Data Structures: Arrays

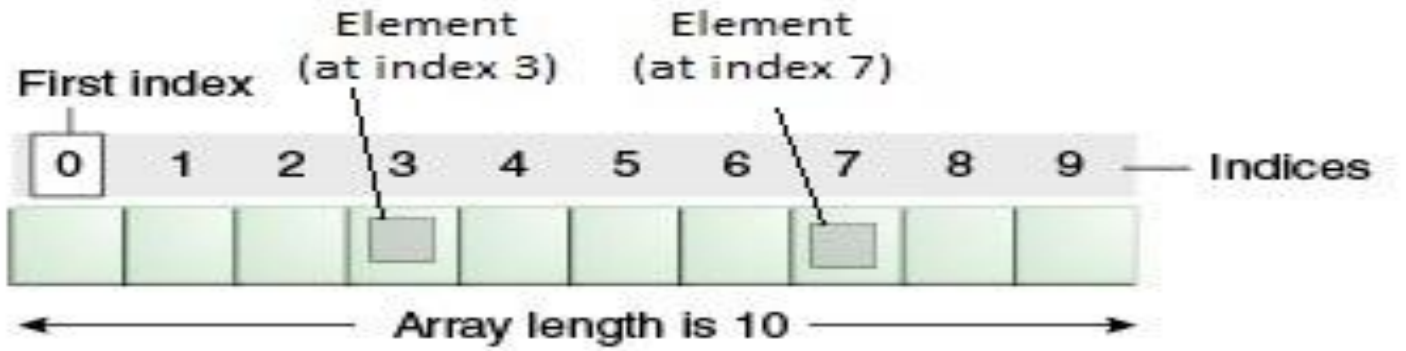
Relevel
by Unacademy

Concepts

- What is an Array?
- Advantages & Disadvantages of an array.
- Use of Array & Basic rules for Array Declaration.
- Types of Arrays.
- Ways to create an array
- Looping through an array
- Destructuring in Javascript.
- Array Destructuring.
- Rest parameters.
- Functions on arrays.

Arrays

An array is described as a group or collection of items (numbers) stored inside memory in a logical order.



- The main function of Array is to store a collection of homogenous data of the same type.
- In an array, all items are kept in a single memory region.
- All arrays are allocated dynamically ; it provides **a variable-size list data structure,** allowing elements to be removed or added in an array**.**
- An array is a data structure where we save comparable objects.
- An array is index-based, with the first element, or the first element in the Array, stored at the 0th index, the second element at the 1st index, the third element at the 2nd index, and so on.
- An array can be used to store primitive values or objects in JavaScript.
- We can create both single-dimensional and multidimensional arrays.

Example 1 (Without Array)

In the below example, we will repeat the same code with different values without using an array.

```
// Here, we are creating five variables to store five different cities
 const city1 = 'London';
const city2 = 'Mumbai';
const city3 = 'Chennai';
const city4 = 'Bangalore';
const city5 = 'Kerala';
// To print all the elements to the console
console.log(city1); // London
console.log(city2); // Mumbai
console.log(city3); // Chennai
console.log(city4); // Bangalore
console.log(city5); // Kerala
```

Example 2 (With Using Array)

Here in this example, we will run the same code using an Array.

```
// To create an array of cities to store values
const cities = ['London', 'Mumbai', 'Chennai', 'Bangalore', 'Kerala']
// To print all the elements entered in a array to the console
for(let i = 0; i < cities.length; i++){
    console.log(cities[i]);
}
 London
 Mumbai
 Chennai
 Bangalore
 Kerala
```

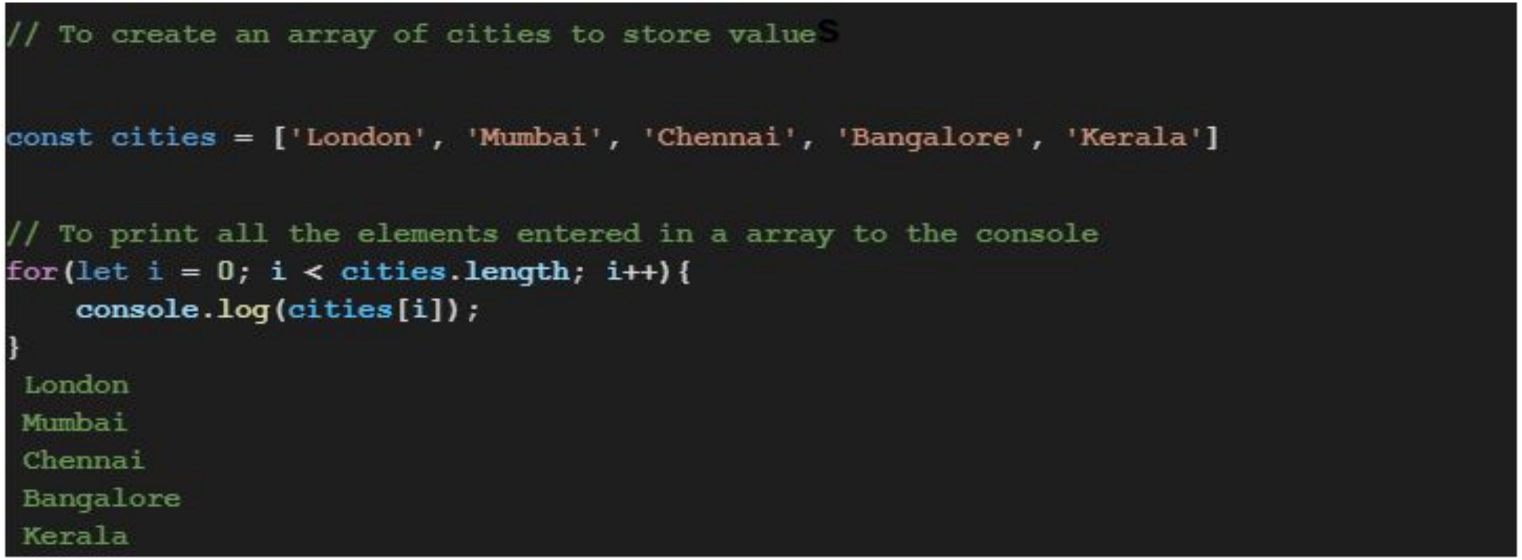
Difference between with and without array

The main difference between with Array and Without Array are,

1) Without array each values are stored in different variables which result in different memory location, whereas using array the values are stored in contiguous memory allocation
2) For developer we can easily maintain code and data manipulation is good for larger data, without array we need to create 100 variable for 100 values but in array we can insert the values in the single array

Arrays

Declare + Initialize Array : let array_names = [“QPR”,”ABD”,IJP”,LOP”]



```
// To create an array of cities to store values

const cities = ['London', 'Mumbai', 'Chennai', 'Bangalore', 'Kerala']

// To print all the elements entered in a array to the console
for(let i = 0; i < cities.length; i++){
    console.log(cities[i]);
}
 London
 Mumbai
 Chennai
 Bangalore
 Kerala
```

Basic rules for Array Declaration

- In a JavaScript array, a variable can be declared with [] after the data type.
- An array can be created using array literal or Array constructor syntax.
- Array literal syntax: var stringArray = ["one", "two", "three"];
- Array constructor syntax: var numericArray = new Array(3);

Arrays

Types of Array

Homogeneous Arrays : In homogenous array only single type of data is stored in array

```
let array = [“Jacob", “John", “Peter"];
```

```
let array = [27, 24, 30];
```

Heterogeneous Arrays : In heterogeneous array mixed data types can be used to store in array

let array = [“RANDOM”,1889,true,112]

Arrays

Types of Array

Multi-dimensional Arrays : An array which has an array stored within another array is called multidimensional array

```
var array = [["Matthew", "27"], ["Simon", "24"], ["Luke", "30"]];
```

Jagged Arrays : Jagged arrays are similar to multidimensional array with the exception being that a jagged array
does not require a uniform set of data.

```
let array = [
    ["Matthew", "27", "Developer"],
    ["Simon", "24"],
    ["Luke"]
];
```

Advantages

Arrays represent numerous data items of a similar type using a single name.
In arrays, the elements can be accessed in any order by using the index number.
Arrays allocate memory in contiguous memory locations for every element present in an array.
Henceforth there is no possibility of additional memory being allocated in the case of arrays. This saves memory spillage or overflow of memory and helps to regulate memory usage in any database.
we can also search for a particular element from array by traversing it.

Arrays Keeps the data in the well organized way.

Disadvantages

Size Limit: In the Array, we can only store elements of a fixed size. It does not expand in size during use.
If the size requirement is less then memory wastage is on the card. For example, you declared array of size 100 but later found out that you only needed to store 82 elements so its a complete wastage of 18 elements storage space. Because when you declared array with size 100, memory space to store 100 element was already blocked on disk.

Array is homogenous in nature which means element of only one data type can be stored in one array. But in real life scenarios, we may be required to store elements of different types. For example, details about student such as name (String), age(integer), subject(String), etc. But array doesn't have such facility.

Deletion of any element in the array will result in more time, for a worst case if you want to remove 1st element then it will shift all the element so it will take more time.

Strings

Strings are sequence of characters that are enclosed in single or double quotes

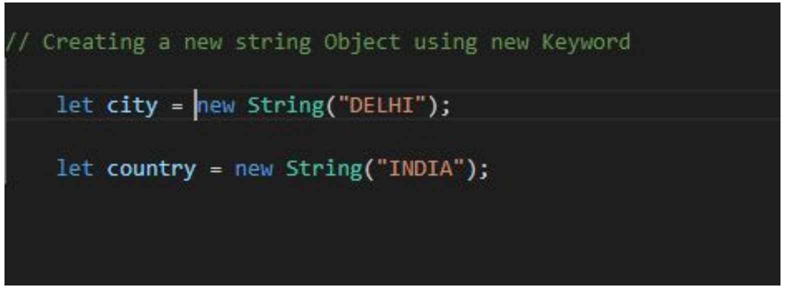
Eg : “Javascript”,”React js”,”Node js”

How to create Strings in Js :

String Literals - The easiest and the most common way of creating a String is using String Literals .Eg:

Let city =”NEW DELHI”

New Keyword - Using the Javascript new keyword we can create a new String object .**Eg :**



```
// Creating a new string Object using new Keyword

  let city = new String("DELHI");

  let country = new String("INDIA");
```

Different ways to create Arrays in JavaScript

In Javascript, there are many ways in which an array can be created

Array Literals - The easiest and the most common way of creating an array is using Array Literals.

Example:

```
const cities = ["London"," Mumbai"," Orange"," Cherry"]
```

New Keyword - Using the new Javascript keyword, we can create a new array.

Example:

```
 const cities = new Array("London"," Mumbai"," Cherry"," Kerala ")
```

Common Operations performed on Arrays :

Length of array :

```
const cities = [“London”,”Mumbai”,” Bangalore”,” Kerala"]
 let length =cities.length
 console.log(length) ------------------> 4
```

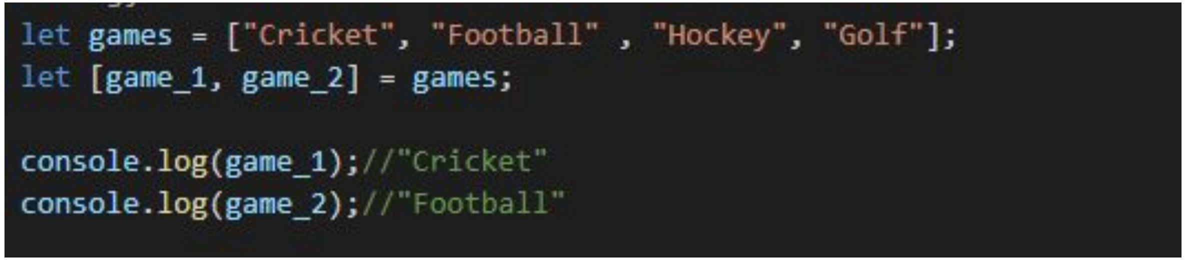
Note → Array index always starts from 0, which means the first element is stored at index 0 and the last element will be stored at index length-1.

Destructuring in Javascript:

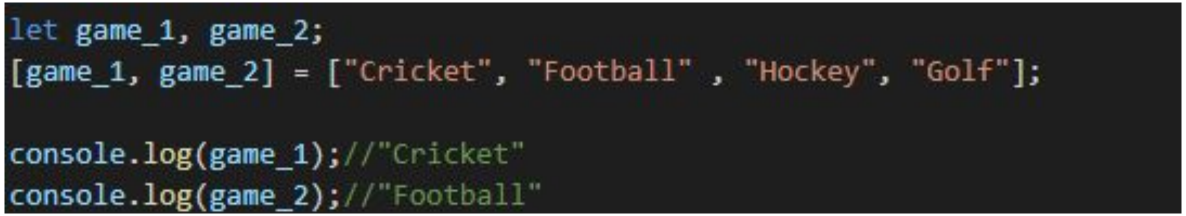
Destructuring in JavaScript is an expression that makes it possible to unpack values from arrays, or properties from objects. We can extract data from arrays and objects and assign them to distinct variables.The value that should be unpacked from the sourced variable is defined on the left-hand side.

Let us consider We have a array of 10 elements and we need to get the values of 1st two elements and store it in a variable so here we assign a variable with a similar format of data structure.

Array Destructuring

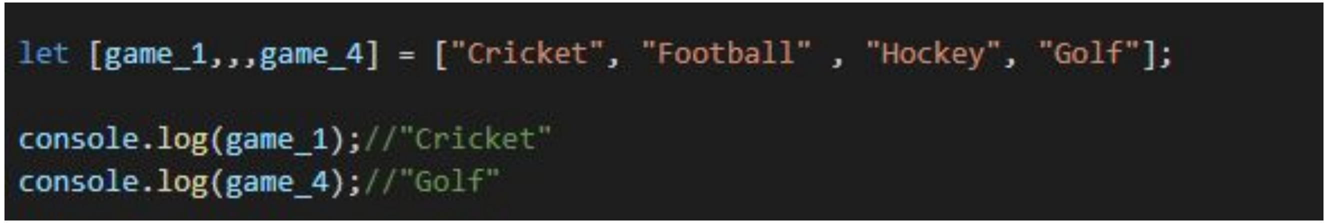


```javascript
let games = ["Cricket", "Football" , "Hockey", "Golf"];
let [game_1, game_2] = games;

console.log(game_1);//"Cricket"
console.log(game_2);//"Football"
```



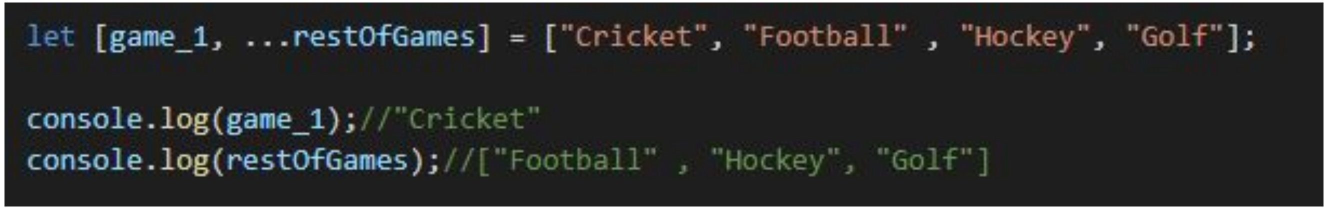
```javascript
let game_1, game_2;
[game_1, game_2] = ["Cricket", "Football" , "Hockey", "Golf"];

console.log(game_1);//"Cricket"
console.log(game_2);//"Football"
```

Skipping Elements in an Array:



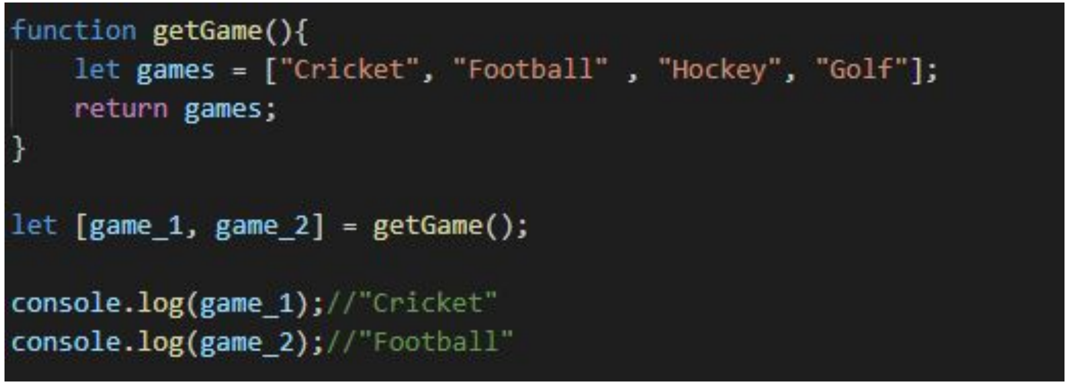
```
let [game_1,,,game_4] = ["Cricket", "Football" , "Hockey", "Golf"];

console.log(game_1);//"Cricket"
console.log(game_4);//"Golf"
```

Assigning rest of the Array:



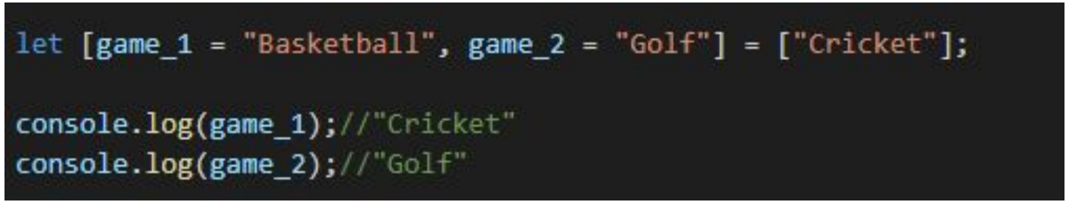
```
let [game_1, ...restOfGames] = ["Cricket", "Football" , "Hockey", "Golf"];

console.log(game_1);//"Cricket"
console.log(restOfGames);//["Football" , "Hockey", "Golf"]
```

Destructuring assignment Using Functions:



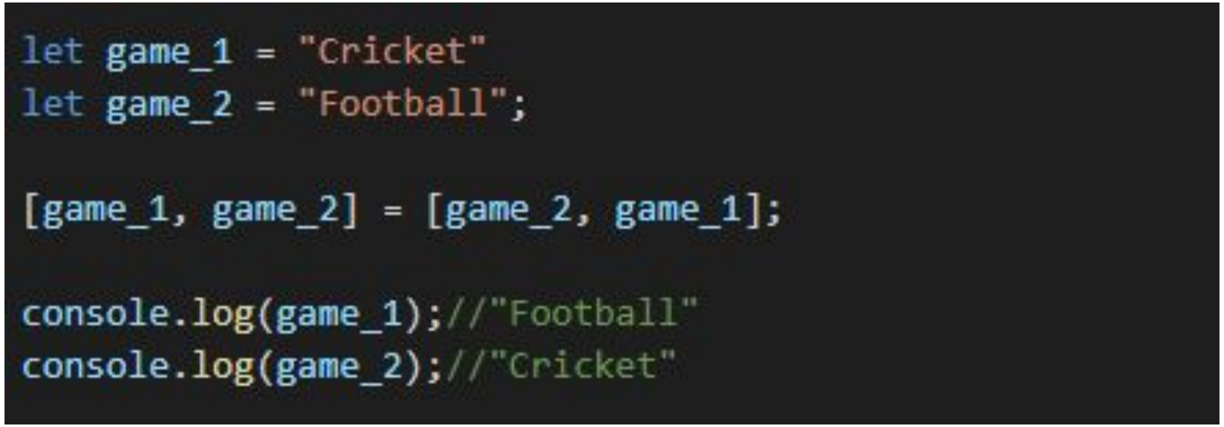
```
function getGame(){
    let games = ["Cricket", "Football" , "Hockey", "Golf"];
    return games;
}

let [game_1, game_2] = getGame();

console.log(game_1);//"Cricket"
console.log(game_2);//"Football"
```

Using Default Values:



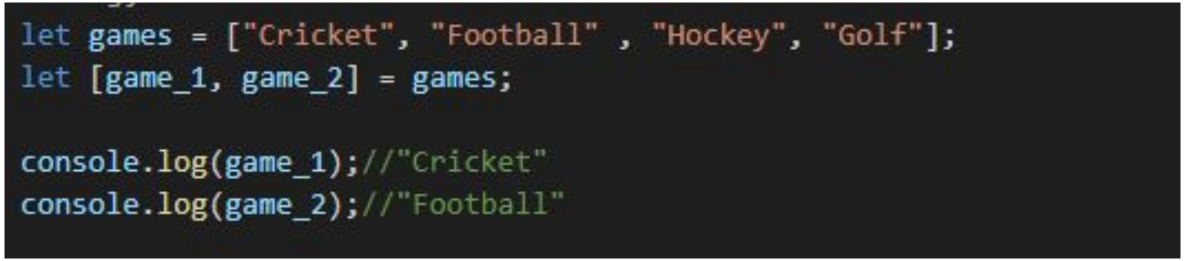
```
let [game_1 = "Basketball", game_2 = "Golf"] = ["Cricket"];

console.log(game_1);//"Cricket"
console.log(game_2);//"Golf"
```

Swapping values using Destructuring assignment:

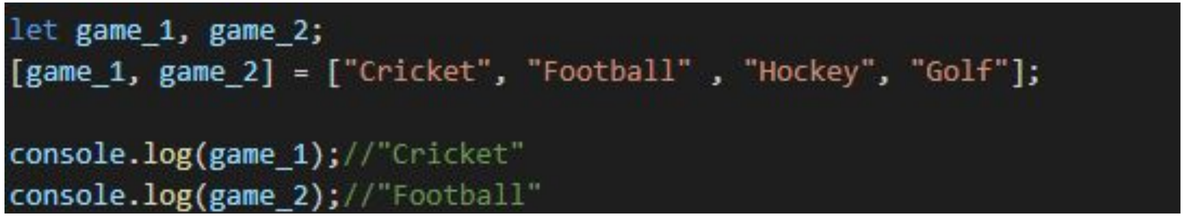


```
let game_1 = "Cricket"
let game_2 = "Football";

[game_1, game_2] = [game_2, game_1];

console.log(game_1);//"Football"
console.log(game_2);//"Cricket"
```

Array Destructuring



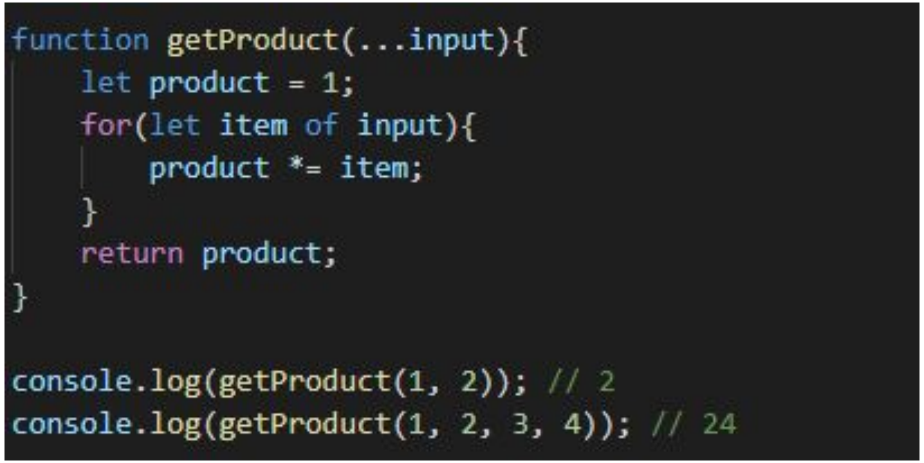
```
let games = ["Cricket", "Football" , "Hockey", "Golf"];
let [game_1, game_2] = games;

console.log(game_1);//"Cricket"
console.log(game_2);//"Football"
```



```
let game_1, game_2;
[game_1, game_2] = ["Cricket", "Football" , "Hockey", "Golf"];

console.log(game_1);//"Cricket"
console.log(game_2);//"Football"
```

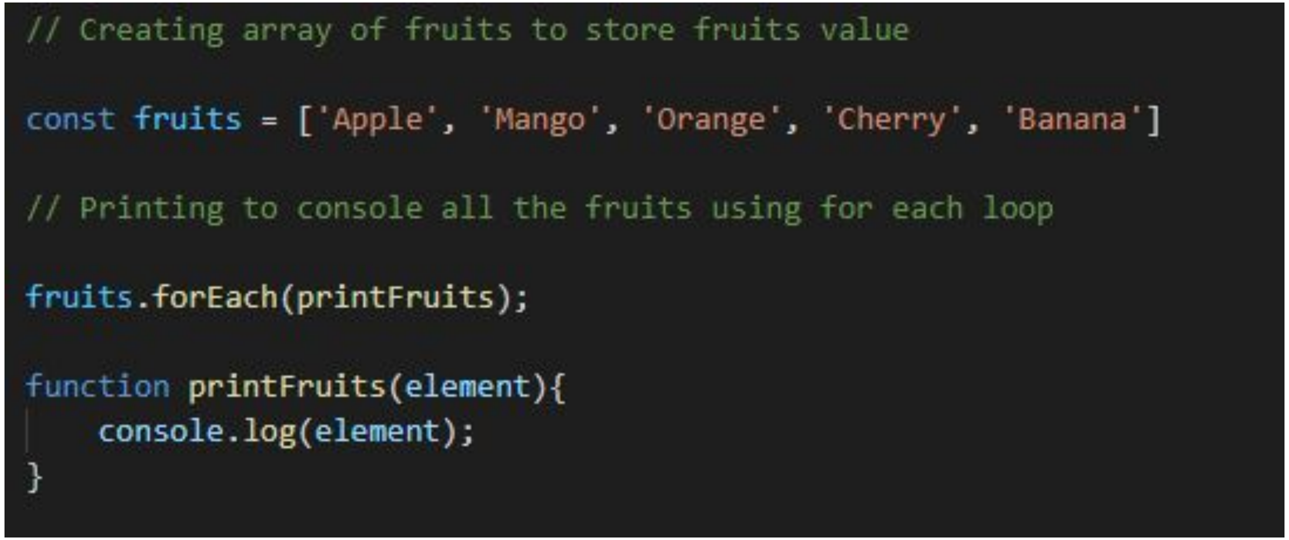
Rest Parameters:

```
Example:
     const arr = [1, 2, 3, 4, 5, 6]
     const arr1 = [...arr, 7]
     console.log(arr1) // [1, 2, 3, 4, 5, 6, 7]
```



```
function getProduct(...input){
    let product = 1;
    for(let item of input){
        product *= item;
    }
    return product;
}

console.log(getProduct(1, 2)); // 2
console.log(getProduct(1, 2, 3, 4)); // 24
```

forEach() Loop:



```
// Creating array of fruits to store fruits value

const fruits = ['Apple', 'Mango', 'Orange', 'Cherry', 'Banana']

// Printing to console all the fruits using for each loop

fruits.forEach(printFruits);

function printFruits(element){
    console.log(element);
}
```

Explanation forEach() function

- Here `const fruits` is an array that contains five elements in it, we have invoked *`fruits.forEach()`* , in this function fruits is the name of the array, by the use of forEach() function all elements of the array are called one by one from the 0th element in the array until it reaches the last element of the array.

- The forEach() method cannot be executed for an array of empty elements

- Now that we do have better knowledge of arrays, let's explore some more important methods such as filter() and map().

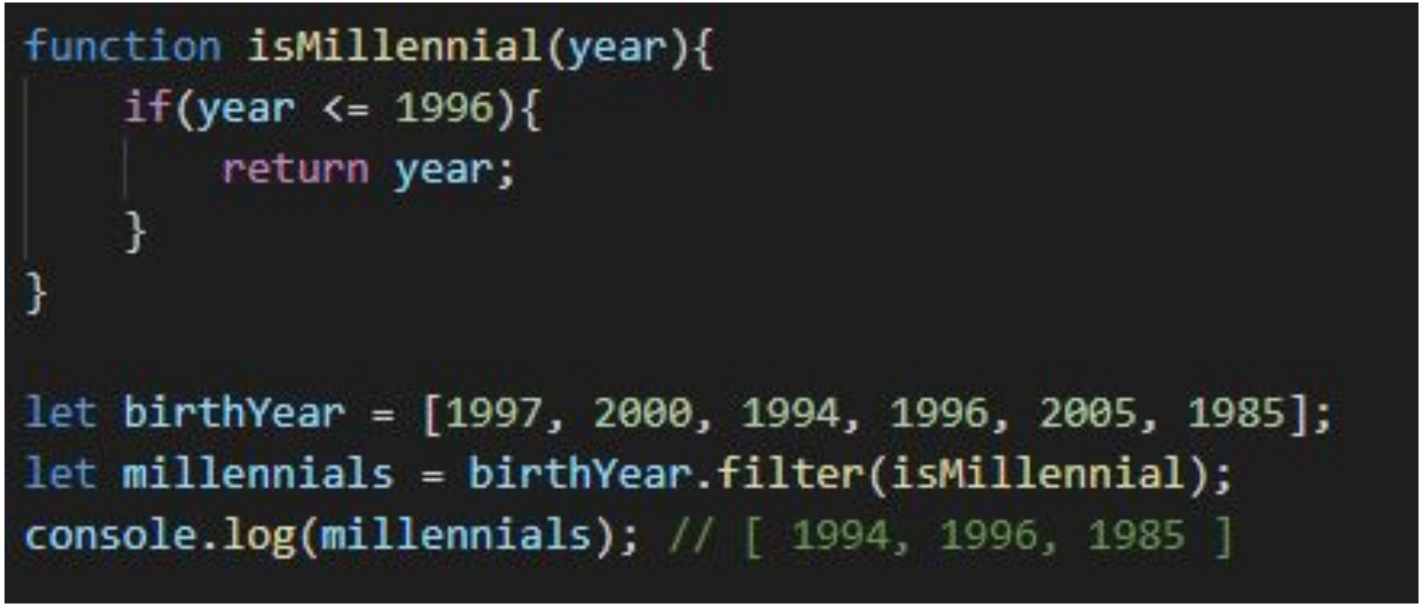
filter() method:

filter() method creates a new array from a given array which consists of only those elements from the given array which satisfy a certain test/condition.

Syntax:

```
array.filter(callback(element, index, arr), thisArg)
```

Example:



```
function isMillennial(year){
    if(year <= 1996){
        return year;
    }
}

let birthYear = [1997, 2000, 1994, 1996, 2005, 1985];
let millennials = birthYear.filter(isMillennial);
console.log(millennials); // [ 1994, 1996, 1985 ]
```

map() method:

map() method applies a function, performs the desired operation on each element of the given array, and returns a new array consisting of all the elements after applying the function. map() allows elements of the array to be manipulated as per the user's preference.

Syntax:

```
array.filter(callback(element, index, arr), thisArg)
```

Split()

split function is used to split the string and form an array based on the delimiter.
It will accept a delimiter as an argument and it will split the array based on the delimiter.
If we pass ‘,’ as an delimiter then the split will happen when the string contains ‘,’ and form an new array.

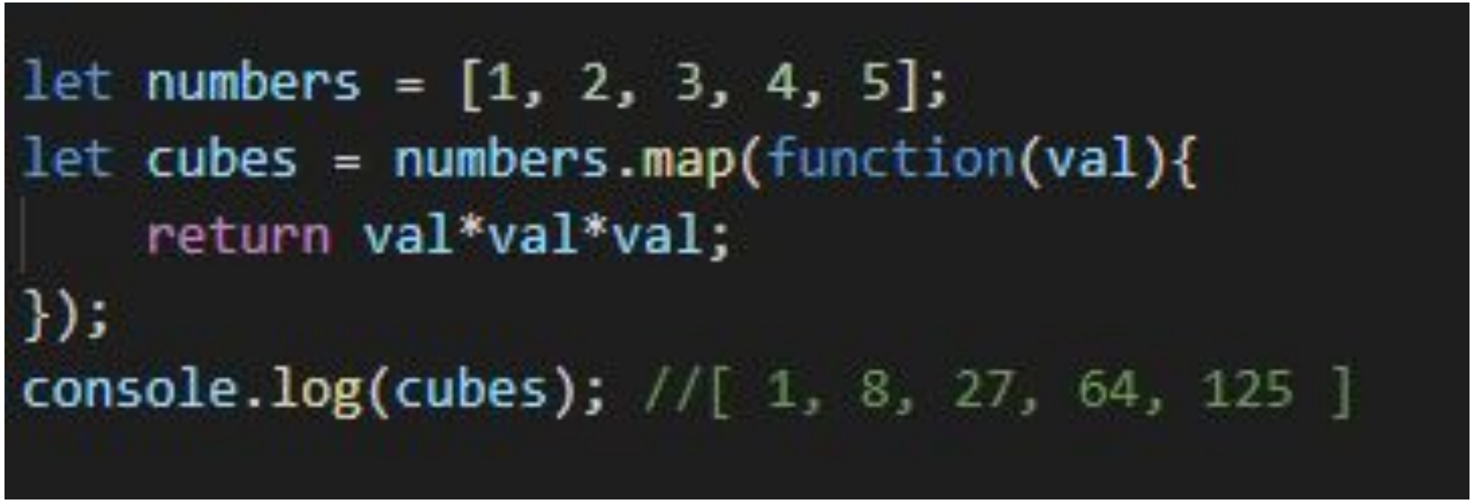
```
const str = “relevel”
const splitStr =
str.split(“”)
console.log(splitStr) //
[‘r’, ‘e’, ‘l’, ‘e’, ‘v’,
‘e’, ‘l’]
```

Slice()

The slice() method returns a copy of a portion of an array into a new array selected from start to end (end not included) where start and end represent the index of items in that array. The original array will not be modified.

```
const animals = ['ant', 'bison', 'camel', 'duck',
'elephant'];

    console.log(animals.slice(2));
    // expected output: Array ["camel", "duck",
    "elephant"]

    console.log(animals.slice(2, 4));
    // expected output: Array ["camel", "duck"]
```

Example:


```
let numbers = [1, 2, 3, 4, 5];
let cubes = numbers.map(function(val){
    return val*val*val;
});
console.log(cubes); //[ 1, 8, 27, 64, 125 ]
```

Sorting of an array of numbers

```
// Sorting of an array of numbers
let numbers = [4, 12, 2, 45, 36, 16, 700]

// Sorting from Lowest to highest
let lowestToHighest = numbers.sort((a, b) => a - b);
//Output: [2,4,12,16,36,45,700]

//Sorting from Highest to lowest
let highestToLowest = numbers.sort((a, b) => b-a);
//Output: [700,45,36,16,12,4,2]
```

Functions on arrays

Functions	Action Performed by function
asList()	This function returns a fixed-size list backed by the specified Arrays
binarySearch()	This function Searches for the definite element in the Array with the help of the Binary Search Algorithm
binarySearch(array, fromIndex, toIndex, key, Comparator)	This function Searches a range of the definite Array for the specified Object using the Binary Search Algorithm
deepEquals(Object[] a1, Object[] a2)	This function returns true if the two specified arrays are totally equal to each other.
deepHashCode(Object[] a)	This function returns a string representation of the “deep contents” of the specified Arrays.
setAll(originalArray, functionalGenerator)	setAll is used to make Sets every element of the specified Array using the generator function provided.
parallelSetAll(originalArray, functionalGenerator)	This function sets all the elements of this Array in parallel, using the provided generator function.
hashCode(originalArray)	Returns an integer hashCode of any particular array instance.

Functions on arrays

hashCode(originalArray)	Returns an integer hashCode of any particular array instance.
copyOf(originalArray, newLength)	This function Copies the identified range of the indicated Array into new Arrays.
compare(array 1, array 2)	This function Relates two arrays passed as parameters lexicographically. (Two or more strings are lexicographically equal/parallel if they are the same length and contain the same characters in the same positions.)

MCQ

Q 1 - Which of the following statements concerning JavaScript features is correct?

A - JavaScript is an interpreted, lightweight programming language.
B - JavaScript is a scripting language for building networked applications.
C - JavaScript is an add-on to Java that 's integrated with it.
D - All of the above.
Ans: D

Q 2 - How can you find out how many arguments were supplied to a function?

A - Making use of the args.length property.
B - Using the length property of arguments
C - Both of the previous options.
D - None of the above.
Ans: B

MCQ

Q 3 - Which type of variable is only visible within the function in which it is defined?

A - stands for "global variable."
B - stands for local variable.
C - Both of the mentioned options.
D - None of the previous.
Ans: B

Q 4 - Which of the following String object functions breaks a String object into an array of strings by separating substrings?

A- search()
B- replace()
C- slice()
D- split()
Ans: D

MCQ

Q 5 - Which function is used to convert array to string?

A - array.string()
B - array.toString()
C - string(array)
D - parseString(array)

Ans: B

Practice Problem:

1. Program to demonstrate destructuring in nested objects

1. Program to demonstrate use of rest parameters

Next Class Teaser

1. What are Objects
2. JSON
3. Properties of JSON objects.
4. Difference in JSON Object and Javascript Object.
5. Looping through an object.
6. Different ways for Looping through an object.
7. Object Destructuring.
8. Destructuring in Nested Objects.
9. Rest in Object Destructuring.

Thank you