Callbacks and their problems

Created	@May 6, 2023 11:03 PM
Class	
Type	
Materials	
Reviewed	☐

Callbacks and their problems

Callbacks the the arguments that we pass into a Higher order function. In legacy code of JS you will find heavy usage of callbacks, and even still we rely for a lot of things on callbacks. But they have some disadvantages as well.

- Callback Hell
- Inversion of control

Callback hell

It is technically a readability problem. Readability problem means that it makes the code less readable. Why ? Callback hell refers to a case, where we have a callback inside a callback, and so on which makes kind of like a nested callback like structure.

```
fetchCustom("www.google.com", function downloadCallback(response) {
    console.log("Downloaded response is", response);
    writeFile(response, function writeCallback(filenameResponse) {
        console.log("new file written is", filenameResponse);
        uploadFile(filenameResponse, "www.drive.google.com", function uploadCallback(uploadResponse) {
            console.log("Successfully uploaded", uploadResponse);
        });
    });
});
```

So in the above piece of code, we have a function `fetchCustom` which has a callback as an argument named as `downloadCallback` which has another function call to `writeFile` which has a callback as an argument named as `writeCallback` and then this `writeCallback` also had=s a function call which takes another callback `uploadCallback` as an argument. This kind of nested callback inside a callback is called Callback Hell.

So you can see it makes kind of a pyramid like structure which makes the whole code very less readable, because as a programmer you will experience that a line by line execution is something that is more readable and understandable.

But this is not the biggest problem of callback as it is just about having a less clean code. It doesn't maintain good code hygiene but still works for most of the cases, but due to less readability programmer is prone to making mistakes or may be not fully understand what their code does and why it does so.

Inversion Of Control

This can be considered as a far more problematic scenario. Most of the people who learn JS, mainly thinks that Callback hell is the only problem with callbacks, but that's not true, The problem of inversion of control can lead to significantly bugger bugs and time consuming debugging effort.

What is inversion of control ?

So currently if there is a higher order function, you call the function and pass your callback in the higher order function.

```
function fun (cb) {
  cb();
}
fun(() => {console.log("i am a callback");});
```

Here fun is a Higher order function and `cb` is the callback which we pass as an argument to the function fun while calling it.

Now how the callback should be handled i.e. how it should be called, how many times it should be called, whether it should be even call or not even once, all of this decisions is gonna be taken by the Higher order function.

If we want to execute the callback only once, but by mistake function fun executes it thrice then what ?

```
function fun (cb) {
  cb();cb();cb();
}
fun(() => {console.log("i am a callback");});
```

You might feel but we have implemented the function fun, so we will take care of it right ?

But the problem is we use a lot of Higher order functions, some which are implement in JS for

example: array.map or array.filter, some are also given from runtime env, for example: setTimeout, setInterval etc. and sometimes we will use some third party implementations as well. For example let's say you are going to integrate a payment gateway (like Juspay or Razorpay), and the payment gateway gives you a function where they are expecting you to pass a callback. Let's say they give you a function `isValid` which checks that whether the users credit card is valid or not and then in this `isValid` function we pass a callback which we assume will be executed once the Credit card is detected valid, and let's say that in this callback we will write the code to deduct some amount from the users card.

Now how are we so sure that Razorpay's function is not calling our callback more than once? Or is it even calling it ? And don't we know ? Because we didn't implement it !! Somebody else did. Now you might think, we will read the code, before using it, how about that ? let's say when you were integrating this code, `isValid` function was correct but after sometime, a bug got introduced due to which they started by mistake calling your callback thrice. So are we going to daily read this code to ensure this should not happen ? No.

So next day, if your customer tries to buy something on your app, they will be charges three times.

And why all of this is happening ? Because we gave the control of our own implementation of deducting money from the users card, to the Higher order function `isValid` provided by someone else.

This is where we transfer the control of our callback to some other function, which might not be able to handle things properly.

This is called inversion of control.

How do we solve the problem of callbacks ?

Both the problems i.e. inversion of control and callback hell can be solved using `Promises`.