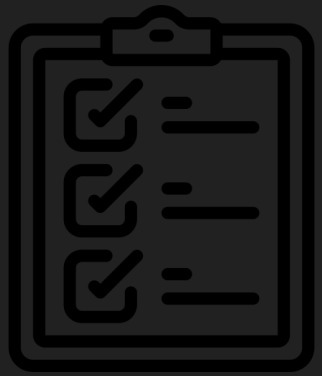
Introduction au langage JavaScript

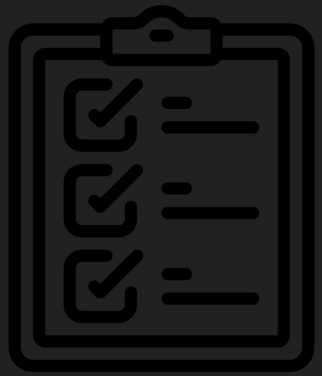
Maîtrisez le langage du web interactif

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 iteratives
- Les tableaux
- Les fonctions
- Les objets prédéfinis
- La POO
- Fonctions anonymes et fléchées
- JavaScript DOM
- Gestion des événements

Les objets prédéfinis

Les objets prédéfinis

En JavaScript, il existe plusieurs objets prédéfinis qui fournissent des fonctionnalités et des méthodes pour effectuer diverses tâches. Voici quelques-uns des objets prédéfinis couramment utilisés en JavaScript :

- Array : Cet objet est utilisé pour stocker et manipuler une collection ordonnée d'éléments. Il fournit des méthodes telles que push, pop, shift, unshift, splice, etc., pour ajouter, supprimer ou modifier des éléments dans un tableau.
- String : Cet objet est utilisé pour manipuler des chaînes de caractères. Il possède des méthodes pour concaténer, rechercher, extraire des sous-chaînes, convertir en majuscules ou en minuscules, etc.

Les objets prédéfinis

- Math : Cet objet fournit des méthodes et des propriétés pour effectuer des opérations mathématiques. Il inclut des méthodes telles que Math.random, Math.floor, Math.ceil, Math.abs, etc.
- Date : Cet objet est utilisé pour représenter des dates et des heures. Il fournit des méthodes pour accéder et manipuler des informations sur les dates, telles que getFullYear, getMonth, getDate, getHours, etc.
- JSON : Cet objet fournit des méthodes pour convertir des objets JavaScript en format JSON (JavaScript Object Notation) et vice versa. Il inclut des méthodes telles que JSON.stringify et JSON.parse.

Les objets prédéfinis

L'objet Math

```
Math.PI; // Une propriété qui représente la valeur de Pi (environ 3.14159).
Math.abs(x); // Renvoie la valeur absolue de x.
Math.ceil(x); // Renvoie le plus petit entier supérieur ou égal à x.
Math.floor(x); // Renvoie le plus grand entier inférieur ou égal à x.
Math.round(x); // Renvoie la valeur de x arrondie à l'entier le plus proche.
Math.max(x1, x2, ..., xn); // Renvoie le plus grand nombre parmi les arguments donnés.
Math.min(x1, x2, ..., xn); // Renvoie le plus petit nombre parmi les arguments donnés.
Math.random(); // Renvoie un nombre flottant pseudo-aléatoire compris entre 0 et 1.
Math.sqrt(x); // Renvoie la racine carrée de x.
Math.pow(x, y); // Renvoie x élevé à la puissance y.
```

Les objets prédéfinis

L’objet Date

```
new Date(); // Crée un nouvel objet Date représentant la date et l'heure actuelles.
new Date(valeur); // Crée un nouvel objet Date à partir d'une valeur spécifiée
getDate(); // Renvoie le jour du mois (entre 1 et 31) pour la date spécifiée.
getMonth(); // Renvoie le mois (entre 0 et 11) pour la date spécifiée.
getFullYear(); // Renvoie l'année (en format complet à quatre chiffres) pour la date spécifiée.
getHours(); // Renvoie l'heure (entre 0 et 23) pour la date spécifiée.
getMinutes(); // Renvoie les minutes (entre 0 et 59) pour la date spécifiée.
getSeconds(); // Renvoie les secondes (entre 0 et 59) pour la date spécifiée.
```

Les objets prédéfinis

```
let date = new Date();
console.log(date); // 2023-06-28T18:00:46.279Z
```

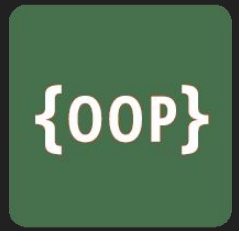
La sortie que vous obtenez, est le format ISO 8601 de la date et de l'heure représentées par l'objet Date.

Analysons cette représentation en détail :

- 2023-06-28 : Il s'agit de la partie de la date, dans l'ordre année-mois-jour.
- T : C'est le séparateur utilisé pour indiquer la transition vers la partie de l'heure.
- 18:00:46.279 : Il s'agit de la partie de l'heure, dans l'ordre heure:minute:seconde.milliseconde.
- Z : C'est l'indicateur de fuseau horaire, où "Z" signifie le fuseau horaire UTC (temps universel coordonné) ou zéro décalage horaire.

Il est important de noter que la représentation exacte peut varier en fonction du système d'exploitation, du navigateur ou de la plateforme sur laquelle le code est exécuté.

POO



POO

La programmation orientée objet (POO) est un paradigme de programmation qui structure le code en utilisant des objets pour représenter des concepts, des entités ou des éléments du monde réel. Son objectif principal est d'abstraire le monde réel et de résoudre des problèmes complexes en les décomposant en objets interagissant entre eux.

En POO, un objet est une instance d'une classe, qui est un modèle ou un plan permettant de créer des objets. Une classe définit les propriétés (variables) et les méthodes (fonctions) communes à tous les objets créés à partir de cette classe.

POO

```
class Personne {
   constructor(nom, age) {
     this.nom = nom;
     this.age = age;
   }
    saluer() {
     console.log("Bonjour, je m'appelle "+this.nom+ " et j'ai "+ this.age+ " ans.");
   }
}
```

POO

Dans cet exemple, la classe Personne est définie avec deux propriétés (nom et age) et une méthode (saluer()) qui affiche un message de salutation avec le nom et l'âge de la personne.

Une fois qu'une classe est définie, des objets individuels, appelés instances de classe, peuvent être créés à partir de cette classe. Chaque instance d'objet aura ses propres valeurs pour les propriétés définies dans la classe, mais partagera les mêmes méthodes.

POO

Objet

Un objet est une instance spécifique d'une classe. En programmation orientée objet (POO), un objet représente une entité autonome qui regroupe des données (propriétés) et des comportements (méthodes) liés. Il est créé à partir d'une classe en utilisant le mot-clé `new`.

Un objet peut être considéré comme une version concrète ou concrétisée d'une classe. Il possède des valeurs spécifiques pour ses propriétés et peut exécuter les méthodes définies dans sa classe.

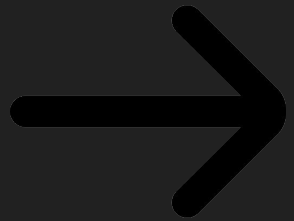
POO

Voici un exemple d'objet créé à partir de la classe Personne que nous avons définie précédemment :

```
var personne1 = new Personne("Alice", 25);
var personne2 = new Personne("Bob", 30);

personne1.saluer(); // Affiche "Bonjour, je m'appelle Alice et j'ai 25 ans."
personne2.saluer(); // Affiche "Bonjour, je m'appelle Bob et j'ai 30 ans."
```

Fonctions anonymes et fléchées



Fonctions anonymes et fléchées

Une fonction anonyme, également appelée fonction sans nom, est une fonction déclarée sans spécifier de nom. Elle est souvent utilisée dans des contextes où une fonction est requise comme valeur, telle que les rappels (callbacks)

Fonctions anonymes et fléchées

```
setTimeout(function() {
    console.log("Ceci est un rappel de fonction anonyme.");
}, 2000);
```

Dans cet exemple, une fonction anonyme est utilisée comme rappel pour exécuter du code après une attente de 2 seconde, grâce à la fonction setTimeout.

Fonctions anonymes et fléchées

Les fonctions fléchées offrent une syntaxe plus concise que les fonctions traditionnelles. Elles peuvent souvent être écrites sur une seule ligne sans nécessiter l'utilisation explicite du mot-clé return.
La syntaxe des fonctions fléchées permet d'éviter l'encombrement du code avec des accolades et le mot-clé return pour des fonctions courtes et simples.

```
// Fonction traditionnelle
function addition(a, b) {
    return a + b;
}

// Fonction fléchée équivalente
const additionFlechee = (a, b) => a + b;
```

Liens utils

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