JavaScript Class Expressions

Summary: in this tutorial, you'll learn how to use JavaScript class expressions to declare new classes.

Introduction to JavaScript class expressions

Similar to functions, classes have expression forms. A class expression provides you with an alternative way to define a new class.

A class expression doesn't require an identifier after the `class` keyword. You can use a class expression in a variable declaration and pass it into a function as an argument.

For example, the following defines a class expression:

```
let Person = class {
    constructor(name) {
        this.name = name;
    }
    getName() {
        return this.name;
    }
}
```

How it works.

On the left side of the expression is the `Person` variable. It's assigned to a class expression.

The class expression starts with the keyword `class` followed by the class definition.

A class expression may have a name or not. In this example, we have an unnamed class expression.

If a class expression has a name, its name can be local to the class body.

The following creates an instance of the `Person` class expression. Its syntax is the same as if it were a class declaration.

```
let person = new Person('John Doe');
```

Like a class declaration, the type of a class expression is also a function:

```
console.log(typeof Person); // function
```

Similar to function expressions, class expressions are not hoisted. It means that you cannot create an instance of the class before defining the class expression.

First-class citizen

JavaScript classes are first-class citizens. It means that you can pass a class into a function, return it from a function, and assign it to a variable.

See the following example:

```
function factory(aClass) {
    return new aClass();
}

let greeting = factory(class {
    sayHi() { console.log('Hi'); }
});

greeting.sayHi(); // 'Hi'
```

How it works.

First, define a `factory()` function that takes a class expression as an argument and returns the instance of the class:

```
function factory(aClass) {
    return new aClass();
```

```
}
```

Second, pass an unnamed class expression to the factory() function and assign its result to the greeting variable:

```
let greeting = factory(class {
    sayHi() { console.log('Hi'); }
});
```

The class expression has a method called `sayHi()`. And the `greeting` variable is an instance of the class expression.

Third, call the `sayHi()` method on the greeting object:

```
greeting.sayHi(); // 'Hi'
```

Singleton

Singleton is a design pattern that limits the instantiation of a class to a single instance. It ensures that only one instance of a class can be created throughout the system.

Class expressions can be used to create a singleton by calling the class constructor immediately.

To do that, you use the `new` operator with a class expression and include the parentheses at the end of the class declaration as shown in the following example:

```
let app = new class {
    constructor(name) {
        this.name = name;
    }
    start() {
        console.log(`Starting the ${this.name}...`);
    }
}('Awesome App');

app.start(); // Starting the Awesome App...
```

How it works.

The following is an unnamed class expression:

```
new class {
    constructor(name) {
        this.name = name;
    }
    start() {
        console.log(`Starting the ${this.name}...`);
    }
}
```

The class has a `constructor()` that accepts an argument. It also has a method called `start()`.

The class expression evaluates to a class. Therefore, you can call its constructor immediately by placing parentheses after the expression:

```
new class {
    constructor(name) {
        this.name = name;
    }
    start() {
        console.log(`Starting the ${this.name}...`);
    }
}('Awesome App')
```

This expression returns an instance of the class expression which is assigned to the app variable.

The following calls the `start()` method on the app object:

```
app.start(); // Starting the Awesome App...
```

Summary

- ES6 provides you with an alternative way to define a new class using a class expression.

- Class expressions can be named or unnamed.
- The class expression can be used to create a singleton object.