Classe en Javascript

Introduction

Les classes en JavaScript sont une manière de créer des objets. Elles sont introduites dans ECMAScript 6 (ES6) et sont une manière plus simple et plus claire de créer des objets et de gérer l'héritage. Voyez les classes comme des fonctions constructeurs améliorées ou comme un plan d'architecture pour créer des maisons.

Derrière les classes en JavaScript, il y a toujours des prototypes, c'est simplement une manière de les utiliser qui ressemble à la programmation orientée objet classique dans d'autres langages de programmation.

Héritage

Comme vu dans le cours sur les prototypes, les objets en JavaScript sont basés sur un système de prototype. Les classes en JavaScript utilisent ce système de prototype pour l'héritage. L'héritage c'est de pouvoir partager des propriétés et des méthodes entre des objets afin de réutiliser du code.

Syntaxe

Pour créer une classe en JavaScript, on utilise le mot-clé `class` suivi du nom de la classe. À la différence des fonctions constructeurs, les classes en JavaScript n'ont pas de parenthèses après le nom de la classe.

Voici un exemple de classe `Personne` :

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

    direBonjour() {
        console.log(`Bonjour, je m'appelle ${this.nom} et j'ai ${this.age} ans.`);
    }
}
```

Constructeur

Le constructeur est une méthode spéciale qui est appelée lorsqu'on crée une nouvelle instance de la classe. C'est dans le constructeur qu'on initialise les propriétés de l'objet.

Méthodes

Les méthodes sont des fonctions qui sont attachées à la classe. Elles sont appelées sur une instance de la classe. Il ne faut pas utiliser le mot-clé `function` pour définir une méthode dans une classe.

Instanciation

Une instance est un objet (concret) créé à partir d'une classe (plan). Imaginez la classe comme un plan de construction pour l'objet et l'instance comme le bâtiment construit à partir de ce plan.

Pour créer une instance de la classe, on utilise le mot-clé `new` suivi du nom de la classe et des arguments du constructeur.

```
const personne = new Personne("Jean", 30);
personne.direBonjour(); // Bonjour, je m'appelle Jean et j'ai 30 ans.
```

Utilité des classes

Les classes en JavaScript permettent de structurer le code de manière plus claire et plus lisible. Elles permettent de regrouper les propriétés et les méthodes d'un objet en un seul endroit.

Partout où vous pourriez réutiliser un système ou un groupe de fonctions, vous pourriez envisager de créer une classe.

Normes de nommage

Les classes en JavaScript suivent la même convention de nommage que les fonctions constructeurs. Elles commencent par une majuscule et utilisent la notation camelCase.

```
class MaClasse {}
```

De plus, elles sont généralement nommées avec un nom singulier qui décrit l'objet qu'elles représentent.

Finalement, généralement, on place les classes dans des fichiers séparés dans un dossier `classes` pour les distinguer des autres fichiers que l'on importe dans notre code principal au besoin.

Exemple concret

Dans cet exemple, nous allons créer une classe pour un objet dans une liste d'oeuvres d'art de notre site Web.

1. Commençons par faire un plan de notre objet. Notre objet aura les propriétés suivantes :

- `titre` : le titre de l'oeuvre
- `auteur` : l'auteur de l'oeuvre
- `annee` : la durée de l'oeuvre
- `image` : le lien vers l'oeuvre
- `description` : la description de l'oeuvre
- `conteneurHTML` : le conteneur HTML où l'oeuvre sera affichée

Au niveau des méthodes, notre objet aura les méthodes suivantes :

- `injecterHTML` : injecte le code HTML dans la page Web

2. Créons notre classe `Oeuvre` :

```
class Oeuvre {}
```

3. Ajoutons le constructeur à notre classe. Dans le constructeur, on initialise les propriétés de l'objet qui contiendront les informations de l'oeuvre.

```
constructor(titre, auteur, annee, image, description, conteneurHTML) {
        this.titre = titre;
        this.auteur = auteur;
        this.annee = annee;
        this.image = image;
        this.description = description;
        this.conteneurHTML = conteneurHTML;
        this.elementHTML;// On initialise la propriété videoHTML mais on ne la
définit pas ici
    }
```

4. Ajoutons les méthodes à notre classe. Maintenant que nous avons défini les propriétés de notre objet, nous pouvons y accéder dans les méthodes de la classe en utilisant le mot-clé `this`.

```
injecterHTML() {
        let gabarit = `
         <div class="oeuvre">
            <img src="${this.src}" alt="${this.titre}"/>
            <h2>${this.titre}</h2>
            <p>${this.description}</p>
            <small>${this.annee}</small>
            <p>Par ${this.auteur}</p>
         </div>
        `;

        this.conteneurHTML.insertAdjacentHTML("beforeend", gabarit);
        this.elementHTML = this.conteneurHTML.lastElementChild;

        this.elementHTML.addEventListener("click", function() {
            this.mettreEnFavoris();
        }.bind(this));
    }
```

```
    mettreEnFavoris() {
        this.elementHTML.classList.toggle("favoris");
    }
```

5. Créons une instance de la classe `Oeuvre` et testons les méthodes de notre objet.

```
const conteneur = document.querySelector(".conteneur");

const oeuvre = new Oeuvre(
    "La Joconde",
    "Leonardo da Vinci",
    1503,

"https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Mona_Lisa%2C_by_Leonardo_
da_Vinci%2C_from_C2RMF_retouched.jpg/800px-
Mona_Lisa%2C_by_Leonardo_da_Vinci%2C_from_C2RMF_retouched.jpg",
    "La Joconde est un tableau de l'artiste Léonard de Vinci, réalisé entre 1503 et
1506.",
    conteneur
);

video.injecterHTML();
```