**Navachethan.M – 1NT18IS099**

Obj\_class.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>class and objects</h1>

    <label for="input1"> Name:</label>

        <input type="text" id="i1">

        <label for="input2"> Age:</label>

        <input type="text" id="i2">

        <input type="button" value="Result" onclick="myfun()"/>

        <p id="ans1"></p>

        <p id="ans2"></p></br>

        <label for="input3"> Name:</label>

        <input type="text" id="i3">

        <label for="input4"> Brand:</label>

        <input type="text" id="i4">

        <label for="input5">Feature:</label>

        <input type="text" id="i5">

        <input type="button" value="Result" onclick="myfun2()"/>

        <p id="ans3"></p>

        <p id="ans4"></p></br>

    <script src="ob\_class.js"></script>

</body>

</html>

Ob\_class.js:

let names = {

    fname: "Bruce",

    lname: "Wayne"

}

console.log(names.fname);

console.log(names.hasOwnProperty("mname"));

console.log(names);

//Object.create

function LionObject(name, age) {

    let Lion = Object.create(constructorObject);

    Lion.name = name;

    Lion.age = age;

    return Lion;

}

let constructorObject = {

    speak: function(){

        return "Lion is the King"

    }

}

let leo = LionObject("Mufasa", 10);

console.log(leo);

console.log(constructorObject.speak());

//Usage of new keyword

function LionObject1(name, age) {

    this.name = name;

    this.age = age;

}

LionObject1.prototype.speak = function() {

    return "Prince simba";

}

let john = new LionObject1("Simba", 3);

console.log(LionObject1.prototype.speak());

/\*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).\*/

class Drive {

    constructor(name, brand) {

        this.name = name;

        this.brand = brand;

    }

    speed() {

        return `${this.name} has speed`;

    }

    luxury() {

        return `${this.name} is luxurious`;

    }

}

let car = new Drive("Duster", "Renault");

console.log(car.speed());

console.log(car.luxury());

function Drive1(name, brand) {

    this.name = name;

    this.brand = brand;

}

Drive1.prototype.speed = function(){

    return `${this.name} has speed`;

}

Drive1.prototype.luxury = function() {

    return `${this.name} is luxurious `;

}

let car1 = new Drive1("X3", "BMW");

console.log(car1.speed());

console.log(car1.luxury());

/\*Let's see how subclasses are done with class.

Here, we need a parent which the subclass inherits from.

Examine the following code:\*/

class Drive2 {

    constructor(name, brand) {

        this.name = name;

        this.brand = brand;

    }

    speed() {

        return `${this.name} has speed`;

    }

    luxury() {

        return `${this.name} is luxurious`;

    }

}

class tesla extends Drive2 {

    constructor(name, brand, speciality) {

        super(name, brand);

        this.speciality = speciality;

    }

    feature() {

        return ` ${this.speciality} is an added advantage`;

    }

}

let wheels = new tesla("Model-S", "Tesla", "selfdrive");

console.log(wheels.speed());

console.log(wheels.feature());

//responsive method

function myfun(){

    function LionObject(name, age) {

        let Lion = Object.create(constructorObject);

        Lion.name = name;

        Lion.age = age;

        return Lion;

    }

    let constructorObject = {

        speak: function(){

            return "Lion is the King"

        }

    }

    let n="";

    let a="";

    let leo = LionObject(n=document.getElementById("i1").value,

    a = Number(document.getElementById("i2").value));

    document.getElementById("ans1").innerHTML = leo.name+"-"+leo.age;

    document.getElementById("ans2").innerHTML = constructorObject.speak();

    //console.log(leo);

    //alert(leo.name+" "+leo.age);

    //alert(constructorObject.speak());

    //console.log(constructorObject.speak());

}

function myfun2(){

    class Drive2 {

        constructor(name, brand) {

            this.name = name;

            this.brand = brand;

        }

        speed() {

            return `${this.name} has speed`;

        }

        luxury() {

            return `${this.brand} is luxurious`;

        }

    }

    class tesla extends Drive2 {

        constructor(name, brand, speciality) {

            super(name, brand);

            this.speciality = speciality;

        }

        feature() {

            return ` ${this.speciality} is an added advantage`;

        }

    }

    let n="";

    let b="";

    let f="";

    let wheels = new tesla(n=document.getElementById("i3").value,

    b=document.getElementById("i4").value,

    f=document.getElementById("i5").value,);

    //alert(wheels.speed());

    document.getElementById("ans3").innerHTML = wheels.speed()+"-"+wheels.luxury();

    document.getElementById("ans4").innerHTML = wheels.feature();

    //console.log(wheels.speed());

    //console.log(wheels.feature());

}

// trial example

/\*function myfun3(){

    function LionObject(name, age) {

        let Lion = Object.create(constructorObject);

        Lion.name = name;

        Lion.age = age;

        return Lion;

    }

    let constructorObject = {

        speak: function(){

            return "Lion is the King"

        }

    }

    let n= document.getElementById("i6").value;

    let a=Number(document.getElementById("i7").value);

    let leo = LionObject(n,a);

    //document.getElementById("ans5").innerHTML = leo.name+"-"+leo.age;

    //document.getElementById("ans6").innerHTML = constructorObject.speak();

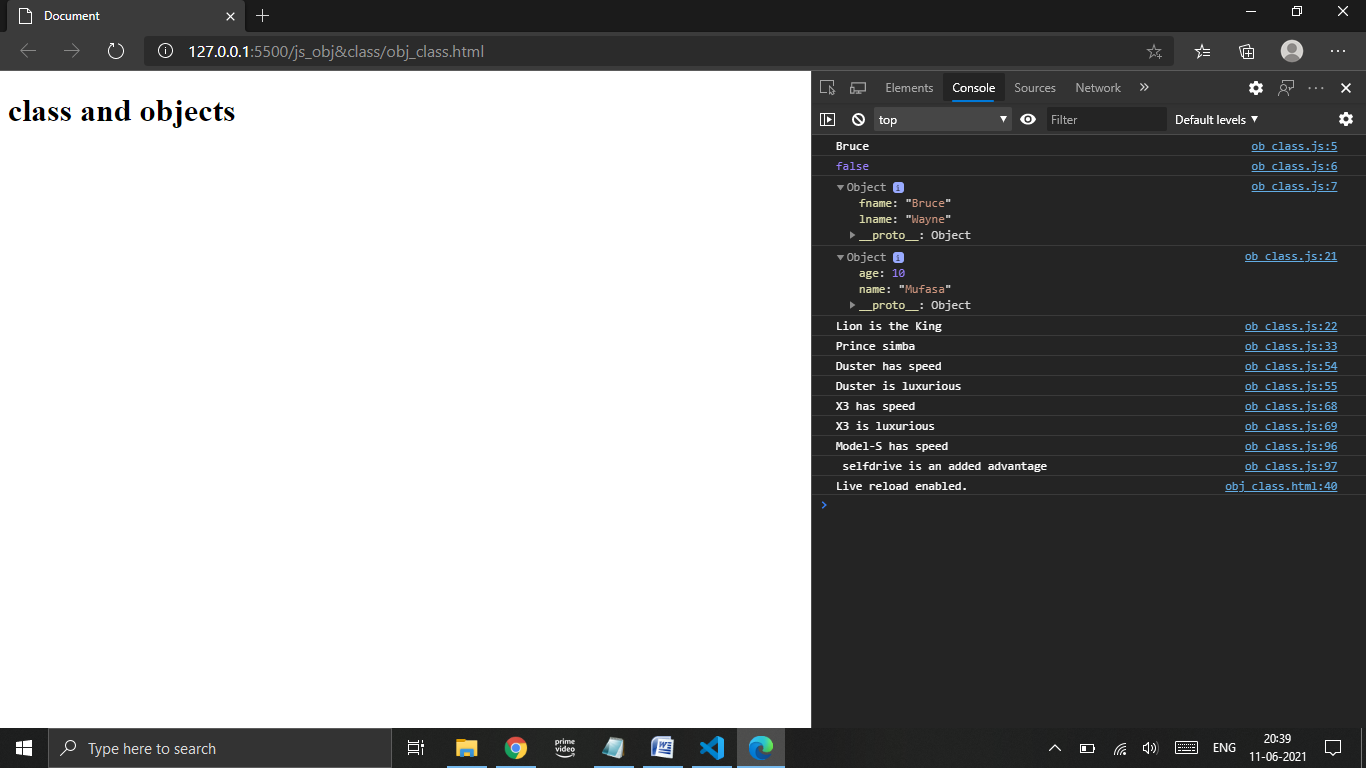
    //console.log(leo);

    alert(leo);

    //alert(constructorObject.speak());

    //console.log(constructorObject.speak());

} \*/

Output:

