We can pass array element as individual arguments

```
function sum(x,y,z){
    return x+y+z;
}

const numbers=[1,2,3];
console.log(sum(...numbers));
```

Currying :-

```
function add(x){
    return function(y){
        return x+y;
    };
}

console.log(add(3)(4));
```

PAGE NO. :
DATE : / /

What is async in JS

It is a keyword to creat async function.

~~async function ()~~

async function getData() {

either we can return promise

}

This function always return a promise, ~~either~~ if you return a value or integer it simple wrapped into a promise and returned.

Async and await are used to handle promises

PAGE NO.:
DATE: / /

Note

"await can be used inside the async function"

Async function will wait for the promise to resolve the only next line will execute.

JS engine used to wait for the promise to resolve.

When callstack sees any await, then that function in which await is there suspend for some time. and it will out from call stack.

Now when promise will resolve then that function start executing in call stack from where it left.

PAGE NO.:
DATE: / /

Debouncing and throttling →

When you need to optimize the performance of your code that runs repeatedly within a short period of time. These techniques allows you to control the rate at which your code is executed and reduce the number of times it is called.

Rebouncing is a technique that delays the execution of a function until the user stops performing a certain action for a specific period of time.

like a send mail which is sending the mail, you can debounce the function that makes the API call, so that after some delay only you will able to

PAGE NO.:
DATE: / /

Searchbar :-

When user stop searching the only it will make a API call, this way we can avoid making too many API calls that might overload server

Throttling :- It is a technique that limits the execution of a function to once in every specified time interval.

Call method

We are borrowing some function from different object.

```
let name ={
    firstname : "akshay"
    secondname : "saini"
    printfullname : function(){
        console.log(this.firstname
                    this.lastname);
    }
}
```

```
let name2 = {
    firstname : "sachin",
    secondname : "tendulkar"
}
```

```
name.printfullname.call(name2);
```

Apply method

Suppose we have more arguments.

Apply method takes arguments in an array.

Bind method

Same as call, but it will return a function.

```
let myname = name.printfullname
                .bind(name2);

myname();
```

When we define a variable undefined then we are trying to convey that variable value doesn't exist. When we define a variable to null then we are trying to convey that the variable is empty.

Debouncing

Eg:-

Html

```
<input type=text
onkeyup="result()"
>
```

JS

```
let count=0;
function response(){
    console.log(++count);
}
function debounce(attrl,
                  delay){
let timer;
return function(){
let context=this,
args=arguments
clearTimeout(timer);
timer=setTimeout(
function(){
attrl.apply
(context,
args);
},delay)
}
const result=
debounce(
response,300);
```

Factorial(5)

$n * (n-1) * (n-2)$

PAGE NO.:
DATE: / /

So, when we call result(), it sets a timer to wait for 300 ms. If you call 'result()' multiple times with in that 300 ms, it will keep resetting the timer, once the 300 ms have passed without any new calls, it finally execute the response function, logging the incremented count to the console. This mechanism prevents the response function from being called too frequently,

Find method	Filter method
• It returns a single element when condition matches.	• It returns a whole array of all the element which matches condition.