Day 5 - Arrays

Arrays

Can store multiple values. Each value has an index, and each index refers to a memory address. Values can be accessed by using their index. Index starts at 0 and ends at the index that is 1 minus the length of the array.

It’s a collection of different data types that are ordered and modifiable. You can store duplicate elements and different data types. Can be empty, or may have different data type values.

Creating an empty array

In JS, an array can be made in different ways. It’s common to use *const* instead of *let* to declare an array variable, since with *const*, you do not use that variable name again.

Using Array constructor

```
const arr = Array()
// or
let arr = new Array()
console.log(arr) // []
```

Using square brackets([])

```
// The most recommended way to create an empty list
const arr = []
console.log(arr)
```

Creating an array with values

```
const numbers = [1,2,3,4,5]
const fruits = ['banana', 'apple', 'orange']
console.log('Numbers: ', numbers)
console.log('Number of numbers', numbers.length)

console.log('Fruits: ', fruits)
console.log('Number of fruits', fruits.length)
```

Creating array with different data types

```
const arr = [
  'Chris',
  23,
  true,
  { country: 'United States', state: 'Arizona' },
  { skills: ['JS', 'React', 'HTML'] }
]
console.log(arr)
```

Creating an array using split

```
let companiesString = 'Facebook, Google, Microsoft, Apple, IBM, Oracle, Amazon'
const companies = companiesString.split(', ')

console.log(companies) // ['Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon']
```

Accessing array items using index

We access each element in an array using their index.

```
const fruits = ['banana', 'orange', 'mango', 'lemon']
let firstFruit = fruits[0]
let lastFruit = fruits[fruits.length - 1]

console.log(firstFruit) // banana
console.log(lastFruit) // lemon
```

```
const webTechs = [
  'HTML',
  'CSS',
  'JavaScript',
  'React',
  'Redux',
  'Node',
  'MongoDB'
] // List of web technologies

console.log(webTechs)  // all the array items
console.log(webTechs.length)  // to know the size of the array

console.log(webTechs[0])  // HTML
console.log(webTechs[webTechs.length - 1])  // MongoDB
```

Modifying array element

Once an array is created, we can modify the contents of the array elements.

```
const numbers = [1, 2, 3, 4, 5]
numbers[0] = 10  // changing 1 at index 0 to 10
numbers[1] = 20  // changing 2 at index 1 to 20
```

Methods to manipulate array

There's different methods to manipulate an array. These're some of the available methods to deal with arrays: *Array, length, concat, indexOf, slice, splice, join, toString, includes, lastIndexOf, isArray, fill, push, pop, shift, unshift*.

Array Constructor

To create an array:

```
const arr = Array() // creates an empty array
console.log(arr)
```

```
const eightEmptyValues = Array(8) // it creates eight empty values
console.log(eightEmptyValues) // [empty x 8]
```

Creating static values with fill

fill: Fill all array elements with a static value.

```
const fourXValues = Array(4).fill('X') // it creates four element values filled with 'X'
console.log(eightXValues) // ['X', 'X', 'X', 'X']

const three0Values = Array(3).fill(0) // it creates three element values filled with 0
console.log(three0Values) // [0, 0, 0]
```

Concatenating array using concat

concat: To concatenate two arrays

```
const firstList = [1, 2, 3]
const secondList = [4, 5, 6]
const thirdList = firstList.concat(secondList)

console.log(thirdList) // [1, 2, 3, 4, 5, 6]
```

Getting array length

length: To know the size of the array

```
const numbers = [1, 2, 3, 4, 5]
console.log(numbers.length) // -> 5 is the size of the array
```

Getting index of an element in an array

indexOf: To check if an item exists in an array. If it exists it returns the index else it returns -1.

```
const numbers = [1, 2, 3, 4, 5]

console.log(numbers.indexOf(5)) // -> 4
console.log(numbers.indexOf(0)) // -> -1
console.log(numbers.indexOf(1)) // -> 0
console.log(numbers.indexOf(6)) // -> -1

// Check an element if it exists in an array

const fruits = ['banana', 'orange', 'mango', 'lemon']
let index = fruits.indexOf('banana')  // 0

if (index === -1){
  console.log('This fruit does not exists in the array')
} else {
  console.log('This fruit does exist in the array')
}
// We can also use a ternary here
index === -1 ? console.log('This fruit does not exists in the array'): console.log('This fruit does exist in the array');
```

Getting last index of an element in array

lastIndexOf: Gives the position of the last item in the array. If it exists, returns the index. If not, returns -1.

```
const numbers = [1, 2, 3, 4, 5, 3, 1, 2]

console.log(numbers.lastIndexOf(2)) // 7
console.log(numbers.lastIndexOf(0)) // -1
console.log(numbers.lastIndexOf(1)) // 6
```

includes: To check if an item exists in an array. If it does, returns true. If not, returns false.

```
const numbers = [1, 2, 3, 4, 5]

console.log(numbers.includes(5)) // true
console.log(numbers.includes(10)) // false

const fruits = ['apple', 'orange', 'banana']
console.log(fruits.includes('apple')) // true
console.log(fruits.includes('kiwi')) // false
console.log(fruits.includes('orange')) // true
```

Checking array

Array.isArray: To check if the data type is an array

```
const numbers = [1, 2, 3, 4, 5]
console.log(Array.isArray(numbers)) // true

const number = 100
console.log(Array.isArray(number)) // false
```

Converting array to string

toString: Converts array to string

```
const numbers = [1, 2, 3, 4, 5]
console.log(numbers.toString()) // 1,2,3,4,5
```

Joining array elements

join: Used to join the elements of the array. The argument passed in the join method will be joined in the array and return as a string. By default, joins with a comma, but we can pass different string parameters which can be joined between the items.

```
const numbers = [1, 2, 3, 4, 5]
console.log(numbers.join()) // 1,2,3,4,5
console.log(numbers.join('')) // 12345
console.log(numbers.join(' ')) // 1 2 3 4 5
console.log(numbers.join(', ')) // 1, 2, 3, 4, 5
console.log(numbers.join(' # ')) // 1 # 2 # 3 # 4 # 5
```

Slice array elements

Slice: Cut out multiple items in range. Takes two parameters: starting and ending position. Doesn’t include ending position.

```
const numbers = [1,2,3,4,5]
console.log(numbers.slice()) // -> copies all items
console.log(numbers.slice(0)) // -> copies all items
console.log(numbers.slice(1,4)) // -> [2,3,4] doesn't include the ending position
```

Splice method in array

Splice: Takes three parameters: starting position, number of items to be removed and number of items to be added.

```
const numbers = [1, 2, 3, 4, 5]
numbers.splice()
console.log(numbers)  // -> remove all items

const numbers = [1, 2, 3, 4, 5]
numbers.splice(0,1)
console.log(numbers)  // remove the first item

const numbers = [1, 2, 3, 4, 5, 6]
numbers.splice(3, 3, 7, 8, 9)
console.log(numbers) //-> [1, 2, 3, 7, 8, 9] removes three items and replaces three items
```

Adding item to an array using push

Push: adding item in the end. To add item to the end of an existing array we use the push method.

```
const arr = ['item1', 'item2', 'item3']
arr.push('new item')
console.log(arr)
// ['item1', 'item2', 'item3', 'new item']
```

Removing the end element using pop

pop: Removing item in the end.

```
const numbers = [1, 2, 3, 4, 5]
numbers.pop() // -> remove one item from the end
console.log(numbers) // -> [1,2,3,4]
```

Removing an element from the beginning

shift: Removing one array element in the beginning of the array.

```
const numbers = [1, 2, 3, 4, 5]
numbers.shift()  // -> remove one item from the beginning
console.log(numbers) // [2,3,4,5]
```

Add an element from the beginning

unshift: Adding array element in the beginning of the array.

```
const numbers = [1, 2, 3, 4, 5]
numbers.unshift(0) // -> add one item from the beginning
console.log(numbers) // -> [0,1,2,3,4,5]
```

Reversing array order

reverse: Reverse the order of an array.

```
const numbers = [1, 2, 3, 4, 5]
numbers.reverse() // -> reverse array order
console.log(numbers) // [5, 4, 3, 2, 1]

numbers.reverse()
console.log(numbers) // [1, 2, 3, 4, 5]
```

Sorting elements in array

sort: Arrange array elements in ascending order. Sort takes a call back function, which we'll learn how to use later.

```
const webTechs = [
  'HTML',
  'CSS',
  'JavaScript',
  'React',
  'Redux',
  'Node',
  'MongoDB'
]
webTechs.sort()
console.log(webTechs) // ["CSS", "HTML", "JavaScript", "MongoDB", "Node", "React", "Redux"]
```

Array of arrays

Array can store different data types including an array itself. Let's create an array of arrays.

```
const firstNums = [1, 2, 3]
const secondNums = [1, 4, 9]

const arrayOfArray = [[1, 2, 3], [1, 2, 3]]
console.log(arrayOfArrays[0]) // [1, 2, 3]

const frontEnd = ['HTML', 'CSS', 'JS', 'React', 'Redux']
const backEnd = ['Node', 'Express', 'MongoDB']
const fullStack = [frontend, backEnd]
console.log(fullStack)  // [['HTML', 'CSS', 'JS', 'React', 'Redux'], ['Node', 'Express', 'MongoDB']]
console.log(fullStack.length) // 2
console.log(fullStack[0]) // ['HTML', 'CSS', 'JS', 'React', 'Redux']
console.log(fullStack[1]) // ['Node', 'Express', 'MongoDB']
```