**Red colour data is research during Seebiz**

**ARRAYS**

**From lec#35 &36**

**O.S >** Array is a collection of the variables

>Data store in array is in the form of variable’s collection

>There is no index because array is not a key-value pair data structure

**Methods:**

**Addition**

.push > add value in the end of array and return a new length of the array.

.unshift > add value in the front of the array and return a new length of the array.

**Subtraction/Deletion**

.pop > remove the value from the end

.shift > remove the value form the start of the array

.indexOf > is used to get the index # of value

.includes > is used to check if the value available will return **TRUE**

**.**entires(it will return key means index and also the value

**Arrays** in JavaScript can work both as a **queue and as a stack**. They allow you to add/remove elements, both to/from the beginning or the end. **In computer science, the data structure that allows this, is called** [**deque**](https://en.wikipedia.org/wiki/Double-ended_queue)**.**

**Now start (⌐■\_■)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Methods** | | **New Array** | **Mutate original** | **Return** | | **parameters** | **Negative** | **sparse** | **Remarks** |
| **Slice ()**  **Use to slice the array** | yes | | no | Copy of Array  Shallow | * **Single** parameter means starting * **double** means starting and ending   slice()  slice(start)  slice(start, end) | | -1 means start from end | array returned from slice() may be sparse if the source is sparse. | Both in single and double parameters method parameter will not included.  The slice() method is a [copying method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). |
| **splice ()**  **use to add element in the array** | yes | | yes | An array containing the deleted elements. If we remove O element it will return empty array. | * **double** means starting and for remove element   **in this way it works as slice**()   * **triple** one for start 2nd for removing and 3rd for addition   splice(start)  splice(start, deleteCount)  splice(start, deleteCount, item0)  splice(start, deleteCount, item0, item1)  splice(start, deleteCount, item0, item1, /\* … ,\*/ itemN) | | -1 means start from end |  | The splice() method is a [mutating method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). |
| **pop()** | No | | Yes | returns the element it removed: | pop()  undefiend | |  |  | method is a mutating method. It changes the length and the content of this. |
| **push()** | No | | yes | Return changed length of the array |  | |  |  | method is a [mutating method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). |
| **Shift()** | No | | Yes | returns the element it shifted |  | |  |  | method is a [mutating method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). |
| **unShift()** | No | | Yes | Return changed length of the array |  | |  |  | method is a [mutating method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). |
| **toSring ()** | No | | No | method does not change the original array. |  | |  |  | method returns a string with array values separated by commas. |
| **join()** | No | | No | method returns an array as a string. |  | |  |  | Any separator can be specified. The default is comma (,). |
| **concat()** | Yes | | No | method returns a new array, containing the joined arrays. |  | |  |  |  |
| **flat()** | yes | | No | returns a [shallow copy](https://developer.mozilla.org/en-US/docs/Glossary/Shallow_copy) that contains the same elements as the ones from the original array. | flat()  flat(depth)  depth level specifying how deep a nested array structure should be flattened. Defaults to 1. | |  |  | Tmethod is a [copying method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). It does not alter this but instead |
| **map()**  **.** | creates a new array from calling a function for every array element. | | No |  | map(callbackFn)  map(callbackFn, thisArg) | |  | **o.s**  if we change the array after map method start it will work on new one it assume basic length | callbackFn is invoked only for array indexes which have assigned values. It is not invoked for empty slots in [sparse arrays](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections#sparse_arrays).  The map() method is an [iterative method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#iterative_methods).  The map() method is a [copying method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods). |
| forEach() | No | | No | Return undefined |  | |  |  |  |

**Methods:**

.**slice** [The slice() method slices out a piece of an array. It is basically copying method]

( it will take start parameters and on their basis it will return NEW SHALLOW COPY of array. Without mutating the original array) .We can also use starting as well as ending parameter for slice.

.**splice** (it work same like slice but disturb the original array.> it takes two values first is starting point and second is the no# of element{it doesn’t make a shallow copy of the original array} it mutate the original array.

For splice method we need index that we want to delete

**.reverse** (it reverse the array but it also mutate the original array

**.concat** (it join two array but it **NOT** **MUTATE** the original array

{just like spread operator (…arr, …arr1}

**.at(-1** it will return the last element in the array. This is the replacement of the arr.length-1/arr[12])

**Loop over Array**

**.forEach (**it is the modern form of the forOf ( loop)**)** it takes an call back function so its order is high. Function which takes an function as a parameter is called as HOrderF

-- Array.forEach(function(currentValue, currentIndex, array/map) =----

**For Sets** this **forEach**(function is used in same way and same parameters but it will not have key-value pair it just have the value

DATA TRANSFORMATION WITH **MAP, FILTER AND REDUCE**

**.Map**

map is similar to the forEach method but it make brand new array (it apply method on the element and return a new array

{it map the element of original array on the new array that’s why we call it map }

* It also contain three parameter just like forEach
* It return a new array but forEach perform action on each iteration that is side effect of the forEach and it also work on original array 2nd side effect

**Important point learn today**

callbackFn will not visit any elements added beyond the array's initial length when the call to map() began. Changes to already-visited indexes do not cause callbackFn to be invoked on them again. If an existing, yet-unvisited element of the array is changed by callbackFn, its value passed to the callbackFn will be the value at the time that element gets visited. [Deleted](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/delete) elements are not visited.

**.filter(**

Filter is used in place of the if statement

It returns a new array that contain the filtered value

**Array.filter(function(val,i){return val>0});**

**.reduce**

It return the sum of all the value mean single value

**Array.reduce(function(val ,i){},0[this is for setting initial value =0])**

It takes 4 parameters(accumulator, current-value, I, array)

.

**.find**

It works same like filter but there are two main difference

* Find return the single element
* Find does not return new array
* It only work until first condition is satisfied

**.findIndex**

It return the index of searched value and work same as find method

**.some:**

it work same as includes but it work on condition rather includes just check equality

if any of the element satisfy the condition it works

**.every:**

It works if all the element of array will satisfy the condition of array

**.flat:**

It work with arrays to join them in single array but it work only upto the first level

**.flatMap**

It work for both mapping and flattening means that both functionality will perform here for nested arrays