1.let num1 = 5;

let num2 = "5";

// This will be true due to type coercion

console.log(num1 == num2);

// This will be false because it checks both value and type

console.log(num1 === num2);

2. let value = 0;

if (value) {

console.log("Truthy");

} else {

console.log("Falsy");

}

**Falsy values include false, 0, ""**

3.How to check array length and make it empty

Let a=[1,2,3,4]

Console.log(a)

a.length=0;

console.log(a)

'11' - 1 = 11 - 1 = 10

**Duplicating array using set**

const numbers = [1, 2, 3, 4, 4, 4, 4, 5, 6, 6, 7];  
const fruits = ["apple”, "pear", "banana", “apple”, "apple", “cherry"];  
  
const uniqueNumbers ..new Set(numbers)];  
const uniqueFruits = [...new Set(fruits)];  
  
console. log(uniqueNumbers); // [ 1, 2, 3, 4, 5, 6, 7 ]  
console. log(uniqueFruits); // ["apple”, "pear", “banana”, "cherry"

we can increase the length of an array by simple assigning values to index.

const arr = [];  
  
console.log(arr.length); // 0  
  
  
arr[4] = 1;  
console.log(arr.length); // 5  
  
arr[400] = 1;  
console.log(arr.length); // 401

## we can set values agains string keys other than numerical indices

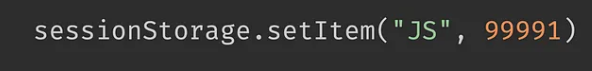
const arr = [];  
  
  
arr["JS"] = 1; // no effect on length  
console.log(arr.length); // logs 0  
console.log(arr["JS"]); // logs 1

# What will be logged in console?

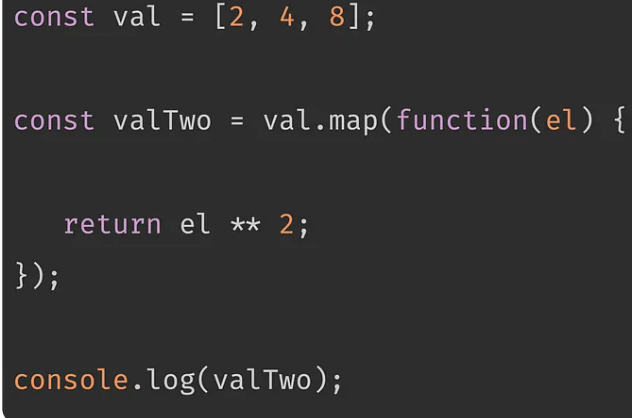
var a = 1;  
  
 function toTheMoon() {  
 var a; // var has function scope, hence it's declaration will be hoisted  
 if(!a) {  
 a = 10;  
 }  
 console.log(a); // 10 precendence will be given to local scoped variable.  
 }  
  
  
 toTheMoon();  
 console.log(a); // 1 refers to the `a` defined at the top.

Output 10 1

# How long will this data will be available?



sessionStorage.setItem("JS", 99991) will be available for as long as the current browser session is active. Once the session ends (e.g., the user closes the tab or browser), the data will be cleared. As a matter of fact it will survive the refresh.



**Output: 4 16 64**

const array = [{name:'JS'}, '2', 'Hello'];  
delete array[1];  
  
console.log(array.length); // ??

output:3

**2. we can increase the length of an array by simple assigning values to index.**

const arr = [];  
  
console.log(arr.length); // 0  
  
  
arr[4] = 1;  
console.log(arr.length); // 5  
  
arr[400] = 1;  
console.log(arr.length); // 401

6. console.log([] + []);

Output :” “

7. console.log(typeof NaN);

Output:number

Difference between array and set.

# Uniqueness:

**- Array:**- Can contain duplicate values.

**-Set:**- Only stores unique values; duplicates are automatically eliminated.

// Array  
const arrayWithDuplicates = [1, 2, 2, 3, 4, 4, 5];  
console.log(Array.from(new Set(arrayWithDuplicates))); // [1, 2, 3, 4, 5]

# Accessing Elements

**- Array:**- Access elements by index (`array[index]`).

**- Set:**- No direct indexing; values are accessed using iteration or conversion to an array.

- Array:  
- Has a `length` property indicating the number of elements.

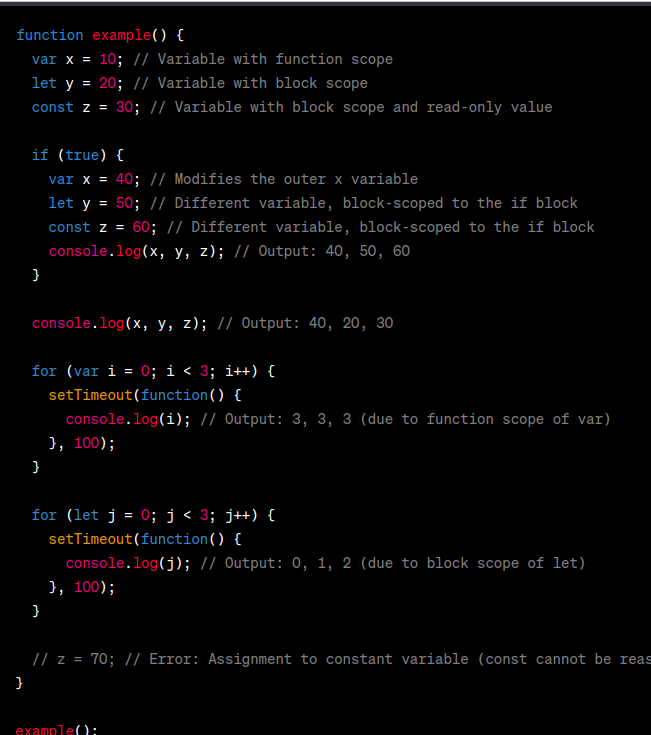
- Set:  
- Has a `size` property instead of `length`.

// Array  
const arrayExample = [1, 2, 3];  
console.log(arrayExample.length); // 3  
  
// Set  
const setExample = new Set([1, 2, 3]);  
console.log(setExample.size); // 3

- How can you convert an object to an array in JavaScript? Provide an example.

Answer: To convert an object to an array in JavaScript, you can use the Object.entries() method. This method returns an array of a given object's own enumerable property [key, value] pairs. Here's an example:

const person = {  
 name: 'John',  
 age: 30  
};  
const array = Object.entries(person);  
console.log(array); // Output: [['name', 'John'], ['age', 30]]



const findOnesAndTwos = (array) => {  
 const sum = array.reduce((acc, num) => acc + num, 0);  
 console.log(`Total 1s are: ${sum}`);  
 console.log(`Total 0s are: ${array.length - sum}`);  
};  
  
findOnesAndTwos([1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0])  
// Total 1s are: 7  
// Total 0s are: 6