Tuwashi Notes

11th February 2023

JavaScript - class - 3

Basics - 3 (In built Objects && Arrays)

(1) Math Object :-

↳ In built Object for mathematical functions.

→ Math.random()
↳ generates Random no. b/w 0 & 1

→ Math.max(2,1,4,3)
↳ for maximum no.

→ Math.round(1.8)
↳ round of 1.8 i.e 2

→ Math.abs(-2)
↳ absolute, returns positive for positive and positive for negative.
here, 2 will be returned.

(2) String Object :-

There are two types of strings in JS

String → primitive.
↳ let name = 'Tuwash';

String → object
String Constructor Function.
↳ let name = new String('name');
typeof(name) → object

we can convert primitive String to object using . notation.

name.length, name.includes('Tus')
name.startsWith('Tus'), name.endsWith('ash');
(name.toUppuCase()); name.toLowerCase();
name.trim(), name.replace('Tus', 'Pus');
and multiple other functions.....

TO Split :-

```
let message = 'This is my message';
let word = message.split(" ");
console.log(word);
```

(3) Template Literal

to use single ' in String.

(\') → slash is used
these are the notations

like
for newline (\n)

But another alternative without using \ slashes we use

Template Literal

by BackTick is used.

(``)

```
let msg = `This is
my ' message`;
```

↓
Same will be the output.

also we can add variable, in backtick using $

```
let msg = `This
is my messege.
Hello ${name};`
```

↑
In a same order
& name = 'Turuvash'
will be printed.

(4) Date and Time :-

Date :-

~~let date = newDate();~~

(1) let date = new Date();
clg(date);
← current date & Time

(2) let date2 = new Date('June 20 1998 07:15');
clg(date2);

(3) let date3 = new Date(1998, 6, 20, 7);
↑ year ↑ month indexing Starts from 0 6 is july. ↑ date ↑ Time (hr)

also
change year,
↓

```
date3.setFullYear(1947)
clg(date3);
```

Function / method (Getter / Setter)

When we use Function / method to

Set value → Setter

Recieve value → getter

Arrays :- (Object / Reference type collection of all types of items)

- Adding new Elements
- Finding Elements
- Removing Elements
- Splitting Elements
- Combining Elements.

1) Creation :-

```
                0   1   2   3        4   → Indexes
                ↓   ↓   ↓   ↓        ↓
let number = [ 1,  3,  5,  7,  'Tuwash' ]
```

2) Adding / pushing new Element :-

we can access array using Indexes

number [0] → 1 (value in index 0)

Insert :-

- End
- Beginning
- middle

let number = [1, 4, 5, 7)

(i) End → [1, 4, 5, 7, 9]

→ number.push (9)

(ii) Beginning → [8, 1, 4, 5, 7]

→ number.unshift (8)

(iii) Middle →

→ number.splice (2, 0, 'a', 'b', 'c')

index, deletion, adding

3) Find out Number (Searching Element)

```
number.indexOf(2);
```

→ if we want to check if a number exist in an array.

```
if (numbers.indexOf(10) != -1)
    console.log("present")
```

↖ Not Right way.

Good practice

```
Console.log(numbers.includes(7));
```

↳ true / False (returns)

Also

```
numbers.indexOf(4, 2);
```

4 ↑ search, 2 ↑ index to start → (-1) Any

(-1) is answer when you write index which is not present

* We have done these on primitive

Now on References

```
let courses = [
    { no: 1, name: 'Love' },
    { no: 2, name: 'Babbar' }
]
```

↖ Array of object is created

```
clg(courses);
```

In reference we cant find using indexOf & includes because Searching in reference is not same as primitive.

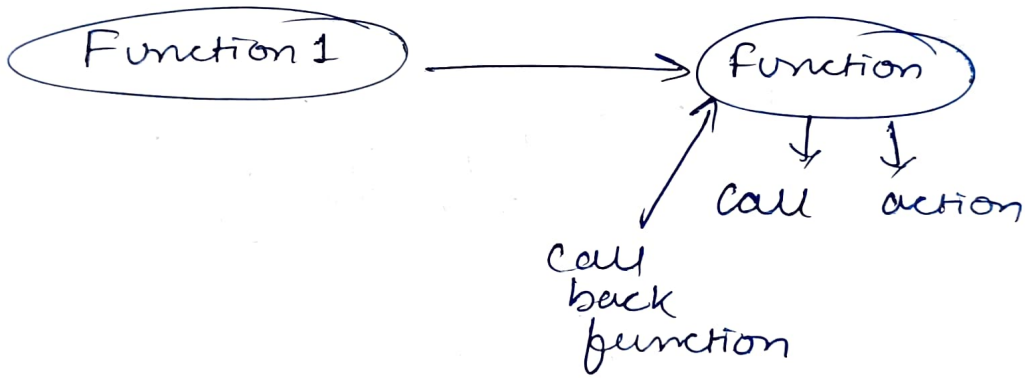
For primitive it search by value

For Reference it Search by Address.

We use Callback functions here,

↓

Function passed into another function as an argument, which is then invoked inside the outer function to complete action.




```
let c = Course.find( function (course) {
    return course.course_name == 'Love'
})
clg (c);
```

Syntax :-

arrayName.find( ) ← predicate function i.e condition to find object

predicate Object → function (course)
{
return course.name == 'Love'
}

Arrow Function (more concise)

✓ let course = course.find(course =>
course.name == 'Love');

we remove { return }

only when we have 1 value
Single value.

no input parameter then arrow function
() =>

④ Removing Element :- [1, 2, 3, 4]

- end → pop()
- Beginning → shift()
- middle → splice(3, 1)
  - 3 ↑ index
  - 1 ↑ no. of element you want to delete

⑤ Emptying an Array :-

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

① numbers = [] → empty. → automatically removed by garbage collector then,

↑ its not deleted still its stored

For deleting:

(2) `numbers.length = 0`

↳ this is what we do to make array empty.

also

(3) `number.splice( 0 , number.length )`

0 ↑ index

number.length ↑ no. of element you want to delete

(4) also,

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

↳ using loop.

(6) Combining & Slicing Arrays :-

Combine

```
let first = [1, 2, 3];
let Second = [4, 5, 6];
    using concat() → method.
let combined = first.concat(Second);
```

Slicing,

using slice() method

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

↓

Slice this

slice (__ __) → (x, y)

↓ range where x is included y is excluded.

↑ Start Index

↑ end Index

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

0 1 2 3 4 5

↓ Include

↓ exclude

Slice (2, 4) to get (3, 4)

↓ Slice

* if we give one parameter

slice (2)

↳ then from 2nd index all removed.

* if slice()

↳ copy of original array called as Full slicing.

* Spread Operator

```
let first = [1, 2, 3]
let second = [4, 5, 6]
let Combined = [...first, ...second]
```

also to add

```
let combined = [...first, 'a', ...second, 'b']
```

to copy

```
let another = [...combined]
```

* Iterating an Array =

loop

→ for of loop is on iterables

→ foreach → also

```
let arry = [1, 2, 3, 4]
```

for of

```
For (let value of arry) {
    clg (value)
}
```

For Each

```
arry.forEach (function (number) {
    clg (number)
});
```

(DO change this to arrow Function :)

* Joining Arrays

num = [1, 2, 3]

to join them like (1,2,3)

using join() method.

```
num = [1, 2, 3, 4]
const Joined = num.join()
clg (Joined)
```

↳ 1,2,3,4

Split() method

Creates an array.

```
let msg = "This is my message";
let parts = msg.split(" ")
clg (parts)
```

['This', 'is', 'my', 'message']

```
let Joined = parts.('-')
clg (Joined)
```

* Sorting Arrays

by using sort() method

Sort is to arrange in increasing or decreasing order

by defaut ascending order.

let num = [10, 50, 20, 60, 30]

num.sort()

clg (num)

↳ [10, 20, 30, 50, 60]

also reverse using

num.reverse()

[60, 50, 30, 20, 10]

→ We cant do Sort() in object like this we have to add predicate function.

* Filtering Arrays :-

↳ using filter();

number.filter( )

↑ callback function.

For +ve

let num = [1, 2, -3, -4]

let filtered = num.filter(function(value) {
return value >= 0
})

clg (filtered)

↳ [1, 2]

output.

* Mapping Arrays → map each element of array to something else.

↳ map() method

Same like ASCII

a → 97

b → 98

Same →

```
let numbers = [7, 8, 9, 10];

numbers.map(function(value){
    return 'Student_no' + value;
});
```

↓

```
['Student_no 7', 'Student_no 8' ..
. . . . ]
```

mapping with objects

```
let num = [1, 2, -3, -5]
let filtered = num.filter(value => value >= 0)
    clg(filtered)
let item = filtered.map(function(num){
    let obj = {value: num};
    return obj;
});
clg(item)
```

↑

```
[ {value: 1}, {value: 2} ]
```