Introduction au JavaScript Orienté Objet

Classes et Héritage

Qu'est-ce que la Programmation Orientée Objet (POO) ?

- La POO est un paradigme de programmation basé sur des objets.
- Les objets sont des instances de classes.
- Les classes définissent les propriétés et les méthodes des objets.
- JavaScript utilise des prototypes pour l'héritage, ce qui permet une grande flexibilité.

Définition des Classes

- Avant ES6, JavaScript utilisait des fonctions constructrices et des prototypes.
- Avec ES6, JavaScript introduit la syntaxe des classes.
- Les classes permettent de définir des constructeurs, des méthodes et des propriétés.

Définition des Classes : Exemple

```
class Personne {
    constructor(nom, age) {
        this.nom = nom;
        this.age = age;
    }

    presenter() {
        console.log(`Je m'appelle ${this.nom} et
        j'ai ${this.age} ans.`);
    }
}
```

Bonnes Pratiques de Nommage

- **Nommage des Classes** : Utilisez des noms de classes en PascalCase (chaque mot commence par une majuscule).
  - Exemple : class Utilisateur, class Produit.
- **Nommage des Fichiers** : Utilisez des noms de fichiers en PascalCase (chaque mot commence par une majuscule).
  - Exemple : Utilisateur.js, Produit.js.
- **Nommage des Méthodes et Propriétés** : Utilisez des noms en camelCase (premier mot en minuscule, mots suivants en majuscule).
  - Exemple : nomComplet, ajouterAuPanier.
- **Consistance** : Maintenez une consistance dans le nommage pour améliorer la lisibilité et la maintenabilité du code.
- **ATTENTION** : un fichier contenant une classe contient **uniquement la classe**

Constructeurs et Instances

- Le constructeur est une méthode spéciale appelée lors de la création d'une instance de la classe.
- Les instances sont des objets créés à partir de la classe.
- Chaque instance peut avoir des propriétés et des méthodes spécifiques.

Constructeurs et Instances : Exemple

```
const personne1 = new Personne("Alice", 30);
personne1.presenter(); // Je m'appelle Alice et j'ai 30 ans.
```

Concept d'Héritage

- L'héritage permet à une classe de dériver des propriétés et des méthodes d'une autre classe.
- Utilisation du mot-clé extends pour créer une sous-classe.
- Les sous-classes peuvent surcharger les méthodes de la super-classe.

Concept d'Héritage : Exemple

```
class Etudiant extends Personne {
    constructor(nom, age, etudes) {
        super(nom, age);
        this.etudes = etudes;
    }

    presenter() {
        console.log(`Je m'appelle ${this.nom},
        j'ai ${this.age} ans et j'étudie
        ${this.etudes}.`);
    }
}
```

Surcharge de Méthodes

- La surcharge de méthodes permet de redéfinir une méthode de la super-classe dans la sous-classe.
- Utilisation de super() pour appeler la méthode de la super-classe.
- Permet d'ajouter ou de modifier le comportement des méthodes héritées.

Surcharge de Méthodes : Exemple

```
class Professeur extends Personne {
    constructor(nom, age, matiere) {
        super(nom, age);
        this.matiere = matiere;
    }

    presenter() {
        super.presenter();
        console.log(`J'enseigne ${this.matiere}.`);
    }
}
```

Avantages de l'Héritage

- Réutilisation du code.
- Organisation et structuration du code.
- Facilité de maintenance et d'évolution.
- Polymorphisme : possibilité de définir des méthodes de manière différente dans les sous-classes.

Conclusion

- Les classes et l'héritage sont des concepts fondamentaux en POO.
- JavaScript utilise l'héritage prototypique pour permettre la réutilisation et l'extension des fonctionnalités.
- Les classes ES6 facilitent la définition et l'utilisation des objets en JavaScript.
- Les travaux pratiques permettent de mettre en application ces concepts de manière concrète.

TP 1: Création de Classes Simples

- Objectif : Créer une classe Livre avec des propriétés titre, auteur et anneePublication.
- Ajouter une méthode afficherDetails qui affiche les détails du livre.

TP 2: Héritage et Surcharge de Méthodes

- Objectif : Créer une classe Employe avec des propriétés nom, prenom et salaire.
- Créer une sous-classe Manager qui hérite de Employe et ajoute une propriété departement.
- Surcharger la méthode afficherDetails pour inclure le département.

TP 3: Héritage Multiple

- Objectif : Créer une classe Vehicule avec des propriétés marque et modele.
- Créer une sous-classe VoitureElectrique qui hérite de Vehicule et ajoute une propriété autonomie.
- Créer une sous-classe VoitureAutonome qui hérite de VoitureElectrique et ajoute une méthode activerAutonomie.