

0x08. React Redux reducer+selector

Front-end My Platform JavaScript ES6 React



Weight: 1

Projects(/projects/current)



Project over - took place from Aug 22, 2024 6:00 AM to Aug 29, 2024 6:00 AM



QA Reviews I can make(/corrections/to_review)



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)



Curriculums(/dashboards/my_curriculums)



Concepts(/concepts)



Conference rooms(/dashboards/video_rooms)



Servers(/servers)



Sandboxes(/user_containers/current)



Tools(/dashboards/my_tools)

Resources



Video on demand(/dashboards/videos)

Read or watch:

- Reducers (/rltoken/SzgQcaVZ6qtF1ccU-S2DiA)
- Selectors (/rltoken/m3ctiAA74QV6YYqZ8YBZTQ)
- Writing tests (/rltoken/E5mFy6WxHnMflwxYhy2gzw)
- Immutable Map documentation (/rltoken/oeA22lgPb_GvU1nOzWoA3w)
- Normalizr (/rltoken/fmN8EIQtqvKbLVgJuRyM0Q)
- Normalizing State Shape (/rltoken/wCbecNeGJhMu3hu38S7RCw)



Peers(/users/peers)

Discord(<https://discord.com/app>)

Learning Objectives

At the end of this project, you should be able to explain to anyone

(/rltoken/e1Q_JKJmhjTNmRU9kO6GFw), **without the help of Google:**

My Profile(/users/my_profile)

- The purpose of a reducer and the role it plays within your application





- Why a reducer should stay as pure as possible
 - Why mutations should not happen within a reducer
 - The use of Immutable within the reducer
 - The use of Normalizr within the app
 - Selectors: what they are and when to use them
- Home(/)



Requirements



- My Planning(/planning/me)
- Allowed editors: vi, vim, emacs, Visual Studio Code
- All your files should end with a new line



- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using node 12.x.x and npm 6.x.x



- A README.md file, at the root of the folder of the project, is mandatory
- QA Reviews I can make/corrections to review
- Push all of your files, including package.json and .babelrc
- All of your functions must be exported



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Tasks



Curriculums(/dashboards/my_curriculums)

0. Write a basic reducer

mandatory



Concepts(/concepts)

- Reuse the latest dashboard project you worked on in the React course 0x08-React_Redux_action_creator+normalizr



Conference rooms(/dashboards/video_rooms)

Create the basic state



In a file named reducers/uiReducer.js, define the initial state of the reducer for the UI:

Servers(/servers)

- It should have one boolean property isNotificationDrawerVisible
- It should have one boolean property isUserLoggedIn
- It should have one empty object user



Create the reducer function



Tools(/dashboards/my_tools)

In the same file, import all the actions that you created in the file actions/uiActionTypes and create a reducer function named uiReducer :



Video on demand(/dashboards/videos)

- DISPLAY_NOTIFICATION_DRAWER should set isNotificationDrawerVisible to true
- HIDE_NOTIFICATION_DRAWER should set isNotificationDrawerVisible to false
- LOGIN_SUCCESS should set isUserLoggedIn to true
- LOGIN_FAILURE should set isUserLoggedIn to false
- LOGOUT should set isUserLoggedIn to false



Peers(/users/peers)

Write the tests



Discord(https://discord.com/app)

In a file named reducers/uiReducer.test.js, define the test suite for our simple reducer:

- Write a test verifying the state returned by the uiReducer function equals the initial state when no action is passed
- Write a test verifying the state returned by the uiReducer function equals the initial state when the action SET_NOTIFICATION_DRAWER is passed

My Profile(/users/my-profile)





- Write a test verifying the state returned by the `uiReducer` function, when the action `DISPLAY_NOTIFICATION_DRAWER` is passed, changes correctly the `(/) isVisible` property

Tips:



- Don't forget to set up the default case in your switch function

Requirements:



- My Planning(/planning/me)
- You should not mutate the state within the reducer
- You must use the spread operator to change the state



- All the tests in the project should pass



Repo: QA Reviews I can make(/corrections/to_review)

- GitHub repository: `alx-react`
- Directory: `0x08-react_redux_reducer_selector`
- File: `task_0/dashboard/src/reducers/uiReducer.js`,
`task_0/dashboard/src/reducers/uiReducer.test.js`



Curriculums(/dashboards/my_curriculums)

1. Use Immutable for the UI Reducer

mandatory



Concepts(/concepts)

Now that you have set up a basic reducer, let's reuse what we learned in the Immutable module and apply it to that reducer:



- Install `Immutable.js` within the project
- Update the `uiReducer.js` file to use `Map` from `Immutable.js`
- Update the different part of the reducer function to use `set` from `Map`
- Update the test suite, so it takes into account the changes

Tips:



- Sandboxes(/user_containers/current)
- You can use the `toJS` function within your tests for an easy comparison
- Remember that `Immutable.js` always return a new `Map` after a modification



Tools(/dashboards/my_tools)

Requirements:



- For better performances, do not use any `fromJS`, `toJS` function within the reducer
- All the tests in the project should pass

Repo:



- Peers(/users/peers)
- GitHub repository: `alx-react`
- Directory: `0x08-react_redux_reducer_selector`



- File: `task_1/dashboard/src/reducers/uiReducer.js`,
`task_1/dashboard/src/reducers/uiReducer.test.js`



My Profile(/users/my_profile)

2. Create a reducer for Courses

mandatory

Create a load action

In the `courseActionTypes` file, create a new action corresponding to when the API returns the list of courses. You can name it `FETCH_COURSE_SUCCESS`

Create the course reducer and default state

In a file `courseReducer.js`, write a reducer function. The default state should be an empty array.

Define the `FETCH_COURSE_SUCCESS` action

When the action creator sends the action `FETCH_COURSE_SUCCESS`, it also sends the list of courses in a `data` attribute. The action would look like:

```
{
  type: FETCH_COURSE_SUCCESS,
  data: [
    {
      id: 1,
      name: "ES6",
      credit: 60,
    },
    {
      id: 2,
      name: "Webpack",
      credit: 20,
    },
    {
      id: 3,
      name: "React",
      credit: 40,
    },
  ],
}
```

When updating the state of the reducer, you should also set the attribute `isSelected` to false for every item in the list. The expected data from the reducer should be:

Video on demand(/dashboards/videos)

Peers(/users/peers)

Discord(<https://discord.com/app>)

My Profile(/users/my_profile)



```
[
  {
    (/) id: 1,
      name: "ES6",
      isSelected: false,
      credit: 60
    },
    {
      id: 2,
      name: "Webpack",
      isSelected: false,
      credit: 40
    },
    {
      id: 3,
      name: "React",
      isSelected: false,
      credit: 40
    },
    {
      id: 4,
      name: "Evaluation quizzes",
      isSelected: false,
      credit: 40
    }
]
```

Define the SELECT COURSE and UNSELECT COURSE actions

When the action creator sends the action `SELECT_COURSE`, it also sends an index corresponding to the id of the course to update. The action would look like:

```
{
  type: SELECT_COURSE,
  index: 2
}
```

The expected data from the reducer should be:

```
[
  {
    id: 1,
    name: "ES6",
    isSelected: false,
    credit: 60
  },
  {
    id: 2,
    name: "Webpack",
    isSelected: true,
    credit: 40
  },
  {
    id: 3,
    name: "React",
    isSelected: false,
    credit: 40
  }
]
```

My Profile(/users/my_profile)



When the action creator sends the action `UNSELECT_COURSE` , it also sends an index corresponding to the id of the course to update. The action would look like:

```
(/)  
{  
  type: UNSELECT_COURSE,  
  index: 2  
}
```

The expected data from the reducer should be:

```
[  
  { Projects(/projects/current)  
    id: 1,  
    name: "ES6",  
    isSelected: true,  
    credit: 60  
  },  
  { Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)  
    id: 2,  
    name: "Webpack",  
    isSelected: false,  
    credit: 20  
  },  
  { Curriculums(/dashboards/my_curriculums)  
    id: 3,  
    name: "React",  
    isSelected: false,  
    credit: 40  
  },  
  { Conference rooms(/dashboards/video_rooms)  
    id: 4,  
    name: "React",  
    isSelected: false,  
    credit: 40  
  }  
]
```

Write the tests

In a `courseReducer.test.js` , write a test suite for the new reducer. Define the following tests:

- Test that the default state returns an empty array
- Test that `FETCH_COURSE_SUCCESS` returns the data passed
- Test that `UNSELECT_COURSE` returns the data with the right item updated
- Test that `UNSELECT_COURSE` returns the data with the right item updated

ps: Video on demand(/dashboards/videos)

- Use ES6 for this reducer, we can look at Immutable later

Requirements:

- Try to make the shape of object as efficient as possible, for example you can use ES6 Map
- All the tests in the project should pass

Discord(<https://discord.com/app>)

Repo:

- GitHub repository: `alx-react`
- Directory: `0x08-react_redux_reducer_selector`
- File: `task_2/dashboard/src/actions/courseActionTypes.js`,
`task_2/dashboard/src/reducers/courseReducer.js`,





(/)

3. Create the reducer for notifications

mandatory



Home(/)

Create a load action



My Planning(/planning/me)

In the `notificationActionTypes` file, create a new action corresponding to when the API returns the list of notifications. You can name it `FETCH_NOTIFICATIONS_SUCCESS`



Projects(/projects/current)

Create the notifications reducer and default state

In a file `notificationReducer.js`, write a reducer function. The default state should be an object with:



- `notifications`, which will store the list of notifications
- `filter`, which will be the attribute storing which filter is selected



Define the `FETCH_NOTIFICATIONS_SUCCESS` action

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

When the action creator sends the action `FETCH_NOTIFICATIONS_SUCCESS`, it also sends the list of notifications in a data attribute. The action would look like:



```
{
  Curriculums( /dashboards/my_curriculums )
  type: FETCH_NOTIFICATIONS_SUCCESS,
  data: [
    { Concepts( /concepts )
      id: 1,
      type: "default",
      value: "New course available"
    },
    {
      id: 2,
      Servers( /servers )
      type: "urgent",
      value: "New resume available"
    }
  ]
}
Sandboxes( /user_containers/current )
{
  id: 3,
  type: "urgent",
  Tools( /dashboards/my_tools )
  value: "New data available"
}
]
Video on demand( /dashboards/videos )
```

When updating the state of the reducer, you should also set the attribute `isRead` to false for every item in the list. The expected data from the reducer should be:



Peers(/users/peers)

Discord(<https://discord.com/app>)

My Profile(/users/my_profile)

```

{
  filter: "DEFAULT",
  notifications: [
    {
      id: 1,
      isRead: false,
      type: "default",
      value: "New course available"
    },
    {
      id: 2,
      isRead: false,
      type: "urgent",
      value: "New resume available"
    },
    {
      id: 3,
      isRead: false,
      type: "urgent",
      value: "New data available"
    }
  ]
}

```

Define the MARK_AS_READ action

When the action creator sends the action `MARK_AS_READ`, it also sends an index corresponding to the id of the notification to update. The action would look like:

```

{
  type: MARK_AS_READ,
  index: 2
}

```

The expected data from the reducer should be:

```

{
  filter: "DEFAULT",
  notifications: [
    {
      id: 1,
      isRead: false,
      type: "default",
      value: "New course available"
    },
    {
      id: 2,
      isRead: true,
      type: "urgent",
      value: "New resume available"
    },
    {
      id: 3,
      isRead: false,
      type: "urgent",
      value: "New data available"
    }
  ]
}

```



My Profile(/users/my_profile)


```

{
  filter: "DEFAULT",
  notifications: [
    {
      id: 1,
      isRead: false,
      type: "default",
      value: "New course available"
    },
    {
      id: 2,
      isRead: false,
      type: "urgent",
      value: "New resume available"
    },
    {
      id: 3,
      isRead: false,
      type: "urgent",
      value: "New data available"
    }
  ]
}

```

Define the SET_TYPE_FILTER action

When the action creator sends the action `SET_TYPE_FILTER`, it also sends a filter attribute with either `DEFAULT` or `URGENT`. The action would look like:

```

{
  type: SET_TYPE_FILTER,
  filter: "URGENT"
}

```

The expected data from the reducer should be:

```

Tools(/dashboards/my_tools)

Video on demand(/dashboards/videos)

```

```

Peers(/users/peers)

Discord(https://discord.com/app)

```



My Profile(/users/my_profile)

```

{
  filter: "URGENT",
  notifications: [
    {
      id: 1,
      isRead: false,
      type: "default",
      value: "New course available"
    },
    {
      id: 2,
      isRead: false,
      type: "urgent",
      value: "New resume available"
    },
    {
      id: 3,
      isRead: false,
      type: "urgent",
      value: "New data available"
    }
  ]
}

```

Tips:

Concepts(/concepts)

- Use ES6 for this reducer, we can look at Immutable later

Requirements:

Conference rooms(/dashboards/video_rooms)

- Try to make the update of object as efficient as possible, for example you can use ES6 Map
- All the tests in the project should pass

Servers(/servers)

Repo:

Sandboxes(/user_containers/current)

- GitHub repository: alx-react
- Directory: 0x08-react_redux_reducer_selector

Tools(/dashboards/my_tools)

- File: task_3/dashboard/src/actions/notificationActionTypes.js, task_3/dashboard/src/reducers/notificationReducer.js, task_3/dashboard/src/reducers/notificationReducer.test.js

Video on demand(/dashboards/videos)

4. Normalizr & Immutable

mandatory



Peers(/users/peers)

As you can see, updating a specific item in a collection is rather complicated and error prone. Using Normalizr is a good opportunity to simplify mutation



Discord(<https://discord.com/app>)

Course schema

Create a new file `schema/courses.js`. In the file define a schema entity for `courses`. Create a function `coursesNormalizer` that would take `data` as argument and normalize the data with the schema you created.

My Profile(/users/my_profile)

In the **course reducer function**:

- Update the initial state to use an `Immutable.js Map`
- When `FETCH_COURSE_SUCCESS` action is called, normalize the data with the function you created (`/`) and merge it with the state
- When `SELECT_COURSE` or `UNSELECT_COURSE` is called, use the `setIn` function from `immutable` to update the value of the item `Home(/)`

Update the notification schema

In the file `schema/notifications.js`, create a function `notificationsNormalizer` that would take `My Planning(/planning/me)` data as argument and normalize it with the notification schema you created in the previous course.

Update the notification reducer

In the notification reducer function:

- ✓ • Update the initial state to use an `Immutable.js Map` `QA Reviews I can make(/corrections/to_review)`
- When `FETCH_NOTIFICATIONS_SUCCESS` action is called, normalize the data with the function `notificationsNormalizer` you created and merge it with the state
- ? • When `SELECT_NOTIFICATION` is called, use the `setIn` function from `immutable` to update the value of the state `Evaluation quizzes(/dashboards/my_evaluation/quizzes)`
- When `MARK_AS_READ`, use the `setIn` function from `Immutable` to update the value of the item in the state

Update the test files/suites:

- Update the course reducer test file to match the new reducer `Curriculums(/dashboards/my_curriculums)`

Tips:

- You can use the `fromJS` function from `Immutable.js` to easily create the initial state from an object `Concepts(/concepts)`
- You can use the `toJS` function from `Immutable.js` to easily compare the expected data `Conference rooms(/dashboards/video_rooms)`
- Selecting an unselecting a course item should only take one line now
- Marking a notification item as read should only take one line now `Servers(/servers)`

Requirements:

- >_ • All the tests in the project should pass. `Sandboxes(/user_containers/current)`

Repo:

- GitHub repository: `alx-react`
- Directory: `0x08-react_redux_reducer_selector`
- File: `task_4/dashboard/src/schema/courses.js`, `task_4/dashboard/src/reducers/courseReducer.js`, `task_4/dashboard/src/schema/notifications.js`,

- `task_4/dashboard/src/reducers/notificationReducer.js`, `task_4/dashboard/src/reducers/courseReducer.test.js`, `task_4/dashboard/src/reducers/notificationReducer.test.js`


Discord(<https://discord.com/app>)

5. Selectors


mandatory


Selectors are an efficient way to access the data from the state because a selector is not recomputed unless one of its arguments change. `My Profile(/users/my_profile)`

Let's create a few selectors for the Notifications reducer in


 src/selectors/notificationSelector.js

-  Create a first selector for the filter named `filterTypeSelected`, that will return the value of the filter


-  Create another selector for the notifications named `getNotifications`, that will return the list of notifications in a Map format


-  Create another selector for the notifications named `getUnreadNotifications`, that will return the list of unread notifications in a Map format


Create a test suite for your selectors in a file named `src/selectors/notificationSelector.test.js`:

-  Projects(/projects/current)
- test that `filterTypeSelected` works as expected
 - test that `getNotifications` returns a list of the message entities within the reducer
 - test that `getUnreadNotifications` return a list of the message entities within the reducer


Tips:


-  Evaluate your tests, your dashboards state variable using the reducer you created. And pass the state to the selector functions
- You can also look into using `Reselect` for your own projects when you have advanced needs for filtering, reducing and merging data from the state

 Requirements: Curriculums(/dashboards/my_curriculums)

-  Concepts(/concepts)
- All the tests in the project should pass

Repo:


-  Conference rooms(/dashboards/video_rooms)
- GitHub repository: `alx-react`
 - Directory: `0x08-react_redux_reducer_selector`
 - File: `task_5/dashboard/src/selectors/notificationSelector.js`,
`task_5/dashboard/src/selectors/notificationSelector.test.js`

 Sandboxes(/user_containers/current)

 Tools(/dashboards/my_tools)

 Video on demand(/dashboards/videos)

Copyright © 2024 ALX, All rights reserved.

 Peers(/users/peers)

 Discord(<https://discord.com/app>)



My Profile(/users/my_profile)