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

```
const cars = ["Saab", "Volvo", "BMW"];
const multiple = [3, "cow", false]; // it can hold different datatypes
```

Arrays are mutable. Means it can be changed.

Why Use Arrays?

An array can hold many values under a single name, and you can access the values by referring to an index number.

Converting an Array to a String

The JavaScript method `toString()` converts an array to a string of (comma separated) array values.

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
// or
console.log(fruits.toString());
```

Arrays are Objects

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

But, JavaScript arrays are best described as arrays.

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;
```

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`. When to use `Objects`.

- JavaScript does not support associative arrays.
- 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**.

Associative Arrays

Many programming languages support arrays with named indexes.

Arrays with named indexes are called `associative arrays` (or hashes).

JavaScript does not support arrays with named indexes.

In JavaScript, arrays always use numbered indexes.

Example

```
const person = [];
person[0] = "John";
person[1] = "Doe";
```

```
person[2] = 46;
person.length; // Will return 3
person[0]; // Will return "John"
```

How to Recognize an Array

```
const fruits = ["Banana", "Orange", "Apple"];
let type = typeof fruits; // object
```

The `typeof` operator returns `object` because a **JavaScript array is an object**. Then how do we know if a variable is an array?

Solution 1:

To solve this problem ECMAScript 5 (JavaScript 2009) defined a new method `Array.isArray()`:

```
Array.isArray(fruits); // true
```

Solution 2:

The `instanceof` operator returns true if an object is created by a given constructor:

```
const fruits = ["Banana", "Orange", "Apple"];
fruits instanceof Array; // true
```