**JavaScript Arrays**

* An array is a special variable, which can hold more than one value.

**Creating an Array**

* Using an array literal is the easiest way to create a JavaScript Array.
* It is a common practice to declare arrays with the **const** keyword.

**Syntax:**

const *array\_name* = [*item1*, *item2*, ...];

**Example1:**

const cars = ["Audi", "Hyundai", "BMW"];

* Spaces and line breaks are not important. A declaration can span multiple lines:

**Example2:**

const cars

= [  
  "Audi",  
  "Hyundai",  
  "BMW"  
];

* You can also create an array, and then provide the elements:

**Example3:**

const cars = [ ];  
cars[0]= "Audi";  
cars[1]= "Hyundai";  
cars[2]= "BMW";

* Using the JavaScript Keyword **new Array( )**

**Example4:**

const cars = new Array("Audi", "Hyundai", "BMW");

**Example:**

For simplicity, readability and execution speed, use the array literal method.

**Accessing Array Elements**

* You access an array element by referring to the **index number**:
* const cars = ["Audi", "Hyundai", "BMW"];

let my\_car = cars[0];

**Note:** Array indexes start with 0.

[0] is the first element. [1] is the second element.

**Changing an Array Element**

This statement changes the value of the first element in cars:

cars[0] = "Maruti";

**Access the Full Array**

With JavaScript, the full array can be accessed by referring to the array name:

**Example:**

* const cars = ["Audi", "Hyundai", "BMW"];

document.getElementById("demo").innerHTML = cars;

**Arrays are Objects**

* Arrays are a special type of objects.
* The typeof operator in JavaScript returns "object" for arrays.

**Example:**

const person = {firstName:"Jawaharlal", lastName:"Nehru", age:46};

**Array Elements Can Be Objects**

* JavaScript variables can be objects.
* Arrays are special kinds of objects.
* Because of this, you can have variables of different types in the same Array.

**For Example:**

* You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:

myArray[0] = Date.now;  
myArray[1] = myFunction;  
myArray[2] = myCars;

**Array Properties and Methods**

* Arrays in JS have built-in array properties and methods:

**cars.length   // Returns the number of elements in the array  
cars.sort()   // Sorts the array**

**Looping Array Elements**

* One way to loop through an array, is using a for loop:

**Example1:**

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let len = fruits.length;  
  
let text = "<ul>";  
for (let i = 0; i < len; i++) {  
  text += "<li>" + fruits[i] + "</li>";  
}  
text += "</ul>";

* You can also use the Array.forEach() function:

**Example2:**

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
  
let text = "<ul>";  
**fruits.forEach(myFunction);**text += "</ul>";  
  
**function myFunction(value) {  
  text += "<li>" + value + "</li>";**  
**}**

**Adding Array Elements**

* The easiest way to add a new element to an array is using the push() method:

**Example:**

const fruits = ["Banana", "Orange", "Apple"];  
fruits.push("Lemon");  // Adds a new element (Lemon) to fruits

**Array size is increased dynamically**

* New element can also be added to an array using the length property:

**Example:**

const fruits = ["Banana", "Orange", "Apple"];  
fruits[fruits.length] = "Lemon";  // Adds "Lemon" to fruits

NOTE:

Adding elements with high indexes can create undefined "holes" in an array:

**Associative Arrays**

* Many programming languages support arrays with named indexes.
* Arrays with named indexes are called associative arrays (or hashes).
* In JavaScript, **arrays** always use **numbered indexes**.

**Example:**

const person = [ ];  
person[0] = "Vicky";  
person[1] = "Venkit";  
person[2] = 40;  
person.length;    // Will return 3  
person[1];        // Will return "Venkit"

**Note:**

If you use named indexes, JavaScript will redefine the array to an object.

After that, some array methods and properties will produce incorrect results.

**Example:**

const person = [ ];  
person["firstName"] = "Versatile";  
person["lastName"] = "Vikram";  
person["age"] = 20;  
person.length;     // Will return 0  
person[0];         // Will return undefined

person[“firstName”] 🡪 Versatile

**The Difference between Arrays and Objects**

* In JavaScript, **arrays** use **numbered indexes**.
* In JavaScript, **objects** use **named indexes**.
* Arrays are a special kind of objects, with numbered indexes.

**When to Use Arrays and When to use Objects.**

* You should use **objects** when you want the element names to be **strings (text)**.
* You should use **arrays** when you want the element names to be **numbers**.

**JavaScript new Array()**

* JavaScript has a built-in array constructor new Array().
* We can safely use [ ] instead.

These two different statements both create a new empty array named points:

const points = new Array();  
const points = [];

These two different statements both create a new array containing 6 numbers:

const points = new Array(40, 100, 1, 5, 25, 10);  
const points = [40, 100, 1, 5, 25, 10];

The new keyword can produce some unexpected results:

// Create an array with three elements:  
const points = new Array(10, 20, 30);

// Create an array with two elements:  
const points = new Array(10, 20);

// Create an array with one element ???  
const points = new Array(10);   // Won’t create error while printing

const points = [ 30 ]; // Won’t create error while printing

Caution:

* Unexpected error will come, Avoid to use new Array( ) to create an array
* Array is created with 40 undefined elements

**Check the array**

const fruits = ["Banana", "Orange", "Apple"];

let arr1 = typeof fruits; // Result is object, because JS array is an object

let arr2 = Array.isArray(fruits); // Result is true, indicates fruit is an array

let arr3 = fruits instanceof Array; // Result is true, indicates fruit is an array