

0x05. React state

Front-end JavaScript ES6 React



Weight: 1



Projects(/projects/current)



Project over - took place from Aug 7, 2024 6:00 AM to Aug 13, 2024 6:00 AM



An auto-generated name for 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)

imgflip.com

ONE DOES NOT SIMPLY

MUTATE THIS STATE DIRECTLY

Resources

Tools(/dashboards/my_tools)

Read or watch:



Video on demand(/dashboards/videos)

- State and lifecycle (/rltoken/rL2YOlSKl2HlcenyNz5cqQ)
- SetState and State callback (/rltoken/AeohbCmE3m-m-xgABFWglA)
- Context (/rltoken/SZ1mD7nEblEVUPrrN6SrRA)
- Forms and Controlled components (/rltoken/IBhcM3C1FOwY7uGYyNMclg)
- Lifting State Up (/rltoken/iZ1dULJUZE85Lh0-yTSpYg)
- React Hooks (/rltoken/4ISNYIQ67BkW53J7kGCccQ)
- Enzyme State (/rltoken/rSRoY2jGlahIH8KkZDWK0w)
- Enzyme SetState (/rltoken/D9Kq4M6VVxAga9-iJVgsYg)
- Enzyme Instance (/rltoken/wqn34UANx7UE2nkoIU-ntg)
- Enzyme Simulate (/rltoken/GdM5eK75XXsl1EVqAJ4q5w)



Peers(/users/peers)



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

My Profile(/users/my_profile)



Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/with token /SzjDoz06BNzJeQqmc1p_Q), **without the help of Google**.



- What the state of a component or a container is
- The lifecycle of a component
- How to modify a state and execute code in the right order



- What a controlled component is
- How to use Forms in React
- How to reuse smaller components, keep them pure, and lift its state to principal containers



- The use of a React Hook and how to create one
- How to test State changes with Enzyme



QA Reviews I can make (/corrections/to_review)

Requirements



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



Concepts (/concepts)

Tasks



Conference rooms (/dashboards/video_rooms)

0. Adding a local state for notifications

mandatory



Servers (/servers)

Using the previous project (0x05. React inline styling), we have modularized our React application without worrying about interactions and state, which is usually a recommended process of development. Now, our application is in a good place to start adding logic and state.



Sandboxes (user containers / current)

Modify the App component in task_0/dashboard/src/App.App.js :



Tools (/dashboards/my_tools)

Create a local state to store a `displayDrawer` element:



Video on demand (/dashboards/videos)

- Define the default state for the state in the constructor of the Class
- Create a function named `handleDisplayDrawer` that will change the value of `displayDrawer` to `true`
- Create a function named `handleHideDrawer` that will change the value of `displayDrawer` to `false`



Peers (/users/peers)

Modify the Notifications import in task_0/dashboard/src/App/App.js :



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

- Pass to the component the prop `displayDrawer` using the local state
- Pass the new functions `handleDisplayDrawer` and `handleHideDrawer`

Modify the App test suite in task_0/dashboard/src/App/App.test.js :

- Add a test to verify that the default state for `displayDrawer` is `false`. Verify that after calling `handleDisplayDrawer`, the state should now be `true`
- Add a test to verify that after calling `handleHideDrawer`, the state is updated to be `false`



Modify the Notifications component in `task_0/dashboard/src/Notifications/Notifications.js` :



- Define the `propTypes` and the `defaultProps` for the new props
- (/) When clicking on Your notifications , call `handleDisplayDrawer`
- When clicking on the close button, call `handleHideDrawer`



Home(/)

At this point, after reloading the app, you should be able to show / hide the notifications panel

Modify the Notifications test suite in



My Planning(/planning/me)

`task_0/dashboard/src/Notifications/Notifications.test.js` :

- Add a test to verify that clicking on the menu item calls `handleDisplayDrawer`
- Add a test to verify that clicking on the button calls `handleHideDrawer`



Projects(/projects/current)

Tips:



- Remember that you implemented `shouldComponentUpdate` . You will need to modify the logic to allow the component to rerender when the prop `displayDrawer` changes



- Use `setState` and `instance` when creating tests with `Enzyme`
- Remember to use spies to verify if a function is being called. You can pass a spy as a property

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Requirements:



- Do not forget to bind the functions you are passing to the children to improve performances
- When running, there should not be any lint error in the console

Curriculums(/dashboards/my_curriculums)



Repo:

Concepts(/concepts)

- GitHub repository: `alx-react`
- Directory: `0x05-React_state`
- File: `task_0/dashboard/src/App/App.js`, `task_0/dashboard/src/App/App.test.js`,
`task_0/dashboard/src/Notifications/Notifications.js`,
`task_0/dashboard/src/Notifications/Notifications.test.js`



Conference rooms(/dashboards/video_rooms)



Servers(/servers)



Sandboxes(/user_containers/current)

Check submission



Controlled components and state callback

mandatory



create a form within the Login component & handle login submit in

Video on demand(/dashboards/videos)

`task_1/dashboard/src/Login/Login.js` :

- Create a local state with the value `isLoggedIn` set to `false`
- Wrap the inputs within a `form` element
- Replace the `submit` by an input element with type `submit`
- Create a function named `handleLoginSubmit` that will update the local state by setting `isLoggedIn` to `true`
- When the form is submitted call the newly created login submission handling function



Users(/users/people)



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

create controlled components in the Login component in `task_1/dashboard/src/Login/Login.js`

- Modify the local state to add two new values `email` and `password` . By default these values are empty but not `null`
- Modify the two inputs elements in the form and set their values to use the local state

My Profile(/users/my_profile)





- Create two function `handleChangeEmail` and `handleChangePassword` that the two inputs will call when the value in the input field is changed. The local state should update with what the user (/) is typing

modify state callback in `task_1/dashboard/src/Login/Login.js`



- Modify the local state to add one value `enableSubmit`. It should be set to `false` by default
- Modify the Submit button to only be enabled when the `enableSubmit` value of the local state is



`My Planning(/planning/me)`

- Every time the user changes the value of the `email` or `password` field, verify that both fields are not empty. If they are not empty, you can enable the submit button



`Projects(/projects/current)`

add tests in the `task_1/dashboard/src/Login/Login.js` suite



- Add a test to verify that the submit button is disabled by default
- Add a test to verify that after changing the value of the two inputs, the button is enabled

Tips:



- To simulate an input change, you can use the `simulate` method and use the `change` event

Requirements:



- The state should have all the default values set in the constructor
- Do not forget to bind the functions you are passing to the children to improve performances
- When submitting the form, the page should not reload



- Make sure that the button is always enabled when the two inputs are not empty. And make sure that the button is always disabled when one of the input's value is empty
- When running, there should not be any lint error in the console



`Conference rooms(/dashboards/video_rooms)`

Repo:



- `GitHub repository: alx-react`
- Directory: `0x05-react_state`



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



2. Context `Tools(/dashboards/my_tools)`

mandatory



Create a simple React Context in `task_2/dashboard/src/App/AppContext.js`

`Video on demand(/dashboards/videos)`

- Define a default user object, containing `email`, `password`, and `isLoggedIn`
- Define a default `logout` function



- Create a React context containing a `user` object and a `logout` function. Set both default values using the previously created elements

Create the local state for the App in `task_2/dashboard/src/App/App.js`



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

- Modify the local state of the App container by adding a `user` object and `logout` function. Set the default (or empty) values using the ones you previously created for the context



Create the login and logout functions in the App container in `task_2/dashboard/src/App/App.js` and use them

`My Profile(/users/my_profile)`



- Create a `logIn` function that takes as argument `email` and `password`. When the function is called, update the user object in the local state. Set the `email` and `password`. Set the `(/) isLoggedIn` value to `true`



- Modify the `logOut` function to reset the value of the `user` object in the local state
- Remove the two props `isLoggedIn` and `logOut` from the `App`, they will now be used through the state



- Refactor the render method to use the objects from the state to display the `CourseList` or the `My Planning` state

- Pass the new `logIn` function to the `Login` component



task_2/dashboard/src/Login/Login.js

- Remove from the state the `isLoggedIn` property, since we don't use it anymore
- Modify the `handleLoginSubmit` to call the newly created `logIn` function



QA Reviews I can make (corrections to _review)

Checkpoint



Take a moment to test your application. At this point, you should be able to login inside the app:

- Enter a few letters to the email and the password inputs, and click on the submit button. You should see the list of courses and the login component should be unmounted.
- Verify that the Notifications panel is still working correctly. You should be able to show/hide the panel by clicking on the `notifications` close button.



Curriculum (dashboards/my_curriculum)

Setting the context in task_2/dashboard/src/App/App.js :



- In the `App` container, wrap the entire app with the `AppContext` created earlier using the `provider` element. Set the value to the `user` and `logOut` function using the local state



Modify the `Header` component in task_2/dashboard/src/Header/Header.js :

- Modify the `Header` to inherit the context using the `ContextType` API
- Add a `Header` section, that is only displayed when the value `isLoggedIn` in the `user` object within the context is `true`
 - This section should display "Welcome **email** (logout)"
 - Add the id `logoutSection` to the section
 - Clicking on the `logout` link, should call the `logOut` function included within the context



Sandboxes (user_containers/current)



Tools (dashboards/my_tools)

At this point, you should be able to login (remove `Login` component and show `CourseList` component) and logout within the map (remove `CourseList` component, show `Login` component, show new header section)



Video on demand (dashboards/videos)

Create the tests!

in task_2/dashboard/src/Header/Header.test.js



- Make sure the current tests pass correctly
- Add a test that mounts the `Header` component with a default context value. Verify that the `logoutSection` is not created



- Add a test that mounts the `Header` component with a user defined (`isLoggedIn` is `true` and an email is set). Verify that the `logoutSection` is created

- Add a test that mounts the `Header` component with a user defined (`isLoggedIn` is `true` and an email is set) and the `logout` is linked to a spy. Verify that clicking on the link is calling the spy




in task_2/dashboard/src/App/App.test.js

My Profile (users/my_profile)

- Refactor the previous tests to not use props but state instead

- Refactor the test checking if `logout` is being called by verifying if the state is updated correctly instead
- 7) Create a test to verify that the `login` function updates the state correctly
- Create a test to verify that the `logout` function updates the state correctly

 **tips:** Home(/)

- Remember that function components can not use the context, you might need to convert the `Header` function component to a Class
- Unfortunately, Enzyme does not fully support the newly created Static Context API of React. Therefore do not use the `setContext` API of Enzyme since it is not compatible. Instead wrap the component using the `Context.Provider` to pass different context values

Requirements:

- ✓ **QA Reviews:** I can make (/corrections/to/review)
- Be careful that Context is using reference identity to determine when to re-render. Don't create directly an object within the Provider value
- Don't forget to export elements that are going to be reused through the app (e.g default user)
- ? **Evaluation quizzes:** (/dashboards/my_current_evaluation_quizzes)
- Don't forget to clean unused state and props after refactoring
- Don't forget to set the `propTypes` and `defaultProps` for any new prop

 **Repo:** Curriculums(/dashboards/my_curriculums)

- GitHub repository: `alx-react`
- Directory: `0x05-react_state`
- File: `task_2/dashboard/src/App/AppContext.js`, `task_2/dashboard/src/App/App.js`, `task_2/dashboard/src/Login/Login.js`, `task_2/dashboard/src/Header/Header.js`, `task_2/dashboard/src/Header/Header.test.js`, `task_2/dashboard/src/App/App.test.js`

 Servers(/servers)

3. Context consumer & advanced state

mandatory

 Sandboxes(/user_containers/current)
Context consumer: modify the Footer component in `task_3/dashboard/src/Footer/Footer.js`

- Without making the component a Class, make the component subscribe to the context changes
- When the user is logged in, display a new paragraph containing a link with the text `Contact us`

Modify the test suite in `task_3/dashboard/src/Footer/Footer.test.js`:

- Refactor the test to make them work correctly with the changes
- Add a test to verify that the link is not displayed when the user is logged out within the context
- Add a test to verify that the link is displayed when the user is logged in within the context

 **Advanced state: modify the App container in `task_3/dashboard/src/App/App.js`:**

- Set the `listNotifications` within the state
- Create a function `markNotificationAsRead`. It accepts an `id` (number) in argument. When the function is called, it remove the notification with that id from the list of notifications within the state
- Pass the list of notifications to the `Notifications` component using the state
- Pass the newly created function to the `Notifications` component

Modify the test suite in `task_3/dashboard/src/App/App.test.js`:



- Add a test to verify that `markNotificationAsRead` works as intended. You can for example set the state with a mock list of notifications, then call the function and verify that the state of the (/) container has been updated correctly

Modify the Notifications container in



task_3/dashboard/src/Notifications/Notifications.js :

- Refactor the code to delete the function `markAsRead` , we can now use the real one
- Refactor the code to delete the `shouldComponentUpdate` , the component is only using prop and state, and you should be able to optimize the performance with `PureComponent`
- Use the newly created function `markNotificationAsRead` to mark a notification as read
- Define the prop type and the default for `markNotificationAsRead`



Modify the test suite in task_3/dashboard/src/Notifications/Notifications.test.js :



- QA Reviews I can make(/corrections/to_review)
- Refactor the tests to match the new container

Checkpoint



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Take a moment to test your application. At this point:

- When you log in, you should be able to see a new footer
- When you display the panel of notifications, you should each of them disappear on click



Tips:

Curriculumums(/dashboards/my_curriculumums)

- If you end up having to reuse mock data a lot, feel free to export and import the same mockup from the state file



Concepts(/concepts)

Requirements:



- Conference rooms(/dashboards/video_rooms)
- Don't forget to clean unused state and props after refactoring
- Don't forget to set the `propTypes` and `defaultProps` for any new prop



Servers(/servers)

Repo:



- Github repository: `user-clone-react`
- Directory: `0x05-react_state`



- File: `task_3/dashboard/src/Footer/Footer.js`,
`task_3/dashboard/src/Footer/Footer.test.js`, `task_3/dashboard/src/App/App.js`,
`task_3/dashboard/src/App/App.test.js`,
`task_3/dashboard/src/Notifications/Notifications.test.js`



Video on demand(/dashboards/videos)

4. Introduction to react hook

mandatory



Peers(/users/peers)

Using React Hooks, modify the CourseListRow component in



task_4/dashboard/src/CourseList/CourseListRow.js :

- Add a new style named `rowChecked` with the background color `#e6e4e4`
- When the row is a simple row, add a checkbox within the first cell
- When the user checks the box, the styling of the row should use `rowChecked`



Tips & Requirements:

My Profile(/users/my_profile)



- Make sure that the checkbox of each row is correctly checked or unchecked after the user interaction
 - React Hooks are new in React and the number of resources can be limited. Refer to the official documentation of React. The exercise is simple enough that should not need anything else
 - Do not modify the component to a Class. You need to use the React Hook
- Home(/)



Repo:



- My Planning(/planning/me)
- GitHub repository: alx-react
- Directory: 0x05-react_state
- File Projects (/projects/and/ent)/CourseList/CourseListRow.js



QA Reviews I can make(/corrections/to_review)



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)



Curriculums(/dashboards/my_curriculums)

Copyright © 2024 ALX, All rights reserved.



Concepts(/concepts)



Conference rooms(/dashboards/video_rooms)



Servers(/servers)



Sandboxes(/user_containers/current)



Tools(/dashboards/my_tools)



Video on demand(/dashboards/videos)



Peers(/users/peers)



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



My Profile(/users/my_profile)