Async JS Exercises

Agenda

This is mostly a **revision** class. We have used callbacks and promises already in levelZero. In this class we will understand some terminologies associated with it. There aren't new concepts.

We will also see async/await but that's just syntax over promises.

how to do this?

COPY

```
watchTheWholeSessionNow()
  .then(tryAgainYourself)
  .then(discussWithYourTeam)
  .then(() => {
    if (conceptsStillNotClear === true) {
      askInDoubtClearingSessions()
    }
  })
```

callbacks

ex1: where have you seen callbacks?

challenge

We have seen callbacks in levelZero. Can you tell us where?

understanding

We have used callbacks all along in vanillaJS and in React too. However, it's time to understand the other side of this.

ex2: write your own function which takes a callback

challenge

Write a function `strLength()` which takes

1. Your name
2. A function which it will call with the length of your name

Now, call this function with the two parameters.

1. Your name. This is simple.
2. A function. This function will get the length of the string. Use that length to print a message: `OMG! my name is X char long!`

understanding

When you're designing your API. A callback is nothing but a function that the user of your API will give you. In return, you give your user a **contract** saying that you'll call her function with X data.

Understand that this data can be calculated ***synchronously or in an asynchronous*** manner. It doesn't matter as long as you give that data.

live ex3: function which takes two callbacks

challenge

Write a function `willThanosKillMe()` . This function will take three parameters

1. Your name
2. Function to call if Thanos doesn't kill you.
3. Function to call if Thanos does kill you.

This function will call success callback if your name has even characters. You won't die.

But if your name has odd number of characters then you're going to die. Sorry! :(

Now, call this function with the given parameters. The success function should console a happy message: `Yay! I am alive!` and the failure function should console your will: `Give my bose speakers and apple headphones to my sister`

P.S. Sorry for the grim example. ☹️

understanding

Say in your function you were calling a server on Titan to see whether Thanos is going to kill your or not. Obviously that network call would take time. Making the entire program wait for one network call wouldn't look good na! Hence, the callbacks.

Doubt time. Were you able to do this?

ex4: use setTimeOut()

challenge

- Learn how to use `setTimeout()`
- Now, write a function that takes a message and a delay and print that message after the delay.

understanding

`setTimeout()` places the callback function on the event queue after the given time.

remember

- It doesn't guarantee time
- `setTimeout(fn, 0)` is used to do calculation when callstack is empty. Therefore not blocking the render.

live ex5: predict outputs

Put your answers in chat for every question.

1.

COPY

```
setTimeout(() => console.log('A'), 0)
setTimeout(() => console.log('C'), 0)
setTimeout(() => console.log('B'), 0)
```

2.

COPY

```
setTimeout(() => console.log('A'), 1000)
setTimeout(() => console.log('B'), 400)
setTimeout(() => console.log('C'), 1300)
```

3.

COPY

```
console.log('A')
setTimeout(() => console.log('B'), 0)
```

```
  console.log('C')
```

homework

If you didn't predict this right, discuss that particular number with your group after the class. This means your understanding from yesterdays's class is not solid yet.

challenge

- learn how setInterval works
- **6.1** write a function which takes a message and time. The function should print that message every X interval.
- **6.2** Write a function that takes a number. Then print a countdown from that number to 0. At zero print "Bang Bang!" ← The important question is sometimes asked in FAANG interviews as well.

h/w ex7: onClick in React

This is mostly a revision of previous sessions. Mixing vanillaJS concepts with ReactJS for 7.1 and watch **https://youtu.be/Icr3pGbz3iE?t=5848** if you're unable to do 7.2.

challenge

- **7.1** Create a button in React and print the event
  - Can you print the button text from this event?
- **7.2** Create a list in React. Use array of objects. Use map to render the list
  - On every list there should be an onClick handler. Clicking on this should print the details of the object.

promises

ex8: why promises?

challenge

In an interview, someone asks you to explain why `promises()` are better and why you should prefer them over `callbacks()` . What will be your answer?

COPY

```
getA(getB(getC))

// Easy to follow code, avoid callback hell

getA(){
        doX();
        doY()
        getB(data => {
                        doOne();
                        doTwo();
                        getC(cData => {
                                        doEleven()
                                        doTwelve();
                                }
                        }
}
```

ex9: states of promise

challenge

In an interview,

- explain how promise work
- explain the different states of promise

    - pending, fulfill, reject.
- syntax of promise consumption

```
callAPromise().then(successHandler).catch(rejectHandler)
```

COPY

h/w ex10: understand promise constructor

Tanay has made a fakeFetch implementation for the class.

Understand how to make your own promises and make your own version of `fakeFetch()` . Discuss this with your team.

```
function fakeFetch(msg, shouldReject) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (shouldReject) {
        reject(`error from server: ${msg}`)
      }
      resolve(`from server: ${msg}`)
    }, 3000)
  })
}
```

COPY

live ex11: print data on success

challenge

use the `fakeFetch()` to get data and show on success.

live ex12: print data on success, print error on failure

challenge

Call `fakeFetch(msg, true)` to get a rejected promise. Handle the error with the error handler. Show a message using `console.error` for errors.

live ex13: chaining

challenge

Create a function `getServerResponseLength(msg)` This function will use `fakeFetch()` internally with the message and return the length of the response received by the server.

Note: Instead of returning the response from the server this **should return the length**.

Hint: It will return in a promise.

understanding

Whatever you return from `.then()` also becomes a promise. And this is how the chain propagates.

live ex14: waterfall data

challenge

Write a function `syncCallsToServer(msg1, msg2)` which will take two messages and call `fakeFetch()` with the second message only when the first message has returned from the server.

understanding

Think of this as a situation where you need to get `userID` from the server to get the `order` data for the user. You would need userID as part of the query to get order. How would you do that?

async-await

Nothing but promises which look better.

Two best practices to keep in mind:

1. Use async-await as much as possible.
2. Always take care of the error handling as well.

COPY

```
async function printDataFromServer() {
  const serverData = await anyPromiseWhichWillReturnData() // .then(data => { skskski
  console.log(serverData);
}

// notice that function need an `async` keyword.

// Doing this in es6 arrow function would be

const printDataFromServer = async () => {
        try {
                const serverData = await anyPromiseWhichWillReturnData();
          console.log(serverData);
        } catch (err) {
         console.error(err)
        }
}

/**
Note: In arrow the async keyword is used before the ().
While in normal functions, it is used before the `function` keyword itself.
```

live ex15: use async-await with fakeFetch

challenge

Given the syntax above. Call `fakeFetch()` with some msg and use await to get the data and then print it.

live ex16: use async-await to do waterfall

challenge

Do question number 14 with async-await this time.

h/w convert all promise related questions to async await

- Do this for all the exercises above.
- Take care of error handling as well.
- Read about it here **https://javascript.info/async-await**

h/w important: parallel calls in async-await

We did the synchronous `fakeFetch()` fall. How would you do two parallel calls without blocking each other?

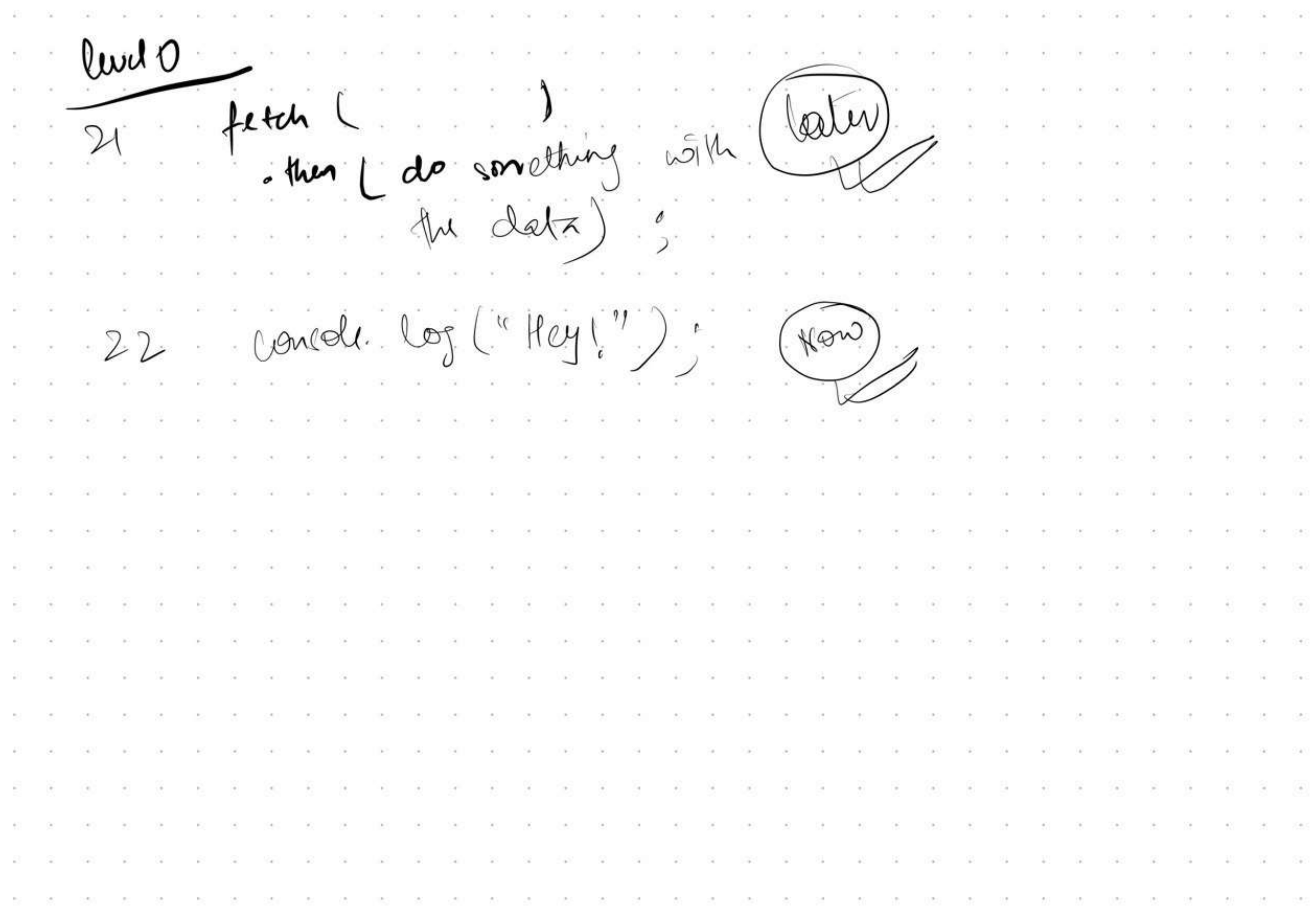
How to catch different errors in async await? In promises as well.

live homeworks

Read more on these topics

1. What is the return value of setTimeout?
2. Why promises are better than callbacks?
3. Microtask queue vs callback queue
4. Promise chaining in JS

Callstack Understanding

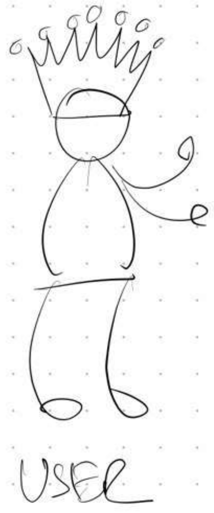


level 0

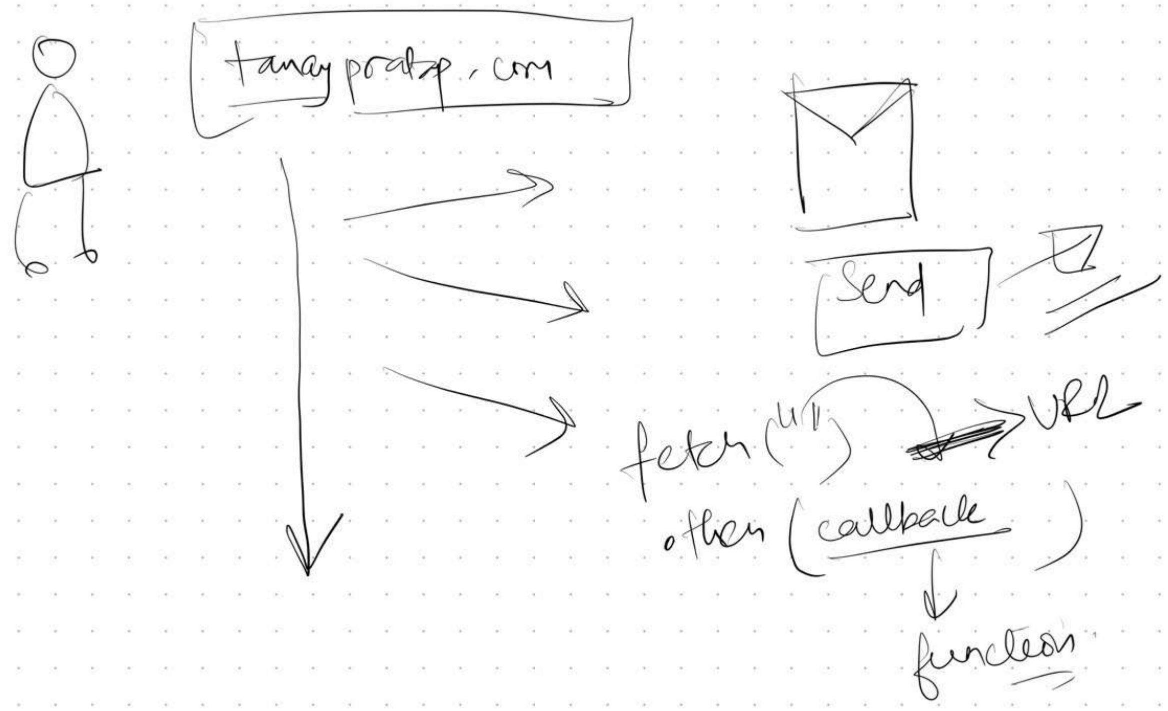
```
21    fetch (           )
        .then ( do something with
                the data) ;          later

22    console.log("Hey!");           now
```

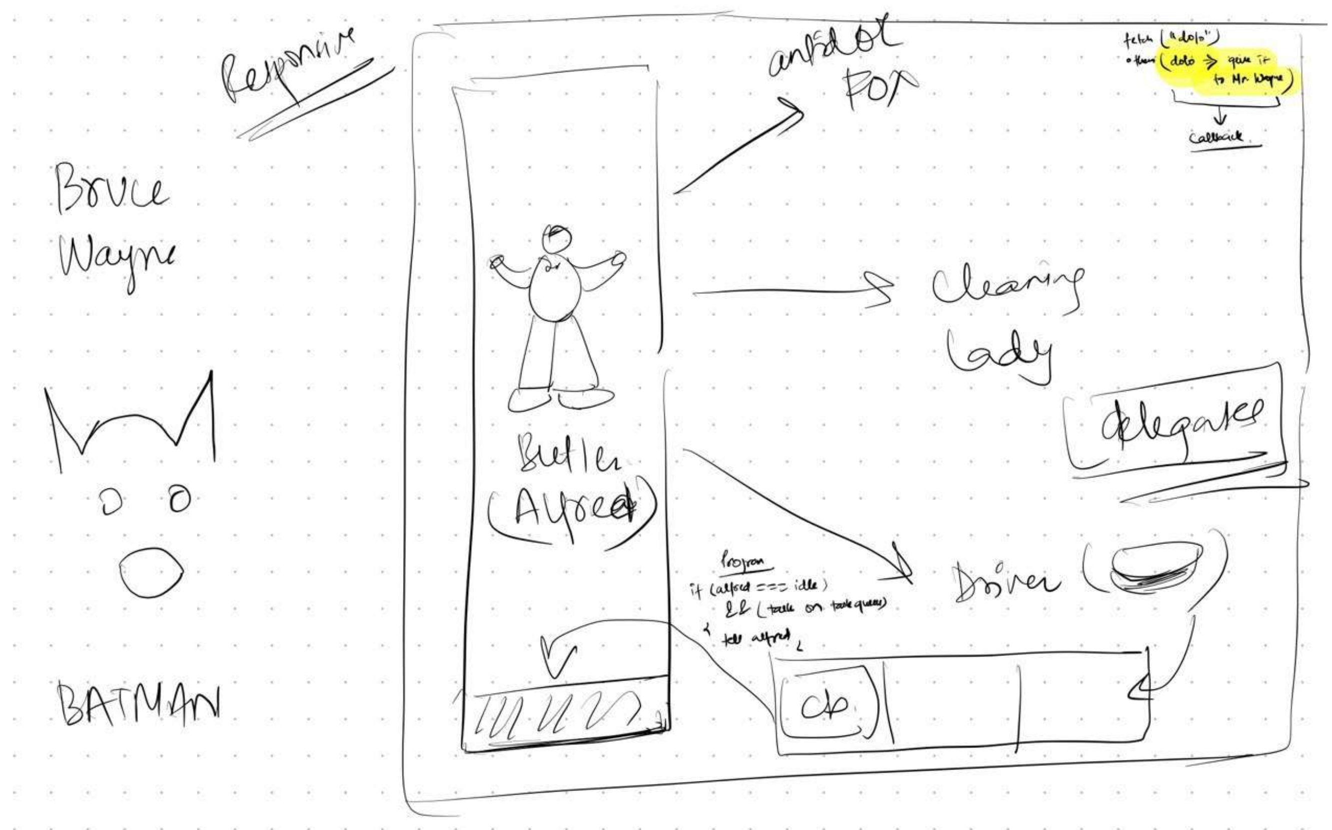
Why do we need Async?



You can't make the user wait/unresponsive.

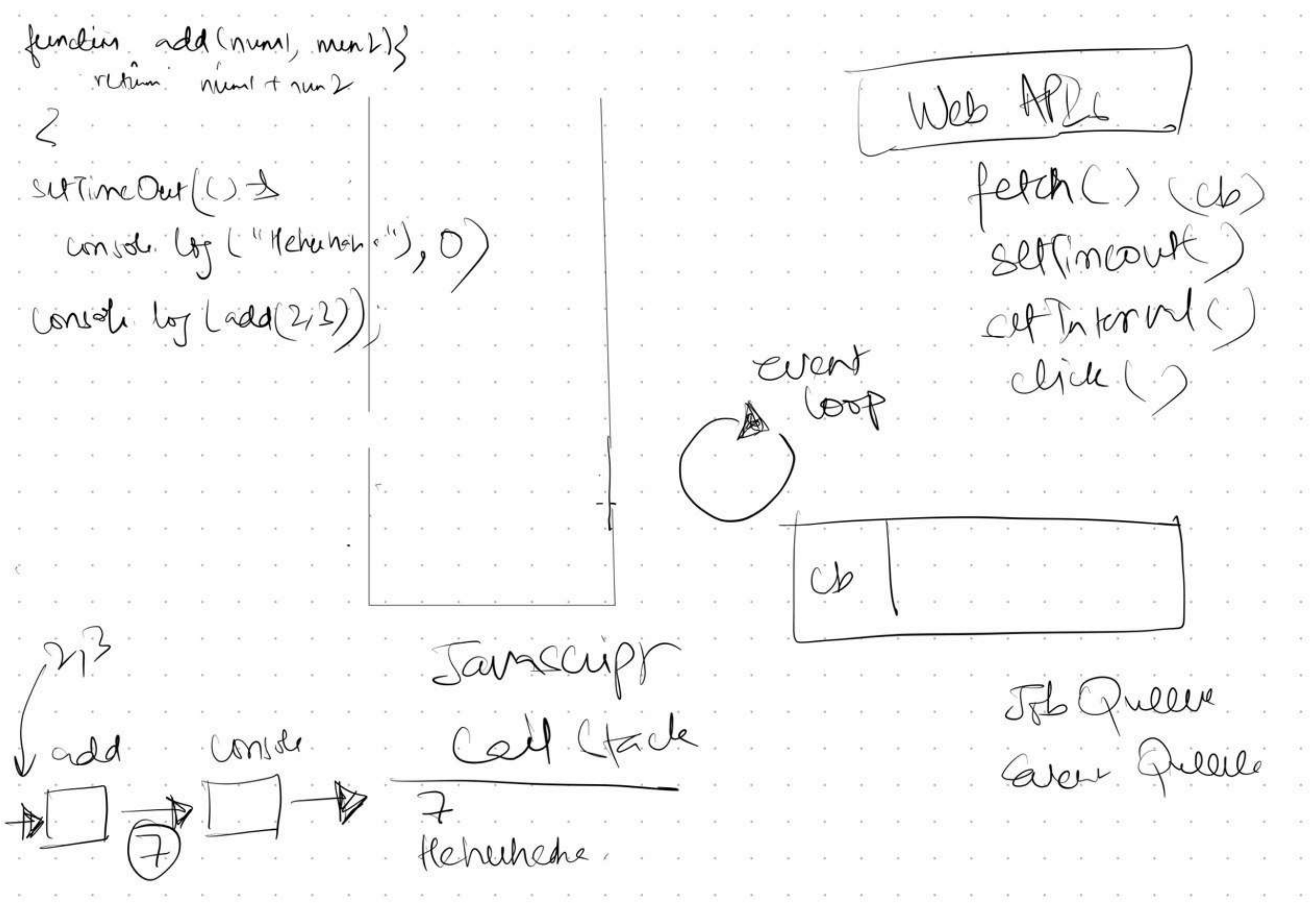


tanaypratap.com
Send
fetch("")
.then(callback)
URL
function



```
fetch ("data")
.then (data => give it
            to Mr Wayne)
```

```
Program
if (alfred === idle)
   && (task on task queue)
{
   tell alfred
}
```



Previous

‹ Async JS

Next

Vikram: Talk your CSS troubles ›