"ஆ"

React:- → A Java script library for building user Interface.

→ not a frame work like Angular and Vue

→ Because It don't have routing, which frame works have.

→ using external libraries It works better than Angular and Vue.

React → single page ↙ mobile App.

→ ~~It~~ use to make single page applications (SPA)

78-93) Video for ES6

ES6 => (let → variable, const →) Public => Var.

ES6	Arrow function

Normal function
function sayName (name)
{ console.log (name);
}
sayName ("ajeet");
	let sayName = (name) => {
	console.log (name);
	}
	sayName ("ajeet"); (without function).
	or
	let sayName = name => name; (Argument)
	console.log (sayName ("ajeeth");
	⇒ ajeeth.
	only If single argument is passed.
	let double = num => num*2;
	⇒ console.log (double (10));
	⇒ 20

=> // Export and Import (modules)

-> going to use this for React.

Doubt)

1. Bracket pair colorization Toggler Extention.
2. What is emmet ?

Array.

myarr.pop();

myarr.push("Larry")

getElementbyclassName

gEbcN

(VS code में ये डालो तो आ जायेगा।

```
function sum(a,b){
    return (a+b);
}
```

Arrow function

```
sum = (a,b)=>{
    return a+b; }
```

Namaste React

lec.

 → line break

1) package.json is a very imp file, it tracks the version.
   → Never put it in git ignore
2) node_modules is the database for NPM
3) If I have ~~package.lock.json~~ package.json & package.lock.json, we can create node_modules exact the same.

I will never touch node_modules and package_lock.json

HMR ⇒ Hot module Reloading
→ page reload after saving in web server

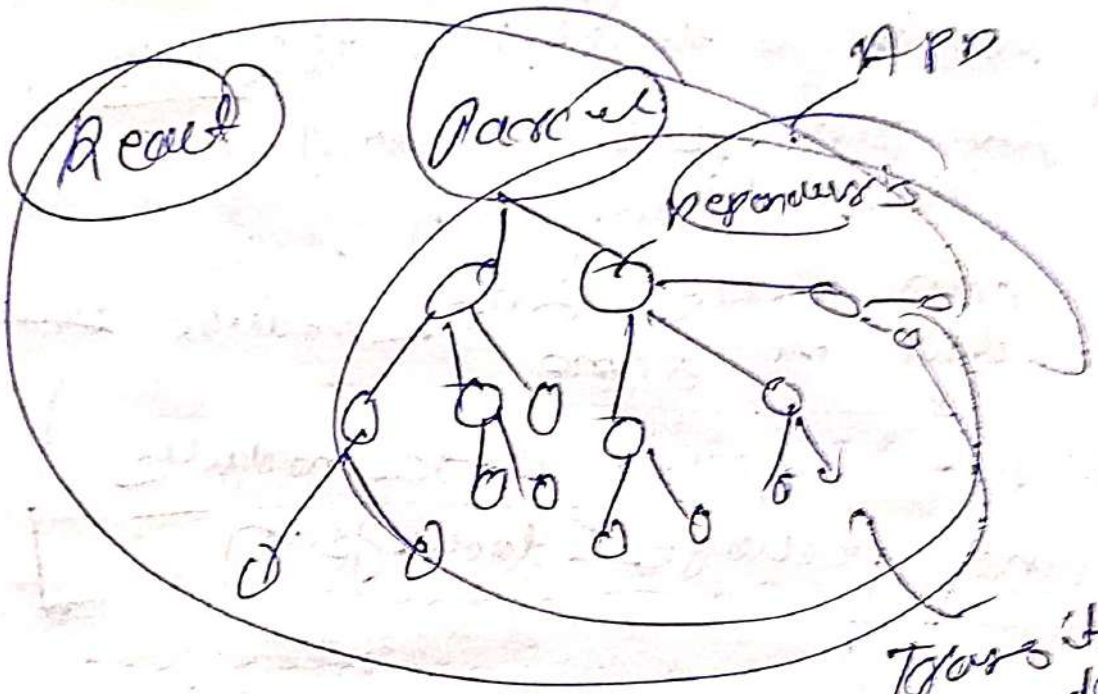
⇒ Anything which is auto-generated should put inside .gitignore.
⇒ or anything that can generate in server.
node_modules from npm install
dist. ⇒ from parcel
parcel-cache - parcel

→ parcel with node_modules provides to the App.

We have a package manager which takes care of transitive dependencies

Transitive dependencies → very imp guy - this word in interview

-> when building app, we have to do a lot of things. (Node Package), so we need dependencies, which is also dependent of something.



(APP)

-> Browserlist make it compatible with most of the browser, which is present in node_modules, and parcel do this for us.

and you use browserlist.dev and add required instruction to package.json

---

-> React.createElement => object => HTML (DOM)

-> JSX is a fancy & convenient way of writing HTML inside JS. It is not a HTML inside JS.

→ React keeps the track of the key.
→ HTML tracks I.D.
→ Facebook developers developed JSX.
→ Babel understand JSX.
  ↳ Reads your code and give you another code.
→ JSX is not React, Its the work of Babel.

{who say it is easy,
frontend is not easy, babel & parcel
like things made it easy}

→ Babel came along parcel.

React component

functional (new way)	class Based (old).

→ Name of component starts with capital letter is a good practice.
→ functional component is a normal function.
→ React element → object.
→ React ~~for~~ component ⇒ function.
→ We can't do css on key.
  Id → html thing & key → React thing

-> Babel basically is a compiler.

-> functional component return JSX.

↳ <Title/>, {Title()}, <Title></Title>

=> 3 ways to write f.c. in JSX.

=) We can write J.S. code in J.S.X in {}.

Always Plan before code

-> Any JSX component, there can only be one component.

const jsx = <h1> JSX </h1> valid.

const jsx = (<h1> JSX </h1> <h2> JSX </h2>) invalid.

So, add this into a div.

OR

use React.fragment component

Imported by React.

empty tag

<React.Fragment> = <>

</React.Fragment> = </>

" object is not JSX, It is Javascript."

→ Inline style in React के लिए Reference है।

→ we can use HTML Inside JSX, not J.S
So, we use obj. and use {}

[Humko style के लिए obj लिखना पड़ा Jo J.S. है वो है।]

→ [Dynamic UI are called Config Driven UI]
└ like in swiggy
Pune have other
other where Banglore have other.

→ we control our UI using config, that's why It is called that.

→ Backend running that config.

[I am building the API, by config driven UI, It will be great plus point.]
└→ for Interview.

{ system design }

→ Passing prop → passing property
→ passing Data

Note - side one
we receive Argument in our parameter.

Props

Props -> property -> parameter

→ connect this with Normal funct. parameter to understand.

res.list[0]
res.list[1]
res.list[2]
} ⇒ Arguments.

⇒ React Wraps up all argument in to a variable called props.

→ Arguments are obj

⇒ If confused refer Akshaysaini video, lec 5, (2:04:00 2nd)

→ we can pass multiple props.

ex. < res = {res.list[0]} hello="world" />

→ Props pass करना मतलब Argument pass करना।

→ object destructuring is used in props.

→ optional chaining.

→ जिस के अन्दर हम जावास्क्रिप्ट J.S. Code
 likh सकते हैं।

→ Props को func. जैसे कहते हैं।

Restaurant is a obj
and prabe, is for printing Restaurant
 in consol so, we use {{}}
 for Restaurant and not for prez

Praebs is a arg of the Argue.

---

name = {restaurantList[0].data.name}
 (this is a J.S. in {}
 for J.S.x, this
 only a instruction,
 so it won't work.
 We have to store
 this ~~in a var~~ in name
 and then we can
 use name

name: (KFC)

this instructs is
giving us
this (KFC)
and
(so). we wrote name in
 Restaurant card function.

-> Restaurant list is Array of objects, that why we are using .map function

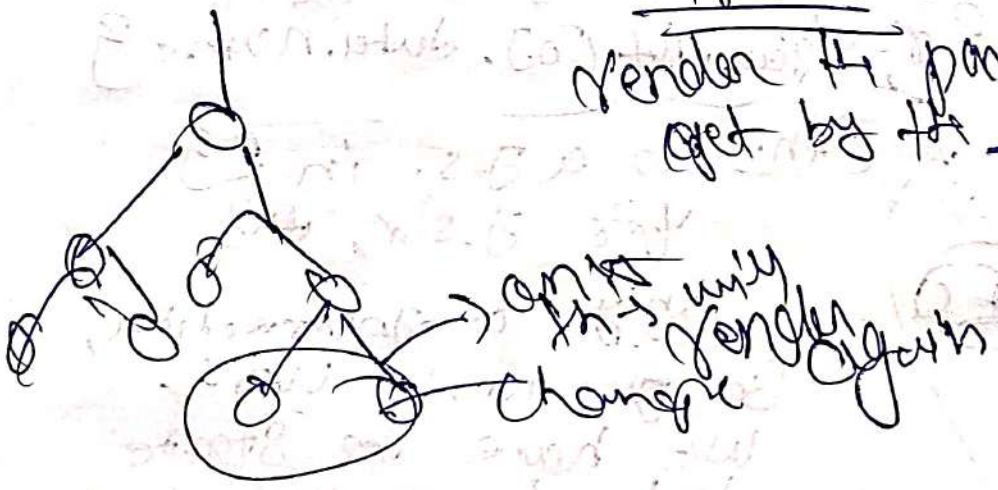
```
ecs (2 - 2:30) esmps
```

↳ watch it.

---

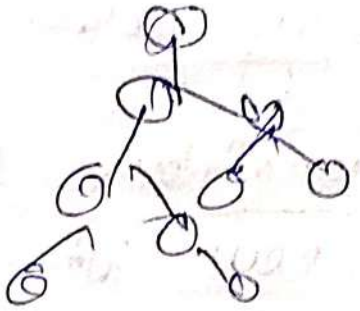
① Virtual dom is a representation of a Dom

② ~~Reconciliation~~ Reconciliation is used in react ↑ diff algorithm to find the difference b/w trees and render the part we get by the difference.



③ for order of elements (eg. sing) like what is changed it uses key.

Tree - virtual representation of the DOM.



React fibre →
new reconciliation engine for DIFF. Best practice

No key <<< index key << unique key
↳ least option.

---

⑥ Explain Interviewer about config-driven UI, and you know it quite well.

(Org Divide) So, UI render automatically.

=) If elements are same we don't need key because old will React will know it by its tag. Like

=> "we used public API of webpack"

---

contents/config.js
are same.

→ React uses one way data-binding

⇒ Hook → Is a function (Hooks)
that's why we call it
like [const [searchText] = useState();] (ESSD) (IIFE)

→ We use useState(); to create State variable.

→ This function returns an Array and the first element of the Array is your variable ~~given~~ name, ~~which is [searchText] here~~

→ [searchText] is a local state variable

⇒ useState() is a Hook which is used to create state variables.

returns = [variable name, function to update the variable].

Angular have two way binding.
which make unpredictable
sometimes / starts
why react came up
with this.

syntax for creating
variable in React.

const SearchText = "hellow";
in J.S.
is equivalent
to, const [S.T, S.S.T] = useState("hellow")
in React

default value

const searchtext

→ syntactic sweet =?

lec 6 1:30 - 1:45 (Jump)

local state variable

one way binding better.

→ React watches all state variable

→ whole component re-renders when it variable changes.

Reconciliation is the main thing why it is faster than others and we use it.

→ Diff algorithm

→ D.P. & behind the scenes.

Virtual dom

→ Tree is like a very big J.S. object.

→ React is to know that it have to keep UI is sync with the variable

→ Why we use state variable

→ as it is best possible with local variable due to one-way binding.

or we can say

React only track local state variable

→ It optimised our way in a great manner as it only changes on the element which is changed. Changes like only text element in HTML structure, this is called re-rendering.

→ It re-renders the component, because we may use the local state variable multiple time in the specific content

→ When we don't have to change variable we use local variable.
→ for exp. for test.

MICROSERVICES:

ARCHITECTURE

→ Here for a particular big Application, numerous projects are present, such as, for (UI, Backend, frontend, logs mgmt .. etc).
→ there can be more than one UI (now) project using different ports.
→ Different project ⇒ Diff. ports configure
→ here, Database replica is present.
→ so many databases are present.

⇒ Major advantage
→ easier to test.

Swiggy.com
microservice
→ we connect diff. services (UI, F.E, B.E etc) through diff. port no. under same domain name.
→ Github like company have diff. domain name also.

Imp

API → 1:15:00 से देखो।

Note

Interview के दौरान पूछा गया है

lec Exploring the world!

⇒ Two ways to call "API"

(1) load page → API (200ms) → render page (500ms) → 200ms

Total = 500ms

Time

(2) loads the page (1000ms) → API (300ms) → (500ms) update the UI

good way because of user Experience :

renders the page initial with initial UI

↳ get rendered page earlier than the 1st one.

In React, we generally always do this (2nd way)

↳ React gave us functionalities to make this happen.

→ 2nd most Important functionality useEffect();

Imp ↑

→ useEffect (() => { } , []);

Dependency array

This is a call back function, which which will be called ~~when used it~~ according to useEffect function/ Hook.

↳ React takes the responsibility to that It, call It at specific right time.

Re-Render happens when component changes and when props changes.

→ call back function runs, whenever It render (component).

→ that's we give Dependency Array.

→ In dependency Array is empty, call back function will only call once, as It is not dependent on anything's change.

Imp
→ empty dep. array → useEffect call after render.
→ dep on [searchText] → — " — + everytime after rerend (my search text changes).

Browser will restrict us, If we call swiggy API ~~with~~ by localhost, for searching parameters.

Watch camp Akshay saini

→ Inside JSX {} only JS. expression works not statement
↳ a=10, console.log()

→ so, we have to convert statement into expression to run.

console.log()

→ If we don't pass anything in ~~dep. array~~ use effect pass after every render. []

→ If empty [], ~~called~~ use effect called only once after initial render.

→ If [ ] present, called every time it changes.

Note :-

~~Never~~ create component inside component, we can but it's not a good practice

→ If we have com. inside App layout it will render every time and makes it very inefficient.

→ Never use useState, inside if / for (creates inconsistency) cuz useState has its own variable and react don't like this.

Not → I (1 min ई आउट के अंदर)

maybe ~~ही~~ console.log हटाया ~~है~~ था, सिर्फ
इसलिए रेंडर तो नहीं किया रहा होगा।

-> Never use use-state outside functional component.

{ CDN is more convenient to store image and other than keepin it locally, -> as it is more efficient & less. }

-> ~~Few~~ Don't install packages, untill the thing are complex.

> use formik library for form
> -> most used

Routing

Notes:- 1) we are using (createbrowserrouter).

2) ~~Route~~ we need to install a component Router provider from (react-router-dom),

-> we need to provide the app router to our app so we are using "Router provider"

-> It will have Array ~~of paths~~, which is the configuration, It is a list of path, each path is an object.

→ Because I want to render according to my router we will have to provide it in react.

→ Anything starts with "use" is a Hook.!

→ Anchor tag: refresh the whole page not good for Single page Application (SPA)

→ Two Types of Routing
1. Client Side Routing
2. Server Side Routing.

→ the code is already there, It is just putting up there.

→ Link to also uses anchor tag <a> behind the Scenes.

→ Want header & footer always.
→ want to change Outlet header & footer.

→ content in ~~outlet~~ outlet will change
→ outlet ~~sh~~ should render according to config.

→ All childrens will go to the outlet according to the route.

actually route is not changing Here, in SPA, the branch is changing,
→ the browser is ~~just~~ showing us ~~that~~ in that way" → lec17 (1:44:00)

→ ~~on root~~ route not changed, something in route changed!

→ outlet will not show in HTML ~~as~~ element in console.

Dynamic routing

→ use params can read the URL.

→ so we can read a dynamic url params

Param → url Id को read करना है और लाता है।

→ use to pass data to components.

thou is an Auto Import thing

=> components makes our code :-
1. modular
2. Readable
3. Testable
4. maintainable
5. Reusable

→ Interview में बहुत पूछा जाता है।

→ If API, keeps sending same specific data, optional chaining will help.
(for ex. Area name)

Lecture → 9 (lets get classy) - will help a lot in Interviews

→ Class based components
↳ more code, messy, Hard to maintain
→ Today people mostly not use this. as compare to functional component but still better than jQuery
→ Highly asked in Interviews !!

→ for nested route, make children of children.

→ ./about/profile

& use relative path only

path: "profile"

=> Slash ("/") means from the route localhost/

↳ without / it will take present path, & will go localhost/profile

Restaurant - menu

Mohd Amaan

→ In nested route we need to go to put "outlet"!

→ outlet is always created in parent.

→ At the end of the day, class based component is J.S = class.

V. Imp of class

⇒ can't create a class component without a render method.

→ only mandatory for class based component.

→ In fun. com. it returns JSX.

→ In class. Bas. / It's render method returns JSX.

→ We use extend here, because this class inherits the React.Component property

↳ J.S. thing → this give the class the superpower

⇒ for props we use this.props.name

⇒ Read, attach all the props to "this"

47:20

~~const [count, setCount] = useState(0);~~

Constructor (props) {
super (props);
}

51 min

→ constructor is place use for Initialization

→ constructor (props) {
super(props);
// this is the best place to create variable.

1) whenever class called → constructor called
or
Invoked

use state → So constructor(props) {
super(props);
// create state
this.state = {
count : 0,
};
}.

{ this state is object, which has this variable } →

Note

We do not mutate directly

↳ use use

this.setState({
count : 1,
});

{ watch during 3 min }

new concept is very specific

here, we can update variables whole together.

→ Because this kick start reconciliation process.

It is changed in new React (func) set count earlier it was very confusing for developers

Reconciliation

→ Player class mein mahi methods hoyge jo 'player' kar sakta hai.

let harry = new player()

harry.out()

{ run()
hitsix()
hitfour()
out() }

class player{ alert
out() { you are out }
quit() { player is ... }
}

{ → from code with Harry.

→ किसी भी class का जब object बनता तो constructor चलता है।

React life cycle (Every class is a life cycle)

→ nomenclature.

⇒ First constructor is called then render then componentDidMount()

Q. We use API call in useEffect() to make it fast & efficient.

→ So, we render what is in our local state then we fetch API.

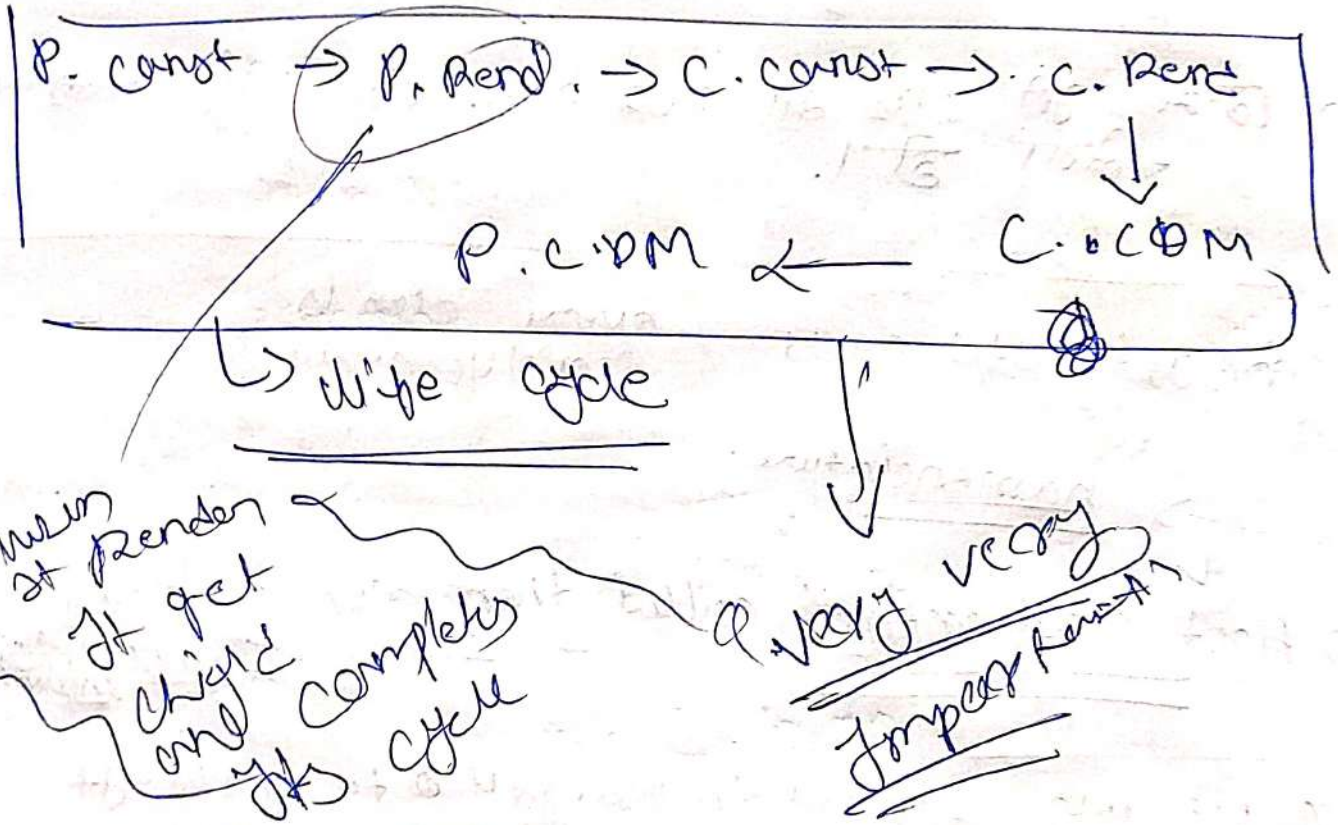
→ after fetching it renders once again.

→ So we call API from [componentDidMount] } best place to call.

(Q.) Where do we initialise our state?

Ans Constructor, because it called invoked everytime, when class is invoked.

1:35 – 1:38 → very imp



1:49 min

Render phase of all children along with parent done because if any one of the children have API call, the next children render will be delayed.

So, it will patch up the render phase of 1st and the 2nd child.

"Babel helps to convert JSX into html"

=) So, React life cycle have two phase.
- (1) ~~mounting~~ render phase — constructor, render
- commit phase -> (ComponentDidMount) & (React updates DOM and Refs)

Take from react lifecycle-methods-diagram/.

So, life cycle will look like this:

|F.C. -> First chil.|

P. constructor
↓
P. render
↓
F.C. constructor
↓
F.C render
↓
S.C constructor
↓
S.C render.

| DOM updated for children | } React will update the DOM!

↓
F.C ComponentDidmount (CDM)

S.C C.D.M.

P.C C.D.M.

Research homework! (Senior advance Developer Interview question)

Q. why we can make ComponentDidmount() A Async function but can't make ~~useEffect~~ useEffect() an Async.

useEffect function

V. Imp → Initially react will render things for you then It will call "API", because It uses Async function and It will have to take Data to load.

So, If there is an "API call" in child then.

P. constr.

↓

P. Render

↓

C. const.

↓

C. render

↓

P. C.D.M

↓

// API call, Json will load

Child - ~~render~~ C.D.M

↓

Child - render

this.setState() will trigger this render and the reconciliation process

this re-render cycle is known as updating

→ Its a Async component so It's called but It want to fetch data meanwhile P.CDM is called first

very Imp

=> When we call API, Data is rendered, we just have to ~~call~~ update it.

=> It updates the update cycle;

=> ComponentDidMount → called after first render.

=> ComponentDidUpdate → can called after every next render.

relate this to ~~useState~~ useEffect and dependency array.

Note: (but it is not equivalent to useEffect)

=> C. constructor F.C

↓

C. Render F.C

↓

// API

C. CDM F.C

↓

C. render F.C

Component DidUpdate.

=> {Component will Unmount :- Just before component is about to unload from DOM

↓

when we go to other page.}

→ Disclaimer / Note (Very Imp)

- Never ever compare React class life cycle method with React functional components.

---

componentDidUpdate(prevProps, prevState) {
    If (this.state.count != prevState.count
    && this.state.count2 != ,, ,,) {
        // code
    }
}

→ Way of writing dependency array in class based components

↑ refer chat gpt or Akshay Saini

★ when we go from about to contact, the concept of page is changing. but remember this is a (SPA), so, ★ there is a lot of things to clear in page before leaving, thus we use ~~Component~~ component will unmount. → Component will unmount.

→ watch Akshay Saini video for component will unmount ↳ very Awesome.

} If there is no props() & ~~setstate()~~ ~~update~~ w Component Did update will not happen. / happens when we change some ~~data~~.

=> Returning Inside function { return() => { - - - } }

-> we use this in functional component for unmounting when we leaving the page.

-> unmount happens after we route to diff. page.

Sec-II (optimizing our App.) (CH-9)

-> (code splitting and lazy loading)

-> (How to build custom hooks.)

=> moving forward to advance sessions.

=> why we build hooks:

1. reusability
2. readibility

-> Two important things

=> one separation of concern / maintainability -> perfect

=> more testable (If we separate different components)

-> easy to debug (So maintainable).

-> If we have modify, components, don't have to find. It is long long codes., If it is separates.

=> Hooks are also used for kind of same reason function as of function, as at end of the day it's a function only.

→ modularity & means we have broked down codes in meaning full pieces.
→ as every component has it own responsibility

→ If a file, export multiple things use how to named export and if a file is exporting a single component use default.

Function that returns a JSX => functional component

→ Hooks does not need to return a JSX, It is a piece of J.S. code
→ In extract some logic or value or J.S. or object, array

Imp.

→ use states return Array.

(can use that doesn't return anything, so it is also used in such a way.) May be exception?

custom
=) Hooks advantage is that If we don't have Hook and code inside component and if there is a change in code then whole component Reconcile but If we make a Hook, changes Hook will reconcile only.

→ If I have to build a feature, using hooks is amazing.

→ but not always.

Isonline = ?

addEventListener('online', (event) => { });

⇒ online = (event) => { }; (whenever you go online this code will trigger) Imp

⇒ We can't make state variable inside normal function; that's why hooks & can't get reconciliation inside it.

⇒ In Isonline Hook we use state (either online & offline के लिए) similar way

Developer have developed useParams. and similar Hooks.

{Build Id library and add it online ex. Isonline() lib. and add it in Resume}

→ mostly देखा गया अगर server / ya online lete hai to useEffect use करते हैं।

{ex. starts tin-talk-to, on ~~customer~~ content no. when offline }.

1:27 -> How to make it a senior Developer code.

↳ we introduced because there can be multiple event handlers.

1:39 -> To make consistent & systematic & maintainable.

Parcel

Bundlers => Bundle your code and bills.

↳ for whole components it creates a .sing JS file.

(1:52 - 1:54)

=> for development the size of the index file is more, in development the size of the JS file will be small. (minimizes in even more in development).

=) now, if bundlers take all the components (may be 100s, 1000s) then index.js file will be very large and it will make our app slow (very slow) } may be it will burst 😊 }

=> large scale production Ready application, they can not work, if we create just one bundle (giant bundle).

{Doubt} {-> why not shimmer run, while using useOnline() hook!. {when we get offline}}.

=> Study early return once.

I think, जब Net off होता है, तब उधर कोई component हो, code true होकर Run हुआ हो, Browser का नहीं read करेगा पहले का नहीं करेगा that's why we get offline status and not Shimmer! (I guess) {Check this out if true or not!}.

=> Splitting code:-
- -> chunking
- -> code splitting
- -> Dynamic Bundling
- -> Lazy loading
- -> On demand loading
- -> Dynamic Import

} -> same thing

-> Each part is a different bundle in paytm {-> mobile recharge -> movie tickets} etc.

-> will help in system design Interviews

Senior Interview: will think he has senior developer qualities

eg. make my trip is a large one so, will not more than that.

{Food villa is not that big to use bundling but we will use to learn} -> Imp

-> few lines of code can optimize your app heavily.

→ this kind of dynamic Import!

const Instrument = Lazy(() => import("./col-1"));

(Eventually It is a promise).

↳ from React library as name Import.

→ React will try to render the component, which is not even there (Instrument), so It will suspend the operation.

⇒ upon on Demand loading → upon render → suspend loading.

→ because the (Instrument - JS) will take time to load & React is rendering It even before that.

| The Solution is Suspense |

↳ my funny logic is because It is suspense what is loading — way to remember

⇒ we have "fallback" feature in 'Suspence'

for Shimmer (for fallback UI)

Imp Note

① Inside component never use lazy load / Dynamic Import.

↳ make less performant.

↳ Because It will lazy load after every render cycle. (उस हर component बार बार लोड होगा।)

↳ use on the top always.

12:38 → 2:39
↳ motivation

(connect functionality)

CSS / Tailwind

To write optimized CSS and to save time.

→ How to style?

→ makes the CSS writing experience good.

{ SASS }
↳ less only at the end of the day.
↳ CSS pre-processor

→ Inline style takes a J.S. object (in react)

style = {}

const styleButton = {color: red}

style = {styleButton}

or

style = {{color: red}}

Passing J.S object inside style

J.S in JSX

→ Not a good way of writing CSS.
CSS flexibility is not there, Hard coded.

→ Tell styling element in system design interview also.

Why we use CSS library?
→ (1) Built-in components.
→ (2) consistent UI.
↓
like all buttons are similar.
→ (3)

Tailwind CSS → uses class Approach.

material UI → uses component Approach.

→ It is a normal package, and we can have more than one package if needed.

→ Ideally, we should use one only.

→ consistency is very important for UI / UX

Cons of using JS

1. bundle size
2. lose control over designs.
3. restrict you in some way

Big pros

1. It makes the development very fast.
2. easy to code.
3. consistent design when working in team.
4. Take care of responsiveness (phone & laptop)

Total types (discussed)

1. Normal native CSS
2. SCSS
3. Inline CSS
4. Component libraries (material UI)
5. Tailwind CSS frameworks.
6. Styled Components
   - CSS with HTML at a single place
   - writing CSS in JS

→ as a senior engineer, know the pros and cons of all for interviews

Chakra UI, Ant UI, material UI, Base UI. { 1:30 min.

Tailwind CSS

-> when we start using, it changes default behaviours of a lot of things & tags.
-> It overwrite everything.
-> It tells us that you have to code in my way.

PostCSS :- transform css with power of javascript.
-> Dev-dependency tool.
-> used for compilation files.
-> Basically, we need to tell our parcel we are using tailwind css, so convert.

Tailwind जिन को convert करता है की, Browser वह समझ सके।

=> Initialise it with npx tailwind css init, after installation.

npx is to execute directly.

=> tailwind.config.js. (1:30) { about configuration }

=> we can use it in the file extensions mentioned in content [] { -> tailwind init file produces this.

-> • Postcssrc -> (To tell bundler (parcel), while building things up, we are using tailwind, so, compile our tailwind).

-> Ctrl + Spacebar (when [illegible] extension not giving suggestions).

-> use square bracket notation to design according your magnitude.

```
<h1 class="w-[200px]">
```

-> Tailwind takes care, that only that class will go to production, which is used from tailwind.

-> try to stick with native tailwind classes, not create new customized classes.

=> Using Tailwind we can style it on the go.

=> It also works with HMR.

Tailwind

Pros

1. Easy debug
2. less code
3. No duplicate css.
4. less bundle size.
5. Faster development.
6. customizable
7. control over things

It's a mature framework

Cons

1. Initial High learning curve.
2. makes the class name look little ugly. (but overall you have to write less CSS).

→ now, we can do everything inside our J-S. file only. (HTML, CSS, JS).

→ Interviewer will ask why you are using this steps, so you have explain those. (mainly in system design).

Chapter - II (Data is the new oil)

→ Handling data is very important.
→ There in one UI layer and one data layer in web app.
→ Babel converts JSX (React) into J-S. objects. structure.
→ the object structure is virtual DOM.
→ and then it is helpful for reconciliation.
→ for reconciliation, it compares the previous virtual Dom with the current virtual DOM.
↳ whatever the difference ↳ diff.
is then reflected to the actual DOM in sync.
↳ can say, React in sync with actual DOM.
↳ what we see in web page.
→ whole UI layer is powered by Data layer.

Data

we manage UI with states and props and with more things

Prop -> When we have to pass Data from one component to the other components. It is prop.

State -> It is a local variable, scope limited to the ~~component~~ container only.

-> prop is for local state for its parent.

-> All restaurant is a state for body, and we are passing it as prop to restaurant card.

Props DRILLING :- (passing props from parent to child to child)

AppLayout
(state=user)
- <Body user={user} />
  - <RestaurantContainer user={user} />
    - RestaurantCard user={user}
      - <h4>{user}</h4>

-> React Developer tool will help us in debugging Data layer.

[29 min] → difference b/w UI layer & Data layer

31:35

~~Array~~

→ what if we want to send props/data from children to parent. (In very rare case).

→ Custom Hooks → can be one option.

⇒ Data management will decide whether your app is good, scalable, reusable or not

⇒ Data is one of the most ~~crucial~~ crucial part in your Application.

managing
⇒ structure your Data good.

⇒ using props Drilling it will render all components props is passing through, even if that components don't need that prop { ← (cons)

└ this lead to difficulty in debugging also. (because in in console we see body, with props ~~the~~ which it don't use lead to confusion.).

55 min (Instrument bana ke pehle code pe no)

→ we built our own accordion. (show/hide)

→ Every section has its own props & state.

→ we want to modify the state of sibling (for collapse accordion).

↳ (we cannot change the state of siblings) section

1:42 → so, we give the right to change state from section to change its own state to parent component for state change.

→ this concept is known as lifting the ~~things~~ state up.

Imp 1:42 (-> Hide is not working.
-> because when we click Hide
it runs setIsVisible again
0 in future
eventually (about)
is true only.

Important
Hide button
React

See
In hide change setIsVisible
to setIsVisible section.

-> React Dev tool
-> great tool for debugging
-> ① profiler
-> ② components
-> give you ~~0 mount~~ time of components to load.

-> this makes your App performant.

Learn more about React Dev tools

-> works while development
-> not ~~outside~~ in production

-> props drilling is a bad approach some times.
-> data which have to used deposited
-> so we place this in central space. (React context)
-> React context -> In React only

-> local ~~Storate~~ Storage
not a good option, reading the operation costly operation, not reliable, Inside browser.

→ some components use Redux store.
→ Redux store is a heavy operation.

→ Any component in App can use it.
→ Shared state for the whole App.

→ without using third

React Context → diff. libraries for this operation available
→ React has its own context.

→ Context
It work by global variable.
→ as it is not tracked by React, so, Reconciliation will not work.

→ passing big object or some piece of data

→ Function at the end of the day.
→ useContext is a Hook
which is also a function.

→ we can have multiple context in our app. (we can created for users / we can create for cards etc).

→ Context is not specific to any component.

→ states and props are right to components

→ We use context when data is needed all across the App.

→ So, we can't Replace props & state with

→ and context is not specific as method.

Context is like the useState() for your whole big Application.

↳ the useContext() is a hook.

→ In class components we don't have hooks so, we will use it as component inside it.
(with different syntax)

↳ <UserContext.~~Provider~~ consumer>

⇒ Data is the new oil
→ climate part very Important.

Redux, context → used in React, Router, Basically provider used by meta Developers

pom

UserContext.displayName = "UserContext";

to display context name in Profiler for debugging.

→ UseContext ⇒ 2:15 – Dec End

⇒ Data is the new oil

Dec → lets build our store :-

→ Redux

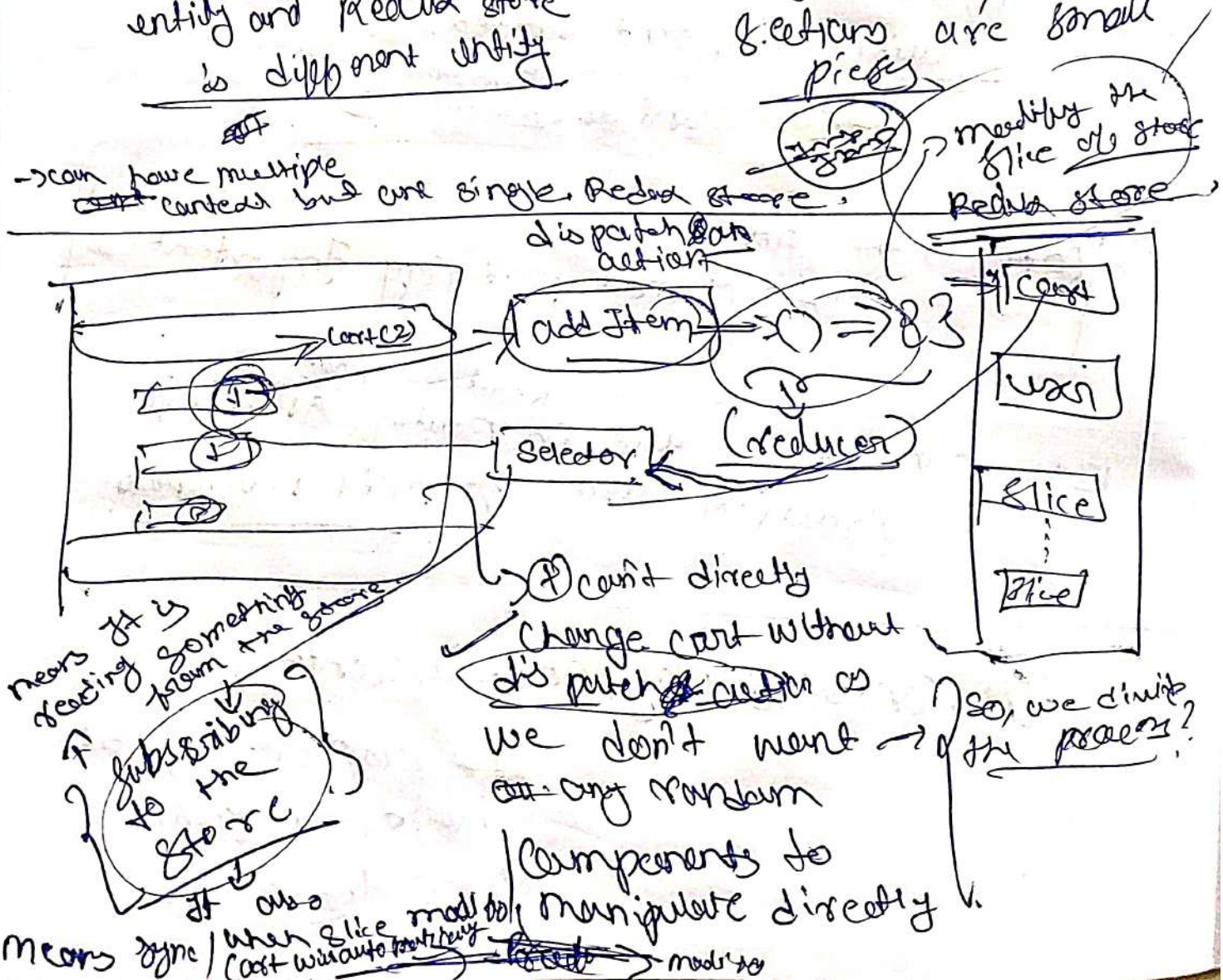
→ we use it to manage data layer of the app.
→ Handling huge amount of data.

23

→ Eventually Redux store is a big object which have diff sections, those sections are small pieces

→ your app is different entity and Redux store is different entity

→ can have multiple contexts but one single Redux store,



Dark theme / light theme is the best example for ~~context~~ context. ? It is handling state from header.

~~Redux~~ Redux.

→ when we click on the plus button (+) we dispatch an action which calls a reducer function which updates the slice of the redux store.

Redux is complex to ~~the~~ setup and have huge learning curve.

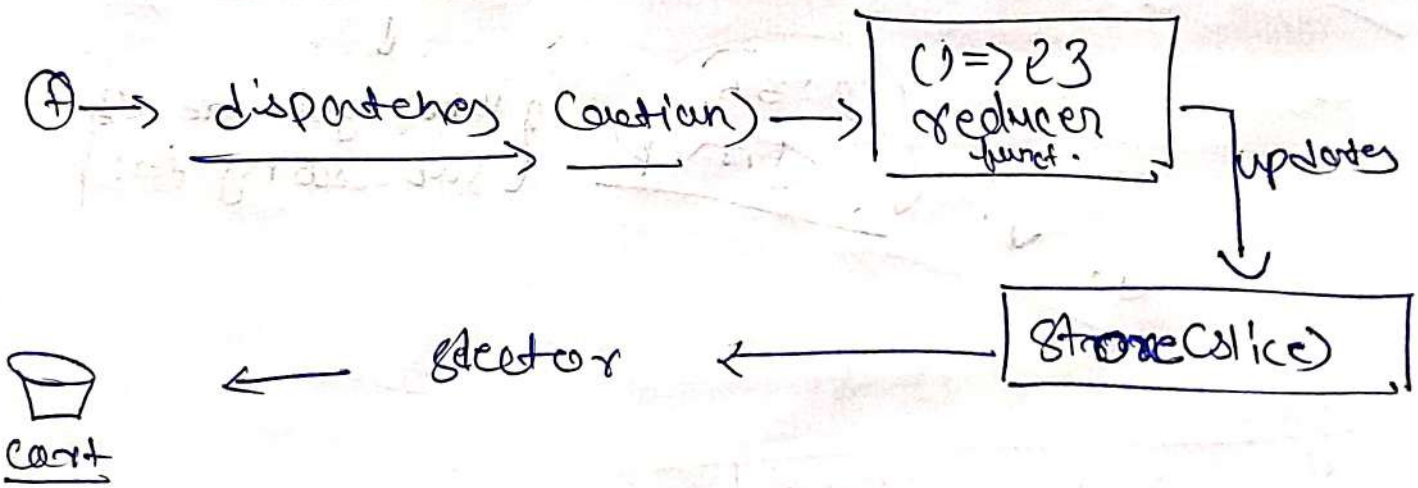
So, it came with Redux-Toolkit.

Very very Imp

Interview में जरूर पूछते हैं.

How we read it?

→ we use a selector (selecting the slice from Redux store), and this will update the cart.



→ Selector is a hook at the end of the day.

→ and Hook is a function at the end of the day.

1st lib @redux → core of redux.

2nd lib react-redux ⇒ bridge b/w react & redux.

---

addItem: (state, action) => { }

↳ initial state.

↳ Remember it doesn't return anything.

Import default cartSlice.reducer;

↳ It will combine all reducer, combined it and then export

2:05 (Jump)

2:19 useSelector (store => store.cart.items)

↓

{ what you are subscribing to. }

2:36 jump

V. V. V.

In Redux (RTK)

Use

1. Middleware
2. Thunks
3. RTK Query

state -> initial at starting later current state before action completed

action -> data coming in.

action payload
-> data we pass on the click of the button (+1).

dispatch (addItem ("Grapes"));
↑ dispatch ↓ action ↓ with payload

2:47
↳ jump
listen at

-> How to test your application in React:-

-> we are going to learn how to setup testing framework in our app.

-> to check if we are adding new code, we are not disturbing previous code.

=> headless browser?

=> React testing library is a package of testing library.

=> uses jest behind the scenes.

=>

① Install React testing library.

↓

Install jest (no it's dependent on V-13)

↓

configure jest. (jest --init) ← create jest.config file way to do it to do jest. v. easily 45 min.

↓

Choose test environment. (currently browser like environment).

↓

coverage option

↓

Choose provider for coverage (V8, babel)

→ .babelrc required requir JSON.

→ Find difference between Json & J.S. object.

→ Json takes double quote

(1:10 min)

hp 2 — when we have to do React only once.

npx → run package without installation

npx = npm run

47 min (How to run test cases)

→ Testing also has a huge learning curve.

mock fetch / running

```
global.fetch = jest.fn();
```

fetch returns a promise.

→ Machine coding Interview (V.V.V. Imp)

5 Interview tasks.

1. To-do list
2. Fetch Data from API.
3. Forms
4. Quiz app
5. nested - trees
6. Carousels.
7. Hooks
8. API call Debounce
9. Searching
10. Sorting
11. Infinite Scroll.
12. Higher order components
13. E-commerce websites.
14. Counter app
15. ~~Debounce~~ Debouncing.

12 Tic-tac-toe → Toughest part

1 live coding 2.

managing your Time

→ solution

→ practice

before

2 planning in Interview & Execution

~~Live~~ V.V.V. Imp.

{Youtube}

→ **Planning & Execution**

→ Requirement clarification

(I) ① features (Discuss It).

② Tech-stack.

eg →Redux.
→ Tailwind
→ react router dom
→ bundler.

(5 min discussion)

→ Discuss this If you choose this and give the Justification why are you choosing It.

→ Jest, react testing library.

Plan (5 min) — (UI Design or low level design) → How you structure your Design → How your code will flow.

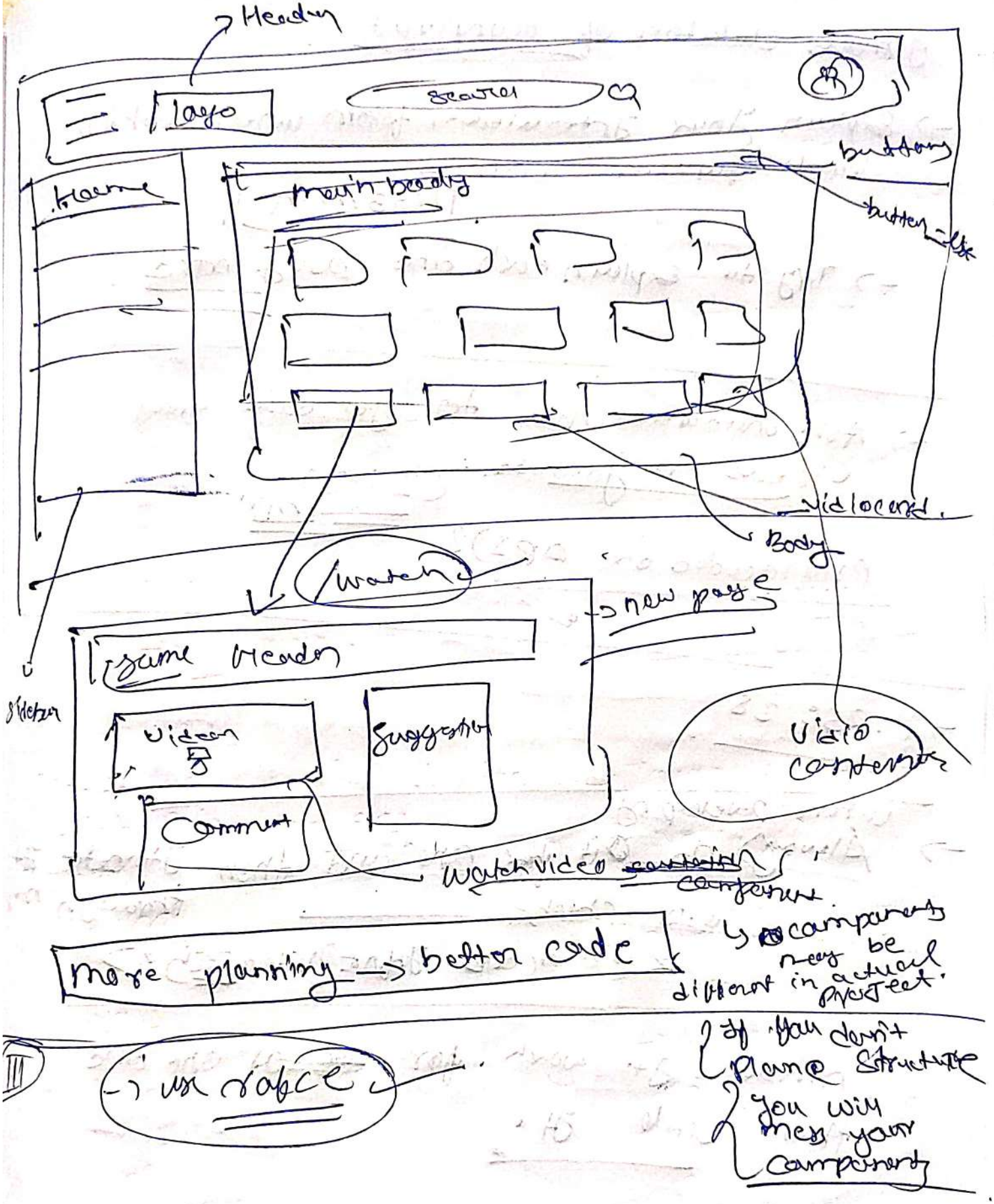
(5-10 min) ↳ (Discuss ~~plan also~~ your approach)

(II)

(47th min) ↳ Imp.

→ (III) Start Dividing components in planning. (5-7 min)

→ PNPX vs NPM (for once) → Imp.


Header
Logo
Search
Home
main body
buttons
button-list
Video card
Body
Watch
new page
same Header
Sidebar
Video
Suggestion
Comment
Video container
Watch video container
more planning → better code
UI race
If you don't plan a Structure you will mess your components

① make skeleton of components.

→ Explain your Interviewer from where which Hook / function came 11:58 min.

→ Try to explain each and every word.

→ Ask Interviewer how to get data: your videos in youtube. (2:08) min (Hardcoded or API)?

→ 2:25:38

---

when developing

→ Always do it for one and then iterate it through map.

~~e.g. check~~

<VideoCard Info={video[0]} />

→ make it work for ~~one~~ one and then scale it.

2:55

Hobbies & Co-curricular

=> URLSearchParams -> Imp.

3:13 -> Imp.

=> get Info In webpage, like subscribers count, like course with help of video ID/ useparams work.

Search { Search bar is one of the most asked questions in Interview. }

and nested comments

Ayushgupta.in/my-yantra

-> essa banna h Daala.

Sab Bawaal ho jayega.

Freelance

or

Youtube. Ayushgupta.in

{ Subscription toggle 3:28:13 for my video on various topic!

Hoc is used for modification!

1: 6 min (set timeout) debouncing

Arrays > object search > map search
search
O(n) O(1)

→ lru cache → study about this..

→ Youtube have two level deep comments, we will make n-level nested comments like Reddit.

→ will use recursion and loop it.

→ n-level nested comments Important

→ V. Imp.

live chat

challenges

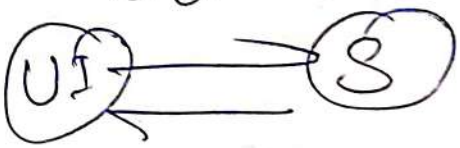
1. Get Data live (Data layer)
2. update the UI. (UI layer).

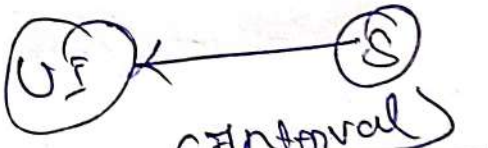
Data (live) (whatsapp)

→ web sockets (bcoz zerodha trading apps)

→ API Polling (Gmail)

(bcoz regular interval)





(Very real time)

(Website scroll → 2:38:00 min)

remove strict mode so that it will not call twice time

<> App.js

</>

index.js

3:47

→ with (dirty) we don't use server side routing.

Dynamic website