```
? qus 1
 function showData() {
     console.log(name); undefinded
     console.log(age); cannot access before initialization
     var name = 'anupam';
     let age = 3;
 }
 showData();

? qus 2
 for (var i = 0; i < 4; i++) {
     setTimeout(() => {
         console.log(i)  each time 4 due to var
     }, 5);
 }
 for (let i = 0; i < 4; i++) {
     setTimeout(() => {
         console.log(i) each time 0 1 2 3
     }, 5);
 }

? qus 3
 const income = {
     skills: 108,
     monthly() {
         return this.skills * 10;
     },
     yearly: () => {
         return this.skill * 12;
     }
 }
 console.log(income.yearly()) nan
 console.log(income.monthly()) 1080

 console.log(+true); 1
 console.log(!"js"); false


!
 const code = {
     type: "web"
 }
 const reactjs = {
     name: "js",
     web: "true"
 }

which options are not correct
/*
```

```
a: reactjs[code.type] valid
b: reactjs[reactjs["type"]]  valid but undefined
c: code.type.web not valid web is value not key
d: all of them are valid if use in strict mode

*/


!
 let a = { greeting: 'HI' };

 let z = a; here the address of a is passed refrence

 z.greeting = "bye";
 console.log(a.greeting)  bye


 !let a = 108;
 let b = new Number(108);
 let c = 108;

 console.log(a == b); true
 console.log(a === b); false new number is object in type
 console.log(b === c); false same reason

! class Lizard {
     static colorChange(newColor) {
         this.newColor = newColor;
         return this.newColor;
     }
     constructor({ newColor = "orange" }) {
         this.newColor = newColor;
     }
 }

 const tommy = new Lizard({ newColor: "orange" });
 console.log(tommy.colorChange('blue'))
 tommy.colorChange is not a function because it is a static function
and static functions are not available on object declaration


!
 let message;
 masage = {
     data: [23]
 }
 console.log(masage)  no error thrown due to mis spell because js
declare all the variables by self and add them to the window scope to
prevent this usestrict mode
```

```
!
 function showModal() {
     console.log(showModal.timeout);
 }
 function works the same as object

 showModal(); ?undefined
 showModal.timeout = 200;

 showModal.timeout = 100;
 showModal(); ? 100

!
 function HumanP(firstName, lastName) {
     this.firstName = firstName;
     this.lastName = lastName;
 }
 const member = new HumanP('frontend', 'master');

 HumanP.getFullName = function () {
     return `${this.firstName} ${this.lastName}`
 }  here we are adding it as a key so it is not accessible to make it
work we neeed to add it in HumanP.prototype.getFullName
 console.log(member.getFullName()); TypeError: member.getFullName is
not a function

? new keyword creates a new object and binds it to the function callled
a constructor call and this keyword points to the object created by new
keyword and return the object with binding to this

 function Human(fName, lName) {
     this.firstName = fName;
     this.lastName = lName;
 }
 const Mrx = new Human("mr", 'x');
 const Rock = Human('The', 'Rock');

 console.log(Mrx); Human { firstName: 'mr', lastName: 'x' }
 console.log(Rock); undefined

! what are three phases of event propagation?

/*
a: target > capturing > bubbling
b: bubbling > target > capturing
c: target > bubbling > capturing
d: capturing > target > bubbling !correct
```

```
*/

!let salary = 10;
console.log(salary++); 10
console.log(++salary); 12
console.log(salary)12

function sum(a, b) {
    return a + b;
}
console.log(sum(10, "10")) 1010

function getSummary(one, two, three) {
    console.log(one) [ '', ' age is ', '' ]
    console.log(two) vasuki
    console.log(three) 878
}
const fName = 'vasuki';
const age = 878;
getSummary`${fName} age is ${age}`;

function checkAge(data) {
    obj are refernce base so it faila
    if (data === { age: 18 }) {
        console.log("you are an adult");
    } else if (data == { age: 18 }) { it also fail
        console.log('you are still an adult')
    } else {
        console.log('hmm..no age')
    }
}

checkAge({ age: 18 }) hmm..no age


function getType(...args) {
    console.log(typeof args) object [ 10, 8 ]
}
getType(10, 8);

function getAge() {
    "use strict"
    salary = 8212;
    console.log(salary);
}

getAge() ReferenceError: salary is not defined

var num = 8;
```

```
 var num = 10;
 console.log(num) 10


in js obj keys are always string
 const obj = {
     1: 'a',
     2: 'b',
     3: 'c'
 }
 const set = new Set([1, 2, 3, 4, 5]);

 console.log(obj.hasOwnProperty('1')); true
 console.log(obj.hasOwnProperty(1)); true

 console.log(set.has('1')); false set has no key concept direct value
check
 console.log(set.has(1));true

 const obj = {
     a: "1",
     b: "2",
     a: "3"
 };
 console.log(obj) { a: '3', b: '2' }

 for (let i = 1; i < 5; i++) {
     if (i === 3) continue;
     console.log(i);   1 2 4
 }

 String.prototype.youAreAmazing = () => {
     return "you are amazing"
 }
 const who = 'viewer';
 console.log(who.youAreAmazing()) you are amazing [because in the
protoype chain of sting we added our method]

 const a = {};
 const b = { key: "b" };
 const c = { key: "c" };

 a[b] = 222; a[object object]
 a[c] = 333; a[object object]
 console.log(a[b]) 333

 const lang = { name: "reactjs" };
 function getLib(ver) {
     return `${this.name} version ${ver}`;
```

```
}

console.log(getLib.call(lang, 18)); reactjs version 18
console.log(getLib.apply(lang, [18])); reactjs version 18
const bound = getLib.bind(lang);
console.log(bound(18)) reactjs version 18

function sayHi() {
    return (() => 0)()
}
console.log(typeof sayHi()) number

console.log(typeof typeof 1);string

const numbers = [1, 2, 3];
numbers[10] = 11;
console.log(numbers); [ 1, 2, 3, <7 empty items>, 11 ]

(() => {
    let x, y;
    try {
        throw new Error();

    } catch (x) {
        (x = 1), (y = 2);
        console.log(x) 1
    }
    console.log(x); undefined
    console.log(y) two
})()

const data = [..."Apple"];
console.log(data) [ 'A', 'p', 'p', 'l', 'e' ]

let person = { role: "dev" };
const numbers = [person];
person = null;

console.log(numbers); [ { role: 'dev' } ]

console.log(20 + 30 + '10')5010

function getMessage() {
    throw "Hello World";
}

function sayhello() {
    try {
```

```
        const data = getMessage();
        console.log('worked', data);
    } catch (e) {
        console.log('an error', e)
    }
}

sayhello();  an error hello world

console.log(parseInt('10+2')); 10 invalid string so it check it check
if there any possible number in the start or not
console.log(parseInt('7FM')); 7

console.log(
    [1, 2, 3].map(num => {
        if (num > 0) return; // this line causes error
        return num * 2;
    })) [ undefined, undefined, undefined ]


function getInfo(member, year) {
    member.name = 'frontendmaster';
    year = '1947';
}
const person = { name: 'dev' };
const birthYear = '2097';

getInfo(person, birthYear);
console.log(person, birthYear) { name: 'frontendmaster' } 2097

function Hero() {
    this.make = 'Bhagat singh';
    return { make: "vivekanand" };
}
const myHero = new Hero();
console.log(myHero.make) vivkeanad


(() => {
    let x = (y = 10);
     y=10;
    let x=y here y is undeclared so it added in global scope with
undefined
})()
console.log(typeof x); undefined
console.log(typeof y); number

const obj = { a: "mr.x", b: 21 };
const data = { c: true, ...obj };
```

```
 console.log(data) { c: true, a: 'mr.x', b: 21 }

 const obj = {};
 Object.defineProperty(obj, 'a', {
     value: 'cahr'
 }) due to security reason created with define propery so it will not
be shown in loopr to surpass this use enumerable:true
 console.log(obj.a)cahr
 console.log(Object.keys(obj))[]

 const box = { x: 10, y: 20 };
 Object.freeze(box);

 const shape = box;
 shape.x = 100;
 shape.z = 20;

 console.log(shape) { x: 10, y: 20 }

 function addItem(item, list) {
     return list.push(item);
 }
 const result = addItem('orange', ['apple', 'banana']);
 console.log(result)   3 push return the lenght of array after elm is
insertex

 const name = 'Mr.x';
 age = 20;
 console.log(delete name); false
 console.log(delete age); true

delete use to delete keys in obj

 function* generatorFn(i) {
     console.log('A');
     yield i;
     console.log('B');
     yield i + 10;
 }
 const gen = generatorFn(10);
 console.log(gen.next().value);
 console.log(gen.next().value)
/* https:www.instagram.com/p/Ct9bf8bAHw3/
A
10
B
20
*/
```

```
async function getData() {
    return await Promise.resolve('hi')
}

const data = getData();
console.log(data) Promise { <pending> }


data.then(res => console.log(res)) hi

const { fname: feDev } = { fname: "mxx" };
console.log(fname) ReferenceError: fname is not defined

function sum(n1, n2 = n1) {
    console.log(n1 + n2)
}
sum(10) 20

class Person {
    constructor(name) {
        this.name = name;
    }

}
const member = new Person('mrs');
console.log(typeof member) object

let newList = [2, 3].push(4); 3
console.log(newList.push(5)); TypeError: newList.push is not a
function

function getItems(list, ...args, moreItem) {
    return [...list, ...args, moreItem]
}
getItems(['berry', 'apple'], 'pear', 'kiwi')function getItems(list,
...args, moreItem) {
  SyntaxError: Rest parameter must be last formal paramete

function nums(a, b) {
    if (a > b) console.log('a is large')
    else console.log('b is large')
    return a + b;
}

console.log(nums(4, 2));
console.log(nums(1, 2)) a is large
6
b is large
```

```
3

class Person {
    constructor() {
        this.name = 'Frontend';
    }
}
Person = class AnotherPerson {
    constructor() {
        this.name = 'Backend';
    }
}

const member = new Person();
console.log(member.name) /backend


const name = 'Happy sing'
console.log(name()) TypeError: name is not a function

let name = 'dev';
function getName() {
    console.log(name)
    let name = 'frontendmaster'
}
getName() ReferenceError: Cannot access 'name' before initialization

const one = false || {} || null;{}
const two = null || false || ''; ""
const three = [] || 0 || true;[]

console.log(one, two, three)

`${(x => x)('I LOVE ')} JS` i love js

let num = 1;
const list = ['a', 'b', 'c'];
console.log(list[num += 1]) c

let randomValue = { name: 'lydia' };
randomValue = 23;

if (!typeof randomValue === 'string') {
    console.log('it not a string');
} else {
    console.log('yays its a string')
}
output : yays its a strign
```

```
const animals = {};
let dog = { emoji: 'asdf' };
let cat = { emoji: "adfs" };
animals[dog] = { ...dog, name: 'drug' };
animals[cat] = { ...cat, name: 'cat' };

console.log(animals[dog]) { emoji: 'adfs', name: 'cat' }

const fruits = ['a', 'b', 'c'];
fruits.slice(0, 1); only remove elm but not modify the array
fruits.splice(0, 1); it change the array also modify
fruits.unshift('aa');
console.log(fruits) ['aa','b','c']

const add = x => x + x;
function myFunc(num = 2, value = add(num)) {
    console.log(num, value)
}
myFunc(); 2 4
myFunc(3);  3 6

let count = 0;
const nums = [0, 1, 2, 3];
nums.forEach(num => {
    if (num) {
        count += 1;
    }
})
console.log(count)  3 not 4 because first number is 0 wbhich is false
if loop break

const person = {
    name: "frontend",
    address: {
        city: "mdn"
    }
}

Object.freeze(person);
person.name = "backend";
person.address.city = "delhi";

console.log(person) { name: 'frontend', address: { city: 'delhi' } }
name is not changed because it is freezed but address is changed
because it is not freezed

const person = {
    name: 'frontend',
```

```
 }
 Object.seal(person);

 person.name = 'backend';
 person.age = 20;

 delete person.name;
 console.log(person) { name: 'backend' } name is not deleted because it
is sealed

 const handler = {
     set: (target, property, value) => {
         console.log('add a new property')
     },
     get: () => console.log('accessed a property')
 }

 const person = new Proxy({}, handler);
 person.name = 'frontend';
 person.name;

add a new property
 accessed a property

 const MESSAGE = 108;

 function getInfo() {
     console.log(MESSAGE);
     const MESSAGE = 'AFDADSF';
 }
 getInfo(); ReferenceError: Cannot access 'MESSAGE' before
initialization

 const pets = ['a', 'b'];
 ({ item: pets[2] } = { item: 'asdf' });
 console.log(pets) [ 'a', 'b', 'asdf' ]

 const myFunc = ({ x, y, z }) => {
     console.log(x, y, z);
 }
 myFunc(1, 2, 3) undefined undefined undefined

 const FOO = 'FRONTEND';

 console.log(!typeof FOO == 'object');  false
 console.log(!typeof FOOm == 'string'); false

 const add = x => y => z => {
```

```
    console.log(x, y, z);
    return x + y + z;
}
add(10)(20)(30); currying when a funciton return another funciton

const groc = ['a', 'b'];
if (groc.indexOf('a')) {
    console.log('we have a');
} else {
    console.log('we dont have b');
}

const obj = { name: 'js' };
obj.ref = obj;

const str = JSON.stringify(obj); TypeError: Converting circular
structure to JSON

console.log(str)

var magic = 900;
function magic() {
    console.log('magic')
}
console.log(magic)  900

const array = [{
    key: "j"
}, '2', 'x'];
delete array[0];
console.log(array.length, array) 3 [ <1 empty item>, '2', 'x' ]

let z = a = {}
a.name = 'js'; poining to same obj
console.log(z.name) js

function task() {
    return new Promise((res) => {
        res('daata')
    })
}
const result = task().then();
console.log(result) Promise { <pending> }

console.log(1);
new Promise(function (res) {
    console.log(2)
})
```

```
 console.log(3)  1 2 3

 const dataMap = new WeakMap();
 let person = { name: 'js' };

 dataMap.set(person, 'asfd');
 console.log(dataMap.get(person)); afsd
 person = null;
 console.log(dataMap.get(person)); undefined

 var foo = function test() {
     console.log('isnide test')
     test()  -> this can be true
 }
 test() annonymouse function can be called inside a funciton only this
will give reference error

 againTest()
 test() test is not a funitno
 var test = function () {
     console.log('insdie test')
 }
 function againTest() {
     console.log('agian test test')
 }

 const data1 = ['c', 'b'];
 const data2 = ['a', 'b', 'c'];

 console.log(data1.sort() === data1); true
 data1.toSorted(); return the sorted array
 data1.sort()modify the original array

 data1.toReversed()
 data1.reverse()

 const arr = [, , ,];
 console.log(arr.length)3

 let x = 10;
 let y = 'a';
 [x, y] = [y, x];
 console.log(x, y) a 10

 let x = [typeof x, typeof y];
 console.log(x) ReferenceError: Cannot access 'x' before initializatio
right to left'
```

```
const [x, ...y] = [1, 2, 3, 4];
console.log(x, y) 1, [ 2, 3, 4 ]

var age = 99;
console.log(window.age) 99

let name = 'js';
name[0] = 'r';
console.log(name) js

let str = new String("js");
console.log(str === 'js'); false
console.log(str == 'js') true

const obj = {};
obj[obj['a'] = 'b'] = 'c';
console.log(obj) { a: 'b', b: 'c' }


5 ways to create an object
const obj= {};
const obj1 = new Object();
const obj2 = Object.assign({},{});
const obj3 = Object.create({});
const obj4 = new function(){};

const arr = [1, 2, 3];
console.log(arr[5]) undefined

function init(x, y, z) { }
function end(a, b = 0, c) { }

console.log(init.length);3
console.log(end.length) 2

const person = {
    lang: "js",
    show: function () {
        console.log(`hi ${this.lang}`);
    }
}
let fn = person.show;
fn(); hi undefined
 solution
fn = person.show.bind(person);
fn(); hi js

console.log([]===[]); false
console.log({}==={}) false
```

```
 console.log([]==[]) false
 console.log({}=={}) false

temporal dead zone
 console.log(age) undefined
 var age = 99;
 if (function fn() { }) {
     console.log(fn) ReferenceError: fn is not defined
 }

 99['toString'].length + 1; 2

 console.log(4 + '4'); 44
 console.log(8 + +'8')16

 let x;
 if (x == 1 && x == 2 && x == 3) {
     console.log('asfdadsf');
 }

 typeof jsIsAwa; undefined

 function sum() {
     return 2 + 2;
 }
 function sq() {
     return 4 * 4;
 }
 let a = (sum(), sq());
 console.log(a)      16

 const obj = { name: 'x' };
 delete obj.name;
 obj?.name = 'y' /invalid assignment

 let lifeSapn = {
     99: "safa"
 }
 lifeSapn.100 = 'asdf';
 console.log({ lifeSapn }) only string and symbols can be used as a key


 console.log('a'); micor task
 (async function () {
     const x = await 5; await is macro
     console.log('c')
 })()
 console.log('b')
a b c
```

```
const arr = ['ab', 'cd', 'ef'];
const str = 'abcdef';

const strMatch = str.includes('a');
const arrMatch = arr.includes('a');

console.log({ strMatch, arrMatch }) { strMatch: true, arrMatch: false
} because in case of array it check for direct fit means check for same
value

console.log(false == []); true  false == '' means 0 ==0
console.log(false == ![]); true empty array is truthy value

console.log(8888888888888888888) number.max_safeinteger

function show() {
    {
        var x = 9;
        var y = 10;
       (function () {
           var x = 9
           var y = 10
       })()
    }
    console.log(x, y); reference error
    if use var they can be accessed outside the scope to prevent this
we can use iife or change var to let
}
show()

let score = 2;
let message = `your score is ${String(score).padStart(3, 0)}`;

console.log("🚀 ~ message:", message);

JS is dynamically type lang during the time of compilation assign the
data type

let a = 10
let b = 20
let c = 30 - (a = b + 10)
console.log("🚀 ~ c:", c); 0

console.log(NaN == NaN) false
console.log(NaN === NaN) false

to check any value nan or not by using isNaN(value)
```

```
 Number.isNaN(value)'

make the lenght 0;
 let arr = [1, 3, 4];
  arr.length = 0;
  arr.splice(0, arr.length)

 console.log(arr)


 function show() {
     console.log('0', arguments[0]); 21
     console.log('len', arguments.length); 2
     for (let x of arguments) {
         console.log('x', x);  21 js
     }
     console.log(Array.isArray(arguments)) false
 }
 show(21, "js")

 const obj = {};
 Object.defineProperty(obj, "lang", { value: "js" });
 console.log(obj)  {} non enumerable due to use of defineproperty
 console.log(obj.lang) js

 console.log(-0 == 0); true
 console.log(-0 === 0) true
 Object.is(0, -0) false


input a.b.c.d.e

/*output : {
    a:{
        b:{
            c:{
                d:e
            }
        }
    }
}*/

 const str = 'a.b.c.d.e';
 str.split('.').reduceRight(function (acc, red) {
     console.log({
         [red]: acc
     })
     return { [red]: acc }
 });
```

```
create a function without using a function and arrow function
 const a = 10;
 const b = 20;
 const add = new Function('a', 'b', 'console.log(a+b)')
 add(10, 20)


 const str = 'hare krishna hare krishna hare krishna';
 const obj = {};
 for (let x of str) {
      obj[x] = (obj[x] || 0) + 1;
     if (obj[x]) {
         obj[x]++
     } else {
         obj[x] = 1
     }
 }
 console.log(obj)

is everything in js is an object
no

 const arr = [1, 2, 3, 4];
 arr.forEach((val, ind) => {
     if (val == 2) {
         throw new Error('message'); method to break the foreach  or
set the arr.length = 0 to break or app.splice(ind+1,arr.length)
     }
 })

 const key = 'constructor';
 const obj = {};
 if (obj[key]) {
     console.log('hello admin')
 } else {
     console.log('hello guest')
 }

 if (key in obj) {
     console.log('hello admin')
 } else {
     console.log('hello guest')
 }

 output: - hello admin  hello admin

 in js switch statemtn use == or === for case comparison
```

```
 ans : ===

 var God = {
     slogan: 'jaishreeram'
 }
 var god2 = Object.create(God);

 delete god2.slogan;
 console.log(god2.slogan) jaishreeram becuase slogan key was added in
the prototype chain and delete method can't delete the method from the
proto chain only keys can be deleted

 const getData = () => {
     console.log(this) window obj
 }
 const obj = { user: 1 };
 getData.call(obj);

 const superHero = { name: 'silversurfer' };
 const collector = { planet: "xnatar" };
 collector.__proto__ = superHero;
 console.log(collector.name);

 console.log(0 == '0'); true
 console.log(0 == []) true double equal says when you compare non prim
with prim then you have to convert the non prim into prim with help of
tostring prop and empty array converted to string '' which is false
equal to 0
 console.log("0" == []) false "0"==""

 !function () {
     console.log('i am amazed')
 }() type of iifee

 const num1 = Number();
 const num2 = Number(undefined);

 console.log(`num1 = ${num1}`) 0
 console.log(`num2 = ${num2}`) nan

get the last elm of array
 const arr = ['b', 's', 'sad'];
 console.log(arr.at(-1))

 function show() {
     console.log('wow')
 }
 const functionName = 'show';
```

```
show(); not allowed

window[functionName]()

const num1 = 10, num2 = 23;
console.log(num1 - (-num2)) 33

! +[] + [] + ![];
'truefalse'

const arr = [1, 2, 3, 4];
console.log(1 in arr)true in check for key in array
console.log(2 in arr) true
console.log(4 in arr)falase

const obj = {
    data: [1, 2, 3],
    processData: function () {
        this.data.forEach((num) => {
            console.log(num * this.factor);
        })
    },
    factor: 2
}
obj.processData()  2 4 6

function test() {
    try {
        return 1;
    } finally {
        return 2;
    }
}

console.log(test())

console.log(`${{ Object }}`) [object Object]
 breakdown
const obj = {};
obj.Object = () => { };
console.log(obj.toString())

const a = { fn: function () { } };
const b = [function () { }];

const strA = JSON.stringify(a);
const strB = JSON.stringify(b);
console.log(strA){}
```

```
console.log(strB)[null]

the maximum value of interval that settime allow is 2 to the power 32

const list = 'apple,samsua:sdfa';
const devi = list.split(/[,:]/)
console.log(devi)/

const obj = {
    age: 80,
    set age(newAge) {
        if (newAge < 20) throw new Error('age')
    }
}
const newAge = 19;
console.log('age should be greater than 20');
obj.age = newAge

const show = (b, b) => { arrow function doesn't support same parms
    console.log(b)
}
function test(a, a) {
    console.log(a)
}

test(12, 2)


function test(a, a) { console.log(a) }undefined
test(2)

const obj = {
    result: 0,
    add: function (n1) {
        this.result = n1;
        return this;
    },
    multiply: function (n2) {
        this.result *= n2;
        return this;
    },
}

const result = obj.add(10).multiply(10).result; method chaining
console.log("🚀 ~ result:", result); 100

console.log('' == [])true

setTimeout(() => { macro
```

```
    console.log(1)
}, 0);
Promise.resolve().then(() => {
    console.log(2) mircor added in queue
})

queueMicrotask(() => {
    console.log(3) micro added in quer
})
console.log(4)  exect then micro then macro

/*
4
2
3
1
 */

function test() {
    console.log(typeof this)
}
test.call("") object

write a fucntion to flat a array
const arr = [1, 2, [3, 4, [5, 6]]];
 console.log(arr.flat(2))

function flat(data) {
    const a = [];
    if (Array.isArray(data)) {
        data.forEach(function (e) {
            a.push(...flat(e))
        })
    } else {
        a.push(data)
    }
}

write a polyfill for map
const arr = [1, 2, 3, 4];
Array.prototype.myMap = function (cb) {
    const a = [];
    for (let i = 0; i < this.length; i++) {
        a.push(cb(this[i], i))
    }
    return a;

}
```

```
const newArr = arr.myMap(function (e, i) {
    return e * 2;
})
console.log(newArr)

3 ways to convert set into array
const mySet = new Set([1, 2, 3]);
const way1 = [...mySet];
const way2 = Array.from(mySet);
const way3 = new Array(...mySet)

let str = 'js'
JSON.stringify(str) === str; false

class User {
    constructor(name) {
        this.name = name;
    }
    login() {

    }
}

const user1 = new User('x');
const user2 = new User('y');
console.log(user1.login === user2.login) true

const a = 1 + undefined;
const b = 1 + typeof c;
console.log(a) nan
console.log(b)  1undefined

function x() {
    a();
    function a() {
        console.log('a')
    }
    a();
    function a() {
        console.log('b')
    }
    a()
}
x() b b b

let num = 2;
let foo = !--num;
let bar = !--num;
```

```
 console.log(foo) false
 console.log(bar) true

repeat a string 5 times
console.log('Hs'.repeat(5))

 console.log(~~21) 21
 console.log(~~21.21) 21

can you call a function with moon brackets()
 function show() {
     console.log('show')
 }
 new show;

 let bool1 = false;
 let bool2 = new Boolean(false);

 if (bool1) {
     console.log('fist block')
 }
 if (block2) {
     console.log('second block')
 }

second block


 let r = [1, 2, 3, 4][2, 0];
 console.log(r) index 0of first arr


 const { a = 'default', b = 'default', c = 'default', d = 'default', e
= 'default' } = { a: null, b: undefined, c: false, d: 0 }
 console.log({ a, b, c, d, e }) { a: null, b: 'default', c: false, d:
0, e: 'default' }

 console.log(Person); reference error cann't access before
initialization
 class Person { }

 let a = [].every(() => true);
 let b = [].every(() => false);
 console.log(a, b) true true forallqunatifier empty set can contain ay
type of value


 const bird = { name: 'bird' };
 const animal = { name: 'hen' };
```

```
 function show() { console.log(this.name) };
 const objShow = show.bind(bird);
 objShow.call(animal); bird because bind will set the permanent scope
of this keyword

 console.log(null == 0); falase
 console.log(null > 0) false
 console.log(null >= 0) true convert variable to number

 const isPass = false;
 console.log(isPass.rand) undefined

 console.log(2 == 2) true
 console.log(2 == 2 == 2) false
 console.log(2 == 2 == 2 == 2) true lfet to right


 class Magin { }
 console.log(typeof Magin) function

 console.log((true + "")[3]) e


function calc() {
    let i = 0;
    while (i < 2000000) {
        i++;
    }
}
 const before = new Date().getTime();
 calc();
 const after = new Date().getTime();
 const timeTaken = after - before;
 console.log(timeTaken)

 console.time('timer');
 calc()
 console.timeLog('timerend')


 const arr = ['anme'];
 const obj = {};

 obj.name = 'front';
 obj[arr] = 'react';
 console.log(obj.name) front

 function fetch() {
     A = 8;
```

```
    console.log(A)
}
let A;
fetch(); 8  beccause at the time when func call ed a is decalred

function add() {
    return 2 + 2;
}
var add;
console.log(add) fn body


console.log([] + {}) object object
console.log({} + []) obj obj 0

const Person = () => {
    this.name = 'js'
    return this;
}
const person = new Person();
console.log(person.name) reference error arrow fn thith this

var superHerp 'sifec';
let real = 'afds';
console.log(window.superHerp) sifed
console.log(windo.real) undef

let obj = { lan: 'rea' };
const lib = {}
lib.name = obj;
obj = null; console.log(obj.name) react /refere


var x = [typeof x, typeof y][1];
console.log(typeof typeof x) string

console.log([] == '') true
console.log([] == []) fals

{
    function show() {
        console.log('insdie')
    } to make this block wrap under a func or strick mode
}
show() indise
```