JavaScript Session 3

Object Oriented Programming (OOPs)

OOP is a programming paradigm that believes in grouping data (properties) and methods (actions) together inside a box. It demonstrates the pattern of real-world objects.

Note: JavaScript is not an object-oriented language. Neither is it completely a functional language. JavaScript is a **prototype-based procedural language**. It supports both functional and object-oriented patterns of programming.

There are two types of OOP languages:

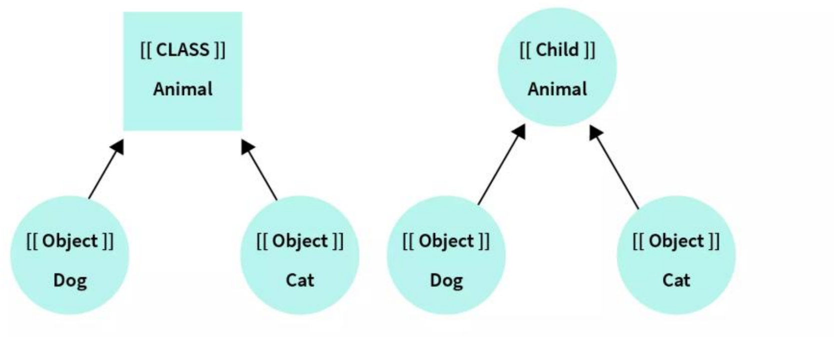
1. Class-Based languages like JAVA, C++.

2. Prototype-Based languages like JavaScript.

Is JavaScript Object-Oriented?

We need to understand the difference between OOP and Prototype-based programming, before finding the answer to this common question - **' Is JavaScript Object-oriented?'**.

- **Object-Oriented Programming (OOP)** The object-oriented paradigm keeps data and actions grouped together inside classes. In OOP, we create classes and then create their instances called objects.
- **Prototype-based Programming** In Prototype-based programming, we derive objects from other already existing objects.

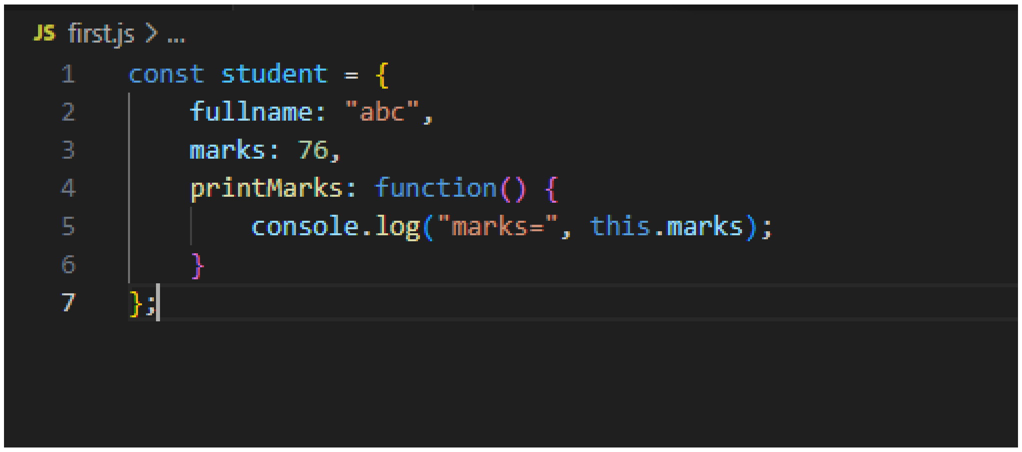


Object:

An Object is a unique entity that contains properties and methods.

The characteristics of an Object are called Properties in Object-Oriented Programming and the actions are called methods. An Object is an instance of a class. Objects are everywhere in JavaScript, almost every element is an Object whether it is a function, array, or string.

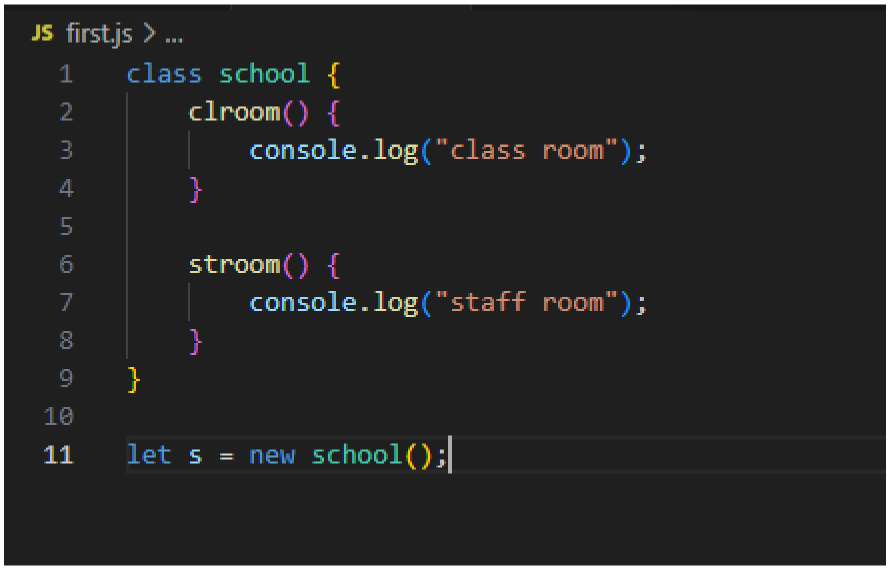
Example:



first.js > ...

```javascript
const student = {
    fullname: "abc",
    marks: 76,
    printMarks: function() {
        console.log("marks=", this.marks);
    }
};
```

(OR)



first.js > ...

```javascript
class school {
    clroom() {
        console.log("class room");
    }

    stroom() {
        console.log("staff room");
    }
}

let s = new school();
```

In this s is object .

Classes in JS

Those objects will have some state (variables) & some behaviour (functions) inside it.

Class is a program-code template for creating objects.

Syntax

class MyClass {

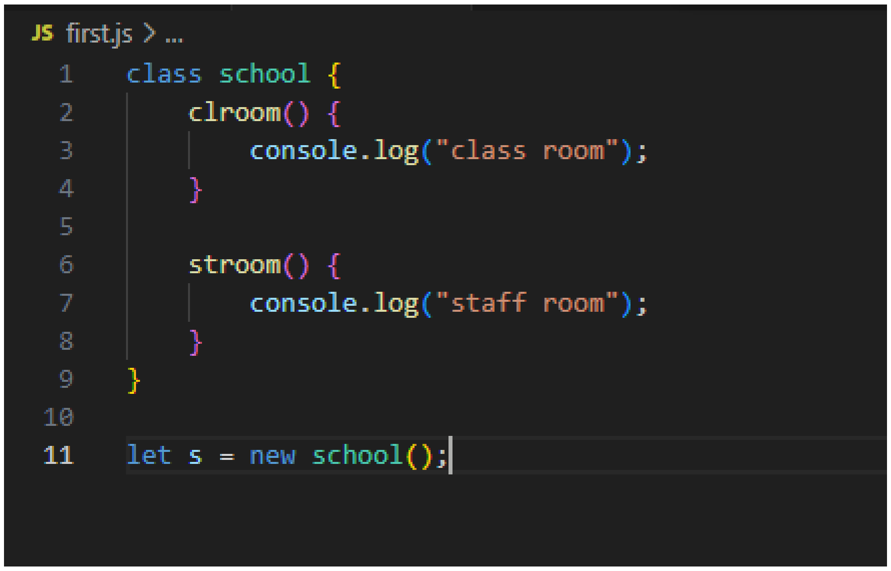
constructor( ) { ... }

myMethod( ) { ... }

}

let myObj = new MyClass( ) ;

Example:



```
class school {
    clroom() {
        console.log("class room");
    }

    stroom() {
        console.log("staff room");
    }
}

let s = new school();
```

Constructor( ) method is :

automatically invoked by new

initializes object

syntax

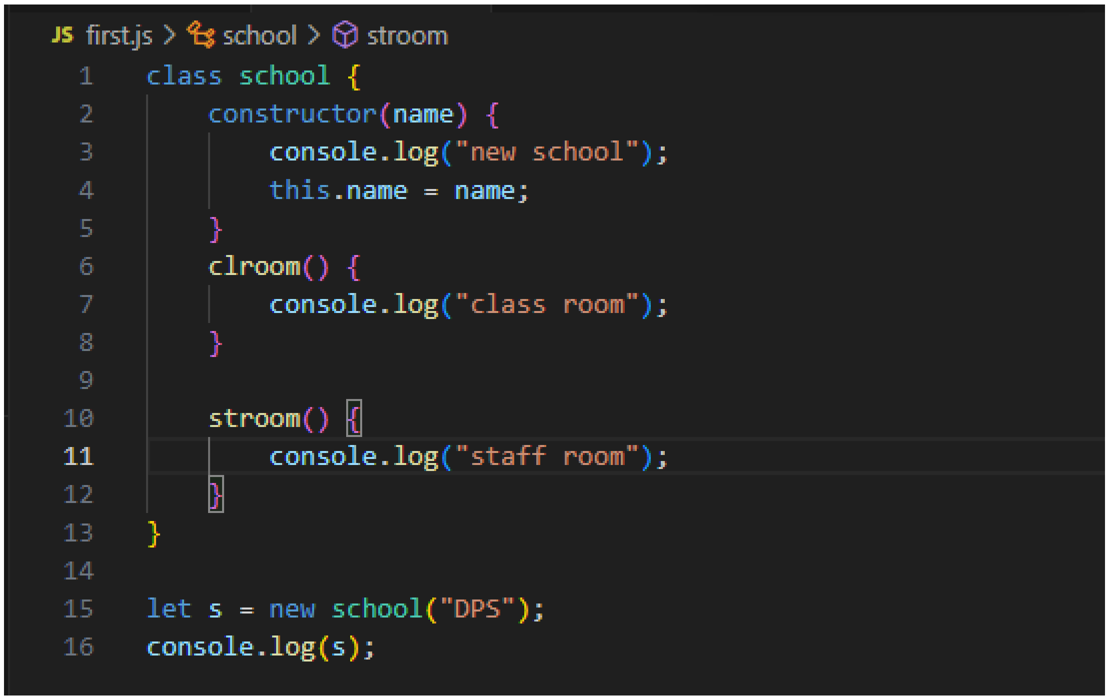
class MyClass {

constructor( ) { ... }

myMethod( ) { ... }

}

Example:



JS first.js > school > stroom

```
class school {
    constructor(name) {
        console.log("new school");
        this.name = name;
    }
    clroom() {
        console.log("class room");
    }

    stroom() {
        console.log("staff room");
    }
}

let s = new school("DPS");
console.log(s);
```

Inheritance in JS

inheritance is passing down properties & methods from parent class to child class.

syntax

class Parent {

}

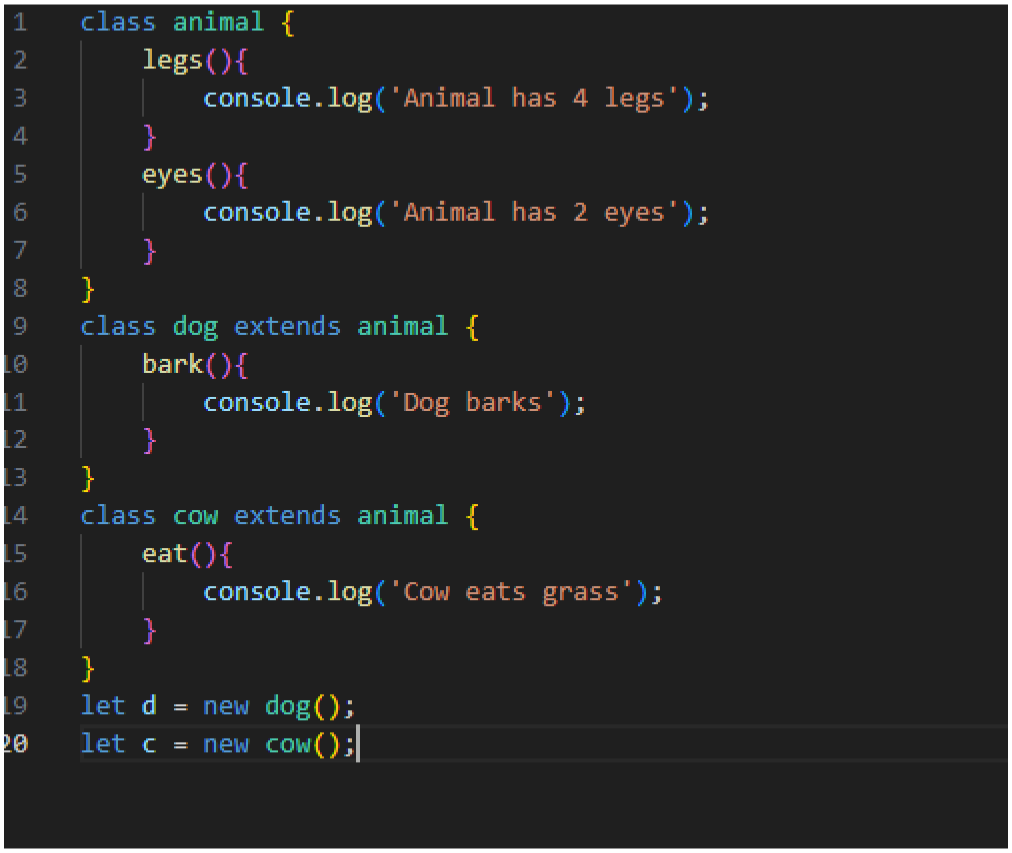
class Child extends Parent {

}

*If Child & Parent have same method, child's

method will be used. [Method Overriding]

Example:

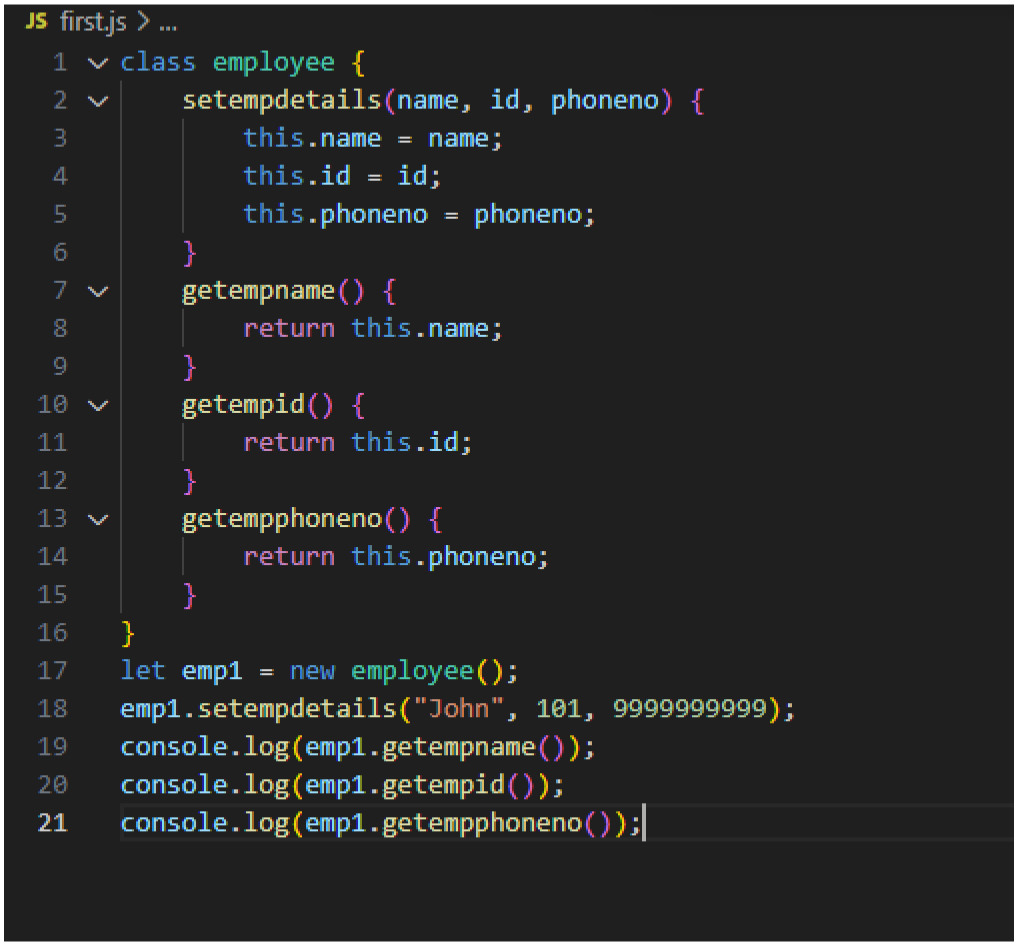


```
class animal {
    legs(){
        console.log('Animal has 4 legs');
    }
    eyes(){
        console.log('Animal has 2 eyes');
    }
}
class dog extends animal {
    bark(){
        console.log('Dog barks');
    }
}
class cow extends animal {
    eat(){
        console.log('Cow eats grass');
    }
}
let d = new dog();
let c = new cow();
```

Encapuslation :

Encapsulation is a fundamental concept in object-oriented programming that refers to the practice of hiding the internal details of an object and exposing only the necessary information to the outside world.

Example:



first.js > ...

```
class employee {
    setempdetails(name, id, phoneno) {
        this.name = name;
        this.id = id;
        this.phoneno = phoneno;
    }
    getempname() {
        return this.name;
    }
    getempid() {
        return this.id;
    }
    getempphoneno() {
        return this.phoneno;
    }
}
let emp1 = new employee();
emp1.setempdetails("John", 101, 9999999999);
console.log(emp1.getempname());
console.log(emp1.getempid());
console.log(emp1.getempphoneno());
```