**What I left in this course bcz of laziness or exhaustness**

**Section 3=>composition, wrapper**

**Section 4=>updation of state when depends upon previous state**

**React JS with** [**Maximilian Schwarzmüller**](https://www.udemy.com/user/academind/)

* React is a client-side JS library for building user interfaces.
* React makes building complex, interactive and reactive user interfaces **simpler.**
* Traditional websites takes time for the response when you click you had to wait for the result but with react mobile/desktop app is reactive.
* React is used to make SPA \*single page application\* just like Netflix we looks like multiple pages but in reality it is just a SPA {because just within a single request with take over complete. pages and then on the request decide which part is to render}.
* React is all about components.

**Components**

React components are JS function that return markup…**DOC**

Components are the building block of application.

Components are based on HTML, JS and little bit CSS

Components are just function which return html code.

**Why We Use:**

**-->Reusability.**

Reusability means don’t need to repeat same code.

**-->Separate our concerns.**

Don’t do too many things in one and same place.

**Approaches**

**Declarative:**

**<**we just select the goal mean what should display on the screen and let the react to do its work**>**

Let Declarative= ’ What‘   🡪GOOD approach 🡪React uses Declarative Approach

It means that Declarative Approach is exactly about the fact means to the point not in extra detail.

O.S= Declarative Approach is just the like the Abstract.

e.g. person book 2 chair

You don’t need to say that specific element should be in the that place or order to REACT as VANILA JS need to explain.

<div>

<p>Hey I am declarative</p>

</div>

**Imperative:**

Let Imperative=**’**How**’ <**clear step by step instructions **>**

Imperative approach is in all detail.

Let para=document.get elementById(‘root’};

Para.textcontent=’hey I am imprarative’;

document.getelementById(‘root’}.append(para}.

e.g. A person go the hotel order coffee ask waiter to book 2 chairs etc.

**29-Exploring The React Project**

* Index .js file is the first file which will be executed first. root is the entry point in the application.
* Index.html is the only file that is present in the project which contain nothing just a div with id=’ root‘ this is the reason why we call it SPA.
* App.js is the main component we can say it a root component and all other components that we will make are custom components. All customs components are import in the App component means in main component.
* Custom component must start with the Capital Letter because when it imported in the App component React recognize it by its style.

**36-Props**

**{----PROPERTIES---}**

Props are used to store and access data coming from the App.js into our customized component.

React Props are like functions arguments in JavaScript *and* attributes in HTML.

To send props into a component, use the same syntax as HTML attributes and to receive props in the custom component as JS function parameter.

**Practical for sending and receiving the data through props.**

**We** must have data in component from where we are calling the other component like in our case<**App.js** is main and **ExpenseItem** is custom component>

|  |  |
| --- | --- |
| Caller / Sender  App.js  }  In it We have data in array or obj format.  Expenses=[  {title:abc,  amount:123,  etc},  { title:xyz,  amount:183,  etc  }  ]  <ExpenseItem item={Expenses}/> | Receiver…….  ExpenseItem(props>{  Props.item.title,  Props.item.amount,  Etc} |

**37-Nested Components**

Means that we should make a component inside the component for the sake of neat and clean code in our components. Mostly developers make a new component of the repetitive code.

**39-Composition**

Make a component from the small component is call composition.

But we can use composition in another aspect like wrapper

Wrapper means sometimes we need to remove duplication from the html files of component we make wrapper.

**41 JSX**

it is special not standard syntax which is enable in React project which we create in by command of **create-react-app**

Without jsx code will be more complicated just like we have done in App.js.

Where we also clear that why we should wrap cod in single div.

**45-States**

**If we want to reEvaluate the data/value coming through props we <old method>used to store data in variable and update it but React<react does’t care whatever you change it only listen to state changer> doesn’t work here that’s why we use state to update the data.**

By using the props we can’t be able to change the data dynamically so the concept of the States come into notice bcz by using states we can make our page dynamic.

**useState:**

it is React most important Hook.

Normal assignment is not understand by the **React** that’s why  we use useState to update the props.

useState is a function provided.

**Identify the Hooks:**

They contain ‘keyword’ ‘use’ and always use inside the react component function.

**57-Two way Binding**

Its means that when we enter value at the same time we set the entered value to the state so that when we need to reset the input state we at-least have a backup of the entered value.

1-read the entered data

2-store the date into e.g.value={enteredvalue}

**58-Send Data up to the Parents**

Props: are used to pass the data from top-to-down means parent to child without missing/skipping the intermediate channel/component.

For sending data upward we use another method in which pointer to function and function itself is used to send data upward.

|  |  |
| --- | --- |
| Parent Component  In parent we make a function which receive the obj and using spread operator <…obj>  store in the obj  When call the child components pass that own-made function to the child for collecting data.  **Practical:**  Function name(enteredValue>  {  Const obj={  …enteredValue,  Id=math.Random<>  }  Console.log<obj>;  }  ////when we call the component then pass the created function  <FilteredValue  onChangeVal={name}/> | Child Component  In child we collect data from the user and store it in the obj then we pass back that obj to the parent  **Practical:**  Component containing the value  Function FilteredValue<props>  {        -           -           -          Props.onChangeVal<collectedData/enteredData>  } |

**59-Lifting up State**

We have learnt about sending the data from parent to child and from child to parent

**BUT**

Sometimes we need to share the data to siblings so that purpose we don’t have proper channel so for this we send data to the parent  then send to **another child through props.**

**6O-statefull and stateless**

|  |  |
| --- | --- |
| Stateless  Dumb  Presentational component | Statefull Component  Smart Components |

**65-key**

<<<we should use the id to help the react to identify the element it improve the performace otherwise react search the whole data and render something on the screen>>>

**66-logical lecture 😥**

**75-Dynamically adding inline styles**

In this I learn how to style dynamically but inline which is not good because it will override all other css classes and get high priority

First time I completed the exercise on my own

**76-dynamically css classes**

I have completed another exercise on my own.

**77-styled Component introduction**

It is problem that if in long project 2 or more components and same css style name which may disturb the another component’s styling.

**Solution:**

1st Styled Component:

This is a weird way of applying style but very effective. All the css file will mix with the JS file and things become lightly difficult.

2nd Module CSS:

**1o9-useEffects**

**If you know the React JS classes life cycle methods useEffect is the combination of all the methods**

**{componentDidMount,componentDidUpdate,componentWillUnmount}**

**It is a React JS Hook and its job is to handle the sideEffects like http or email/password verification and so on**

**It helps you when you use the code which execute in response of another response/action**

 React purpose is to render the table and get changing by the user and update the DOM and then rerender. But other than this everything will control by the useEffect.

For example: most websites send request to web/local browser to get or store data then React re-render the date on the screen if we request data and mistakenly apply infinite loop and React is rendering and re-rendering the data again and again such type of error controlled by the useEffect.

**113-Using the useEffect clean function**

**Debouncing:**

When user enter the value in the input and take a pause at that time we check the validation this is called debouncing.

{Most of the time we check validation on every keystroke which slow down the performance}

for that purpose we use the cleanUp function that is built-in by React named as **clearTimeOut()**

**115-useReducer**

it is use in replacement of the useState which means that it is use to handle the more complex state.

when ever state is updating on the basis of another different(not the snapshot of the same state) state. Where we should use useReducer.

e.g.  setForm is updating on the updation of email and password

const[state, stateupdatingfunction / dispatchFunction]=useReducer(reducerFunction,initstate,setInitState);

dispatchFunction=it will send the action to reducer function which takes latest state and action return the updated state

**122-Context**

**Component wide state storage “Behind the Scene”**

**LIMITATIONS FOR USING CONTEXT:**

Context is not optimize for high frequency changes.

Hooks are used to access the feature of class components in function component

**126-Rules for using Hooks**

1. Only call React hook inside of the React Function or custom Hooks
2. Only call React Hooks on the top of the function not inside any condition or any other function

**This is for useEffect Hook**

All the data which we use in the component should be use as dependencies rather than data comes from the surroundings of the components.

**REACT ROUTER {STEP-BY-STEP}**

* Angular, React, Vue and Laravel etc. All use the Router technique to navigate from one page to another page.
* Anchor tag is used to for same purpose but by using it our whole application get reload it reduce performance.
* 4o4 page: is error which occur only when we visit page which is not find. to=/\* is used for all other link
* Param also use in all front end libraries and framework but their method is different everywhere.
* Param is used to receive the data{that I use in my router website}
* useSearchParam is react Router Hook. It is use to get/set the value from the LinkBar. setSearchParam is used to set the value manually from the component. It means age=23.
* Important is that setSearchParam is set by the use of callback function.
* useNavigate is react-Router-Hook which is used in button to move from one page to another page.
* useLocation is **react-router-hook** is used to receive the data coming from url. Data means key, state, hash and path.
* Protected Routing is used for the purpose to protect some pages without login. Means user can’t access any page until does not get login. It is not a react feature.

**Context API:**

* It is used to store data and provides data to components. By using it we save our data from sending to un-related component.
* By the use to props and lifting up state our data go to un-related component and make our program complex.
* Provider and consumer is the main concepts in context API.

**Redux:**

* Redux is just a store to manage the application state {not a component state}.
* Redux is not a database but it is like a array which is used to store and retrieve the data.

**Life cycle method**

3-phases :> Mounting Updating Unmounting

When life cycle methods run in program??

These methods run

1-when component load.

2-when state and props update.

3- when component remove.