OOPs in Java script

There are certain features or mechanisms which makes a Language Object-Oriented like:

- **Object**
- **Classes**
- **Encapsulation**
- **Inheritance**

1. **Object**– An Object is a **unique** entity that contains **property** and **methods**.

- For example “car” is a real life Object, which has some characteristics like color, type, model, horsepower and performs certain action like drive.
- The characteristics of an Object are called as Property, in Object-Oriented Programming and the actions are called methods.
- An Object is an **instance** of a class.

Ex:

```
let person = {
    first_name:’VidyaSagar',
    last_name: ‘Vasireddy',

    //method
    getFunction : function(){
        return (`The name of the person is ${person.first_name}
                    ${person.last_name}`);
    },
    //object within object
    phone_number : {
        mobile:'12345',
        landline:'6789'
    }
}
console.log(person.getFunction());
console.log(person.phone_number.landline);
```

Classes are **blueprint** of an Object.

A class can have many Object, because class is a **template** while Object are **instances** of the class or the concrete implementation.

Before we move further into implementation, we should know unlike other Object Oriented Language there is **no classes in JavaScript** we have only Object.

To be more precise, JavaScript is a prototype based object oriented language, which means it doesn’t have classes rather it define behaviors using constructor function and then reuse it using the prototype.

```
class Vehicle {
    constructor(name, maker, engine) {
    this.name = name;
    this.maker =  maker;
    this.engine = engine;
  }
  getDetails(){
      return (`The name of the bike is ${this.name}.`)
  }
}
// Making object with the help of the constructor
let bike1 = new Vehicle(‘Maruti', ‘Baleno', ‘1200cc');
let bike2 = new Vehicle(‘Kia’, ‘Seltos', ‘1400cc');

console.log(bike1.name);    // Hayabusa
console.log(bike2.maker);   // Kawasaki
console.log(bike1.getDetails());
```

3. Encapsulation – The process of **wrapping property and function** within a **single unit** is known as encapsulation.

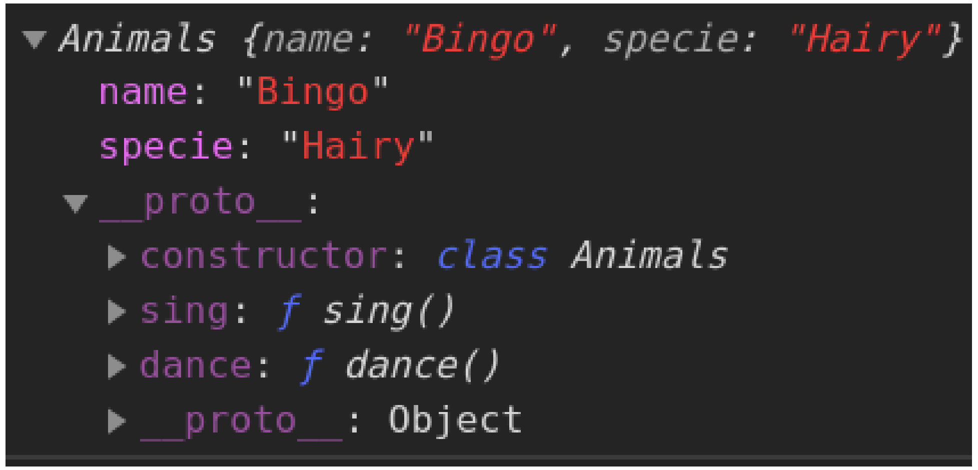
4. Inheritance – It is a concept in which some property and methods of an Object is being used by another Object.

Unlike most of the OOP languages where classes inherit classes, JavaScript Object inherits Object i.e. certain features (property and methods)of one object can be reused by other Objects.

```
class Animals {
        constructor(name, specie) {
                this.name = name;
                this.specie = specie;
        }
        sing() {
                return `${this.name} can sing`;
        }
        dance() {
                return `${this.name} can dance`;
        }
}

let bingo = new Animals("Bingo", "Hairy");
console.log(bingo);
```

This is the result in the console:



```
▼Animals {name: "Bingo", specie: "Hairy"}
   name: "Bingo"
   specie: "Hairy"
 ▼__proto__:
   ▶constructor: class Animals
   ▶sing: ƒ sing()
   ▶dance: ƒ dance()
   ▶__proto__: Object
```

The __proto__ references the Animals prototype (which in turn references the Object prototype).

From this, we can see that the constructor defines the major features while everything outside the constructor (sing() and dance()) are the bonus features (**prototypes**).

In the background, using the new keyword approach, the above translates to:

```
function Animals(name, specie) {
        this.name = name; this.specie = specie;
}
Animals.prototype.sing = function(){
        return `${this.name} can sing`;
}
Animals.prototype.dance = function() {
        return `${this.name} can dance`;
}
let Bingo = new Animals("Bingo", "Hairy");
```

```
Ex: class Animal {
    constructor(legs) {
        this.legs = legs;
    }
    walk() {
        console.log('walking on ' + this.legs + ' legs');
    }
}

class Bird extends Animal {
    constructor(legs) {
        super(legs);
    }
    fly() {
        console.log('flying');
    }
}


let bird = new Bird(2);
bird.walk();
bird.fly();
```

Output:

```
"walking on 2 legs"
"flying"
```