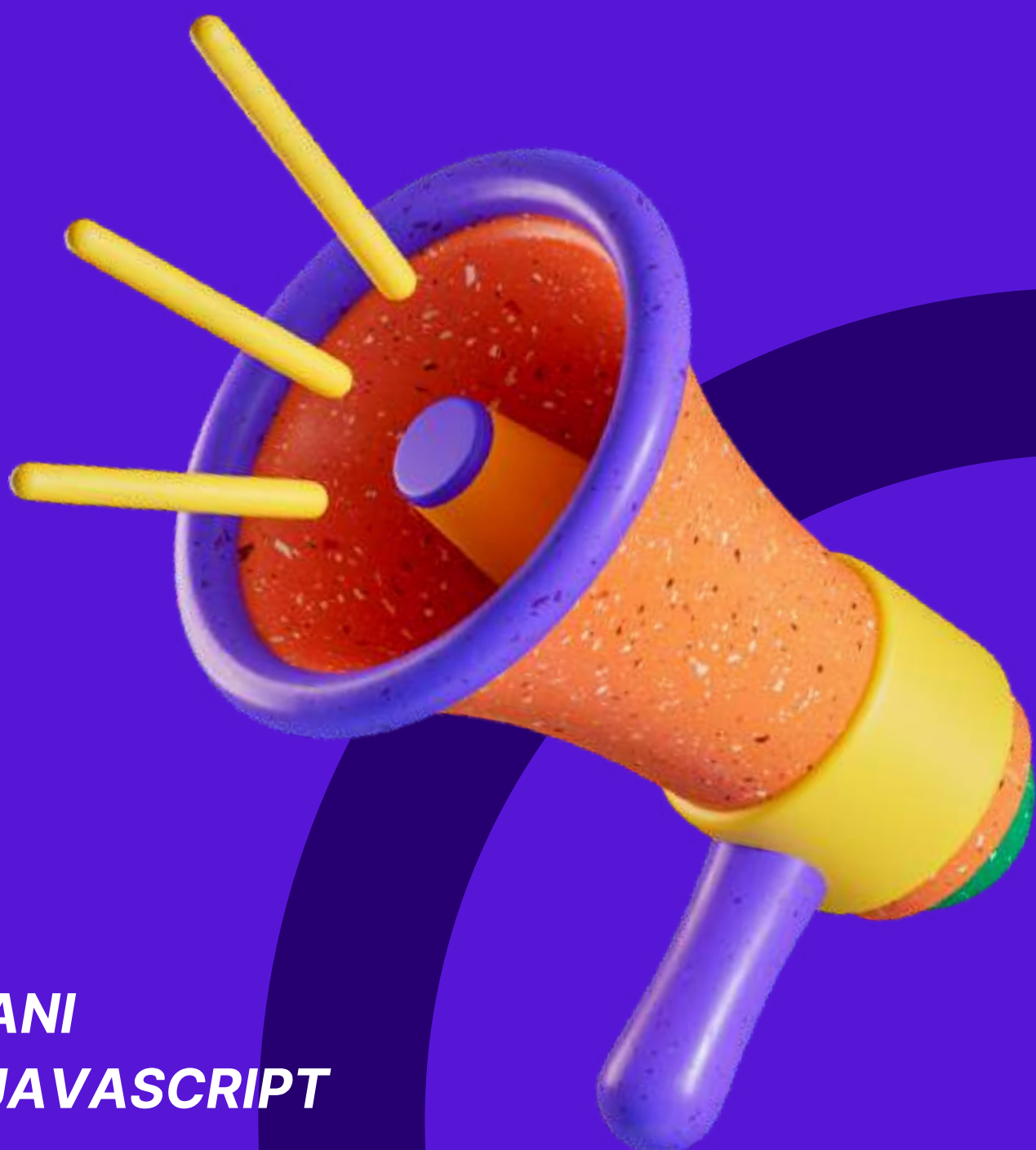
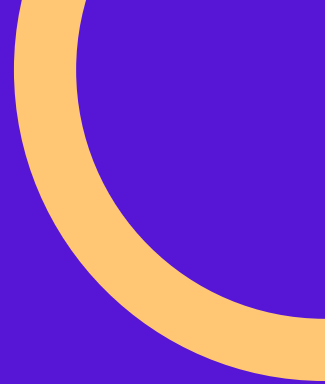


CALLBACKS IN JAVASCRIPT



RIZWAN RABBANI
SOFTWARE ENGINEER-JAVASCRIPT



**WHILE WE'RE IN
CALLBACKS, THERE'S
SOMETHING YOU
SHOULD KNOW ABOUT
JAVASCRIPT.**

**JAVASCRIPT IS SINGLE
THREADED.**

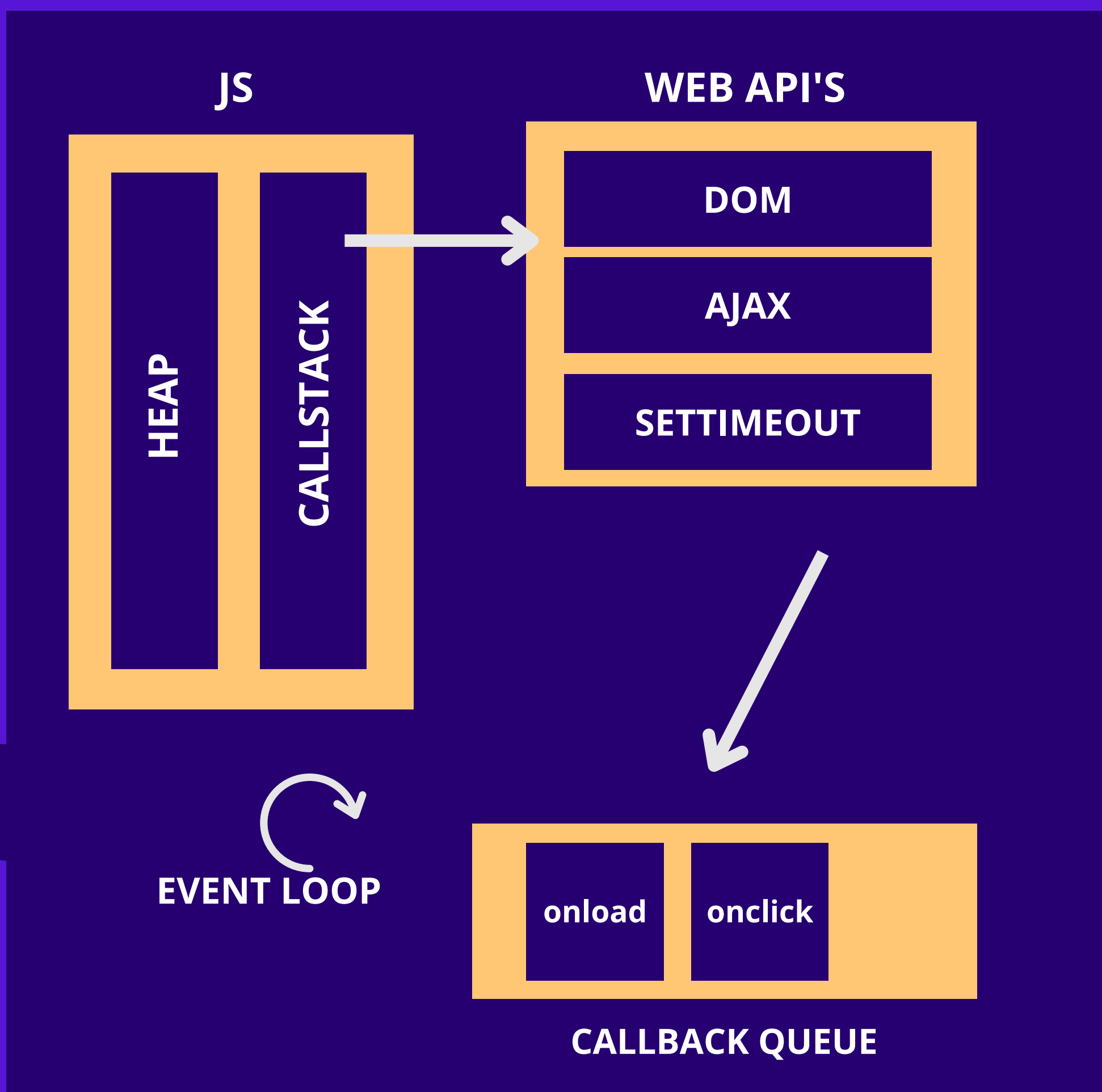


1

***THIS MEANS THAT IT HAS A SINGLE
CALL STACK AND MEMORY HEAP, IT
EXECUTES CODE SEQUENTIALLY, AND
IT MUST FINISH ONE PIECE OF CODE
BEFORE MOVING ON TO THE NEXT.***



I know you're curious. How will asynchronous tasks be accomplished if Javascript is synchronous and single threaded? The Event loop in Javascript handles all of this. Please do not be worried; we will explore the Event Loop in depth in the future.



WHAT IS CALLBACK?

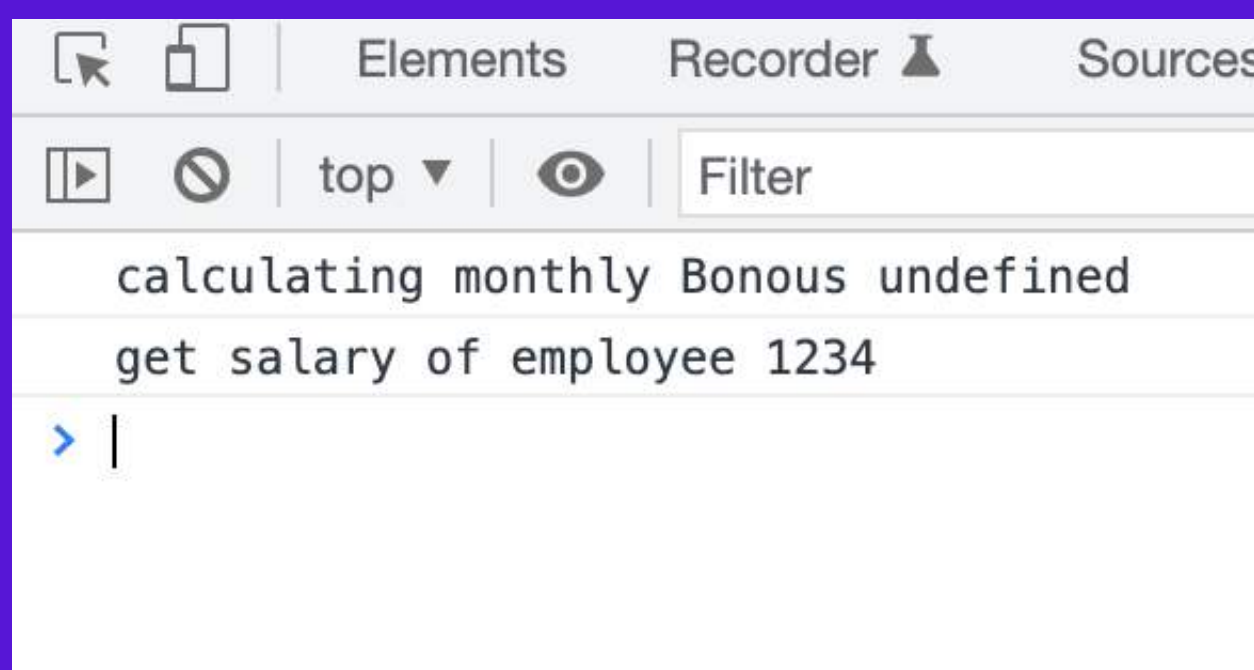
Functions are first-class residents in JavaScript. Therefore, you can provide a function as an argument to another function.


So, in javascript, when we pass a function as an argument into another function to be used later, this function is known as a callback function.

***Consider the following scenario:
Assume you want to fetch some user details from API (here we are using setTimeout to make it work like API) and then run some calculation on this data.***

```
7  function getSalary(employeeId){  
8      let salary;  
9      setTimeout(()=>{  
10         salary = "8000$"; // suppose  
11         console.log(`get salary of employee ${employeeId}`)  
12     },3000);  
13  
14     return salary;  
15  
16 }  
17  
18  
19 function monthlyBonousOnSalary(salary){  
20     console.log(`calculating monthly Bonous ${salary}`)  
21 }  
22  
23 let amount = getSalary(1234);  
24 monthlyBonousOnSalary(amount);
```

***If you run the following code:
You will receive the following result:***



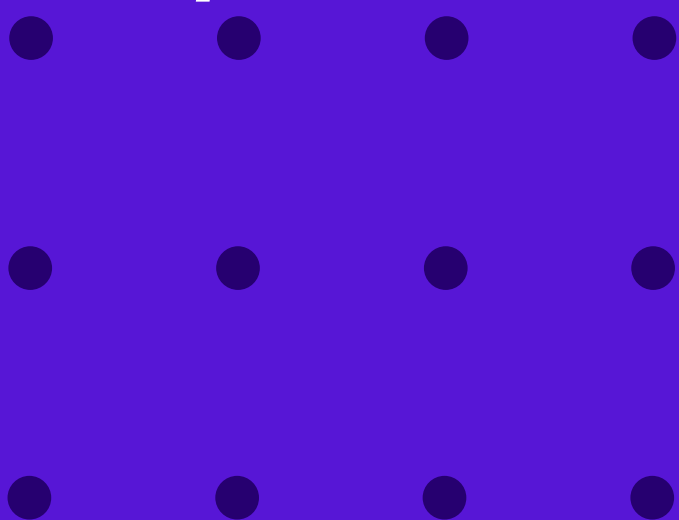


This is not what we expected because `monthlyBonusOnSalary` is called before `getSalary`.

The correct order should be:

- 1. `getSalary`***
- 2. `smonthlyBonousOnSalary`***

To fix this, pass the `monthlyBonousOnSalary()` function to the `getSalary()` function and execute the `monthlyBonousOnSalary()` function inside the `getSalary()` function once the download is complete, as shown below:



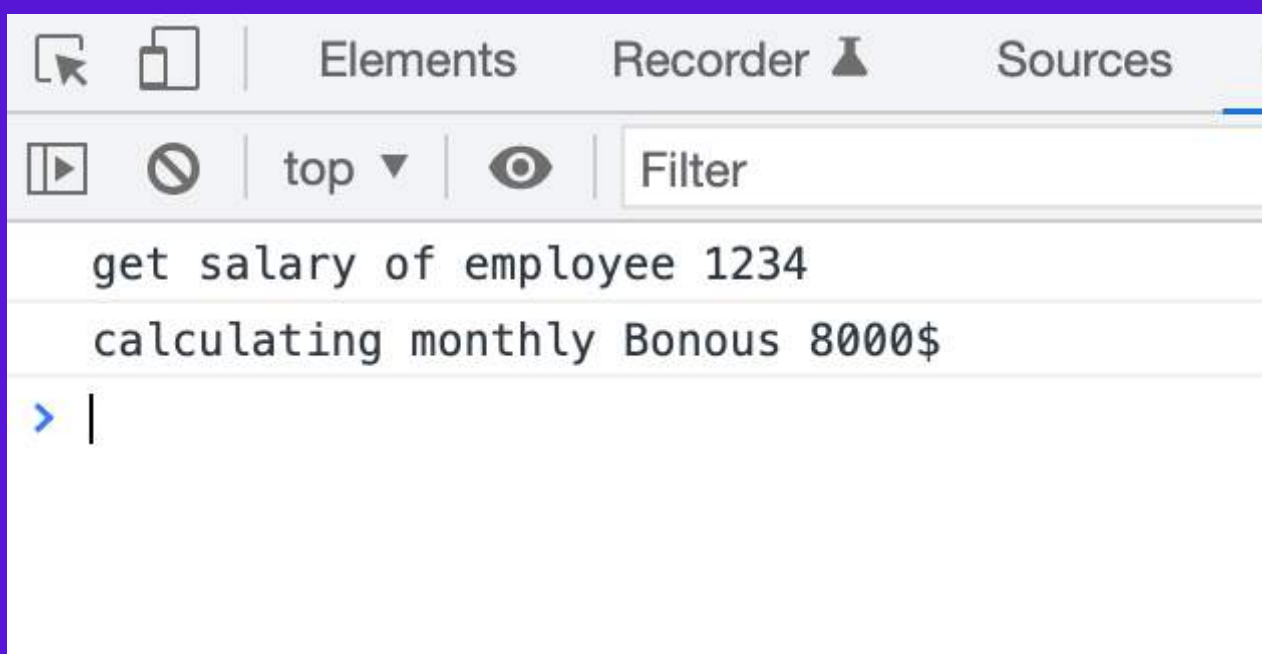
```
function getSalary(employeeId, callback){
  let salary;
  setTimeout(()=>{
    salary = "8000$"; // suppose
    console.log(`get salary of employee ${employeeId}`)
    callback([salary])
  }, 3000);

  return salary;
}

function monthlyBonousOnSalary(salary){
  console.log(`calculating monthly Bonous ${salary}`)
}

getSalary(1234, monthlyBonousOnSalary);
```

Now it works as expected.
In this case, monthlyBonusOnSalary() is a callback that is passed into a function.



WAS THIS HELPFUL TO YOU?

Be sure to save
this post for later
reading!



RIZWAN RABBANI
SOFTWARE ENGINEER-JAVASCRIPT

