Arrays (An introduction)

JavaScript Arrays

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

```
const cars = ["Saab", "Volvo", "BMW"];
```

Homogeneous Array and Heterogeneous Array

```
let homogeneous_arr = ['alpha', 'beta', 'gamma'];
```

```
homogeneous_arr
Array(3) [ "alpha", "beta", "gamma" ]
```

```
let heterogenous_arr = ['alpha', 1, true, ['beta', 'gamma']]
```

```
heterogenous_arr
Array(4) [ "alpha", 1, true, (2) […] ]
```

Why Use Arrays?

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
let car1 = "Saab";
let car2 = "Volvo";
let car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

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

- Creating an Array
- Accessing Array Elements
- Changing an Array Element
- Converting an Array to a String
- Accessing the First and The Last Array Element
- How to Recognize an Array

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:
const array_name = [item1, item2, ...];

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

Arrays are Not Constants

The keyword const is a little misleading.
It does NOT define a constant array. It defines a constant reference to an array.
Because of this, we can still change the elements of a constant array as in:

const cars = ["Saab", "Volvo", "BMW"];
cars[0] = "Toyota"; / Here we have changed the first element of the element of the array */*
cars.push("Audi");

Syntax for creating arrays

```
const cars = ["Saab", "Volvo", "BMW"];

OR

const cars = [
  "Saab",
  "Volvo",
  "BMW"
];

OR

const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```

Using the JavaScript Keyword new

The following example also creates an Array, and assigns values to it:

```
const cars = new Array("Saab", "Volvo", "BMW");
```

The examples above is exactly the same as:

```
const cars = ["Saab", "Volvo", "BMW"];
```

There is no need to use new Array().

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 = ["Saab", "Volvo", "BMW"];
let car = cars[0];
```

Saab	Volvo	BMW
0	1	2

Note: Array indexes start with 0.

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

Changing an Array Element

You can change the element at 0th index by assigning a value to it.

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

```
cars[0] = "Opel";
```

Example:

```
const cars = ["Saab", "Volvo", "BMW"];
cars[0] = "Opel";
```

Converting an Array to a String

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

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

Would become:

```
Banana,Orange,Apple,Mango
```

The length Property

The length property of an array returns the length of an array (the number of array elements).

The length property is always one more than the highest array index.

Accessing the First and The Last Array Element

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fruit = fruits[0];

const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fruit = fruits[fruits.length - 1];
```

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.

A word of advise: Do not use “new Array()”

```
// Create an array with three elements:
const points = new Array(40, 100, 1);

// Create an array with two elements:
const points = new Array(40, 100);

// Create an array with one element ???
const points = new Array(40);
console.log(points);
/* ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, */

// It create an array with 40 undefined elements.
```

To create an array with one element:

```
const points = [40];
```

How to Recognize an Array

A common question is: How do I know if a variable is an array?
The problem is that the JavaScript operator typeof returns "object":

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

The typeof operator returns object because a JavaScript array is an object.

Solution 1: To solve this problem ECMAScript 5 (JavaScript 2009) defined a new method *Array.isArray():*

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

Solution 2: The instanceof operator returns true if an object is created by a given constructor:

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

fruits instanceof Array;
```

Looping Array Elements

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

Example

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

// We are dynamically creating HTML in the following loop:

// Here it is writing an opening tag for unordered list
let text = "<ul>";
for (let i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}

// And, here is the closing tag for unordered list
text += "</ul>";
```

Array.forEach()

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

Whatever we did in the previous slide, we are doing it using forEach() now.

forEach(): basically says ‘call this function for each element of the array’.

Example

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

let text = "<ul>";
fruits.forEach(myFunction);
text += "</ul>";

function myFunction(value) {
  text += "<li>" + value + "</li>";
}
```