**Batch -** T5

**Practical No. -** 5

**Title –** Study and implementation of ReactJs

**Student Name -** Sharaneshwar Bharat Punjal

**Student PRN -** 23520011

**Perform following problem statements for DOM using Javascript**

1. **What is React and what problem does it solve?**

React is a JavaScript library for building user interfaces, particularly single-page applications where data updates dynamically. It allows developers to create large web applications that can change data, without reloading the page. React solves problems related to performance and code maintainability by breaking down the UI into reusable components.

1. **What are React components and how are they used?**

React components are the building blocks of a React application. They are reusable, isolated pieces of code that represent parts of the UI. Components can be functional or class-based, and they can accept inputs (props) and manage their own state.

1. **What is JSX in React?**

JSX (JavaScript XML) is a syntax extension for JavaScript that looks similar to HTML. It is used in React to describe what the UI should look like. JSX allows you to write HTML-like code directly within JavaScript, making it easier to create and understand the UI structure.

1. **What are props in React and how do they differ from state?**

Props (short for "properties") are inputs passed to components from their parent components. They are immutable, meaning they cannot be changed by the component receiving them. Props are used to pass data and event handlers to child components.

State, on the other hand, is a special object managed within the component itself. It holds data that may change over time, and any change to the state triggers a re-render of the component. Unlike props, state is mutable and is used to keep track of dynamic data.

1. **What is state in React and how does it work?**

State is a built-in object in React components that holds data that can change over the lifetime of the component. When the state changes, React automatically re-renders the component to reflect the new data. State is managed internally by the component, and changes to the state are done using the setState method in class components or the useState hook in functional components.

1. **What are React lifecycle methods, and why are they important?**

React lifecycle methods are special methods that get called at different stages of a component's lifecycle, such as when a component is being mounted to the DOM, updated, or unmounted. These methods allow developers to perform actions like fetching data, updating the UI, or cleaning up resources. Common lifecycle methods include componentDidMount, componentDidUpdate, and componentWillUnmount.

1. **Elaborate following with respect to ReactJs**

Event Handling - In React, events are handled similarly to DOM events, but with some differences in syntax. For example, instead of onclick, you use onClick, and the event handler is passed as a function reference.

Conditional Rendering - Conditional rendering in React allows components to render different UI elements or components based on certain conditions. This can be done using JavaScript operators like if, ternary, or logical &&.

Lists and Keys - When rendering lists of elements in React, each element must have a unique key prop. Keys help React identify which items have changed, are added, or are removed, thus optimizing the rendering process.

Forms - React handles forms by controlling the form elements' values through state. This is known as controlled components. The form inputs are tied to the state, and any change in the input reflects in the state.

Hooks - Hooks are functions that let you use state and other React features in functional components. Common hooks include useState for state management, useEffect for side effects, and useContext for using context.

React Router - React Router is a library used to manage navigation and routing in React applications. It allows you to define routes in your app and navigate between them without refreshing the page.

State Management - In larger applications, managing state across multiple components can be challenging. React offers solutions like useReducer, Context API, or third-party libraries like Redux to manage global state.

React Context API - The Context API allows you to create global variables that can be passed around in your React application, avoiding the need to pass props through multiple layers of components.

1. **How can you optimize the performance of a React application?**

Avoid unnecessary re-renders: Use React.memo or PureComponent to prevent re-rendering of components that don’t need to update.

Use useCallback and useMemo: These hooks help prevent the creation of new functions and objects on every render, reducing the number of renders.

Lazy loading: Use React.lazy and Suspense to load components only when they are needed.

Code splitting: Split your code into smaller bundles that can be loaded on-demand.

Avoid anonymous functions in render: Passing anonymous functions directly in JSX can lead to unnecessary re-renders.

Optimize images: Use optimized image formats and lazy-load images to improve load times.

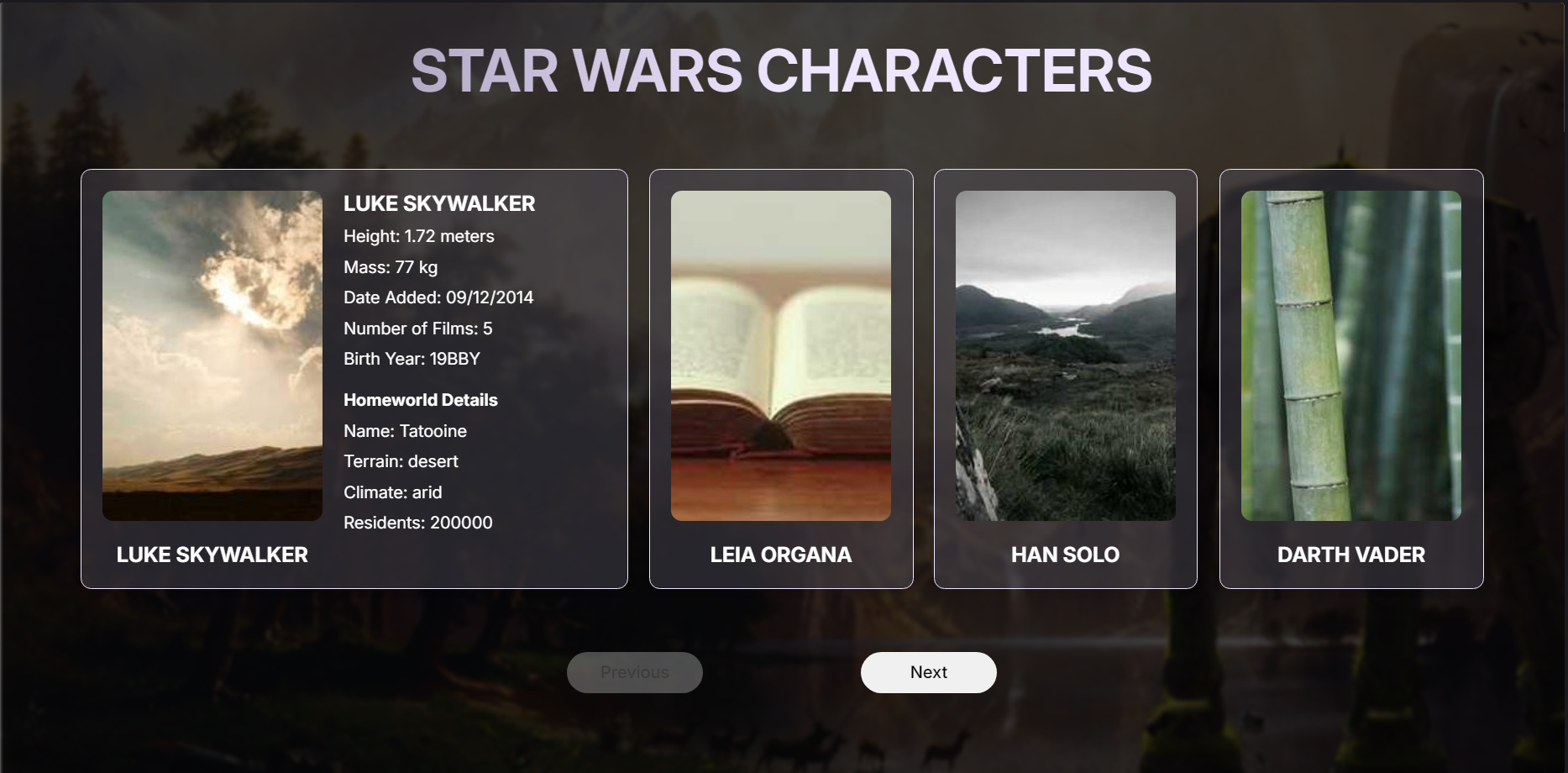
Use production build: Ensure you use the production build of React, which is optimized for performance.

**Problem Statement 1: Star Wars Character App**

(In this problem statement, example of Star Wars is given, you may choose any characters from the series of the movie like Harry Potter, etc. Every group in a batch will have different characters.)

* Using a public API, display a list of all Star Wars characters using the endpoint “/people”. The API has paging, so the developer must also implement pagination. Also, a simple loader for fetching/prefetching data as well as handling the error state (i.e., if the API server is down).
* For every user, we’d like to display a card with the name of each character along with a random picture for each character (see Picsum photos for random picture inspiration). Each character card should be colored based on their species and have some kind of animation when the user hovers over the card. When we click on a character’s card, more information should appear in a modal about the character.
* In the character details modal, we’d like to display information about the person: name as the header of the modal, height displayed in meters, mass in kg, date person was added to the API (in dd-MM-yyyy format), number of films the person appears in and their birth year. We should also fetch information about the person’s home world and display its name, terrain, climate, and number of residents.

**Screenshots –**

****

****