# Spread

```
function sumOne (x, y)
  return x+y;
}
let vari =[9 , 7 ];
console. log (sumone(... vari));  //spread
```

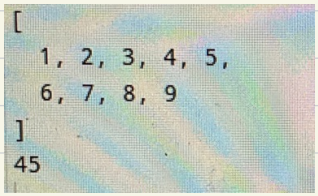
data will be divided into chunks

# Rest

collecting data into array

```
function sum (... args ) {
  //Rest Operator
  console.log (args);
  let sum = 0;
  for (const x of args ){
     sum = sum + x;
  }
  return sum;
}
console.log (sum (1,2,3,4,5,6,7,8,9,0));
```

output: [1,2,3,4,5, 6,7,8,9,0




when we dont know how many inputs we have from user

# Try and Catch

```
try {
  // logic error
} catch ( ) {
    // Error handling message
}
```

eg: try {
    database
} catch (err) {
    console.log("Database request failed");
}

eg: try {
    let first_name = "Anurag";
    console.log(first_name + " " + last_name);
} catch (err) {
    console.log("Variable name missing");

* Finally: regardless of the situation if we are getting or not getting it will run. (if error is thrown or not)

eg: try {
    let first_name = "Anurag";
    console.log(first_name + " " + last_name);
} catch (err) {
    console.log("Variable name missing");
Finally {
    console.log("I will run");
}

HW: Throw

# Errors in JS

* Reference Error
* Syntax Error
* Type Error

* In JS we do not have /0 error. (divide by 0)

# promise

- pending
- fullfilled
- rejected

# Event loop

```
const userone = () => {
 console.log ("hello one");
};
const usertwo = () => {
setTimeout ( () => {
 console.log ("Hello I am inside ");
}, 3000);
 console.log ("hello two');
};

const userthree = () => {
 console.log ("hello three");
};
userone ();
usertwo ();
userthree()
```

output: hello one
hello two
hello three
Hello I am inside

// promise ⇒ resolve / rejected
↓
it will wait till it gets rejected or resolve and then move forward

```
const one = () => {
  return "I am one";
};
const two = () => {
  return new Promise(resolve, reject) => {
  Let Timeout (() => {
    resolve("Credential Correct");
  }, 3000);
});
};
const three = () => {
  return "I am three";
};

const CallMe = async () => {
  let Valone = one();
  console.log(Valone);

  let Valtwo = await two();
  console.log(Valtwo);

  let Valthree = three();
  console.log(Valthree);
};
CallMe();
```

I am one

Promise {<pending>}

I am three

after everything is completed await will be executed

Credentials correct

output : I am one
Promise { <pending> }
I am three

* Async function returns a promise. This promise state can be either resolved or rejected

* Await suspends the called function execution until the promise returns a result for that execution

Promise?: Kasam khata hai ki vo badme ye kaam complete karega

```
#console.log ("This is tutorial 43");  ①

async function harry () {
    console.log ('Inside harry function');  ③.①
    const response = await fetch ('https://api.github.com/users');
    console.log ('before response');
    const users = await response.json ();
    console.log ('users resolved');
    return users;
}

console.log ("Before calling harry");  ②
let a = harry ();  ③
console.log ("After calling harry");  ④
console.log (a);
a.then ((data) => console.log (data));
console.log ("last line of this js file");  ⑤
```

here it will wait and rest of the code will be executed until it fetches the api

will convert api to json format

when we reach back here all the work has already been executed

= here the promise is solved and then (.then) will proceed

→ promise still pending

will be executed after promise is completed

output: This is tutorial 43
Before calling harry
Inside harry function
After calling harry
Promise {<pending>}
Last line of this js File
before response
users resolved
(30) [{...}, {...} ..... {...}]