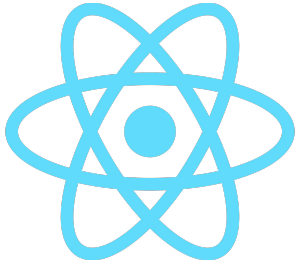


React.js

Lecture 2

Agenda

- Integrate with UI libraries.
- React Hooks
- Sharing data between components (props)
- Questions!

UI Libraries

- Bootstrap and React Bootstrap
  - https://react-bootstrap.github.io/docs/getting-started/introduction
  - https://reactstrap.github.io/?path=/story/home-installation--page
- MUI (Material UI)
  - https://mui.com/material-ui/getting-started/installation/
- Ant design
  - https://ant.design/
- Tailwind CSS

https://prismic.io/blog/react-component-libraries

React Hooks

What is hooks ?

Hooks are a new addition to React in version 16.8 that allows you use state and other React features, like lifecycle methods, without writing a class.

React Hooks

React Hooks are simple JavaScript functions that we can use to isolate the reusable part from a functional component. Hooks can be stateful and can manage side effects.

The Rules of React Hooks

- Only Call Hooks at the Top Level
- Only Call Hooks from React Functions

React Hooks

A hook is a special function that lets you "hook into" various React features. Imagine a function that returns an array with two values. That's what **useState** has:

- **The first value:** a variable with the state.
- The second value: a variable with an handler (a function to change the current state).

React Hooks: useState

useState :

To manage states. Returns a stateful value and an updater function to update it.

State lets a component “remember” information like user input. For example, a form component can use state to store the input value.

We use the array destructure, as we know its first and second position values, and we know they correspond to the state value and to a handler to change it.

Usage :

const [state, setState] = useState(initialStateValue)

https://www.freecodecamp.org/news/learn-react-usestate-hook-in-10-minutes/

React Hooks: useEffect

- Declare an effect. By default, your effect will run after every render.
- Specify the effect dependencies. Most effects should only rerun when needed rather than after every render. For example, a fade-in animation should only trigger when a component appears. Connecting and disconnecting to a chat room should only happen when the component appears and disappears or when the chat room changes. You will learn how to control this by specifying dependencies.
- Add cleanup if needed. Some effects need to specify how to stop, undo, or clean up whatever they were doing. For example, “connect" needs "disconnect," “subscribe" needs "unsubscribe," and “fetch" needs either “cancel" or "ignore." You will learn how to do this by returning a cleanup function.

https://www.freecodecamp.org/news/react-useeffect-absolute-beginners/

React Hooks: useEffect

Every React component goes through the same lifecycle:

- A component mounts when it's added to the screen.
- A component updates when it receives new props or state, usually in response to an interaction.
- A component unmounts when it's removed from the screen.

React Hooks: useEffect

useEffect :

accepts a function that can perform any side effects.

Usage :

```
useEffect(() => {
    return () => {
     // clean your code on unmount component
    }
},[arrayDependencies]);
```

[]: empty array means to fire on mount only

[dependencies]: means to fire on mount and with every change in this dependency value

React Hooks: useRef [self-study]

useRef :

It returns a ref object with a .current property. The ref object is mutable. Refs let a component hold some information that isn't used for rendering, like a DOM node or a timeout ID. Unlike with state, updating a ref does not re-render your component.

Usage :

Const inputRef = useRef();

<input ref={inputRef} ... />

Props

Passing Props to a Component

React components use props to communicate with each other. Every parent component can pass some information to its child components by giving them props. Props might remind you of HTML attributes, but you can pass any JavaScript value through them, including objects, arrays, and functions.

```
<Avatar
    person={{ name: 'Lin Lanying', imageId: '1bX5QH6' }}
    size={100}
/>
```

Read props inside the child component

You can read the props in the child component function

```
function Avatar(props) {
  //props.person and props.size
}
```

Or you can use ES6 destruct option to read the props sent keys

```
function Avatar({person,size = 50}) {}
```

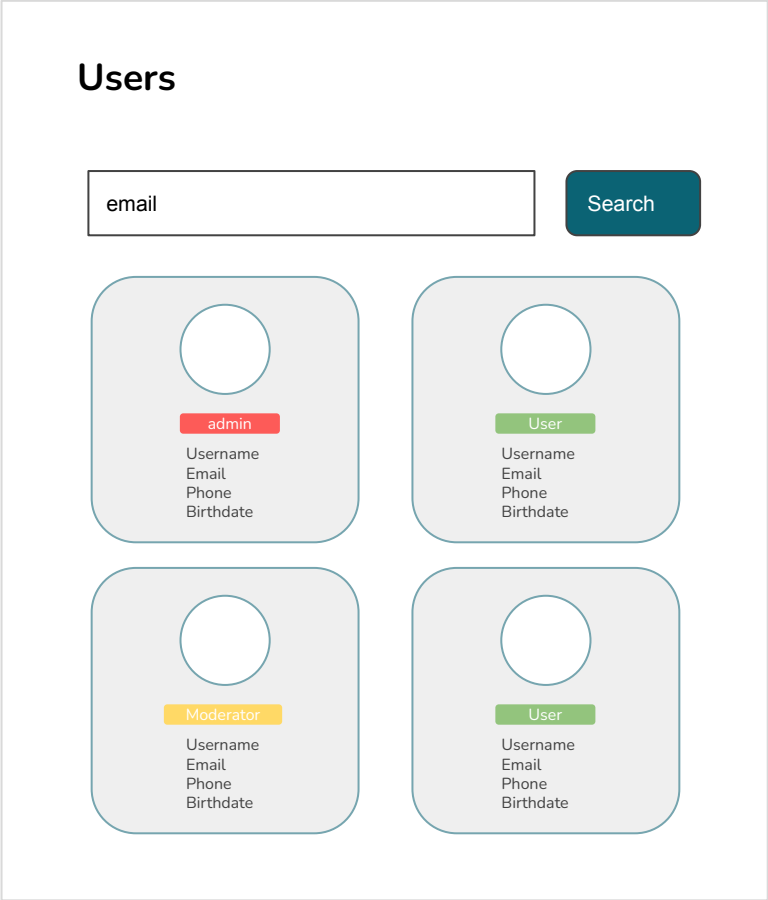
Thank you

Lap

Users list

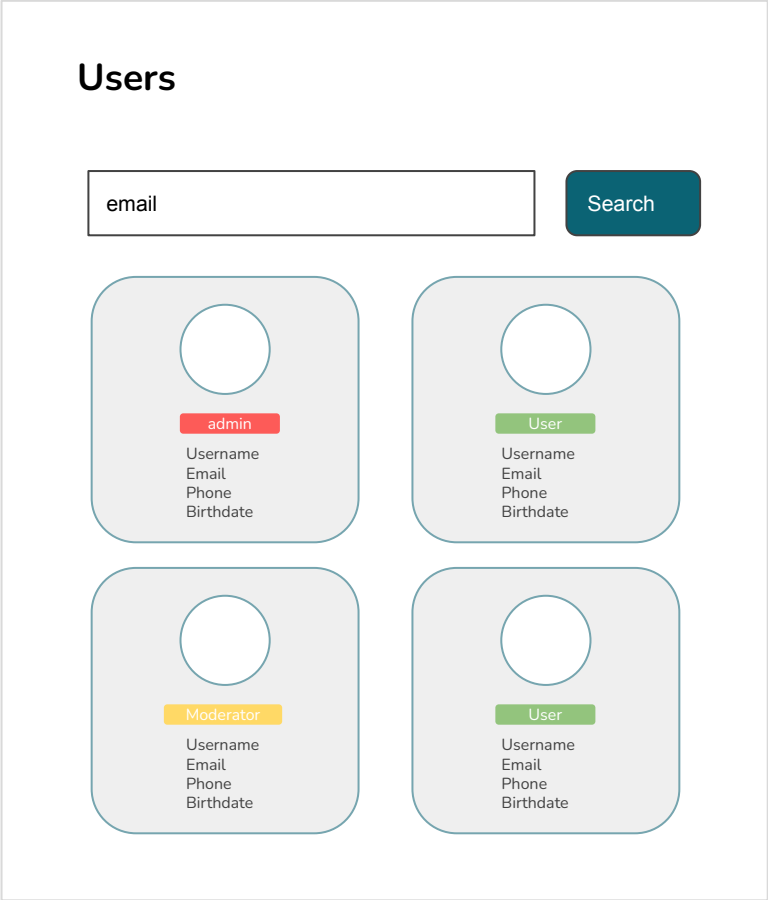
- Using the provided users list to render the users cards with the following:
  - Profile picture
  - Username
  - Email
  - Phone number
  - Birthdate
  - Role chip

- Based on the user role, show a chip with a different color. If admin shows chip with red, if user shows chip with green, if moderator show chip with yellow



Users list

- [Bonus] you can search and in the users list by email, and after user search, a reset button will appear when clicked, which will reset the fields and show all users again.



Task : To Do App

Create a to-do app to create self-notes with the following features:

- User can add new task
- The user can delete Task
- The user can mark as completed, and when marked as completed, it will be marked with a line through. [Bonus]

Using reusable components and passing data through components

Parent (state List)