React Repo for FSD-C-WE-T-B23

Steps to initialize the project as a Git Repo

1. **Initialize Git**: Open your terminal and navigate to the project directory. Run:

```
git init
```

2. Visit GitHub and create a new repository. Do not initialize it with a README, .gitignore, or license.

3. **Add Remote Origin**: After creating the repository, copy the remote URL and run:

```
git remote add origin <your-repo-url>
```

4. **Rename the default branch**: If your Git version is 2.28 or later, you can set the default branch name to `main` by running:

```
git branch -M main
```

5. **Add Files**: Add all files to the staging area:

```
git add .
```

6. **Commit Changes**: Commit the changes with a message:

```
git commit -m "Initial commit"
```

7. **Push to GitHub**: Finally, push your changes to the remote repository:

```
git push -u origin main
```

Component

- Component are the building blocks of React applications.
- They are reusable pieces of code that can be composed to create complex UIs.

JSX

- JavaScript XML (JSX) is a syntax extension for JavaScript that allows you to write HTML-like code within JavaScript.

Props Drilling

- Props drilling refers to the process of passing data from a parent component to a deeply nested child component through multiple layers of components.

ComponentA (data)

- return ComponentB (data) - return ComponentC (data) - return ComponentD (data)

Disadvantages of Props Drilling:

- It can make the code harder to read and maintain.
- It can lead to unnecessary re-renders of components that do not need the data.
- The data is passed through multiple layers of components, which might not need it, leading to performance issues.

Solution to Props Drilling

- Use Context API.

React Components

- We have two types of components in React:

1. Class Components
    - uses class syntax.
    - it is a legacy way of creating components.
    - Stateful components [State -> Component's Memory]
    - They can hold and manage their own state.
    - They have lifecycle methods that allow you to hook into different stages of a component's life (e.g., mounting, updating, unmounting).
    - They are more verbose and require more boilerplate code.
2. Functional Components
    - uses function syntax.
    - it was available since the beginning of React but became more popular in the year of 2019 with the introduction of Hooks.
    - Stateless components
    - They do not have their own state.
    - They do not have lifecycle methods.
    - They are simpler and easier to read.
    - They are more performant than class components.
    - Despite they are performant, they are not stateful and do not have lifecycle methods and hence they are not suitable.

    - Then, in 2019, React introduced Hooks, which allow functional components to have state and lifecycle methods.

Hooks

- Hooks are functions that allow you to use state and other React features in functional components.
- They were introduced in React 16.8.
- All the hooks start with the word "use".
- The most commonly used hooks are:
    - useState: To manage state in functional components.
    - useEffect: To perform side effects in functional components (similar to lifecycle methods in class components).

useRef

- We can use useRef in two ways:
    1. To access DOM elements directly.
    2. To store mutable values that do not cause re-renders when changed. But the value is persistent across renders.

useReducer

- useReducer is a hook that is used for managing complex state logic in functional components.
- It is an alternative to useState and is particularly useful when the state logic involves multiple sub-values.

useMemo

- useMemo is a hook that is used to optimize performance by memoizing expensive calculations.
- It is an uncommon hook and is used to avoid unnecessary re-computations of values that are expensive to calculate.

useEffect

- useEffect is a hook that allows you to perform side effects in functional components.

Note:

- hooks cannot be used inside class components.
- hooks are just functions that can be used inside functional components to bring in a feature that was previously only available in class components.
- useEffect hook is used to bring in lifecycle methods to functional components.
- life cycle methods: Let's say I need some functions to run:
    - when the component is mounted,
    - When the component is updated or re-rendered,
    - When the component is unmounted or removed from the DOM.
- Side effects are operations that runs outside the scope of the component, such as fetching data, subscribing to events, or manipulating the DOM.