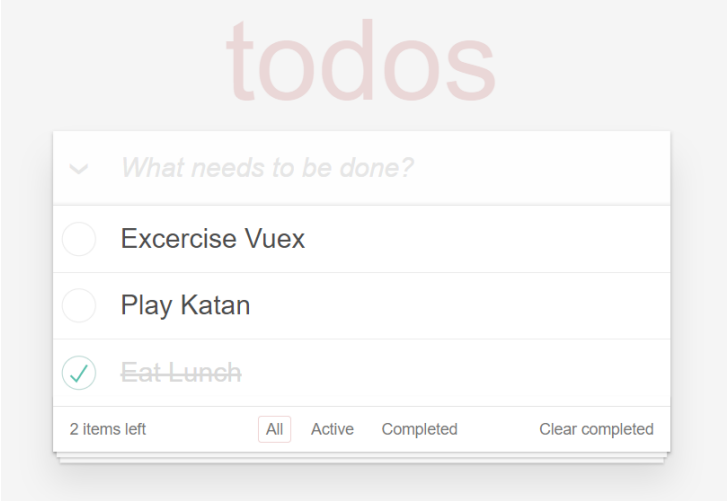
Todo, Todo



Build a todo-app application that uses a store to manage the state

Our data model:

```
const todo = {
    _id: "gZ6Nvy",
    txt: "Master Redux",
    importance: 9,
    isDone: false,
    createdAt: 1711472269690,
    updatedAt: 1711472269690
  }
```

You will start from a starter project with local component state and modify it to work with state management.

{coding academy

Comment

The starter project was crafted by taking the car-project and perform a search-and-replace (case sensitive) in the entire project (Ctrl+Shift+H) of:

- Car to Todo
- car to todo
- CAR to TODO
- maxSpeed to importance
- vendor to txt
- Vendor to Text
- Speed to Importance
- .id to ._id
- Being careful to avoid renaming in library files
- Changed bunch of file names
- From the UserAuth project:
  - Copied the userService
  - Copied the <LoginSignup> component
  - Update the <AppHeader> component

You have it ready to use

Our app is built from the following components:

- TodoApp (Routable, Smart component)
  - TodoList (Dumb)
  - TodoFilter (Dumb)
- TodoPreview (Dumb)
- TodoEdit (Routable, Smart)
- TodoDetails (Routable, Smart)

Terminology

- Dumb – a component that do not dispatch to the store, usually also get the data from parent by props
- Smart – a component that dispatches to the store

{coding academy

State Management

1. Store should manage the following state:
    a. List of todos
    b. isLoading
    c. Current filterBy
    d. User object
2. Use a todoService from the components and commit to the store
3. Add the following features:
4. Confirm before deleting a todo
5. Add color to todo
6. Add filter by <select>: All | Active | Done
7. In the <AppHeader> component show a todos progress-bar (percent of done todos)

userService

Exports the following API:

```
export const userService = {
    getLoggedinUser,
    login,
    logout,
    signup,
    getById,
    query,
    getEmptyCredentials
}
```

The service manages a user entity:

```
const user = {
    _id: "KAtTl",
    username: "muki",
    password: "muki1",
    fullname: "Muki Ja",
    createdAt: 1711490430252,
    updatedAt: 1711490430999
}
```

Add balance and activities:

```
{
        balance: 10000,
        activities: [{txt: 'Added a Todo', at: 1523873242735}]
}
```

User Balance

When user is loggedin, show the user balance next to his name, at the header.

When the user completes a task, his balance increases by 10 (balance just grow bigger, we don't decrease it when toggling the isDone to false)

<UserDetails>

Add a page to the app: <UserDetails> (Routable, Smart)

If the user is looking at his own user-details – allow changing his fullname and set preferences.

- Add to user object: prefs: {color: 'black', bgColor: 'white'}
- Also, render the user's activities list, example:
    - 2 minutes ago: Added a Todo: 'Wash the dishes'
    - Couple of hours ago: Removed the Todo: 'Talk to grandma'

More features

- Show a loading indication when todos are being loaded
- Show no “no todos to show..”, when there aren’t any todos
- Show Success and Failure messages in the component

- Show a progress-bar of todo's completion at the header and at the footer of the app
- Make it look nice
- Push it to github-pages

Bonus

- When filtering by txt, support debouncing
- Add sorting and paging in the todoService

PART 2

- Move the store-related async code into store actions
- Split your store to 2 modules: userStore and todoStore
- Complete the app, make it look good and responsive

Bonus

- Build a backend to your app
    - TIP: copy and modify the backend from the carProj / missBug)
- Push the app to a cloud hosting (render.com)