CMPS 356 – Fall 2022
Web Applications Design and Development

Lab 03
React Hooks

Objective

1. Composing components and lifting the state up to share it among children,
2. Propagating property changes using callbacks,
3. Conditional content rendering,
4. Setting and using various types of hooks for:
   4.1. Setting and manipulating state (`useState`),
   4.2. Applying state change side effects (`useEffect`),
   4.3. Referencing elements (`useRef`),
   4.4. Reducing state variables (`useReducer`).

Prerequisites

1. Thinking in React tutorial: `https://beta.reactjs.org/learn/thinking-in-react`
2. Lifting State Up tutorial: `https://reactjs.org/docs/lifting-state-up.html`
3. React hooks API reference: `https://reactjs.org/docs/hooks-reference.html`
4. React component lifecycle: `https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram`

1. Reactive Todos

1. Explore and run the Todo List application provided under `base/todo-list` that was developed using plain JavaScript.
2. Recreate the application using React under `01-todo-list` and use as many components as needed.
3. Load the list of items from local storage when the application is first loaded.
4. Handle adding a new item and use a reference hook to capture the input value when creating a new item.
5. Handle checking and deleting an item.
6. Handle clearing the list of items.
7. Save the list of items in local storage whenever is it updated so it is available in the future. Make sure not to unintentionally overwrite the list of todos with an empty array after it is loaded. You can use a state variable to guard against this effect or set the initial state of todos directly from local storage.

2. Grouping States using Reduction

1. Create a new directory `02-state-reduction` and a React application under it. Use Figure 1 as reference.
2. Create an `RandomImage` component to fetch a random image from `https://picsum.photos` or `https://random.responsiveimages.io`. You should keep track of the loading status, the image itself, and any error that occurs. Display the image when it loads successfully and an error message when it fails. You should also

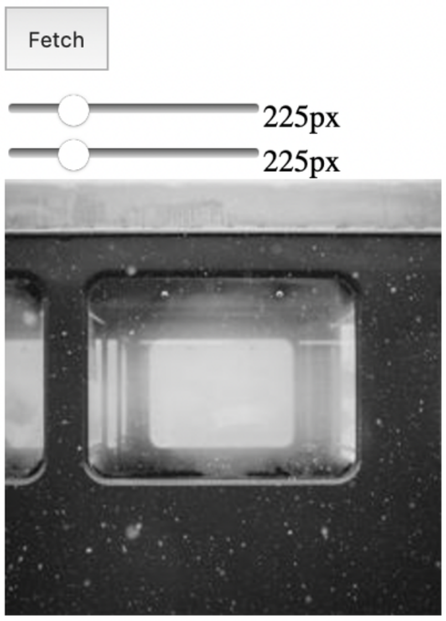
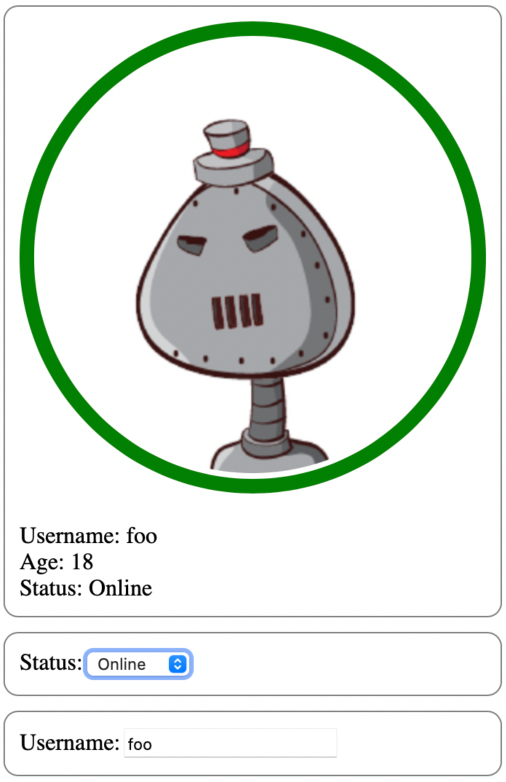


Figure 1: A `RandomImage` instance.

(temporarily) hint to the end-user that the image is loading when you first initiate the request.

3. Add a button to fetch a random image manually.
4. Reduce the multiple state variables into a single state object using a reducer hook and a dispatch method to manage the multiple stages of loading an image from the API.



Username: foo
Age: 18
Status: Online

Status: Online

Username: foo

Figure 2: A Gamer instance.

5. Extend your component so that the width and height of the random image can be updated all while using the same reducer by adding two sliders to control the values of width and height.
6. How can you reduce the number of additional image updates when moving the sliders?

3. Sharing State between Components

1. Create a new directory `03-state-sharing` and a React application under it. Use Figure 2 as reference.
2. Create a component `GamerInformation` that displays a gamer's username, an avatar generated from their username using RoboHash `https://robohash.org/username`, their age, and their online status.
3. Create another component `GamerStatus` that shows a gamer's online status and allows them to change it; use four values: Online, Busy, Away, and Offline.
4. Create one instance of each component for the same gamer and modify the status. How can this change be propagated between from the `GamerStatus` to the `GamerInformation` instance?
5. Share the state by lifting it up to a parent component `Gamer` that encapsulates the `GamerInformation` and `GamerStatus` instances.
6. Add another component `GamerUsername` that allows the gamer to change its username which automatically updates its avatar. Which state variable must be lifted? Which components must be updated to reflect the change?
7. Create a `GamerAvatar` component that display a gamer's avatar with a border color based on its status. Use it in the `GamerInformation` component to display the avatar.