JavaScript Factory Functions

Summary: in this tutorial, you will learn about the JavaScript factory functions which are functions that return objects.

Introduction to the factory functions in JavaScript

A factory function is a function that returns a new object. The following creates a person object named `person1`:

```
let person1 = {
  firstName: 'John',
  lastName: 'Doe',
  getFullName() {
    return this.firstName + ' ' + this.lastName;
  },
};

console.log(person1.getFullName());
```

Output:

```
John Doe
```

The `person1` object has two properties: `firstName` and `lastName`, and one method `getFullName()` that returns the full name.

Suppose that you need to create another similar object called `person2`, you can duplicate the code as follows:

```
let person2 = {
  firstName: 'Jane',
  lastName: 'Doe',
  getFullName() {
    return this.firstName + ' ' + this.lastName;
  },
};

console.log(person2.getFullName());
```

Output:

```
Jane Doe
```

In this example, the `person1` and `person2` objects have the same properties and methods.

The problem is that the more objects you want to create, the more duplicate code you have.

To avoid copying the same code all over again, you can define a function that creates the `person` object:

```
function createPerson(firstName, lastName) {
  return {
    firstName: firstName,
    lastName: lastName,
    getFullName() {
      return firstName + ' ' + lastName;
    },
  };
}
```

When a function creates and returns a new object, it is called a factory function. The `createPerson()` is a factory function because it returns a new `person` object.

The following shows how to use the `createPerson()` factory function to create two objects `person1` and `person2`:

```
function createPerson(firstName, lastName) {
  return {
    firstName: firstName,
    lastName: lastName,
    getFullName() {
      return firstName + ' ' + lastName;
    },
  };
}

let person1 = createPerson('John', 'Doe');
let person2 = createPerson('Jane', 'Doe');

console.log(person1.getFullName());
console.log(person2.getFullName());
```

By using the factory function, you create any number of the `person` objects without duplicating code.

When you create an object, the JavaScript engine allocates memory to it. If you create many `person` objects, the JavaScript engine needs lots of memory space to store these objects.

However, each `person` object has a copy of the same `getFullName()` method. It's not efficient memory management.

To avoid duplicating the same `getFullName()` function in every object, you can remove the `getFullName()` method from the `person` object:

```
function createPerson(firstName, lastName) {
    return {
        firstName: firstName,
        lastName: lastName
    }
}
```

And move this method to another object:

```
var personActions = {
  getFullName() {
    return this.firstName + ' ' + this.lastName;
  },
};
```

And before calling the `getFullName()` method on the `person` object, you can assign the method of the `personActions` object to the `person` object as follows:

```
let person1 = createPerson('John', 'Doe');
let person2 = createPerson('Jane', 'Doe');

person1.getFullName = personActions.getFullName;
person2.getFullName = personActions.getFullName;

console.log(person1.getFullName());
console.log(person2.getFullName());
```

This approach is not scalable if the object has many methods because you have to manually assign them individually. This is why the `Object.create()` method comes into play.

The Object.create() method

The `Object.create()` method creates a new object using an existing object as the prototype of the new object:

```
Object.create(proto, [propertiesObject])
```

So you can use the `Object.create()` as follows:

```
var personActions = {
  getFullName() {
    return this.firstName + ' ' + this.lastName;
  },
};

function createPerson(firstName, lastName) {
  let person = Object.create(personActions);
  person.firstName = firstName;
  person.lastName = lastName;
  return person;
}
```

Now, you can create `person` objects and call the methods of the `personActions` object:

```
let person1 = createPerson('John', 'Doe');
let person2 = createPerson('Jane', 'Doe');

console.log(person1.getFullName());
console.log(person2.getFullName());
```

The code works perfectly fine. However, in practice, you will rarely use the factory functions. Instead, you use classes or constructor/prototype patterns.

Summary

- A factory function is a function that returns a new object.
- Use `Object.create()` to create an object using an existing object as a prototype.