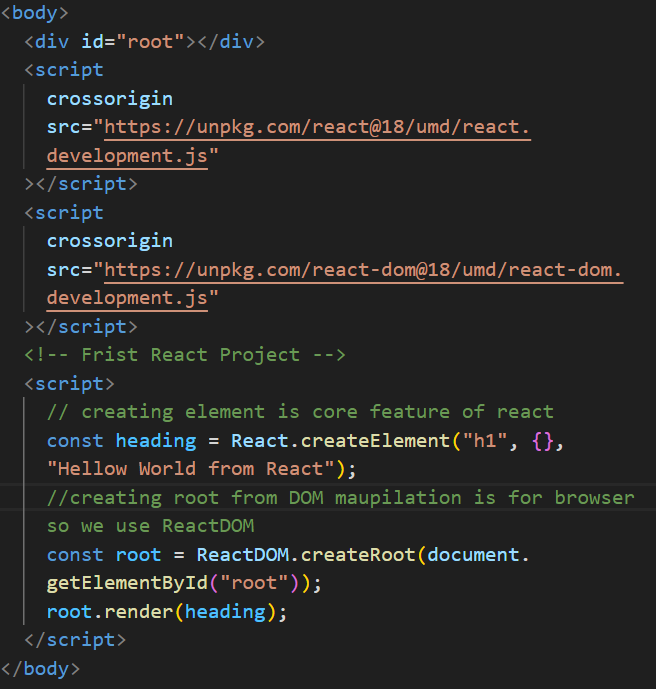
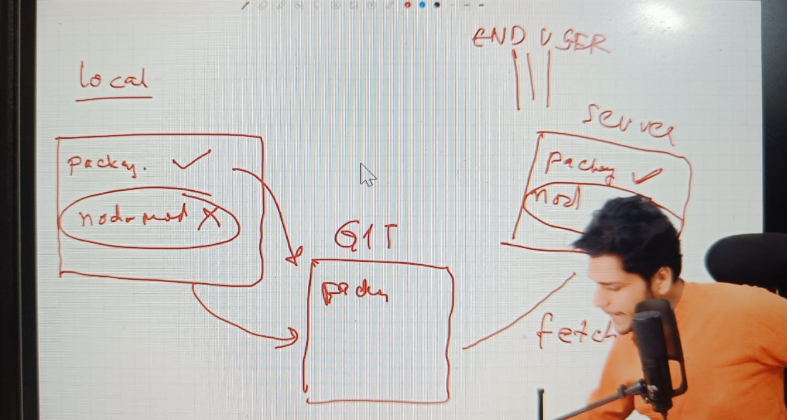
**Namaste - React**

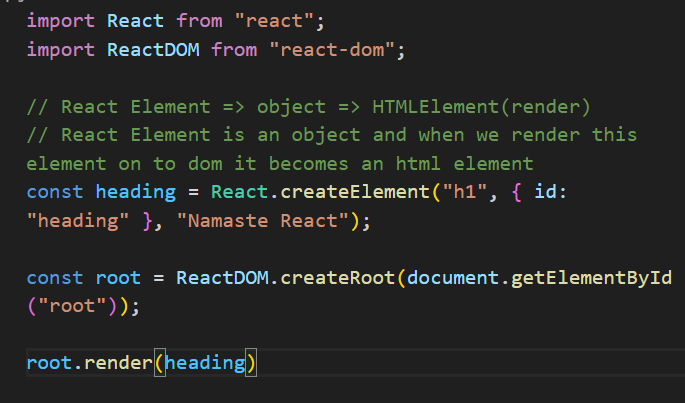
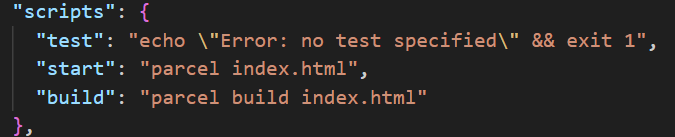
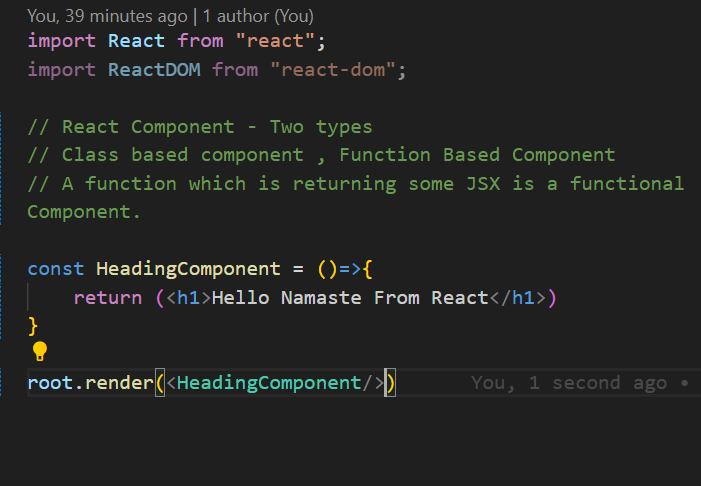
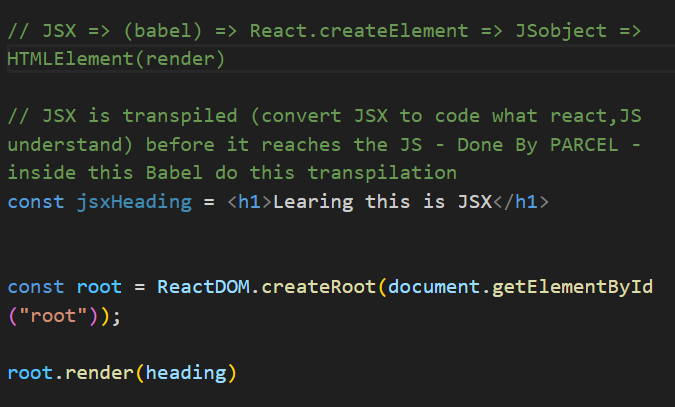
***Ep -01 || Inception***

* We can use react in our project using CDN - react script tag in html file , there are two file , one is react , 2nd is react-dom (used mainly for dom manipulation )
* React is just JS code , written by facebook eng
* 
* Here, heading is react element (not h1 tag) which is JS object and it have some props and other stuff. And root.render is responsible to convert it to h1 tag and put it to dom.
* This will became more complex and complex later on, if dom tree is big , that’s why JSX is introduced (JavaScript XML)
* React is a library , it can work on small portions of app.
* Ques - what Is CDN , what is crossorigin

***Ep -02 || Igniting our App***

* Npm full is not a Node Package Manager. (NPM don’t have full form) , NPM manages packages but it does not stand for node package manager , all the libraries and utility we need comes from npm and NPM manages that in our project.
* Package.json is configuration for npm. Why we need it - NPM manages all package that we install , sometimes this packages will also known as dependency and npm take care of these dependency like what its version and all in package.json.
* Most important package is bundler (bundle our app , compress) , vite , parcel , webpack are some example.
* There are two types of dependency – dev dependency , normal dependency(use in production also) [npm install -D parcel] -D is dev dependency.
* Package-lock.json – keeps track of exact version, which is installed.
* Node-moudles – fetches all the code of dependency and put it inside nodemodules
* I question – if we just install parcel – node modules should have only parcel in it why other many dependencies present -- this is because parcel has its own dependencies , its dependencies has its own dependency and so on.., that’s why node modules become huge and this is known as transitive dependency.
* If we want to install a package we write npm , if we want to execute a package we write npx.
* What is import React from ‘react’ , we are importing react from node modules.
* What is bundler do (ex parcel) – DevBuild , Local server , HMR – hot module replacement , file watching algorithm , Image optimization , Minification , compress , bundling , compress , consistent hashing , code spilting , differential bundling , diagnostics , tree shaking , different dev and prod bundles
* 

***Ep -03 || Laying the foundation***

* If we don’t want to write command again and again to start project in dev and prod , we make scripts for it for fast process inside package.json
* 
* When we do react.render it takes heading object and convert it to html element
* This way of creating element is tough so facebook developers created JSX – JavaScript Syntax helps to create react element
* JSX is not html in JS , it is HTML like syntax
* One component passed inside other component <Title/> is known as component composition.
* Inside JSX if we put {} we can run any JS code inside it. JSX also takes care of Cross site scripting attacks
* 

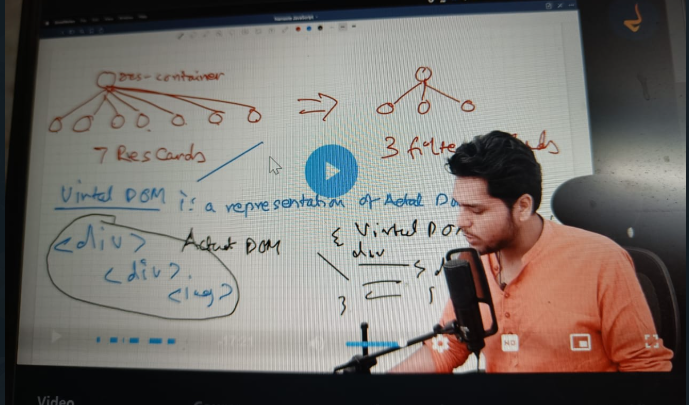
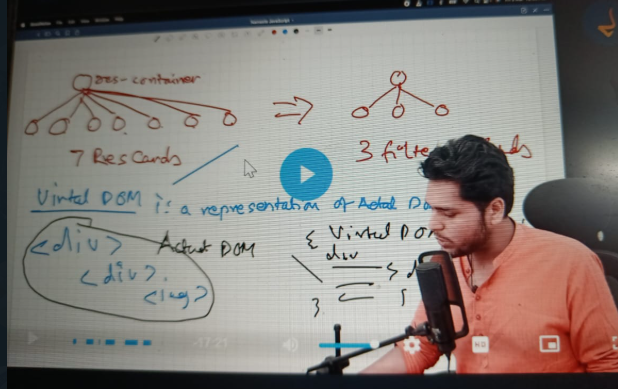
***Npm vs Npx --***

* Npm –
* npm is primarily a package manager for JavaScript. It is used to **install**, **uninstall**, and **manage dependencies** in your project.
* You use npm to execute scripts defined in the package.json file of your project (e.g., npm run start).
* When you run npm run start, npm looks for a start script defined in your package.json file and executes it.
* NPX – (Node Package eXecute)
* npx is a tool that comes with npm (version 5.2 and above) and is used to **execute binaries from the node\_modules/.bin folder** or temporarily download and execute packages **without installing them globally**.
* Unlike npm run, npx doesn’t look at your package.json scripts by default. Instead, it directly executes a binary or package.
* When you use npx <package-name>, npx checks if the package is available either **locally in your project’s node\_modules folder** or **globally**.
* If the package isn’t found in either location, npx temporarily downloads it to a cache and executes it, then deletes it afterward. This is helpful if you just need to run a tool once and don’t want to clutter your system with a global install or add it to your project dependencies.
* Let’s say you want to use the create-react-app tool:
* **With npm install -g create-react-app**: create-react-app is globally installed, so you can run create-react-app from anywhere on your system.
* **With npx create-react-app my-app**: npx will download and execute create-react-app (if not already installed globally), allowing you to use it without permanently installing it.

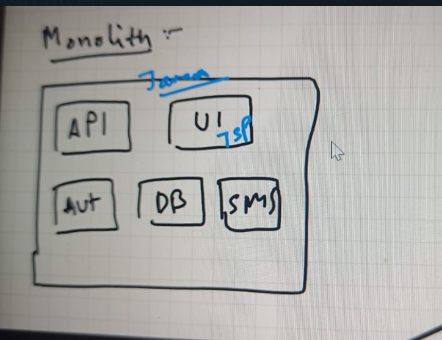
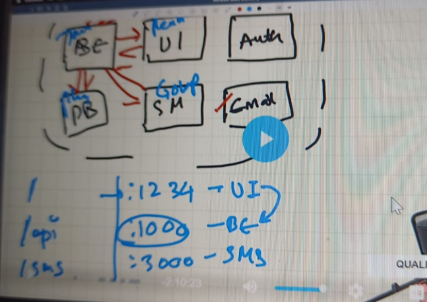
***Ep -04 || Talk Is cheap show me the code***

* First things first , visualize how our app should look like. (This is known as wireframe) || Then Think of components we can have looking at that wireframe
* Props – props are just normal arguments for a function
  + Ways to destructure props – {props.name} , {name} on component itself ,const {name} = props;
* Swiggy API Knowledge– Config-Driven-UI –
  + Our website is driven by data (by config) comes from backend– (like for different location there are different offer and restarant so different ui and card)
  + API data is config
  + We can even change a color based on this , we just need to pass color trough api
* Why unique key is required –
  + If we don’t provide key to a similar element , react will not be able to differenceiate between new dom element comes up in react dom tree , bcz of that react re structure entire dom tree /structure which decreases performance.

***Ep -05 || Lets Get Hooked***

* IMP – Never Ever keep any hardcoded data , string , anything hardcoded into component file. (It can be kept inside utils, common folder) this folder is we keep things which we use in entire projects. Name Constant in capital and snakecase
* Default import can just be used like import xyz from ‘./ffdf’ , named export are used like import {xyz} from ‘./xyz’
* we update the list it does not rerender ,, then we need state it rerender component and update the ui
* whenever react state variable updates , react rerenders a component
* HOW REACT WORKS ---
  + Reconcialation Algorithm (React fiber) – So lets say we have 7 res cards , react filter out 3 cards , whenever we have initial UI react creates a virtual DOM of it (Actual dom is <div> <div> and virtual DOM is representation of actual DOM (react virtual dom is nothing but object we can see it using consolelog of any component))
  + Diff Algorithm – Diff algorithm finds out the difference between two virtual dom the updated virtual dom and previous virtual dom it calculate difference and then actually updates the DOM on every render cycle.
  + Why react is fast – because react is doing efficient dom manipulation as it has virtual dom
*  

***Ep -06 || Exploring the world***

* Monolithic – we have one single project and inside that project we have everything like API code , UI Code Auth code , DB sms etc. Problem with this is if we have to do one single change we have to deploy entire Project again
*  
* Microservices – We have different service for different work like , Account service ,UI services , DB, sms etc .. Entire microserves talk to each other based on requirements , and this is also known as sepeation of concern and single respo.. principle (Each service is resp.. to do its own job) All services are running on differenct ports like 1234, 1000, 3000 etc then later they have domain name
* Two Approaches to make an API Call –
  + Page Loads 🡪 Make API Call(ex takes 500ms) 🡪 Render it on UI
  + Page Loads 🡪 Render UI (Skeleton) 🡪 Make API Call 🡪 Re-Render Page [BetterUX] (We follow this approach)
* **useEffect –** First components loads then useEffect will trigger.
* We load a fake page , until we get actual data from api.
* **useState** – Triggers Rerender