**Episode 4: Talk is cheap, show me the code**

**Theory:**

1. Is JSX mandatory for React?

JSX is not a requirement for using React. Using React without JSX is especially convenient when you don’t want to setup compilation in your build environment. Each JSX element is just syntactic sugar for calling React.createElement(component, props, children).

When we use JSX in React, tools like Babel which is a Javascript transpiler converts it to the browser understandable Javascript code. The main purpose of using React with JSX is that a combination of a programming language and a markup language makes the application robust and boosts its performance. So, anything that is done in JSX can also be done without JSX, using plain Javascript as well.

2. Is ES6 mandatory for React?

ES6 is the standardization of Javascript for making code more readable and more accessible. If we don’t use ES6 in React, there is an alternative to perform. We use create-react-class instead of ES6. Create-react-class is basically a module that we can use.

ES6 is an upgraded version of ES5 and something like Babel must compile down ES6 to ES5 for the browser to use and understand. JSX is a specific syntax, used in React that looks similar to html, but it is actually HTML like syntax and it is definitely not HTML in Javascript.

3. {TitleComponent} vs {< TitleComponent />} vs {< TitleComponent>< TitleComponent />} in JSX

{TitleComponent} – This value describes the TitleComponent as a Javascript expression or a variable.

{< TitleComponent />} – This value represents a component that is returning JSX value. In simple terms TitleComponent is a function that is returning some JSX code. If component is written inside the {< />} expression.

{< TitleComponent>< TitleComponent />} – < TitleComponent /> and < TitleComponent>< TitleComponent /> are equivalent only when < TitleComponent /> has no child components. The opening and closing tags are created to include the child components.

4. How can I write comments in JSX?

To put comments inside JSX, we can use the syntax {/\* \*/} to wrap around the comment text.

5. What is <React.Fragment></React.Fragment> and <></>?

React.Fragment is a feature in React that allows us to return multiple elements from a React component by allowing us to group a list of children without adding extra nodes to the DOM. To return multiple elements from a React component, you will need to wrap the element in a root element.

The <> and </> is a React shortcut for React.Fragment component. We cannot use the key or prop syntax with the shortcut syntax.

6. What is Reconciliation in React?

React Reconciliation is the process through which React updates the browser DOM and makes React work faster. React uses a diffing algorithm so that component updates are predictable and faster. React would first calculate the difference between the real DOM and the copy of DOM (Virtual DOM), when there is an update of components. React stores a copy of Browser DOM which is called as Virtual DOM. When we make changes or add data, react creates a new Virtual DOM and compares it with the previous one. Comparison is done by diffing algorithm. React compares the Virtual DOM and the Real DOM. It finds out the changed nodes and updates only the changed nodes in Real DOM leaving the real nodes as it is. This process is known as Reconciliation in React.

7. What is React Fiber?

React Fiber is a concept of React.js that is used to render a system faster, smoother and smarter. The Fiber reconciler, which became the default reconciler for React 16 and above, is a complete rewrite of React’s reconciliation algorithm to solve some long-standing issues in React. React Fiber is asynchronous, it can do the following things:

* Pause, resume and restart rendering work on components as new updates come in
* Reuse previously completed work and even abort it if not needed
* Split work into chunks and prioritize tasks based on importance.

8. Why we need keys in React? When do we need keys in React?

A key in React is a special string attribute we need to include when creating lists of elements in React. Keys in React are used to identify which items in the list are changed, updated or deleted. Keys are used to give an identity to the elements in the lists.

If we don’t have key prop, then React will re-render everything inside the particular component and it is highly inefficient if we have a long list of items to render by using React. Basically, it will clean the container and will render everything inside it again when we are not having the key prop.

When we give ids, React will render that one particular thing only and not the entire container.

Also, React itself says that “Never use index as a key”. It is mentioned in the React Documentation.

9. Can we use index as keys in React?

Yes, we can use index as keys in React, but it is not considered a good practice to use them because the order of items may change. This can negatively impact performance and may cause issues with component state. Keys are taken from each object which is being rendered. There might be a possibility that if we modify the incoming data react may render them in an unusual order.

10. What is props in React?

Props stand for “Properties”. Props are arguments passed into React components. Props are used in React to pass data from one component to another (from a parent component to child component). They are useful when we want the flow of the data in our application to be dynamic.

We can also say that it is an object which stores the value of the attributes of a tag and work like HTML attributes.

11. What is a Config Driven UI?

Config Driven UI is a technique that allows us to create user interfaces based on a configuration file such as JSON or a Typescript file that defines the layout and content of the UI components. This can be useful for creating dynamic and customizable UIs without hardcoding them.

Config Driven UI are based on configuration of the data application receives. It is a good practice to use Config Driven UIs to make application dynamic.

12. What is DOM?

DOM refers to the Document Object Model that represents the content of XML or HTML document as a tree structure so that the programs can be read, accessed and changed in the document structure, style and content.

13. What is Virtual DOM?

React uses Virtual DOM and it is like a lightweight copy of the actual DOM (A virtual representation of the DOM). So, for every object that exists in the original DOM, there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document.

Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen. So, each time when there is a state in our application, the virtual DOM gets updated first instead of the real DOM.

14. Difference between Real DOM and Virtual DOM?

|  |  |
| --- | --- |
| **Real DOM** | **Virtual DOM** |
| Real DOM represents actual structure of the webpage. | Virtual DOM represent the virtual/memory representation of the webpage. |
| DOM manipulation is very expensive | DOM manipulation is very easy |
| There is too much memory wastage | No memory wastage |
| It updates slow | It updates fast |
| It can directly update HTML | It can’t update HTML directly |
| It allows us to directly target any specific node (HTML element) | It can produce about 2,00,000 virtual DOM Nodes/second |
| It represents the UI of our application | It is only a virtual representation of the DOM |
| It is maintained by the browser after parsing the HTML elements | It is maintained by Javascript libraries. |
| After manipulation, it will re-render the entire DOM | After manipulation, it will re-render the changed elements |
| It is less efficient due to re-rendering of DOM after each update | It is highly efficient as it performs batch updates |

**Coding:**

﻿Build a Food Ordering App

Think of a cool name for your app

Build a AppLayout

Build a Header Component with Logo & Nav Items & Cart

Build a Body Component

Build RestaurantList Component

Build RestaurantCard Component

Use static data initially

Make your card dynamic (pass in props)

Props - passing arguments to a function - Use Destructuring & Spread operator

Render your cards with dynamic data of restaurants

Use Array.map to render all the restaurants

PS. Basically do everything that I did in the class, in the same sequence. Don't skip small things.

**References:**

﻿Code Link: <https://bitbucket.org/namastedev/namaste-react-live/src/master/>

React without JSX: <https://reactjs.org/docs/react-without-jsx.html>

Virtual DOM: <https://reactjs.org/docs/faq-internals.html>

Reconciliation: <https://reactjs.org/docs/reconciliation.html>

React Fiber Architecture: <https://github.com/acdlite/react-fiber-architecture>

React Without ES6: <https://reactjs.org/docs/react-without-es6.html>

Index Keys as Anti-Pattern: <https://robinpokorny.com/blog/index-as-a-key-is-an-anti-pattern/>