|  |
| --- |
| function check(a,b,c)  {  if(a>b && a >c)  {  alert(`${a} is greater`);  }else if(b >a && b > c)  {  alert(`${b} is greater`);  }  else if(c > a && c > b)  {  alert(`${c} is greater`);  }  }  check(12,150,20) |
| * Swap of number without third variable   x = x \* y; // x now becomes 15      y = x / y; // y becomes 10      x = x / y; // x becomes 5  a=a+b; // 30  b=a-b; // 20  a=a-b // 10 |
| * **How to sort a object**  var a = [   {name:'vinay',mark:50},  {name:'Amit',mark:100},  {name:'Bca',mark:120}  ];  var test = a.sort((a,b) =>{  if(a.name > b.name){return -1}  else{  return 1  }  }); **// decending order by name**  var test = a.sort((a,b) =>{  if(a.name > b.name){return -1}  else{  return 1  }  }); **// accending order by name** |
| * How to find a in a object userlist =[{name: ‘test’,roll:21}]   userlist.find(o => o.name === ‘test’) |
| * Filter on basis of objects   let test2 = a.filter(a => a.mark >= 100 && a.name=='Bca') |
|  |

var a = 1;

var b = '1';

if(a == b){

//true

}

var a = 1;

var b = '1';

if(a === b){

//false

}

var a = 1; // console.log(typeof(a)) // number

var b = '1'; // console.log(typeof(b)) // string

Which type of variable among global and local, takes precedence over other if names are same?

A local variable takes precedence over a global variable with the same name.

var a = 10;

function b(){

console.log(a); // output : 10

}

b() // 10

var b = 10;

function a(){

var b = 0;

}

console.log(b); // 10

\* what is output of below code

"use strict";

b = 10; // Ans : in this line it will return an error like b is not defined.

\* output of below code

(function test(){

c=10

})()

console.log(c); // Ans : 10 c treatd as global variable

\* what is output of below code

console.log(typeof(null)) // Ans : object

**(function(){**

**var a = b = 3;**

**})();**

Ans: answer will be same in normal funciton and self invoked function: actually shorthand for:

b = 3;

var a = b;

a defined? false

b defined? True

**on above function based question :**

console.log(typeof a); // undefined

console.log(typeof b); // number

* **What is output of below code**

var myObject = {

foo: "bar",

func: function() {

var self = this;

console.log("outer func: this.foo = " + this.foo); // outer func: this.foo = bar

console.log("outer func: self.foo = " + self.foo); // outer func: self.foo = bar

(function() { // if will use arrow function then this is available

console.log("inner func: this.foo = " + this.foo); // inner func: this.foo = undefined

console.log("inner func: self.foo = " + self.foo); // inner func: self.foo = bar

})();

}

};

myObject.func();

In the outer function, both this and self refer to myObject and therefore both can properly reference and access foo.

function foo1()

{

return {

bar: "hello"

};

}

function foo2()

{

return

{

bar: "hello"

};

}

console.log(foo1()); // {bar: "hello"}

console.log(foo2()); // undefined : because extra enter key press

console.log(typeof NaN) number

console.log(1 + "2" + "2"); // "122"

console.log(1 + +"2" + "2"); // "32"

console.log(1 + -"1" + "2"); // "02"

console.log(+"1" + "1" + "2"); // "112"

console.log( "A" - "B" + "2"); // "NaN2"

console.log( "A" - "B" + 2); // NaN

What is a “closure” in JavaScript? Provide an example.

A closure is an inner function that has access to the variables in the outer (enclosing) function’s scope chain. The closure has access to variables in three scopes; specifically: (1) variable in its own scope, (2) variables in the enclosing function’s scope, and (3) global variables.

use || and && in expression

//////// accending ///////////

var numArray = [140000, 104, 99];

for (var i = 0; i < numArray.length; i++) {

var target = numArray[i];

for (var j = i - 1; j >= 0 && (numArray[j] > target); j--) {

numArray[j+1] = numArray[j];

}

numArray[j+1] = target

}

console.log(numArray);

/////// descending///////////////

var numArray = [140000, 104, 99];

for (var i = 0; i < numArray.length; i++) {

var target = numArray[i];

for (var j = i - 1; j >= 0 && (numArray[j] < target); j--) {

numArray[j+1] = numArray[j];

}

numArray[j+1] = target

}

console.log(numArray);

Prototype in JavaScript

JavaScript is a prototype based language, so, whenever we create a function using JavaScript, JavaScript engine adds a *prototype* property inside a function, **Prototype property** is basically an object (also known as Prototype object),

**What is TypeScript ?**

TypeScript extends JavaScript by adding types.By understanding JavaScript, TypeScript saves you time catching errors and providing fixes before you run code.

Any browser, any OS, anywhere JavaScript runs. Entirely Open Source.TypeScript is an open-source language which builds on JavaScript. TypeScript code is transformed into JavaScript code via the TypeScript compiler or Babel. This JavaScript is clean, simple code which runs anywhere JavaScript runs: In a browser, on Node.JS or in your apps.

1. Find vowels is existing or not

function isVowelRegEx(char) {

if (char.length == 1) {

return /[aeiou]/.test(char);

}

}

console.log(isVowelRegEx(''));

1. Get vowels of the given string

function findVowels(str) {

return str.match(/[aeiou]/ig);

}

console.log(findVowels('aebracadabri'));

1. **How to count element of array**

const data = [5, 5, 5, 2, 2, 2, 2, 2, 9, 4]  
function count(arr) {  
 return arr.reduce((prev, curr) => (prev[curr] = ++prev[curr] || 1, prev), {})  
}  
console.log(count(data))

**get sum(1,2,3,4,5...)**  
function sum(...arr){

return arr.reduce((pre,curr)=>(pre+curr));

}

**sum(1,2,3,4)**

**Sum(1)(2)(3)()**

var sum = function (a) {

  return function (b) {

    if(b){

      return sum(a+b)

    }

    return a;

  }

}

add(1)(2,2));

function add(...args) {

if (!args.length) return 0;

const result = args.reduce((accumulator, value) => accumulator + value, 0);

const sum = (...innerArgs) => {

if (innerArgs.length === 0) return result;

return add(...args, ...innerArgs);

};

return sum;

}

console.log(add(1)(2,2));

1. Prime number program a number that is divisible only by itself and 1 (e.g. 2, 3, 5, 7, 11).  
function checkPrime(num){  let t=0

  for(let i=2;i<=num;i++){

    if(num%i===0){

      t++

    }

  }

  if(t===1){

    return 'Prime Number'

  }else{

    return 'Not Prime Number'

  }

}

2. perfect number, a **positive integer that is equal to the sum of its proper divisors.** excluding the **number** itself (6=1+2+3=6)  
function checkPerfect(num){

  var t=0;

  for(let i=1;i<num;i++){

    if(num%i===0){

      t=t+i

    }

  }

  if(t===num){

    return 'perfect Number'

  }else{

    return 'Not perfect Number'

  }

}