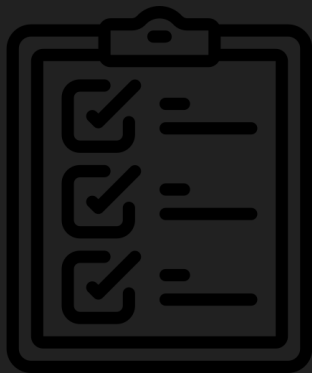


Introduction au langage JavaScript

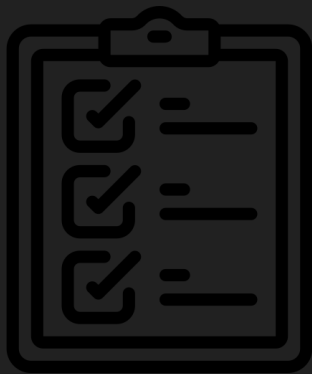
Maîtrisez le langage du web interactif

Plan du cours



- Les objectifs
- Le JavaScript
- Histoire et évolution
- Avantages
- Outils de développement
- La balise script
- Variables
- Commentaires
- Type de données
- Opérateurs arithmétiques
- Opérateurs de comparaison

Plan du cours



- Opérateurs logiques
- Structures conditionnelles
- Structures itératives
- Les tableaux
- Les fonctions
- Les objets prédéfinis
- La POO
- Fonctions anonymes et fléchées
- JavaScript DOM
- Gestion des événements

Objectifs



Les Objectifs

- Comprendre les fondamentaux
- Manipuler le DOM
- Gérer les événements
- Manipuler les données
- Gestion des erreurs
- Bonnes Pratique

Le JavaScript



Le JavaScript

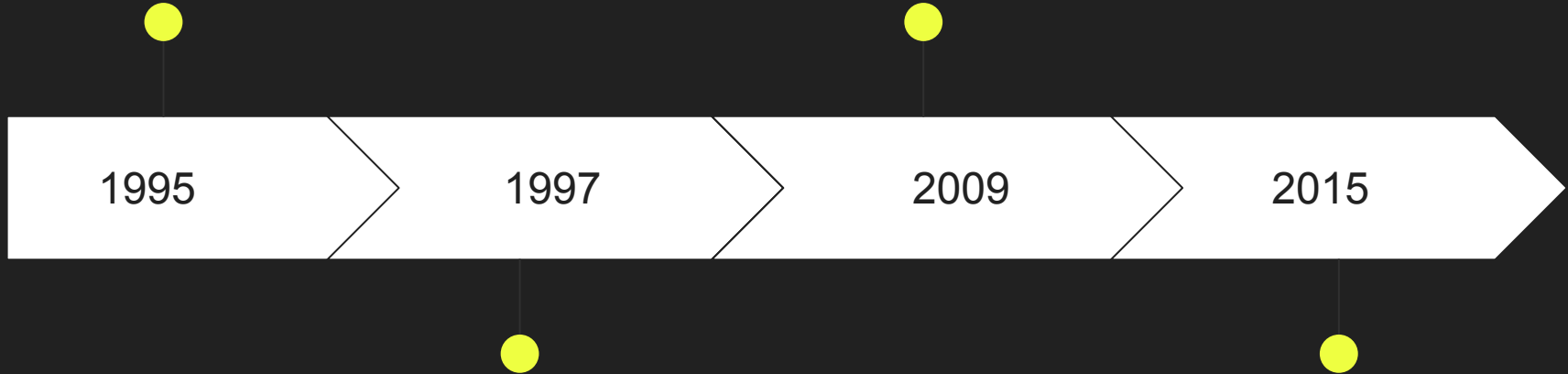
- Langage de programmation script interprété
- Langage orienté objet et événementiel
- Langage Interactif et dynamique
- Principalement utilisé dans le développement web
- Langage polyvalent

Histoire et évolution



Brendan Eich ingénieur chez Netscape communications créa LiveScript destiné au navigateur Netscape actuel Firefox

L'introduction de Node.js en 2009 a permis l'exécution de JavaScript côté serveur, ouvrant la voie au développement d'applications web complètes à l'aide d'un seul langage



Soumis à l'ECMA International (European Computer Manufacturers Association) pour standardisation

ECMAScript 6 (ES6) ou ECMAScript 2015 apporte des améliorations (classes, de modules, de fonctions fléchées, de la déstructuration...)

Avantages



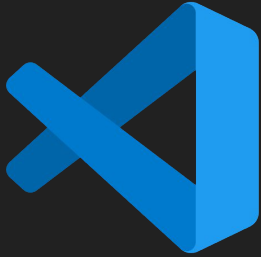
Avantages

- Exécution côté client et côté serveur
- Facilité d'apprentissage
- Compatibilité avec tous les navigateurs modernes
- Interactivité avec le DOM
- Large communauté et active
- Riche écosystème
- L'asynchronicité

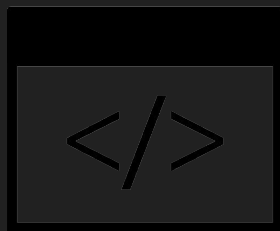
Outils de développement



Outils de développement



La balise script



La balise script

Pour écrire du code Javascript deux possibilité


1. Directement dans le code HTML avec la balise **script**
2. Dans un fichier spécialement dédié à l'écriture de code JavaScript ce fichier doit avoir l'extension **“.js”**

La balise script

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script>
    code JavaScript
  </script>
</head>
<body>
  <script>
    code JavaScript
  </script>
</body>
</html>
```

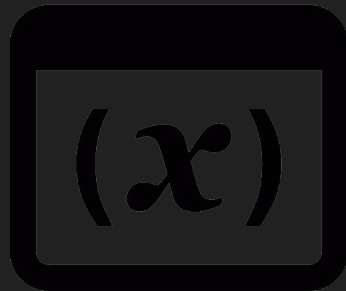

La balise script

```
<!DOCTYPE html>
<html lang="en">
<head>
  ...
  <script src="fichier1.js"></script>
  ...
</head>
<body>
  ...
  <script src="fichier2.js"></script>
  ...
</body>
</html>
```



fichiers JavaScript

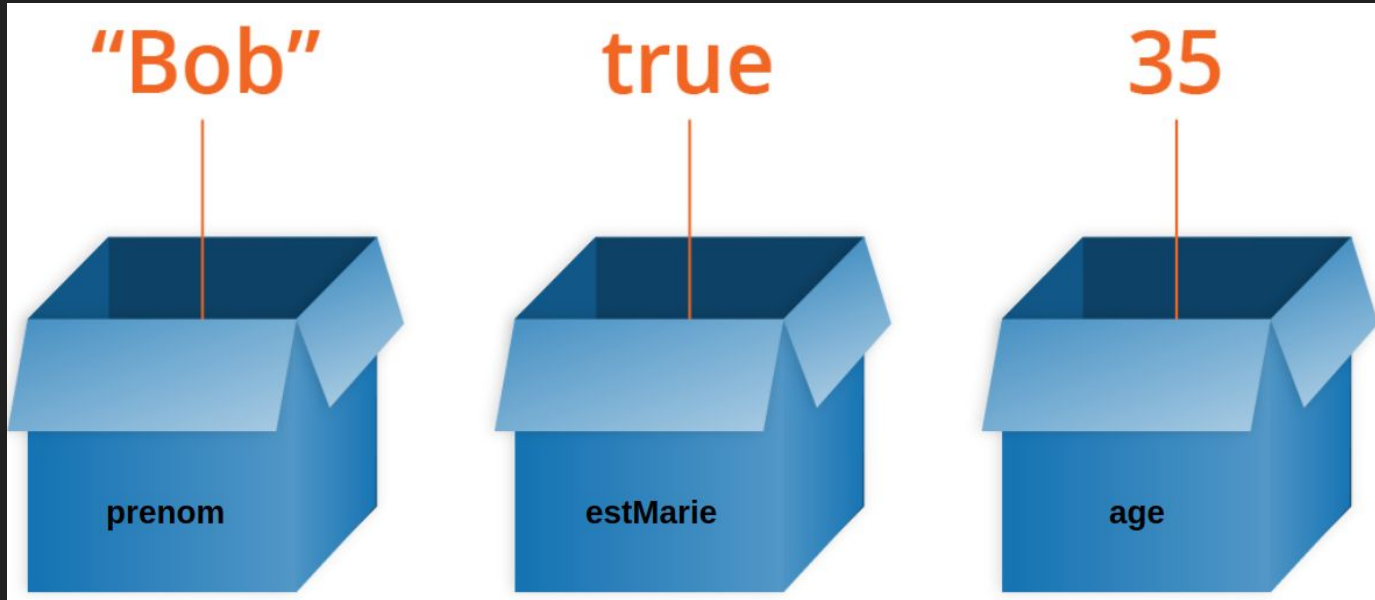
Variables



Variables

- Le principe des variables en programmation est de permettre le stockage et la manipulation de données. Une variable est un espace réservé en mémoire où une valeur peut être stockée et référencée à l'aide d'un nom.
- Une variable a une durée de vie qui en générale correspond à celle de l'exécution du programme.
- La donnée stockée dans une variable est d'un certain type (nombre, du texte,...)

Variables



Variables (declaration)

En JavaScript pour déclarer une variable il faut utiliser un des trois mots clé suivant pour faire comprendre à l'interpréteur qu'on souhaite réserver un emplacement mémoire afin d'y stocker une information

1. **Var**
2. **let**
3. **const**

Variables (affectation)

Après avoir créé une variable on peut lui assigner une valeur en utilisant le signe “**=**”.

exemple:

```
const nom = "Dupond";  
let prenom = "Robert";  
var age = 15;
```

Variables (règles de nommage)

Les noms de variables en JavaScript doivent respecter les règles suivantes:

1. Comporter uniquement des caractères alpha numérique(**a - z | A - Z | 0 - 9**) plus les signes \$ et _ qui sont autorisés
2. Doit obligatoirement commencer par un caractère alphabétique ou l'un des signes \$ et _
3. Ne doit pas être un mot clé réservé du JavaScript
4. Doit être unique

Variables (règles de nommage)

Exemple de nom de variable valide:

```
let age = 25;  
let firstName = "John";  
let _lastName = "Doe";  
let $count = 10;  
let myVariable = "test";
```


Variables (règles de nommage)

Exemple de nom de variable non valide:

```
let 123abc; // Commence par un chiffre
let my-variable; // Contient un tiret (-)
let first name; // Contient un espace
let @test; // Contient un caractère spécial (@)
let let; // Utilise un mot clé réservé ("let" est un mot clé réservé du
JavaScript)
```

Variables(différences var, let, const)

Noté qu'il existe quelques différences entre les mots clé **var**, **let** et **const** ainsi une variable déclarée à l'aide du mot clé **var** peut être re-déclaré.

exemple:

```
var age = 10;  
var age = 20;
```

Dans ce cas la variable age est déclaré et la valeur 10 lui ai assigné à la ligne 1 ensuite à la ligne 2 la variable age prend la valeur 20 c'est juste un remplacement de valeur sur la variable age qui est effectué.

Variables(différences var, let, const)

Contrairement au mot clé **var** on ne peut pas re-déclarer une variable à l'aide des mot clé **let** et **const**.

exemple:

```
let age = 10;  
let age = 20;
```

```
const nom = "Dupond";  
const nom = "Dupont";
```

Ces déclarations généreront une erreur.

Variables(différences var, let, const)

Une variable déclarée avec le mot clé **const** garde sa valeur constante durant l'exécution du programme alors que pour **let** et **var** sa valeur peut changer durant l'exécution du programme.

exemple:

```
const nom = "Dupond";  
const nom = "Dupont";  
var prenom = "Toto";  
prenom = "Titi";
```

```
const nom = "Dupond";  
nom = "Dupont";
```

ceci n'est pas autorisé

Commentaires



Commentaires

En javascript un commentaire est un code qui ne sera pas pris en compte par l'interpréteur au moment de l'exécution du programme. Les commentaires servent uniquement à documenter votre code.

Pour écrire un commentaire en JavaScript on peut le faire sur une seule ligne en plaçant deux caractères slash “//” devant votre commentaire.

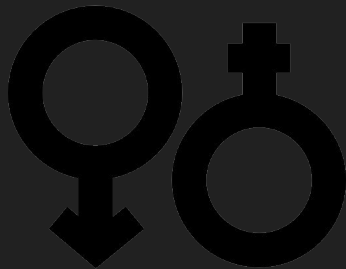
```
// la variable pour contenir l'age de l'utilisateur  
let age = 10;  
age = 20; // modification de l'age
```

Commentaires

Vous pouvez aussi écrire de commentaire sur plusieurs lignes cependant il faut placer votre commentaire entre les caractères “/*” et “*/”

```
/*  
je developpe une application web en javascript qui aura les fonctionnalité  
suivante  
1 - fonctionnalité 1  
2 - fonctionnalité 2  
3 - .....  
*/
```

Type de données



Type de données

Le JavaScript est un langage à typage dynamique ainsi pas besoin de préciser le type d'une variable au moment de la déclaration l'interpréteur va automatiquement typer la variable en fonction de sa valeur.

De plus, une variable peut changer de types d'un moment à l'autre durant l'exécution du programme en fonction des valeurs qu'elle contient.

Type de données

Les nombres le type Number représente les valeurs numériques, entières ou décimales:

```
let age = 25;  
let pi = 3.14;
```

Type de données

Une chaîne de caractère représente une séquence de caractères entourée de guillemets simples ou doubles.

```
let salutation = "Bonjour";  
let prenom = 'toto';
```

Type de données

Une chaîne de caractère représente une séquence de caractères entourée de guillemets simples ou doubles.

```
let salutation = "Bonjour";
```

```
let prenom = 'toto';
```

Concaténation de chaînes de caractères

```
let nom = 'Alice';
```

```
let message = 'Bienvenue, ' + nom + '!'; // Utilisation de l'opérateur de  
concaténation (+)
```

```
console.log(message); // Affiche "Bienvenue, Alice!"
```

Type de données

Longueur d'une chaîne de caractères

```
let str = 'Hello';  
console.log(str.length); // Affiche 5 (le nombre de caractères dans la  
chaîne)
```

Accès aux caractères d'une chaîne de caractères

```
let str = 'JavaScript';  
console.log(str[0]); // Affiche 'J' (le premier caractère de la chaîne)  
console.log(str.charAt(4)); // Affiche 'S' (le caractère à l'index 4 de la  
chaîne)
```

Type de données

Méthodes de manipulation des chaînes de caractères

```
let str = 'Bonjour JavaScript';  
console.log(str.toLowerCase()); // Affiche 'bonjour javascript'  
                                   (conversion en minuscules)  
console.log(str.toUpperCase()); // Affiche 'BONJOUR JAVASCRIPT'  
                                   (conversion en majuscules)  
console.log(str.indexOf('JavaScript')); // Affiche 8 (index de la première  
occurrence de 'JavaScript')  
console.log(str.replace('JavaScript', 'Python')); // Affiche 'Bonjour  
Python' (remplacement de 'JavaScript' par 'Python')
```

Type de données

Échappement de caractères

```
let str = 'J\'aime les guillemets simples'; // Utilisation de  
l'échappement avec un backslash (\)  
console.log(str); // Affiche "J'aime les guillemets simples"  
  
let chaine = "il a dit \"j'ai fini\"";  
console.log(chaine); // affiche il a dit "j'ai fini"
```

Type de données

Le type boolean représente une valeur de vérité qui peut être soit `true` (vrai) soit `false` (faux).

```
let estMineur = false;  
let isActive = true;
```


Type de données

Tableau (Array) : Il représente une collection d'éléments, où chaque élément peut être de n'importe quel type de données.

```
let tableau = [1, 'deux', true, ["un", 3]];
```

Type de données

Il représente une collection de propriétés clé-valeur. Les objets en JavaScript peuvent être des objets intégrés (comme Math, Date, etc.) ou des objets définis par l'utilisateur.

```
let personne = {  
  nom: 'Jean',  
  age: 30,  
  adresse: 'Paris'  
};
```

Type de données (conversion)

En JavaScript, il est possible de convertir des valeurs d'un type à un autre en utilisant des méthodes de conversion intégrées.

Conversion en nombre :

Number()

```
let str = "123";  
let num = Number(str); // Convertit la chaîne de caractères en nombre  
console.log(num); // Affiche 123
```

Type de données (conversion)

Conversion en chaîne de caractères :

String() : Convertit une valeur en chaîne de caractères.

```
let num = 123;  
let str = String(num); // Convertit le nombre en chaîne de caractères  
console.log(str); // Affiche "123"
```

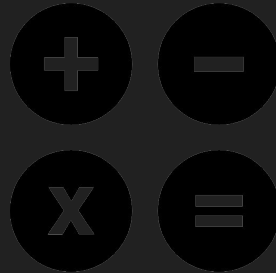
Type de données (conversion)

Conversion en booléen :

Boolean() : Convertit une valeur en booléen. Les valeurs suivantes sont évaluées à false: false, 0, "", null, undefined, NaN. Toutes les autres valeurs sont évaluées à true.

```
let num = 0;  
let bool = Boolean(num); // Convertit le nombre en booléen  
console.log(bool); // Affiche false
```

Opérateurs arithmétiques



Opérateurs arithmétiques

En JavaScript, il existe plusieurs opérateurs arithmétiques pour effectuer des opérations mathématiques sur des valeurs numériques. Voici les opérateurs arithmétiques les plus couramment utilisés

Opérateurs arithmétiques

Addition (+) : L'opérateur d'addition est utilisé pour additionner deux valeurs.
Par exemple :

```
let resultat = 5 + 3; // resultat vaut 8
```


Opérateurs arithmétiques

Soustraction (-) : L'opérateur de soustraction est utilisé pour soustraire une valeur d'une autre.

```
let resultat = 10 - 4; // résultat vaut 6
```

Opérateurs arithmétiques

Multiplication (*) : L'opérateur de multiplication est utilisé pour multiplier deux valeurs.

```
let resultat = 15 / 5; // resultat vaut 3
```

Opérateurs arithmétiques

Division (/) : L'opérateur de division est utilisé pour diviser une valeur par une autre.

```
let resultat = 15 / 5; // résultat vaut 3
```

Opérateurs arithmétiques

Modulo (%) : L'opérateur de modulo renvoie le reste de la division entière de la première valeur par la deuxième valeur.

```
let resultat = 10 % 3; // résultat vaut 1 (le reste de la  
division de 10 par 3)
```

Opérateurs arithmétiques

Incrémentation (++) : L'opérateur d'incrémentation est utilisé pour augmenter la valeur d'une variable de 1.

```
let compteur = 5;
```

```
compteur++; // compteur vaut maintenant 6
```

Opérateurs arithmétiques

Décrémentation (--) : L'opérateur de décrémentation est utilisé pour diminuer la valeur d'une variable de 1.

```
let compteur = 8;
```

```
compteur--; // compteur vaut maintenant 7
```

Opérateurs de comparaison



Opérateurs de comparaison

En JavaScript, voici les opérateurs de comparaison les plus couramment utilisés

Opérateur d'égalité (==) : Vérifie l'égalité entre deux valeurs, en effectuant une conversion de type si nécessaire.

```
console.log(5 == 5);    // true
```

```
console.log('5' == 5);  // true (conversion de type)
```

```
console.log(5 == '6');  // false
```


Opérateurs de comparaison

Opérateur de non-égalité (!=) : Vérifie si deux valeurs sont différentes, en effectuant une conversion de type si nécessaire.

```
console.log(5 != 3);      // true  
console.log('5' != 5);    // false (conversion de type)  
console.log(5 != '6');    // true
```

Opérateurs de comparaison

Opérateur de strict égalité (===) : Vérifie l'égalité entre deux valeurs, sans effectuer de conversion de type. Les types doivent être identiques et les valeurs doivent correspondre.

```
console.log(5 === 5);    // true
console.log('5' === 5);  // false (types différents)
console.log(5 === '5');
```

Opérateurs de comparaison

Opérateurs de comparaison (<, >, <=, >=)

```
console.log(5 < 10);      // true
console.log(5 > 10);      // false
console.log(5 <= 5);     // true
console.log(5 >= 10);     // false
```

Opérateurs logiques



Opérateurs de comparaison

En JavaScript, les opérateurs logiques permettent de combiner des expressions booléennes et d'évaluer des conditions complexes. Voici les opérateurs logiques les plus couramment utilisés :

Opérateurs de comparaison

Opérateur logique ET (&&) : Renvoie true si toutes les expressions sont évaluées à true, sinon renvoie false.

```
console.log(true && true);    // true
console.log(true && false);   // false
console.log(false && false);  // false
```

Opérateurs de comparaison

Opérateur logique OU (||) : Renvoie true si au moins l'une des expressions est évaluée à true, sinon renvoie false.

```
console.log(true || true);    // true
console.log(true || false);   // false
console.log(false || false);  // false
```

Opérateurs de comparaison

Opérateur logique NON (!) : Inverse la valeur d'une expression booléenne.
Renvoie true si l'expression est false, et renvoie false si l'expression est true

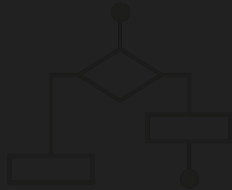
```
console.log(!true);    // false  
console.log(!false);   // true
```


Opérateurs de comparaison

Ces opérateurs logiques peuvent être combinés pour créer des expressions plus complexes.

```
console.log((5 > 3) && (10 < 20));           // true
console.log((5 > 10) || (7 < 3));             // false
console.log(!(5 === 5) && (4 !== '4'));      // false
```

Structures conditionnelles



Structures conditionnelles

En JavaScript, il existe plusieurs structures conditionnelles pour exécuter des blocs de code en fonction de certaines conditions. Voici les principales structures conditionnelles en JavaScript

Structures conditionnelles

la structure if

La structure if permet d'exécuter un bloc de code si une condition est évaluée à true.

```
if (condition) {  
    // Code à exécuter si la condition est vraie  
}
```

Structures conditionnelles

la structure if....else

La structure "if...else" en JavaScript permet d'exécuter un bloc de code si une condition est vraie et un autre bloc de code si la condition est fausse.

```
var heure = 14;
if (heure < 12) {
  console.log("Bonne matinée !");
} else {
  console.log("Bonne après-midi !");
}
```

Structures conditionnelles

on peut également ajouter des conditions supplémentaires en utilisant l'instruction "else if".

```
var heure = 14;
if (heure < 12) {
    console.log("Bonne matinée !");
} else if (heure >= 12 && heure < 18) {
    console.log("Bonne après-midi !");
} else {
    console.log("Bonne soirée !");
}
```

Structures conditionnelles

La structure switch

La structure "switch" en JavaScript permet d'évaluer une expression et d'exécuter différents blocs de code en fonction de la valeur de cette expression. Voici un exemple d'utilisation de la structure "switch"

Structures conditionnelles

```
var couleur = "rouge";
switch (couleur) {
  case "bleu":
    console.log("La couleur est bleue.");
    break;
  case "rouge":
    console.log("La couleur est rouge.");
    break;
  case "vert":
    console.log("La couleur est verte.");
    break;
  default:
    console.log("La couleur n'est ni bleue, ni rouge, ni verte.");
}
```


Structures conditionnelles

La structure conditionnelle ternaire

La structure ternaire en JavaScript est une alternative concise à la structure conditionnelle "if...else". Elle permet d'évaluer une condition et de retourner une expression en fonction de cette condition.

```
var heure = 14;  
var message = (heure < 12) ? "Bonne matinée !" : "Bonne après-midi !";  
console.log(message);
```

Liens utils

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- <https://www.w3schools.com/js/default.asp>
- <https://devdocs.io/javascript/>