Implementing Classes

This lesson introduces the ES6 version of classes, their syntax, and teaches us how to create new object instances using a class.

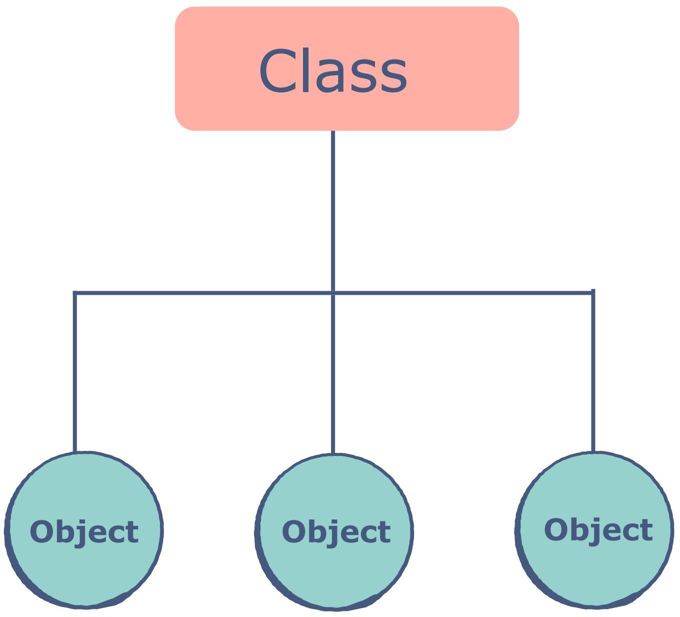
WE'LL COVER THE FOLLOWING

As discussed in the first chapter, the ES6 version of JavaScript offers some new features as well as improvements. One of those improvements was the introduction of the *keyword* `class`.

In the previous chapter, we discussed *constructor functions*, which are used in the ES5 version to implement the concept of classes. However, in the ES6 version, the `class` keyword is used, which cleans up the syntax for implementing the same concept, thus making it easier to understand.

What Is a Class?

We discussed that there is no concept of classes in the ES5 version. What we needed was a blueprint containing all properties and methods which could then be inherited by the object instances created from that blueprint. This saves us the time and effort to create separate object literals. The ES6 version introduces the `class` *keyword* to create classes which are used to implement this functionality. The classes created then become the blueprints from which the objects inherit their properties.



Multiple Objects created from a class

Declaring a Class

We know that the ES6 version uses the `class` *keyword* but how do we declare a class? Let’s take a look at it below.

Syntax

The following is the syntax for declaring a class:

```
class ClassName {
  constructor() {
    //initializing class properties
  }
  //class methods defined
}
```

Class Syntax

In a class-based implementation, the *keyword* `class` is used followed by the *class name*. One of the differences between the *constructor function* and *class-based* implementation is that in the former, the body of the function acts as the *constructor* where all properties are defined, whereas in the latter, there is a separate `constructor` function defined inside the class used to initialize the properties.

Example

Let's take the `Employee` *constructor* function from the previous chapter and convert it into a class below:

```
//creating a class named employee
class employee{
  //creating the constructor function
  constructor(name,age,designation){
    //all properties defined as they were in the constructor function
    this.name = name
    this.age = age
    this.designation = designation
    this.displayName = function() {
      console.log("Name is:",this.name)
    }
  }
}
//displaying type of "employee"
console.log(typeof employee)
```

Explanation

When the above code is run, it displays the type of `employee` constructer as a *function*.

What the code above does is:

- Creates a *function* constructor named `employee` when the class is declared.
- The `constructor` provides the code for this constructor function. The `constructor` is used to initialize the properties as it did in the ES5 version implementation. Hence, even though the syntax has changed, the underlying concept is the same.

> **Note:** There are no commas used between the properties inside the constructor. Using them will result in an error.

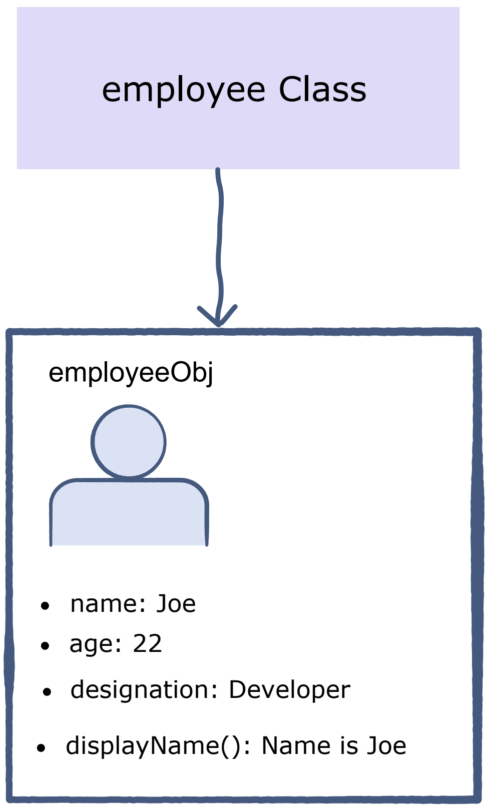
Creating an Object Instance

How do we create *new* object instances from a class? Let's take a look at it in the example below:

```
//creating a class named employee
```

```
class employee{
  //creating the constructor function
  constructor(name,age,designation){

    //all properties defined as they were in the constructor function
    this.name = name
    this.age = age
    this.designation = designation
    this.displayName = function() {
      console.log("Name is:",this.name)
    }
  }
}
//creating an object instance named "employeeObj"
var employeeObj = new employee('Joe',22,'Developer')
//displaying the properties of employeeObj
employeeObj.displayName()
console.log("Age is",employeeObj.age)
console.log("Designation is:",employeeObj.designation)
```



employeeObj created from the class "employee"

Explanation

Since `employee` is a *constructor* function itself, the method to create an object instance from a class is exactly the same as that in the ES5 version. The `new` *keyword* is used to initialize a new object, `employeeObj`. The `constructor` method then runs for this object, assigning the values passed into it to the properties.

In this lesson, methods were declared inside the constructor in the class. However, methods can also be defined inside the class itself. Let’s learn how to do that in the next lesson!