Class Inheritance

In this lesson, we discuss the syntax and concept of class inheritance in Javascript.

Inheritance in JavaScript

Usually, classes are used for inheritance in object-oriented programming. Here is a brief refresher on inheritance; inheritance enables new classes to take on the properties and methods of existing classes. A class that another class inherits is called a *superclass* or *base class*. A class that inherits from a superclass is called a *subclass* or *derived class*. In JavaScript, the `extends` keyword can be used to inherit one class from another class as shown in the following example. So, the derived class, `ReactDeveloper`, inherits all the abilities from the base class, `Developer`, and also adds its specialized capabilities, namely the `getJob()` method to it.

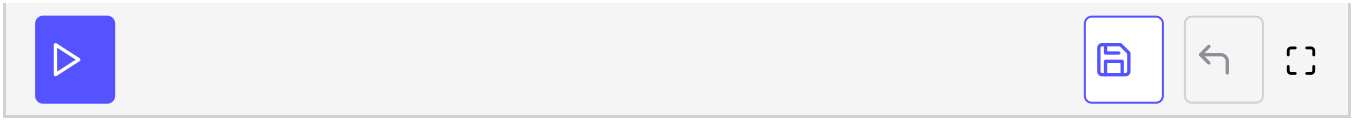
```
class Developer {
  constructor(firstname, lastname) {
    this.firstname = firstname;
    this.lastname = lastname;
  }

  getName() {
    return this.firstname + ' ' + this.lastname;
  }
}

class ReactDeveloper extends Developer {
  getJob() {
    return 'React Developer';
  }
}

var me = new ReactDeveloper('Robin', 'Wieruch');

console.log(me.getName());
console.log(me.getJob());
```



However, as you will progress in your React journey, you will realize that

inheritance is somewhat limited compared to class composition. We will see how that is in the next lesson.