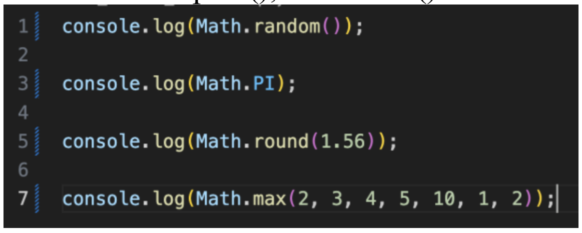
JAVASCRIPT Basics

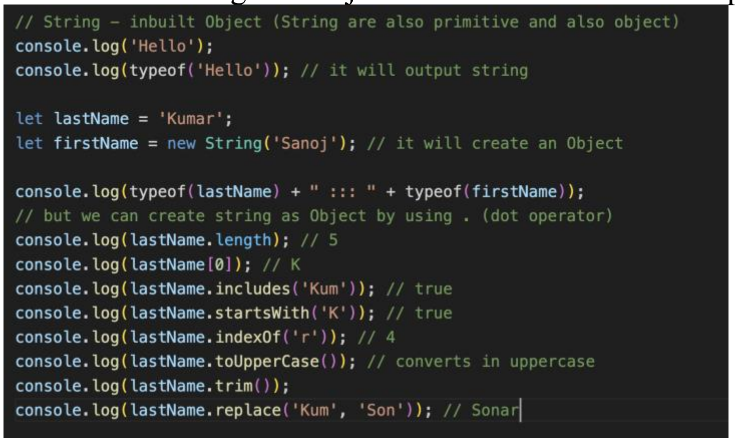
Inbuilt Objects

- Math:
    - Giving some mathematic related function and constant.
    - Like Math.pow(), Math.max() etc.



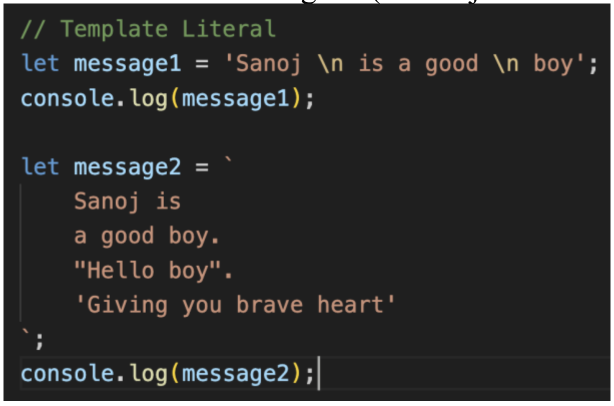
```javascript
console.log(Math.random());

console.log(Math.PI);

console.log(Math.round(1.56));

console.log(Math.max(2, 3, 4, 5, 10, 1, 2));
```

- String:
    - String can be primitive and can also be object.
    - If we are using string as normal for output purpose then it does not have any properties.
    - But if we are using it as object then it has these kinds of properties.



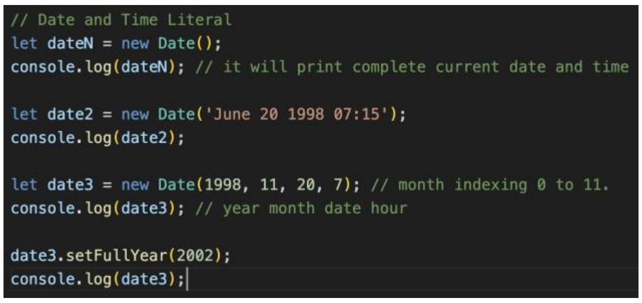
```javascript
// String - inbuilt Object (String are also primitive and also object)
console.log('Hello');
console.log(typeof('Hello')); // it will output string

let lastName = 'Kumar';
let firstName = new String('Sanoj'); // it will create an Object

console.log(typeof(lastName) + " ::: " + typeof(firstName));
// but we can create string as Object by using . (dot operator)
console.log(lastName.length); // 5
console.log(lastName[0]); // K
console.log(lastName.includes('Kum')); // true
console.log(lastName.startsWith('K')); // true
console.log(lastName.indexOf('r')); // 4
console.log(lastName.toUpperCase()); // converts in uppercase
console.log(lastName.trim());
console.log(lastName.replace('Kum', 'Son')); // Sonar
```

- Template Literal:
    - If we want to create some kind of paragraph type of things.
    - If we want to add dynamically numbers from input while forming the string there, we can use template literal.
    - If we want to use single quote there, we need to use some kind of \' or Escape Sequence.
    - We can do this using ` `. (button just below the escape button).



```javascript
// Template Literal
let message1 = 'Sanoj \n is a good \n boy';
console.log(message1);

let message2 = `
    Sanoj is
    a good boy.
    "Hello boy".
    'Giving you brave heart'
`;
console.log(message2);
```

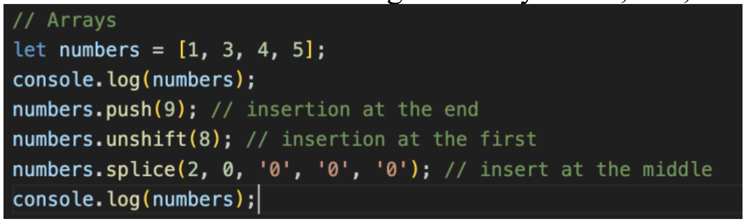
- Date and Time
    - Read at MDN Docs.



```javascript
// Date and Time Literal
let dateN = new Date();
console.log(dateN); // it will print complete current date and time

let date2 = new Date('June 20 1998 07:15');
console.log(date2);

let date3 = new Date(1998, 11, 20, 7); // month indexing 0 to 11.
console.log(date3); // year month date hour

date3.setFullYear(2002);
console.log(date3);
```

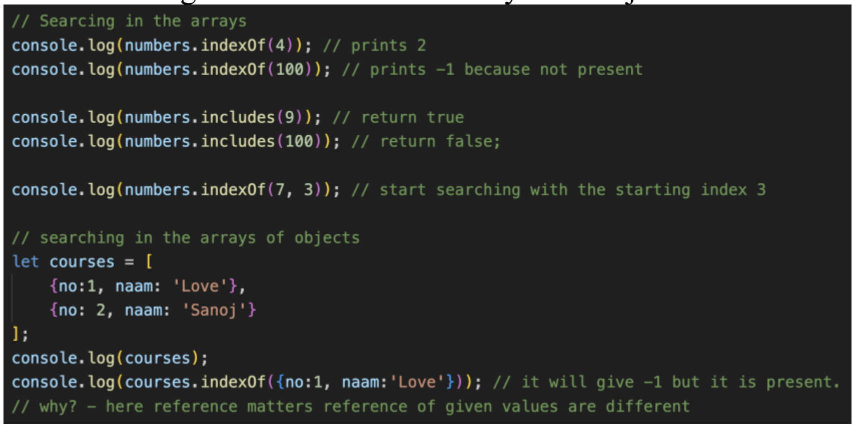
Arrays: Collection of items (same or different)

- Adding New Elements:
  - Indexing will be given from 0 to length - 1.
  - Insertion can be done using three ways: start, end, middle



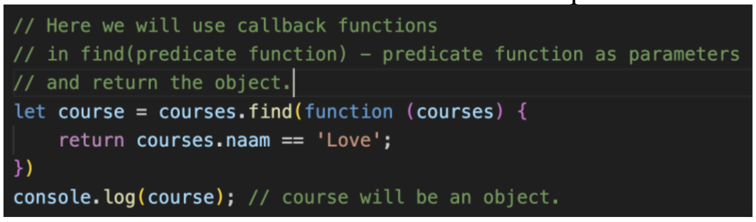
```
// Arrays
let numbers = [1, 3, 4, 5];
console.log(numbers);
numbers.push(9); // insertion at the end
numbers.unshift(8); // insertion at the first
numbers.splice(2, 0, '0', '0', '0'); // insert at the middle
console.log(numbers);
```

- Finding Elements:
  - For finding out the elements in array or in objects/references.



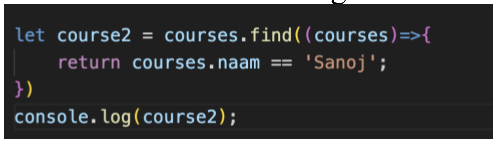
```
// Searcing in the arrays
console.log(numbers.indexOf(4)); // prints 2
console.log(numbers.indexOf(100)); // prints -1 because not present

console.log(numbers.includes(9)); // return true
console.log(numbers.includes(100)); // return false;

console.log(numbers.indexOf(7, 3)); // start searching with the starting index 3

// searching in the arrays of objects
let courses = [
    {no:1, naam: 'Love'},
    {no: 2, naam: 'Sanoj'}
];
console.log(courses);
console.log(courses.indexOf({no:1, naam:'Love'})); // it will give -1 but it is present.
// why? - here reference matters reference of given values are different
```

  - Now to search in array of object/reference we will use callback function
  - Callback functions is a function passed into other functions as an argument, which is then invoked inside the outer function to complete some kind of routine or actions.



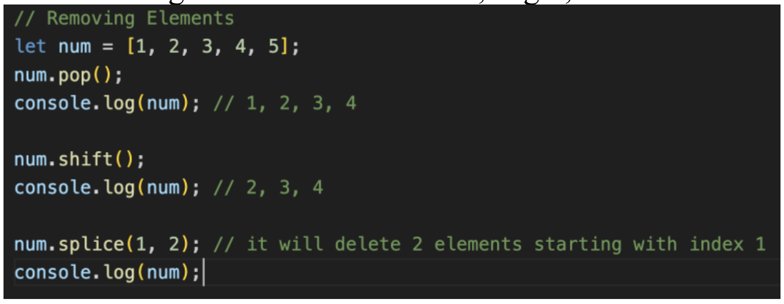
```
// Here we will use callback functions
// in find(predicate function) - predicate function as parameters
// and return the object.
let course = courses.find(function (courses) {
    return courses.naam == 'Love';
})
console.log(course); // course will be an object.
```

  - This can be done using arrow functions also.



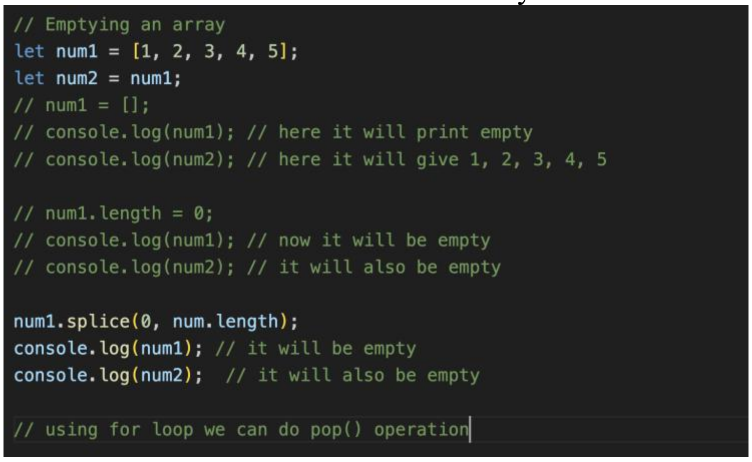
```
let course2 = courses.find((courses)=>{
    return courses.naam == 'Sanoj';
})
console.log(course2);
```

- Removing Element:
  - Removing the elements from end, begin, middle.



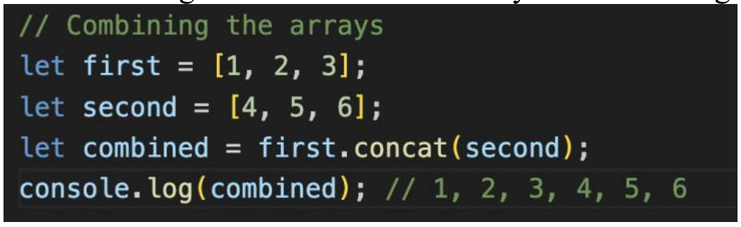
```
// Removing Elements
let num = [1, 2, 3, 4, 5];
num.pop();
console.log(num); // 1, 2, 3, 4

num.shift();
console.log(num); // 2, 3, 4

num.splice(1, 2); // it will delete 2 elements starting with index 1
console.log(num);
```

- Emptying the array:
    - To delete all the elements of the arrays



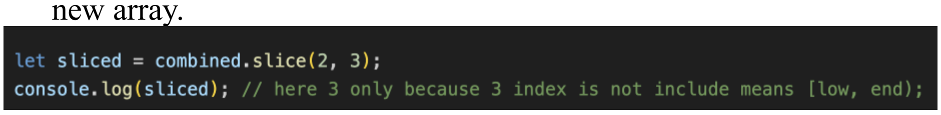
```
// Emptying an array
let num1 = [1, 2, 3, 4, 5];
let num2 = num1;
// num1 = [];
// console.log(num1); // here it will print empty
// console.log(num2); // here it will give 1, 2, 3, 4, 5

// num1.length = 0;
// console.log(num1); // now it will be empty
// console.log(num2); // it will also be empty

num1.splice(0, num.length);
console.log(num1); // it will be empty
console.log(num2);  // it will also be empty

// using for loop we can do pop() operation
```

- Combining and slicing the arrays:
    - Combining the elements of two arrays into one using concat() method.



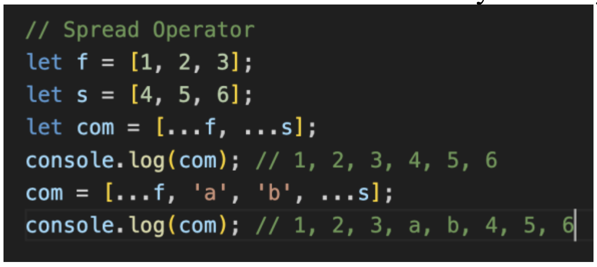
```
// Combining the arrays
let first = [1, 2, 3];
let second = [4, 5, 6];
let combined = first.concat(second);
console.log(combined); // 1, 2, 3, 4, 5, 6
```

- Slicing is done when we want to delete some array of the array element and wants to store in new array.



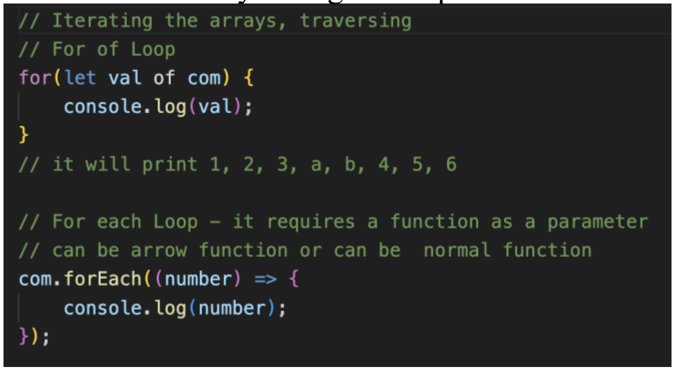
```
let sliced = combined.slice(2, 3);
console.log(sliced); // here 3 only because 3 index is not include means [low, end);
```

- Spread Operator:
    - It is also used to concatenate arrays and objects.



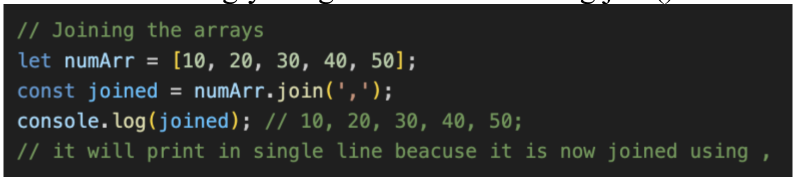
```
// Spread Operator
let f = [1, 2, 3];
let s = [4, 5, 6];
let com = [...f, ...s];
console.log(com); // 1, 2, 3, 4, 5, 6
com = [...f, 'a', 'b', ...s];
console.log(com); // 1, 2, 3, a, b, 4, 5, 6
```

- Iterating an Array:
    - Iterate the arrays using for loop and much more



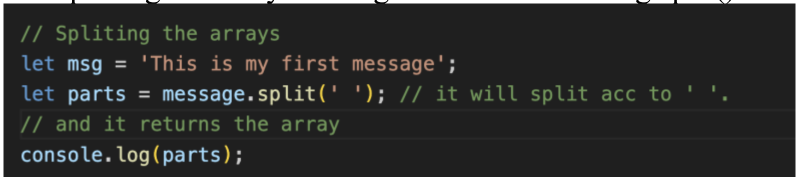
```
// Iterating the arrays, traversing
// For of Loop
for(let val of com) {
    console.log(val);
}
// it will print 1, 2, 3, a, b, 4, 5, 6

// For each Loop - it requires a function as a parameter
// can be arrow function or can be  normal function
com.forEach((number) => {
    console.log(number);
});
```

- Joining the Arrays:
    - Join accordingly via given condition using join().



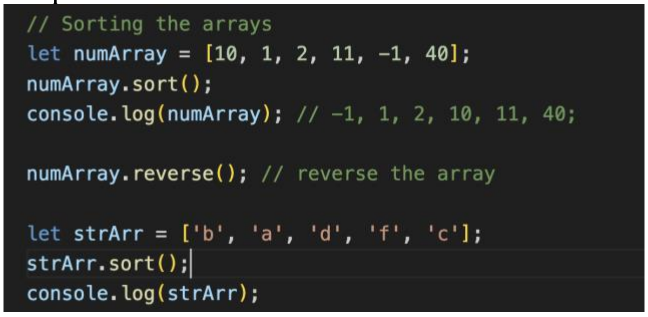
```
// Joining the arrays
let numArr = [10, 20, 30, 40, 50];
const joined = numArr.join(',');
console.log(joined); // 10, 20, 30, 40, 50;
// it will print in single line beacuse it is now joined using ,
```

- Splitting Elements:
    - Splitting the array acc to given condition using split().



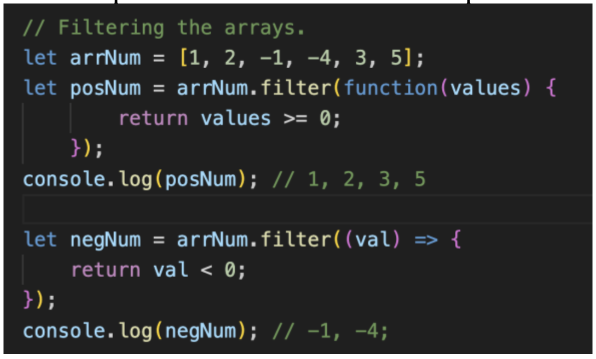
```
// Spliting the arrays
let msg = 'This is my first message';
let parts = message.split(' '); // it will split acc to ' '.
// and it returns the array
console.log(parts);
```

- Sorting arrays:
    - Sort the arrays in ascending / descending or in some order using sort().
    - It can be sort in any order according to our use we can do this via sending the function as a parameter to the sort function.



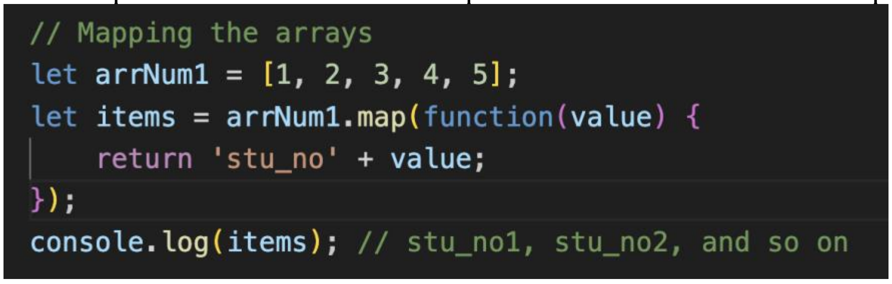
```
// Sorting the arrays
let numArray = [10, 1, 2, 11, -1, 40];
numArray.sort();
console.log(numArray); // -1, 1, 2, 10, 11, 40;

numArray.reverse(); // reverse the array

let strArr = ['b', 'a', 'd', 'f', 'c'];
strArr.sort();
console.log(strArr);
```

- Filtering Arrays:
    - Filter the arrays according to some condition using filter().
    - It requires another function as a parameter which tell how to filter the array elements.



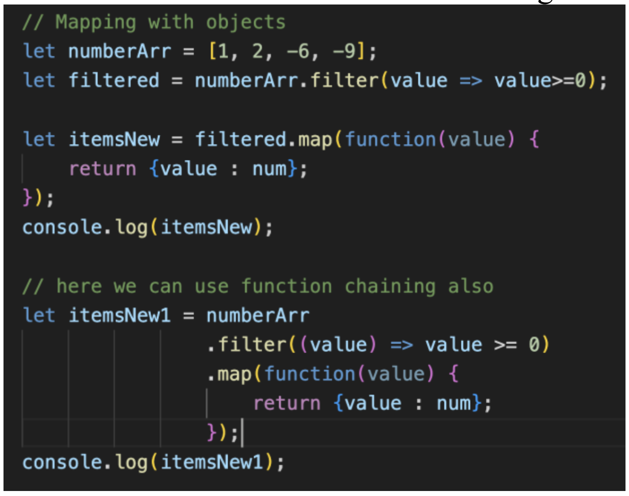
```
// Filtering the arrays.
let arrNum = [1, 2, -1, -4, 3, 5];
let posNum = arrNum.filter(function(values) {
        return values >= 0;
    });
console.log(posNum); // 1, 2, 3, 5

let negNum = arrNum.filter((val) => {
    return val < 0;
});
console.log(negNum); // -1, -4;
```

- Mapping in Arrays:
    - Maps each element of the array with something else using map().
    - It requires another function as a parameter which tell how to map or which to map.



```
// Mapping the arrays
let arrNum1 = [1, 2, 3, 4, 5];
let items = arrNum1.map(function(value) {
    return 'stu_no' + value;
});
console.log(items); // stu_no1, stu_no2, and so on
```

- Mapping with objects:
    - To map objects using some condition.
    - Here we can also use function chaining.



```
// Mapping with objects
let numberArr = [1, 2, -6, -9];
let filtered = numberArr.filter(value => value>=0);

let itemsNew = filtered.map(function(value) {
    return {value : num};
});
console.log(itemsNew);

// here we can use function chaining also
let itemsNew1 = numberArr
                .filter((value) => value >= 0)
                .map(function(value) {
                    return {value : num};
                });
console.log(itemsNew1);
```