Arrays

Adding Elements

- Adding elements at the end of an array.
  Using push() we can add elements at the end of an array

  example:

  ```
  Const numbers = [3,4];

  numbers.push(5,6);

  Console.log(numbers);
  ```

  result: [3,4,5,6]

- Adding begining of an array use unshift().

  example:

  ```
  numbers.Unshift(1,2);
  console.log(numbers);
  ```

  result: [1,2,3,4,5,6]

- To add elements at any position and deleting elements from array use splice()

  example:

  ```
  numbers.Splice(2, 0, 'a', 'b')
  ```

  (2 = index number, 0 = deleting element)

  result: [1, 2, "a", "b", 3, 4, 5, 6]

Finding Elements

in Primitive type

Using the indexof() we can find the position (index number) of a given number in an array. If the number not available it return -1.

example:

```
Const numbers = [1,2,3,1,4];
Console.log(numbers.indexof('a'));
    result: -1
Console.log(numbers.indexof(1));
Console.log(numbers.lastindexof(1));
    result: 0
            3
```

- To check the given number is available on array Include() is used.

example:

```
Console.log(numbers.includes(1));
    result: true
```

Second parameters of all these index are from index. from their they start to search.

```
Console.log(numbers.indexof(1,2));   (2 → from index)
    result: 3
```

in Reference type

The find() method returns the value of the first element in the array that satisfies the provided testing function. otherwise undefined is returned.

Call back

Example:

```
Const courses = [ { id:1, name:'a' },
                  { id:2, name:'b' }
                ];

Const course = courses.find(function(course){
    return course.name === 'a';
});

Console.log(course);
    result: {id:1, name:"a"}
```

Use findIndex() to return the index number of the element.

Arrow function

Arrow function is ~~used to~~ make the code shorter and than call back function.

Example:

```
Const Courses = [ { id: 1, name: 'a' },
                  { id: 2, name: 'b' }
                ];

Const course = Courses.find ( Course => Course.name === 'a' );
```

Removing Elements

```
Const numbers = [1, 2, 3, 4];
```

- Removing elements from end of an array.

```
Pop ();
```

Example:

```
Const last = numbers. Pop ();
Console. log ( numbers);
Consol. log ( last);
```

Result: [1, 2, 3]

- Removing element from the begining of an array use shift ()

```
Const first = numbers. shift ();
Console. log (first);
```

result: [2, 3]

- Removing element from any Position of an array use splice ()

Example

~~Const foos~~

~~Console. log (splice numbers. splite~~

.) `numbers. splice ( 2, 1);`

```
Console. log ( numbers);
```

result: [1, 2, 4]

.) `numbers. splice ( 2, 2);`

```
Console. log ( numbers);
```

result: [1, 2]

Emptying an Array

Deleting all the elements from an array.

```
let numbers = [1, 2, 3, 4];
let another = numbers;
```

Solution 1

```
numbers = [];
```

Solution 2

```
numbers.length = 0;
```

Solution 3

```
numbers.splice(0, numbers.length);
```

Solution 4

```
while (numbers.length > 0)
numbers.pop();
```

Combining and slicing Arrays

```
Const first = [1, 2, 3];
Const second = [4, 5, 6];
```

Combining

```
Const combined = first.concat(second);
```

Slicing

```
Const slice = Combined.Slice(2, 4);
Const slice = Combined.Slice(2);
```

In reference type

```
Const first = [{ id: 1, name: 'a' }, ];
Const second = [4, 5, 6];
Const combined = first.Conlat(second);
```

The Spread Operator

```
Const first = [1, 2, 3];
Const second = [4, 5, 6];
Const combined = [... first, 'a', ...second, 'b'];
console.log(combined);
result: [1, 2, 3, a, 4, 5, 6, b]

const copy = [...Combined];
Console.log(copy);
result: [1, 2, 3, a, 4, 5, 6, b]
```

Iterating an Array

```
Const numbers = [1, 2, 3];
```

(1) For (number of numbers)

```
Console.log(number);
```

(2). numbers.forEach(function(number) {

```
Console.log(number);
});
```

(3) numbers.forEach(number => console.log(number));

In for each loop index parameter is optional.

```
numbers.forEach ((number, index) =>
    console.log (index, numbers));
```

Joining Array

Example:

```
Const numbers = [1, 2, 3];
  Const joined = numbers.join(',');
    Console.log(joined);
```

Result: 1, 2, 3

```
Const message = 'This is my first message';
  Const parts = message.split(' ');
    Console.log(parts);
  Const comb = parts.join('-');
    Console.log(comb);
```

result: This-is-my-first-message

Sorting Array

```
Const numbers = [3, 1, 2];
  numbers.sort();
  Console.log(numbers);
```

Result: [1, 2, 3]

```
  numbers.reverse();
    Console.log(numbers);
```

Result: [3, 2, 1]

Sorting Objects in an array

```
Const courses = [ {id: 1, name: 'Node'},
                  {id: 2, name: 'JavaScript'}
                ];

courses.sort(function(a, b){

Const NameA = a.name.toUpperCase();
Const NameB = b.name.toUpperCase();

if (NameA < NameB) return -1;
 if (NameA > NameB) return 1;
   return 0;
});
  Console.log(courses);
```

Testing the Elements of an Array

```
Const numbers = [1, 2, 3, 4, 5];

const allPositive = numbers.every ( function (value)
  { return value >= 0; } );

  Console.log ( allpositive );
```

result: True

```
Const numbers= [1, 2, -1, 3, 4, 5];

Const atleastOnePositive = numbers. Some (function (value)
    { return value >= 0; } );

    Console: log (atleast One Positive);
```

result : True

Filtering an Array

```
Const numbers = [1, 2, -3, 4, 5]

  const filtered = numbers.filter (
                  value => return value >= 0);

  Console. log ( filtered );
```

Maping

Mapping an Array

```
Const numbers = [1, 2, 3, 4, 5];

Const item = numbers. map ( value => `<li>` + value +
                  `</li>` );

Const html = item. Join (' ');

  Console. log ( html );

Const html = `<ul>` + item. Join(' ') +
                  `</ul>`;
```