Assignment 4

Owner - Mohit R

1. Is JSX mandatory for react?

- JSX is not a requirement for using React.
- Each JSX element is just **syntactic sugar for calling React.createElement**(component, props, ...children).
- So, anything you can do with JSX can also be done with just plain JavaScript.
- Example code written with JSX:
  Const head = <h1>This is with JSX</h1>
- Example code without JSX:
  Const head = React.createElement("h1", {}, "This is without JSX");

2. Is ES6 mandatory for react?

- ES6 is not mandatory for using React but it's highly recommended to use ES6. Any javascript expressions (pure js) can be used in JSX to render React elements.
- Currently, lot of React projects use ES6 features in React ecosystem, so it's better to have knowledge on ES6 features like modules, DE-structuring, spread operator, template literals, classes, map, filter and reduce array methods.

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

- **{ TitleComponent }** - This value in jsx is considered as jsx expression or variable. If no such variable is present, no output will be shown in the browser. Console throws the following warning
  index.js:1 Warning: Functions are not valid as a React child. This may happen if you return a Component instead of <Component /> from render. Or maybe you meant to call this function rather than return it.
- **{<TitleComponent />}** - This value in jsx is meant for rendering a component (i.e) function that return jsx. This is self-closing tag.
- **{<TitleComponent> </TitleComponent>}** - This is same as {<TitleComponent />} if there are no child inside TitleComponent. If there are children, then those values come inside {<TitleComponent>} and </TitleComponent>}.

4. How to write comments in JSX?

- Comments are written like any other javascript code. In javascript, we use // to comment a single line and /* */ to comment multiple lines.
- Similarly, in jsx we enclose js code inside {} and hence comments are also enclosed within { } . Only difference is for single line comment instead of {//} use {/* */}

-
  ```
  const Header = () => {
    return (
     <h1>Namaste React</h1> {/* This is single line comment */}
      {/*
      *
      * This is multi line comments
      */}
      )}
  ```

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

- Each jsx element (component) can have only one parent. This is because jsx element is converted to React.createElement(parent, props, ...children) before rendering in the DOM.
- But the common pattern in React is for a component to return multiple elements. For grouping, we can enclose them within <div> </div>. But there can be situations where <div> </div> should not be used. In such cases, Fragments can be used to group a list of children without adding extra nodes to the DOM.
- The new, short syntax for declaring Fragment is empty tags <> </>. It can be used in the same way as any other element but it doesn't support keys or attributes.
- What if React fires a key warning? There will be cases where we might use Fragments while mapping a list of elements . And React will fire a key warning since every element must have a unique key. In such cases, Keyed Fragments can be used. key is the only attribute that can be passed to <React.Fragment></React.Fragment>. This is not possible with <></>.

6. What is virtual DOM?

- Virtual DOM (VDOM) is a programming concept where a copy/virtual representation of the UI is kept in memory and synced with the "real" DOM tree by a library called React-DOM. This process is called Reconciliation.
- In React, a virtual DOM is associated with React elements since they are the objects representing the UI. React, however, also uses internal objects called “fibers” to hold additional information about the component tree. They may also be considered a part of “virtual DOM” implementation in React.

7. What is Reconciliation in react?

- React uses diffing algorithm to diff one tree (actual DOM) from another which determines what needs to be updated and only re-renders the diff.
- In React, we pass props to a component. When any of the prop changes, a reconciliation process is triggered internally by react which traverses the whole component hierarchy to mark any changes required in the given component at a time.
- Reconciler vs Renderer => Reconciler does the work of computing which parts of the tree have changed. Renderer uses this info to update the rendered app.

8. What is React Fiber?

- React Fiber is the new reconciliation engine in React 16.
- The goal of React Fiber is to increase its suitability for areas like animation, layout, and gestures.
- Its headline feature is incremental rendering: the ability to split rendering work into chunks and spread it out over multiple frames.

9. Why do we need a key in React? When do we need keys in React?

- A key is a special string attribute you need to include when creating lists of elements. Keys help React identify which items have changed, are added, or are removed.
- Keys must be used when siblings are of different elements types.

10. Can we use index as key in react?

- A key is the only thing React uses to identify DOM elements. It is not recommended to use indexes for keys if the order of items may change. This can negatively impact performance and may cause issues with component state.

- But, nothing is better than anything. If we don't give a key, react by default assign id of that list item as it's key.
- NO key << INDEX as key <<<<<< Unique id as key from data

11. What are props in React? Ways to use props?

- Props (properties) passed in Component which are like arguments passed in a js function call and received by that function as parameters.
- Every parent component can pass some information to its child components by giving them props. Props are like HTML attributes, but you can pass any JavaScript value through them, including objects, arrays, and functions.
- Types of Props: array, bool, function, object
- Passing Props to Component - props are the only argument to your component. React component functions accept a single argument, a props object.

Ways to pass props to component	**Ways to receive the props in another component**
1. Add props to the JSX, just like you would with HTML attribute	All props are sent into a single props object
<Profile name = { "Mohit"} age={34} />	const Profile = (props) => { let name = props.name; let age = props.age; }
2. Similar to the way mentioned in 1	Props object can be destructed using {} to receive only the required props
<Profile name = { "Mohit"} age={34} />	const Profile = ({name, age}) => { }
3. Using spread syntax	And props objects destructed using {}
<Profile {...props} />	const Profile = ({name, age}) => {}

- However, props are immutable which means unchangeable. When a component needs to change its props (for example, in response to a user interaction or new data), it will have to "ask" its parent component to pass it different props—a new object! Its old props will then be cast aside, and eventually the JavaScript engine will reclaim the memory taken by them.

12. What is Config driven UI?

- Config-driven UI is one of the UI design patterns in which the UI is rendered based on the configuration parameter sent by the server (backend).
- This is one of the popular patterns used in the industry now.