Apprendre le JavaScript - Introduction aux classes

Les classes en JavaScript permettent de regrouper des fonctions et des variables qui ont un lien entre elles. Cela permet de mieux organiser le code et de le rendre plus lisible. Les classes sont très utiles pour définir des objets qui ont des propriétés et des méthodes communes et qui seront instanciés, ou copiés, plusieurs fois.

Classes en JavaScript

La classe est un modèle qui permet de créer des objets. Elle définit les propriétés et les méthodes communes à tous les objets créés à partir de cette classe. Les objets créés à partir d'une classe sont appelés des **instances** de cette classe.

Pour définir une classe en JavaScript, on utilise le mot-clé `class` suivi du nom de la classe. Par convention, le nom des classes commence par une majuscule. Voici un exemple de classe en JavaScript :

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

    bonjour() {
        console.log(`Hello, my name is ${this.prenom} ${this.nom}`);
    }
}
```

Constructeur

Le constructeur est une méthode spéciale qui est appelée lors de la création d'une nouvelle instance de la classe. Il permet d'initialiser les propriétés de l'objet. Le constructeur est défini avec la méthode `constructor`.

Propriétés et méthodes

Les propriétés et les méthodes d'une classe sont définies à l'intérieur de la classe. Les **propriétés** sont des **variables** qui définissent l'état de l'objet, tandis que les **méthodes** sont des **fonctions** qui définissent le comportement de l'objet.

Normes de nommage

Les classes en JavaScript suivent généralement les conventions de nommage de la programmation orientée objet.

Le nom de la classe est généralement écrit en **PascalCase** (ou **UpperCamelCase**), où chaque mot commence par une majuscule. Par exemple, `Personne` est le nom de la classe.

Les propriétés et les méthodes d'une classe sont généralement nommées en **camelCase**. Les noms des classes commencent par une majuscule, tandis que les noms des propriétés et des méthodes commencent par une minuscule. Par exemple, `Personne` est le nom de la classe, `prenom` et `nom` sont des propriétés, et `bonjour` est une méthode.

this

Le mot-clé `this` fait référence à l'objet courant. Il permet d'accéder aux propriétés et aux méthodes de l'objet à l'intérieur de la classe. Par exemple, `this.prenom` fait référence à la propriété `prenom` de l'objet courant. Lorsque l'on crée une nouvelle instance de la classe, le mot-clé `this` fait référence à cette instance (copie).

Instanciation d'une classe

Pour créer une nouvelle instance d'une classe, on utilise le mot-clé `new` suivi du nom de la classe et des arguments du constructeur. Voici un exemple d'instanciation de la classe `Personne` :

```
let personne1 = new Personne("Maxime", "Lacasse"); // Crée une nouvelle instance de
la classe Personne
personne1.bonjour(); // Affiche "Hello, my name is Maxime Lacasse"

let personne2 = new Personne("David", "Ross"); // Crée une nouvelle instance de la
classe Personne
personne2.bonjour(); // Affiche "Hello, my name is David Ross"
```

Dans cet exemple, nous créons deux instances de la classe `Personne` : `personne1` et `personne2`. Chaque instance a ses propres propriétés `prenom` et `nom`, qui sont initialisées lors de la création de l'instance.

Pourquoi utiliser des classes ?

Les classes en JavaScript permettent de mieux organiser le code en regroupant les fonctions et les variables qui ont un lien entre elles. Elles permettent de définir des objets qui ont des propriétés et des méthodes communes et qui seront instanciés plusieurs fois. Les classes facilitent la réutilisation du code et le rendent plus lisible et maintenable.

Exemple d'une classe pour une boite modale

Dans cet exemple, nous allons créer une classe

```
class BoiteModale {
    constructor(titre, message, conteneurHTML) {
        this.titre = titre;
        this.message = message;
        this.modaleTemplate = document.querySelector("template#modale").content;
        this.modaleHTML;
        this.conteneurHTML = conteneurHTML;
```

```
        this.injecterHTML();
    }

    injecterHTML() {
        let clone = this.modaleTemplate.cloneNode(true);
        this.conteneurHTML.appendChild(clone);

        this.modaleHTML = this.conteneurHTML.lastElementChild;
        this.modaleHTML.classList.add("invisible");
        this.modaleHTML.querySelector(".modale__titre").textContent = this.titre;
        this.modaleHTML.querySelector(".modale__message").textContent =
this.message;

        let bouton = this.modaleHTML.querySelector(".modale__btn-fermer");
        bouton.addEventListener(
            "click",
            function () {
                this.fermer();
            }.bind(this)
        );
    }

    afficher() {
        this.modaleHTML.classList.remove("invisible");
    }

    fermer() {
        console.log("Fermeture de la modale");
        this.modaleHTML.remove(); //On supprime la modale du DOM
    }
}
```

Il est possible de créer une nouvelle instance de la classe `BoiteModale` et d'afficher une boite modale en utilisant les méthodes `afficher` et `fermer` :

```
let conteneur = document.querySelector(".conteneur");
let boiteModale = new BoiteModale("Titre de la modale", "Message de la modale",
conteneur);
boiteModale.afficher();

// Fermer la boite modale après 3 secondes
setTimeout(() => {
    boiteModale.fermer();
}, 3000);
```