Inheritance in ES6

This lesson introduces the keywords extends and super and explains their use in implementing inheritance in JavaScript.

WE'LL COVER THE FOLLOWING

In the previous chapter, the implementation of class in the ES6 version of JavaScript was discussed. The version introduced the *keyword* `class` used to define a class.

Now let's learn how to implement inheritance in the ES6 version of JavaScript.

Syntax

The underlying concept regarding the implementation of *inheritance* is the same in both versions ES5 and ES6; however, the latter provides a cleaner syntax as well as some new keywords that can be used during the implementation.

The ES6 version uses the *keyword* `extends` which creates a class that is the child of another class.

The following is the syntax used in ES6:

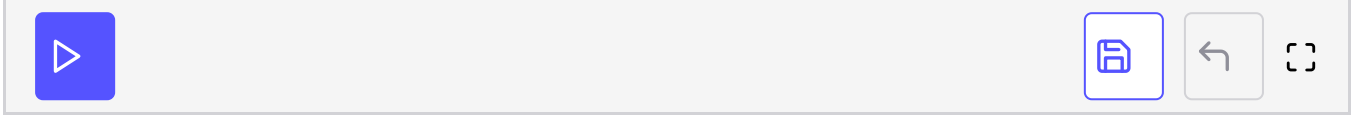
```
class childClassName extends parentClassName{
    //body containing constructor and methods
}
```

First, the child class to be made is written followed by the `extends` *keyword*, then the name of the parent class, i.e., the class upon which the child class is to be based.

Example

Let's take a look at an example to see how inheritance in ES6 is implemented:

```
//Running this code will give an error
//creating a class named employee
class Shape{
  //creating the constructor function
  constructor(shapeName,shapeSides){
    //all properties defined as they were in the constructor function
    this.name = shapeName
    this.sides = shapeSides
  }
  displayName(){
    console.log("Name is:",this.name)
  }
}
//creating a child class Rectangle that inherits from the Shape class
class Rectangle extends Shape{
  constructor(shapeLength,shapeWidth){
    this.length = shapeLength
    this.width = shapeWidth
  }
  getArea(){
    return this.length*this.width
  }
}
```



When the above code is run, the error: `'this' is not allowed before super()` is shown. What does it mean?

`super` Keyword

The use of the keyword `new` initializes `this` so that it can be used to assign values to properties. But when the *keyword* `extends` is used to create child classes, this initialization doesn't happen, which is why the code above returns an error.

As the child class is derived from the parent class, the initialization of `this`

for the child class is done by the parent class constructor. Hence, the parent

class constructor needs to be called inside the child class constructor, which is done by using `super()`.

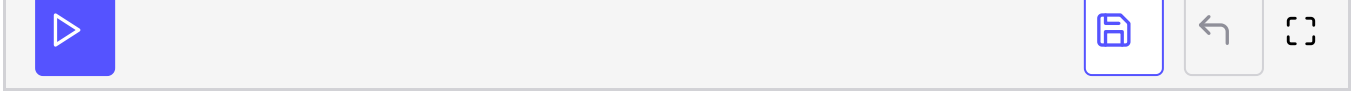
`super()` acts as the parent class constructor; the parent class constructor parameters can be passed into it. If `super()` is called with the parameters inside the child class constructor, the parent class properties will get initialized for the child class, i.e., it will inherit the parent class properties.

Example

Let's modify the code above by calling `super()` in it now.

> Note: As explained, `super()` needs to be called before `this` keyword is used inside the child class constructor.

```
//creating a class named employee
class Shape{
  //creating the constructor function
  constructor(shapeName,shapeSides){
    //all properties defined as they were in the constructor function
    this.name = shapeName
    this.sides = shapeSides
  }
}
//creating a child class Rectangle that inherits from Shape class
class Rectangle extends Shape{
  constructor(shapeName,shapeSides,shapeLength,shapeWidth){
    //initializing "this" before it is used
    //Shape class properties getting initialized for Rectangle
    //Shape class properties getting inherited by Rectangle class
    super(shapeName,shapeSides)
    //Rectangle properties defined
    this.length = shapeLength
    this.width = shapeWidth
  }
  getArea(){
    return this.length*this.width
  }
}
//creating object of the Rectangle class
var rec = new Rectangle('Rectangle',4,3,5)
//accessing properties and methods in "rec"
console.log("Name:",rec.name)
console.log("Sides:",rec.sides)
console.log("Length",rec.length)
console.log("Width",rec.width)
console.log("Area",rec.getArea())
```



Explanation

As seen in the code above:

- `Rectangle` class is the child class derived from the parent class `Shape` using `extends`.
- The constructor for `Rectangle` is passed all the parameters including those of the `Shape` class.
- Calling `super()` initializes `this` in `Rectangle`. It acts as the `Shape` class constructor. Thus, passing it its parameters initializes the properties in `Shape` for `Rectangle` which thereby inherits those properties.
- Since properties in `Shape` are inherited by `Rectangle`, the `name` and `sides` properties will exist for `rec` object created. Hence their values are displayed upon access.

In the next lesson, let's discuss how to override methods and properties in classes in the ES5 and ES6 versions of JavaScript.