Prototype-Based Object-Oriented

```
// defining a constructor
function Person(name){
    this.name = name;
}

// adding a method
Person.prototype.getName = function(){
    return(this.name);
};

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

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);
```