Prototype-based Inheritance

```
// defining a constructor calling a super class
function Employee(name, title){
    this.title = title;
    Person.call(this, name);
}

// setting up the inheritance
Employee.prototype = new Person();

// fixing the constructor
Employee.prototype.constructor = Employee;

// creating an object
const e = new Employee('Mariam','CEO');
console.log(e.getName());
console.log(e.title);
console.log(e.constructor.name);
console.log(e instanceof Employee);
console.log(e instanceof Person);
```

Class-based Object-Oriented

```
// Defining a class
class Person(){

  constructor(name){
    this.name = name;
  }

  get name(){
    return this.name;
  }

}

// creating an object
const p = new Person('Mariam');
console.log(p.name());
```