# Inheritance - The ES6 way

inheritance in ES6, accessing constructors and other class methods at various places in the code, introduction to concise method syntax and Javascript code conventions

Let's see the ES6 version of inheritance. As we'll see later, the two versions are not equivalent, we just describe the same problem domain with ES6 code.

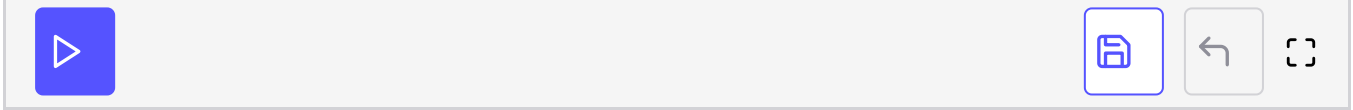
```
class Shape {
    constructor( color ) {
        this.color = color;
    }

    getColor() {
        return this.color;
    }
}

class Rectangle extends Shape {
    constructor( color, width, height ) {
        super( color );
        this.width = width;
        this.height = height;
    }

    getArea() {
        return this.width * this.height;
    }
}

let rectangle = new Rectangle( 'red', 5, 8 );
console.log( "Area:\t\t" + rectangle.getArea() );
console.log( "Color:\t\t" + rectangle.getColor() );
console.log( "toString:\t" + rectangle.toString() );
```



Classes may encapsulate

- a constructor function
- additional operations extending the prototype
- reference to the parent prototype.

Notice the following:

- The `extends` keyword defines the is-a relationship between `Shape` and `Rectangle`. All instances of `Rectangle` are also instances of `Shape`.
- The `constructor` method runs when you instantiate a class. You can call the constructor method of your parent class with `super` (more on `super` later).
- Methods can be defined inside classes. All objects can call methods of their class and all classes that are higher in the inheritance chain.
- Instantiation works in the same way as the instantiation of an ES5 constructor function.
- The methods are written using the *concise method syntax*. We will learn about this syntax in depth in Chapter 7 - Objects.

You can observe the equivalent ES5 code by pasting the above code into the BabelJs online editor.

The reason why the generated code is not equivalent with the ES5 code we studied is that the ES6 class syntax comes with additional features. You will never need the protection provided by these features during regular use. For instance, if you call the class name as a regular function, or you call a method of the class with the `new` operator as a constructor, you get an error.

> **Convention:** Your code becomes more readable, when you capitalize class names, and start object names and method names with a lower case letter. For instance, `Person` should be a class, and `person` should be an object.

Let's talk about the `super` keyword in the next lesson.