Object Oriented Programming in JavaScript

Diyar Parwana

There are some features that make a programming language Object-Oriented.

1- Classes

2- Objects

3- Encapsulation

4- Inheritance

1- Classes

Example: 1

```
<script type="text/javascript">
  class Rectangle {
    constructor(height, width) {
      this.height = height;
      this.width = width;
    }


    // Getter
    get area() {
      return this.calculateArea();
    }


    // Method
    calculateArea() {
      return this.height * this.width;
    }
  }

  const square = new Rectangle(20, 20);

  alert(square.area);
  console.log(square.area);
</script>
```

Example: 2

```
<script type="text/javascript">

  class Bil {
    constructor(name, regnr, engine) {
      this.name = name;
      this.regnr =  regnr;
      this.engine = engine;
    }

    getBilDetails(){
        return (`Name: ${this.name} && Regnr: ${this.regnr} `)
    }
  }


  // Making objects with the help of the constructor
  let obj1 = new Bil('Volvo', 'RPXO99');
  let obj2 = new Bil('Sab', 'LPI900');

  console.log( "Object 1 "+ obj1.name + obj1.regnr);    // Volvo RPX099

  console.log( "Object 2 "+ obj2.name + obj2.regnr);    // Sab LPI900

// Or we can use the getBilDetails to the display the the name for us...
  console.log(obj1.getBilDetails());
  console.log(obj2.getBilDetails());

</script>
```

2- Objects

Example: 1

```
<script type="text/javascript">

  //Defining object
  let person = {
      firstName:'Adam',
      lastName: 'Testsson',

      //Object method
      getFunction : function(){
          return (`Name: ${person.firstName} ${person.lastName}`)
      },


      //Creating another object within an object
      phoneNumber : {
          mobile:'076000000',
          country:'SE'
      }
  }


  console.log(person.getFunction());
  console.log("Phone: "  + person.phoneNumber.mobile + "\n" + "Country: "  +
person.phoneNumber.country);

</script>
```

Example: 2

```
<script type="text/javascript">

  //using a constructor
  function person(FirstName,LastName){
     this.FirstName = FirstName;
     this.LastName = LastName;
  }

  //Creating new instances of the person object
  let obj1 = new person('Alpha','Testsson');
  let obj2 = new person('Beta','Testsson');
  let obj3 = new person('Charlie','Testsson');


  console.log(obj1.FirstName); // Alpha
  console.log(`${obj2.FirstName} ${obj2.LastName}`); // Beta Testsson
</script>
```

Example: 3

```
<script type="text/javascript">
  // How to use Object.create()???
  // An object with  properties
  const person = {
      isStudent : false,
      displayIntro : function(){
          console.log(` This person is ${this.name}. \n Is he a student? : ${this.isStudent}.`)
      }
  }

  // Object.create() method
  const obj1 = Object.create(person);

  // "name" is a property set on "obj1", but not on "person"
  obj1.name = 'Testsson';
  // Inherited properties can be overwrittens
  obj1.isStudent = true;
  obj1.displayIntro();

// This person is Testsson.
// Is he a student? : true.
</script>
```

3- Encapsulation

Encapsulation – Having the properties and functions within a single class is known as encapsulation.

Example: 1

```
<script type="text/javascript">
  class student{
      constructor(name,id){
          this.name = name;
          this.id = id;
      }

      setAdress(add){
          this.add = add;

      }

      setAge(age){
          this.age = age;

      }

      getDetails(){
          console.log(`Name: ${this.name}  \n Id:${this.id}  \n Age: ${this.age} \n Address:
${this.add}`);
      }
  }

  let student1 = new student('Testsson',1000);
  student1.setAdress('Storgatan 1');
    student1.setAge(35);
  student1.getDetails();

  /*
  Name: Testsson
   Id:1000
   Age: 35
   Address: Storgatan 1
  */

</script>
```

4- Inheritance

Example: 1

```
<script>
class Car {
  constructor(brand) {
    this.Name = brand;
  }

  present() {
    return 'My car is  ' + this.Name;
  }

}

// Using the "extends" keyword to inherit all methods from another class
class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', and it is ' + this.model;
  }
}

let myCar = new Model("VS10", "Volvo");
console.log(myCar.show());

//My car is  VS10, and it is Volvo
</script>
```