Class 4:Assignment

▼ 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)`

Anything you can do with JSX can also be done with plain javaScript

▼ Is ES6 mandatory for React?

No, ES6 is not mandatory for React but it is highly recommended.

The latest projects being created in React are using ES6.

▼ `{TitleComponent}` vs `{<TitleComponent />}` vs `{<TitleComponent> </TitleComponent>}`

`{ TitleComponent }`
- This value in JSX is considered a JSX expression or variable. If no such variable is present, no output will be shown in the browser.

`{ <TitleComponent /> }` - This value in JSX is meant for rendering a component (i.e) a function that returns JSX. This is a 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> }`.

▼ How can I write comments inside JSX?

While using `{ }` when can write comments as used in JavaScript.

```
{
// single line comment
/* multi-line
   comment
*/
}
```

▼ What is `<React.Fragment> </React.Fragment>` and `<></>`?

The `React.Fragment` component lets you return multiple elements in a `render()` method without creating an additional DOM element. You can also use it with the shorthand `<></>` syntax

The only difference between them is that the shorthand version does not support the key attribute.

```
render() {
  return (
    <>
      <h1>Some text</h1>
      <h2>A heading</h2>
    </>
  );
}
```

▼ What is Virtual DOM?

The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM.

▼ What is Reconciliation in React?

- React stores a copy of the actual DOM which is called `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 with Real DOM. It finds out the changed nodes and updates only the changed nodes in Real DOM leaving the rest nodes as it is. This process is called Reconciliation.

▼ What is React Fiber?

- React Fiber is a completely backward-compatible rewrite of the old reconciler. This new reconciliation algorithm from React is called Fiber Reconciler.
- The main goals of the Fiber reconciler are incremental rendering, better or smoother rendering of UI animations and gestures, and responsiveness of the user interactions.
- The reconciler also allows you to divide the work into multiple chunks and divide the rendering work over multiple frames. It also adds the ability to define the priority for each unit of work and pause, reuse, and abort the work.

▼ Why do we need keys in React?

Keys help React identify which items have changed, are added, or are removed.

The best way to pick a key is to use a string that uniquely identifies a list item among its siblings. Most often you would use IDs from your data as keys:

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li key={number.toString()}>
    {number}
  </li>
);
```

▼ Can we use an index as a key in React?

A *key* is the only thing React uses to identify DOM elements.

What happens if you push an item to the list or remove something in the middle? If the *key*
 is the same as before React assumes that the DOM element represents the same component as before. But that is no longer true.

▼ What is a prop 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 a child component(s)).

They are useful when you want the flow of data in your app to be dynamic.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));
const element = <Welcome name="Sara" />;
root.render(element);
```

▼ What is a 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 pattern used in the industry now.

▼