12 September 2022 20:03

1. How to create a Array

   [ ] or Array ( ) ✓

2. String.split → Array

3. Accessing an element in Array using indexes

4. Modifying an array element

   arr = [1, 2, 3]
   arr[0] = 4 → modify

5. Sort method

---

1. Push
2. Pop ✓
3. join
4. Concat

deleting an element from a specific index

slice
splice
shift
unshift
delete

substr( )
S.I, No. of characters

substring (S.I, E.I)

"abcdef".substr(1,3)
1 2 3
substring(1,3)

[dexe]

"abcdef . substr

1 2 3

bcd . substring(1,3)

bc

1) slice (S.I, E.I)

excluded

↓

cut

slice out the subarray

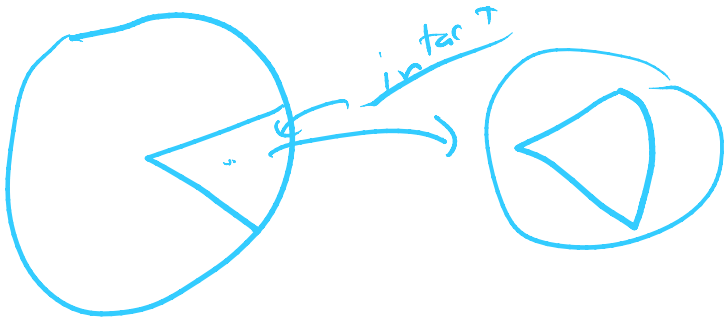
↓

It doesn't the change the original array

2) splice (S.I, No. of elements)

↓

cut out the subarray from the original array in real



[1, 2, 3, 4, ~~5~~, 6]

[0 1 2

[3, 4, 5, 6]

splice (2)

5!

[1,2]

[]

splice(0)

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

Write a generic function

[1]

[1,2]

[1,2,3]

[1,2,3,4]

[1,2,3,4,5]

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

[1,2,3,4

[1, 2, 3, 4]

[1, 2]

[1, 2, 3, 4]

0 1

[1, 2, 3, 4] → .slice(0, 1)

[1]

.slice(0, 2)

[1, 2]

.slice(0, 3)

[1, 2, 3]

.slice(0, 4)

[1, 2, 3, 4]

1 → 4

≤ arr.length

arr = [1, 2, 3, 4]

[1]

[2]

[3]

[4]

---

arr = [1, 2, 3, 4]

[1, 2, 3, 4]

[2, 3, 4]

[3, 4]

[4]

$O(\frac{n}{2}) \to O(n)$

slice

W	A	B
$O(n)$	$O(n)$	$O(1)$

[1, 2, 3, 4 ...] n

slice(0, 1) → $O(1)$

slice(0, arr.length) → $O(n)$

n

$O(\frac{n}{3}) \to O(n)$

$n - (n-3) = +3$

slice(n-3, n) $O(1)$

3 constant

slice$(n - \frac{n}{3}, n)$

$n - (n - \frac{n}{3}) = \frac{n}{3}$ $O(n)$

[1, 2, 3, 4] → .slice(0, n) ← arr.length

[1, 2, 3, 4]

.slice(1, n)

[2, 3, 4]

[1, 2, 3, 4] .slice(1, n)

[2, 3, 4] .slice(2, n)

[3, 4] .slice(3, n)

[4]

```
for(let i = 0; i < arr.length; i++) { //
o(n)
    console.log(arr.slice(i,arr.length)) n
// o(n)
}
```

$O(n^2)$

n + n — — ntimes

n + n + n — — — ntimes

$n(1+1+1 \text{ --- ntimes}) = n(n) = n^2$

[1, 2, 3, 4]

[1] → .slice(0, 1) (1)

[2] → .slice(1, 2) (1)

[3] → .slice(2, 3) (1)

[4] → .slice(3, 4) (1)

i, i+1

arr.length − 1

$O(1)$

splice (S.I , No. of Elements to Remove , No. of Elements to Add , ---- Elements

3 , 1, 2, 3

function Print (a){
console. log (a)
}

Print( )

splice (S.I, No. of elements , 9, 10

splice (S.I , No. of elements to Remove , ~~No. of~~ , --- elements you want to add

[1, 2, 3, 4] O(n)

splice( ) → remove adding

[1, 2, 3, 9, 10] O(n)

[1, 2, 3, 9, 10] ① O(n)

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Splice(3, 0, ~~4, 5, 6, 7, 8~~)

~~a[1,2,3,4,5]~~ O(n)

[1, 2, 3] [4, 5, 6, 7, 8] [9, 10] O(n)

3n → O(1) O(n) O(n)

Creation + assignation

original array variable

Creation

Removes the first element

Shift

pop removes the last element

Unshift (element)

push (element)

element

Unshift ( )

↓

add the element to the beginning

↓

adds the element to the end

[1, 2, 3, 4] n

[1], [2, 3, 4] n-1

O(n)

[1, 2, 3,]

arr = [1, 2, 3] (arr[arr.length] = 4) O(1)

---

arr1 [1, 2, 3, 4] arr2 [5, 6, 7, 8]

↓

[1, 2, 3, 4]

[1, 2, 3, 5]

[1, 2, 5, 6]

[1, 5, 6, 7]

[5, 1, 7, 8]

[1, 2, 3, 4] → [5, 6, 7, 8]

splice(_, 0) + slice(0,0) → [1, 2, 3, 4]

[1, 2, 3, 4] [] $O(n^2)$

[1, 2, 3, 5]

splice(n-1, 1) slice(0, 1)

[5] → [1, 2, 3, 5]

+

[1, 2, 3] 3-1

slice(0, 2)

[5, 6] → [1, 2, 5, 6]

splice(n-1, 1)

[1, 2]

+

[1, 2, 5, 6]

[1, 5, 6, 7]

slice(0, 3)

splice(n-1, 1)

[1]

[5, 6, 7

$O(1)$

[1, 2, 3, 4] [5, 6, 7, 8]

n = arr.length

4

[1, 2, 3, 4]

[1, 2, 3, 5]

[1, 2, 5, 6]

[1, 5, 6, 7]

[5, 6, 7, 8]

[1, 2, 3, 4] removing

splice(_, 0)

splice(n-1, 1)

[1, 2, 3]

splice(n-1, 1)

[1, 2]

[5, 6, 7, 8]

slice(0, 0)

slice(0, 1)

[5]

Add

[1, 2, 3, 5]

[1, 2, 5, 6]

slice(0, 2)

[5, 6]

splice (n-2, 1)

[1,2]

n-3

n-4

[1, 2, 5, 6]

slice(0, 2)

[5, 6]

Splice

[1, 2, 3] 0 1 2 3

[... n numbers]

n-3

O(n)

O(n)

already exist

splice(3, 1)

[1, 1, 1, 1]

splice(3, 1)

O(1)

O(n)

[1, 2, 3, 4]

splice(arr.length-1, 1)

O(1)

[5, 6, 7, 8]

slice(0, i+1)

O(1)

0, 1

[1, 2, 3]

[1, 2]

[1]

[]

[1, 2, 3, 5]

[1, 2, 5, 6]

[1, 3, 6, 7]

[5, 6, 7, 8]

[5]

[6]

[7]

[8]

1, 2

2, 3

3, 4

O(n)

O(n)

[1, 2, 3, 4]

[1, 2, 3, 4]

[1, 2, 3, 4] [5, 6, 7, 8]

O(n)

Objects

1. Intialize { }
2. Assign (key : value) — obj[key] = value
3. Access a value — obj[key]
4. Array of keys — Object.keys(obj) → [... keys]

4
1. map
2. filter
3. foreach
4. reduce

2 { 2 kinds of for loop

arr.map (function)

↓

method

↓

Array

complete Array

arr.map (function (ele, index, A) {

3)

executed length of Array

.map (function (ele, index, array) {

return ele × ele

1. 1, 0, [1, 2, 3, 4]
2. 2, 1, [1, 2, 3, 4]
3. 3, 2, [1, 2, 3, 4]
4. 4, 3, [1, 2, 3, 4]

3) [1, 2, 3, 4]

[1, 4, 9, 16]

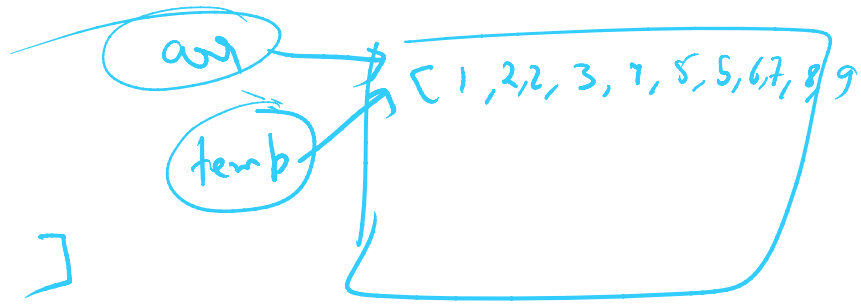
i = 0 1 2

```
let arr = [1,2,3,4,5,5,6,7,8,9]
```

```
let arr = [1,2,3,4,5,5,6,7,8,9]

for(let i = 0; i < arr.length; i++) {
    if(arr[i] % 2 === 0) {
        arr.splice(i,1)
    }
}
```

```
let arr = [1,2,2,3,4,5,5,6,7,8,9]
let temp = arr
```



[ele]

filter (function (ele, in, c.A) {

Array

true → keep the element

false → Remove the element / don't keep the element

}

[ ].filter (function (ele, index, c.A) {

return true or false

return true → keep the element

or false → will not keep the element

When to use **map**

If we have to change the elements of an array by a single logic

When to use **filter**

If we have to remove the elements of an array by a single logic

Serverside → increasing → 10 in single api call

Frontside → constant single api → 1100

[1100]

page number = x (y)

offset = 10

arr.slice((P.N-1)×offset)

arr.slice((P.N-1)×offset)

10, 20

~~off~~
P.N×offset)

next