@abdulbarikhatri.dev

callback vs promises vs async/await

Part 01

.js

Contents

.js

Callback In Javascript

A callback is a function. It is used to perform tasks after some operation is completed.

You might have already seen callbacks being used before, let me show you

```
setTimeout(() => {
  console.log("Hello Callback🚀 !")
 }, 100)
```

WHAT!!! setTimeout IS CALLBACK

.js

No, setTimeout is not a callback!!

The function that is passed as an argument to setTimeout is a callback, which is called after 100 ms

Another Example

```
function setToken(token) {
  localStorage.setItem('$token', token)
}

login(setToken)
```

setToken is a callback function that is passed in **login**

Login will call it after a while. We do not know when, but it's the login's responsibility to call it.

.js

@abdulbarikhatri.dev

Callback HELL

Callback hell is a situation where we have multiple nested callbacks which makes code unreadable and difficult to maintain

```
setTimeout(() => {
    console.log('1')
    setTimeout(() => {
        console.log('2')
        setTimeout(() => {
            console.log('3')
        }, 100)
    }, 100)
}, 100)
```

WHEN WILL THE CONSOLE PRINT 3???

It’s hard to understand due to callback hell.

.js

Promises

So, promises solve the problem of callback hell. Promises are Object that “promises” us to return some value or reject depending upon the result.

How to create a promise Object?

It’s EASY!!!

```
let a = 20
const myPromise = new Promise((resolve, reject) => {
    if (a <= 20) {
      resolve('a is less than or equal to 20')
    } else {
      reject('a is greater than 20')
    }
})
```

You need to pass a callback function while creating promises, can’t avoid callback😔 but can avoid callback hell 👼.

.js

But how to use this promise??

You can get the resolved value and reject value that is passed in the resolve and reject function using **.then** and **.catch**

```
myPromise
  .then(value => {
    console.log(value)
  }).catch(error => {
    console.log(error)
  })
```

.then receives a callback that has the resolved value as first argument

.catch receives a callback that has the rejected value as first argument

.js

So, promises are just objects that can be used to resolve (i.e return a value) or reject based on the operation or functionality

@abdulbarikhatri.dev

Promise Chaining

You can chain multiple promises to avoid callback hell.

```
myPromise
  .then(first => {
    return anotherPromise
  })
  .then(second => {
    return lastPromise
  })
  .then(third => {
    console.log(third)
  })
  .catch(error => {
    console.log(error)
  })
```

So, you can return a promise from .then and after that can chain multiple .then callbacks.

ProTip: You can chain as many .then as you want. Because .then itself returns a promise😎

.js

@abdulbarikhatri.dev

Guess the Output??

```
// Can you predict the output of this code
// assume a promise by the name of myPromise exists and
// it always resolves

myPromise
   .then(first => {
     console.log(first)
    })
   .then(second => {
     console.log(second)
   })
```

Comment your output below !!!

.js

@abdulbarikhatri.dev

SAVE THIS POST IF YOU FIND IT USEFUL 🔖

.js

@abdulbarikhatri.dev

Follow for more Content!!

Like and Repost 😇

.js