Time stamps

1 Setup

Client:

setup the react project (I had a setup copy)

npm install axios moment react-file-base64 redux-thunk

npm install @material-ui/core

axios – when I want data from external source(backend). It’s exactly like fetch but a bit better? (Discuss later)

server:

npm install body-parser cors express mongoose nodemon

cors – cross origin resource sharing. It allows to relax the security applied to an API (this is done by bypassing control allow origin header, which specify which origin can access the API).

"type": "module", in package.json allows us to use import keyword

Server side code Part 1:

Its crystal clear.

Break summary

Connect to DB in index.js

postMessag.js (models) – data part

routes & controller – its like different server pages.

Nothing new till now.

redux – all actions towards our backend will be done using redux

actions

reducers

redux store:

flow: data from form(client) to DB using redux?

Understanding Version 1:

I click on Submit from form, that calls createPost From actions (which sends the data to server). It also dispatches to the reducer that does what here?

router.post("/", createPost); activates when createPost from index.js (api is called)… that axois posts on it

54:27 to 55:02 – redux flow ig?.

Understanding Version 2:

From form createposts (actions) is called with data. Data is stored to DB & dispatch function to reducer is called. Reducer updates the data in its store

(This will get clearer in part 2 ig) & displays the data in UI (dynamic site)

I still don’t get why I needed a function that will return a function… he says we did it to make it async… but can’t we do it directly??

QUESTION: Why not directly in place of redux?

static vs dynamic:

my understading:

if I send a message

Dynamic: the user can see the changes in UI.

For static, I’ll have to refresh the page?

PART 2:

Svg background – for that blue background image

1st we put the post UI.

2 worked on update feature. Didn’t use redux. It was a lot of work (passed id from App.js to Form & Posts) in 17:35 to 27 part

Console.log () error vs error.message – 1st one is better as gives full detail

He also did some silly mistakes (typo) & solved it step by step. Felt good. Debugging is an imp skill.

3 Delete

Same process. 1st setup in server(route & controller) then in client using redux (as mentioned for create above)

Done!! This was really really fun! Excited for GSSOC

1:00:00 to 1:10:00 good practices stuff

Environment variable: need to protect the DB url n other imp keys.

Npm I dotevn

(for that .env to work)

1:11:00 deploying to the web

TODO: learn the basic terms like state, prop etc

Revise everything.

Phase2:

Adding authentication (this will work as template for my code).

Client:

npm install jwt-decode react-google-login

Server:

npm install bcryptjs jsonwebtoken

We r making the

Google auth – it was easy. Didn’t have to write a single code in server.

Custom auth:

1st did the client stuff work: getting form data & supplying to redux.

1:43:00 I want only real user to be able to do stuff

Redux flow for auth:

Auth component(dispatch) -> action(signin) -> api (get some data from database & return to me)

2:17:17 See how he debugs

Debug: console -> network -> terminal