codecademy

Apprendre la syntaxe JavaScript : classes

Méthodes statiques

Dans une classe JavaScript, le `static` mot clé définit une méthode statique pour une classe. Les méthodes statiques ne sont pas appelées sur des instances individuelles de la classe, mais sur la classe elle-même. Par conséquent, il s'agit généralement de méthodes générales (utilitaires).

```
chien de classe {
  constructeur ( nom ) {
    cela . _nom = nom ;
  }

  introduire ( ) {
    consoler . log ( 'C'est ' + ceci .
_name + ' !' ) ;
  }

  // Une méthode statique
  écorce statique ( ) {
    consoler . log ( 'Wouf !' ) ;
  }
}

const myDog = new Dog ( 'Buster' ) ;
monChien . introduire ( ) ;

// Appel de la méthode statique
Chien . écorce ( ) ;
```

Classer

JavaScript prend en charge le concept de *classes* comme syntaxe de création d'objets. Les classes spécifient les propriétés et méthodes partagées que les objets produits à partir de la classe auront.
Lorsqu'un objet est créé sur la base de la classe, le nouvel objet est appelé une *instance* de la classe. De nouvelles instances sont créées à l'aide du mot-`new` clé.
L'exemple de code montre une classe qui représente un `Song` . Un nouvel objet appelé `mySong` est créé en dessous et la `.play()` méthode sur la classe est appelée. Le résultat serait le texte `Song playing!` imprimé dans la console.

```
chanson de classe {
  constructeur ( ) {
    cela . titre ;
    cela . auteur ;
  }

  jouer ( ) {
    consoler . log ( 'Chanson en cours de
lecture !' ) ;
  }
}
```

```
const mySong = nouveau morceau ( ) ;
ma chanson . jouer ( ) ;
```

Constructeur de classe

Classes can have a `constructor` method. This is a special method that is called when the object is created (instantiated). Constructor methods are usually used to set initial values for the object.

```
class Song {
  constructor(title, artist) {
    this.title = title;
    this.artist = artist;
  }
}

const mySong = new Song('Bohemian
Rhapsody', 'Queen');
console.log(mySong.title);
```

Class Methods

Properties in objects are separated using commas. This is not the case when using the `class` syntax. Methods in classes do not have any separators between them.

```
class Song {
  play() {
    console.log('Playing!');
  }

  stop() {
    console.log('Stopping!');
  }
}
```

extends

JavaScript classes support the concept of inheritance — a child class can *extend* a parent class. This is accomplished by using the `extends` keyword as part of the class definition.
Child classes have access to all of the instance properties and methods of the parent class. They can add their own properties and methods in addition to those. A child class constructor calls the parent class constructor using the `super()` method.

```
// Parent class
class Media {
  constructor(info) {
    this.publishDate = info.publishDate;
    this.name = info.name;
  }
}

// Child class
class Song extends Media {
  constructor(songData) {
    super(songData);
    this.artist = songData.artist;
  }
```

```
}

const mySong = new Song({
  artist: 'Queen',
  name: 'Bohemian Rhapsody',
  publishDate: 1975
});
```