Create object using

1. Object Literal

const emp1 = {

    id:1,

    empname:"Ajay",

    get: ()=>

{

     console.log("ID is " + emp1.id);

     console.log("Name is " + emp1.empname);

}

}

const emp2 = {

    id:2,

    empname:"Deepak",

    get: ()=>

{

     console.log("ID is " + emp2.id);

     console.log("Name is " + emp2.empname);

}

}

  emp1.get();

  emp2.get();

Here we can add more properties or methods

1. // Defining class in a Traditional Way.
2. function Vehicle(name,maker,engine){
3. this.name = name,
4. this.maker = maker,
5. this.engine = engine
7. };
9. Vehicle.prototype.getDetails = function(){
10. console.log('The name of the bike is '+ this.name);
11. }
13. let bike1 = new Vehicle('Hayabusa','Suzuki','1340cc');
14. let bike2 = new Vehicle('Ninja','Kawasaki','998cc');
15. bike1.get();
16. console.log(bike1.name);
17. console.log(bike2.maker);
19. console.log(bike1.getDetails());

3. UIsng Object.Create()

<script>

    // Object.create() example a

    // simple object with some properties

    const coder = {

        isStudying : false,

        printIntroduction : function(){

            console.log(`My name is ${this.name}. Am I

            studying?: ${this.isStudying}.`)

        }

    }

    // Object.create() method

    const me = Object.create(coder);

    // "name" is a property set on "me", but not on "coder"

    me.name = 'Mukul';

    // Inherited properties can be overwritten

    me.isStudying = true;

    me.printIntroduction();

We can create class (ES6)

class emp

{

    constructor(id, empname)

    {

        this.id=id;

        this.empname = empname;

    }

    get= ()=>

    {

         console.log("ID is " + this.id);

         console.log("Name is " + this.empname);

    }

}

const emp1 = new emp(1,"Deepak");

const emp2= new   emp(2,"Ajay");

emp1.get();

emp2.get();

Inheritance

class emp

{

    constructor(id, empname, manager)

    {

        this.id=id;

        this.empname = empname;

        this.manager = manager;

    }

    get = ()=>

    {

         console.log("ID is " + this.id);

         console.log("Name is " + this.empname);

         console.log("Manager is " + this.manager)

    }

}

class fullTime extends emp{

    constructor(id, empname, manager, dept)

    {

        super(id,empname, manager);

        this.dept = dept;

    }

    get1 = ()=>

    {

       console.log("dept is " + this.dept)

    }

}

const ft = new fullTime(1,"Deepak", "aa","HR");

// const emp2= new   emp(2,"Ajay");

// emp1.get();

ft.get();

ft.get1();

class emp

{

    #address;

    constructor(id, empname, manager, address)

    {

        this.id=id;

        this.empname = empname;

        this.manager = manager;

        this.#address = address;

    }

    get = ()=>

    {

         console.log("ID is " + this.id);

         console.log("Name is " + this.empname);

         console.log("Manager is " + this.manager)

          console.log("Address is " + this.#address);

    }

}

class fullTime extends emp{

    constructor(id, empname, manager, address, dept)

    {

        super(id,empname, manager, address);

        this.dept = dept;

    }

    get1 = ()=>

    {

       console.log("dept is " + this.dept)

    }

}

const ft = new fullTime(1,"Deepak", "aa", "Delhi", "HR");

// const emp2= new   emp(2,"Ajay");

// emp1.get();

ft.get();

ft.get1();

// Abstraction example

    function person(fname,lname){

        let firstname = fname;

        let lastname = lname;

        let getDetails\_noaccess = function(){

            return (`First name is: ${firstname} Last

                name is: ${lastname}`);

        }

        this.getDetails\_access = function(){

            return (`First name is: ${firstname}, Last

                name is: ${lastname}`);

        }

    }

    let person1 = new person('Mukul','Latiyan');

    console.log(person1.firstname);

    console.log(person1.getDetails\_noaccess);

    console.log(person1.getDetails\_access());