript** - Livre gratuit en ligne

🎓 **Supinfo Resources** - Matériel pédagogique de l'école

BESOIN D'AIDE ?

Consultez vos instructeurs ou les ressources Supinfo