**Theory Chapter 04**

**Q: Is JSX mandatory in react?**

**A:** No JSX I not mandatory in React one can write code use React.createElement. JSX is converted to React.createElement by babel behind the scene. If one wants to skip the compilation in build env. One can skip JSX as it ll be convenient in that situation . Any thing which can be done using JSX can also done by using plain javscript.

**Q: Is ES6 mandatory for React?**

**A:** No, One can use older versions of Js to write code in React . One can use the following code for creating React module like. createReactClass has similar API like ES6

var createReactClass = require('create-react-class');

var Greeting = createReactClass({

render: function() {

return <h1>Hello, {this.props.name}</h1>;

}

});

**Q: Comments in JSX**

**A:**  In the similar way to js one can write comments in JSX using {/\*\*/}

**Q: What is React.Fragment?**

**A: It’s a** common patter in React for returning multiple children/ elements from a component . Fragment helps us to return multiple elements without adding extra node to DOM . Shorter way to write React.Fragment is <></> . Key is the only attribute which can be passed in React.Fragment.

**Q: What is virtual DOM?**

**A:VDOM** is basically the virtual representation of our Real Dom in memory . VDOM is kept in sync with the real Dom with the help of REACTDOM api.

* VDOM IS ASSOCIATED WITH REACT ELEMENTS ARE NOTHING AT THE END THEY ARE JUST THE OBJECT WHICH REPRESENTS THE UI.
* REACT USES INTERNAL OBJECTS SOMETHING KNOW AS FIBERS TO HOLD THE ADDITIONAL INFORMATION ABOUT THE COMPONENTS AND THEY CAN BE CONSIDERED AS A PART OF VDOM

**Q: Reconciliation in React?**

**A:** Theprocess of keeping VDOM and RealDom in sync and to decide weather a update to the real dom is required or not . By comparing with the newly created VDOM

**DIFFING ALO –** it tells exactly what is changed in the DOM and what exactly needs to be updated in the DOM

* **When Root nodes/DOM elements are different**

1. If the root node is different then the whole old tree is tear down and new tree is build
2. When old is tear down then componentWillUnmount is called . and when new tree is build from scratch componentDidMount is called with new state and props. Any node below the root will undergo the same process. New component instance is made

* **When roots/DOM elements are same**

1. React then check for its attribute change and only update the changed attribute
2. Component Instance ll remain only update the props and componentDidUpdate is called

**Q: React Fiber?**

**NOTE : Reconciliation and re rendering are two separate concern.**

**Reconciliation –** deals in finding what data has changed when there is change in state/props

**Re-rendering** – it uses the information of reconciliation algo to update the real DOM

This separation of concern means that React DOM and React Native can use their own renderers while sharing the same reconciler, provided by React core.

**FIBRE -**  it basically contains addition information about the components . It re-implements the reconciliation process . It mainly represents the unit of work

It the reimplementation of react core algo

1. Main thing in this is incremental rendering - prioritizing the updates in DOM—one can abort the update—one can use previously rendered information – re rendering should happen in chunks .

**Q:**  **Why we need keys in React ? When do we need keys in React?**

1. **Key** helps to give the idea to React which element is stable between different renders.
2. When mutating/adding new element in children of the same type , it ll mutate every child instead of just adding the new element . if we are adding the element in the 1st position
3. Key has be unique among sibling. No need to make a its key unique globally.

**When to use**

When one have multiple children of same type .For eg multiple li of ul

**Q: Can we use index as keys in React?**

**A: Yes**  one can use idx as a key . but one should avoid using this. One can use idx as key only if the items are not reorder /filtered /static data , otherwise updation ll be slow.

**Q: What are props in React? Ways to**

**A: Properties(Props)** is the way to pass the data from one component to other

Components accept attribute input (called props) and return React Elements which describes what should appear on the screen.

Props are read only . Components should not modify its own props.

**Q: What is config driven ui?**

**A: UI**  which are render depending upon a config / data send by backend.

1. Config driven ui
2. Data driven ui using config

**Config driven ui**

**In**  this our component is taking a config using and then it is rendering element based on that . Taking an example of form . one can create a config for form elements and all the types of elements one needs in its form , along with the type of element and then render elements based on that

For eg it’s the config for better understanding

const config =[

{

name :’Name’,

type :text,

field:’input’,

isRequired:false,

label :’Full Name’,

value:’’,

placeholder:’Enter your full name’  
},

{

name :’Password’,

type :’password’,

field:’input’,

isRequired:true,

label :’Password’,

value:’’,

placeholder:’Enter your Password  
}

]

**Data Driven UI using config**

In this we are mapping the data form api to our config . like in the above example instead of having array we ll be having key value pair and after getting the data will map that data to our field . like after getting the data from backend like the username ll map to that our config