```
//prototype object
// What is prototype? Js er je kono ekta property jeta object k point kore

//js a inheretence hoi prototype er maddome but onno programming ye inheretence hoi class er maddome

//let person = [] or let person = new Array();// both same

//console.log(Array.prototype);

// vanilla js a kono class nei

//js a amra constrator theke object toiri kori, by default every function is constrator

//log er upgrade version dir();

//shob object e mashter object theke toiri hoi(Object)

//object
let person = {};

person.name = "Mila";
person.age = 30
person.eat = function () {
    console.log(`person is eating`);
};
person.seep = function () {
    console.log(`person is sleeping`);
};

//return object from function

let p = function Person(name, age) {
    let person = {};

    person.name = name;
    person.age = age;

    person.eat = function () {
        console.log(`person is eating`);
    };
    person.seep = function () {
        console.log(`person is sleeping`);
    };
    return person;
}
// console.log(p);
console.dir(p);




 function Person(name, age) {//constructure function tai P capitall letter
    let person = {};

    person.name = name;
    person.age = age;

    //(method)
    person.eat = function () {
        console.log(`person is eating`);
    };
    person.seep = function () {
        console.log(`person is sleeping`);
    };
    return person;
}

const mila = Person("Mila", 35);
const mili = Person("Mili", 65);




//object count optimization

const personMethod = {
```

```
    //(property object er)
    //egula method
    eat() {
        console.log(`person is eating`);
    },
    sleep() {
        console.log(`person is sleeping`);
    },
    play(){
        console.log(`person is play`);
    }
}

function Person(name, age) {
    let person = {};

    person.name = name;
    person.age = age;

    person.eat = personMethod.eat;
    person.seep = personMethod.sleep;
    person.play = personMethod.play;
    return person;
}

const mili = Person("Mili", 35);
const mili = Person("Mili", 65);




///object create()


//object
const captain = {
    name: "mashrafi",
    age: 23,
    country: 'bd',
}
//object
const player = Object.create(captain);
console.log(player.name);//run this for prototype




//prototype introduction
const personMethod = {
    //(property object er)
    //egula method
    eat() {
        console.log(`person is eating`);
    },
    sleep() {
        console.log(`person is sleeping`);
    },
    play(){
        console.log(`person is play`);
    }
}

function Person(name, age) {
    let person = Object.create(personMethod);
    console.log(person);

    person.name = name;
    person.age = age;
    return person;
}

const sakib = Person("Sakib", 35);
sakib.play();
const mili = Person("Mili", 65);
```

```
function test() {}
console.log(test.prototype);
console.dir(test);
```

```
//constrator function

 function Person(name, age) {
        let person = Object.create(Person.prototype);
        console.log(person);

        person.name = name;
        person.age = age;
        return person;
    }
    //C.F er prototype er modda amra kisu method add korci ai khane
Person.prototype = {
    eat() {
        console.log(`person is eating`);
    },
    sleep() {
        console.log(`person is sleeping`);
    },
    play() {
        console.log(`person is play`);
    }
}
const sakib = Person("Sakib", 35);
sakib.play();
const mili = Person("Mili", 65);
```

```
// prototype
//x.getX ai function er bitore rakle arek ta x1 function create korbe jar modda getx thakbe s
o onek boro hobe jotil hoiya jabe
var x = function() {
    this.x = 5;
    x.getX = function() {
        return this.x;
    }
  }
  var x1 = new x();
  console.dir(x1);


//correct way protor  modda getX thakbe
var x = function() {
  this.x = 5;
}
x.prototype.getX = function() {
    return this.x;
}
```

```
var x1 = new x();
console.dir(x1);
```

```
//new and this keyword
 function Person(name, age) {
        // let this = Object.create(PersonWithNew.prototype);
        console.log(person);

        this.name = name;
        this.age = age;
        // return this;
    }
    //C.F er prototype er modda amra kisu method add korci ai khane
Person.prototype = {
    eat() {
        console.log(`person is eating`);
    },
    sleep() {
        console.log(`person is sleeping`);
    },
    play() {
        console.log(`person is play`);
    }
}
const sakib = new Person("Sakib", 35);
sakib.play();
const mili = new Person("Mili", 65);
```

```
//class in js (class conversion)
class Person {
    constructor(name, age){
        this.name = name;
        this.age = age;
    }
    eat(){
        console.log('Person is eating');
    }
    sleep(){
        console.log('Person is sleeping');
    }
}

const sakib = new Person( 'Sakib', 23);
sakib.eat();

// prototype Array
let person = [];//both same

let person = new Array();
person.push('sakib');
// console.log(person);
console.log(Array.prototype)
```

```
// //prototype
// function Person(name, Age){
//     this.name = name;
//     this.age = age;

//     this.eat = function(){
//         console.log(`${this.name} is eating`);
//     }
// }
// const sakib = new Person('sakib', 34);
// const tamim = new Person('Tamim', 24);







// //prototype

// function Person(name, Age){
//     this.name = name;
//     this.age = age;
// }
// Person.prototype = {
//     eat: function(){
//         console.log(`${this.name} is eating`);
//     }
// const sakib = new Person('sakib', 34);
// const tamim = new Person('Tamim', 24);




// //prototype chain// check browsr console
// var f = function Person(){};
// console.dir(f);


// var f = function Person(){};
// Object.prototype.mila = function(){
//     console.log("I am Mila");
// };
// var p = {};
// p.mila();


var f = function Person(){

};

//mashter object a amra sumit dukiya dilam tai amra sumit excess korte partasi
Object.prototype.sumit = function(){
    console.dir('Mila')
}
var p = {};
p.sumit();







//prototype inheritance
function Person(name, age){
    //parent class
    this.name = name;
    this.age = age;

}

function Cricketer(name, age, type, country){
    //sub class
   Person.call(this);
   this.name = name;
   this.age = age;
    this.type = type;
```

```
    this.country = country;
}
Person.prototype = {
    eat: function (){
        console.log(`${this.name} is eating`)
    }
}
//object .create er modda ami perent k diyasi akhon Person.prototype er modda ja ase Cricketer.prototype niya ashlo
Cricketer.prototype = Object.create(Person.prototype);
//constructor ke objer write korlam karon Person.call default construror er modda nai
Cricketer.prototype.constructor = Cricketer;//Cricketer.prototype.constructor er modda function Cricketer diya disi
Cricketer.prototype.play = function(){
    console.log(`${this.name} is playing`);
}
let sakib = new Person('Sakib', 34);
let tamim = new Cricketer('Tamim', 23, 'All Rounder', 'BAngladesh');//new Cricketer dhara function cricketer call hossey

console.log(sakib.eat());

// class conversion
class Person{//parent class
    constructor(name, age){
        this.name = name;//property
        this.age = age;
    }
eat() {//method
    console.log(`${this.name} is eating`);
}
}

class Cricketer extends Person{//sub class
    constructor(name, age, type, country){
        super(name, age);
        this.name = name;
        this.age = age;
        this.type = type;
        this.country = country;
    }
play(){
    console.log(`${this.name} is playing`);
}
}

let tamim = new Cricketer('Tamim', 23, 'All Rounder', 'BAngladesh');
console.log(tamim.name);
let sakib = new Person('Sakib', 23);
sakib.play();//eta kaj korbe na karon parent child er excess pai na
sakib.eat();//its working



//geter & setter

class Person {//parent class
    constructor(name, age) {
        this.name = name;//property
        this.age = age;
    }
    eat() {//method
        console.log(`${this.name} is eating`);
    }

    get setName() {//method //getter
        return this.name;//Or return mila;
    }
    set setName(name) {//setter
        this.name = name;
    }
}

let sakib = new Person('Sakib', 23);
console.log(sakib.setName);//not console.log(sakib.setName());
sakib.setName = 'Tamim';
console.log(sakib.name);
```

```
//static method
class Person {//parent class
    constructor(name, age) {
        this.name = name;//property
        this.age = age;
    }
    eat() {//method
        console.log(`${this.name} is eating`);
    }

    static isEquilAge() {
       // static
        console.log(`I am Static`);
    }
}

let sakib = new Person('Sakib', 23);
Person.isEquilAge();//Person.eat() not posible
```

```
class Person {//parent class
    constructor(name, age) {
        this.name = name;//property
        this.age = age;
    }
    eat() {//method
        console.log(`${this.name} is eating`);
    }

    static isEquilAge(cricketer1, cricketer2) {//static
        return cricketer1.age === cricketer2.age
    }
}

let sakib = new Person('Sakib', 23);
let tamim = new Person('tamim', 23);
console.log(Person.isEquilAge(sakib, tamim));
```

```
class Person {//parent class
    constructor(name, age) {
        this.name = name;//property
        this.age = age;
    }
    eat() {//method
        console.log(`${this.name} is eating`);
    }

    static isEquilAge() {//static isEquilAge() ai method er modda this mane Person class sha sakib k chine na sakib object er shate t
ar kono shomporko nai;
        return this.name;//Ans Person    // ai jonnoi er age === diya check kora hoise
    }
}

let sakib = new Person('Sakib', 23);
console.log(Person.isEquilAge());//class.method korsi ai khane aita shudu ai static hower karone e partasi
```

```
//polmorphism
//child class or inhereted class jodi tar parent er kono kisu change/modify kore shai modify korer procedure kei bole polymorphison
// class conversion
class Person{//parent class
    constructor(name, age){
        this.name = name;//property
        this.age = age;
    }
eat() {//method
    console.log(`${this.name} is eating`);
}
}

class Cricketer extends Person{//sub class
    constructor(name, age, type, country){
        super(name, age);
        this.name = name;
        this.age = age;
        this.type = type;
        this.country = country;
    }
    eat() {//polmorphism
        console.log(`${this.name} is eating rice and vagitable`);
    }

    // eat() {//polmorphism
    //     super.eat();
    //     console.log(`${this.name} is eating rice and vagitable`);
    // }
}

let tamim = new Cricketer('Tamim', 23, 'All Rounder', 'BAngladesh');
tamim.eat();







//scopes
//parent child ke shob diya dai

var x = 23;
//windows scope
//parent er dunia
function myFunc(){
    //myFunc or child er duniya
    ///myFunc scope
    var y = 10;
    console.log(`${x} from myFunc()`);
}
myFunc();
console.log(x); //or
console.log(window.x);//but aita likte hoi na




// "use strict";
//windows scope
function myFunc(){
    x = 10;//window scope a 10 pabo //strick likle r pabo na // var x = 10 likle o pabe na
    console.log(`${x} from myFunc()`);
}

console.log(window.x);
```