Chapter 5 - Arrays

Arrays are Variables which can hold more than one value.

```
const fruits = ["banana", "apple", "grapes"]
const a1 = [7, "Harry", false]
```
↳ can be different types

Accessing Values

```
let numbers = [1, 2, 7, 9]

numbers[0] → 1
numbers[1] → 2
```

Finding the length

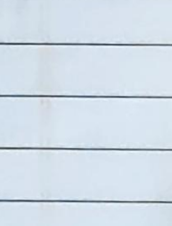
```
let numbers = [1, 7, 9, 21]
               ↙  ↓  ↓   ↘
               0  1  2    3

numbers[0] → 1
numbers.length → 4
```

Changing the values

```
let numbers = [7, 2, 40, 9]

numbers[2] = 8
```
↳ "numbers" now becomes [7, 2, 8, 9]
Arrays are mutable
Arrays can be changed



In JavaScript, arrays are objects. The typeof operator on arrays returns object

```
Const n = [1, 7, 9]

typeof n → returns "object"
```

Arrays can hold many values under a single name

Array methods

There are some important array methods in JavaScript. Some of them are as follows:

1. toString() → Converts an array to a string of comma separated values

```
let n = [1, 7, 9]
    n.toString() → 1, 7, 9
```

2. join() → joins all the array elements using a separator

```
let n = [7, 9, 13]
    n.join("-") → 7-9-13
```

3. pop() → removes last element from the array

```
let n = [1, 2, 4]
    n.pop() → updates the original array
              returns the popped value
```

4> push() → Adds a new element at the end of the array

```
let a = [7, 1, 2, 8]
a.push(9) → modifies the original array
          ↳ returns the new array length
```

5> shift() → Removes first element and returns it

6> unshift() → Adds element to the beginning. Returns new array length

7> delete → Array elements can be deleted using the delete operator

```
let d = [7, 8, 9, 10]
delete d[1]          → delete is an operator
```

8> Concat() → Used to join arrays to the given array

```
let a1 = [1, 2, 3]
let a2 = [4, 5, 6]
let a3 = [9, 8, 7]

a1.concat(a2, a3) → Returns [1, 2, 3, 4, 5, 6, 9, 8, 7]
    ↓
Returns a new array
Does not change existing arrays
```

9> Sort() → sort() method is used to sort an array alphabetically.

```
let a = [7, 9, 8]
a.sort()
```

↳ a changes to [7, 8, 9]
[modifies the original array]

Sort() takes an optional compare function. If this function is provided as the first argument, the Sort() function will consider these values (the values returned from the compare function) as the basis of sorting.

10> splice() → Splice can be used to add new items to an array

```
Const numbers = [1, 2, 3, 4, 5]
numbers.splice(2, 1, 23, 24)
```

Returns deleted items, modifies the Array ←

position to add

No of elements to remove

Elements to be added

11> slice() → Slices out a piece from an array. It creates a new array

```
Const num = [1, 2, 3, 4]
num.slice(2)      → [3, 4]
num.slice(1, 3)   → [2, 3]
```

Date
Page

12. reverse ( ) → Reverses the elements in the source array.

Looping through Arrays
Arrays can be looped through using the classical JavaScript for loop or through some other methods discussed below

1. forEach loop → calls a function, once for each array element

```
Const a = [1, 2, 3]
 a.forEach (( value, index, array ) => {
           // function logic
          });
```

2. map ( ) → Creates a new array by performing some operation on each array element.

```
Const a = [1, 2, 3]
 a.map ( (value, index, array) => {
          return value * value;
          })
```

3. filter ( ) → Filters an array with values that passes a test. Creates a new array

```
Const a = [1, 2, 3, 4, 5]
      a.filter (greater_than_5)
```

Date
Page

4, reduce method → Reduces an array to a single value

Const n = [1, 8, 7, 11]
let sum = numbers. reduce (add)
↳ A function
$\overset{\downarrow}{}$ 1+8+7+11

5, Array. from → Used to create an array from any other object

Array. from (" Harry ")

6, for ... of → For - of loop can be used to get the values from an array

7, for ... in → for-in loop can be used to get the keys from an array.