Async Javascript

Hello everyone👋🏻
Let’s get started with Asynchronous Javascript 😎
(Press **Ctrl+Shift+L** on Windows to view this document in Dark Mode😎)

This doc has coding related stuff mostly explaination part is little messed up

Table of Contents:

Exercise solutions

```
const strLength=(name,cb)=>{ const lengthOfName=name.length;
cb(lengthOfName); } const printName=(nameLength)=>{ console.log (`OMG! My
Name is ${nameLength} long`) } strLength("Ishaan",printName)
```

We are passing the `printName` function as a callback function to `strLength` .
A callback is nothing but a function that the user of your API will give you.

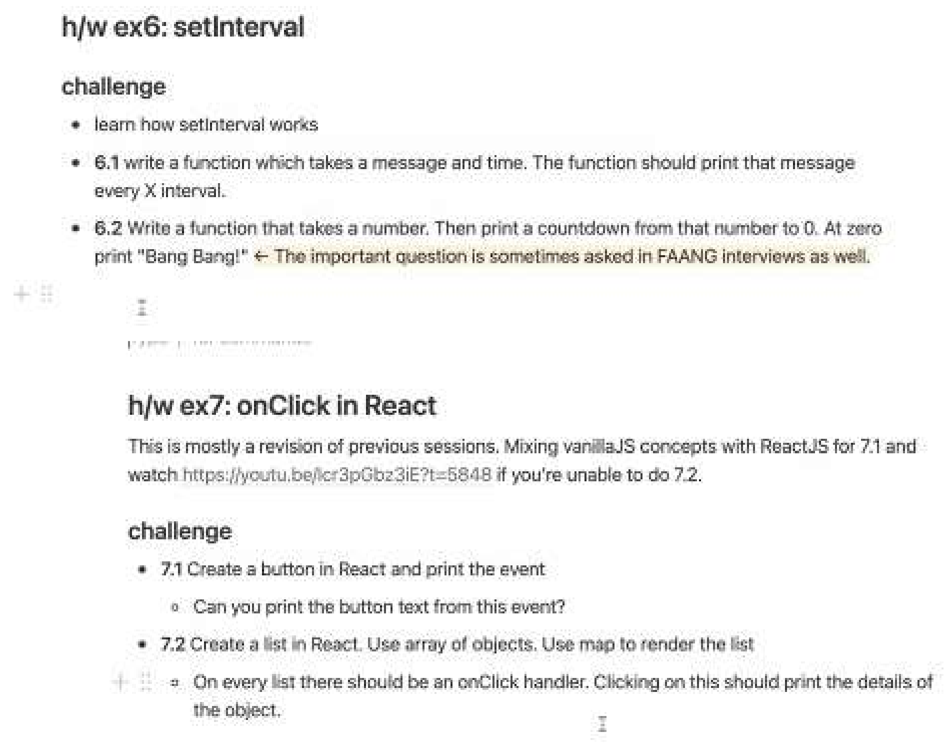
MDN Doc of addEventListener referring to callBack.

```
const willThanosKillMe=(name,iLiveCb,iDieCb)=>{ if(name.length%2===0){
iLiveCb(); }else{ iDieCb(); } } const iLiveCb=()=>{console.log("Yayy I am
Alive")} willThanosKillMe("Tanay",iLiveCb,()=>{console.log("Give my
headphones")})
```

```
setTimeout(callbackFunction,timer); //syntax for setTimeout const
printAfterDelay=(msg,delay)=>{ setTimeout(()=>{ console.log(msg) },delay) }
printAfterDelay("tanay",5000) //returns the name after 5 seconds and //a
value related to Timer Id discussed below
```

The returned timeoutID is a positive integer value which identifies the timer created by the call to setTimeout(). This value can be passed to clearTimeout() to cancel the timeout.

Homework:



h/w ex6: setInterval

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/lcr3pGbz3iE?t=5848 if you're unable to do 7.2.

challenge

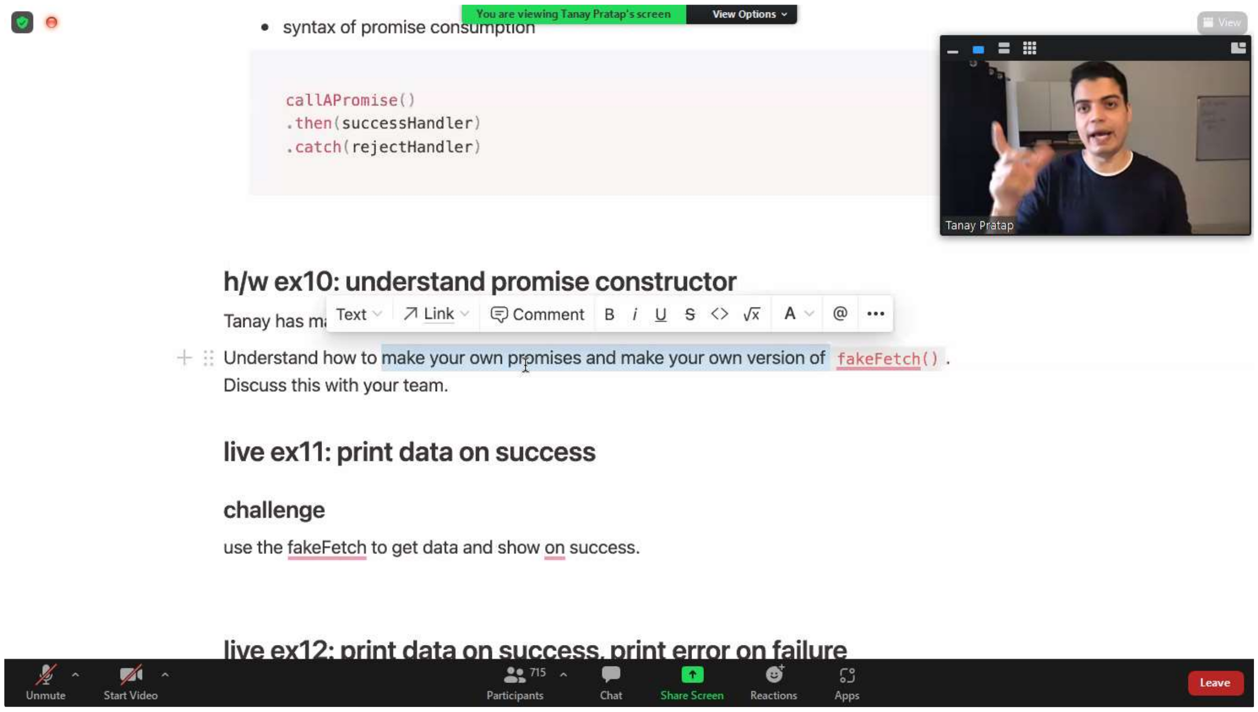
- 7.1 Create a button in React and print the event
  - Can you print the button text from this event?
- 7.2 Create a list in React. Use array of objects. Use map to render the list
  - On every list there should be an onClick handler. Clicking on this should print the details of the object.

📌 Why Promises:

Read Blogs about why promises are better than callbacks.
You need to have an idea about how promises work and the different states of promises (fulfil,reject and pending).

```
//Promise syntax callAPromise() .then(successHandler) .catch(errorHandler)
```

Homework question(Important to do).



- syntax of promise consumption

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

h/w ex10: understand promise constructor

Tanay has m

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

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

```
function fakeFetch(msg, shouldReject) { return new Promise((resolve, reject)
=> { setTimeout(() => { if (shouldReject) { reject(`error from server:
${msg}`); } resolve(`from server: ${msg}`); }, 3000); }); } //show Data of
success fakeFetch("tanay is awesome").then(data=>console.log(data)) const
onSuccessHandler= data=>console.log(data) const errorHandler=
err=>console.error(err) //show Failure fakeFetch("tanay is awesome",true)
.then(onSuccessHandler) .catch(errorHandler)
```

Jake Archibald video on event loop

```
const getServerResponseLength=(msg)=> fakeFetch(msg).then(data=>data.length)
getServerResponseLength("tanay hahaha")
```

This returns to us a promise from fakeFetch and then from the result of that promise we were calculating the length of the response related to it. That's the formation of a promise Chain.

```
const syncCallsToServer=(msg1,msg2)=>fakeFetch(msg1) .then(
dataForMsg1=>fakeFetch(msg2)
.then(dataForMsg2=>console.log({dataForMsg1,dataForMsg2})) ) const
parallelCallsToServer=(msg1,msg2)=>{
fakeFetch(msg1).then(output1=>console.log({output1}))
fakeFetch(msg2).then(output2=>console.log({output2})) }
```

📌 Async-Await

2 Best Practices to Keep in Mind:
1) Use Async Await as much as possible.
2) Always take care of error handling (important for PR reviews in projects)

Use Async Await with Fake Fetch and try the above mentioned promises with Async Await.
Async is a part of Javascript not the web api.

```
const getData= async()=>{ const data = await fakeFetch("something"); //if
comparing to older terms which we learnt earlier here is where //you will run
the promise near await and // then the processing part which was done earlier
in then can be done below console.log(data); } const asyncOp = async (msg)=>{
try{ const response=await fakeFetch(msg); console.log(response)
}catch(mistake){ console.log(mistake) } } asyncOp("chilly") const
syncCallsToServer2= async (msg1,msg2) =>{ try { const resp1=await
fakeFetch(msg1); const resp2 = await fakeFetch(msg2);
console.log(resp1,resp2); }catch(error){ console.log(error) } }
syncCallsToServer2("chilly","flakes")
```

Homework: Try parallel Calls with async await.

How to catch different errors in async Await,promises?

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?