Arrays

```
// DECLARING AN ARRAY
let someName = "Hello there!";
let num = 23;
//array declaration starts with "[" and ends with "]"
let arr = [2, someName, num, 'hello'] //we can put any value, or expression that evaluates into a value into an array

// ACCESSING VALUES IN ARRAY
//by accessing value at index
let someValue = arr[0]; //assigns the value of the 0th index of arr, the value 2, into the variable someValue
//by looping through all indices
for(let i=0; i<arr.length; i++) { //loop from 0 index to array length
        console.log(arr[i]);
}
//CHECKING IF VARIABLE IS AN ARRAY
typeof arr //returns 'object'.  so typeof is not useful to determine an array
Array.isArray(arr) //returns true
```

Arrays - common methods

```
let arr = [2, 1, 4]
//>> arr.push(val)
//        changes arr; adds val to end of arr
//        returns new length of arr
let len = arr.push("hello") //now, arr is [2, 1, 4, "hello"], returned value is 4 which is new length of arr
//>> arr.pop()
//        changes arr; removes last element from arr
//        returns the last element of arr
let val = arr.pop() //now, arr is [2, 1, 4], returned value assigned to val is "hello", which was the last element of arr
//>> arr.shift()
//        changes arr; removes first element from arr
//        returns the first element of arr
let val2 = arr.shift() //now, arr is [1, 4], returned value assigned to val2 is 2, which was the first element of arr
//>> arr.unshift(val)
//        changes arr; adds val to beginning of arr
//        returns new length of arr
let len2 = arr.unshift("hello") //now, arr is ["hello", 1, 4], returned value assigned to len2 is 3, which is new length of arr
```

Arrays - common methods II

```
let arr = [2, "hello", 4, 1, 5]
//>> arr.indexOf(val)
//      does not change arr
//      returns the index at which val is located in arr, or returns -1 if val is not in arr
let idx = arr.indexOf(4) //idx will have value 2, because value 4 is in arr at index 2
let idx1 = arr.indexOf(24) //idx1 will have value -1, because 24 is not in arr
//>> arr.includes(val)
//      does not change arr
//      returns true if val is in arr, returns false if val is not in arr
let isThereValue = arr.includes(4) //will return true assigned to variable isThereValue
//>> arr.slice(startIdx, endIdx)
//      does not change arr
//      returns new array from startIdx of arr to endIdx-1 of arr
let anotherArray = arr.slice(1, 3) //will return array ["hello", 4] assigned to anotherArray
let oneMoreArray = arr.slice() //will return same value as arr, [2, "hello", 4, 1, 5] to oneMoreArray
```

Arrays - common methods III

```
let arr = [2, 5, 4, 3, 6]
//>> arr.reverse()
//        changes arr; reverses arr
//        returns the reversed array
let rev = arr.reverse() //arr will have value [6, 3, 4, 5, 2].  rev will also have [6, 3, 4, 5, 2]
//>> arr.splice(startIdx, count, replace1, replace2)
//        changes arr; removes count number of elements in arr,
//        starting from startIdx, and adds replace1, replace2 etc. in that location of arr
//        returns the removed elements as an array
let bArr = arr.splice(1, 3, 'hey', 'you') //arr will now have [6, 'hey', 'you', 2], bArr will have [3, 4, 5]
// we can also use arr.splice() to only insert values into middle of array
let cArr = arr.splice(3, 0, 'you') //arr will now have [6, 'hey', 'there','you', 2], cArr will have []
//>> arr.join(val)
//        does not change arr
//        returns a string of all elements of arr joined with val. if no val, then joined with ","
let joinedStr = arr.join(" – ") //joinedStr will be "6–hey–there–you–2"
let joinedStr2 = arr.join() //joinedStr2 will be "6,hey,there,you,2"
```

Arrays - common methods IV

```
let arr = [2, 5, 4, 3, 6]

let arr2 = [7, 8]

//>> arr.concat(arr2)

//      does not change arr

//      returns an array with arr2 added to end of arr

let concatArr = arr.concat(arr2) //concatArr will be [2, 5, 4, 3, 6, 7, 8]

// ACCESSING VALUES IN MULTI DIMENSION ARRAYS

let arr3 = [12, 23, 34, [45, 56], 67]

let someValue = arr[3][0]; //someValue will have 45, which is [0]th element of [45, 56]

let someValue2 = arr[3][1]; //someValue will have 56, which is [1]st element of [45, 56]
```

Objects

// DECLARING AN OBJECT

```
let obj = {name: 'Raju', isFellow: true, addr: {city: 'Ktm', ctry: 'Nepal'}}
//obj declaration starts with "{" and ends with "}"
```

// ACCESSING VALUES IN OBJECT

```
//by [key] : putting key as a string
let someValue = obj["name"]; //assigns the value 'Raju', which is in key 'name' of obj
//by . operator : putting key after "."
let anotherValue = obj.name //assigns the value 'Raju' to anotherValue, which is in key 'name' of obj
let objFrom = obj.addr.city //assigns the value 'Ktm' to objFrom, which is in key 'city' of key 'addr' of obj
```

//CHECKING typeof OBJECT

```
typeof obj //returns 'object'
```

// ADDING VALUE IN OBJECT

```
obj.age = 26 // creates new key 'age' in obj and assigns it the value 26
obj['age2'] = 26 // creates new key 'age2' in obj and assigns it the value 26
```

Objects - accessing all key/value pairs

```
let obj = {name: 'Raju', isFellow: true, addr: {city: 'Ktm', ctry: 'Nepal'}}
//>> 'in' WILL CHECK IF KEY IS IN OBJECT
let isItTrue = 'name' in obj //isItTrue will have value true, because 'name' key exists in obj
//>> looping through keys in object
for(let someKey in obj) {
        console.log(obj[someKey]) //loop through all keys in obj and put key in someKey one at a time
}
//>> Object.keys(obj) will return an array with all the keys in obj
let objKeys = Object.keys(obj) //objKeys will be the array ['name', 'isFellow', 'addr']
```

Objects - methods

```
let obj = {name: 'Raju', isFellow: true,
        sayHello: function(str) {
                return `hello ${str}, my name is ${this.name}`
        },
        callFunc: function(paramFunc, val) {
                return paramFunc(val)
        }
}
// 'this': someFunc is a method with 'this'
let retStr = obj.sayHello('Subash') //retStr will have "hello Subash, my name is Raju"
//when calling obj.sayHello, in the code block of obj.sayHello, 'this' will be replaced by obj
//so the return statement will now be "return `hello ${str}, my name is ${obj.name}`

// passing function as argument: obj.callFunc takes 2 params paramFunc, and val
function newFunc(name) {return `hello ${name}`}
// obj.callFunc, in the code block, executes the newFunc by passing it val
let retStr = obj.callFunc(newFunc, "buddy") //retStr will have "hello buddy".  obj.callFunc calls newFunc by passing "buddy"
```