Practice Challenge: Animal Shelter (CodeGrade)

(https://github.com/learn-co-curriculum/react-hooks-practice-animal-shelter) (https://github.com/learn-co-curriculum/react-hooks-practice-animal-shelter/issues/new)

Overview

You'll build a small React application where you'll update state in response to a fetch payload and pass props among components to handle updates.

Animal Shelter

Time to put all of our hard-earned knowledge to the test! This lab is fairly big, and will require you to use everything you've learned up to this point. Don't be intimidated, there are plenty of tests to guide you along the way! In this lab, we'll be working on a front-end for an animal shelter. Sadly, there still are way too many cute pets without any owners. Let's help them out by creating a UI in React!

We **strongly** recommend completing this lab using Behavior Driven Development (BDD)—test the functionality in the browser **before** running the tests. You'll have a much better time seeing the results in the browser.

Run `npm install` to install our dependencies.

Then, run `npm run server` to start up `json-server` on `http://localhost:3001`.

In another tab, run `npm start` to start up our React app at `http://localhost:3000`.

Deliverables

- A user should be able to change the Animal Type filter/drop down to specify the type of animal they want to adopt.
- A user should be able to click on the 'Find pets' button, and they will see all of pets *only* for the type they specified in the drop down (you'll be fetching to a mock API to get this data).
- A user can click on 'Adopt' to adopt that pet. They cannot un-adopt it. No backsies!

Don't worry about persistence. We will not be saving this data to a real API. So if a pet is adopted, on refresh of the page, they will be available for adoption again. We are only going to focus on building the front end UI.

Instructions

On a high level, you will be working on several components that form the UI of the animal shelter adoption application. There are several components that need your attention. All of these components can be found in the `components/` folder. Starting from the top-level and working our way down through all its descendants:

App

1. The app's initial state is already defined. App has two children: the `<Filters />` and `<PetBrowser />` components.
2. App should pass a **callback** prop, `onChangeType`, to `<Filters />`. This callback needs to update `<App />`'s `filters.type` state.
3. `<Filters />` needs a **callback** prop, `onFindPetsClick`. When the `<Filters />` component calls `onFindPetsClick`, `<App />` should fetch a list of pets using `fetch()`.

- Assuming `json-server` is up and running, you can make a fetch to this exact URL: `http://localhost:3001/pets` with an **optional query parameter** to get your data.
- Use `App` 's state.filters to control/update this parameter
- If the `type` is `'all'` , send a request to `/pets`
- If the `type` is `'cat'` , send a request to `/pets?type=cat` . Do the same thing for `dog` and `micropig` .
- The pet data received will include information on individual pets and their adoption status.

1. Set `<App/>` 's `pets` state with the results of your fetch request, and pass the pet data down as a prop to `<PetBrowser />`
2. Finally, `App` should pass a **callback** prop, `onAdoptPet` , to `<PetBrowser />` . This callback should take in an id for a pet, find the matching pet in the `pets` array in `App` , and set the `isAdopted` property to `true` .

Filters

1. Should receive an `onChangeType` callback prop. This callback prop gets called whenever the value of the `<select>` element changes with the **value** of the `<select>` .
2. Should receive an `onFindPetsClick` callback prop. This callback prop gets called when the users clicks the 'Find pets' button.

PetBrowser

1. Should receive a `pets` prop. This is an array of pets that the component uses to render `<Pet />` components. App should determine which pets to pass down as props. App should be responsible for filtering this list based on the types of pets the user wants to see.
2. Should receive an `onAdoptPet` prop. This callback prop gets passed to its `<Pet />` children components.

Pet

1. Should receive a `pet` prop. Use the attributes in this data to render the pet card correctly. It should show the pet's `name` , `type` , `age` and `weight` . Based on the pet's `gender` , the component also needs to contain either a male (`♂`) or female (`♀`) symbol.
2. Each `pet` *may or may not* have an `isAdopted` property set to `true` . Using this property, render the correct button in the pet's card; if the pet is adopted, show the disabled button. Otherwise, show the primary button to adopt the pet.

3. Should receive an `onAdoptPet` callback prop. This callback prop gets called with the pet's `id` when the user clicks the adopt pet button — *not* when they click the disabled button!

Resources

- **Forms** (https://reactjs.org/docs/forms.html)
- **Events** (https://reactjs.org/docs/handling-events.html)
- **Using the State Hook** (https://reactjs.org/docs/hooks-state.html)

This tool needs to be loaded in a new browser window

Load Practice Challenge: Animal Shelter (CodeGrade) in a new window