Task 4: Implementing Redux for State Management in a Profile Website

Objective:

Explore and master Redux for state management, including setting up Redux, creating reducers and actions, handling asynchronous operations with Redux Thunk, and connecting Redux to React components in a user profile website.

Pre-requisites:

- Basic understanding of HTML, CSS, and JavaScript
- Familiarity with a code editor like Visual Studio Code
- Basic knowledge of React and Redux

Concepts Covered:

- Setting Up Redux
- Creating Reducers and Actions
- Implementing Redux Thunk for Asynchronous Actions
- Connecting Redux to React Components

Concepts:

1. **Setting Up Redux:**

   Install Redux and React-Redux in your project. Create a basic Redux store and connect it to your React app.

```
npm install redux react-redux redux-thunk
```

```
// src/store/index.js
import { createStore, applyMiddleware } from 'redux';
import { Provider } from 'react-redux';
import thunk from 'redux-thunk';
import rootReducer from '../reducers';

const store = createStore(rootReducer, applyMiddleware(thunk));

const StoreProvider = ({ children }) => (
    <Provider store={store}>
        {children}
    </Provider>
);

export default StoreProvider;

// src/index.js
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import StoreProvider from './store';
```

```
ReactDOM.render(
    <StoreProvider>
        <App />
    </StoreProvider>,
    document.getElementById('root')
);
```

2. **Creating Reducers and Actions:**

   Create a reducer and some actions for managing the user profiles (like adding a profile, removing a profile).

```
// src/reducers/profileReducer.js
const initialState = {
    profiles: []
};

const profileReducer = (state = initialState, action) => {
    switch (action.type) {
        case 'ADD_PROFILE':
            return {
                ...state,
                profiles: [...state.profiles, action.payload]
            };
        case 'REMOVE_PROFILE':
            return {
                ...state,
                profiles: state.profiles.filter(profile => profile.id !==
action.payload)
            };
        default:
            return state;
    }
};

export default profileReducer;

// src/reducers/index.js
import { combineReducers } from 'redux';
import profileReducer from './profileReducer';

const rootReducer = combineReducers({
    profiles: profileReducer
});

export default rootReducer;

// src/actions/profileActions.js
export const addProfile = profile => ({
    type: 'ADD_PROFILE',
    payload: profile
});

export const removeProfile = profileId => ({
    type: 'REMOVE_PROFILE',
    payload: profileId
});
```

3. **Implementing Redux Thunk for Asynchronous Actions:**

   Use Redux Thunk middleware to handle asynchronous actions, such as fetching profile data from an API.

```
// src/actions/profileActions.js
import axios from 'axios';

export const fetchProfiles = () => {
    return async dispatch => {
        try {
            const response = await axios.get('/path/to/profiles.json');
            dispatch({ type: 'SET_PROFILES', payload: response.data });
        } catch (error) {
            console.error('Error fetching profiles:', error);
        }
    };
};

export const setProfiles = profiles => ({
    type: 'SET_PROFILES',
    payload: profiles
});
```

```
// src/reducers/profileReducer.js
const initialState = {
    profiles: []
};

const profileReducer = (state = initialState, action) => {
    switch (action.type) {
        case 'ADD_PROFILE':
            return {
                ...state,
                profiles: [...state.profiles, action.payload]
            };
        case 'REMOVE_PROFILE':
            return {
                ...state,
                profiles: state.profiles.filter(profile => profile.id !==
action.payload)
```

```
            };
        case 'SET_PROFILES':
            return {
                ...state,
                profiles: action.payload
            };
        default:
            return state;
    }
};

export default profileReducer;
```

4. **Connecting Redux to React Components:**

   Use the `useSelector` and `useDispatch` hooks from React-Redux to connect the Redux state to your components, such as displaying a list of profiles and adding a profile.

```
// src/components/ProfileList.js
import React, { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { fetchProfiles, addProfile, removeProfile } from '../actions/
profileActions';

const ProfileList = () => {
    const profiles = useSelector(state => state.profiles.profiles);
    const dispatch = useDispatch();

    useEffect(() => {
        dispatch(fetchProfiles());
    }, [dispatch]);

    const handleAddProfile = () => {
        const newProfile = { id: Date.now(), name: 'New User', occupation:
'Developer' };
        dispatch(addProfile(newProfile));
    };

    const handleRemoveProfile = (profileId) => {
        dispatch(removeProfile(profileId));
    };

    return (
        <div>
            <h2>Profile List</h2>
            <button onClick={handleAddProfile}>Add Profile</button>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>
                        {profile.name} - {profile.occupation}
                        <button onClick={() => handleRemoveProfile(profile.id)}
>Remove</button>
                    </li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

Setup:

1. **Install Visual Studio Code (VS Code):**

   Download and install VS Code from [Visual Studio Code].

2. **React Developer Tools:**

   Install the React Developer Tools browser extension for [Chrome] or [Firefox].

3. **Git:**

   Install Git for version control. You can download it from [Git].

4. **Node.js and npm:**

   Ensure Node.js and npm are installed on your machine. You can download and install them from [Node.js].

5. **Setting Up the React Project:**
   - Open your terminal.
   - Navigate to the directory where you want to create your project.
   - Run the following command to create a new React project:

   ```
   npx create-react-app profile-website
   ```

6. **Installing Additional Dependencies:**
   - Navigate to the project directory:

   ```
   cd profile-website
   ```

   - Install Redux, React-Redux, and Redux Thunk:

   ```
   npm install redux react-redux redux-thunk
   ```

Tasks:

Part 1: Introduction to Redux Basics (30 minutes)

1. **Setting Up Redux (15 minutes):**

- Install Redux and React-Redux in your project. Create a basic Redux store and connect it to your React app.
- Example:

```
npm install redux react-redux redux-thunk
```

```
// src/store/index.js
import { createStore, applyMiddleware } from 'redux';
import { Provider } from 'react-redux';
import thunk from 'redux-thunk';
import rootReducer from '../reducers';

const store = createStore(rootReducer, applyMiddleware(thunk));

const StoreProvider = ({ children }) => (
    <Provider store={store}>
        {children}
    </Provider>
);

export default StoreProvider;

// src/index.js
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import StoreProvider from './store';

ReactDOM.render(
    <StoreProvider>
        <App />
    </StoreProvider>,
    document.getElementById('root')
);
```

- Goal: Learn the initial setup process for using Redux in a React application.

2. **Creating Reducers and Actions (15 minutes):**
   - Create a reducer and some actions for managing the user profiles (like adding a profile, removing a profile).
   - Example:

```
// src/reducers/profileReducer.js
const initialState = {
    profiles: []
};

const profileReducer = (state = initialState, action) => {
    switch (action.type) {
        case 'ADD_PROFILE':
            return {
                ...state,
                profiles: [...state.profiles, action.payload]
            };
        case 'REMOVE_PROFILE':
            return {
                ...state,
                profiles: state.profiles.filter(profile => profile.id !==
action.payload)
            };
        case 'SET_PROFILES':
            return {
                ...state,
                profiles: action.payload
            };
        default:
            return state;
    }
};

export default profileReducer;

// src/reducers/index.js
import { combineReducers } from 'redux';
import profileReducer from './profileReducer';

const rootReducer = combineReducers({
    profiles: profileReducer
});

export default rootReducer;

// src/actions/profileActions.js
export const addProfile = profile => ({


    type: 'ADD_PROFILE',
    payload: profile
});

export const removeProfile = profileId => ({
    type: 'REMOVE_PROFILE',
    payload: profileId
});

export const setProfiles = profiles => ({
    type: 'SET_PROFILES',
    payload: profiles
});
```

- Goal: Understand the concept of reducers and actions in Redux for managing application state.

Part 2: Advanced Redux Concepts (30 minutes)

1. **Implementing Redux Thunk for Asynchronous Actions (15 minutes):**
    - Use Redux Thunk middleware to handle asynchronous actions, such as fetching profile data from an API.
    - Example:

```
// src/actions/profileActions.js
import axios from 'axios';

export const fetchProfiles = () => {
    return async dispatch => {
        try {
            const response = await axios.get('/path/to/profiles.json');
            dispatch({ type: 'SET_PROFILES', payload: response.data });
        } catch (error) {
            console.error('Error fetching profiles:', error);
        }
    };
};
```

    - Goal: Grasp how to manage asynchronous operations in Redux.
2. **Connecting Redux to React Components (15 minutes):**
    - Use the `useSelector` and `useDispatch` hooks from React-Redux to connect the Redux state to your components, such as displaying a list of profiles and adding a profile.
    - Example:

```
// src/components/ProfileList.js
import React, { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { fetchProfiles, addProfile, removeProfile } from '../actions/
profileActions';

const ProfileList = () => {
    const profiles = useSelector(state => state.profiles.profiles);
    const dispatch = useDispatch();
    useEffect(() => {
        dispatch(fetchProfiles());
    }, [dispatch]);

    const handleAddProfile = () => {
        const newProfile = { id: Date.now(), name: 'New User', occupation:
'Developer' };
        dispatch(addProfile(newProfile));
    };

    const handleRemoveProfile = (profileId) => {
        dispatch(removeProfile(profileId));
    };

    return (
        <div>
            <h2>Profile List</h2>
            <button onClick={handleAddProfile}>Add Profile</button>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>
                        {profile.name} - {profile.occupation}
                        <button onClick={() =>
handleRemoveProfile(profile.id)}>Remove</button>
                    </li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

- Goal: Learn to integrate Redux state into React components and dispatch actions based on user interactions.

Instructions:

1. Write the required code in the appropriate files in the `src/components` , `src/actions` , `src/reducers` , and `src/store` directories.
2. Open the terminal and navigate to the project directory.
3. Run the React project using:

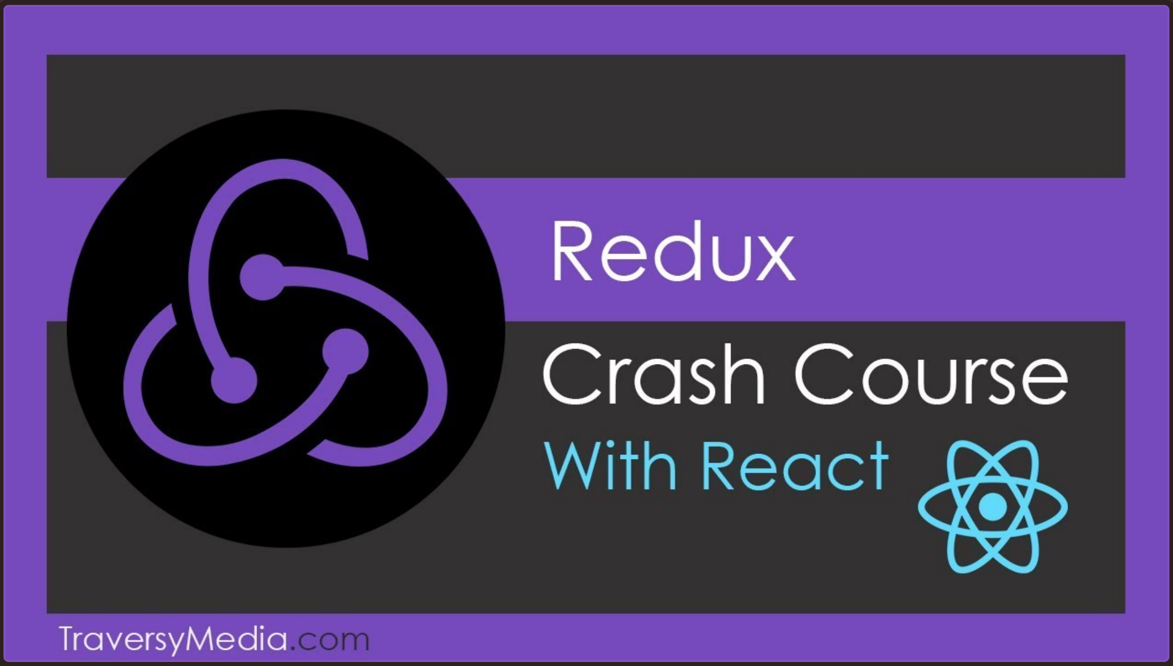
```
npm start
```

4. Open the project in a web browser to ensure the code displays correctly.
5. Use the browser's developer tools and React Developer Tools to debug and inspect the elements.

Resources:

- Redux - A JS library for predictable and maintainable global state managem...
  A JS library for predictable and maintainable global state management
  https://redux.js.org/
- React Redux | React Redux
  Official React bindings for Redux
  https://react-redux.js.org/
- Github
  https://github.com/reduxjs/redux-thunk
- Getting Started – React
  A JavaScript library for building user interfaces
  https://reactjs.org/docs/getting-started.html
- Create React App
  Set up a modern web app by running one command.
  https://create-react-app.dev/
- Documentation for Visual Studio Code
  Find out how to set-up and get the most from Visual Studio Code. Optimized for building and d...
  https://code.visualstudio.com/docs

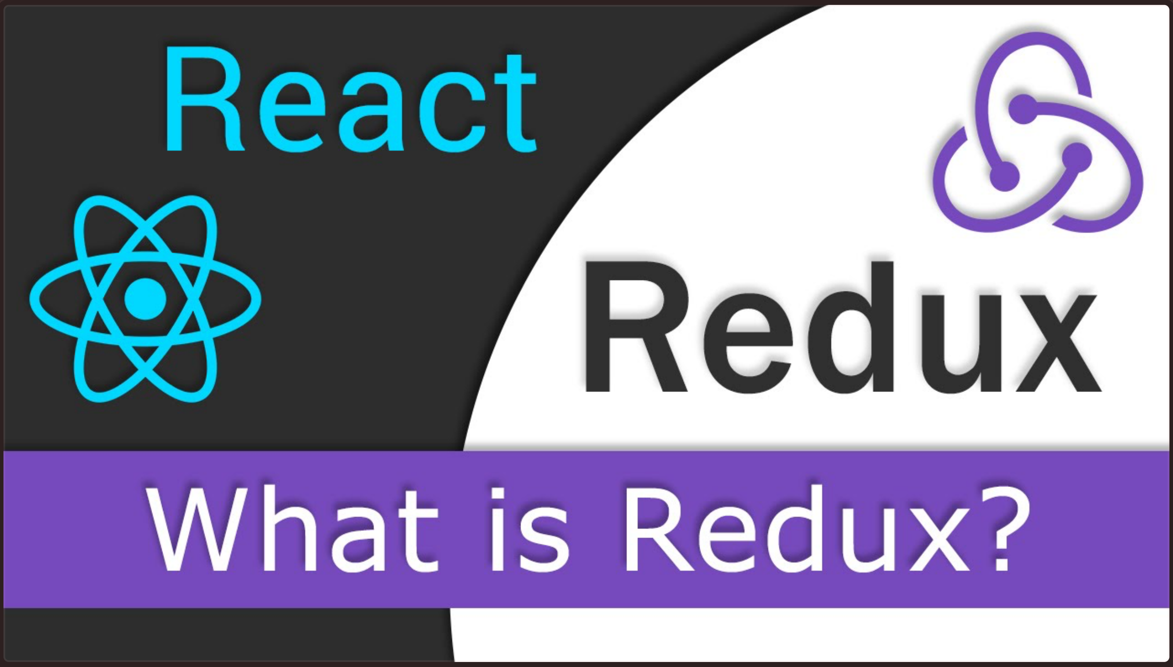
Videos:

- 

  Traversy Media 1:09hours 817,141 Views 12,883 Likes

-

  Codevolution 15:01min 918,321 Views 10,624 Likes

- 

  thenewboston 4:53 min 346,797 Views 2,002 Likes

-

  CODERS NEVER QUIT 12:40 min 22,606 Views 222 Likes

GitHub Instructions:

1. **Open in Visual Studio Code:**

   After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, VSCode will open the repository directly. If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. **Open the Terminal in VSCode:**

In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

3. **Complete the Task:**

   In VSCode, write your solution in the appropriate files in the `src/components` , `src/actions` , `src/reducers` , and `src/store` directories.

4. **Run and Test Your Code:**

   Open your terminal, navigate to your project directory, and run:

   ```
   npm start
   ```

5. **Commit Your Changes:**

   In the VSCode terminal, add your changes to git:

   ```
   git add src/components/* src/actions/* src/reducers/* src/store/*
   ```

   Commit your changes with a meaningful message:

   ```
   git commit -m "Completed task 24"
   ```

6. **Push Your Changes to Your Repository:**

   Push your changes to your forked repository:

   ```
   git push origin main
   ```

7. **Create a Pull Request:**

   Go to your repository on GitHub.

   Click on the "Pull Requests" tab.

   Click the "New Pull Request" button.

   Ensure the base repository is the original template repository and the base branch is `main` .

   Ensure the head repository is your forked repository and the compare branch is `main` .

   Click "Create Pull Request".

   Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing src/components/* src/actions/* src/reducers/*
src/store/*

# Navigate to the project directory
cd profile-website

# Run your code
npm start

# Add, commit, and push your changes
git add src/components/* src/actions/* src/reducers/* src/store/*
git commit -m "Completed task 4"
git push origin main

# Create a pull request on GitHub
```