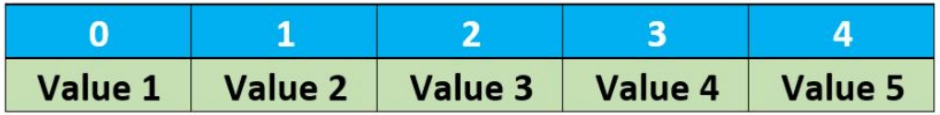
Javascript Arrays

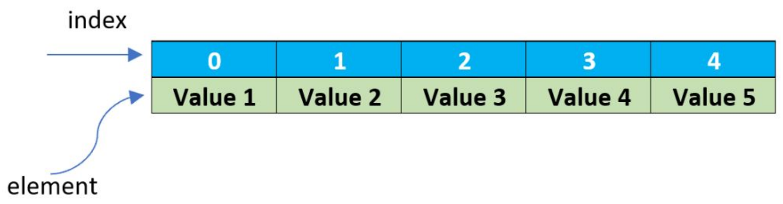


0	1	2	3	4
Value 1	Value 2	Value 3	Value 4	Value 5

JavaScript Arrays

JavaScript Arrays

- **An array** is a single variable holding a **list of elements**.
- Each **element** of the array is referenced by the **index**.
- Each **element** in the list can be **individually accessed**.
- The array can hold **mixed types of values**.



JavaScript Arrays

- The **size** of the array is **dynamic** and **auto growing**. So it is not necessary to mention the array size explicitly.

Need for an Array

- If we are supposed to work with many items, say 100 or more. It will be difficult for us to declare each item, but arrays will help us in this situation.

- We can store **many items** under a **single variable name**.

- **Values** can be accessed by referring to the **index number**.

Creating an Array

skill academy

- The Array can be created in three ways.

1. **JavaScript array literal**

Syntax

```
var arrayname=[value1,value2.....valueN];
```

Example

```
var emp=["Sonoo","Vimal","Ratan"];
```

Creating an Array

skill academy

➢ **JavaScript Array directly (new keyword)**

The **new keyword** is used to create an **instance of an array**.

Syntax

```
var arrayname=new Array();
```

Example

```
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";
```

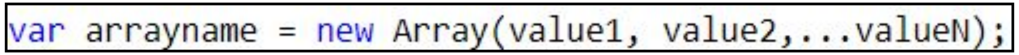
Creating an Array

skill academy

➢ **JavaScript array constructor (new keyword)**

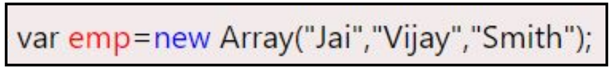
The instance of the array is created by **passing arguments** to the **constructor** instead of providing the elements explicitly.

Syntax



```
var arrayname = new Array(value1, value2,...valueN);
```

Example



```
var emp=new Array("Jai","Vijay","Smith");
```

- Among these ways, creating an array by using **array literal** is the **easiest way** to create a JavaScript Array.

- **'const'** Keyword is commonly used to **declare an array**.

Accessing Array Elements

skill academy

- We can access the **array elements** using **array indexes**.
- Array indexes always start with zero.

Syntax:

```
arrayName[index]
```

Example:

```
let car = cars[0];
```

- We can access the **full array** simply by referring to the **array name**.

Example:

```
const cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
```

Array Methods & Properties

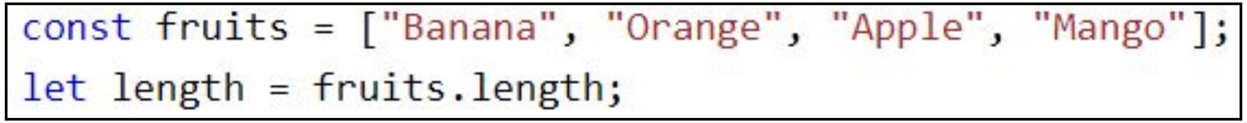
skill academy

Length of an Array

- The **length of an array** or **number of elements** in an array can be returned from the **length property** of an array.

- The length property will always return **one plus the highest array index**. Since the array **index starts from zero**.

Example



```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let length = fruits.length;
```

- The above example code will give **4** as output.

Array Methods

- JavaScript has lots of **built-in array methods**. Some important methods are listed below

Methods	Description
push()	It **adds elements** to the **end** of an array.
pop()	It **removes** and returns the **last element** of an array.
shift()	It **removes** and returns the **first element** of an array.
unshift()	It **adds elements** in the **beginning** of an array.

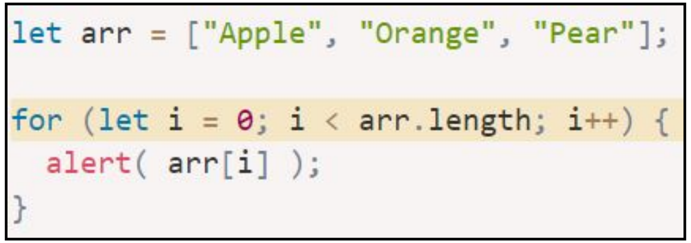
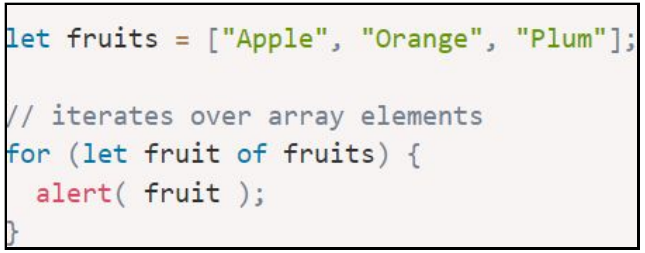
Array Methods

skill academy

Methods	Description
concat()	It returns a **new array** object that contains **merged arrays**.
sort()	It returns the **element** of the given array in a **sorted order**.
isArray()	It tests if the **passed value** is an **array**.
indexOf()	It **searches** the specified **element** in the given **array** and returns the **index** of the **first match.**

Looping Array Elements

- Only **for loop** and **array.forEach()** are used to loop through the array.
- In **forEach()** the function is called **once** for **each** element in an array.

For Loop example	ForEach Loop example
	

```
let arr = ["Apple", "Orange", "Pear"];

for (let i = 0; i < arr.length; i++) {
  alert( arr[i] );
}
```

```
let fruits = ["Apple", "Orange", "Plum"];

// iterates over array elements
for (let fruit of fruits) {
  alert( fruit );
}
```

Advantages of For each loop

- **For Each loop** makes the **code shorter** and **easier to understand**.
- **No** need to create **extra counter variable** in **for each loop,** which will help in **easy debugging**.
- For each **automatically stops after iterating** all elements in an array.