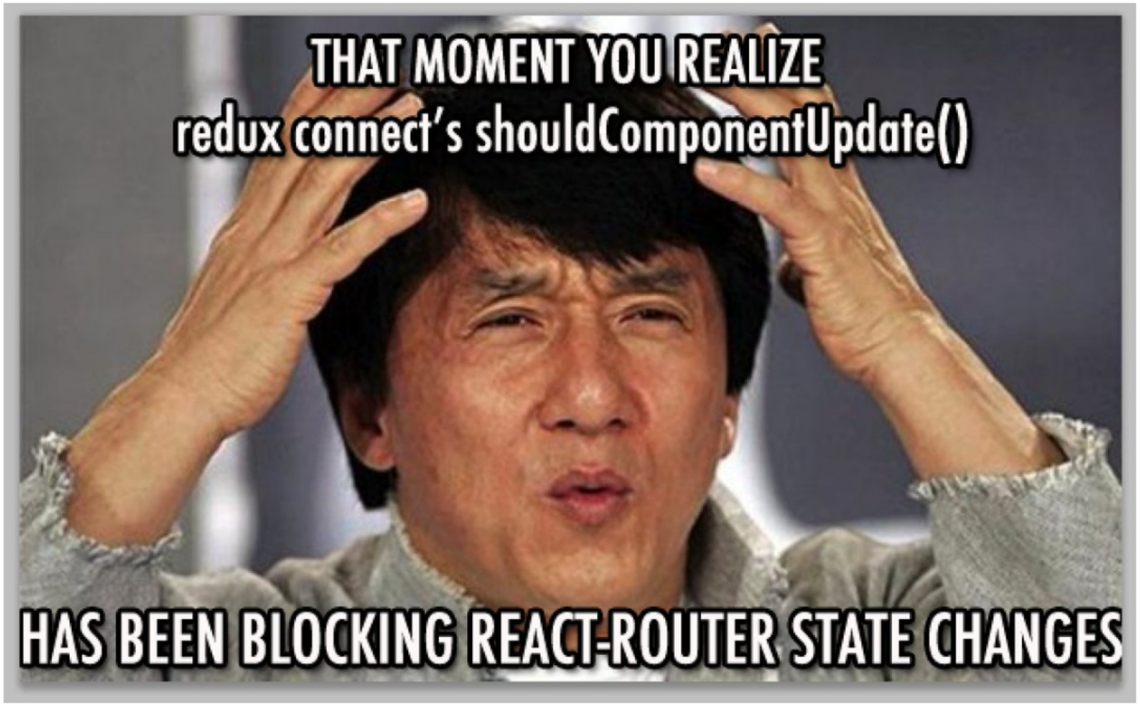
- **Altogether: 100.0%**
  - Mandatory: 100.0%
  - Optional: no optional tasks

Overall comment:
Good job


THAT MOMENT YOU REALIZE
redux connect's shouldComponentUpdate()
HAS BEEN BLOCKING REACT-ROUTER STATE CHANGES

Resources

Read or watch:

- Redux CreateStore
- Redux Connect
- Redux Provider
- Redux Middleware
- Redux Thunk
- Redux devtools
- Redux Reselect

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google:**

- Redux connectors and how to use them
- The different functions you can pass to a connector (mapStateToProps, mapDispatchToPros)
- How to map an action creator to a component using a connector
- How to map an async action creator to a component with Redux Thunk
- What Redux Providers are and how to set up your app's store
- How you can improve a connector's performance using Reselect
- How to use Redux's dev tools to debug the state of your application

Requirements

- Allowed editors: `vi`, `vim`, `emacs`, `Visual Studio Code`
- All your files should end with a new line

- A `README.md` file, at the root of the folder of the project, is mandatory
- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using node `12.x.x` and `npm 6.x.x`
- Push all of your files, including `package.json` and `.babelrc`
- All of your functions must be exported

Provided files

`dashboard/dist/courses.json`

Click to show/hide file contents

`dashboard/dist/login-success.json`

Click to show/hide file contents

`dashboard/dist/notifications.json`

Click to show/hide file contents

Tasks

0. Write mapStateToProps

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Reuse the latest dashboard project you worked on in the React course `0x09-React_Redux_reducer` and install `react-redux`

Within the `App/App.js` file:

- Create a function named `mapStateToProps`. This should connect the `uiReducer` you created and the property `isLoggedIn` that your component is already using
- Import `connect` from Redux, and connect the `mapStateToProps` to the component

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_0/dashboard/src/App/App.js`

View results

1. Create a small store

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the `index.js` file:

- Create a store using `createStore` from Redux that would contain the `uiReducer` state
- Implement a provider passing the store that you created to the main `App`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_0/dashboard/src/index.js`

View results

2. Test MapStateToProps

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Export the `mapStateToProps` function you created if you haven't already, and in the `App.test.js` file:

- Create a new suite to test the function
- Create a test that verify that the function returns the right object when passing the

```
let state = fromJS({
  isUserLoggedIn: true
});
```

Should return `{ isLoggedIn: true }`

Tips:

- At this point your app is not functional but you should be able to load the page without crashing
- Remember that the state of uiReducer is using Map from Immutable

Requirements:

- No error should be displayed within the console
- All the tests in the project should pass

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_0/dashboard/src/App/App.test.js`

View results

3. Update mapStateToProps

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the `App.js` file:

- Update the `mapStateToProps` function to also pass to the component the value for `displayDrawer` . It should be connected to the Reducer attribute `isNotificationDrawerVisible`
- Update the render function of the component to use the value `displayDrawer` coming from the props instead of the state

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_1/dashboard/src/App/App.js`

View results

4. Connect your actions creators

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the `App.js` file:

- Connect to the component the actions creators `displayNotificationDrawer` and `hideNotificationDrawer`
- You should use the simplified version for the `mapDispatchToProps` . No need to import `bindActionCreators`
- Modify the render function of the component to use the functions passed within the props instead of the action within the Class component

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_1/dashboard/src/App/App.js`

View results

5. Refactor your code

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the `App.js` file:

- You can delete the old function `handleDisplayDrawer`
- You can delete the old function `handleHideDrawer`
- Remove any state property related to the display of the drawer
- Remove any binding in the constructor
- You are now passing to your components props. You need to define `propTypes` and `defaultProps`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_1/dashboard/src/App/App.js`

View results

6. Update your tests

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

You can now refactor the `App.test.js` file:

- You don’t need to test the functions `handleDisplayDrawer` and `handleHideDrawer` since everything is already tested using the Redux mechanism
- You need to update the test you previously created to support the new attribute

Tips:

- At this point your app should be functional and able to display/hide the drawer using the Redux state
- Remember that the state of `uiReducer` is using Map from Immutable

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_1/dashboard/src/App/App.test.js`

View results

7. Async actions & Thunk middleware

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Let’s implement the `LoginRequest` / `logout` actions creators accross the entire application. `LoginRequest` is calling an API and is Async. Therefore, Redux will not support it. We will need to use a middleware

Install `redux-thunk` within your project. And in the `index.js` file, apply the middleware to your store

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_2/dashboard/src/index.js`

View results

8. Connect LoginRequest to the App

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Modify the file `App/App.js`:

- Connect the action creator `loginRequest` and map it to the `login` prop
- Modify the component to use the new `login` function from the props instead of the one within the class
- Refactor the component to remove any `login` or `logout` function and bind

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_2/dashboard/src/App/App.js`

View results

9. Connect user state to the Footer

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Modify the file `Footer/Footer.js`

- Create a `mapStateToProps` function
- Map the `user` props to the `user` within the Redux state
- Connect the Footer component to the function you created
- Modify the render function and remove any use of the Context. Instead use the `user` prop

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_2/dashboard/src/Footer/Footer.js`

View results

10. Connect Logout action creator to the Header

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Modify the file `Header/Header.js`

- Create a `mapStateToProps` function
- Map the `user` props to the `user` within the Redux state
- Connect the Header component to the function you created
- Connect the Header component to the `logout` action creator
- Modify the render function and remove any use of the Context. Instead use the `user` prop. When the user clicks on the link, it should now dispatch the `logout` action creator

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_2/dashboard/src/Header/Header.js`

View results

11. Modify the uiReducer

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Now that we can have the entire login request and the entire feedback loop, let's modify a few things within the reducer:

- When the action `LOGIN` is passed, set the user within the state to the one passed within the action
- When the `LOGOUT` action is passed, make sure to set the `user` to `null`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_2/dashboard/src/reducers/uiReducer.js`

View results

12. Modify the test suites

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Modify the test suites of the different components you modified:

- In the `App.test.js`, `Footer.test.js`, and `Header.test.js` to import the Stateless components instead of the connected component
- Remove any use of the `mount` function, and convert everything to use the `shallow` function
- You should remove any use of `setState` within the tests and pass directly the props to the stateless components
- Remove any test linked to the `login`, `logout` function within App, and Header
- Add a test in `uiReducer` to support the new action you just created

Tips:

- At this point your app should be functional and able to display/hide the drawer, login/logout using the Redux state

- Remember that the state of uiReducer is using Map from Immutable
- You can now see that your components logic is simplified. They only respond to props change. The logic is happening within the action creators

Requirements:

- Do not forget to add `defaultProps` and `PropTypes` to any component receiving props
- No error should be displayed within the console
- All the tests in the project should pass

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_2/dashboard/src/App/App.test.js, task_2/dashboard/src/Footer/Footer.test.js, task_2/dashboard/src/Header/Header.test.js, task_2/dashboard/src/reducers/uiReducer.test.js`

View results

13. Understand how to use the Redux Chrome extension

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Install the Redux DevTools extension on your Chrome browser:

- Modify the `index.js` to support the extension
- Use the application and note the different actions being registered when you are logging in / logging out
- Note that a version of the state is saved along the different actions and you can jump at a different moment of the user journey

Tips:

- Read the documentation of the extension to learn how to support the Chrome extension as well as the Thunk middleware
- This extension can be one of the most powerful tool to debug an application. Make sure to become familiar with it

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_3/dashboard/src/index.js`

Check submission View results

14. Combine store: Root reducer

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Since you have more than one reducer for your application, you will need to combine them into the store.

Create a new file `reducers/rootReducer.js`, in this file, export a `rootReducer`:

- the root should contain every reducer
- `courses` maps to `courseReducer`
- `notifications` maps to `notificationReducer`
- `ui` maps to `uiReducer`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_4/dashboard/src/reducers/rootReducer.js`

View results

15. Combine store: modify the application

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the `index.js`, create the store using the root reducer instead of only the ui reducer:

- Any component connected to the state will probably need to be updated since you added a nested level

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_4/dashboard/src/index.js`

View results

16. Combine store: write the tests

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Modify the test suites:

- In the `App.test.js`, modify `mapStateToProps` to correctly work with the new format of the reducer
- Create a `rootReducer.test.js` file to test the root reducer's initial state for the following structure:

```
{
  courses: Map {},
  notifications: Map {},
  ui: Map {}
}
```

Requirements:

- No errors in the browser's console
- All tests should pass
- Use `combineReducer` to create the root reducer

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_4/dashboard/src/App/App.test.js, task_4/dashboard/src/reducers/rootReducer.test.js`

View results

17. Connect notifications: New Action Creator

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

We now know how to connect a simple component to a reducer. Let's work on connecting a more complex component to the the entire API.

Add the following three action creators to `notificationActionCreators.js`

- `setLoadingState` whose parameter is a boolean. It will send the `SET_LOADING_STATE` action and the boolean.
- `setNotifications` whose parameter is an array. It will send the `FETCH_NOTIFICATIONS_SUCCESS` action with the data.
- `fetchNotifications` (which does not have a parameter). Calling it will dispatch `setLoadingState` with the boolean set to true
  - It fetches `/notifications.json`
  - Once the fetch is realized, it will dispatch `setNotifications` with the data
  - At the end of the query it sets the loading state to `false` again

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_5/dashboard/src/actions/notificationActionCreators.js`

View results

18. Connect notifications: Improve reducer

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the function `notificationReducer` within `notificationReducer.js`:

- Make sure to add a `loading` attribute to the initial state.
- Modify the `notifications` object to have the right initial state when merging the data coming from the API
- Create a `SET_LOADING_STATE` case and update the state accordingly
- Modify the `FETCH_NOTIFICATIONS_SUCCESS` case to perform a `mergeDeep` with the data

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_5/dashboard/src/reducers/notificationReducer.js`

View results

19. Connect notifications to the reducer

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the `Notifications.js` component:

- Map the prop `listNotifications` to the `messages` within the `notifications` state
- Map the action `fetchNotifications` to the component
- In `componentDidMount` , call `fetchNotifications`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_5/dashboard/src/Notifications/Notifications.js`

View results

20. Connect notifications: clean up

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

With this new behavior, let's clean up old functions and test data

- Delete `NotificationItemShape.js`
- Remove the notification list and delete `markNotificationAsRead` within `App.js`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_5/dashboard/src/App/App.js`

View results

21. Connect notifications: update the test suites

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Modify the test suites to pass the tests:

- Update `notificationReducer.test.js` to support the new attributes and default state
- Clean up `App.test.js` for the function you just removed
- Modify `Notifications.js` and `Notifications.test.js` to make sure that every tests pass correctly

Add new tests:

- Add a test in `Notifications.test.js` to verify that the function `fetchNotifications` is called when the component is mounted
- Add a test for `setLoadingState` , `setNotifications` , and `fetchNotifications` to verify that they each create the right actions
- Add a test for `SET_LOADING_STATE` to verify that it updates the reducer correctly

Tips:

- At this point, when you load the page, you should be able to see the list of notifications coming from the API using the developer tools or when clicking on the notifications toggle
- Use some CSS to make the notifications panel readable
- Look at `valueSeq` from Immutable to iterate on your list of notifications without having to use `toJS()`

Requirements:

- Make sure to update the different Proptypes so you don't have any errors in the console
- Make sure to create the new actions that the action creators are using

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_5/dashboard/src/reducers/notificationReducer.test.js, task_5/dashboard/src/App/App.test.js, task_5/dashboard/src/Notifications/Notifications.js, task_5/dashboard/src/Notifications/Notifications.test.js, task_5/dashboard/src/actions/notificationActionCreators.test.js`

View results

22. Selectors

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

To improve performance in your connector, you should always use selectors when you can. Let's modify the Notifications connector to reuse the selector we built in the previous project:

- Update `Notifications.js` to use `getUnreadNotifications`
- Map the `markAsAread` action creator to the component, and use it for `markNotificationAsRead`

Tips:

- At this point, when you load the page, you should be able to see the list of notifications. Clicking on one notification should make it disappear from the list

Requirements:

- Make sure to update the selector to use the same `valueSeq` you created previously
- Make sure to update the tests to work as expected
- Make sure to create the new actions that the action creators are using

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_6/dashboard/src/Notifications/Notifications.js, task_6/dashboard/src/Notifications/Notifications.test.js`

View results

23. Connect courses: create a course selector

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In `selectors/courseSelector.js`, create a selector that will:

- Get all the course entities from within the reducer
- Return a List using `valueSeq` from Immutable

Write a new file `courseSelector.test.js`, that will verify that the selector is working correctly

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_7/dashboard/src/selectors/courseSelector.js, task_7/dashboard/src/selectors/courseSelector.test.js`

View results

24. Connect courses: create a fetch courses function

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In `actions/courseActionCreators.js`:

- Create a new function named `fetchCourses`, that will query the API in `courses.json` (provided in the project description, put it in your `dist` folder)
- When the API returns the data, dispatch the action `setCourses`

In `courseActionCreators.test.js`, create a test to verify that the fetch is working correctly

Repo:

- GitHub repository: `alx-react`

- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_7/dashboard/src/actions/courseActionCreators.js, task_7/dashboard/src/actions/courseActionCreators.test.js`

View results

25. Connect the courses component

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In `CourseList.js`, connect the component to:

- The three action creators: `fetchCourses`, `selectCourse`, and `unSelectCourse`
- Connect the data to the list of courses using `getListCourses` selector
- When the component mount, call the action `fetchCourses`

Create a new function `onChangeRow`:

- It will take two arguments: `id (string)`, `checked (boolean)`
- When `checked` is `true`, call the action `selectCourse` with the id
- When `checked` is `false`, call the action `unSelectCourse` with the id

Modify `CourseListRow`:

- Pass a new prop, `isChecked`, to the component that will pass the `isSelected` attribute coming from the state of the reducer
- Pass the `onChangeRow` function to the component
- Modify the component to not use its local state anymore

In the file `CourseList.test.js`, create two new tests:

- Verify that the action is dispatched when the component is mounted
- Verify that the two actions are correctly dispatched when the `onChangeRow` function is called

Tips:

- At this point, when you load the page and you log in, you should be able to see the list of courses. Make sure that everything is working correctly using the developer tools or using the Redux tool
- When checking or unchecking a row, you should see the state in the Redux tool updated. You should also see the change on the UI
- Be careful that the API is sending Strings instead of Number for the IDs. You will probably need to adapt your reducers and tests
- Delete the `CourseShape` file now

Requirements:

- Make sure to update the tests to work as expected

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_7/dashboard/src/CourseList/CourseList.js, task_7/dashboard/src/CourseList/CourseList.test.js`

View results

26. Memoized selectors: Redux Reselect

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Our current selectors are useful but they are not really performant. Imagine a very long list of notifications with multiple filters on the type and on the read status. This would require a lot of resources to compute. Memoized selectors are very powerful in this sense.

Install Redux Reselect and create a new selector named `getUnreadNotificationsByType` in `notificationSelector.js`:

- This selector should combine the state of the filter, and the list of notifications
- When the filter is set to default, it should return all the unread notifications
- When the filter is set to urgent, it should return all the unread and urgent notifications
- Delete `getUnreadNotifications`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_8/dashboard/src/selectors/notificationSelector.js`

View results

27. Memoized selectors: update the UI

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In the Notifications component:

- Update the component to use the new selector you just created
- Map the component to the Action `setNotificationFilter`
- Add two buttons under the text `Here is the list of notifications`. The first one contains `‼`. On click, it set the filters of notifications to `URGENT`. The second one contains `🔹`. On click, it sets the filter of notifications to `DEFAULT`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_8/dashboard/src/Notifications/Notifications.js`

View results

28. Memoized selectors: update the test suite

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

In `Notifications.test.js`, add two new tests:

- Clicking on the first button should call `setNotificationFilter` with `URGENT`
- Clicking on the second button should call `setNotificationFilter` with `DEFAULT`

In `notificationSelector.test.js`:

- Update the previous tests to work correctly
- Create a new test to verify that the selector returns unread urgent notifications when the filter is set

Tips:

- At this point, you should be able to load the notifications panel, filter the list using the two new buttons, and mark items as read

Requirements:

- Make sure to update the tests to work as expected

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_8/dashboard/src/Notifications/Notifications.test.js, task_8/dashboard/src/selectors/notificationSelector.test.js`

View results

29. Container/Component

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Our components can become very verbose when we start adding connectors and actions. It is also becoming harder to tests what is supposed to be our React component, and the interations of the application. To simplify our architecture, we can use the concept of containers and components:

- Create a new file `NotificationsContainer.js`. This component will take care of connecting to the state, and fetching the notifications on mount
- The component should render the `Notifications` components and pass the required props to it
- Modify the file `Notifications.js`. It should now become a functional component
- Create a new test file for `NotificationsContainer.js`. It should make sure the fetching is happening on mount
- Modify `Notifications.test.js` file to only support the new behavior of the file

Tips:

- No need to repeat every single prop, you can use the spread operator

Repo:

- GitHub repository: `alx-react`
- Directory: `0x09-react_redux_connectors_and_providers`
- File: `task_9/dashboard/src/Notifications/Notifications.js, task_9/dashboard/src/Notifications/Notifications.test.js, task_9/dashboard/src/Notifications/NotificationsContainer.js, task_9/dashboard/src/Notifications/NotificationsContainer.test.js`

View results

Ready for a new review