Connecting React Components to Redux

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Connecting React Components to Redux

Connecting React Components to Redux

React Redux provides several methods that allow you to easily connect React Components to Redux.

Provider

The Provider component is used to pass the Redux store to your React Components. To use the Provider component, pass in your store as an attribute and then set your top level react component as a child component to the Provider component. You need to do this, otherwise the store won't be accessible to your React components.

TODO EDIT THIS

```
import React from 'react'
import { render } from 'react-dom'
import { Provider } from 'react-redux'
import { createStore } from 'redux'
import todoApp from './reducers'
import App from './components/App'

const store = createStore(todoApp)

render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
)
```

connect()

The `connect()` method is used to pass in store state and action dispatchers as props to your React components. The `connect()` method should be used inside `container` components and not `presentational components`.

You can import the `connect()` method like this:

```
import {connect} from 'react-redux`
```

To use the `connect()` method, use the following syntax:

```
connect(
  mapStateToProps,
  mapDispatchToProps
)(PresentationalComponent)
```

Why are there so many parenthesis? Its because the `connect(mapStateToProps,mapDispatchToProps)` portion returns a method, which is then executed with the PresentationalComponent as an argument.

Its the same as doing the following:

```
const functionToConnectPresentationalComponentWithProps = connect(ma
functionToConnectPresentationalComponentWithProps(PresentationalComp
```

mapStateToProps

The first argument is `mapStateToProps` which is a function that specifies what part the store state should be passed in as props.

```
const mapStateToProps = (state) => {
  return {
    propName1: state.stateAttributeName1,
    propName2: state.stateAttributeName2
  }
}
```

You just return an object that specifies the names for your props and you map in the state attributes that you want accordingly.

mapDispatchToProps

The second argument is `mapDispatchToProps` which is a function that species which action dispatchers should be passed in as props.

```
const mapDispatchToProps = (dispatch) => {
  return {
    propName1: (arg) => { //props name is usually onClick,onChange o
      dispatch(actionCreator1(arg))
    }
    propName2: (arg) => {
      dispatch(actionCreator2(arg))
    }
  }
}
```

If you don't need to add any action dispatchers, just don't provide a second argument to `connect()`.

PresentationalComponent

This argument is used to indicate which React component you want to attach new props to.

Example

Container component using connect():

```
//takes items from store state and passes it as a props to ItemsList
import { connect } from 'react-redux'
import ItemList from './components/ItemList'

const mapStateToProps = (state) => {
  return {
    items: state
  }
}

const mapDispatchToProps = (dispatch) => {
  return {
    onDelete: id => {
      dispatch(deleteItem(id))
    }
  }
}

const ItemsListContainer = connect(mapStateToProps,mapDispatchToProp
```

ItemsList for reference:

```
import React from 'react'
import { render } from 'react-dom'

const Item = (props) => {
  return (
    <div>
      <div>Item : {props.name} | Price: {props.price} </div>
      <button onClick = {() => props.onDelete(props.index)}>Delete</
    </div>
  )
}
const ItemsList = (props) => {
  return (
    <div>
      {
        props.items.map( (item, index) => {
          return (
            <Item onDelete = {props.onDelete} index = {index} name =
          )
        })
      }
    </div>
  )
}
```

Try out this live example on CodeSandbox: https://codesandbox.io/s/mj19l4m059