Redux

Wednesday, October 30, 2019 1:46 PM

https://viblo.asia/p/chuong-2-ung-dung-redux-dau-tien-cua-ban-07LKXA8JZV4

1) Install Redux:
   - npm install redux
   - npm install react-redux

Goal:
- Make a change on lv2 children
- Lv1 children and parent will be reload
- Store value in Redux store state
- Updating value in Redux store state
- Catch the change and reload the component as needed

Flow:
- We have 3 parts of action for this:
1) The storage that holding all the state with default value
2) The action part that will make a change on the store
3) The subscribe will listen on the change and make the component that involve reload

PART I:
Init the redux store and get the default value
Step 1: Create sample store and the listener for the store's changing

Create listener:
/reducers/reducers.js

```
export default (state) => {
    return state;
};
```

(*) The reducer will listen on the change of store and return the new value for subscriber.
(*) In the code above, it will return the state no matter what action is dispatched

Step 2: Create Store:

```
//Create Redux Store
import { createStore } from 'redux';
//Import reducer
import reducers from '../reducers/reducers';
//set default value
const initialState = { tech : "React"};
export const store = createStore(reducers, initialState
);
```

(*) When the first time access the store will return it default value, so we set this state via initialState

Step 3: Access to Store and get the param
We can access to store by simply using:

```
//import store
import { store } from './stores/stores';
```

and get the state value by:

```
store.getState().tech
```

PART II
Create and dispatch the action to make a change on Redux store
Step 1: Define the action type
/actions/actionType.js

```
export const SET_TECHNOLOGY = "SET_TECHNOLOGY";
export const EMPTY_TECHNOLOGY = "EMPTY_TECHNOLOGY";
```

(*) Whenever we make a new change it will be dispatch an event/ action to the Redux store and update.
(*) Action type is the name of event/ action

Step 2: Define action to handle dispatch request base on action type: /actions/actions.js and send the requested param to reducers

```
import {
    SET_TECHNOLOGY,
    EMPTY_TECHNOLOGY
} from './actionType';
export function action(actionType, state) {
    switch (actionType) {
        case SET_TECHNOLOGY:
            return {
                type: actionType,
                tech: state
            };
        case EMPTY_TECHNOLOGY:
            return {
                type: actionType,
                tech: null
            }
        default:
            return state;
    }
}
```

Step 3: The reducer will listen on an action, update the state value and response it for the Front End.
To do this, we need to modify a little bit:

```
import {
    SET_TECHNOLOGY,
    EMPTY_TECHNOLOGY
} from "../actions/actionType";
export default (state, action) => {
    switch (action.type) {
        case SET_TECHNOLOGY:
        case EMPTY_TECHNOLOGY:
            return {
                ...state,
                tech: action.tech
            };
        default:
            return state;
    }
};
//Define in reducer the action and the way it inform
to Front end
//Should be pure function, no API, no Update external
 API...
```

PART III
Add Subscriber to listen on the response from reducers, it will update the changed value that we get from store.getStore().[property_name] and notice the component relative to reload.

We will binding it on /index.js for whole component

```
//import store for subscribe change
import { store } from "./stores/stores";
const render = function () {
    ReactDOM.render(
        <App />,
        document.getElementById("root")
    );
}
render();
//Using store subscribe
store.subscribe(render);
```

PART IV
Dispatch an event:

Import Action:

```
//import action
import { action } from "./actions/actions";
```

Import Action Type:

```
//import actionType
import {
    SET_TECHNOLOGY,
    EMPTY_TECHNOLOGY
} from "./actions/actionType";
```

PART V
Redirect after setting Redux:
Using React Router DOM History

```
//Import BrowserHistory
import { useHistory } from 'react-router-dom';
```

Create Button to handle Login Redirect

```
function LoginButton() {
    let history = useHistory();
    function handleClick() {
        store.dispatch(actionLoggedIn(HANDLE_LOGGED_IN_STATUS, true));
        history.push("/");
    }
    return (
        <button type="button" className="btn btn-lg btn-primary btn-
block" onClick={handleClick}>
            Login
        </button>
    );
}
```

Using Login Button to render Redirect

```
<LoginButton />
```