Array, Set and Map

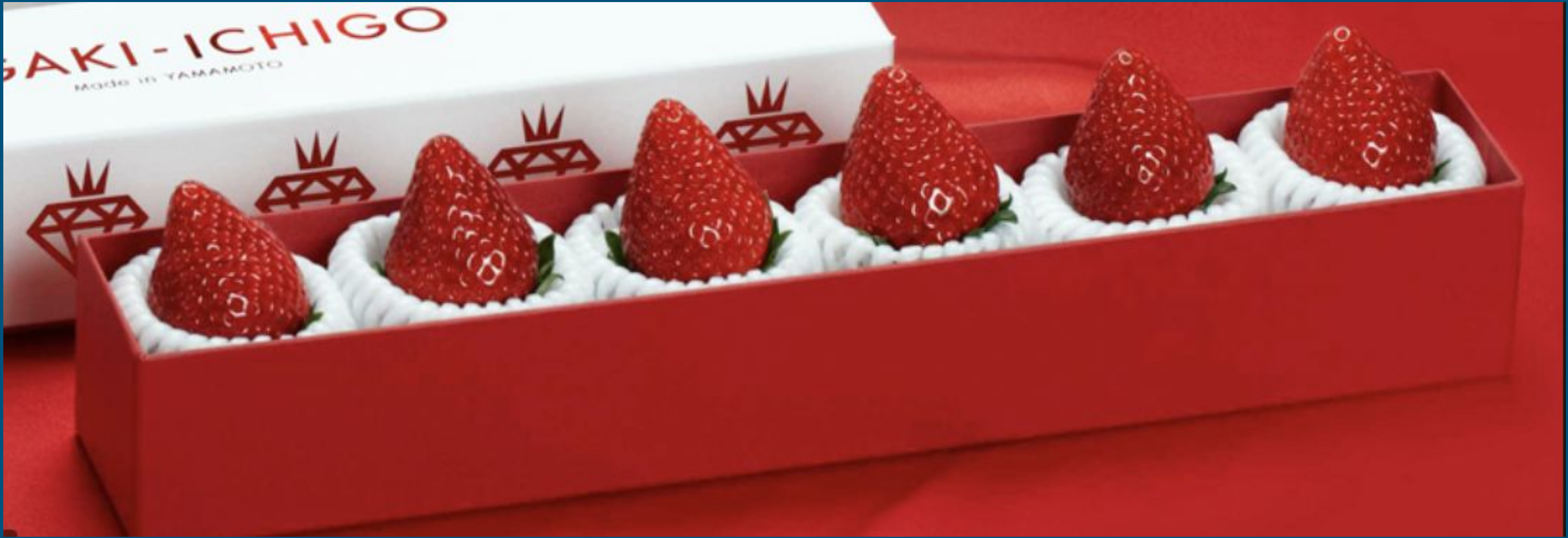
By CODEMIND Technology

Contact us 966 5044 698
966 5044 598

Array in JavaScript

- Basic of Array
- Array methods
- Array traversing
- Array Programs and assignments

One Container with multiple fruits



Array in JavaScript

- Array is reference or non primitive data type which can store multiple values
- Array allows duplicate elements
- Each value in an array has a numeric position, known as its index and it starts from 0
- Array can contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays

Syntax:

```
let arrayName = [ element1, element2, element3, elementN ];
Or
let arrayName = new Array[ element1, element2, element3, elementN ];
```

Note: It is a common practice to declare arrays with the const keyword.

What is typeof Array? What is reindexing?

Accessing and Updating array element

Accessing array element

Syntax:

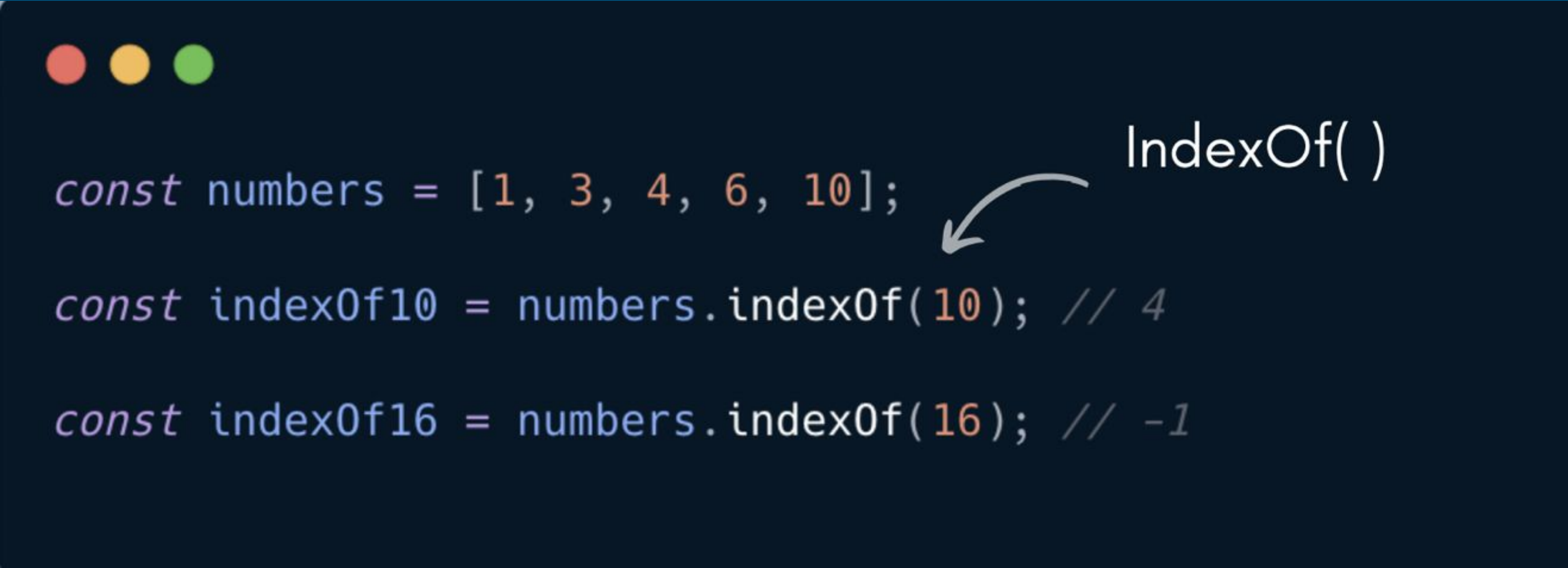
```
arrayName[indexValue];
```

Changing or Updating an array element

```
arrayName[indexValue] = “new value”;
```

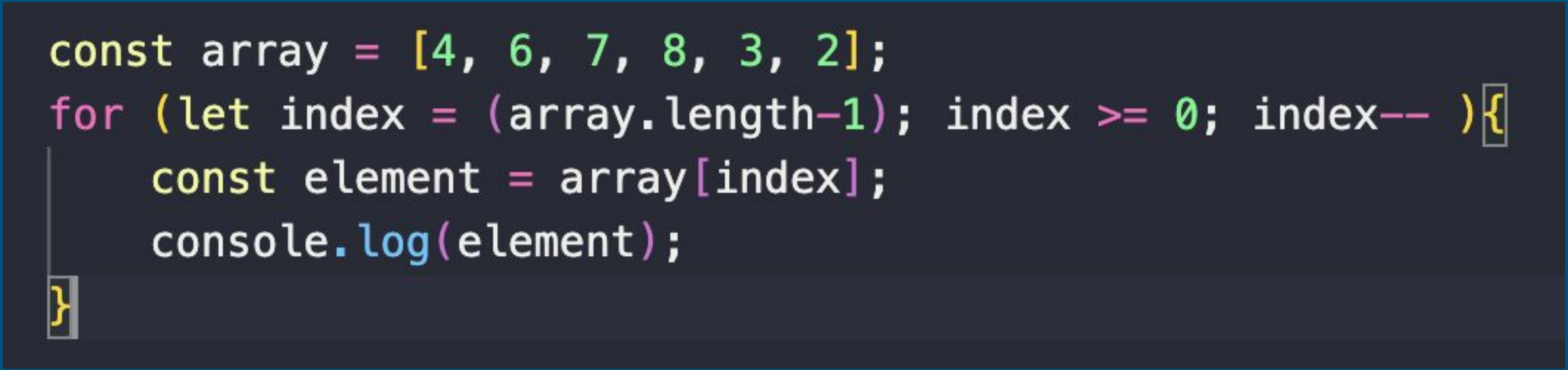
indexOf() method

The indexOf( ) method returns the index of the first occurrence of the substring or element that we specify as the argument



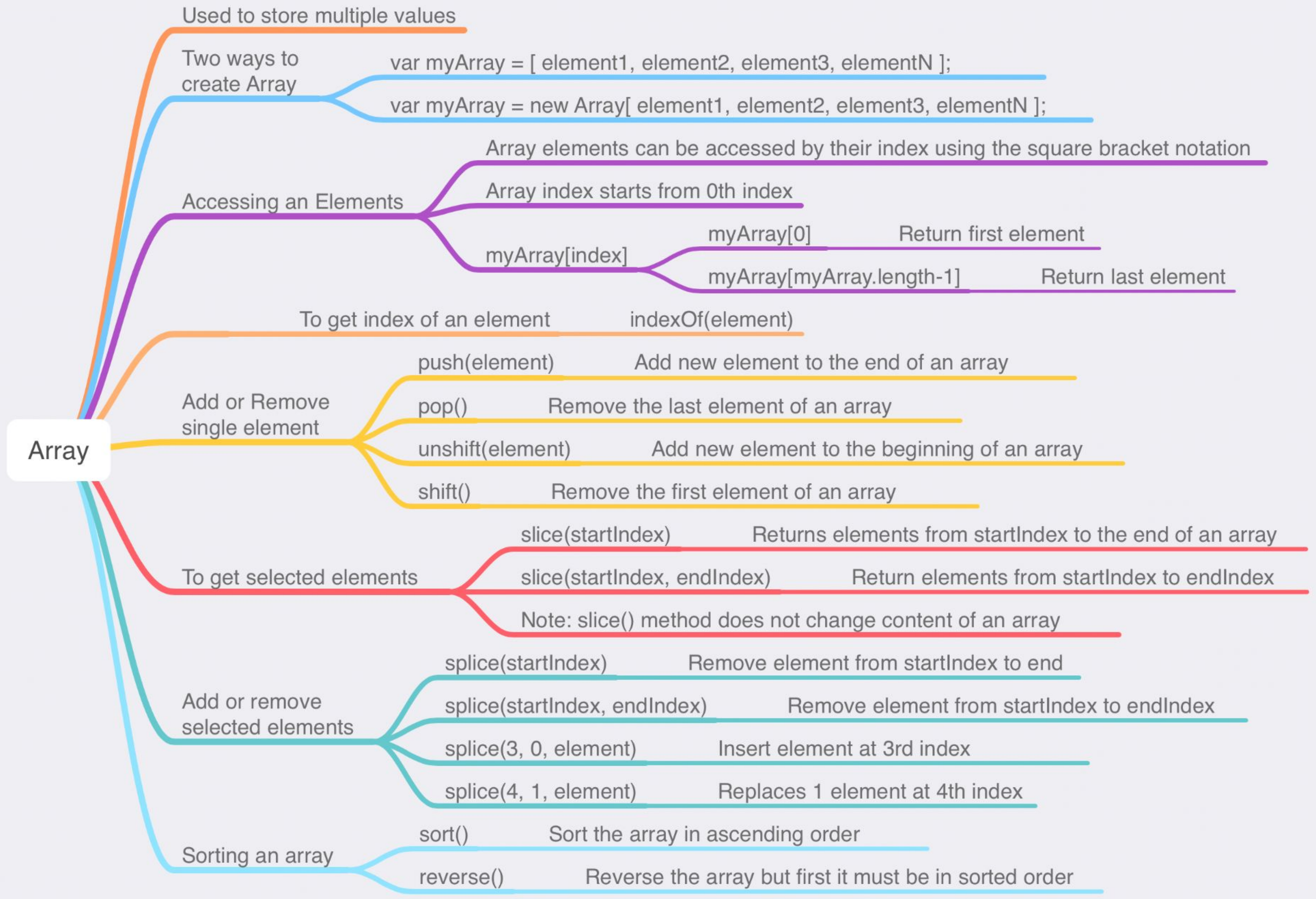
```
const numbers = [1, 3, 4, 6, 10];

const indexOf10 = numbers.indexOf(10); // 4

const indexOf16 = numbers.indexOf(16); // -1
```

Traverse the array in reverse order ?



```
const array = [4, 6, 7, 8, 3, 2];
for (let index = (array.length-1); index >= 0; index-- ){
    const element = array[index];
    console.log(element);
}
```

WAP to find the even positioned value ?

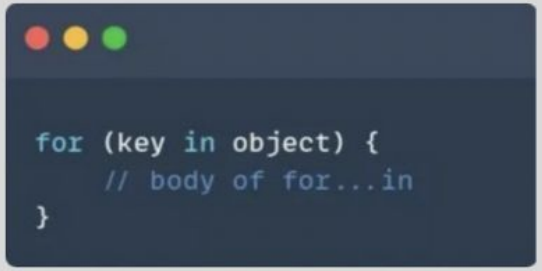


- Array
  - Used to store multiple values
  - Two ways to create Array
    - var myArray = [element1, element2, element3, elementN];
    - var myArray = new Array[element1, element2, element3, elementN];
  - Accessing an Elements
    - Array elements can be accessed by their index using the square bracket notation
    - Array index starts from 0th index
    - myArray[index]
      - myArray[0] — Return first element
      - myArray[myArray.length-1] — Return last element
  - To get index of an element — indexOf(element)
  - Add or Remove single element
    - push(element) — Add new element to the end of an array
    - pop() — Remove the last element of an array
    - unshift(element) — Add new element to the beginning of an array
    - shift() — Remove the first element of an array
  - To get selected elements
    - slice(startIndex) — Returns elements from startIndex to the end of an array
    - slice(startIndex, endIndex) — Return elements from startIndex to endIndex
    - Note: slice() method does not change content of an array
  - Add or remove selected elements
    - splice(startIndex) — Remove element from startIndex to end
    - splice(startIndex, endIndex) — Remove element from startIndex to endIndex
    - splice(3, 0, element) — Insert element at 3rd index
    - splice(4, 1, element) — Replaces 1 element at 4th index
  - Sorting an array
    - sort() — Sort the array in ascending order
    - reverse() — Reverse the array but first it must be in sorted order

for in loop

• The for...in loop in JavaScript allows you to iterate over all property keys of an object.

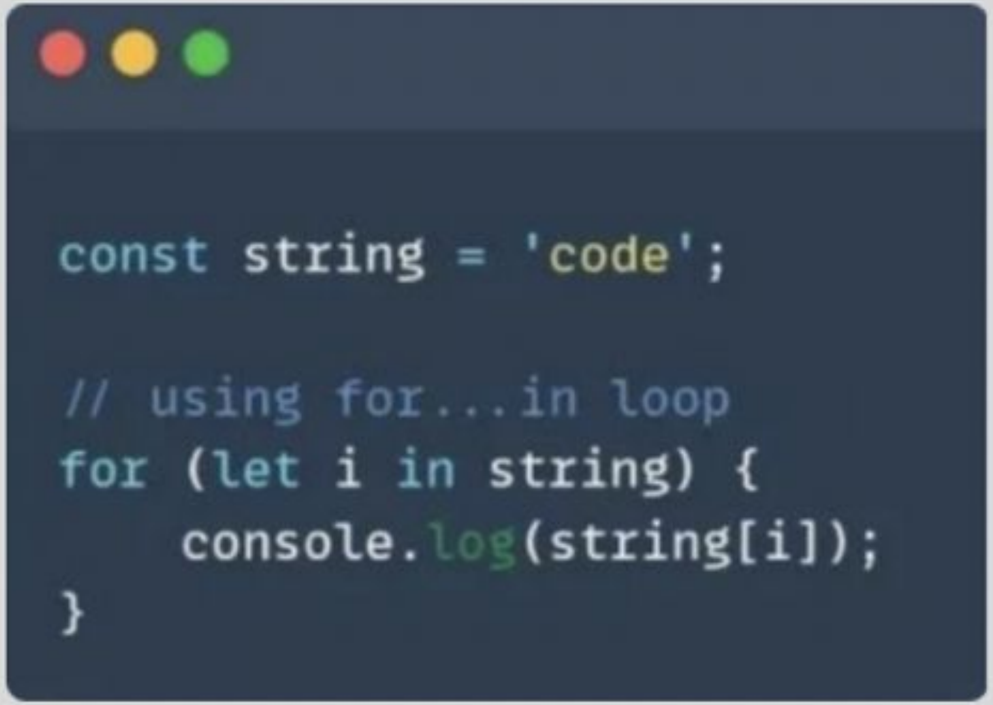
• The syntax of the for...in loop is:



```
for (key in object) {
    // body of for...in
}
```

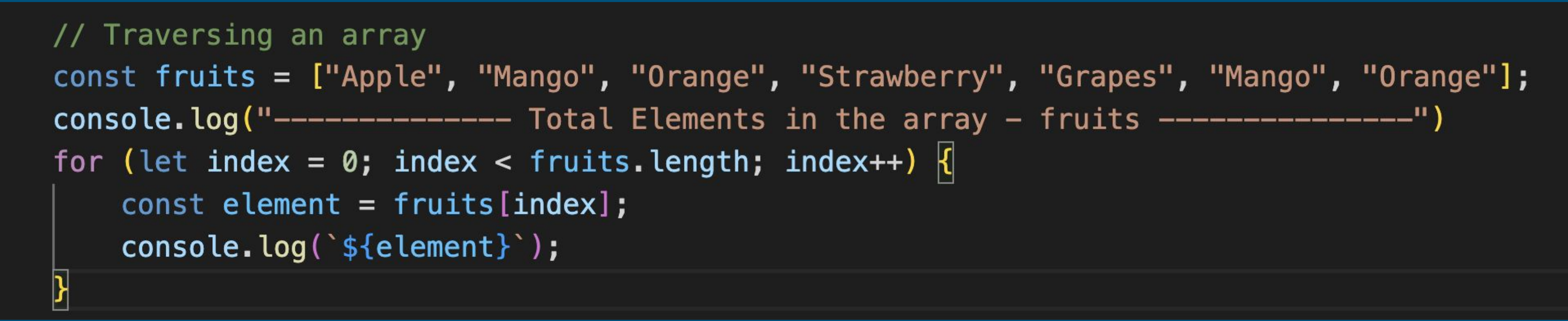
• In each iteration of the loop, a key is assigned to the key variable.

- for…in loop to iterate over string values.



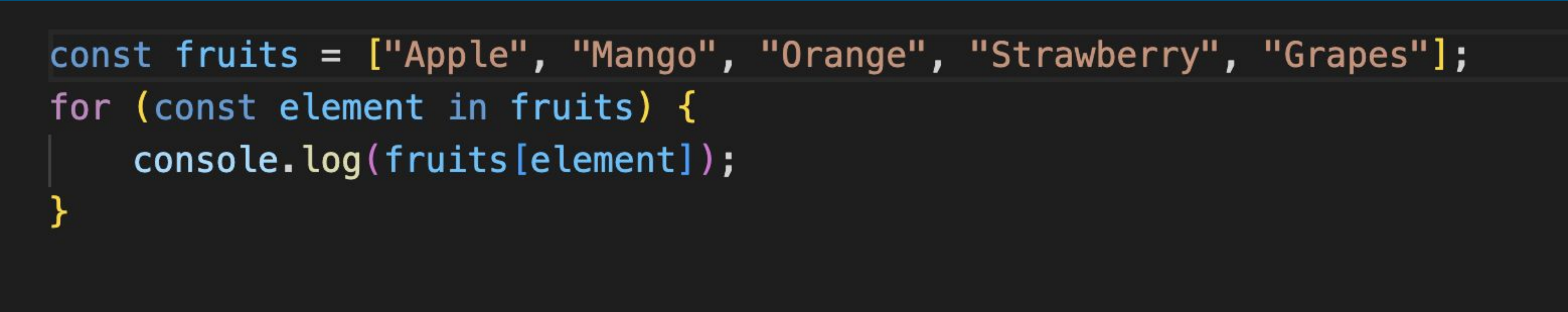
```
const string = 'code';

// using for...in loop
for (let i in string) {
    console.log(string[i]);
}
```

for loop: To traverse over an array using traditional for loop



```javascript
// Traversing an array
const fruits = ["Apple", "Mango", "Orange", "Strawberry", "Grapes", "Mango", "Orange"];
console.log("-------------- Total Elements in the array - fruits ---------------")
for (let index = 0; index < fruits.length; index++) {
    const element = fruits[index];
    console.log(`${element}`);
}
```

for in loop: To traverse over an array



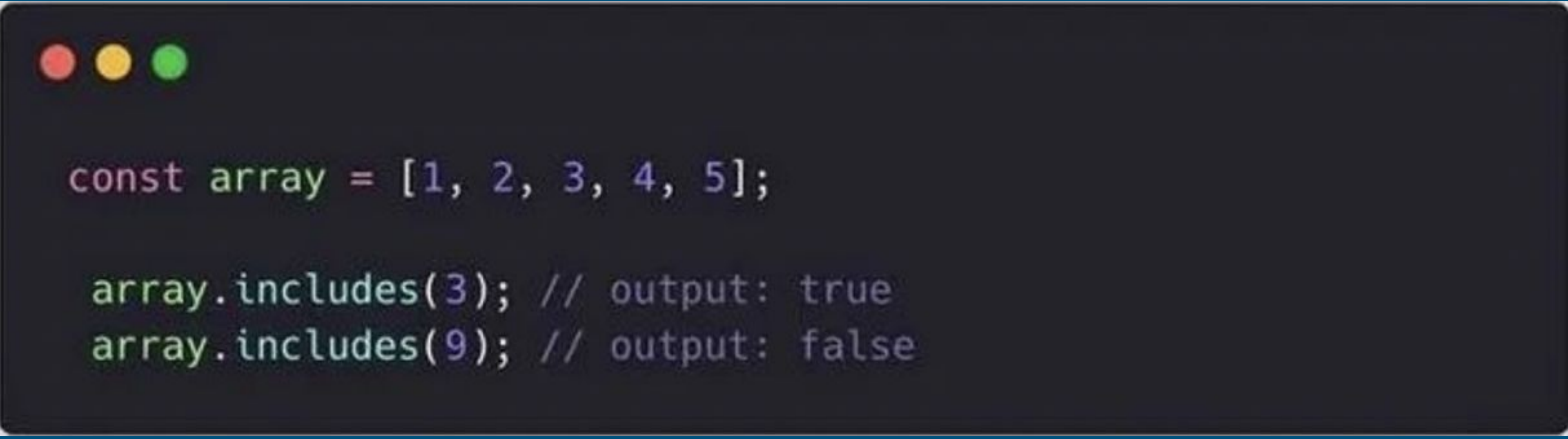
```javascript
const fruits = ["Apple", "Mango", "Orange", "Strawberry", "Grapes"];
for (const element in fruits) {
    console.log(fruits[element]);
}
```

What are the different data types of values we can add in array ?

- Number
- String
- Boolean
- Object

includes() method

It check whether the array contains the given value or not



```
const array = [1, 2, 3, 4, 5];

 array.includes(3); // output: true
 array.includes(9); // output: false
```

What is iterable

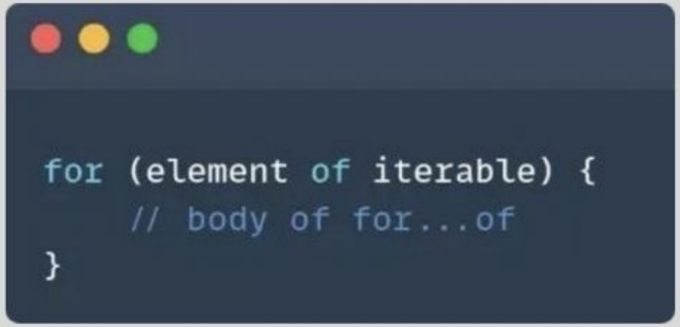
Iterable is an object which can be looped over or iterated over with the help of a for loop.

String, Array, Set, Map these are all built-in iterables, because each of their prototype objects implements an @iterator method.

for of loop

- The for...of loop in JavaScript allows you to iterate over iterable objects.

- The syntax of the for...of loop is:



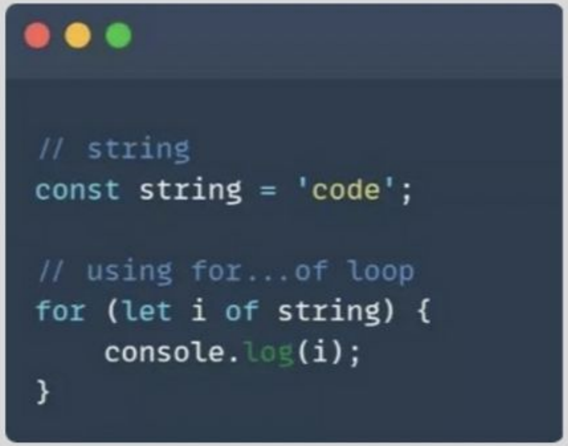
```
for (element of iterable) {
    // body of for...of
}
```

- iterable - an iterable object (array, set, strings, etc).
- element - items in the iterable.

For of loop to traverse over an array

```
let fruits = ["Apple", "Mango", "Orange", "Strawberry", "Grapes"];
for (const element of fruits) {
    console.log(element);
}
```

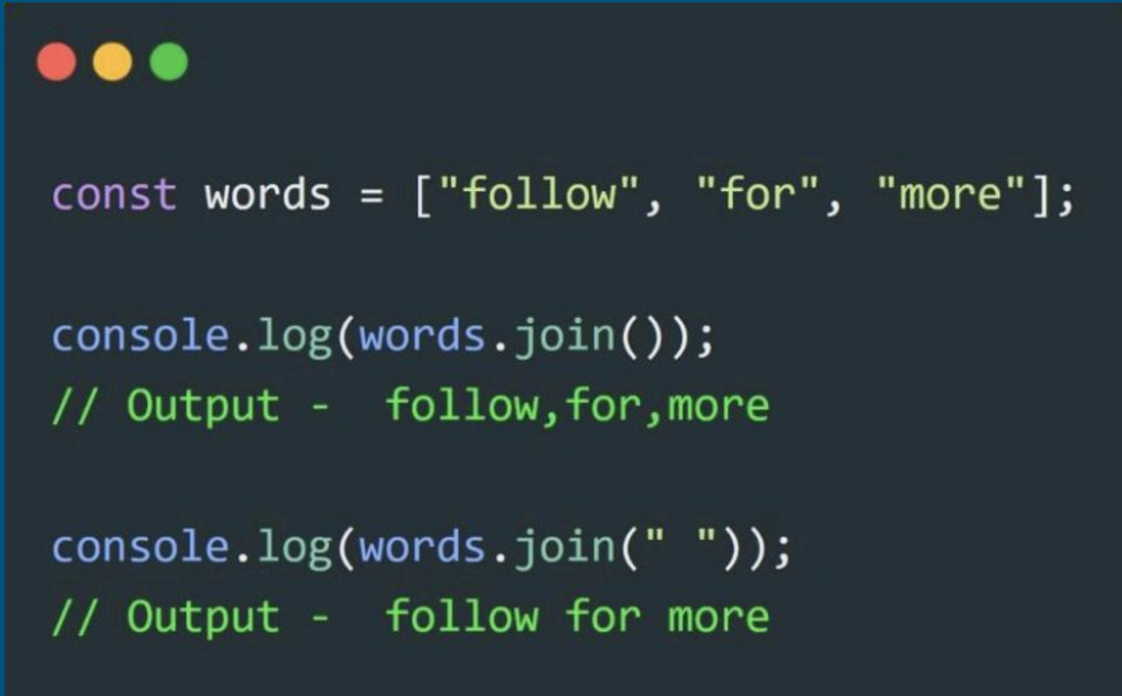
- for...of loop to iterate over string values.



```
// string
const string = 'code';

// using for...of loop
for (let i of string) {
    console.log(i);
}
```

join( ) method

- join() methods returns an array as a string & does not change the original array
- The default separator is a comma (,) separator, We can also specify any separator



```
const words = ["follow", "for", "more"];

console.log(words.join());
// Output -  follow,for,more

console.log(words.join(" "));
// Output -  follow for more
```

concat() method

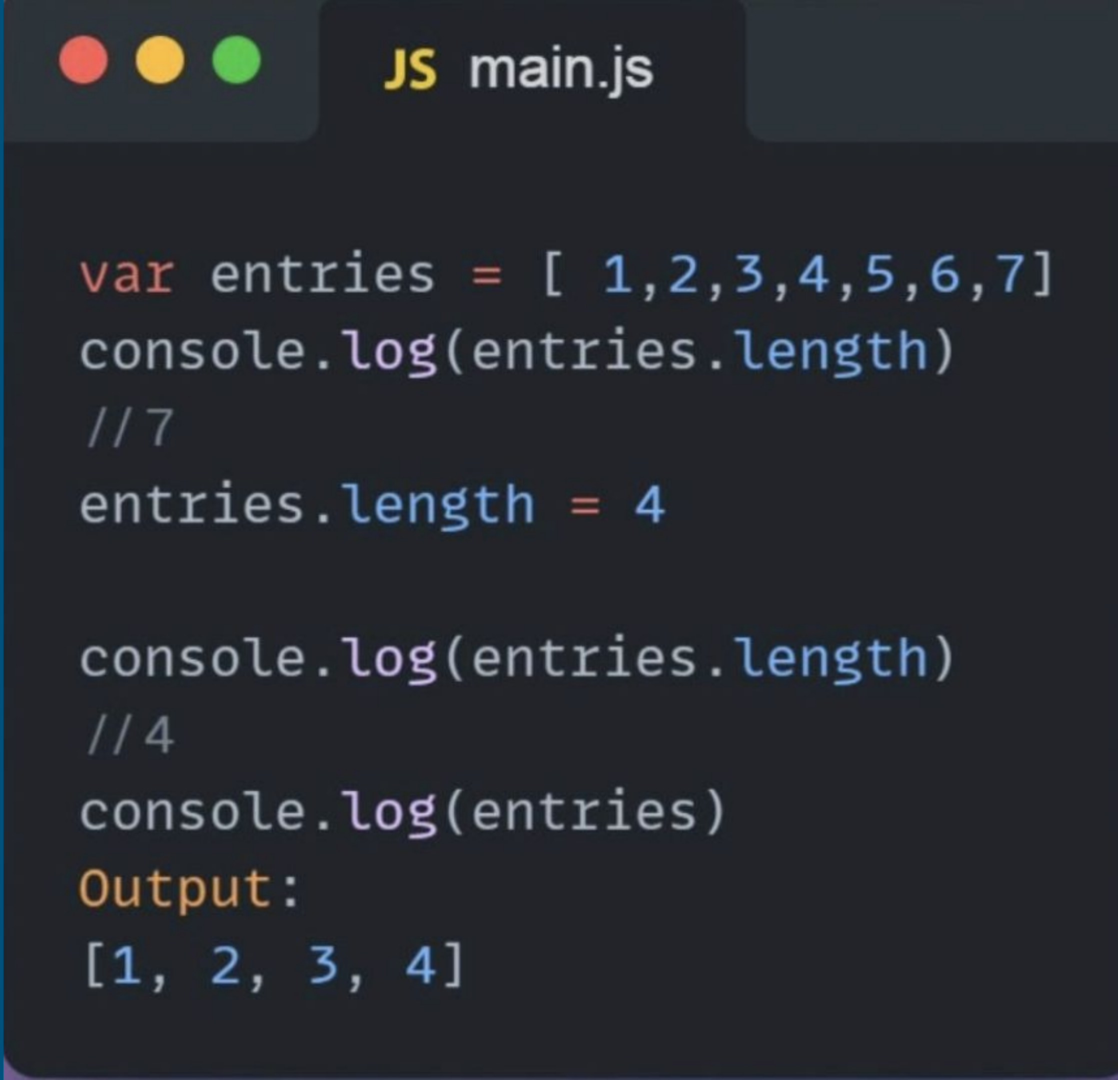
This method is used for concatenation

1. This method concat two strings
2. This method can be used to concat or merge two arrays

Example:

```
let arr1= [1,2,3];
let arr2 = [4,5];
let arr3 = arr1.concat(arr2);
console.log(arr3); // [ 1, 2, 3, 4, 5 ]
```

Resize an array



main.js

```
var entries = [ 1,2,3,4,5,6,7]
console.log(entries.length)
//7
entries.length = 4

console.log(entries.length)
//4
console.log(entries)
Output:
[1, 2, 3, 4]
```

How to add Elements in an Array

- At the time of array creation we can add elements
    - const myArray = [“A”, “B”, true, 1, 2, 3]
- Also we can use push( ) or splice( ) method
    - const array = [ ];
    - array.push(23); or   array.splice();

Array: Performance

- Why push(), pop() methods are faster than shift() and unshift()?
- What is reindexing?

The push() and pop() methods runs faster than unshift() and shift(). Because push() and pop() methods simply add and remove elements at the end of an array therefore the elements do not move, whereas unshift() and shift() add and remove elements at the beginning of the array that require re-indexing of whole array

Assignment 01: File→ 10_array_assigA.js

const arrayFruits = ["Banana", "Orange", "Apple", "Mango", “Water Melon”];

For a given array fruits perform below operations as:

1. Log the first and last element on console
2. Add element → “Papaya” before the element ‘Banana’ and then log array on console
3. Remove ‘Mango’ from the array
4. Add element or insert an element ‘Pineapple’ at the last position
5. Insert an element - ‘Dragon Fruit’ before “Water Melon”
6. Replace an element ‘Orange’ with ‘Kiwi’
7. Log the elements starting from index 1 to 4
8. Only select last 3 element and log on console: Use the length property

Assignment: File→10_Arrayassig02.js Please before posting on whatsapp group verify your result once.

const arrayNumbers = [20, 31, 40, 25, 23, 11, 29, 9, 60, 2, 11];

1. Find the total elements available or length and log on console
2. Log the first element and last element in arrayNumbers and log on console
3. Log the thirst last element using length property and log on console
4. Find the all even numbers using for in loop and log on console
5. Find the odd numbers for in loop and log on console
6. Find all the even positioned elements from arrayNumbers, sum it and log on console
7. Find all the odd positioned elements from arrayNumbers, sum it and log on console
8. Find the sum of all elements from arrayNumbers, log on console
9. Find the numbers which are multiple of 5
10. Is number 115 available in arrayNumbers ?
11. Is number 23 available in arrayNumbers ?
12. Insert numbers → 55, 66 before index 3 and log array on console
13. Delete 3 elements starting from index 4 and log arrayNumbers on console