https://shorturl.at/kAHNV

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Array Manipulators**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

==========================================================

Array Manipulators

==========================================================

1 map

2 filter

3 reduce

4 reduceRight

5 forEach

6 for...of

7 for...in

8 push

9 unshift

10 pop

11 shift

12 some

13 every

14 find

15 includes

16 splice

17 findIndex

18 slice

19 copyWithin

20 indexOf

21 sort

22 length

23 delete

24 from

25 join

26 fill

27 flat

28 reduce

29 flatMap

30 entries

31 fromEntries

32 split

33 lastIndexOf

34 concat

35 substr

36 substring

37 trim

01. map():-

- This function is used to manipulate each and every element in array

- it returns an array

//Eg01

let arr = [10, 20, 30, 40, 50]

//multiply each element by 2

console.log(arr.map((element, index) => {

return element \* 2

}))

//Eg02

let arr = [1, 2, 3, 4, 5]

//o/p ['$1','$2','$3','$4','$5']

console.log(arr.map((element, index) => {

return '$' + element

}))

//Eg03

let arr1 = [1,2,3]

let arr2 = ['one', 'two', 'three']

//o/p [ [ 1, 'one' ], [ 2, 'two' ], [ 3, 'three' ] ]

console.log(arr1.map((element, index)=>{

return [element, arr2[index]]

}))

02. filter():-

- this function creates an array based on condition

//Eg01

let arr1 = [10, 20, 30, 40, 50]

//create array with elements greater than 30

console.log(arr1.filter((element, index) => {

return element > 30

}))

//Eg02

let arr2 = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

//create an array elements greater than or equal to 100

console.log(arr2.filter((element, index) => {

return element >= 100

}))

//Eg03

let arr3 = [10, 20, 30, 40, 50]

//o/p [300, 400, 500]

console.log(arr3.map((element, index) => {

return element \* 10

}).filter((element, index) => {

return element > 200

}))

03. reduce() left to right 0 -> 1

04. reduceRight() right to left 0 <- 1

let arr = [1, 2, `3`, 4, 5]

console.log(arr.reduce((pv, cv) => {

return pv + cv

}))

console.log(arr.reduceRight((pv, cv) => {

return pv + cv

}))

05. forEach

06. for...of

07. for...in

let arr = [1, 2, 3, 4, 5]

console.log(arr)

for(let i = 0; i < arr.length; i++)

{

arr[i] \*=10

console.log(arr[i])

}

console.log(arr)

arr.forEach((element, index)=>{

element \*= 10

console.log(element)

})

console.log(arr)

for(let x of arr){

console.log(x)

}

let demo = {

'sub\_one': 'HTML',

"sub\_two": "CSS",

sub\_three: 'Bootstrap'

}

for(let x in demo)

//console.log(x)

console.log(demo[x])

08. push():- add element at end

09. unshift():- add element at begining

10. pop():- remove element from end

11. shift():- remove element from begining

let arr = [20, 30, 40]

console.log(arr) //[20, 30, 40]

arr.push(50)

console.log(arr) //[ 20, 30, 40, 50 ]

arr.unshift(10)

console.log(arr) //[ 10, 20, 30, 40, 50 ]

console.log(arr.pop()) //50

console.log(arr) //[ 10, 20, 30, 40 ]

console.log(arr.shift()) //10

console.log(arr) //[ 20, 30, 40 ]

11. some() :-if any one element in array satisfies the condition

then it will return true, otherwise false.

12. every():-if all elements in array satisfy the condition

then it will return true, otherwise false.

let arr = [10, 20, 30, 40, 50]

console.log(arr.some((element, index)=>{

return element > 10

})) //true

console.log(arr.every((element, index)=>{

return element > 10

})) //false

console.log(arr.some((element, index) => {

return element > 50

})) //false

console.log(arr.every((element, index) => {

return element <= 50

})) //true

14. find():-

- this function is used to find the element in array

- if element found, it will return the same element

- if element not found it will return undefined

15. includes():-

- it is boolean function used to check element present

in array or not

let arr = [10, 20, 30, 40, 50]

console.log(arr.find((element, index)=>{

return element == '30'

})) //30

console.log(arr.find((element, index)=>{

return element === '30'

})) //undefined

console.log(arr.includes(30))

console.log(arr.includes('30'))

16. splice() -> swiss army knife for arrays

https://javascript.info/array-methods

let arr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

console.log(arr) //[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

arr.splice(5, 2) //from index 5 delete TWO elements

console.log(arr) //[10, 20, 30, 40, 50, 80, 90, 100]

//delete 80

arr.splice(5, 1)

console.log(arr) //[ 10, 20, 30, 40, 50, 90, 100]

//delete 100

//arr.splice(6, 1)

//arr.splice(arr.length - 1, 1)

arr.splice(-1, 1)

console.log(arr) //[ 10, 20, 30, 40, 50, 90 ]

arr.splice(2, 2)

console.log(arr) //[ 10, 20, 50, 90 ]

//before 90 add 60, 70, 80

arr.splice(3, 0, 60, 70, 80)

console.log(arr) //[ 10, 20, 50, 60, 70, 80, 90 ]

//delete 50 and add 30, 40, 50

arr.splice(2, 1, 30, 40, 50)

console.log(arr) //[10, 20, 30, 40, 50, 60, 70, 80, 90 ]

//add 100 at end

arr.splice(arr.length, 0, 100)

console.log(arr)

17. findIndex():-

- it is used to find index of particular element

let arr = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

let idx = arr.findIndex((element, index) => {

return element === 30

})

console.log(arr) //[10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

console.log(idx) //4

arr.splice(idx, 1)

console.log(arr) //[10, 100, 20, 200, 300, 40, 400, 50, 500]

key = 40

arr.splice(arr.findIndex((element, index)=>{

return element == key

}), 1)

console.log(arr) //[10, 100, 20, 200, 300, 400, 50, 500]

let arr2 = [

{ p\_id: 111 },

{ p\_id: 1111 },

{ p\_id: 222 },

{ p\_id: 333 }

]

console.log(arr2) //[ { p\_id: 111 }, { p\_id: 1111 }, { p\_id: 222 }, { p\_id: 333 } ]

arr2.splice(arr2.findIndex((element)=>{

return element.p\_id == 1111

}), 1)

console.log(arr2) //[ { p\_id: 111 }, { p\_id: 222 }, { p\_id: 333 } ]

18. slice():-

let arr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//in slice first include last exclude

//-ve indices supported

console.log(arr) //[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

console.log(arr.slice(5, 7)) //[ 60, 70 ]

console.log(arr.slice(3, 6)) //[ 40, 50, 60 ]

console.log(arr.slice(5)) //[ 60, 70, 80, 90, 100 ]

console.log(arr.slice(5, -1)) //[ 60, 70, 80, 90 ]

console.log(arr.slice(5, -5))

19. copyWithin()

let arr1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//copy all elements at index 1 -> 1 index right

console.log(arr1.copyWithin(1)) //[10, 10, 20, 30, 40, 50, 60, 70, 80, 90 ]

let arr2 = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

console.log(arr2.copyWithin(5)) //[10, 100, 20, 200, 30, 10, 100, 20, 200, 30]

let arr3 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//copy all elements from index 5 at index 2

console.log(arr3.copyWithin(2, 5)) //[10, 20, 60, 70, 80, 90, 100, 80, 90, 100]

let arr4 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//copy all elements from index no 4 to 6 (excluding 6) at index 2

console.log(arr4.copyWithin(2, 4, 6)) //[10, 20, 50, 60, 50, 60, 70, 80, 90, 100 ]

20. indexOf() :- dont create index for duplicate elements

let arr = [10, 20, 30, 10, 40, 20, 40, 50]

arr.forEach((element, index) => {

console.log(element, index, arr.indexOf(element))

})

console.log(arr.filter((element, index)=>{

return arr.indexOf(element) === index

}))//this code removes duplicates

let mySet = new Set(arr)

console.log([...mySet])

console.log(Array.from(mySet))

21. sort()

let arr = [10, 50, 20, 40, 30]

console.log(arr) //[10, 50, 20, 40, 30]

console.log(arr.sort((num1, num2)=>{

return num1 - num2

})) //[ 10, 20, 30, 40, 50 ]

console.log(arr.sort((num1, num2)=>{

return num2 - num1

})) //[ 50, 40, 30, 20, 10 ]

22. length

let arr = [1, 2, 3, 4, 5]

console.log(arr) //[ 1, 2, 3, 4, 5 ]

console.log(arr.length) //5

console.log(arr[3]) //4

console.log(arr[5]) //undefined

arr.length = 3

console.log(arr[arr.length])//undefined

console.log(arr.length) //3

console.log(arr) //[ 1, 2, 3 ]

23. delete():- element deleted but memory not released

let arr = [1, 2, 3, 4, 5]

console.log(arr) //[ 1, 2, 3, 4, 5 ]

console.log(arr.length) //5

delete(arr[2])

console.log(arr.length) //5

console.log(arr) //[ 1, 2, <1 empty item>, 4, 5 ]

arr.length = 3

arr.length = 5

console.log(arr)

Formatted of above

=========================================================

Array Manipulators

=========================================================

**01. map():-**

- **This function is used to manipulate each and every element in array**

- **it returns an array**

//Eg01

let arr1 = [10, 20, 30, 40, 50]

//multiply each element by 2

console.log(arr1.map((element, index) => {

return element \* 2

}))

//Eg02

let arr2 = [1, 2, 3, 4, 5]

//o/p ['$1','$2','$3','$4','$5']

console.log(arr2.map((element, index) => {

return '$' + element

}))

//Eg03

let arr31 = [1, 2, 3]

let arr32 = ['one', 'two', 'three']

//o/p [ [ 1, 'one' ], [ 2, 'two' ], [ 3, 'three' ] ]

console.log(arr31.map((element, index) => {

return [element, arr32[index]]

}))

**02. filter():-**

- **this function creates array based on condition**

//Eg01

let arr1 = [10, 20, 30, 40, 50]

//create an array with elements greater than 30

console.log(arr1.filter((element, index) => {

return element > 30

}))

//Eg02

let arr2 = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

//create array with elements greater than or equal to 100

console.log(arr2.filter((element, index) => {

return element >= 100

}))

//Eg03

let arr3 = [10, 20, 30, 40, 50]

//o/p [300,400,500]

console.log(arr3.filter((element, index) => {

return element > 20

}).map((element, index) => {

return element \* 10

}))

**03. reduce() left to right 0 -> 1**

**04. reduceRight() right to left 0 <- 1**

//Eg01

let arr1 = [1, 2, 3, 4, 5]

console.log(arr1.reduce((fv, nv) => {

return fv + nv

}))

console.log(arr1.reduceRight((fv, nv) => {

return fv + nv

}))

//Eg02

let arr2 = [1, 2, 3, 4, `5`]

console.log(arr2.reduce((fv, nv) => {

return fv + nv

}))

console.log(arr2.reduceRight((fv, nv) => {

return fv + nv

}))

//Eg03

let arr3 = [`1`, 2, 3, 4, 5]

console.log(arr3.reduce((fv, nv) => {

return fv + nv

}))

console.log(arr3.reduceRight((fv, nv) => {

return fv + nv

}))

**05. forEach**

**06. for...of**

**07. for...in**

**08. push():- add element at end, returns new length of array**

**09. unshift():- add element at beginning, returns new length of array**

**10. pop():- remove element from end, returns removed element**

**11. shift():- remove element from beginning, returns removed element**

let arr = [20, 30, 40]

console.log(arr) //[ 20, 30, 40 ]

console.log(arr.push(50)) //4

console.log(arr) //[ 20, 30, 40, 50 ]

console.log(arr.unshift(10))//5

console.log(arr) //[ 10, 20, 30, 40, 50 ]

console.log(arr.pop()) //50

console.log(arr) //[ 10, 20, 30, 40 ]

console.log(arr.shift()) //10

console.log(arr) //[ 20, 30, 40 ]

**12. some():- if any one element in the array satisfies the condition then it will return true, otherwise false.**

**13. every():- if all elements in the array satisfy the condition then it will return true, otherwise false.**

let arr = [10, 20, 30, 40, 50]

console.log(arr.some((element, index) => {

return element > 10

})) //true

console.log(arr.every((element, index) => {

return element > 10

})) //false

console.log(arr.some((element, index) => {

return element > 50

})) //false

console.log(arr.every((element, index) => {

return element <= 50

})) //true

**14. find() :-**

- **this function is used to find an element in array**

- **if element found it will return the same element**

- **if an element is not found it will return undefined.**

**15. includes() :-**

- **it is boolean function used to check element is present in array or not**

let arr = [10, 20, 30, 40, 50]

console.log(arr.find((element, index) => {

return element == `30`

})) //30

console.log(arr.find((element, index) => {

return element === `30`

})) //undefined

console.log(arr.includes(30)) //true

console.log(arr.includes('30')) //false

**16. splice() -> swiss army knife for arrays**

**https://javascript.info/array-methods**

let arr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

console.log(arr) //[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

arr.splice(5, 2) //from index 5 delete TWO elements

console.log(arr) //[10, 20, 30, 40, 50, 80, 90, 100]

//delete 80

arr.splice(5, 1)

console.log(arr) //[10, 20, 30, 40, 50, 90, 100]

//delete 100

//arr.splice(6, 1)

arr.splice(-1, 1)

console.log(arr) //[ 10, 20, 30, 40, 50, 90 ]

arr.splice(2, 2)

console.log(arr) //[ 10, 20, 50, 90 ]

//before 90 add 60, 70, 80

arr.splice(3, 0, 60, 70, 80)

console.log(arr) //[10, 20, 50, 60, 70, 80, 90]

//delete 50 and add 30, 40, 50

arr.splice(2, 1, 30, 40, 50)

console.log(arr) //[10, 20, 30, 40, 50, 60, 70, 80, 90]

//add 100 at end

arr.splice(9, 0, 100)

console.log(arr) //[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

**17. findIndex():-**

**- it is used to find index of particular element**

let arr = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

let idx = arr.findIndex((element, index) => {

return element == 30

})

console.log(idx) //4

console.log(arr) //[10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

arr.splice(idx, 1)

console.log(arr) //[10, 100, 20, 200, 300, 40, 400, 50, 500]

key = 40

arr.splice(arr.findIndex((element, index) => {

return element == key

}), 1)

console.log(arr) //[10, 100, 20, 200, 300, 400, 50, 500]

let arr2 = [

{ p\_id: 111 },

{ p\_id: 1111 },

{ p\_id: 222 },

{ p\_id: 333 }

]

console.log(arr2)

arr2.splice(arr2.findIndex((element, index) => {

return element.p\_id == 1111

}), 1)

console.log(arr2)

**18. slice():-**

let arr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//in slice first include last exclude

//-ve indices supported

console.log(arr) //[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

console.log(arr.slice(5, 7)) //[ 60, 70 ]

console.log(arr.slice(3, 7)) //[ 40, 50, 60, 70 ]

console.log(arr.slice(5)) //[ 60, 70, 80, 90, 100 ]

console.log(arr.slice(5, -2)) //[ 60, 70, 80 ]

console.log(arr.slice(5,-5)) //[]

**19. copyWithin()**

let arr1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//copy all elements at index 1

console.log(arr1.copyWithin(1)) //[10, 10, 20, 30, 40, 50, 60, 70, 80, 90]

let arr2 = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]

console.log(arr2.copyWithin(5)) //[10, 100, 20, 200, 30, 10, 100, 20, 200, 30]

let arr3 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//copy all elements from index 5 at index 2

console.log(arr3.copyWithin(2, 5)) //[10, 20, 60, 70, 80, 90, 100, 80, 90, 100]

let arr4 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

//copy all elements from index no 4 to 6 at index 2

console.log(arr4.copyWithin(2, 4, 6)) //[10, 20, 50, 60, 50, 60, 70, 80, 90, 100]

**20. indexOf():- don't create index for duplicate elements**

let arr = [10, 20, 30, 10, 40, 20, 40, 50]

arr.forEach((element, index) => {

console.log(element, index, arr.indexOf(element))

})

console.log(arr.filter((element, index) => {

return arr.indexOf(element) === index

})) //this code removes duplicates

//is there any other easy way to remove duplicates ?

let mySet =[ ...new Set(arr)]

console.log(mySet)

**21. sort()**

let arr = [10, 50, 20, 40, 30]

console.log(arr)

console.log(arr.sort((num1, num2) => {

return num1 - num2

})) //[ 10, 20, 30, 40, 50 ]

console.log(arr.sort((num1, num2) => {

return num2 - num1

})) //[ 50, 40, 30, 20, 10 ]

**22. length**

let arr = [1, 2, 3, 4, 5]

console.log(arr)

console.log(arr.length) //5

console.log(arr[3])

console.log(arr[arr.length]) //?

arr.length = 3

console.log(arr[3])

console.log(arr.length)

console.log(arr

**23. delete() :- element deleted but memory not released**

let arr = [10,20,30,40,50]

console.log(arr) //[ 10, 20, 30, 40, 50 ]

console.log(arr.length) //5

delete(arr[2])

console.log(arr.length) //5

console.log(arr) //[ 10, 20, <1 empty item>, 40, 50 ]

arr.length = 3

arr.length = 5

console.log(arr) //?

**24. from() :- string to array**

**25. join() :- array to string**

let str = 'Hello'

let arr = Array.from(str)

console.log(arr)

console.log(arr.join(""))

**26. fill():- element replacement**

let arr = [10, 20, 30, 40, 50]

console.log(arr) //[ 10, 20, 30, 40, 50 ]

console.log(arr.fill(100)) //[ 100, 100, 100, 100, 100 ]

console.log(arr.fill(200, 2)) //[ 100, 100, 200, 200, 200 ]

console.log(arr.fill(300, 2, 4))//[ 100, 100, 300, 300, 200 ]

**27. flat()**

let arr = [1, [2], [3], [4, [5]]]

console.log(arr) //[ 1, [ 2 ], [ 3 ], [ 4, [ 5 ] ] ]

console.log(arr.flat(1))

console.log(arr.flat(2))

//if we dont know level

let arr2 = [1,[[[[2]]]],[3],[[[[[[[[[[[[[[[[[[[[[4]]]]]]]]]]]]]]]]]]]]]]

console.log(arr2.flat(Infinity))

**28. reduce()**

**29. flatMap() :- combination of flat() and map()**

let arr1 = [1, 2, 3]

let arr2 = ['one', 'two', 'three']

console.log(arr1.map((element, index) => {

return [element, arr2[index]]

})) //[ [ 1, 'one' ], [ 2, 'two' ], [ 3, 'three' ] ]

console.log(arr1.flatMap((element, index) => {

return [element, arr2[index]]

})) //[ 1, 'one', 2, 'two', 3, 'three' ]

**30. entries() :- object to array**

**31. fromEntries():- array to object**

**32. split()**

let str = `Welcome to Javascript`

console.log(str.split()) //[ 'Welcome to Javascript' ]

console.log(str.split(" ")) //[ 'Welcome', 'to', 'Javascript' ]

let myStr = 'Mahabharat'

console.log(myStr.split('a')) //[ 'M', 'h', 'bh', 'r', 't' ]

console.log(myStr.split('a', 3))//[ 'M', 'h', 'bh' ]

**33. lastIndexOf()**

let arr = [10, 20, 10, 20, 30, 10]

console.log(arr.lastIndexOf(10)) //5

console.log(arr.lastIndexOf(20)) //3

**34. concat()**

let arr1 = [10]

let arr2 = [20]

let arr3 = [30]

let arr4 = arr1.concat(arr2, arr3)

console.log(arr4) //[ 10, 20, 30 ]

**35. substr()**

**36. substring()**

let str = `Welcome to Javascript`

//Welcome

console.log(str.substr(0, 7))

console.log(str.substring(0, 7))

//to

console.log(str.substr(8, 2))

console.log(str.substring(8, 10))

//Javascript

console.log(str.substr(11))

console.log(str.substring(11))

**37. Trimming functions**

let str = ` Welcome `

console.log(str.length) //9

console.log(str.trim().length) //7

console.log(str.trimStart().length) //8

console.log(str.trimEnd().length) //8

**38. replace() :- This function is used for complete or partial replacement of string**

//Eg01

let str = 'School'

let res = str.replace('School','College')

console.log(str)

console.log(res)

//Eg02

let str = `This is my School`

let res = str.replace('School','College')

console.log(str)

console.log(res)

//Eg03

let str = "red green Red red Green Red"

let res = str.replace(/red/,"Yellow") //only first occurence

console.log(str) //red green Red red Green Red

console.log(res) //Yellow green Red red Green Red

res = str.replace(/red/g,"Yellow") //all occurences

console.log(res) //Yellow green Red Yellow Green Red

res = str.replace(/red/ig,"Yellow") //all occurences ignore case

console.log(res) //Yellow green Yellow Yellow Green Yellow

**39. search():- This function returns the index of first match string**

**returns -1 for unsuccessful search**

let str = "Sound mind in sound body"

console.log(str)

console.log(str.search('sound')) //14

console.log(str.search('Sound')) //0

console.log(str.search(/sound/i)) //0

console.log(str.search('refresh')) //-1

**40. toLocaleLowerCase()**

**41. toLocaleUpperCase()**

- these functions are similar to toLowerCase() and toUpperCase() respectively,

- the difference is that toLocaleLowerCase() and toLocaleUpperCase() functions produce outputs depend on local language of that particular region (i.e. in browser's local language)

let str = "istambul"

let res = str.toUpperCase()

let res1 = str.toLocaleUpperCase('tr')

console.log(str)

console.log(res)

console.log(res1)

**42. charCodeAt():- this function returns the unicode of the character at the specified index in a string.**

**//http://www.columbia.edu/kermit/ucs2.html**

let str = "aAbB"

console.log(str.charCodeAt(0)) //97

console.log(str.charCodeAt(1)) //65

console.log(str.charCodeAt(2)) //98

console.log(str.charCodeAt(3)) //66

**43. valueOf():-returns the primitive value of String object**

**44. toString()**

String.toString() -> converts String object to string

Number.toString() -> method converts a number to a string with base as argument (from 2 to 36)

let str = new String("ABC")

let res = str.valueOf()

console.log(str) //[String: 'ABC']

console.log(res) //ABC

let res1 = str.toString()

console.log(res1) //ABC

let num = 91

console.log(num.toString())

console.log(num.toString(2)) //1011011

console.log(num.toString(8)) //133

console.log(num.toString(16)) //5b

**45. match():-this function accepts regular expression as argument and returns array of matches and returns null if match not found**

let str = 'Importance given to Portfolio'

console.log(str.match(/port/g)) //[ 'port' ]

console.log(str.match(/port/ig)) //[ 'port', 'Port' ]

console.log(str.match(/airport/g)) //null

console.log(str.match(/airport/ig)) //null

=============================================================

call apply and bind functions

=============================================================

call():-

- This function is used to create relationship between two unknown memory locations.

apply():-

- It is same as that of call function

- When we have to pass arguments as an array this function is used.

(array implies no independent arguments).

bind():-

- this function is used to merge two unknown memory locations.

- this function returns a new function

let obj = {

num: 10

}

console.log(obj)

function myFun(arg) {

return this.num + arg

}

console.log(myFun.call(obj, 10))

function newFun(arg1, arg2, arg3) {

return this.num + arg1 + arg2 + arg3

}

console.log(newFun.call(obj, 20, 30, 40))

console.log(newFun.apply(obj, [20, 30, 40]))

let bindFun = newFun.bind(obj)

console.log(bindFun(1, 2, 3))

=============================================================

Closure:-

=============================================================

- Inner function can have access of data from outer function.

- Outer function returns inner function.

- Closure means inner function can have access of data from

outer function even after returning inner function.

function add(x) {

return (y) => {

return x + y

}

}

//here outer function returned inner function

let var\_a = add(5)

let var\_b = add(10)

//here we called returned inner function with access of variable

//from outer function

console.log(var\_a(7))

console.log(var\_b(14))

==========================================================

IIFE:-

==========================================================

- Immediately invoked function expression.

- Introduced in ES9.

- These are self invokable functions, i.e. no need to call IIFEs.

- Syntax

(()=>{})()

//Eg01

(()=>{

console.log('Welcome to IIFE')

})()

//Eg02

((arg1, arg2) => {

console.log(arg1 + arg2)

})(10,20)

//Eg03

let res = (()=>{

return `Good Afternoon`

})()

console.log(res)

================================================================

================================================================