SOFT CLUB

JAVA SCRIPT

LECTURE 4

TABLE OF CONTENTS

SOFT CLUB

GETTING STARTED WITH JAVASCRIPT

An array is an object that holds values (of any type) not particularly in named properties/keys, but rather in numerically indexed position
In JavaScript, an array is an ordered list of values. Each value is called an element specified by an index. ... First, an array can hold values of mixed types.

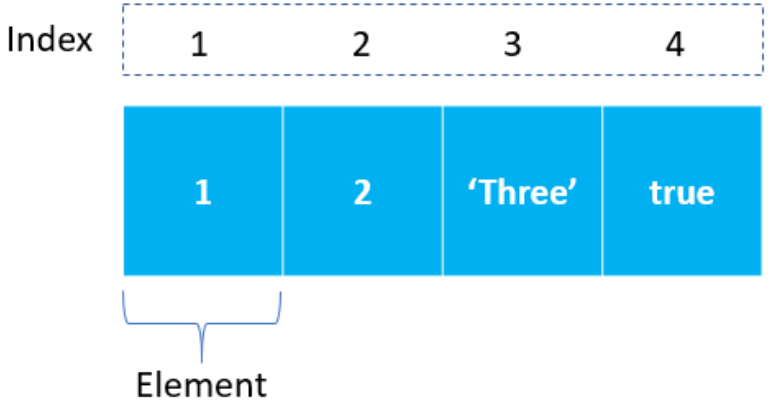
Creating an Array

1. Using an array literal

```
const array_name = [item1, item2, ...];
```

2. Using the new keyword

```
const array2 = new Array("eat", "sleep");
```



CHANGE ELEMENTS IN ARRAY

You can also add elements or change the elements by accessing the index value

```
let a= [ 'eat', 'sleep','open', 'close'];
// this will add the new element 'exercise' at the 2 index
a[2] = 'exercise';
console.log(a);
```

```
node /tmp/58OJlcbBTO.js
[ 'eat', 'sleep', 'exercise', 'close' ]
```

Suppose, an array has two elements. If you try to add an element at index 3 (fourth element), the third element will be undefined. For example,

```
let a= [ 'eat', 'sleep',];
// this will add the new element 'exercise' at the 3 index
a[3] = 'exercise';
console.log(a);
```

```
node /tmp/58OJlcbBTO.js
[ 'eat', 'sleep', <1 empty item>, 'exercise' ]
```

ARRAY METHODS

pop()

push()

shift()

unshift()

toString()

splice()

slice()

reverse()

concat()

join()

map()

filter()

find()

reduce()

forEach()

sort()

indexOf()

includes()

Add an Element to an Array

You can use the built-in method push() and unshift() to add elements to an array.

push()

```
let dailyActivities = ['eat', 'sleep'];
// add an element at the end
dailyActivities.push('exercise');
console.log(dailyActivities);
```

```
node /tmp/58OJlcbBTO.js
[ 'eat', 'sleep', 'exercise' ]
```

unshift()

```
let dailyActivities = ['eat', 'sleep'];
//add an element at the start
dailyActivities.unshift('work');
console.log(dailyActivities);
```

```
node /tmp/58OJlcbBTO.js
[ 'work', 'eat', 'sleep' ]
```

Delete an Element in Array

Pop()

```
let array = ['work', 'eat', 'sleep', 'exercise'];
// remove the last element
array.pop();
console.log(array); // ['work', 'eat', 'sleep']
// remove the last element from ['work', 'eat', 'sleep']
const removedElement = array.pop();
//get removed element
console.log(removedElement); // 'sleep'
console.log(array);  // ['work', 'eat']

let array2 = ['work', 'eat', 'sleep', 'exercise'];
const removedElment2 = array2.pop();

console.log(array2)

console.log(removedElment2)
```

```
node /tmp/qhX0w49VLY.js
[ 'work', 'eat', 'sleep' ]

sleep



[ 'work', 'eat' ]



[ 'work', 'eat', 'sleep' ]

exercise
```

shift()

```
let dailyActivities = ['work', 'eat', 'sleep'];
// remove the first element
const removeElement1 = dailyActivities.shift();
console.log(dailyActivities);

console.log(removeElement1)
```

```
node /tmp/qhX0w49VLY.js

[ 'eat', 'sleep' ]


work
```

JAVASCRIPT ARRAY METHODS

reverse()

```
1  //reverse()
2  const fruits = ["Banana", "Orange", "Apple", "Mango"];
3  console.log(fruits.reverse());
4  //  sort()  and  reverse()
5  const fruits2 = ["Banana", "Orange", "Apple", "Mango"];
6  console.log(fruits2.sort());
7  console.log(fruits2.reverse());
```

```
node /tmp/qhX0w49VLY.js
[ 'Mango', 'Apple', 'Orange', 'Banana' ]


[ 'Apple', 'Banana', 'Mango', 'Orange' ]
[ 'Orange', 'Mango', 'Banana', 'Apple' ]
```

concat()

```
1  const arr1 = ["Cecilie", "Lone"];
2  const arr2 = ["Emil", "Tobias", "Linus"];
3  const arr3 = ["Robin"];
4  const children = arr1.concat(arr2,arr3);
5  console.log(children);
```

```
node /tmp/qhX0w49VLY.js
[ 'Cecilie', 'Lone', 'Emil', 'Tobias', 'Linus', 'Robin' ]
```

JAVASCRIPT ARRAY METHODS

indexOf()

```
1  // Find the first index of "Apple":
2  const fruits = ["Banana", "Orange", "Apple", "Mango" , "Apple"];
3  let index = fruits.indexOf("Apple");
4  console.log(index)
5  //  Start at index 3:
6  const fruits2 = ["Banana", "Orange", "Apple", "Mango", "Apple"];
7  let index2 = fruits2.indexOf("Apple", 3 );
8  console.log(index2)
9  // Find the first index of "Apple", starting from the last element:
10 const fruits3 = ["Banana", "Orange", "Apple", "Mango", "Apple"];
11 let index3 = fruits3.indexOf("Apple", -1);
12 console.log(index3)
```

```
node /tmp/qhX0w49VLY.js
2

4

4
```

includes()

```
1  // includes() returns true if an array contains a specified element:
2  const fruits = ["Banana", "Orange", "Apple", "Mango"];
3  console.log(fruits.includes("Mango"));
4  // Check if fruit[] contains "Banana", starting the search from position 3:
5  const fruits2 = ["Banana", "Orange", "Apple", "Mango"];
6  console.log(fruits2.includes("Banana", 2));
```

```
node /tmp/qhX0w49VLY.js
true

false
```

JAVASCRIPT ARRAY METHODS

splice()

```
1  const fruits = ["Banana", "Orange", "Apple", "Mango"];
2  // At position 2, add 2 elements:
3  fruits.splice(2, 0 , "Lemon", "Kiwi");
4  console.log(fruits)
5
6
7  const fruits2 = ["Banana", "Orange", "Apple", "Mango", "Kiwi"];
8  // At position 2, remove 2 items:
9  fruits2.splice(2, 2);
10 console.log(fruits2)
11
12 var fruits3 = ["Banana", "Orange", "Apple", "Mango"];
13 // At position 2, add 2 elements, remove 1:
14 fruits3.splice(2, 1, "Lemon", "Kiwi");
15 console.log(fruits3)
```

```
node /tmp/qhX0w49VLY.js
[ 'Banana', 'Orange', 'Lemon', 'Kiwi', 'Apple', 'Mango' ]
[ 'Banana', 'Orange', 'Kiwi' ]
[ 'Banana', 'Orange', 'Lemon', 'Kiwi', 'Mango' ]
```

Syntax

```
array.splice(index, howmany, item1, ....., itemX)
```

JAVASCRIPT ARRAY METHODS

slice()

```
1  const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
2  const citrus = fruits.slice(1, 3);
3  console.log(citrus);
4
5  const fruits2 = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
6  const myBest = fruits2.slice(-3, -1);
7  console.log(myBest);
```

```
node /tmp/qhX0w49VLY.js
[ 'Orange', 'Lemon' ]
[ 'Lemon', 'Apple' ]
```

JAVASCRIPT ARRAY METHODS CALLBACKS

JAVASCRIPT ARRAY METHODS

map()

```
1  // Return a new array with the square root of all element values:
2  const numbers = [4, 9, 16, 25];
3  const newArr = numbers.map(Math.sqrt)
4  console.log(newArr)
5  // Multiply all the values in an array with 10:
6  const numbers2 = [65, 44, 12, 4];
7  const newArr2 = numbers2.map(myFunction)
8
9  function myFunction(num) {
10   return num * 10;
11 }
12 console.log(newArr2)
```

```
node /tmp/qhX0w49VLY.js
[ 2, 3, 4, 5 ]
[ 650, 440, 120, 40 ]
```

JAVASCRIPT ARRAY METHODS

SOFT CLUB

forEach()

```
1  const words = ['hello', 'bird', 'table', 'football', 'pipe', 'code'];
2  const capWords = words.forEach(capitalize);
3
4  function capitalize(word, index, arr) {
5    arr[index] = word[0].toUpperCase() + word.substring(1);
6  }
7  console.log(words);
```

```
node /tmp/qhXOw49VLY.js
[ 'Hello', 'Bird', 'Table', 'Football', 'Pipe', 'Code' ]
```

Syntax

```
Array.forEach(callback(item, index, arr), thisValue)
```

JAVASCRIPT ARRAY METHODS

find()

```
1  // Find the value of the first element with a value over 18:
2  const ages = [3, 10, 19, 20];
3  function checkAge(age) {
4    return age > 18;
5  }
6  console.log(ages.find(checkAge));
```

```
node /tmp/qhX0w49VLY.js
19
```

Syntax

```
array.find(function(currentValue, index, arr),thisValue)
```

JAVASCRIPT ARRAY METHODS

sort()

```
1  // 1 only sort()
2  const points2 = [40, 100, 1, 5, 25, 10];
3  points2.sort();
4  console.log(points2)
5  // 2 sort( with function )
6  const points = [40, 100, 1, 5, 25, 10];
7  points.sort(function(a, b){return a-b});
8  console.log(points)
```

```
node /tmp/qhX0w49VLY.js
[ 1, 10, 100, 25, 40, 5 ]

[ 1, 5, 10, 25, 40, 100 ]
```

Find the lowest value and highest value:

```
1  // Sort the numbers in ascending order
2  const points = [40, 100, 1, 5, 25, 10];
3  points.sort(function(a, b){return a-b});
4  let lowest = points[0];
5  console.log(lowest)
6  // Sort the numbers in descending order:
7  const points2 = [40, 100, 1, 5, 25, 10];
8  points2.sort(function(a, b){return b-a});
9  let lowest2 = points2[0];
10 console.log(lowest2)
```

```
node /tmp/qhX0w49VLY.js
1

100
```

Thanks