Closures

↳ setTimeout setInterval
↪

These two functions are not by default given to us by JS.

Then how are we able to use it??

→ setTimeout

↓

It is a function that helps to execute some task once after a certain timer.

→ setInterval

It is a function that helps us to execute some task again & again after a given interval.

setTimeout (taskCallback , time in millisecond);

returns a unique id. → clearTimeout(id)

ex →
```
setTimeout ( function () {
    console.log ("hello");
}, 1000);
```

id → unique id.

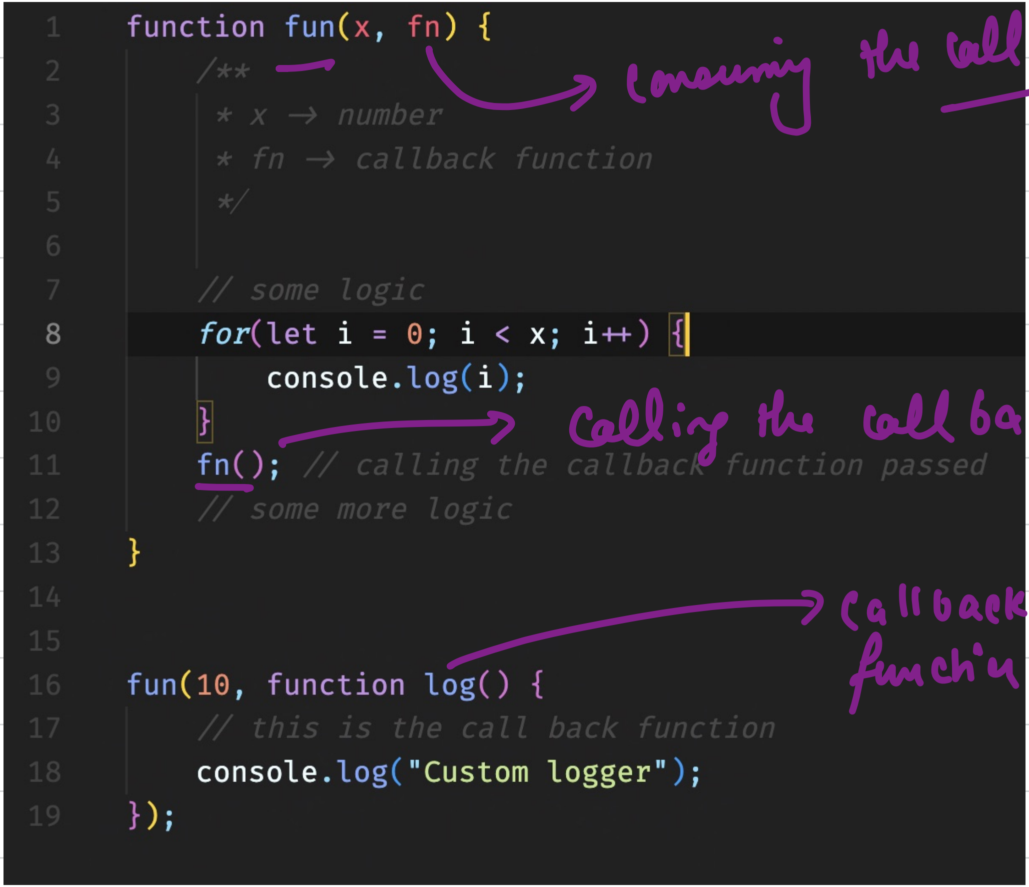
id = SetInterval (taskCallback , interval in ms);

id → clearInterval (id)

Callbacks

→ A callback is a function that is passed to another function.

$$f(\underbrace{g(x)}_{\text{call back}})$$



```
function fun(x, fn) {
    /**
     * x -> number
     * fn -> callback function
     */

    // some logic
    for(let i = 0; i < x; i++) {
        console.log(i);
    }
    fn(); // calling the callback function passed
    // some more logic
}


fun(10, function log() {
    // this is the call back function
    console.log("Custom logger");
});
```

consuming the call back

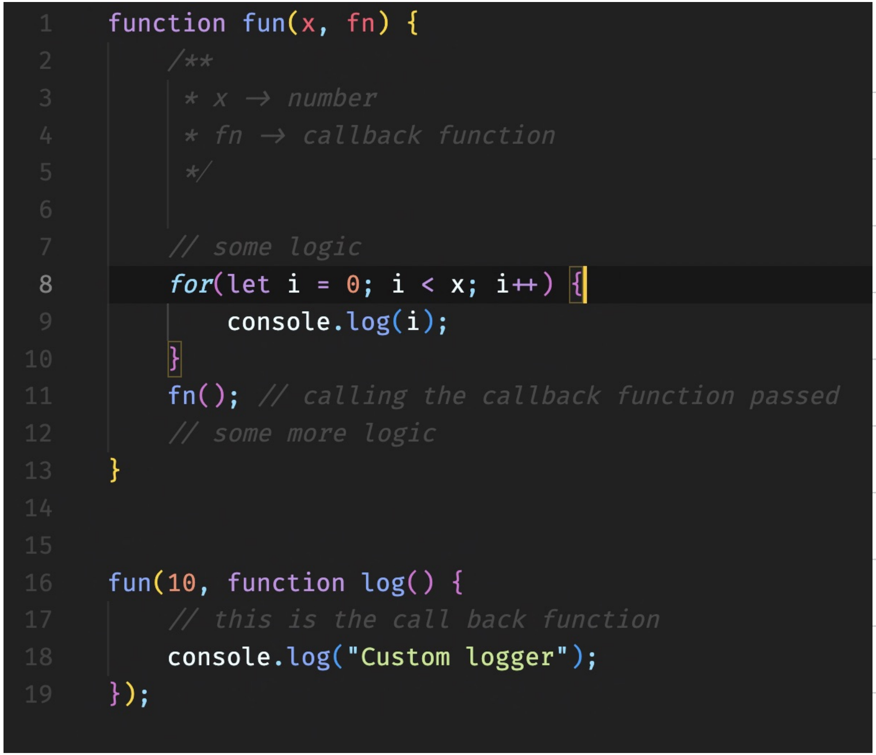
Calling the call back passed.

callback function

arr.map (fn) → call back func^n

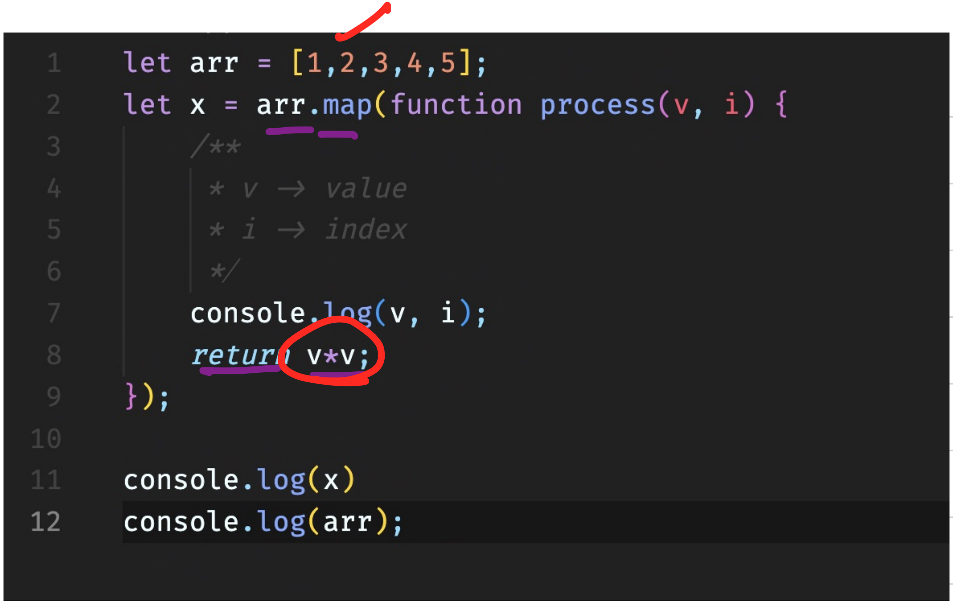
for all the elements of the given array, it passes the elements as an argument to the call back.

```
[1, 2, 3]
arr.map( function process (v, i) {

        // some task
})
```



```
function fun(x, fn) {
    /**
     * x → number
     * fn → callback function
     */

    // some logic
    for(let i = 0; i < x; i++) {
        console.log(i);
    }
    fn(); // calling the callback function passed
    // some more logic
}


fun(10, function log() {
    // this is the call back function
    console.log("Custom logger");
});
```

0
1
2
:
:
9
Custom logger

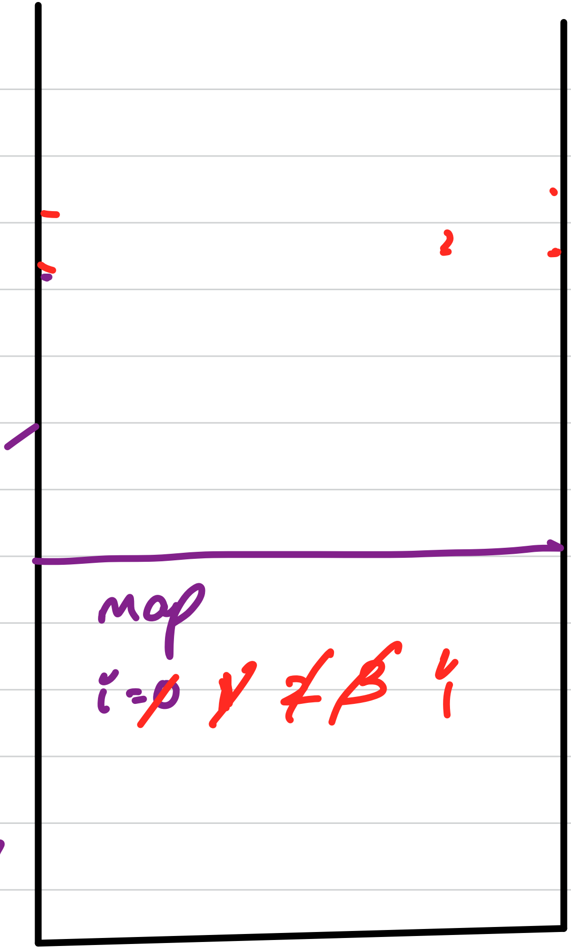


```
let arr = [1,2,3,4,5];
let x = arr.map(function process(v, i) {
    /**
     * v -> value
     * i -> index
     */
    console.log(v, i);
    return v*v;
});

console.log(x)
console.log(arr);
```

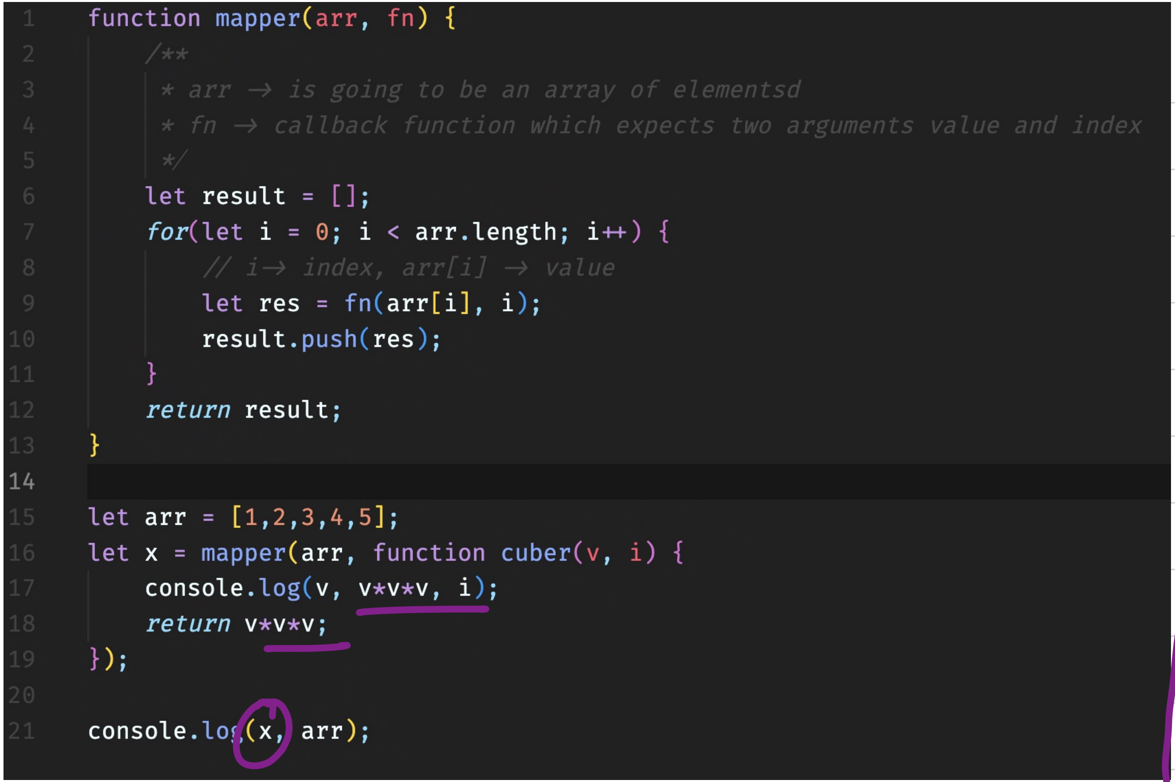
arr = [1, 2, 3, 4, 5]

x = [1, 4, 9, 16, 25

1 0
2 1
3 2
4 3



call stack

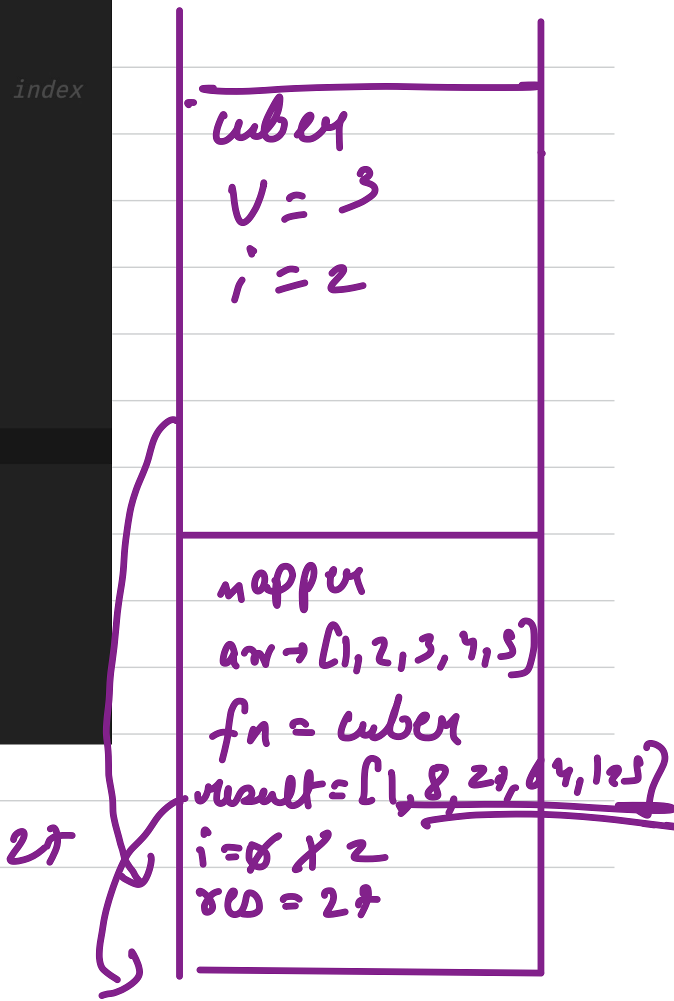


```
function mapper(arr, fn) {
    /**
     * arr -> is going to be an array of elementsd
     * fn -> callback function which expects two arguments value and index
     */
    let result = [];
    for(let i = 0; i < arr.length; i++) {
        // i-> index, arr[i] -> value
        let res = fn(arr[i], i);
        result.push(res);
    }
    return result;
}

let arr = [1,2,3,4,5];
let x = mapper(arr, function cuber(v, i) {
    console.log(v, v*v*v, i);
    return v*v*v;
});

console.log(x, arr);
```

1, 1, 0

2, 8, 1

3, 27, 2



cuber
v = 3
i = 2
mapper
arr → [1, 2, 3, 4, 5]
fn = cuber
result = [1, 8, 27, 64, 125]
i = ~~0~~ ~~1~~ 2
res = 27
27