What subscribe to only the pieces of data in the store that are relevant to their role?	The react components
From where do you dispatch actions, which trigger store updates?	Directly from React components
What is a view library that you provide with data, then it renders the view in an efficient, predictable way?	react
What is a state management framework that you can use to simplify the management of your app state?	redux
In what do you typically create a single redux store that managers the state of your entire app?	React redux app
What subscribe to only the pieces of data in the store that are relevant to their role?	The react components
When you have a complex app, what is better kept in a single location with Redux rather than inside react components which manage their own state locally?	state
What may have local state specific only to them as a n exception?	Individual components
What needs to be used since redux is not designed to work with React out of the box?	React-redux package
What provides a away for you to pass redux state and dispatch to your react components as props?	React-redux package

	`class DisplayMessages extends React.Component {`
	` // change code below this line`
	`  //initialize state`
	`  constructor(props) {`
	`    super(props);`
	`    this.state = {`
	`     input: "",`
	`     messages: []`

```
    };
  }
 // change code above this line
 render() {
   return <div />
 }
};
```

```
class DisplayMessages extends
React.Component {
 constructor(props) {
   super(props);
   this.state = {
     input: '',
     messages: []
   }
    this.handleChange =
this.handleChange.bind(this);
    this.submitMessage =
this.submitMessage.bind(this);
 }
 // add handleChange() and
submitMessage() methods here

 handleChange(e) {
   this.setState({
     input: e.target.value
   });
 }

 submitMessage() {
   let
newList=[...this.state.messages,
this.state.input];

   this.setState({
     messages: newList,
    input: ''
  });
 }

 render() {
   return (
     <div>
       <h2>Type in a new Message:</h2>
       { /* render an input, button, and ul here */ }
       <input onChange={e => this.handleChange(e)} value={this.state.input}></input>
       <button onClick={this.submitMessage}>Submit</button>
       <ul>
       {this.state.messages.map(x => <li>{x}</li>)}
       </ul>
       { /* change code above this line */ }
     </div>
   );
 }
};
```

What needs to be moved from the React component into Redux?

The logic its performing locally in its state

First, define an action type 'ADD' and set it to a const `ADD`. Next, define an action creator `addMessage()`which creates the action to add a message. You'll need to pass a `message`to this action creator and include the message in the returned `action`.
Then create a reducer called

```
// define ADD, addMessage(), messageReducer(), and store here:
const ADD = 'ADD';

const addMessage = (message) => {
 return {
   type: ADD,
```

messageReducer()that handles the state for the messages. The initial state should equal an empty array. This reducer should add a message to the array of messages held in state, or return the current state. Finally, create your Redux store and pass it the reducer.
 	
The const ADDshould exist and hold a value equal to the string ADD
Passed
The action creator addMessageshould return an object with typeequal to ADDand message equal to the message that is passed in.
Passed
messageReducershould be a function.
Passed
The store should exist and have an initial state set to an empty array.
Passed
Dispatching addMessageagainst the store should immutably add a new message to the array of messages held in state.
Passed
The messageReducershould return the current state if called with any other actions.
	```
	   message: message
	 }
	}

	const messageReducer = (state = [],
	action) => {
	 switch(action.type) {
	   case ADD:
	     return [...state,
	action.message];
	   default:
	     return state;
	 }
	}

	const store =
	Redux.createStore(messageReducer);
	```

After creating a redux store to handle the messages array and creating an action for adding new messages, the next step is to provide what access to the Redux store and the actions it needs to dispatch updates?	react
What is provided by react redux to help accomplish this?	React-redux package
What does react redux provide with two key features, Provider and connect?	A small API
What is a wrapped component from react redux that wraps your react app?	Provider

What allows you to access the redux store and siatch functions throughout your component tree?	Provider
What takes two props, the Redux store and the child components fo your app?	Provider
What's this? <Provider store={store}> <App/> </Provider>	Definidn gthe Provider for an App component

The code editor now shows all your Redux and React code from the past several challenges. It includes the Redux store, actions, and the DisplayMessages component. The only new piece is the AppWrapper component at the bottom. Use this top level component to render the Provider from ReactRedux, and pass the Redux store as a prop. Then render the DisplayMessages component as a child. Once you are finished, you should see your React component rendered to the page.	`// Redux Code:`
	`const ADD = 'ADD';`
	
	`const addMessage = (message) => {`
	 `return {`
	   `type: ADD,`
	   `message`
	 `}`
	`};`
	
	`const messageReducer = (state = [], action) => {`
	 `switch (action.type) {`
	   `case ADD:`
	     `return [`
	       `...state,`
	       `action.message`
	     `];`
	   `default:`
	     `return state;`
	 `}`
	`};`
	
	`const store =`

```
Redux.createStore(messageReducer);

// React Code:

class DisplayMessages extends
React.Component {
 constructor(props) {
   super(props);
   this.state = {
     input: '',
     messages: []
   }
   this.handleChange =
this.handleChange.bind(this);
   this.submitMessage =
this.submitMessage.bind(this);
 }
 handleChange(event) {
   this.setState({
     input: event.target.value
   });
 }
 submitMessage() {
   const currentMessage =
this.state.input;
   this.setState({
     input: '',
     messages:
this.state.messages.concat(currentMess
age)
   });
 }
 render() {
   return (
     <div>
       <h2>Type in a new Message:</h2>
       <input
         value={this.state.input}

onChange={this.handleChange}/><br/>
       <button
onClick={this.submitMessage}>Submit</b
utton>
       <ul>
         {this.state.messages.map(
(message, idx) => {
             return (
                <li
key={idx}>{message}</li>
             )
           })
         }
       </ul>
     </div>
   );
 }
};

const Provider = ReactRedux.Provider;

class AppWrapper extends
React.Component {
 // render the Provider here
   render() {
       return (
           <Provider store={store}>
               <DisplayMessages/>
           </Provider>
       )
   }
 // change code above this line
};
```

What allows you to provide state and dispatch to your React components?	Provider component
What must be specified exactly?	State and actions you want
What has access only to the state it needs?	component
The above is accomplished using what two methods?	mapStateToProps() and mapDispatchToProps()

reate a function `mapStateToProps()`. This unction should take `state`as an argument, hen return an object which maps that state t pecific property names. These properties wil ecome accessible to your component via `rops`. Since this example keeps the entire tate of the app in a single array, you can pass hat entire state to your component. Create a roperty `messages`in the object that's being eturned, and set it to `state`.	`const state = [];`
	
	`// change code below this line`
	`const mapStateToProps = (state) => {`
	` return {`
	`   messages: state`
	` }`
	`}`

What function is used to provide specific action creators to your React components so they can dispatch actions against the Redux store?	mapDispatchToProps()
What returns an object that maps dispatch actions to property names, which becom component props?	mapDispatchToProps()
Instead of returning a piece of state, what's each preoperty returns a fucntion that calls dispatch with an actionc reator and any relevant action data?	mapDispatchToProps()

For example, you have a loginUser() action creator that takes a username as an action payload. The object returned from mapDispatchToProps() for this action creator would look something like:	`{`
	`  submitLoginUser: function(username) {`
	`    dispatch(loginUser(username));`
	`  }`
	`}`

The code editor provides an action creator called `addMessage()`. Write the function `mapDispatchToProps()`that takes `dispatch`as an argument, then returns an object. The object should have a property `submitNewMessage`set to the dispatch function, which takes a parameter for the new message to add when it dispatches `addMessage()`.	`const addMessage = (message) => { return { type: 'ADD', message: message } }; // change code below this line const mapDispatchToProps = (dispatch) => { return { submitNewMessage: function(newMessage) { dispatch(addMessage(newMessage)); } } }`
What method from React Redux can handle the task of mapping state and dispatch to the props of one of your React components?	The connect method
What takes two optional arguments: mapStateToProps() and mapDispatchToProps()?	Connect method
What are optional because you may have a component that only needs access to state but doesn't need to dispatch any actions, or vice versa?	mapStateToProps9) and mapDispatchToProps() arguments in the connect method
How to pass in functions as arguments and imemdiately call the result with our component?	connect(mapStateToProps, mapDispatchToProps)(MyComponent)
What must be passed if one argument is to be ommitted?	null
The code editor has the `mapStateToProps()`and	`const addMessage = (message) => { return {`

`mapDispatchToProps()`functions and a new React component called `Presentational`. Connect this component to Redux with the `connect`method from the `ReactRedux`global object, and call it immediately on the `Presentational`component. Assign the result to a new `const`called `ConnectedComponent`that represents the connected component. That's it, now you're connected to Redux! Try changing either of `connect`'s arguments to `null`and observe the test results.	
	```
	   type: 'ADD',
	   message: message
	 }
	};

	const mapStateToProps = (state) => {
	 return {
	   messages: state
	 }
	};

	const mapDispatchToProps = (dispatch)
	=> {
	 return {
	   submitNewMessage: (message) => {
	     dispatch(addMessage(message));
	   }
	 }
	};

	class Presentational extends
	React.Component {
	 constructor(props) {
	   super(props);
	 }
	 render() {
	   return <h3>This is a Presentational
	Component</h3>
	 }
	};

	const connect = ReactRedux.connect;
	// change code below this line
	const ConnectedComponent =
	connect(mapStateToProps,
	mapDispatchToProps)(Presentational);
	```

What generally refers to React components that are not directly connected to REdux, simply responsible for the presentation of UI?	Presentational
BY contrast, what are components that are connected to Redux and are responsible for dispatching actions o the store and often pass store state to child components as props?	containers

The code editor has all the code you've written in this section so far. The only change is that the React component is renamed to `Presentational`. Create a new component held in a constant called `Container`that uses `connect`to connect the `Presentational`component to Redux. Then, in the `AppWrapper`, render the React Redux `Provider`component. Pass `Provider`the Redux `store`as a prop and render `Container`as a child. Once everything is setup, you will see the messages app rendered to the page again.	`// Redux:`
	`const ADD = 'ADD';`
	
	`const addMessage = (message) => {`
	` return {`
	`   type: ADD,`
	`   message: message`
	` }`
	`};`
	
	`const messageReducer = (state = [],`
	`action) => {`
	` switch (action.type) {`
	`   case ADD:`
	`     return [`
	`       ...state,`
	`       action.message`
	`     ];`
	`   default:`
	`     return state;`
	` }`
	`};`
	
	`const store =`
	`Redux.createStore(messageReducer);`
	
	`// React:`
	`class Presentational extends`
	`React.Component {`

```
 constructor(props) {
   super(props);
   this.state = {
     input: '',
     messages: []
   }
   this.handleChange =
this.handleChange.bind(this);
   this.submitMessage =
this.submitMessage.bind(this);
 }
 handleChange(event) {
   this.setState({
     input: event.target.value
   });
 }
 submitMessage() {
   const currentMessage =
this.state.input;
   this.setState({
     input: '',
     messages:
this.state.messages.concat(currentMess
age)
   });
 }
 render() {
   return (
     <div>
       <h2>Type in a new Message:</h2>
       <input
         value={this.state.input}

onChange={this.handleChange}/><br/>
       <button
onClick={this.submitMessage}>Submit</b
utton>
       <ul>
         {this.state.messages.map(
(message, idx) => {
             return (
                <li
key={idx}>{message}</li>
             )
           })
         }
       </ul>
     </div>
   );
 }
};

// React-Redux:
const mapStateToProps = (state) => {
 return { messages: state }
};

const mapDispatchToProps = (dispatch)
=> {
 return {
   submitNewMessage: (newMessage) => {
      dispatch(addMessage(newMessage))
   }
 }
};

const Provider = ReactRedux.Provider;
const connect = ReactRedux.connect;

// define the Container component
here:
const Container =
```

	`connect(mapStateToProps,`
	`mapDispatchToProps)(Presentational);`
	
	`class AppWrapper extends`
	`React.Component {`
	` constructor(props) {`
	`   super(props);`
	` }`
	` render() {`
	`   // complete the return statement:`
	`   return (`
	`     <Provider store={store}>`
	`       <Container />`
	`     </Provider>`
	`   )`
	` }`
	`};`

Now that Redux is connected, what eneds to be extracted out of Presnetational component and into Redux?	State management
Currently hwere is the state being handled?	Locally within the Presentational component

This example also illustrates how a component may have local state: your component still tracks user input locally in its own state. You can see how Redux provides a useful state management framework on top of React.	`// Redux:`
	`const ADD = 'ADD';`
	
	`const addMessage = (message) => {`
	` return {`
	`   type: ADD,`
	`   message: message`
	` }`
	`};`
	
	`const messageReducer = (state = [],`
	`action) => {`
	` switch (action.type) {`
	`   case ADD:`

```
     return [
       ...state,
       action.message
     ];
   default:
     return state;
 }
};

const store =
Redux.createStore(messageReducer);

// React:
const Provider = ReactRedux.Provider;
const connect = ReactRedux.connect;

// Change code below this line
class Presentational extends
React.Component {
 constructor(props) {
   super(props);
   this.state = {
     input: ''
   }
   this.handleChange =
this.handleChange.bind(this);
   this.submitMessage =
this.submitMessage.bind(this);
 }
 handleChange(event) {
   this.setState({
     input: event.target.value
   });
 }
 submitMessage() {
```

```
this.props.submitNewMessage(this.state
.input);

   this.setState({
     input: ''
   });
 }
 render() {
   return (
     <div>
       <h2>Type in a new Message:</h2>
       <input
         value={this.state.input}

onChange={this.handleChange}/><br/>
       <button
onClick={this.submitMessage}>Submit</b
utton>
       <ul>
         {this.props.messages.map(
(message, idx) => {
             return (
                <li
key={idx}>{message}</li>
             )
           })
         }
       </ul>
     </div>
   );
 }
};
// Change code above this line

const mapStateToProps = (state) => {
 return {messages: state}
```

```
};

const mapDispatchToProps = (dispatch)
=> {
 return {
   submitNewMessage: (message) => {
     dispatch(addMessage(message))
   }
 }
};

const Container =
connect(mapStateToProps,
mapDispatchToProps)(Presentational);

class AppWrapper extends
React.Component {
 render() {
   return (
     <Provider store={store}>
       <Container/>
     </Provider>
   );
 }
};
```

`import`statements (these pull in all of the dependencies that have been provided for you in the challenges)

```
// import React from 'react'
// import ReactDOM from 'react-dom'
// import { Provider, connect } from
'react-redux'
// import { createStore,
combineReducers, applyMiddleware }
from 'redux'
// import thunk from 'redux-thunk'

// import rootReducer from
'./redux/reducers'
```

	`// import App from './components/App'`
	
	`// const store = createStore(`
	`//   rootReducer,`
	`//   applyMiddleware(thunk)`
	`// );`
	
	`// ReactDOM.render(`
	`//   <Provider store={store}>`
	`//     <App/>`
	`//   </Provider>,`
	`//   document.getElementById('root')`
	`// );`
	
	`// change code below this line`
	`console.log('Now I know React and Redux!');`

