# React.js Notes – Part 3

Contents

[React.js Notes – Part 3 1](#_Toc127296768)

[Section 9: Diving Deeper: Working with Fragments, Portals, & "Refs" 2](#_Toc127296769)

[99. Module Introduction 2](#_Toc127296770)

[100. JSX Limitations & Workarounds 2](#_Toc127296771)

[101. Creating a Wrapper Component 8](#_Toc127296772)

[102. React Fragments 10](#_Toc127296773)

[103. Introducing React Portals 11](#_Toc127296774)

[104. Working with Portals 12](#_Toc127296775)

[105. Working with "ref"s 22](#_Toc127296776)

[106. Controlled vs Uncontrolled Components 67](#_Toc127296777)

[Section 10: Advanced: Handling Side Effects, Using Reducers & Using the Context API 69](#_Toc127296778)

[108. Module Introduction 69](#_Toc127296779)

[109. What are “Side Effects” & Introducing useEffect 69](#_Toc127296780)

[110. Using the useEffect()Hook 72](#_Toc127296781)

[111. useEffect & Dependencies 86](#_Toc127296782)

[112. What to add & Not to add as Dependencies 119](#_Toc127296783)

[113. Using the useEffect Cleanup Function 121](#_Toc127296784)

[114. useEffect Summary 160](#_Toc127296785)

[115. Introducing useReducer & Reducers in General 178](#_Toc127296786)

[116. Using the useReducer() Hook 191](#_Toc127296787)

[117. useReducer & useEffect 290](#_Toc127296788)

[118. Adding Nested Properties AS Dependencies To useEffect 335](#_Toc127296789)

[119. Use Reducer vs useState for State Management 335](#_Toc127296790)

[120. Introducing React Context (Context API) 336](#_Toc127296791)

[121. Using the React Context API 345](#_Toc127296792)

[122. Tapping Into Context with the useContext Hook 364](#_Toc127296793)

[123. Making Context Dynamic 366](#_Toc127296794)

[124. Building & Using a Custom Context Provider Component 373](#_Toc127296795)

## Section 9: Diving Deeper: Working with Fragments, Portals, & "Refs"

### 99. Module Introduction

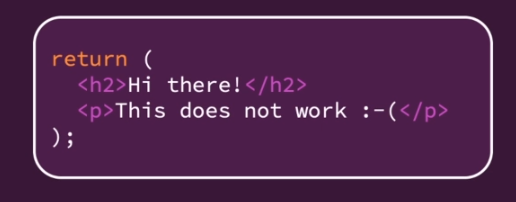
We will look at limitations faced with JSX code and how Fragments can help us overcome those limitations. We'll look at getting a cleaner DOM, better code structure with Portals. We will also look at working with Refs.

### 100. JSX Limitations & Workarounds

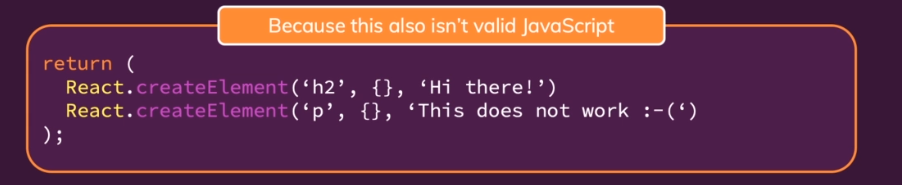
JSX is the code that you return from your components that will in the end be rendered by the real DOM in React.

JSX has limitations.

One limitation is that if we have adjacent root level JSX elements like below, we'll get an error. In React you can't return more than one "root" JSX element (you also can't store more than one "root" JSX element in a variable.

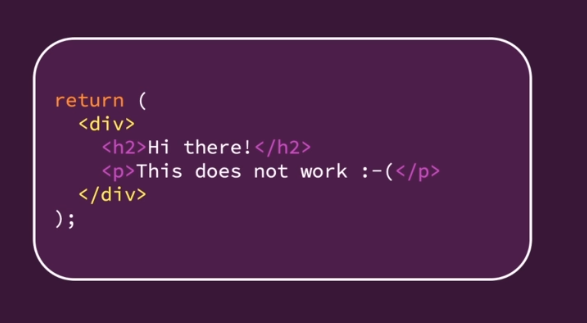


Children elements can be adjacent, but the topmost element can only be one element. This makes sense because in JavaScript we can't return more than one thing.



In the AddUser component we have a wrapping div, and that is a workaround.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <div>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </div>  );  };  export default AddUser; |

You can solve the problem of returning more than one root element by wrapping them in a div.   


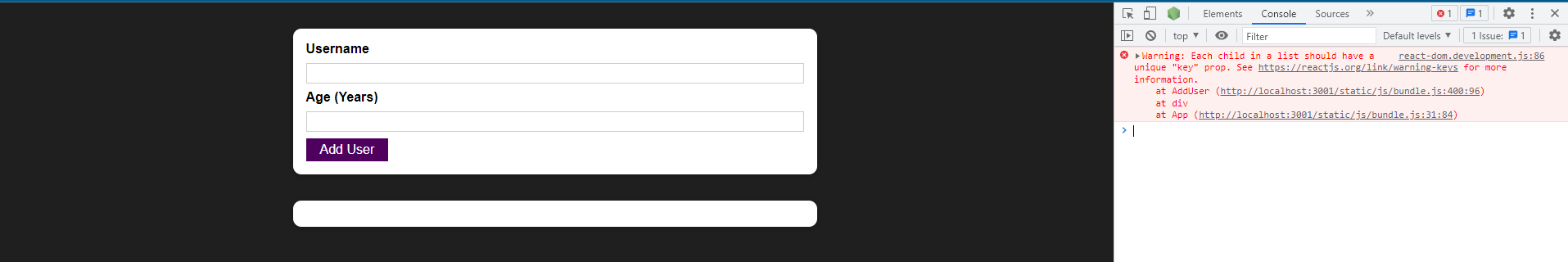
If you wrap the elements in a div, then you are only returning one element. It's similar to returning three numbers by putting them into an array.

**Important:** wrapping element does not have to be a div – ANY element will do the trip.

You could also use a JS array. We could remove our opening and closing div tags and replace with opening and closing square brackets. We would also need to separate the elements with commas and remove the curly braces around the error and ErrorModal because we are no longer inside of JSX here; we are inside of an array.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  **[**  error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )**,**  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  **]**  );  };  export default AddUser; |

We can return an array because React is able to work with arrays of JSX elements. However, we do get a warning in Chrome's developer tools console:



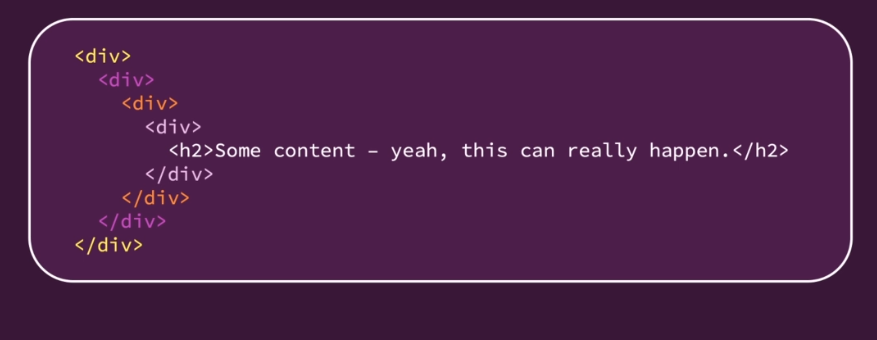
Because we're working with an array of JSX elements, React wants a key on every element. We can simply add a key prop with a custom value to each of the elements.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  [  error && (  <ErrorModal  **key="error-modal"**  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  ),  <Card **key="add-user-card"** className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  ]  );  };  export default AddUser; |

You typically don't use this solution because adding those keys and wrapping it in an array is a little bit cumbersome. It's way easier to simply wrap it in a div like before.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  **<div>**  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  **</div>**  );  };  export default AddUser; |

With the wrapping div or any other wrapping element, a new problem arises. Now we can end up with "<div> Soup" – where you have a real DOM rendered with many nested components and all of those components for various reasons need wrapping divs or other wrapping components, and you have all of these unnecessary divs being rendered in the real DOM even though they're only there because of this limitation in JSX.



In bigger apps, you can easily end up with **tons of unnecessary <div>**s or other elements which add **no semantic meaning or structure** to the page but **are only there because of React's JSX' requirement**.

### 101. Creating a Wrapper Component

We will add a new subfolder in our components folder called "Helpers". We add a Wrapper component. In there we won't import React because we are not going to write any JSX code. Instead we will do this:

|  |
| --- |
| src/components/Helpers/Wrapper.js |
| const Wrapper = props => {  return props.children;  };  export default Wrapper; |

props.children holds all of the content that you are passing between the opening and closing tag of your custom component.

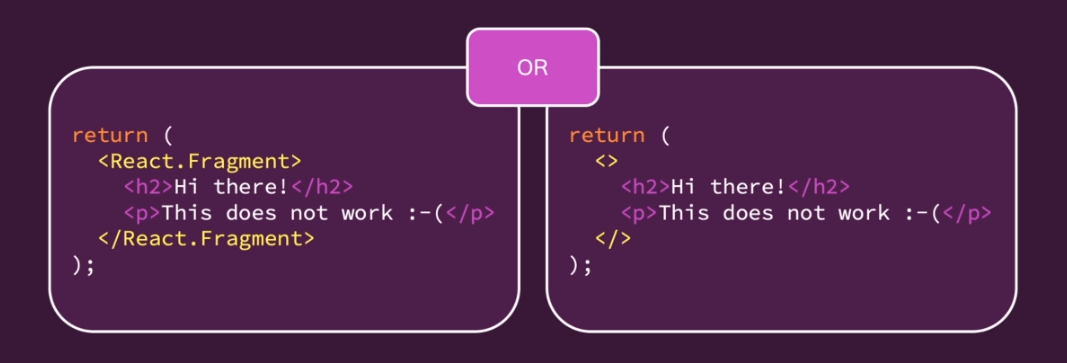
In AddUser we then use our Wrapper by replacing our divs with Wrapper.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  **<Wrapper>**  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  **</Wrapper>**  );  };  export default AddUser; |

The Wrapper is basically an empty component. Everything it does is that it returns props.children.

If we save AddUser, our application works just fine because in AddUser we have a wrapping element. It's a component that won't render anything to the DOM.

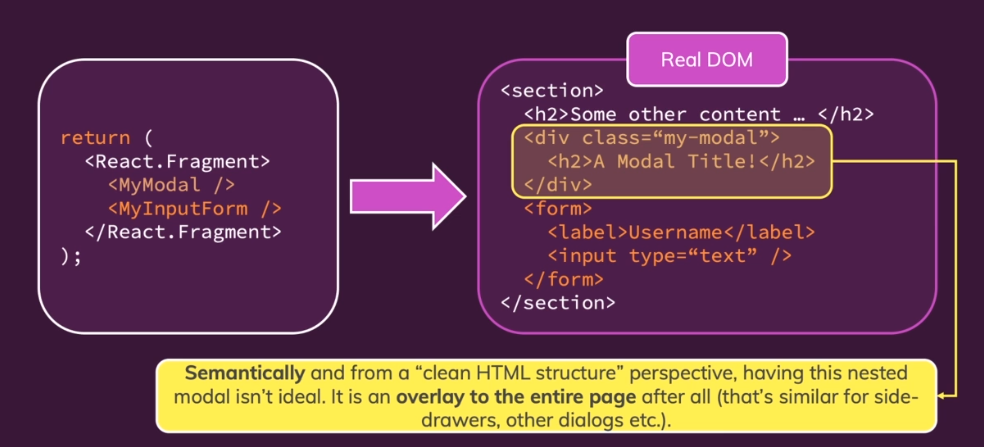
### 102. React Fragments

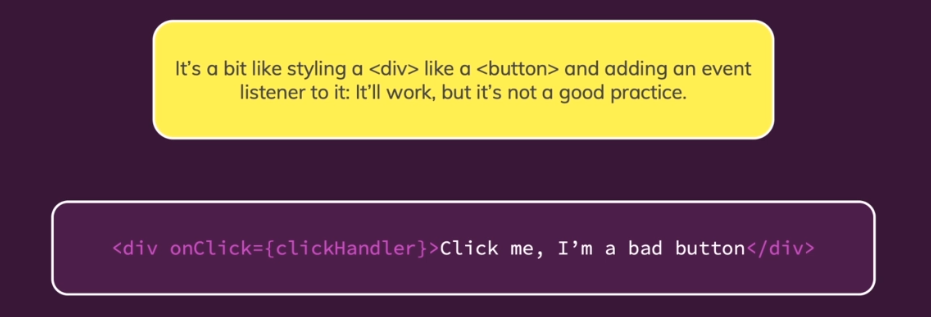
The Wrapper component that we built is not one we need to build because it comes with React. In React it is called the fragment component. You can always use the Fragmenet setup on the left in the screenshot but using the Fragment setup on the right depends on your project setup. 

A fragment is an empty wrapper component that doesn't render any real HTML element to the DOM but fulfills React's JSX requirement. We can use the built in wraper like so in the App component:

|  |
| --- |
| src/App.js |
| import React, { useState } from 'react';  import AddUser from './components/Users/AddUser';  import UsersList from './components/Users/UsersList';  function App() {  const [usersList, setUsersList] = useState([]);  const addUserHandler = (uName, uAge) => {  setUsersList((prevUsersList) => {  return [  ...prevUsersList,  { name: uName, age: uAge, id: Math.random().toString() },  ];  });  };  return (  **<>**  <AddUser onAddUser={addUserHandler} />  <UsersList users={usersList} />  **</>**  );  }  export default App; |

### 103. Introducing React Portals

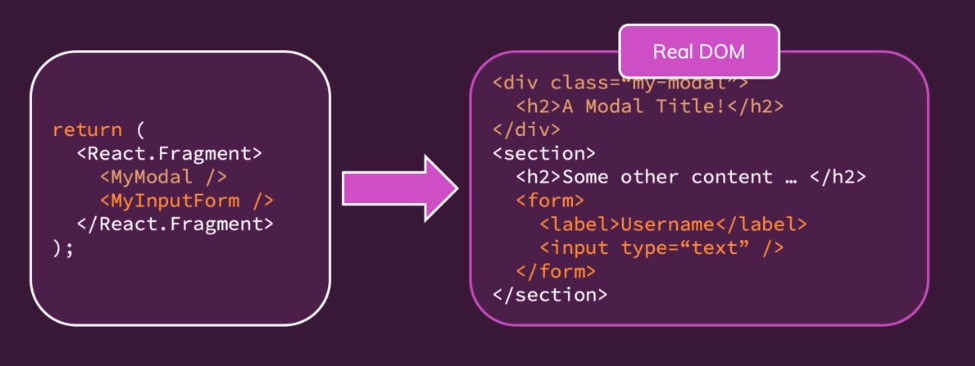




Styling a div like a button is a bad practice for accessibility, bad if a fellow developer has to work on it, and it's just a bad practice in general.

We can use a portal to keep the structure we have on the left in the screenshot below. We want to still be able to write our components the way we want to write them, so that we have no friction when we want to pass data around.



But you still render this differently in the real DOM.

For example, to render the modal html content somewhere else than it would normally go to so that on the left, JSX code re-write hasn't changed, but the rendered HTML code is a bit different from our JSX code. So, the Modal there is not next to the form. You can achieve this with React portals.

### 104. Working with Portals

We of course have an ErrorModal in our project. We can use a fragment in it.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const ErrorModal = (props) => {  return (  **<React.Fragment>**  <div className={classes.backdrop} onClick={props.onConfirm} />  <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  **</React.Fragment>**  );  };  export default ErrorModal; |

We now should use a portal because the entire ErrorModal component should not be rendered in the place it is being rendered.

Portals need two things

1. You need a place where you want to port the component to.
2. You need to let the component know that it should have a portal to that place.

To mark that place, we go into the "public" folder and then into "index.html". In this file it is common to add a div with an id, which you will then use to identify this place later. You could have ids called "backdrop-root" and "modal-root" and you could create multiple such roots for different kinds of Components that should be portaled there.

|  |
| --- |
| public/index.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8" />  <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />  <meta name="viewport" content="width=device-width, initial-scale=1" />  <meta name="theme-color" content="#000000" />  <meta  name="description"  content="Web site created using create-react-app"  />  <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />  <!--  manifest.json provides metadata used when your web app is installed on a  user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/  -->  <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />  <!--  Notice the use of %PUBLIC\_URL% in the tags above.  It will be replaced with the URL of the `public` folder during the build.  Only files inside the `public` folder can be referenced from the HTML.  Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will  work correctly both with client-side routing and a non-root public URL.  Learn how to configure a non-root public URL by running `npm run build`.  -->  <title>React App</title>  </head>  <body>  <noscript>You need to enable JavaScript to run this app.</noscript>  **<div id="backdrop-root"></div>**  **<div id="modal-root"></div>**  <div id="root"></div>  <!--  This HTML file is a template.  If you open it directly in the browser, you will see an empty page.  You can add webfonts, meta tags, or analytics to this file.  The build step will place the bundled scripts into the <body> tag.  To begin the development, run `npm start` or `yarn start`.  To create a production bundle, use `npm run build` or `yarn build`.  -->  </body>  </html> |

You could simplify this a bit and have an id of "overlay-root" rather than "modal-root", which will then hold all kinds of overlays – modals, side drawers, and so on. That's what we'll go with.

|  |
| --- |
| public/index.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8" />  <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />  <meta name="viewport" content="width=device-width, initial-scale=1" />  <meta name="theme-color" content="#000000" />  <meta  name="description"  content="Web site created using create-react-app"  />  <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />  <!--  manifest.json provides metadata used when your web app is installed on a  user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/  -->  <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />  <!--  Notice the use of %PUBLIC\_URL% in the tags above.  It will be replaced with the URL of the `public` folder during the build.  Only files inside the `public` folder can be referenced from the HTML.  Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will  work correctly both with client-side routing and a non-root public URL.  Learn how to configure a non-root public URL by running `npm run build`.  -->  <title>React App</title>  </head>  <body>  <noscript>You need to enable JavaScript to run this app.</noscript>  <div id="backdrop-root"></div>  <div id="**overlay-root**"></div>  <div id="root"></div>  <!--  This HTML file is a template.  If you open it directly in the browser, you will see an empty page.  You can add webfonts, meta tags, or analytics to this file.  The build step will place the bundled scripts into the <body> tag.  To begin the development, run `npm start` or `yarn start`.  To create a production bundle, use `npm run build` or `yarn build`.  -->  </body>  </html> |

We save index.html and then go back to our components.

We now work on the ErrorModal and tell React that it should be portaled somewhere. We can do this by creating a constant, and calling it Backdrop, which will actually be a new Component. We'll add it in this same file because in this app, I only use this Backdrop component in conjunction with my ErrorModal. So we will store both the ErrorModal and Backdrop components in one big file but we could also split it into multiple component files, especially if you would use the backdrop in conjunction with other component files as well. We pass in props and then we return that div that gets my backdrop class: <div className={classes.backdrop} onClick={props.onConfirm} />.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  **const Backdrop = (props) => {**  **return <div className={classes.backdrop} onClick={props.onConfirm} />**  **};**  const ErrorModal = (props) => {  return (  <React.Fragment>  <div className={classes.backdrop} onClick={props.onConfirm} />  <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  </React.Fragment>  );  };  export default ErrorModal; |

We will also add our ModalOverlay component, which gets props, and there I will return the Card and remove Card from where we are currently using it inside of the ErrorModal component.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  **const ModalOverlay = (props) => {**  **return <Card className={classes.modal}>**  **<header className={classes.header}>**  **<h2>{props.title}</h2>**  **</header>**  **<div className={classes.content}>**  **<p>{props.message}</p>**  **</div>**  **<footer className={classes.actions}>**  **<Button onClick={props.onConfirm}>Okay</Button>**  **</footer>**  **</Card>**  **};**  const ErrorModal = (props) => {  return (  <React.Fragment>  <div className={classes.backdrop} onClick={props.onConfirm} />  </React.Fragment>  );  };  export default ErrorModal; |

I have basically split my modal into two separate Components now – Backdrop and ModalOverlay – because that will make working with portals much easier.

We can also remove the div with the class of backdrop from the ErrorModal component.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  const ModalOverlay = (props) => {  return <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  };  const ErrorModal = (props) => {  return (  <React.Fragment>  </React.Fragment>  );  };  export default ErrorModal; |

What do we do inside of the fragment in the ErrorModal? We can add an expression because we're still inside of JSX code, and we want to call a method that is not actually defined on React but on another library that comes together with React, the react-dom library. You can imagine React as being the library that has all the React features, state management, and so on baked in. And react-dom uses React to bring that logic and these features into the web browser, so making them compatible with working with the DOM.

Put in other words, the react library doesn't care whether you run it in an environment that has a DOM or if you would use it to build a native app. So react-dom is kind of the adapter for React to the browser. And, therefore, since now we're going to portal something into a different place in the real DOM we need to import react-dom.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  **import ReactDOM from 'react-dom';**  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  const ModalOverlay = (props) => {  return <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  };  const ErrorModal = (props) => {  return (  <React.Fragment>  </React.Fragment>  );  };  export default ErrorModal; |

Inside of the React.Fragment and on ReactDOM you can now call a createPortal method.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import ReactDOM from 'react-dom';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  const ModalOverlay = (props) => {  return <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  };  const ErrorModal = (props) => {  return (  <React.Fragment>  **{ReactDOM.createPortal()}**  </React.Fragment>  );  };  export default ErrorModal; |

The createPortal method takes two arguments. The first one is your React node that should be rendered and here we can render our Backdrop lets say. It's important that we pass JSX, so pass <Backdrop /> rather than Backdrop because your really render it like this: <Backdrop />. We can then forward our onConfirm prop and get access to props.onConfirm, which I need to pass here to make sure that everything still works. I can now set onConfirm in my ErrorModal and forward the function I get on the onConfirm prop to the onClick prop inside of the Backdrop component.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import ReactDOM from 'react-dom';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  const ModalOverlay = (props) => {  return <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  };  const ErrorModal = (props) => {  return (  <React.Fragment>  {ReactDOM.createPortal(**<Backdrop onConfirm={props.onConfirm} />**, )}  </React.Fragment>  );  };  export default ErrorModal; |

The second argument of createPortal is a pointer to that container in the real DOM where this element should be rendered in. And in my case, we of course want to render our backdrop here in that "backdrop-root" component (the div inside of "public/index.html" that has an id of "backdrop-root"). We select the element where it should be rendered to, and for this, we use a DOM API. We use document.getElementById('backdrop-root') for example and get access to the "backdrop-root".

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import ReactDOM from 'react-dom';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  const ModalOverlay = (props) => {  return <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  };  const ErrorModal = (props) => {  return (  <React.Fragment>  {ReactDOM.createPortal(<Backdrop onConfirm={props.onConfirm} />, **document.getElementById('backdrop-root')** )}  </React.Fragment>  );  };  export default ErrorModal; |

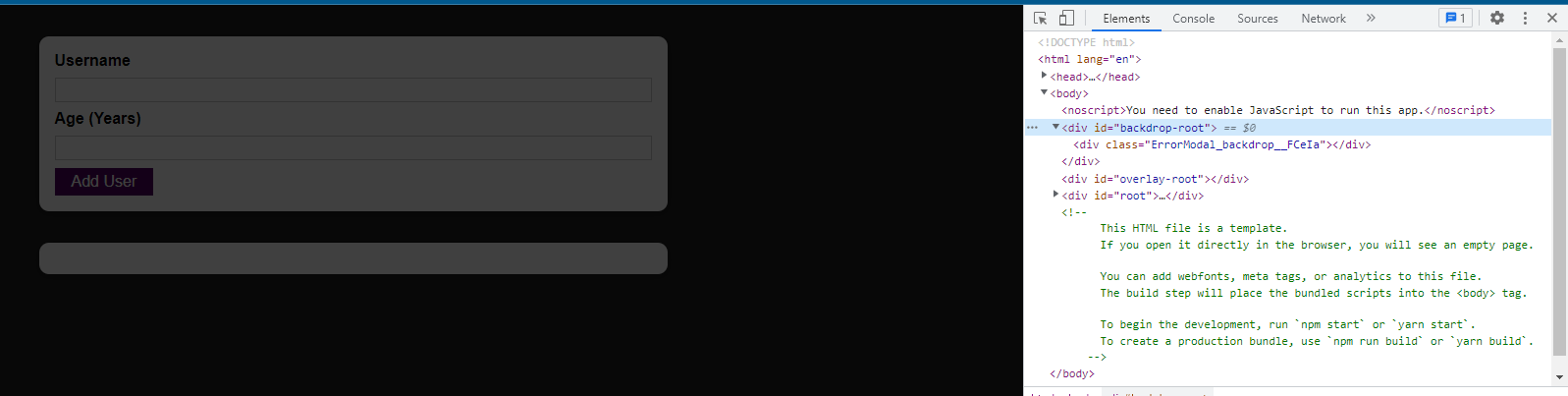
We really get access to a real HTML DOM element, a DOM node here. And we do this with the API that is provided by the browser. document.getElementById has nothing to do with React. WE really get access to a real DOM element with this API.

This is similar how we selected an element in "src/index.js". In "src/index.js" we also rendered the root component with the render method into a place selected with getElementById.

|  |
| --- |
| src/index.js |
| import React from 'react';  import ReactDOM from 'react-dom/client';  import './index.css';  import App from './App';  const root = ReactDOM.createRoot(**document.getElementById('root')**);  root.render(<App />); |

In the ErrorModal we're not rendering an element but inside of an existing application which is already being rendered by React, we portal. We move the HTML content that is about to be rendered into a different place.

If we reload the browser at this point, if we click the backdrop appears, but the modal does not since we haven't added the logic for this yet.



In Chrome's Dev Tools in the Elements tab, we see that in the "backdrop-root", we now have the modal backdrop. And now it will always be there no matter where you would use your error modal in your JSX code. No matter how deeply nested it is in other elements. It will always be here which is of course very close to the body.

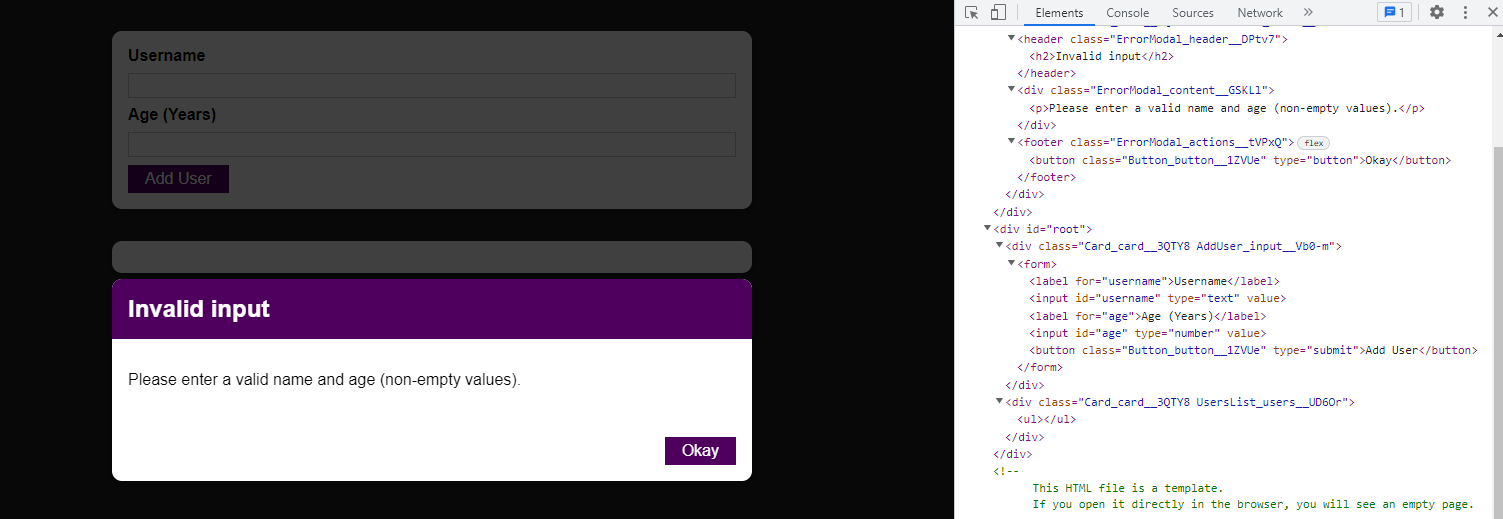
Now we will repeat this for the actual overlay. For that we will add a new expression inside of the ErrorModal next to the expression we just added. Here we use createPortal, but this time we want to render our ModalOverlay component, and my ModalOverlay component needs a bunch of props, which I now of course want to forward. It needs a title, message, and onConfirm.

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import ReactDOM from 'react-dom';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  const ModalOverlay = (props) => {  return <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  };  const ErrorModal = (props) => {  return (  <React.Fragment>  {ReactDOM.createPortal(  <Backdrop onConfirm={props.onConfirm} />,  document.getElementById('backdrop-root')  )}  {ReactDOM.createPortal(  **<ModalOverlay**  **title={props.title}**  **message={props.message}**  **onConfirm={props.onConfirm}**  **/>,**  )}  </React.Fragment>  );  };  export default ErrorModal; |

Now just as with the backdrop, we need to let React DOM know where it should render this HTML content where it needs to be rendered. For that, I'll use document.getElementById and select my overlay-root, which was the other div that I created in "public/index.html."

|  |
| --- |
| src/components/UI/ErrorModal.js |
| import React from 'react';  import ReactDOM from 'react-dom';  import Card from './Card';  import Button from './Button';  import classes from './ErrorModal.module.css';  const Backdrop = (props) => {  return <div className={classes.backdrop} onClick={props.onConfirm} />  };  const ModalOverlay = (props) => {  return <Card className={classes.modal}>  <header className={classes.header}>  <h2>{props.title}</h2>  </header>  <div className={classes.content}>  <p>{props.message}</p>  </div>  <footer className={classes.actions}>  <Button onClick={props.onConfirm}>Okay</Button>  </footer>  </Card>  };  const ErrorModal = (props) => {  return (  <React.Fragment>  {ReactDOM.createPortal(  <Backdrop onConfirm={props.onConfirm} />,  document.getElementById('backdrop-root')  )}  {ReactDOM.createPortal(  <ModalOverlay  title={props.title}  message={props.message}  onConfirm={props.onConfirm}  />,  **document.getElementById('overlay-root')**  )}  </React.Fragment>  );  };  export default ErrorModal; |

And with that saved, if we reload and click "Add User," we get the modal popup. With the modal it works and everything is now rendered in the divs where it should be rendered and no longer nested in our other HTML code.



The idea behind a portal really just is that the rendered HTML content is moved somewhere else. ReactDOM.createPortal can be used anywhere where you would otherwise use JSX code. Of course wrap in curly braces because we're using JavaScript code. So wherever you would normally just use a component, you can use createPortal to move that Component's HTML content somewhere else, only in the actual DOM that is being rendered.

### 105. Working with "ref"s

With fragments and portals we end up with semantically more correct HTML code, which makes your app more accessible and makes sure you don't render too many divs unnecessarily, and in general shows that you're a developer who knows what they are doing.

"ref"s are references but the name in React is just ref, so the short form of reference. In their most basic form, refs allow us to get access to other DOM elements and work with them.

In AddUser we have our inputs.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  **<input**  **id="username"**  **type="text"**  **value={enteredUsername}**  **onChange={usernameChangeHandler}**  **/>**  <label htmlFor="age">Age (Years)</label>  **<input**  **id="age"**  **type="number"**  **value={enteredAge}**  **onChange={ageChangeHandler}**  **/>**  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

And we manage what the user enters by simply keeping track of it.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={**usernameChangeHandler**}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={**ageChangeHandler**}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We simply have our state and with every keystroke we update our state. So with every keystroke, we update the value we get by the user and we store it in our state

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, **setEnteredUsername**] = useState('');  const [enteredAge, **setEnteredAge**] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We feed that state back into the input and then we use that state later on to

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  **<input**  **id="username"**  **type="text"**  **value={enteredUsername}**  **onChange={usernameChangeHandler}**  **/>**  <label htmlFor="age">Age (Years)</label>  **<input**  **id="age"**  **type="number"**  **value={enteredAge}**  **onChange={ageChangeHandler}**  **/>**  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

reset the input but also send it

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  **setEnteredUsername('');**  **setEnteredAge('');**  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

to the place where we need the data.

Updating the state with every keystroke when we only need it when we submit the form seems a bit redundant to me. That's a scenario where refs could help us. Though, refs are not limited to that.

How do refs work? With refs we can setup a connection between an HTML element which is being rendered and our other JavaScript code.

We create a ref which we do with the help of another React hook. We import the useRef hook.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, **useRef** } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We then simply call useRef in our functional component. Like all React hooks, useRef is only useable inside of functional components. useRef is by default undefined. useRef returns a value that allows us to work with that ref later, so which allows us to work with that element to which we are going to connect it. Here, we will name it nameInputRef because I plan on connecting this ref with that first input that allows us to enter a username.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  **const nameInputRef = useRef();**  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

So here we have nameInputRef and we can create another ref by calling useRef again, and that will be my ageInputRef.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  **const ageInputRef = useRef();**  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

So we now have two refs, but they are not doing anything. They're initialized to be undefined because that is the default.

We can let React know that we want to connect a ref to an HTML element by going to that element to which we want to connect the ref and adding a special prop there, the ref prop. Just like the key prop, the ref prop is a built-in prop, which you can add to any HTML element because you can connect any HTML element to one of your references. You will very often do that for inputs because you want to fetch input data, for example, but you can do it with any element. So we have our ref prop and then we set it to {nameInputRef} as a value.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  **ref={nameInputRef}**  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

The nameInputRef is just that constant which stores this first ref.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  **const nameInputRef = useRef();**  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

So with that, I am connecting this ref

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  **const nameInputRef = useRef();**  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

which I created in this component with the JSX code that is being rendered by that same component.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  **ref={nameInputRef}**  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

And now a connection will be established. The first time react reaches this code and renders this code,

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  **<Wrapper>**  **{error && (**  **<ErrorModal**  **title={error.title}**  **message={error.message}**  **onConfirm={errorHandler}**  **/>**  **)}**  **<Card className={classes.input}>**  **<form onSubmit={addUserHandler}>**  **<label htmlFor="username">Username</label>**  **<input**  **id="username"**  **type="text"**  **value={enteredUsername}**  **onChange={usernameChangeHandler}**  **ref={nameInputRef}**  **/>**  **<label htmlFor="age">Age (Years)</label>**  **<input**  **id="age"**  **type="number"**  **value={enteredAge}**  **onChange={ageChangeHandler}**  **/>**  **<Button type="submit">Add User</Button>**  **</form>**  **</Card>**  **</Wrapper>**  );  };  export default AddUser; |

It will actually set the values stored in nameInputRef to the native DOM element that is

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  **const nameInputRef = useRef();**  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

rendered based on this input.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  **<input**  **id="username"**  **type="text"**  **value={enteredUsername}**  **onChange={usernameChangeHandler}**  **ref={nameInputRef}**  **/>**  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

What will end up inside of nameInputRef in the end will really be a real DOM element later.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={**nameInputRef**}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

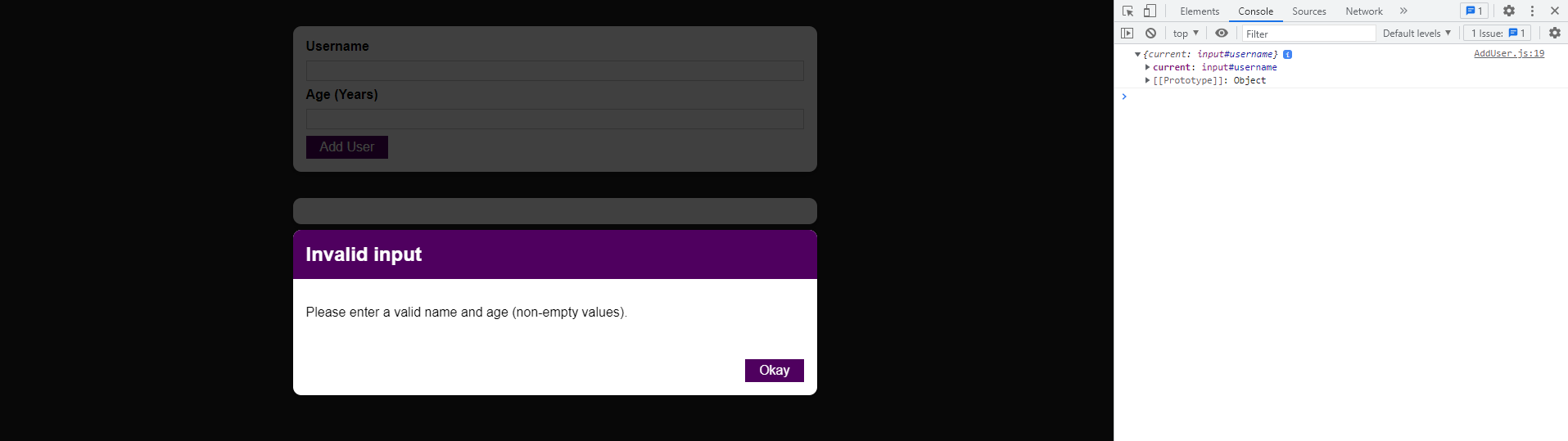
We will do the same for ageInputRef.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  **ref={ageInputRef}**  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We can try console logging our refs.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  **console.log(nameInputRef);**  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We see that there is an object being output and that object has a current property.



This nameInputRef being generated here always is an object, which always has a current prop, and the current prop holds the actual value that ref is connected with. By default it is undefined but

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  **const nameInputRef = useRef();**  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  console.log(nameInputRef);  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

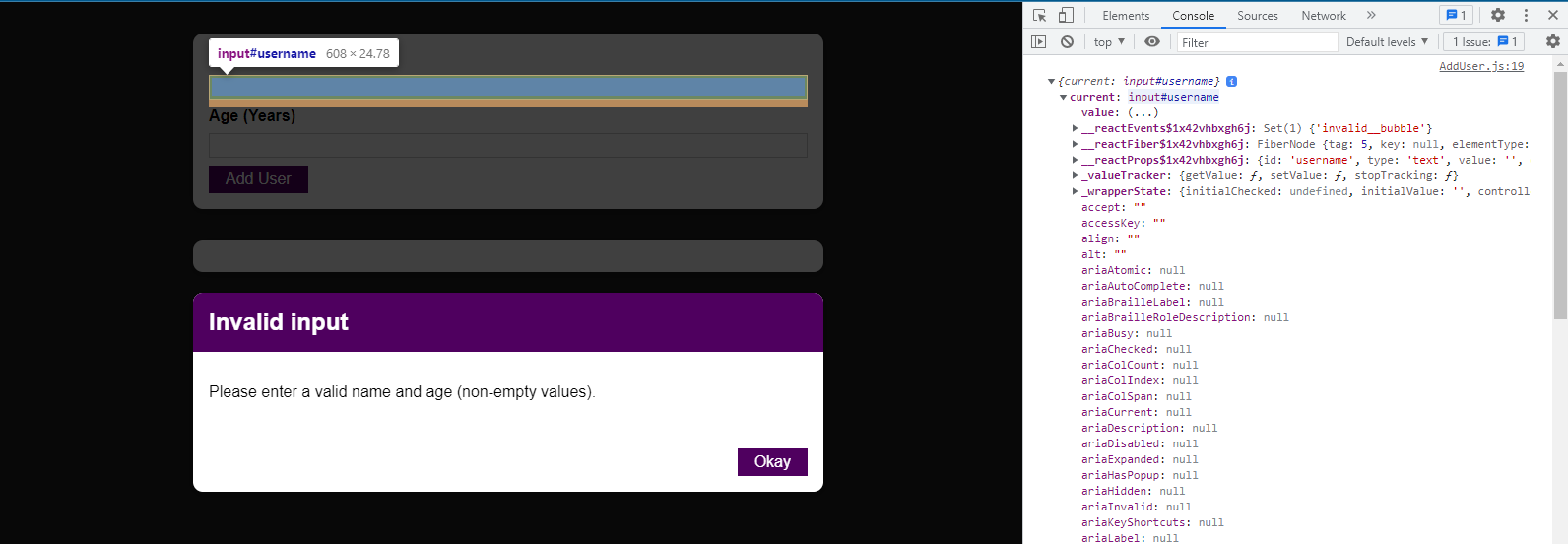
as soon as this code ran

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  console.log(nameInputRef);  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  **<Wrapper>**  **{error && (**  **<ErrorModal**  **title={error.title}**  **message={error.message}**  **onConfirm={errorHandler}**  **/>**  **)}**  **<Card className={classes.input}>**  **<form onSubmit={addUserHandler}>**  **<label htmlFor="username">Username</label>**  **<input**  **id="username"**  **type="text"**  **value={enteredUsername}**  **onChange={usernameChangeHandler}**  **ref={nameInputRef}**  **/>**  **<label htmlFor="age">Age (Years)</label>**  **<input**  **id="age"**  **type="number"**  **value={enteredAge}**  **onChange={ageChangeHandler}**  **ref={ageInputRef}**  **/>**  **<Button type="submit">Add User</Button>**  **</form>**  **</Card>**  **</Wrapper>**  );  };  export default AddUser; |

because of this ref prop,

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  **const nameInputRef = useRef();**  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  console.log(nameInputRef);  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={**nameInputRef**}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

the nameInputRef is connected to that input, and hence it is actually the input, which is being stored as a value in the current prop. What’s being stored here really is the actual DOM node, which you could now manipulate and do all kinds of things with. It is recommended that you don’t manipulate it. Your DOM should only really be manipulated by React. You’re using React to let it do all the heavy lifting. Reading data from the input doesn’t sound too bad because you’re not changing anything with that. You’re just reading data.



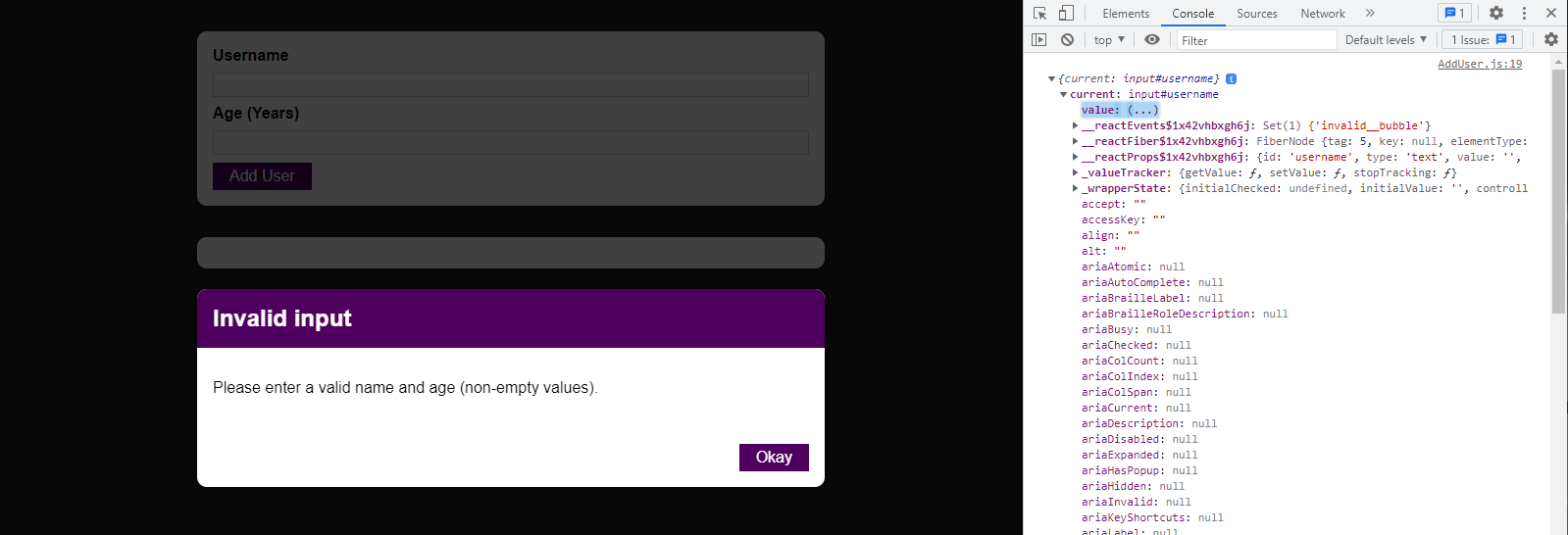
So here, instead of logging this,

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  **console.log(nameInputRef);**  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

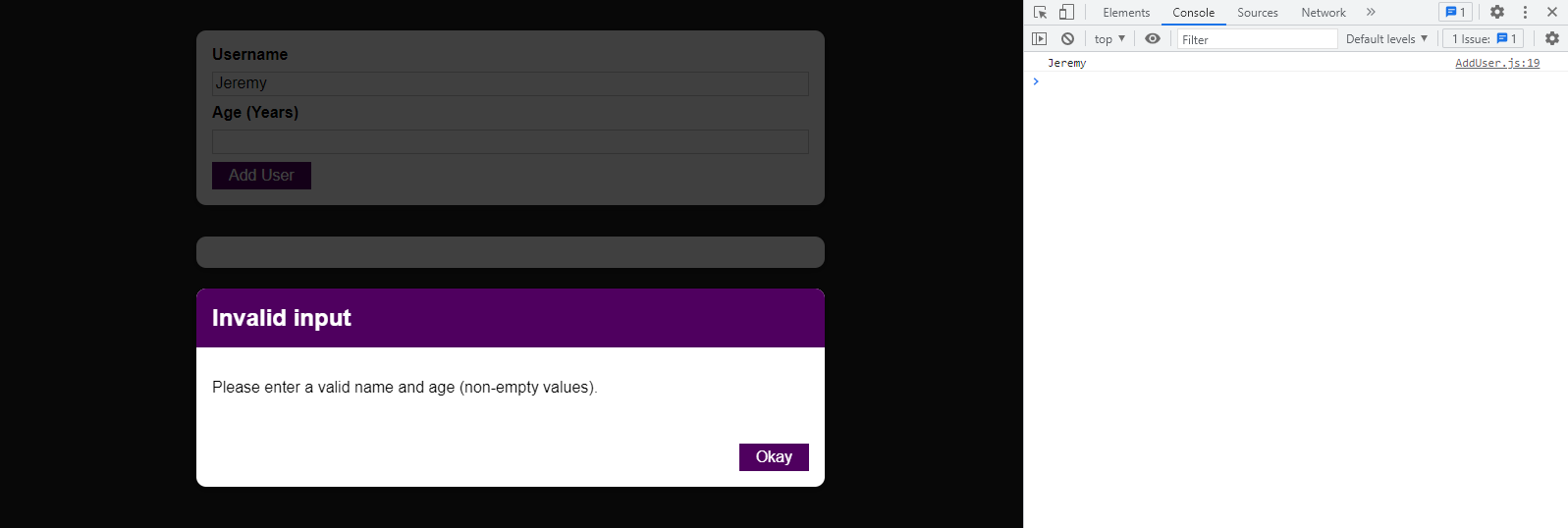
we can, of course, read current.value. Now current refers to the value stored and the value stored is the input element and every input element has a value property in JavaScript.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  console.log(nameInputRef**.current.value**);  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We can actually see the value property here:



If we now save our code, reload the browser and enter “Jeremy” here, we see “Jeremy” logged to the console:



That means we can get access to the values stored in the element without having to log every keystroke. We don’t need state for this. We can just read it when the submit button is pressed.

That means we can now replace console.log maybe with a helper constant, which we use inside of addUserHandler. We’ll name the constant “enteredName” and simply store my nameInputRef.current.value inside of that.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  **const enteredName =** nameInputRef.current.value;  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We can do the same for the age. We can store ageInputRef.current.value inside of a constant called enteredUserAge.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  **const entereUserdAge = ageInputRef.current.value;**  if (enteredUsername.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

Now we can check enteredName here:

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (**enteredName**.trim().length === 0 || enteredAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

and check enteredUserAge here:

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || **enteredUserAge**.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

And we also check enteredUserAge here in this condition, so always the values we retrieve from the refs.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+**enteredUserAge** < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredUsername, enteredAge);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We also use enteredName and enteredUserAge here to submit it to onAddUser.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredUserAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(**enteredName**, **enteredUserAge**);  setEnteredUsername('');  setEnteredAge('');  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We should no longer reset these inputs by resetting the state here because we’re not using the state to get our values anymore.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredUserAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredName, enteredUserAge);  **setEnteredUsername(''); // should no longer be resetting this input**  **setEnteredAge(''); // should no longer be resetting this input**  };  const usernameChangeHandler = (event) => {  setEnteredUsername(event.target.value);  };  const ageChangeHandler = (event) => {  setEnteredAge(event.target.value);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

We still have those listeners but we actually don’t really need those values anymore because we are using refs to get the values.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [enteredUsername, setEnteredUsername] = useState('');  const [enteredAge, setEnteredAge] = useState('');  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredUserAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredName, enteredUserAge);  setEnteredUsername(''); // should no longer be resetting this input  setEnteredAge(''); // should no longer be resetting this input  };  **const usernameChangeHandler = (event) => {**  **setEnteredUsername(event.target.value);**  **};**  **const ageChangeHandler = (event) => {**  **setEnteredAge(event.target.value);**  **};**  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  value={enteredUsername}  onChange={usernameChangeHandler}  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  value={enteredAge}  onChange={ageChangeHandler}  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

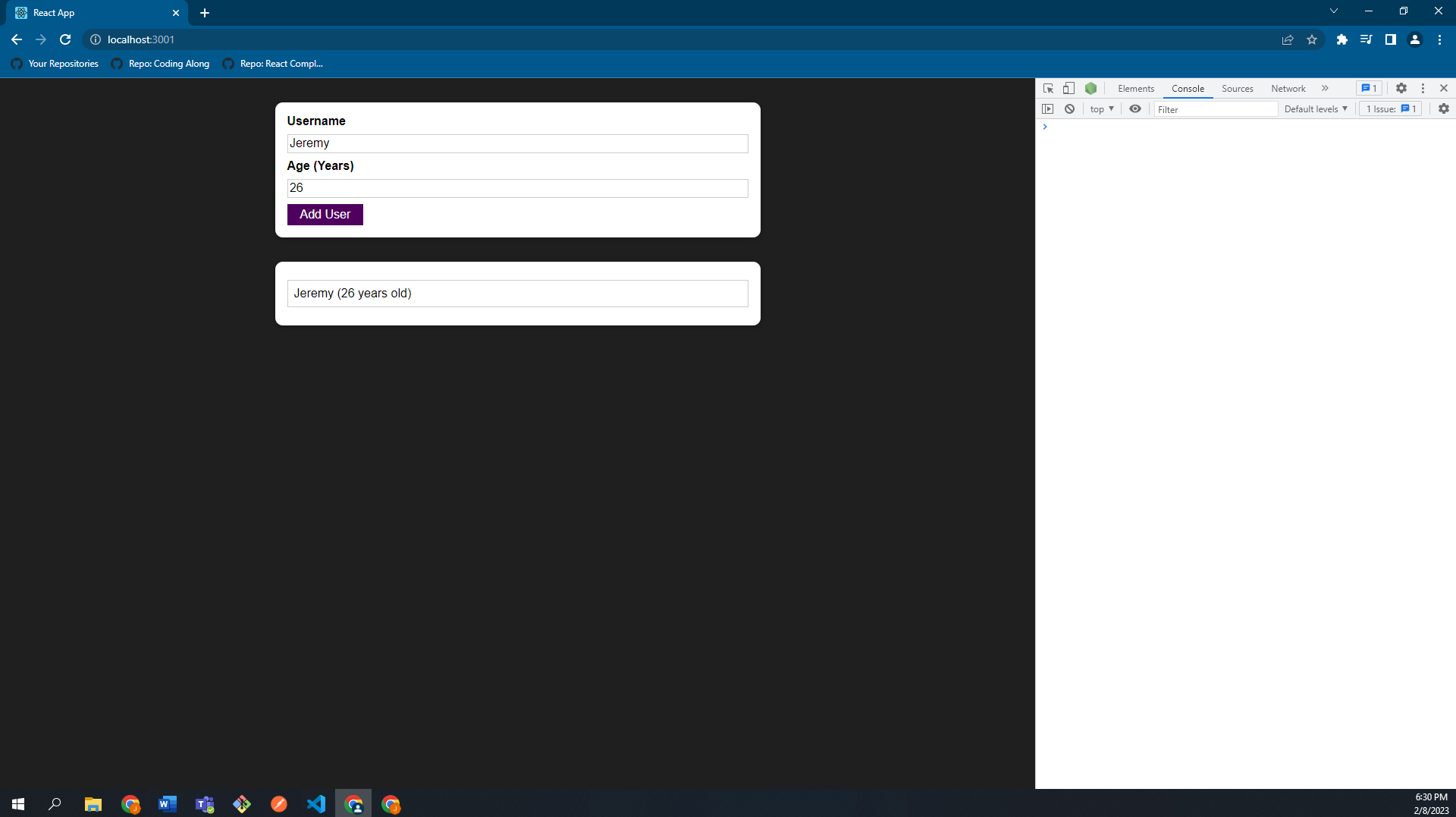
So we can get rid of these states where we listen to every keystroke, get rid of the state updating functions, and get rid of the two handlers. Also, get rid of the value property and the onChange property.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  ~~const [enteredUsername, setEnteredUsername] = useState('');~~  ~~const [enteredAge, setEnteredAge] = useState('');~~  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredUserAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredName, enteredUserAge);  ~~setEnteredUsername(''); // should no longer be resetting this input~~  ~~setEnteredAge(''); // should no longer be resetting this input~~  };  ~~const usernameChangeHandler = (event) => {~~  ~~setEnteredUsername(event.target.value);~~  ~~};~~  ~~const ageChangeHandler = (event) => {~~  ~~setEnteredAge(event.target.value);~~  ~~};~~  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  ~~value={enteredUsername}~~  ~~onChange={usernameChangeHandler}~~  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  ~~value={enteredAge}~~  ~~onChange={ageChangeHandler}~~  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

After getting rid of state-related items in AddUser:

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredUserAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredName, enteredUserAge);  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

With that, we’ve shortened up the code quite a bit, and we’re relying on refs to read the values. Hence, if I reload the browser and add “Jeremy” and “26”, we can still add that user.



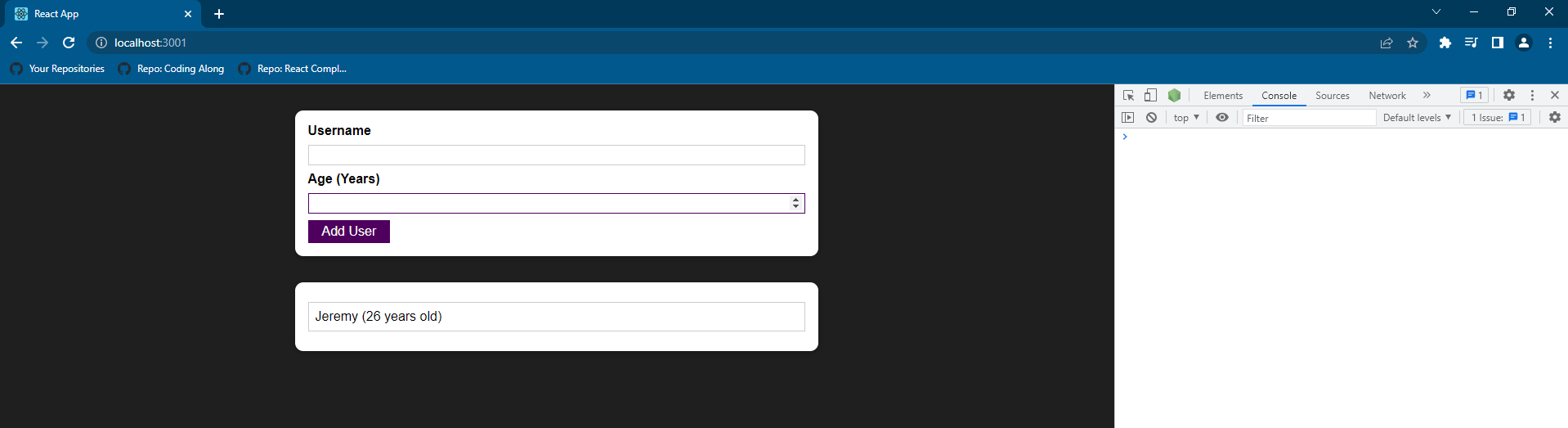
During the process we lost our resetting logic. To bring the resetting logic back, we have two options. We can switch back to the state-based solution, which is not bad but we want to show refs here. Or we can do something which we rarely do but is okay here in the context of an input field value which you want to reset. You can manipulate the DOM without react, and yes, he said that you typically shouldn’t do that, but if it is just resetting the value entered by the user, it is something you can consider doing. We can do the following to reset the values:

nameInputRef.current.value = '';

ageInputRef.current.value = '';

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredUserAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredName, enteredUserAge);  **nameInputRef.current.value = '';**  **ageInputRef.current.value = '';**  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  <input  id="username"  type="text"  ref={nameInputRef}  />  <label htmlFor="age">Age (Years)</label>  <input  id="age"  type="number"  ref={ageInputRef}  />  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

If we now save this and reload our browser, the inputs will be cleared.



Again, we should rarely use refs to manipulate the DOM. Here we’re not really manipulating a DOM. We’re not adding a new element or changing a CSS class. We’re just changing what the user entered. You could always argue to not do that and change back to using the state-based solution.

In general, the question is not whether refs or state is better. You can use either of the two. You will sometimes have use cases where you want to just quickly read a value, for example. And if you only want to read a value and never plan on changing anything, then you don’t really need state because just using state as a keylogger is not that great. It’s a lot of unnecessary code and work. So if you just want to read a value, refs are probably better.

### 106. Controlled vs Uncontrolled Components

This approach of using refs to interact with DOM elements, specifically with input elements, also has a special name. The approach of accessing values with a ref is known as an uncontrolled component. The below inputs (bolded in the below code snippet) would be considered uncontrolled components.

|  |
| --- |
| src/components/Users/AddUser.js |
| import React, { useState, useRef } from 'react';  import Card from '../UI/Card';  import Button from '../UI/Button';  import ErrorModal from '../UI/ErrorModal';  import Wrapper from '../Helpers/Wrapper';  import classes from './AddUser.module.css';  const AddUser = (props) => {  const nameInputRef = useRef();  const ageInputRef = useRef();  const [error, setError] = useState();  const addUserHandler = (event) => {  event.preventDefault();  const enteredName = nameInputRef.current.value;  const enteredUserAge = ageInputRef.current.value;  if (enteredName.trim().length === 0 || enteredUserAge.trim().length === 0) {  setError({  title: 'Invalid input',  message: 'Please enter a valid name and age (non-empty values).',  });  return;  }  if (+enteredUserAge < 1) {  setError({  title: 'Invalid age',  message: 'Please enter a valid age (> 0).',  });  return;  }  props.onAddUser(enteredName, enteredUserAge);  nameInputRef.current.value = '';  ageInputRef.current.value = '';  };  const errorHandler = () => {  setError(null);  };  return (  <Wrapper>  {error && (  <ErrorModal  title={error.title}  message={error.message}  onConfirm={errorHandler}  />  )}  <Card className={classes.input}>  <form onSubmit={addUserHandler}>  <label htmlFor="username">Username</label>  **<input**  **id="username"**  **type="text"**  **ref={nameInputRef}**  **/>**  <label htmlFor="age">Age (Years)</label>  **<input**  **id="age"**  **type="number"**  **ref={ageInputRef}**  **/>**  <Button type="submit">Add User</Button>  </form>  </Card>  </Wrapper>  );  };  export default AddUser; |

They are considered uncontrolled because they’re internal state, so the value that is reflected in them is not controlled by react. We rely on the default behavior of the input where of course a user is able to enter something, and that entered value is reflected. And then we just fetch it, with a react feature, but we don’t feed data back into the inputs.

When I set a new value to those inputs using the following:

nameInputRef.current.value = '';

ageInputRef.current.value = '';

with this workaround, we’re absolutely not using react here. We are using the ref, but in the end, we get access to the native DOM element with that and then I just use the regular DOM api for setting the value of a dom node of an input dom node. That is why the input is considered uncontrolled because we’re not controlling the state of the input element with react.

You can talk about uncontrolled and controlled components also in the context of other components, but most commonly, we have this scenario when we talk about input components, about form components in general, because those components tend to have some internal state natively by the browser, an input element is configured to take user input and save and reflect it. And when we then work with those components in a react app, we want to connect our react state to that. That’s why we typically have this controlled vs. uncontrolled thing when we work with input components in react.

The approach where we manage our state and update that state on every keystroke and we feed that state back into the input with that state back into the input with the value prop. This is the controlled approach. We would then say that those input components are controlled components because their internal state is controlled by react.

## Section 10: Advanced: Handling Side Effects, Using Reducers & Using the Context API

### 108. Module Introduction

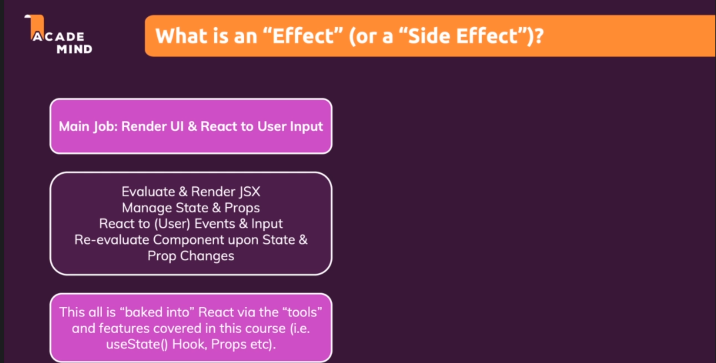
We’re going to take a look at 3 very important concepts that you need to know as a react developer: Effects, Reducers & Context. These features are more advanced (does not mean they are difficult), meaning that you need to know the concepts that came before this.

This module will work with what effects are and how you can work with side effects. We will look at Managing more Complex State with Reducers.

We will also look at Managing App-Wide or Component-Wide State with Context (which is a concept built into react to make sharing state and state updates across components easier).

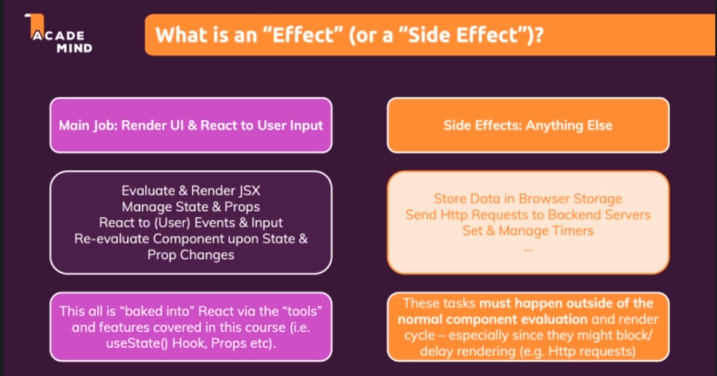
### 109. What are “Side Effects” & Introducing useEffect

Our components in our React app and our React app as a whole and also the react library itself has one main job to Render the UI & React to user input and re-render the UI when it’s needed.

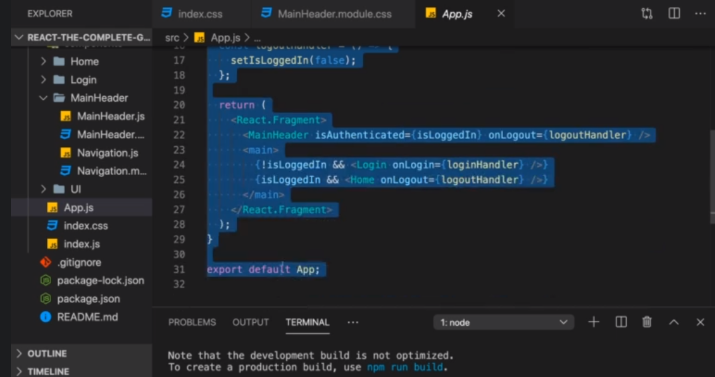


What is an “Effect” (or a “Side Effect”)?

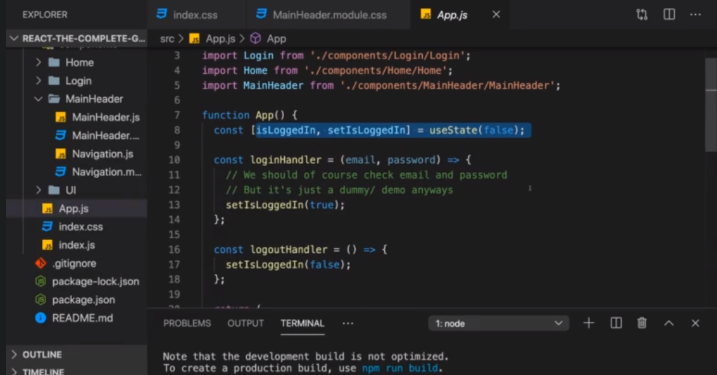
Side Effects are anything else that might be happening in your application. A good example is sending an HTTP request or storing something in browser storage, in local storage, for example. You could also think about timers or intervals which you set in your code. These are all tasks that you often consider in your application. These tasks are all not related to bringing something onto the screen. At least not directly. You may might be sending an http request to then draw something onto the screen once you get the response but sending the request itself and handling potential errors and so on is not something you need React for. It’s not something react cares about. These tasks must happen outside of the normal component evaluation, so outside of your normal component function.



If we were to send in an HTTP request here, then we would send this request whenever



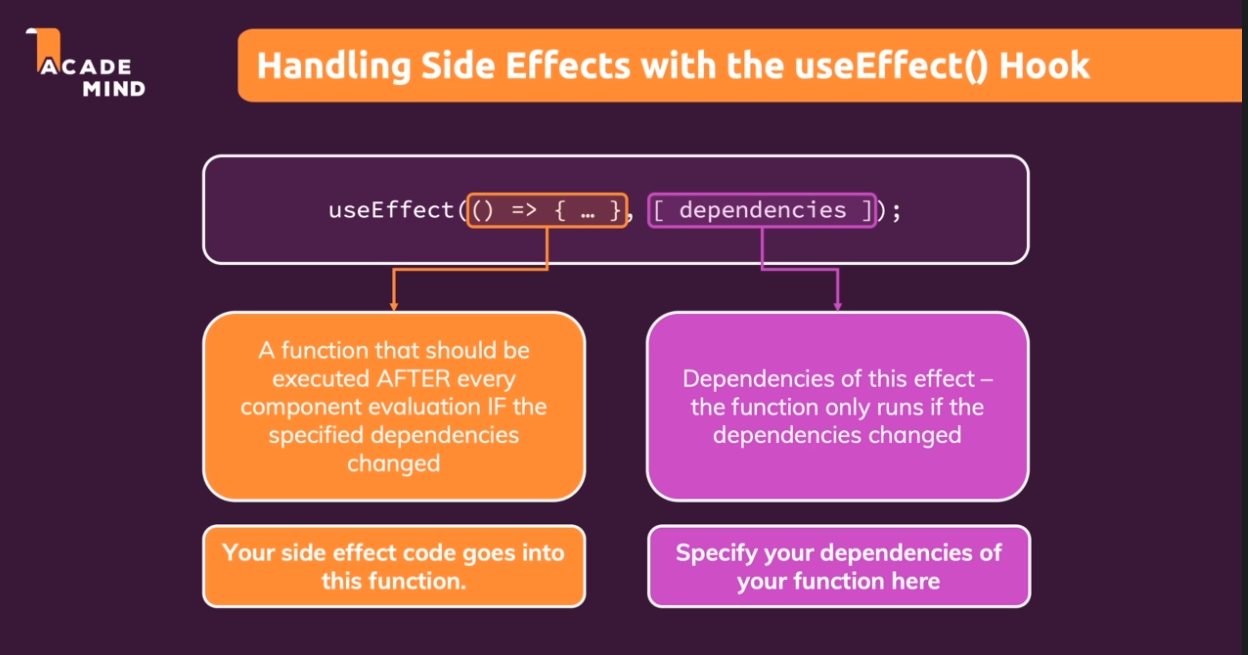
this state changes. That might sometimes be what you want but definitely not always. And if in response to your HTTP request, you for example, eventually change some state, you would even create an infinite loop because



you would send the request whenever the component re-renders and in response to the request you change some state which triggers the function again.

Therefore, such side-effects should not go directly into this App component function because it would most likely create bugs, infinite loops, or simply send too many HTTP requests.

Therefore, we have a better tool for handling side effects, and that’s a special react hook called useEffect(). The useEffect hook is simply another built-in hook, so another function that you can run inside of your component function that will do something special. The useEffect hook is called with two parameters.

The first argument is a function that should be executed AFTER every component evaluation IF the specified dependencies changed. The specified dependencies are the second argument you pass in. That’s an array full of dependencies and whenever such a dependency changes, that first function will re-run. Therefore, in that first function you can put in any side effect code and that code will then only execute when the dependencies specified by you change and not when the component re-renders, only when your dependencies change. 

### 110. Using the useEffect()Hook

We have an app where a user has to enter a valid username and at least 7 characters for the password in order to click login. When the user clicks login, they are “logged in”, but if you reload the page, you always lose the login status. That is probably not something you want.

In reality, when you login, a request would be sent to the backend and get back some login data, for example, some token which identifies the user as authenticated. We will cover more on that in the authentication module.

But you want to make sure that this user’s authenticated status is also still there once you reload the page. At the moment we lose this because in the “App.js” file where I manage this isLoggedIn state, it’s just managed as some react state. Therefore, it’s just in the end managed by some JavaScript variable behind the scenes in React. Now, the nature of that is when you reload your application, your entire react script restarts and all variables from the last execution are lost. That’s how the web and scripts and the browser works. This is nothing React specific.

|  |
| --- |
| src/App.js |
| import React, { useState } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  **const [isLoggedIn, setIsLoggedIn] = useState(false);**  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

Since we lose all the data when this restarts, it would be nice to store it somewhere where it persists the reload. And even better than that, we also want to make sure that whenever this app does start, we check if the data was persisted. And if it is, we log the user in automatically, so that the user doesn’t need to re-enter the details.

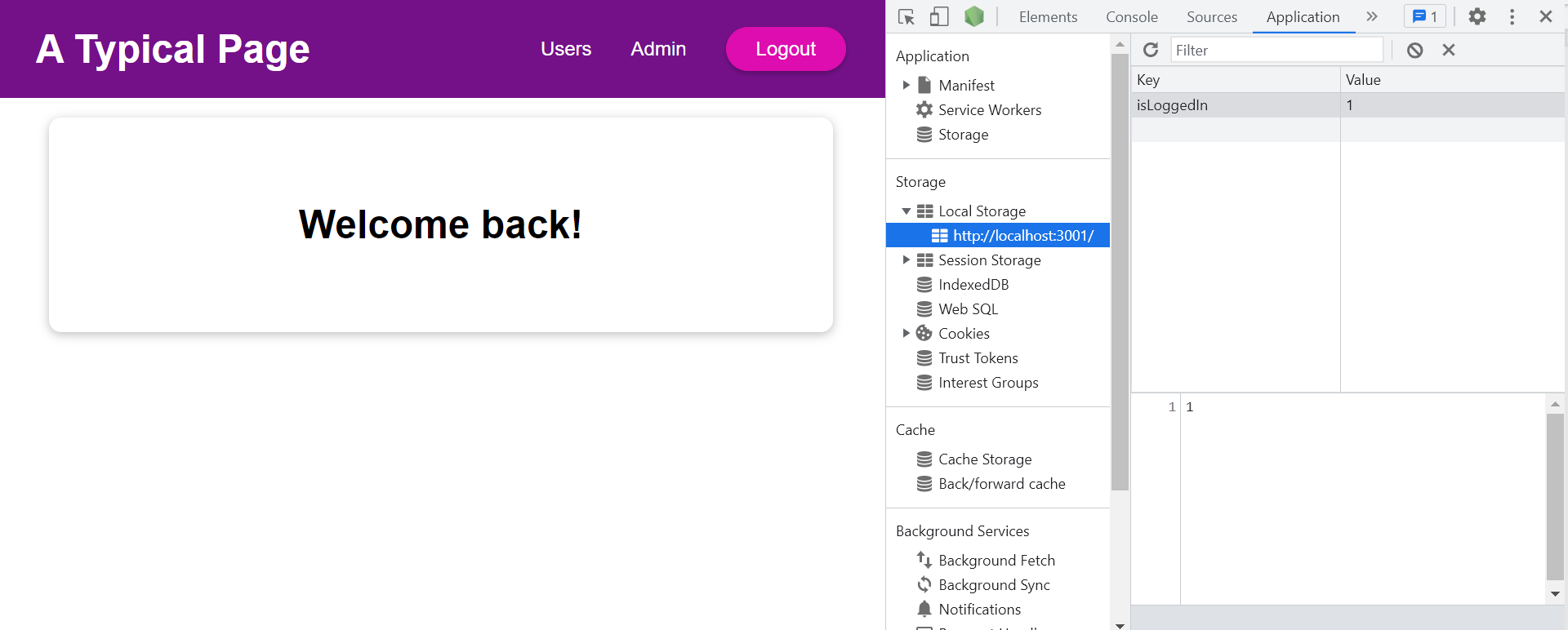
And that’s where we can use useEffect. Now let’s start with storing the data. Here in the loginHandler, I pass true to setIsLoggedIn, which sets isLoggedIn to true. And that’s where I want to store that piece of information in the browser storage. Now the browser has multiple storages that we can use. The most common storage for this would be cookies or local storage, and since it’s particularly easy to work with, we will use local storage. So that’s a storage mechanism built into the browser totally independent of React.

|  |
| --- |
| src/App.js |
| import React, { useState } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  **setIsLoggedIn(true);**  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

Here in the login handler, we could therefore run localStorage. localStorage is a global object that is available in the browser. We can do localStorage.setItem and then give the item any identifier of our choice, like for example, 'isLoggedIn', but that is totally up to you. It should just be a string, though. And the second argument should also be a string, but which you store. And for example, that could be '1' to signal that the user is logged in. '0' could stand for not logged in. Of course, you could also have some identifier like this 'LOGGED\_IN', but this is totally up to you. We will work with '1' and '0'.

|  |
| --- |
| src/App.js |
| import React, { useState } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  **localStorage.setItem('isLoggedIn', '1');**  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

If I save this and reload everything, if I do login again, we can open up the developer tools -> click the Application tab, there you'll find the Storage section with the Local Storage, if you expand Local Storage, you will find your host, and there you should now see the key value pair, which are the two arguments to localStorage.setItem.



We have this line localStorage.setItem('isLoggedIn', '1'); in the loginHandler because it is a function that executes only when the user clicks the button, which is rare enough and is exactly when we want to store something. This is a use case where we don't need useEffect necessarily.

But how about the scenario when the app restarts because the user left the page and comes back or simply because we reload the page? We then want to check if in local storage we have that key value pair. When the app restarts, the app component function runs again. Now therefore, here of course in the App functional component, we could reach out to localStorage, call getItem, and search for 'isLoggedIn', which will return the items stored there, and we could set this to storedUserLoggedInInformation.

|  |
| --- |
| src/App.js |
| import React, { useState } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

And we could check if the storedUserLoggedInInformation is equal to '1', and if it is equal to '1', we could call setIsLoggedIn and pass in true, so that we set the user to logged in even without the loginHandler function being triggered. And the loginHandler would trigger upon pressing the login button normally just because we know that the user is logged in. We should do this after calling useState. The disadvantage of this approach is that we would create an infinite loop because we check if storedUserLoggedInInformation is stored, if it is stored, we pass true to setIsLoggedIn, which sets isLoggedIn to true. Whenever we call a state-setting function, the entire functional component re-executes. Therefore, the if check would run, would render App again, would find 1, would set it again, and so on.

|  |
| --- |
| src/App.js |
| import React, { useState } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

Therefore, we need useEffect here because that allow us to control when this runs.

|  |
| --- |
| src/App.js |
| import React, { useState } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

Hence, from react, besides useState, we import useEffect.

|  |
| --- |
| src/App.js |
| import React, { useState, **useEffect** } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

And then here, we can simply call useEffect();

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  **useEffect();**  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

And now we pass two arguments – the first argument is a function, for example, here we will use an anonymous arrow function. It doesn't have to be an anonymous arrow function but often you use an anonymous arrow function. The second argument is an array of dependencies.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  **useEffect(() => {}, []);**  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

We can move the code that is here:

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  useEffect(() => {}, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

into the anonymous arrow function. We can now run this code in useEffect that we don't want to run directly in the component function.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

Because now this function here is executed by react and it is executed after every component re-evaluation. So after the App component runs, this anonymous arrow function in useEffect will run.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(**() => {**  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  **}**, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

If you then update the state in the anonymous arrow function, the component will run again, but it will only run after a component re-evaluation (component re-render) if

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  **setIsLoggedIn(true);**  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

the dependencies here changed. Now when the app starts for the first time, that will be the case. If the App component function runs for the very first time because your app just started, then the dependencies are considered to have changed because you had no dependencies before you could say. But once the app ran for the first time with this setup here ([]), we have no dependencies of course, but they also didn't change compared to the first execution cycle. So therefore,

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, **[]**);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

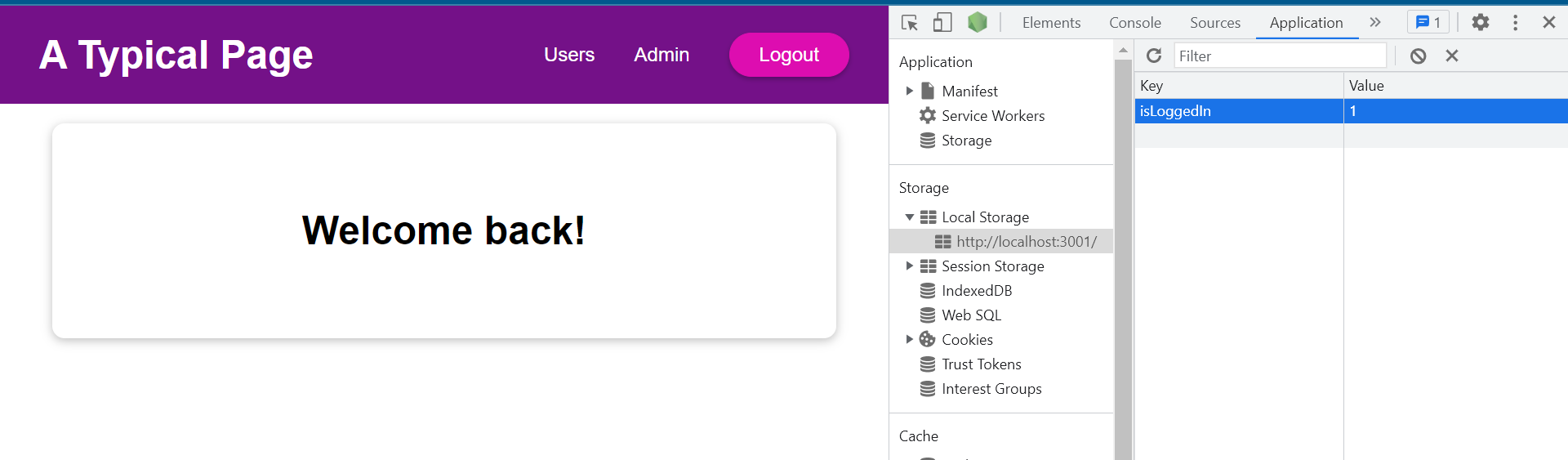
this anonymous function here would indeed only run once, when the app starts, because thereafter the dependencies never change because

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(**() => {**  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  **}**, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

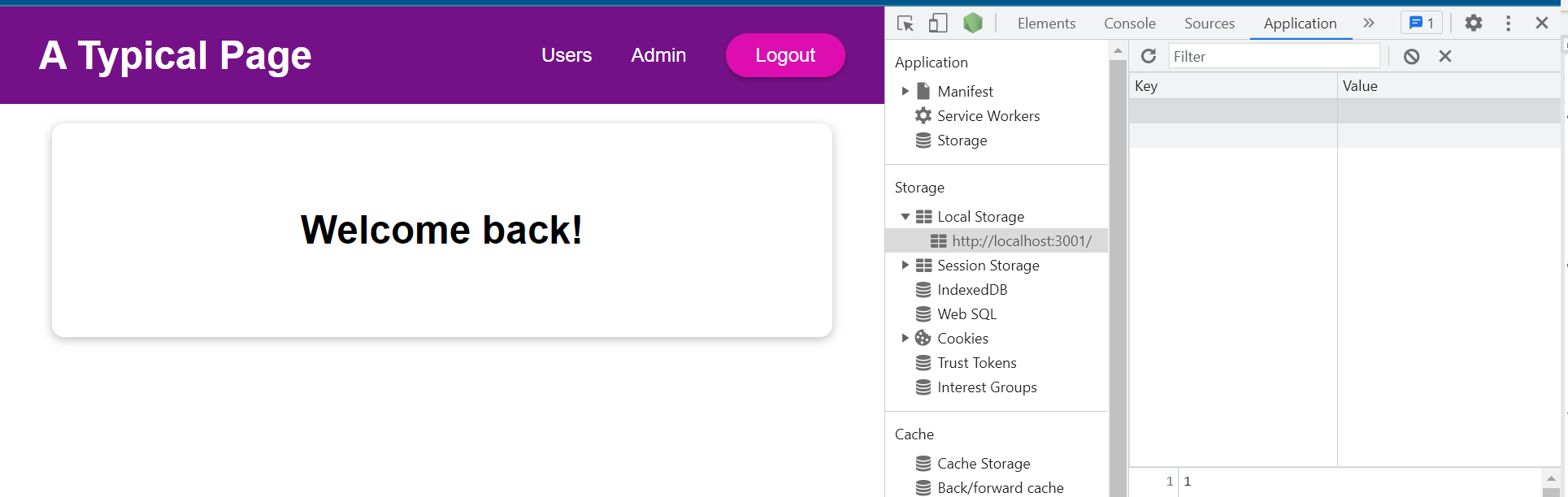
this here specifically has no dependencies.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, **[]**);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

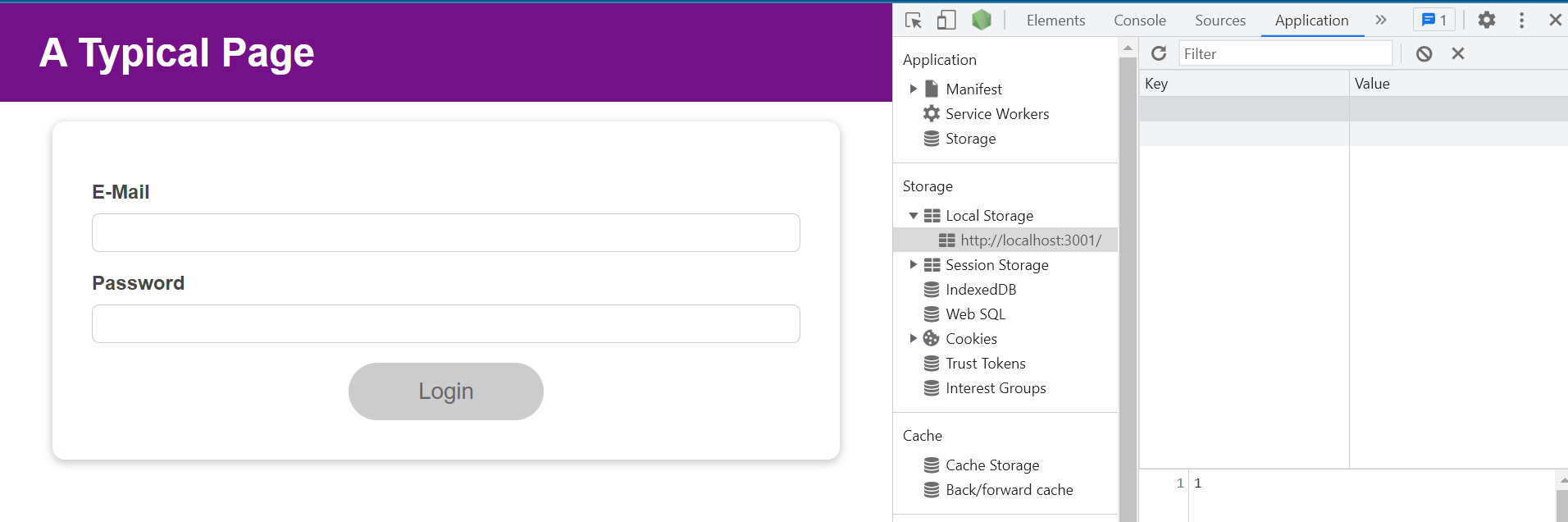
If we save and reload, we should see that we are logged in. We don't end up on the authentication screen but in the logged in state. The reason for that is that we have isLoggedIn stored.



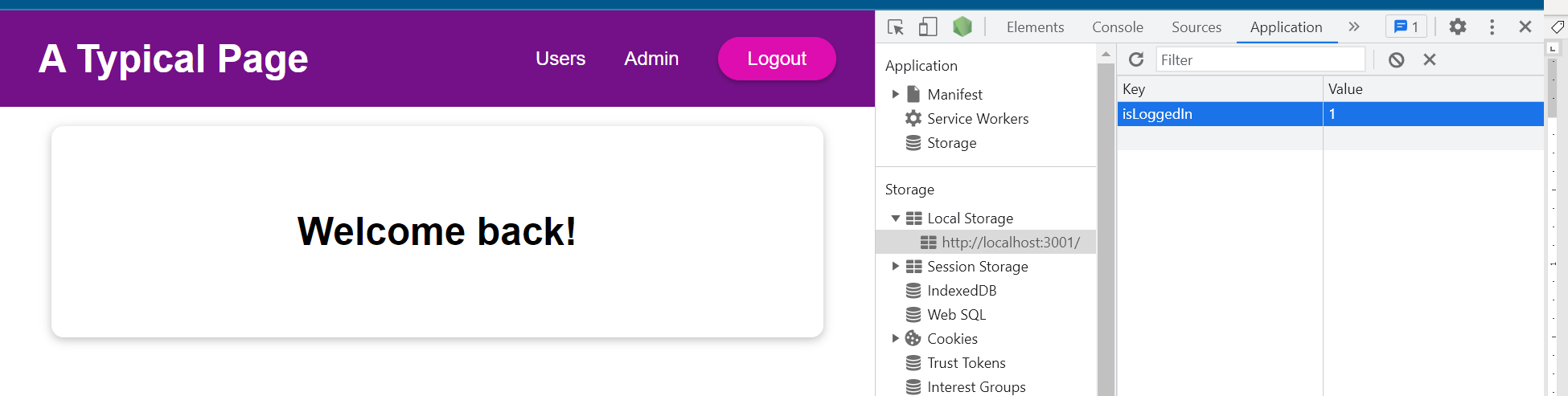
If we were to clear the key and value manually, and



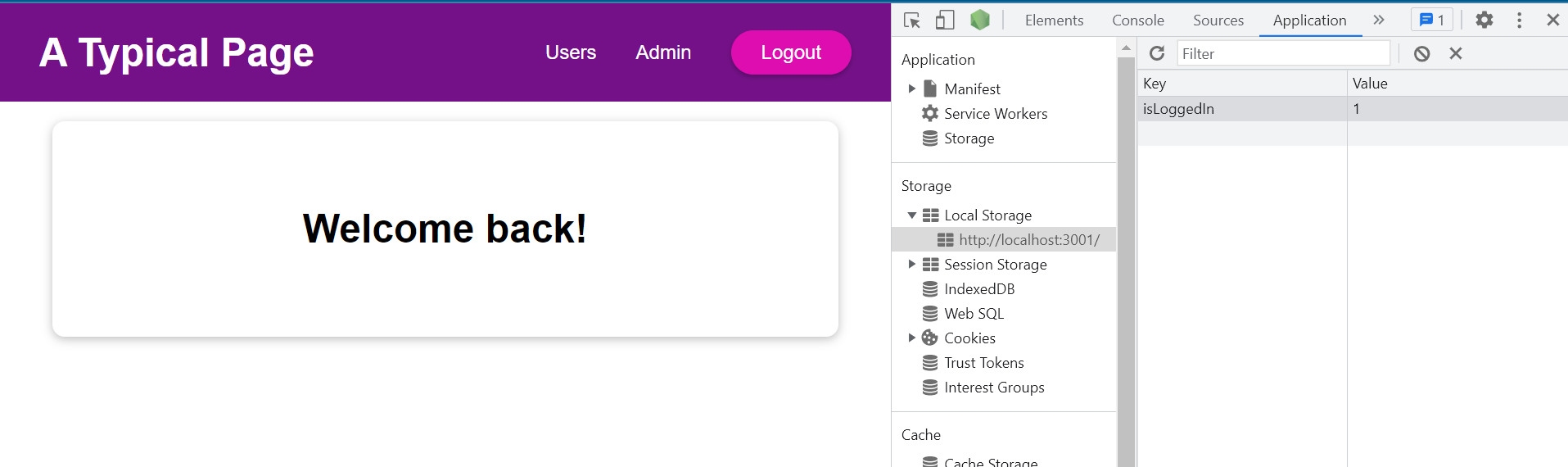
we reloaded the page, we would be back at the login screen.



If we login again, we will see the same behavior as before



If we reload the page, we will stay logged in.



The data fetching is a side effect. It's not directly related to the UI. Of course, the result is, but not the data storage access itself. And we want to run it as a side effect with useEffect in this case to avoid an infinite loop and to make sure that this line of code:

const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');

which could also be performance intensive does not run for every component re-render cycle but only if we wanted to run.

We can refine our data storage example here to make sure that when we click the logout button, we again reach out to localStorage and remove the 'isLoggedIn' key there, so that we don't have to clear it manually.

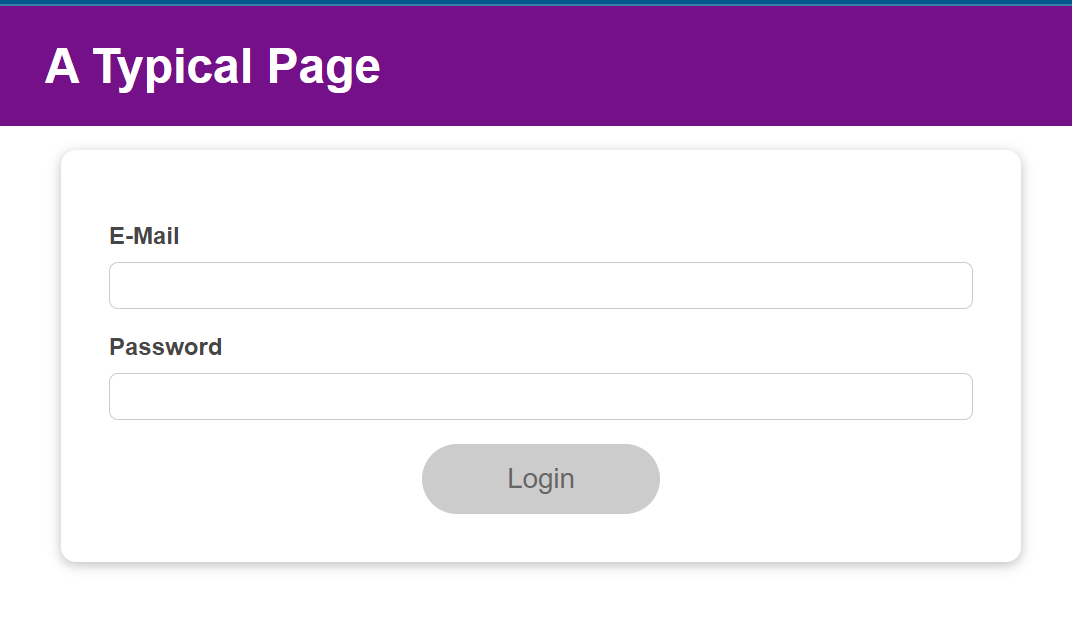
|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  **localStorage.removeItem('isLoggedIn');**  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

### 111. useEffect & Dependencies

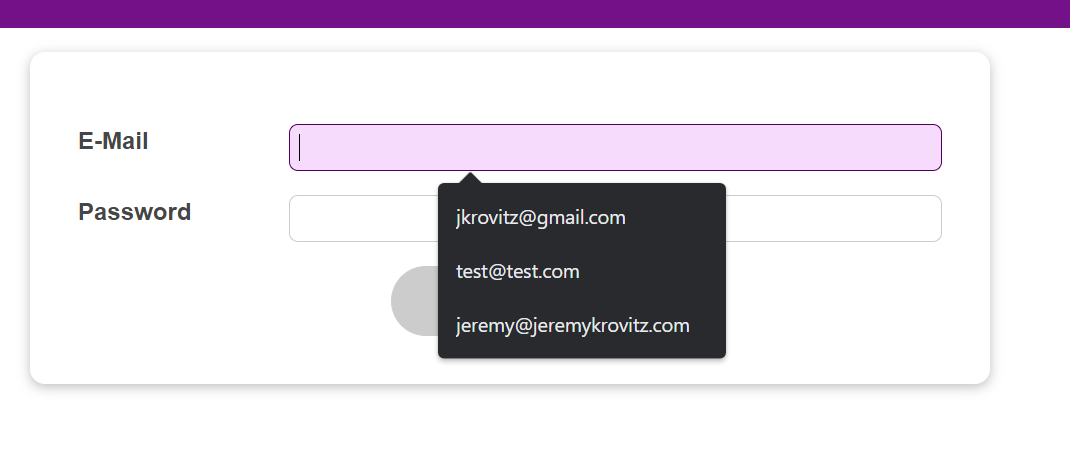
Now often you need dependencies when running useEffect because you don 't just want to run this effect function once when the app starts up but indeed, after every component re-evaluation if and only if a certain dependency changed.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(**() => {**  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  **}**, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

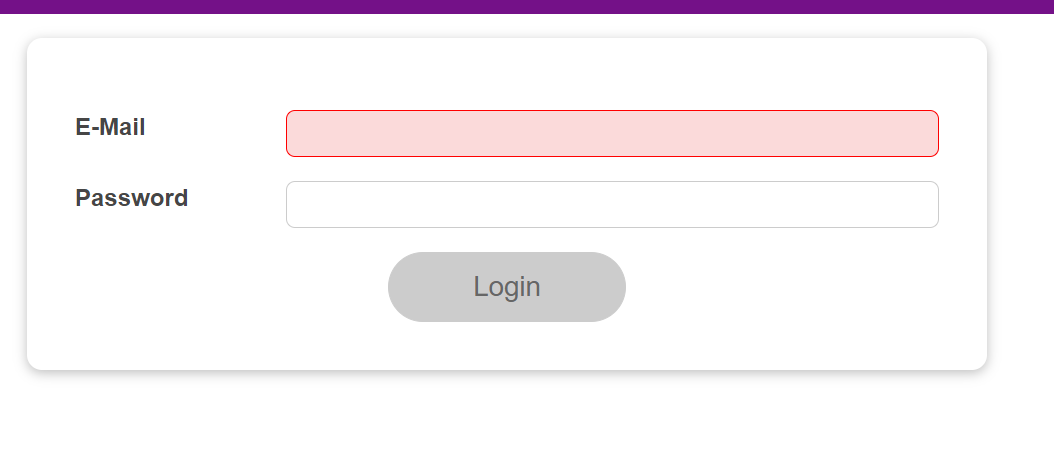
We see an example in the Login component. That's the component where I actually render this form.



There is some validation built-in. If I click into an input, it is marked blue.



If we click out, it is marked red.



Where could we use useEffect? Well, we could use it to restructure our validation logic here.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  **setFormIsValid(**  **event.target.value.includes('@') && enteredPassword.trim().length > 6**  **);**  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

instead of having essentially the same logic just swapped basically,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  **setFormIsValid(**  **event.target.value.trim().length > 6 && enteredEmail.includes('@')**  **);**  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

In the emailChangeHandler and the passwordChangeHandler we could use useEffect to have one place where we mark the form as valid or invalid, with one piece of logic which should trigger whenever either the email or password changed. And that's where we will need extra dependency. Therefore, we can again import useEffect.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, **useEffect** } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

After our state definitions we can call useEffect and

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  **useEffect();**  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

pass the first function to it and have our array of dependencies:

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **}**, **[]**);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

In the function passed into useEffect I will call setFormIsValid, removing it from where it is currently.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **setFormIsValid(**  **event.target.value.includes('@') && enteredPassword.trim().length > 6**  **);**  }, []);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  ~~setFormIsValid(~~  ~~event.target.value.includes('@') && enteredPassword.trim().length > 6~~  ~~)~~;  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

But of course we will not use event.target.value here.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  **~~event.target.value~~**.includes('@') && enteredPassword.trim().length > 6  );  }, []);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We will check if the enteredEmail includes the '@' symbol and that the enteredPassword is longer than six characters.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  **enteredEmail**.includes('@') && **enteredPassword**.trim().length > 6  );  }, []);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

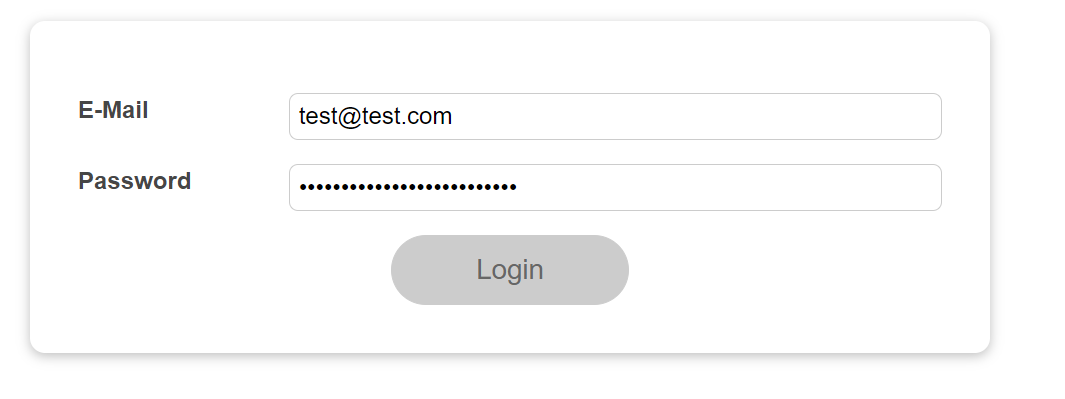
Like this the function inside of useEffect would only run once, and that would be when the Login component is first rendered.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => **{**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  }, []);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  event.target.value.trim().length > 6 && enteredEmail.includes('@')  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Therefore, if I remove that logic from the passwordChangeHandler and the emailChangeHandler:

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, []);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);    **// setFormIsValid has already been removed from this function**  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  **~~setFormIsValid(~~**  **~~event.target.value.trim().length > 6 && enteredEmail.includes('@')~~**  **~~);~~**  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We can reload the page and type in the input fields and the "Login" button never gets enabled:



The reason for that is this effect only ran once when the Login component rendered for the first time. This was the correct behavior in our App component with our isLoggedIn state

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {    const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**  **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

but is not the correct behavior in the Login component. We want to be able to re-evaluate Login and rerun this form validation state setting function for every keystroke in emailChangeHandler and passwordChangeHandler.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  }, []);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We could remove the dependencies array. Do not do that. If you had useEffect without the dependencies array, the function in useEffect would run whenever the Login component was re-evaluated. Therefore, this would be the same behavior as if we put setFormIsValid into the component itself and remove it from useEffect. Now, without any dependencies this code reruns whenever the component is re-rendered, and since we set state in here, this would trigger a re-render cycle itself; hence, we would have an infinite loop.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}**);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So saying nothing about the dependencies is not what we want but no dependencies is also not what we want because then the function would only run once.

You add as dependencies what you're using in your side effect function.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}**);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So in this case we are using the setFormIsValid function.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **setFormIsValid**(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  });  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We are using enteredEmail,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  **enteredEmail**.includes('@') && enteredPassword.trim().length > 6  );  });  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

and we are using enteredPassword.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  enteredEmail.includes('@') && **enteredPassword**.trim().length > 6  );  });  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So we have setFormIsValid, enteredEmail, and enteredPassword. These are the three things we're using here, therefore between these brackets you would add, setFormIsValid, enteredEmail, and enteredPassword.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, [**setFormIsValid**, **enteredEmail**, **enteredPassword**]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

This tells react that after every Login component function execution, it will rerun this useEffect function but only if setFormIsValid, enteredEmail, or enteredPassword changed in the last component re-render cycle. If neither of the three changed, this effect function will not re-run.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}**, [setFormIsValid, enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Actually you can omit setFormIsValid from the dependencies because

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, [~~setFormIsValid,~~ enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

those state updating functions by default are insured by React to never change. So the state updating functions will always be the same across re-render cycles, so you can omit them.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, **setEnteredEmail**] = useState('');  const [emailIsValid, **setEmailIsValid**] = useState();  const [enteredPassword, **setEnteredPassword**] = useState('');  const [passwordIsValid, **setPasswordIsValid**] = useState();  const [formIsValid, **setFormIsValid**] = useState(false);  useEffect(() => {  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

enteredEmail and enteredPassword could have changed. To be precise, with every keystroke, enteredEmail or enteredPassword will have changed. So if we now save our code and refresh the page and try this again, we see that the "Login" button still enables and disables, and the reason for that is



now this effect function re-runs whenever the

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}**, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

dependencies enteredEmail or enteredPassword are changed.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, [**enteredEmail**, **enteredPassword**]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

useEffect helps us make sure that we have one piece of code in one place, instead of as before in multiple places, which re-runs whenever one of the dependencies changes. And that's also something where you use useEffect. So, it's not just for when a component was created for the first time, but it's equally common to run it when certain data, typically some state or some props changed.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **)**;  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Now, inside of this useEffect example, we're not working with local storage. We're not sending an HTTP request. We're not setting a timer or anything like that. Instead, we are updating the React state.

The function is called useEffect. It has one main job. It is there to handle side effects. Often side effects are http requests and so on, but it's also a side effect, if we listen to every keystroke and save that entered data as we do in the emailChangeHandler for example and then we want to trigger another action in response to that. So checking and updating that form validity, in response to a keystroke in the email or password field is also something you could call a side effect. It's a side effect of the user entering data. useEffect is a super important hook that should be executed in response to something, and something could be the component being loaded. It could be the email address being updated. It could be anything. Whenever you have an action, that should be executed in response to some other action, that is a side effect. That is where useEffect can help you.

### 112. What to add & Not to add as Dependencies

In the previous lecture, we explored useEffect and dependencies. You learned that you should add "everything" you use in the effect function as a dependency – i.e. all state variables and functions you use there.

That is correct, but there are a **few exceptions** you should be aware of:

* You **DON'T need to add state updating functions** ( as we did in the last lecture with setFormIsValid): React guarantees that those functions never change, hence you don't need to add them as dependencies (you could though).
* You **DON'T need to add "built-in" APIs or functions** like fetch(), localStorage, etc. (functions and features built-into the browser and hence available globally): These browser APIs / global functions are not related to the React component render cycle and they also never change.
* You also **DON'T need to add variables or functions** you might've **defined OUTSIDE of your components** (e.g. if you create a new helper function in a separate file): Such functions or variables also are not created inside of a component function and hence changing them won't affect your components (components won't be re-evaluated if such variables or functions change and vice-versa)

You must add all "things" you use in your effect function **if those "things" could change because your component (or some parent component) re-rendered**. That's why variables or state defined in component functions, props, or functions defined in component functions have to be added as dependencies!

Here's a made-up dummy example to further clarify the above-mentioned scenarios:

|  |
| --- |
| import ( useEffect, useState ) from 'react';  let myTimer;  const MyComponent = (props) => {  const [timerIsActive, setTimerIsActive) = useState(false);   const { timerDuration } = props; // using destructuring to pull out specific props values   useEffect(() => {  if (!timerIsActive) {  setTimerIsActive(true);  myTimer = setTimeout(() => {  setTimerIsActive(false);  }, timerDuration);  }  }, [timerIsActive, timerDuration]); }; |

In this example:

* timerIsActive is **added as a dependency** because it's component state may change when the component changes (e.g. because the state was updated).
* timerDuration is **added as a dependency** because it's a prop value of that component – so it may change if a parent component changes that value (causing this MyComponent component to re-render as well).
* setTimerIsActive is **NOT added as a dependency** because it's that **exception**: State updating functions could be added but don't have to be added since React guarantees that the functions themselves never change.
* myTimer is **NOT added as a dependency** because it's **not a component-internal variables** (i.e. not some state or a prop value) – it's defined outside of the component and changing it (no matter where) **wouldn't cause the component to be re-evaluated**.
* setTimeout is **NOT added as a dependency** because it's **a built-in API** (built-into the browser) – it's independent from React and your components; it doesn't change.

### 113. Using the useEffect Cleanup Function

Sometimes you have also have an effect that needs to do some cleanup work. Let's say here we are executing this function

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **)**;  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

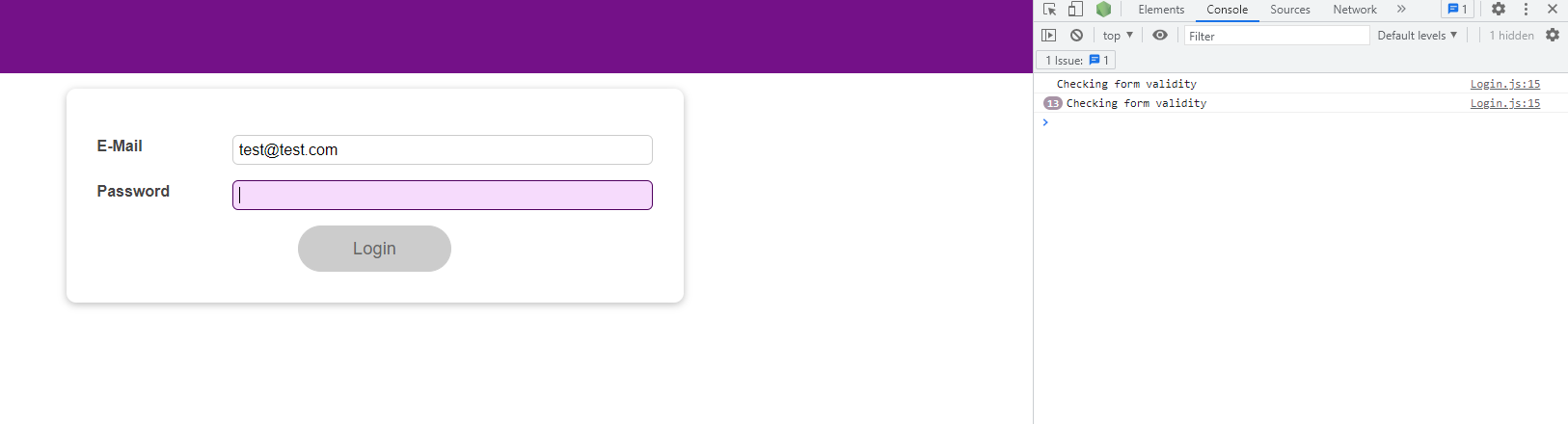
essentially on every keystroke. This is what happens here, and we can prove this by

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, [**enteredEmail**, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

doing console.log('Checking form validity');

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **console.log('Checking form validity');**  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

If we save this and reload the page in the browser, we see that with every keystroke we have "Checking from validity" being logged to the console.



What you do in this function might be a problem. Here we are updating some state. This might already not be ideal. For this simple state update, it won't be a problem, but of course it means that it triggers another function component (in this case the Login functional component) execution and React again needs to check whether it needs to change something in the DOM. That might not be something you want to do for every keystroke.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('Checking form validity');  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Imagine you would do something more complex such as sending an HTTP request to some backend where you check if a username is already in use. You don't want to do that with every keystroke. Because if you do, that means that you're going to be sending a lot of requests. That might be a lot of unnecessary network traffic.

Even this state updating function might not be something that you want to do on every keystroke. Instead, you might want to collect a certain amount of keystrokes or wait for a pause of a certain time duration after a keystroke, and only if the pause is long enough, you go ahead and do your thing.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('Checking form validity');  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

While the user is actively typing, I might not want to check if it's a valid email address. I care about when the user stops typing. If the user stops, for example, stops typing for 500 milliseconds or longer, than we can say okay the user seems to be done, let's see if it's valid. That's a technique called "debouncing". We want to "debounce" the user input. We want to make sure we're not doing something with it on every keystroke but once the user made a pause during typing.

We can use setTimeout, which is a function built-into the browser, to wait for, for example, 500 milliseconds, until we execute some function.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('Checking form validity');  **setTimeout(() => {**  **}, 500);**  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Now, in the setTimeout function we might want to check our form validity or to update our form validity. Now, we would only check if the form is valid every 500 milliseconds.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('Checking form validity');  setTimeout(() => {  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  }, 500);  ~~setFormIsValid(~~  ~~enteredEmail.includes('@') && enteredPassword.trim().length > 6~~  ~~);~~  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

If we move our console.log in here,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setTimeout(() => {  **console.log('Checking form validity');**  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We see that this alone doesn't do much, there is just a slight 500 millisecond delay. The trick is that we actually save the timer and for the next keystroke, we clear the timer, so that we only have one ongoing timer at a time. And only the last timer, will, therefore, complete. And as long as the user keeps on typing, we always clear all other timers. And therefore, we only have one timer that completes, and that completes after 500 milliseconds, which is the delay the user has to issue a new keystroke to clear this timer.

To implement this, in the function that we pass as the first argument, you can return something. Now the something that you return here has to be one specific thing.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  **return**  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

What you return must be a function itself. For example, here, we return an anonymous function, but it also could be a named function like all the other places where we're using anonymous functions. The anonymous arrow function that we are returning is a so called cleanup function.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  return **() => {};**  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

This will run as a cleanup process before useEffect executes this function the next time. Whenever this useEffect function runs, before it runs, except for the very first time when it runs,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **setTimeout(() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  **return () => {};**  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

this cleanup function will run. And, in addition, the cleanup function will run whenever the component you're specifying the effect in unmounts from the dom, so whenever the component is reused. So the cleanup function runs before every

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  return **() => {};**  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

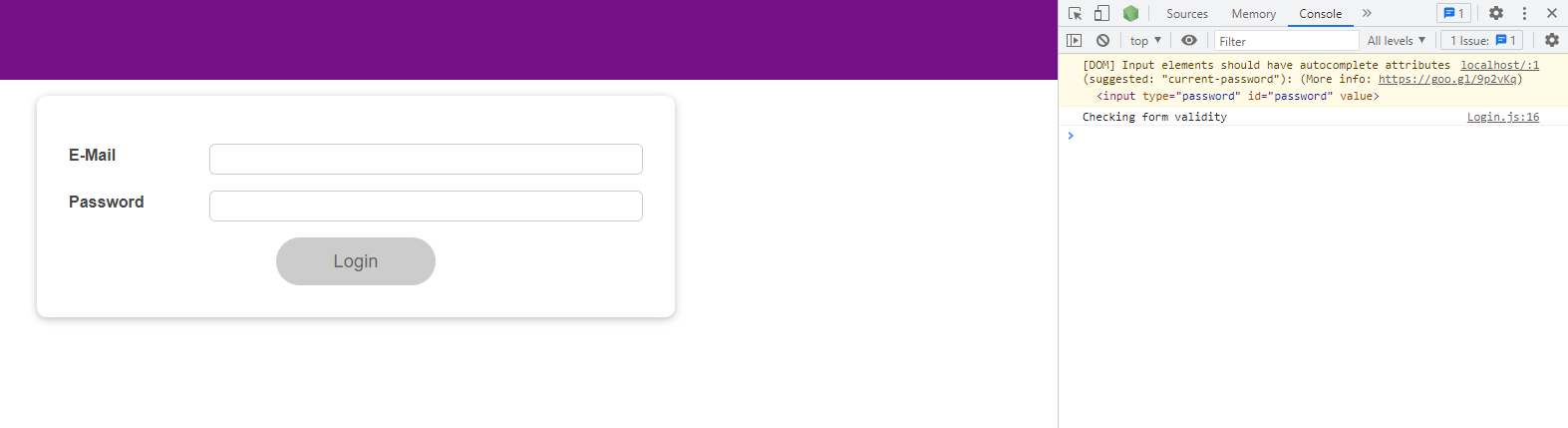
new side effect function execution and before the component is removed. And the cleanup function does not run before the first side effect function execution. But thereafter, it will run before every next side effect function execution.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **setTimeout(() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  **return () => {};**  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

To get an idea of when the cleanup function runs, we can simply console.log('CLEANUP') within the cleanup function.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  return () => {  **console.log('CLEANUP');**  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

If we reload the page we see that "Checking form validity" ran

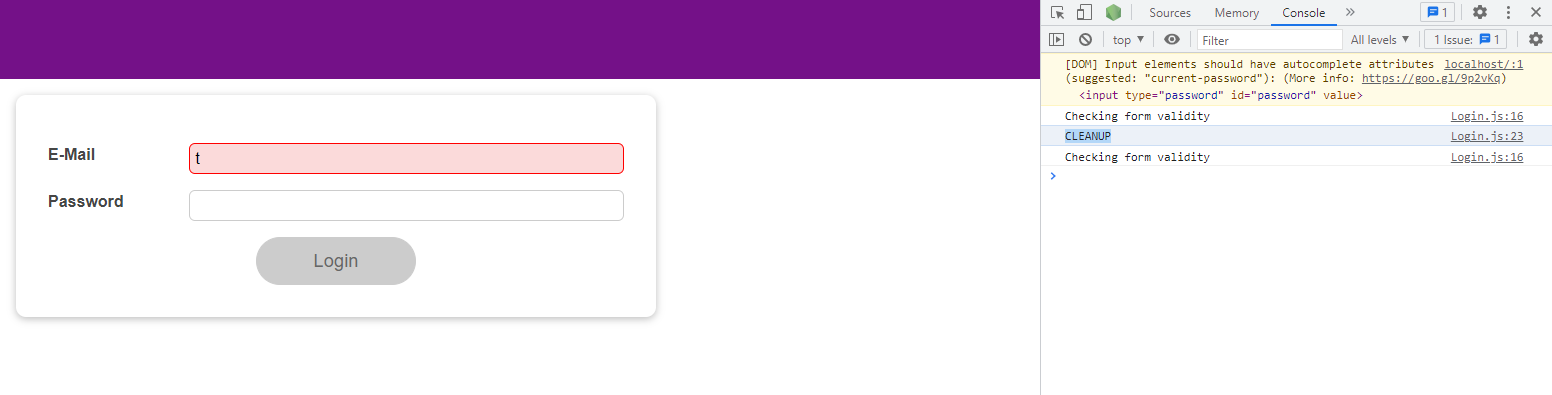


So this code ran, our main side effect function code.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setTimeout(**() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  return () => {  console.log('CLEANUP');  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We don't see the cleanup log because it does not run before the very first side-effect execution.

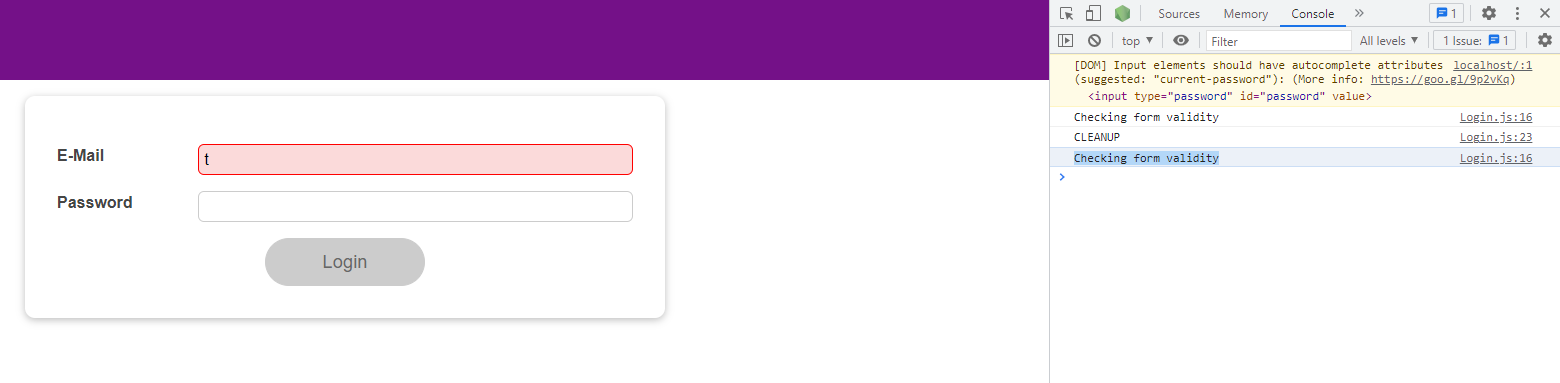
If we go back to the browser, and type just one character in the input, we see cleanup runs immediately.



And then the code inside of this side effect function runs.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  setTimeout(**() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  return () => {  console.log('CLEANUP');  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Wee see "Checking form validity", which means that the side effect function has executed.



With this in mind, we can use the fact that setTimeout returns an identifier for the timer that was set.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **const identifier** = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  return () => {  console.log('CLEANUP');  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

And we can use this identifier to clear this timer with

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = **setTimeout(() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  return () => {  console.log('CLEANUP');  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

the built-in clearTimeout function, which is built-in to the browser. But we don't want to run it here

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  **clearTimeout();**  return () => {  console.log('CLEANUP');  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We want to run it in our cleanup function instead. So inside of our cleanup function, we call clearTimeout.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  ~~clearTimeout();~~  return () => {  console.log('CLEANUP');  **clearTimeout();**  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We pass the identifier of this timeout to the clearTimeout function.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  return () => {  console.log('CLEANUP');  clearTimeout(**identifier**);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

This makes sure that whenever the cleanup function runs,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  return **() => {**  **console.log('CLEANUP');**  **clearTimeout(identifier);**  **}**;  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

I clear the timer that was set

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = **setTimeout(() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

before this cleanup function ran, so in the

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 5000);  return **() => {**  **console.log('CLEANUP');**  **clearTimeout(identifier);**  **}**;  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

last side effect function execution, so that when the next

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **const identifier = setTimeout(() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  **return () => {**  **console.log('CLEANUP');**  **clearTimeout(identifier);**  **};**  **}**, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

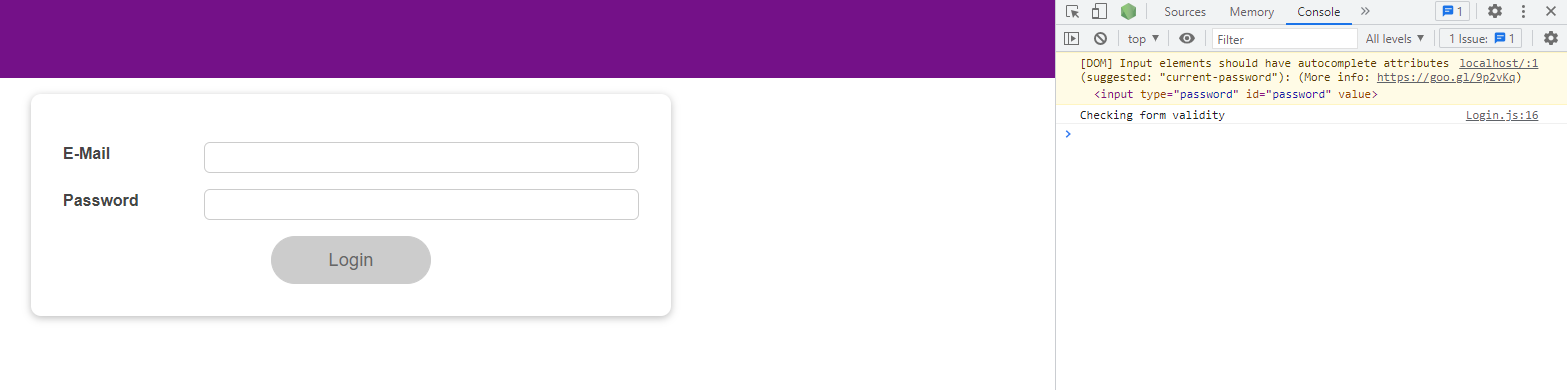
side-effect execution is due,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **const identifier = setTimeout(() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  **return** () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

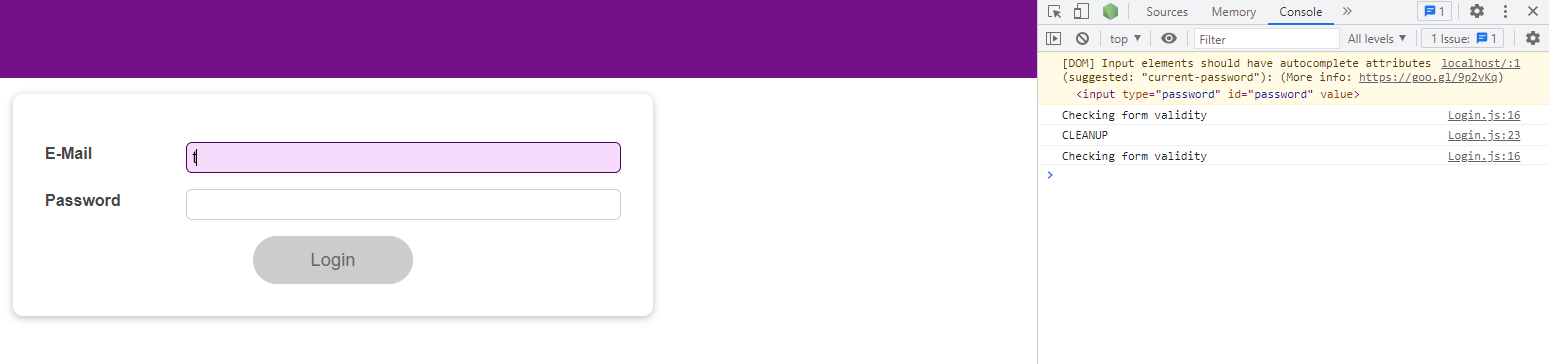
We are able to set a new timer. So we clear the last timer before we set a new one.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = **setTimeout(() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  **}, 5000);**  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

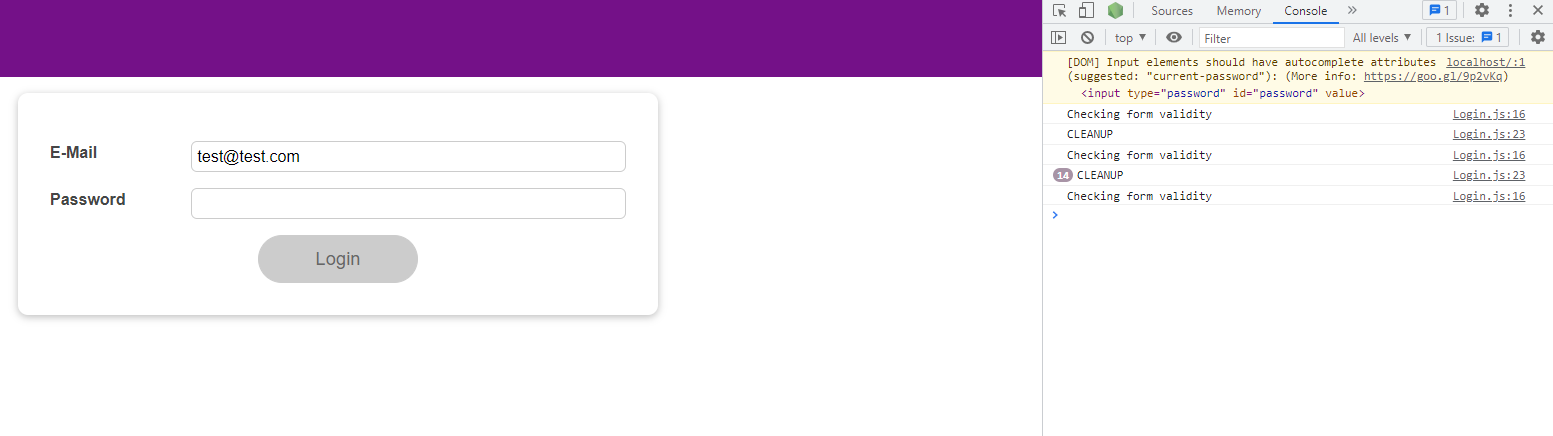
Therefore, now if we reload the page, we see "Checking form validity".



If we type in the email input field, we see "CLEANUP" and then again "Checking form validity".



Now if we type in the email input field very quickly, we see a lot of "CLEANUP"s, but you only see one "Checking form validity".



And that means that this code in here only ran once for all those keystrokes. If we would have been sending an HTTP request here, we now would have only sent once instead of a dozen HTTP requests.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  const identifier = setTimeout(**() => {**  **console.log('Checking form validity');**  **setFormIsValid(**  **enteredEmail.includes('@') && enteredPassword.trim().length > 6**  **);**  }, 5000);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

### 114. useEffect Summary

useEffect besides useState is the most important React hook you have.

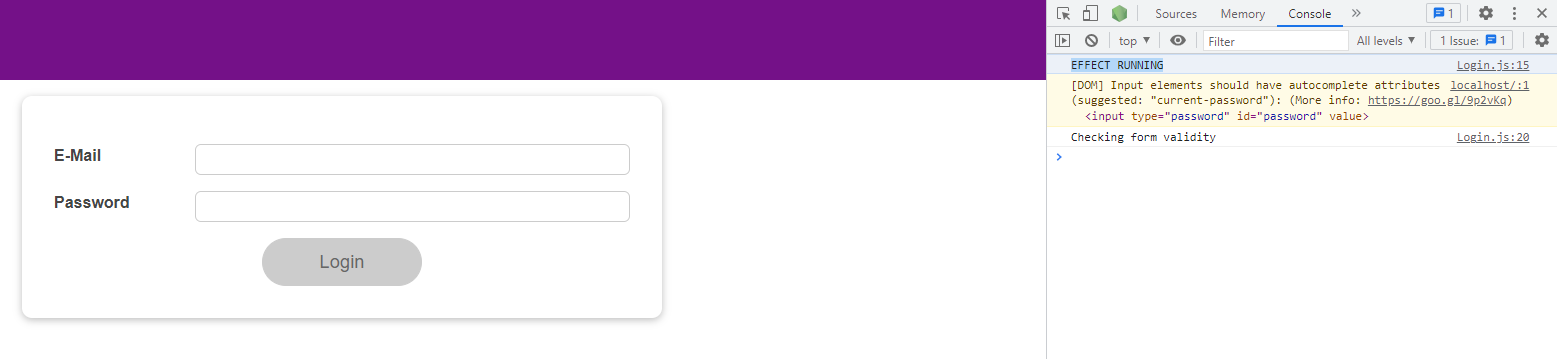
We want to make it clear at which point of time, which part of useEffect kicks in and executes. In the Login component we will add a new useEffect call where I only add this first argument.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  **useEffect(() => {});**    useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

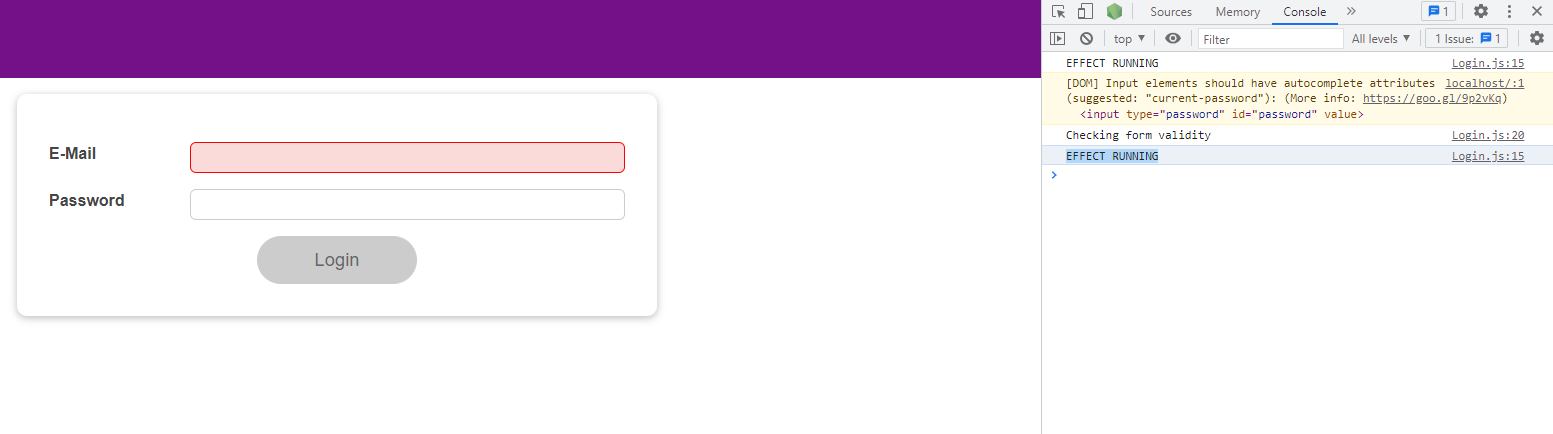
So for the moment, no second argument. In the anonymous function within useEffect we can log "EFFECT RUNNING":

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  **console.log('EFFECT RUNNING');**  });  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

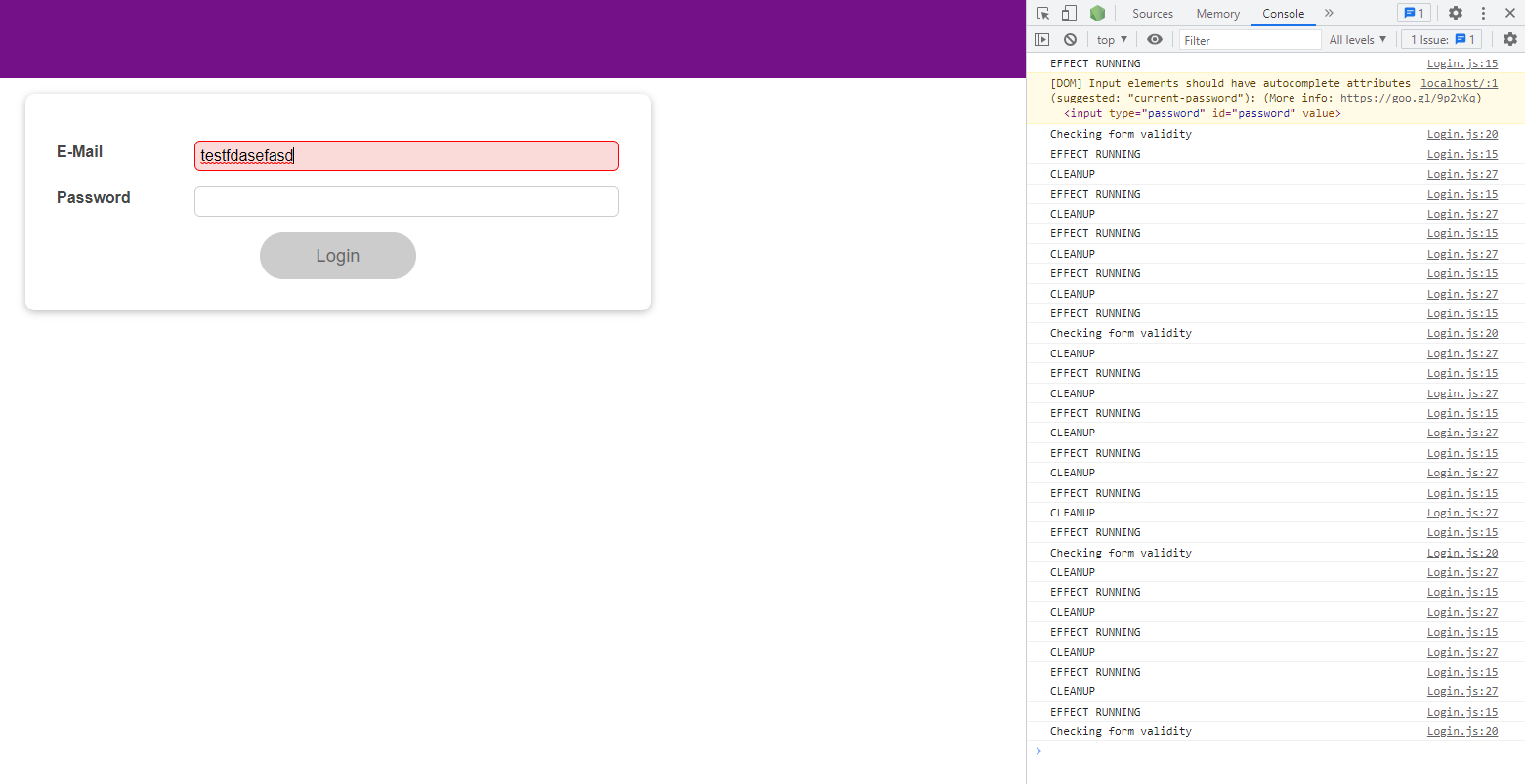
Now at the moment we have no information about the dependencies. We just have this first argument and this is a valid way of using it. Though, you will rarely use useEffect like this. The reason for this is that if we save and reload the page we see that useEffect runs when the component first mounts, so when the Login component is rendered for the first time, but



then also for each state update, for example, if I click in the e-mail field and click out, we see, effect running.



For every keystroke, we see effect running.



So this now really runs for every time the Login component re-runs because we learned that this effect function runs after every component render cycle. Not before each component render cycle and not during each component render cycle, but after each component render cycle, including the first time the Login component was mounted.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **console.log('EFFECT RUNNING');**  **}**);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

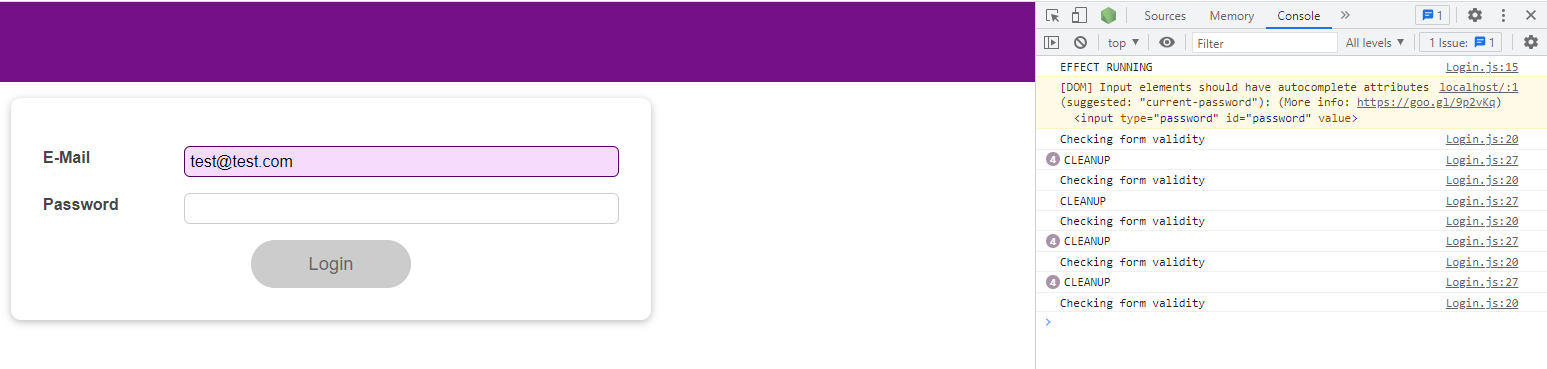
This changes once we add an empty array.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  }, **[]**);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

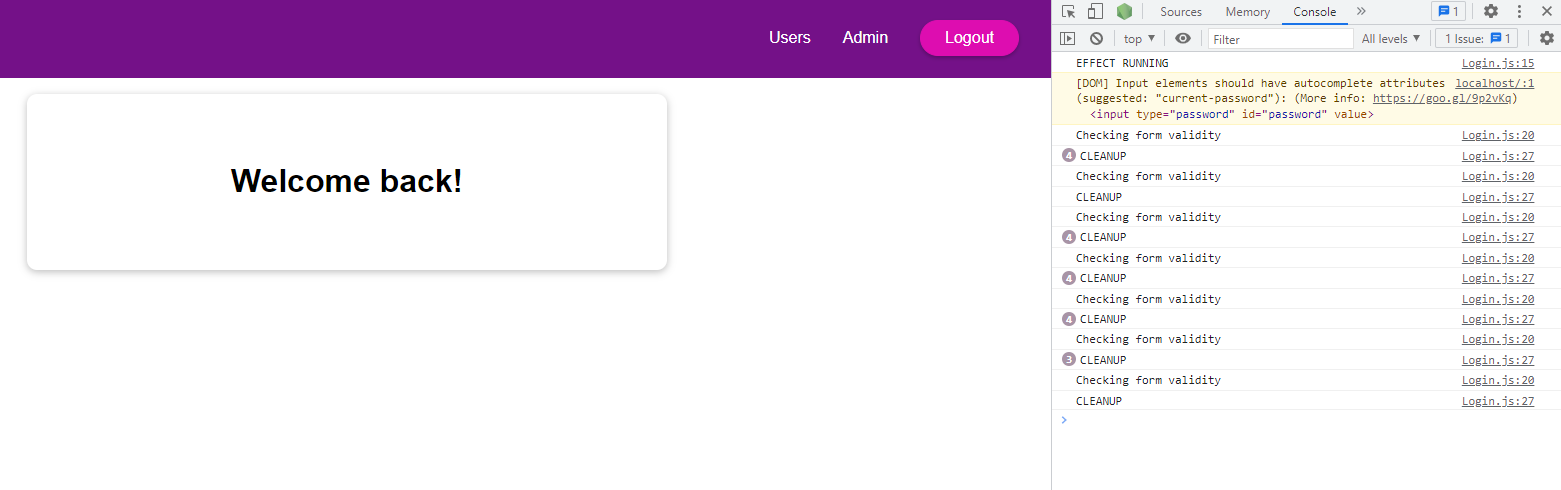
Now this function here only executes for the first time the Login component was mounted and rendered but not thereafter, not for any subsequent re-render cycle.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(**() => {**  **console.log('EFFECT RUNNING');**  **}**, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So when we save and reload the page, we see "EFFECT RUNNING" but on keystrokes, we don't see it.



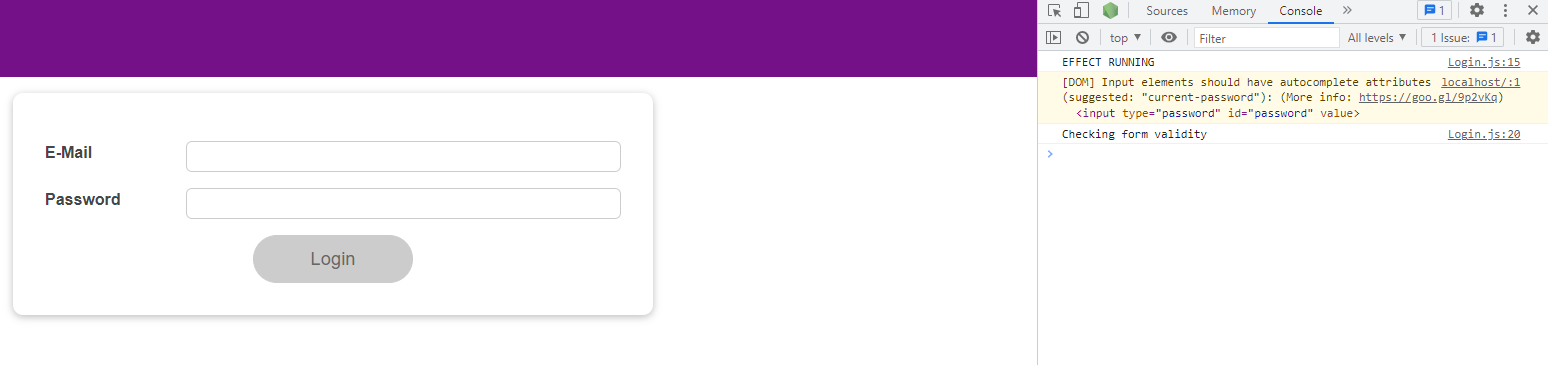
And if we login, we also don't see "EFFECT RUNNING", so it only really ran once.



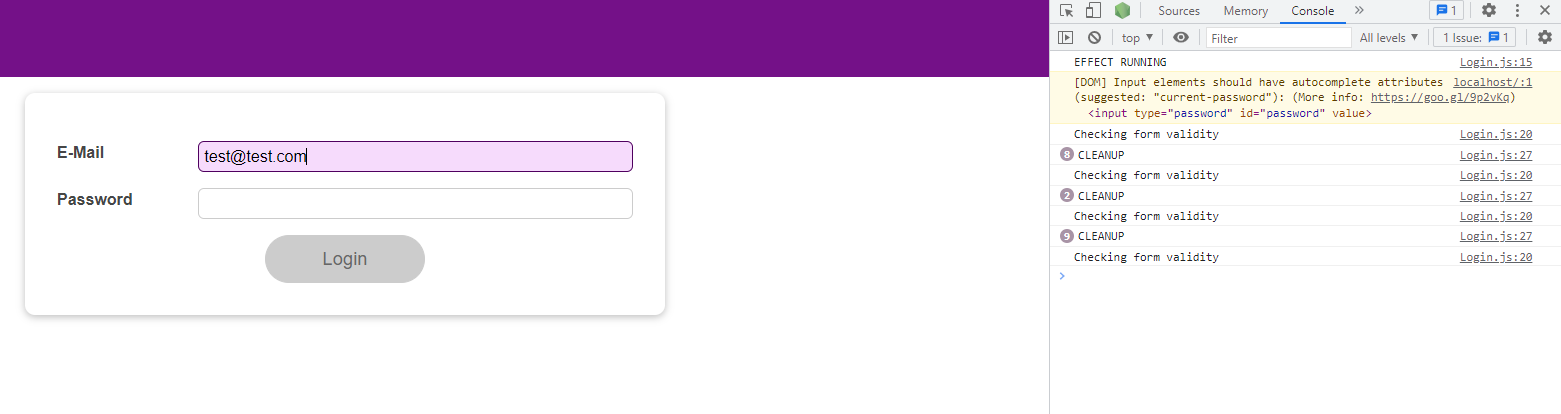
Alternatively, if we add a dependency like enteredEmail or enteredPassword, now the function reruns whenever the component was re-evaluated and this state, in this case here, changed.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  }, [**enteredPassword**]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

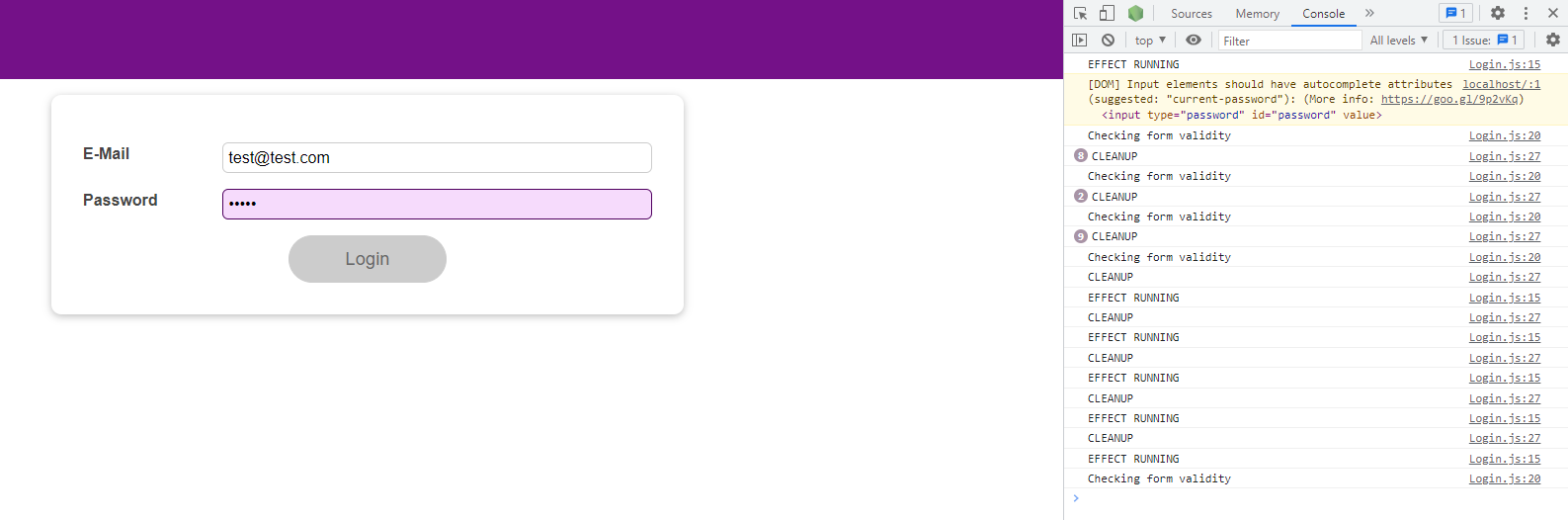
So if we now log out and reload, we see effectRunning for the first time this was mounted.



For keystrokes in the email, nothing changes.



But then for keystrokes in the password, we see "EFFECT RUNNING", though because the password is a dependency.



We also have the cleanup function that we return, which runs before

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return **() => {};**  }, [enteredPassword]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

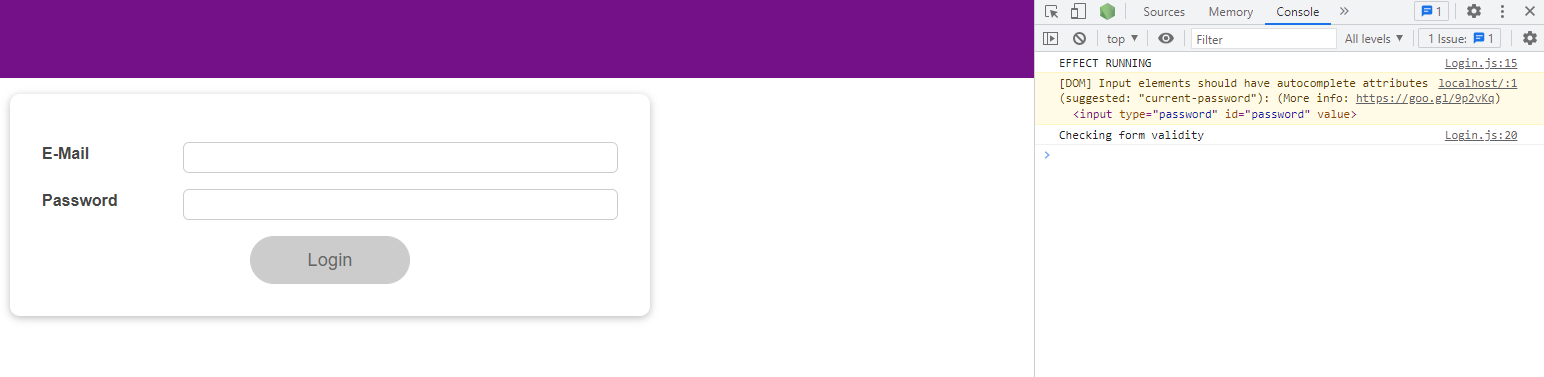
this state updating function as a whole, runs, but not before the first time it runs.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect((**) => {**  **console.log('EFFECT RUNNING');**  **return () => {};**  **}**, [enteredPassword]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

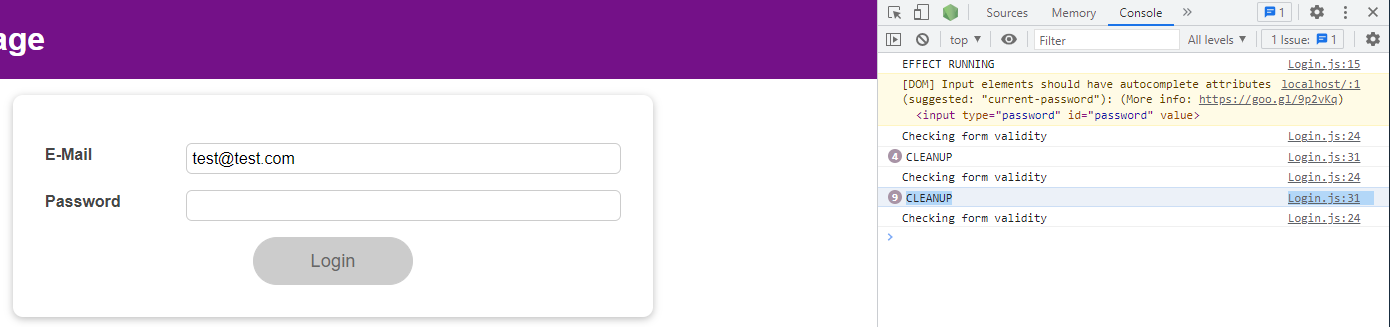
So, we will now log "EFFECT CLEANUP"

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  **console.log('EFFECT CLEANUP');**  };  }, [enteredPassword]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We save this and reload the page, we see only "EFFECT RUNNING" because



for the first render cycle, for the email, nothing happens. Our "CLEANUP" log is from our other cleanup log from our other useEffect. So "EFFECT CLEANUP" did not print yet.



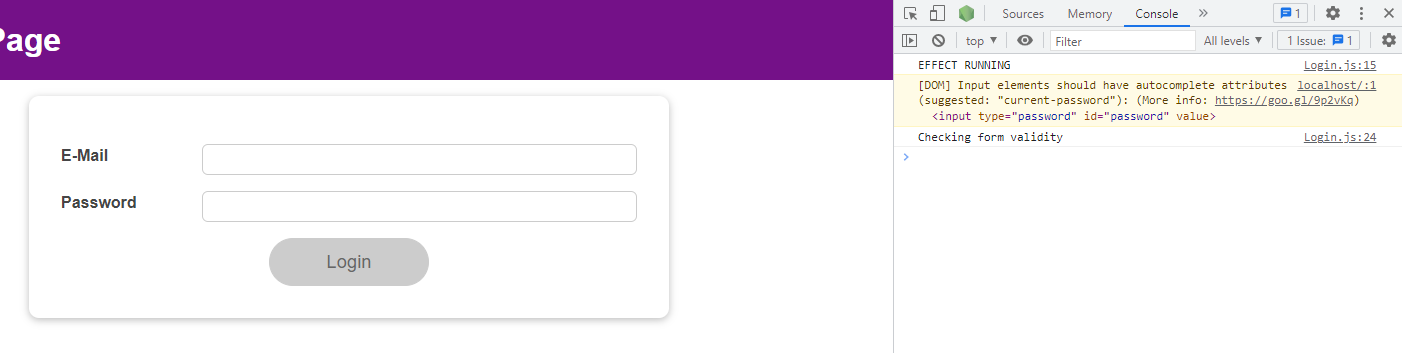
Once we start typing in the password, we see "EFFECT CLEANUP" being triggered, and it triggers before the effect function runs.



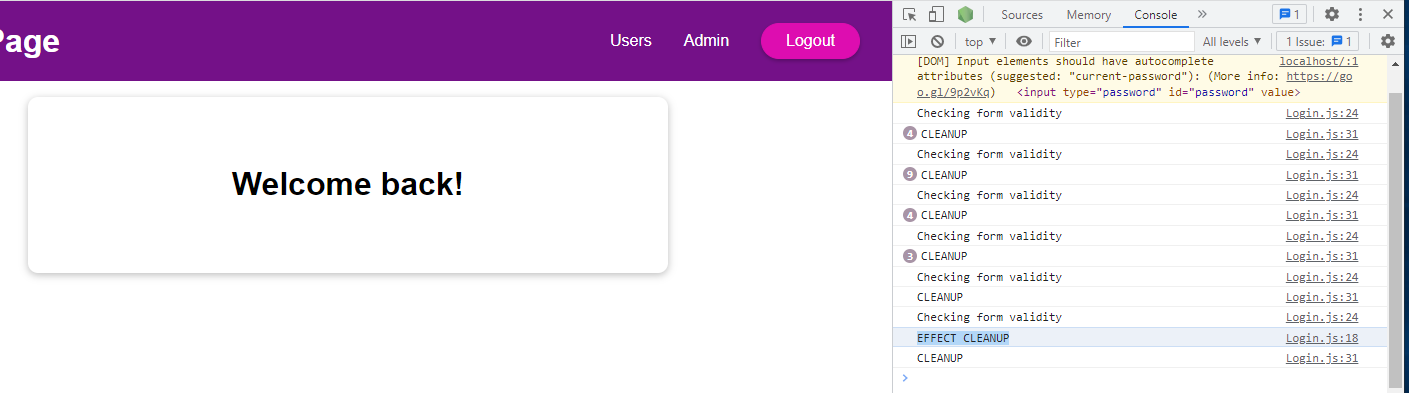
If we had an empty array here, so no dependencies,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, **[]**);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

And reload the browser, we only see that effect running once

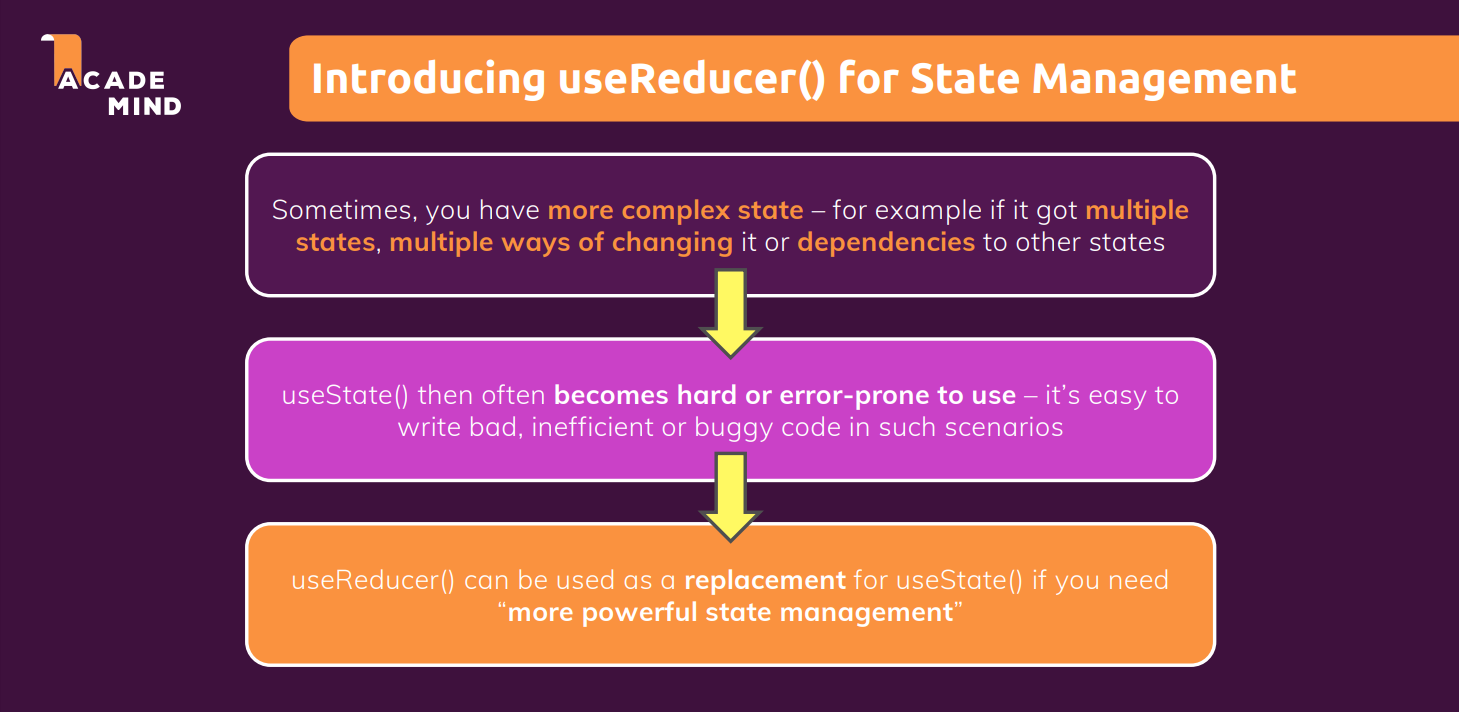


and the cleanup function in this case would run when the component is removed. So in this case, for example, when I login and the component is removed from the DOM, we see "EFFECT CLEANUP".



### 115. Introducing useReducer & Reducers in General

useReducer is another built-in hook, and it will help us with state management. It's a bit like useState but with more capabilities and is especially useful for more complex state. For example, multiple states that belong together, that are managing the same thing, just different aspects of it. Or you have multiple states that change together or are related. In many cases useState and the state from there often becomes hard or error-prone to use and manage, and it's easy then to write bad, inefficient, or potentially buggy code. useReducer is an alternative to useState, so it is a replacement if you need a more powerful state management. This does not mean that you should always use useReducer because it's more powerful. useState is more complex to use, so it requires more setup. For the majority of scenarios, useState is what you should use. There are cases where the extra work of getting useReducer to work is just worth it.



We can go back to our "src/components/Login/Login.js" file and you see I'm managing a couple of states snapshots.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  **const [enteredEmail, setEnteredEmail] = useState('');**  **const [emailIsValid, setEmailIsValid] = useState();**  **const [enteredPassword, setEnteredPassword] = useState('');**  **const [passwordIsValid, setPasswordIsValid] = useState();**  **const [formIsValid, setFormIsValid] = useState(false);**  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

You might be able to spot some related state in the Login component. We managed the enteredEmail and the enteredPassword,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const[**enteredEmail**, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [**enteredPassword**, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

but then we also manage the fact or the question, the response to the question of whether the email or password is valid.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [**emailIsValid**, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [**passwordIsValid**, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

And we managed the overall form validity.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [**formIsValid**, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

You could argue that overall, this is all one big state that describes the overall form state.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  **const [enteredEmail, setEnteredEmail] = useState('');**  **const [emailIsValid, setEmailIsValid] = useState();**  **const [enteredPassword, setEnteredPassword] = useState('');**  **const [passwordIsValid, setPasswordIsValid] = useState();**  **const [formIsValid, setFormIsValid] = useState(false);**  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We are deriving our new emailIsValid state by

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  **setEmailIsValid**(enteredEmail.includes('@'))**;**  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

looking at another state which is something that we should not do. It works in most cases, but in some scenarios it may not work because maybe some state update for enteredEmail wasn't processed in time, and then we would try to updated emailIsValid based on some outdated enteredEmail state.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  setEmailIsValid(**enteredEmail**.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Here we are violating this rule, which we should not be violating and that is a scenario where useReducer is always a good choice. If you update a state, which depends on another state, then merging this into one state could be a good idea. And you could do that without useReducer as well by simply managing an email state which is an object with the value and the validity being part of the same object. You could do that with useState but in such cases when your state becomes more complex, bigger and combines multiple related states, useReducer can also be worth a closer look.

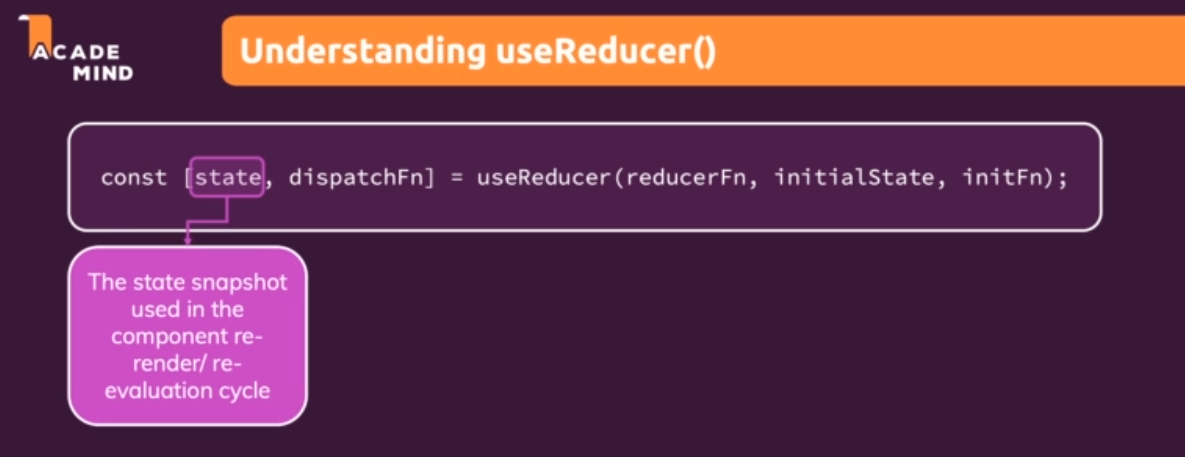
|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  },[]);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity');  setFormIsValid(  enteredEmail.includes('@') && enteredPassword.trim().length > 6  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  };  const validateEmailHandler = () => {  **setEmailIsValid(enteredEmail.includes('@'));**  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

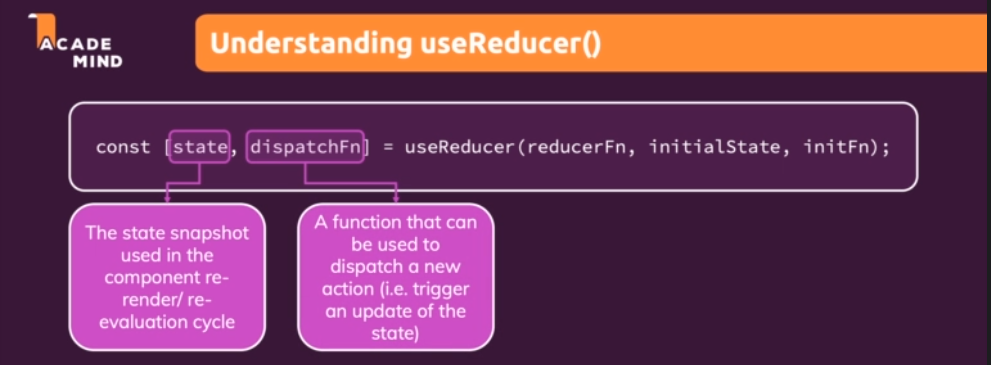
### 116. Using the useReducer() Hook

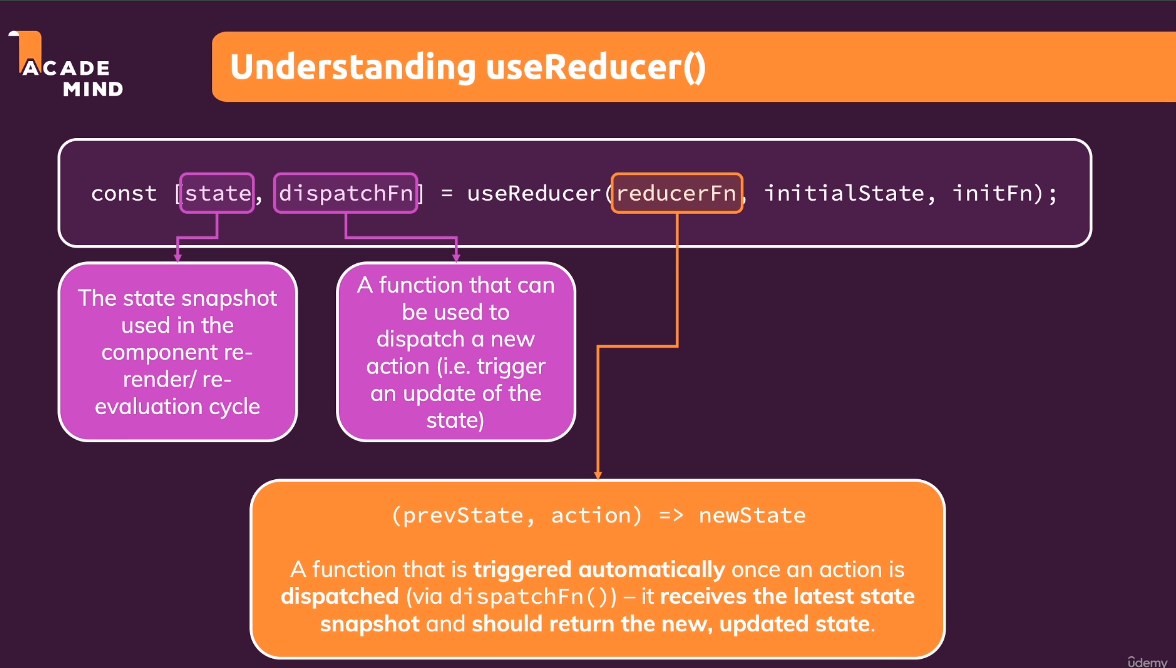
useReducer just like useState, always returns an array with exactly two values. And therefore, you can use array destructuring to pull out these values and store them in separate constants.



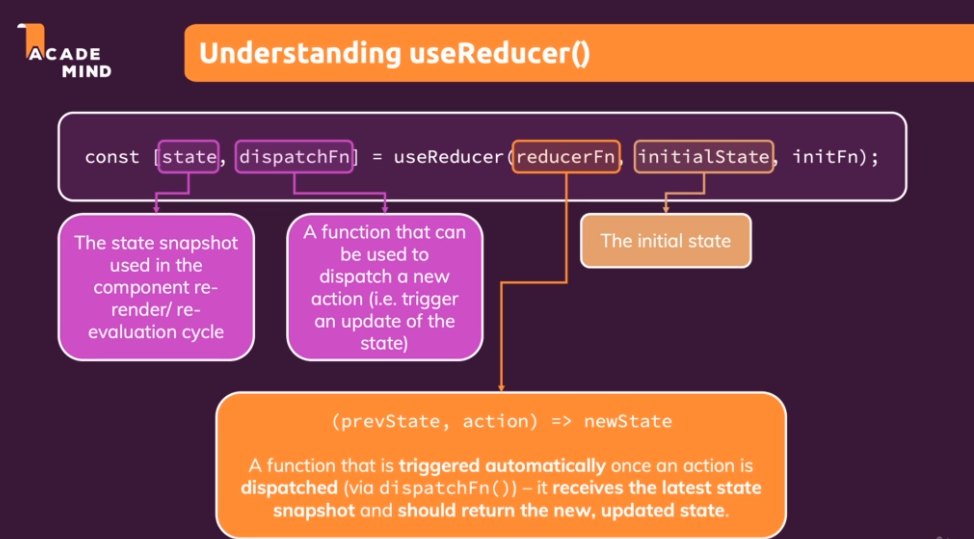
The two values that you are getting are the latest state snapshot because it is a state management mechanism like useState, so of course you get your state snapshot.



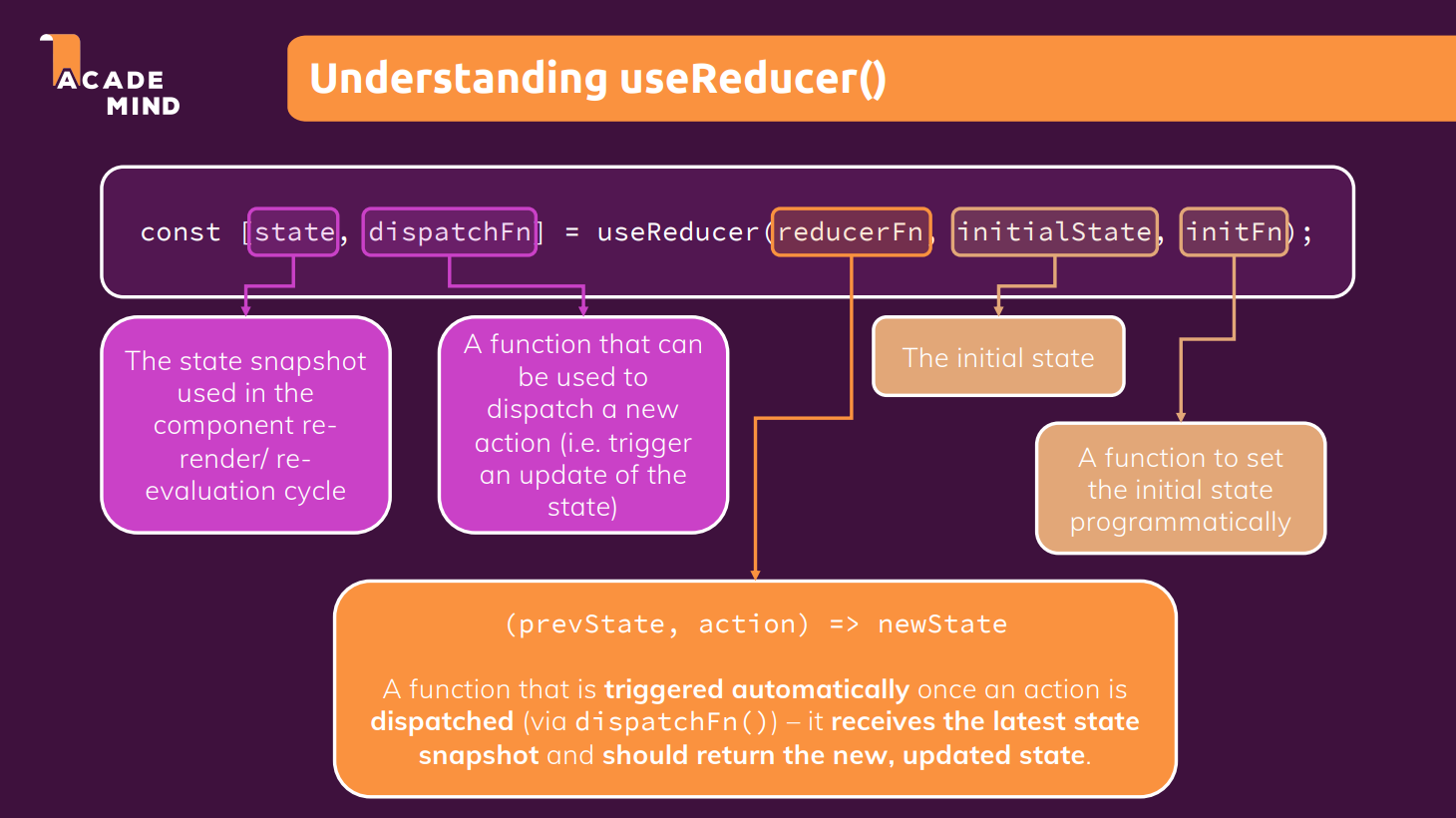
But you also get a function that allows you to update that state snapshot. That's kind of the same as useState but the state updating function will work differently. Instead of just setting a new state value, you will dispatch an action. And that action will be consumed by

the first argument you pass to useReducer, a so-called reducer function or reducerFn. The reducer function is a function that gets the latest state snapshot automatically because this function will be called by React, and it gets the action that was dispatched. Because react will call this reducer function whenever a new action is dispatched. So then it gets the last state snapshot managed by react. And it gets the action that was dispatched that triggered this reducer function execution. The reducer function should return a new, updated state. It's a bit like the function form of the useState hook, but an extended version of that you could say, because of the action dispatched.

In addition, you can also set some initialState and



an initial function (initFn) that should run to set the initial state in case your initial state is more complex.



We could use useReducer here to combine our entered values and validities for the email and the password.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  **const [enteredEmail, setEnteredEmail] = useState('');**  **const [emailIsValid, setEmailIsValid] = useState();**  **const [enteredPassword, setEnteredPassword] = useState('');**  **const [passwordIsValid, setPasswordIsValid] = useState();**  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We could also use it to manage the overall form state with it. So we could either manage one big form state that includes everything or in multiple smaller states. Both would work, both would be fine, but to keep things simple here

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  **setFormIsValid(**  **event.target.value.includes('@') && enteredPassword.trim().length > 6**  **);**  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

we will start with managing our email state with useReducer. And there the goal is to combine the value and the validity into one state managed by useReducer.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  **const [enteredEmail, setEnteredEmail] = useState('');**  **const [emailIsValid, setEmailIsValid] = useState();**  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

First of all we need to import useReducer

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, **useReducer** } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We then call useReducer

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  **useReducer();**  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

useReducer as you learned returns an array with two elements, and we can use array destructuring to pull those elements out of it.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  **const [emailState, dispatchEmail]** =useReducer();  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Then useReducer as a first argument takes a function. Here again, I'm using an anonymous function.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail]=useReducer(**() => {}**);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Though, we actually could also outsource this into a separate named function, maybe to make this easier to read. I'll name this emailReducer, and I'll store my arrow function in this constant, so that I can

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  **const emailReducer = () => {};**  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer( () => {});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

point at emailReducer here:

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = () => {};  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(**emailReducer**);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

The emailReducer function was created outside of the component function because inside of the reducer function, we won't need any data that's generated inside of the component function. So this reducer function can be created outside of the scope of this component function because it does not need to interact with anything defined inside of the component function. All the data which will be required and used inside of the reducer function will be passed into this function when it's executed by React, automatically. So that's why we can define it outside of the component function here.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  **const emailReducer = () => {};**  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Now this emailReducer function receives two arguments, two parameters, our last state snapshot and the action that was dispatched.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (**state**, **action**) => {};  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Inside of the emailReducer function, we should return a new state.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  **return**  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Here the state could be an object where we have the value, which is initially an empty string and the isValid field, which initially is false for example. That also could be our initial state, which

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return **{ value: '', isValid: false}**;  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

is the second argument we pass to useReducer where value is set to an empty string and isValid is set to false.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return { value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, **{value: '', isValid: false}**);  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

That's the initial state we set here for our emailState snapshot.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return { value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [**emailState**, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  enteredEmail.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

The emailState is therefore for example what we can use in our code. For example, here where I want the enteredEmail.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return { value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  **enteredEmail**.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We replace enteredEmail with emailState.value because that's where we're going to store the entered value.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  **emailState.value**.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(enteredEmail.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Also, here we can change enteredEmail to emailState.value

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.value.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(**emailState.value**.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Keep in mind that we have the isValid field in there. We're going to add code that changes its value soon.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', **isValid**: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.value.includes('@') && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.value.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So instead of revalidating here,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  **emailState.value.includes('@')** && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.value.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We could also just check if emailState.isValid.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  **emailState.isValid** && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.value.includes('@'));  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Therefore, we can simplify this here:

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(**emailState.value.includes('@')**);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

to emailState.isValid

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(**emailState.isValid**);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(enteredEmail, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

If we scroll down further in the submitHandler, here we want to forward the value.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(**enteredEmail**, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So that would be enteredEmail.value.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(**emailState.value**, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

And in the JSX code instead of emailIsValid,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  **emailIsValid** === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

we have emailState.isValid

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value**,** enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  **emailState.isValid** === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={enteredEmail}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

And here where I pass the value back into the input,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value**,** enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={**enteredEmail**}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

we have emailState.value.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  const [enteredEmail, setEnteredEmail] = useState('');  const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value**,** enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={**emailState.value**}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

That should be the different places where I'm using my email states. With that, we're no longer using enteredEmail and emailIsValid. Because we are not using these two states, we can remove them. Instead of completely removing them, we will just comment them out.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  **// const [enteredEmail, setEnteredEmail] = useState('');**  **// const [emailIsValid, setEmailIsValid] = useState();**  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We are not done with emailReducer. It always returns this state, which is not what we want.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return **{value: '', isValid: false}**;  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Instead, we need to dispatch an action eventually.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, **dispatchEmail**] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We need to dispatch it here, for example, when we want to update the value

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  **setEnteredEmail(event.target.value);**  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Or here, when we want to update the validity.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  setEnteredEmail(event.target.value);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  **setEmailIsValid(emailState.isValid);**  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Let's start with the value. We update this by calling dispatchEmail, and we pass to it a so-called action. It's up to you what this action is. It can be a string identifier, something like 'NEW\_EMAIL\_VALUE'), it could be a number, but often it will be an object, which has some field that holds some identifier, often the field is then named type and that could be set to something like 'USER\_INPUT', for example. It doesn't have to be all caps. It is just a convention that you see a lot. For the second key, we can add a payload to this action. Here, since we want to save what the user entered, it would make sense to add some payload. For example, we could have a val field, which holds event.target.value, which is the value that the user entered.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  **dispatchEmail({type: 'USER\_INPUT', val: event.target.value});**  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

That will now trigger this function because that's the reducer function we

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  **const emailReducer = (state, action) => {**  **return {value: '', isValid: false};**  **};**  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

passed to useReducer.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(**emailReducer**, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Now inside of emailReducer we can handle this action, for example, with an if statement. We can check if action.type is equal to 'USER\_INPUT'

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  **if (action.type === 'USER\_INPUT')**  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Keep in mind what we dispatch as an action

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, **action**) => {  if (action.type === 'USER\_INPUT')  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

will be an object because that's what we set it to here. And this object happens to have a type field here, so

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT')  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail(**{type: 'USER\_INPUT', val: event.target.value}**);  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

I can check for action.type here and check if the values stored in that type field is that string with the content 'USER\_INPUT'.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (**action.type** === 'USER\_INPUT')  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

If that's the case, we could say I don't want to return this

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {    }  **return {value: '', isValid: false};**  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

empty state snapshot. Instead, I want to return the state snapshot for my email where the value is action.val. That's the payload we appended to our action. Maybe we also want to update the validity here by checking action.val.includes('@'). So, here I'm then updating both the value and isValid whenever I received

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  **return { value: action.val, isValid: action.val.includes('@') };**  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

as action.type 'USER\_INPUT'

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === **'USER\_INPUT'**) {  return { value: action.val, isValid: action.val.includes('@') };  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

For any other action that might reach this reducer, this default state will be returned.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  **return {value: '', isValid: false};**  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  setEmailIsValid(emailState.isValid);  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Now we also need to dispatch an action here on the validateEmailHandler. Instead of setting the email as valid here,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  **setEmailIsValid(emailState.isValid);**  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

I can dispatchEmail, again passing in an object. You should be consistent. Your actions should always have the same structure, so that your code, where you, for example try to access a type property, does not suddenly fail. So once you've decided for an action structure, you ant to stick to it. Therefore, here I again dispatch an object with a type field, now here it's for example, 'INPUT\_BLUR', because the input lost focus; it was blurred. And we don't need to add a value here necessarily because all we care about here is that the input lost focus. There is no extra data that needs to be added. So, here we have a simpler action. It is still an object, still with a type property but without a value.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  **dispatchEmail({type: 'INPUT\_BLUR'});**  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

The fact that the val is missing does not matter because of this line here

|  |
| --- |
| src/components/Login/Login.js |
| src/components/Login/Login.js  import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  **return { value: action.val, isValid: action.val.includes('@') };**  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

where we try to access the value,

|  |
| --- |
| src/components/Login/Login.js |
| src/components/Login/Login.js  import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.**val**, isValid: action.val.includes('@') };  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

will not run for action of type 'INPUT\_BLUR'.

|  |
| --- |
| src/components/Login/Login.js |
| src/components/Login/Login.js  import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: '**INPUT\_BLUR**'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

because we only run this if the action.type is 'USER\_INPUT'.

|  |
| --- |
| src/components/Login/Login.js |
| src/components/Login/Login.js  import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === '**USER\_INPUT**') {  return { value: action.val, isValid: action.val.includes('@') };  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Instead, I will check if the action.type is 'INPUT\_BLUR'.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  **if (action.type === 'INPUT\_BLUR') {**    **}**  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

If action.type is 'INPUT\_BLUR', I will return a new snapshot, a new state value for my emailState. But now the value of course should be the value we had before. I don't want to set the value to empty because the input could blur after the user entered something. I don't want to lose that, so therefore, I'll use my last state snapshot, which I get here, and this is guaranteed to be the absolute last state snapshot. React gives us this state snapshot and it makes sure that it's the latest one.

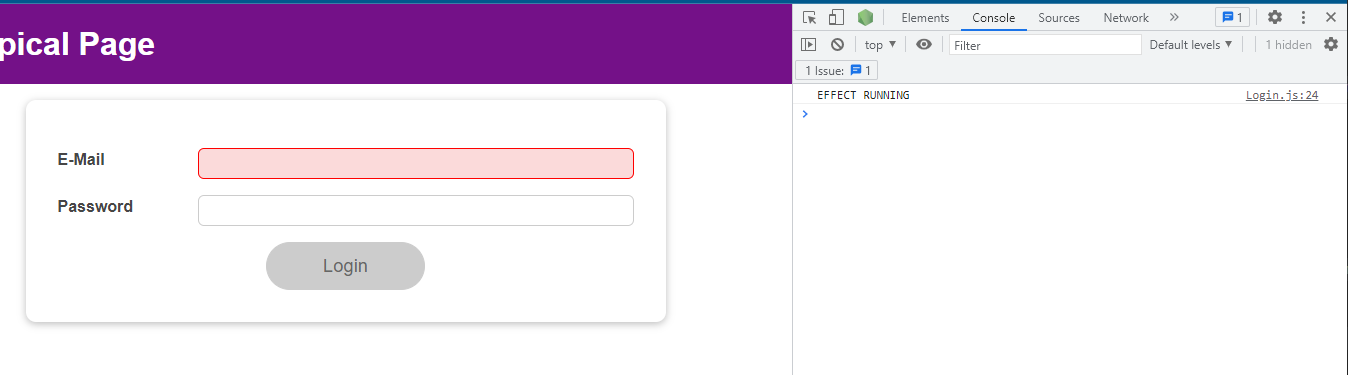
|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (**state**, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {    }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So for the value, we can simply access state.value, for example, to access the last value that was entered for the email. And for isValid we can therefore check if state.value is valid by repeating the validity check .includes('@').

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  **return { value: state.value, isValid: state.value.includes('@')};**  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: false});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We now have this extra action.

When we reload the browser, we see that initially it's blurred.



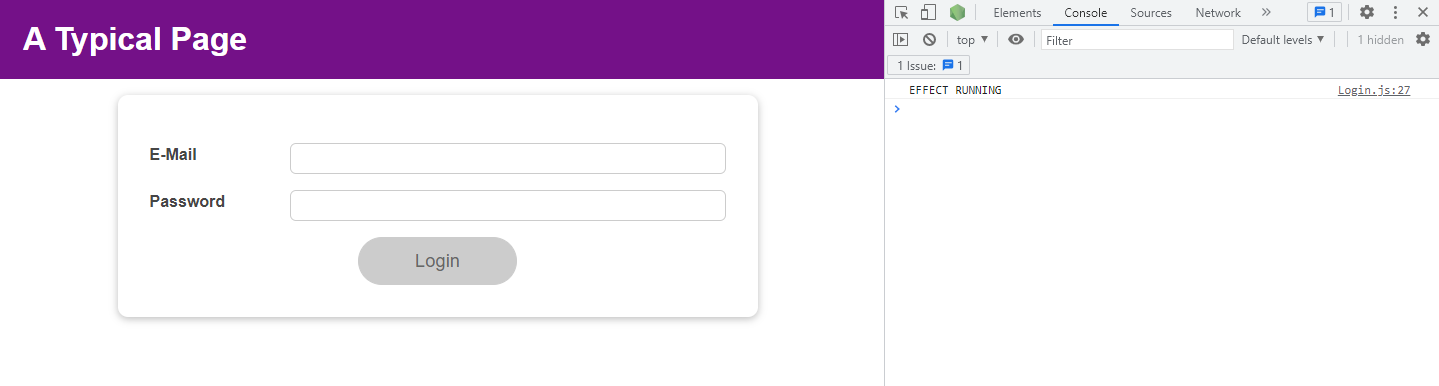
This makes sense because my initial value for isValid is false.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: **false**});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

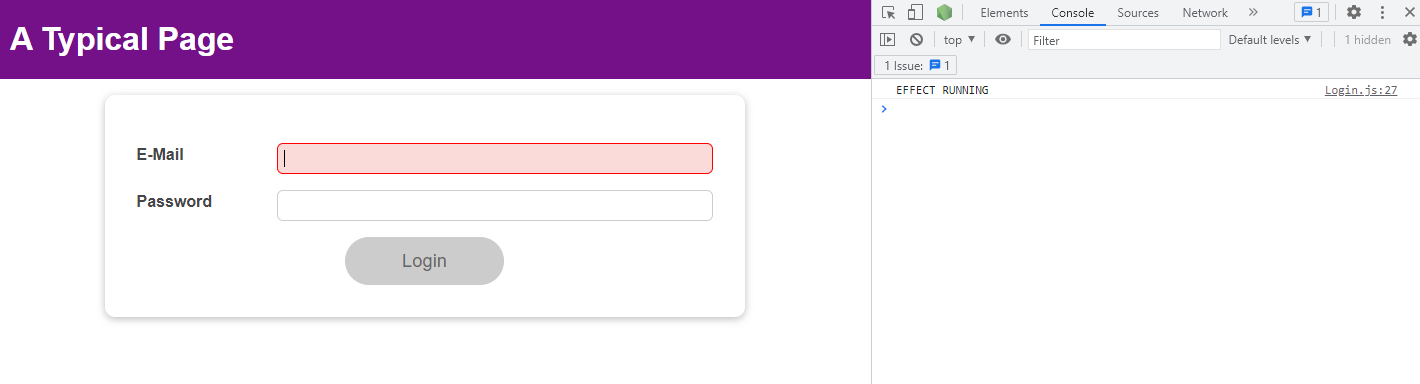
If I set the initial value of isValid to undefined or null,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: **null**});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

you see it's not treated as invalid. It's not treated as if it was blured from the beginning on.



but if I click in there, it is blurred. The general form validation works, now with the help of useReducer.



useReducer allows us to group our email state together and manage it in one place.

|  |
| --- |
| src/components/Login/Login.js |
| src/components/Login/Login.js  import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  **const emailReducer = (state, action) => {**  **if (action.type === 'USER\_INPUT') {**  **return { value: action.val, isValid: action.val.includes('@') };**  **}**  **if (action.type === 'INPUT\_BLUR') {**  **return { value: state.value, isValid: state.value.includes('@')};**  **}**  **return {value: '', isValid: false};**  **};**  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  const [enteredPassword, setEnteredPassword] = useState('');  const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && enteredPassword.trim().length > 6  );  };  const passwordChangeHandler = (event) => {  setEnteredPassword(event.target.value);  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  setPasswordIsValid(enteredPassword.trim().length > 6);  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, enteredPassword);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordIsValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={enteredPassword}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

### 117. useReducer & useEffect

The form validity is of course a bit related to the input validity because the inputs are part of the overall form.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  **const [formIsValid, setFormIsValid] = useState(false);**  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && passwordState.isValid  );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So the code that we have here is still not optimal because we still derive form validity

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  **setFormIsValid(**  **event.target.value.includes('@') && passwordState.isValid**  **);**  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

of other state, which is still not what we want due to React's scheduling the state updates, so we might still refer to an old state here that does not change just because the state is coming from useReducer.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  event.target.value.includes('@') && **passwordState**.isValid  );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

setFromIsValid like this still is not optimal. And in addition, having our state that is split like this even though it technically belongs together is also not something you might want. Now, whether you want this or not is up to you, but the fact that this is not optimal is a fact.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  **setFormIsValid(**  **event.target.value.includes('@') && passwordState.isValid**  **);**  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  setFormIsValid(  emailState.isValid && event.target.value.trim().length > 6  );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We will comment out setFormIsValid.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  // useEffect(() => {  // const identifier = setTimeout(() => {  // console.log('Checking form validity!');  // setFormIsValid(  // enteredEmail.includes('@') && enteredPassword.trim().length > 6  // );  // }, 500);  // return () => {  // console.log('CLEANUP');  // clearTimeout(identifier);  // };  // }, [enteredEmail, enteredPassword]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

and lets go back to the useEffect solution, which actually is a nice solution and update it accordingly.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  **emailState.isValid** && **passwordState.isValid**  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [**emailState**, **passwordState**]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

This would now be a fine way of calling setFormIsValid since it's now inside of an effect

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  **setFormIsValid(**  **emailState.isValid && passwordState.isValid**  **);**  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

we will refer to our state snapshots, but this effect is guaranteed to

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  **emailState.isValid && passwordState.isValid**  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

rerun whenever

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(**() => {**  **const identifier = setTimeout(() => {**  **console.log('Checking form validity!');**  **setFormIsValid(**  **emailState.isValid && passwordState.isValid**  **);**  **}, 500);**  **return () => {**  **console.log('CLEANUP');**  **clearTimeout(identifier);**  **};**  **}**, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

these states change, and therefore, ultimately, it will run with the latest state values.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [**emailState**, **passwordState**]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

So this now is actually an okay way of updating a state based on other state because with useEffect, we're guaranteed that this

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  **emailState.isValid && passwordState.isValid**  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

will run for every state update that React performs, which was not the case before,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(**() => {**  **const identifier = setTimeout(() => {**  **console.log('Checking form validity!');**  **setFormIsValid(**  **emailState.isValid && passwordState.isValid**  **);**  **}, 500);**  **return () => {**  **console.log('CLEANUP');**  **clearTimeout(identifier);**  **};**  **}**, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

where maybe the code ran too early.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  **// setFormIsValid(**  **// event.target.value.includes('@') && passwordState.isValid**  **// );**  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

useEffect only runs after state updates, so here setFormIsValid is okay.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  **setFormIsValid(**  **emailState.isValid && passwordState.isValid**  **);**  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

It turns out that we check the validity whenever the values changes. But if the input is valid already, and we add an extra character to the password, it doesn't really change. It is still valid. However, our effect still re-executes, which may be something that we want to avoid. The effect re-executes because our dependency is the entire emailState and the entire passwordState, not just the validity part of it, which is the part that we're interested in.

Here we can use a technique known as object destructuring, which is the same as array destructuring, just with objects, to pull out certain properties of objects. For example, from emailState, we can pull out isValid and store it in a new constant with the same name.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  **const { isValid } = emailState;**  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We can do the same for passwordState but

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid } = emailState;  **const {} = passwordState;**  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

since I'm already using isValid in this constant now,

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { **isValid** } = emailState;  const {} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We can use a variation of the object de-structuring syntax. We can add a colon here, which now won't assign a value because we're not creating an object here, as we're using it on the left side of the equal sign. We now assign an alias to the extracted property. So, we're pulling out the isValid property, but we're storing it in a constant, which actually could be named emailIsValid.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid**: emailIsValid** } = emailState;  const {} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

We can do the same for the password. We are pulling out the isValid property, but we can store it in a passwordIsValid constant.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid**:** emailIsValid } = emailState;  const { **isValid: passwordIsValid**} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

This (bolded in code snippet below) is an alias assignment, not a value assignment.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid**: emailIsValid** } = emailState;  const { isValid: **passwordIsValid**} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailState, passwordState]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Because we now have the emailIsValid and passwordIsValid constants, we can use them in the dependencies of useEffect rather than the entire state.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid**:** emailIsValid} = emailState;  const { isValid: passwordIsValid} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailState.isValid && passwordState.isValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [**emailIsValid**, **passwordIsValid**]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

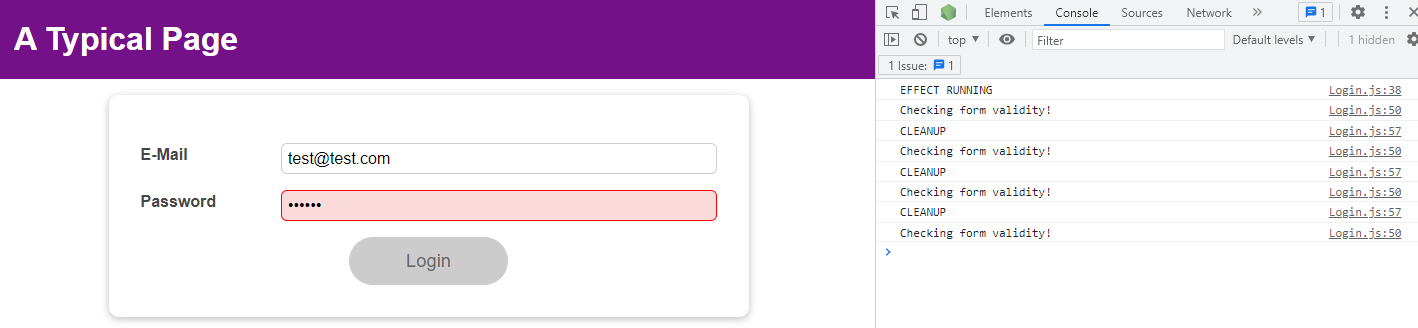
emailIsValid and passwordIsValid should also be used as our values in the useEffect function, as your dependencies should be matched by your values, and your values should be matched by your dependencies.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid: emailIsValid } = emailState;  const { isValid: passwordIsValid} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  **emailIsValid** && **passwordIsValid**  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailIsValid, passwordIsValid]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

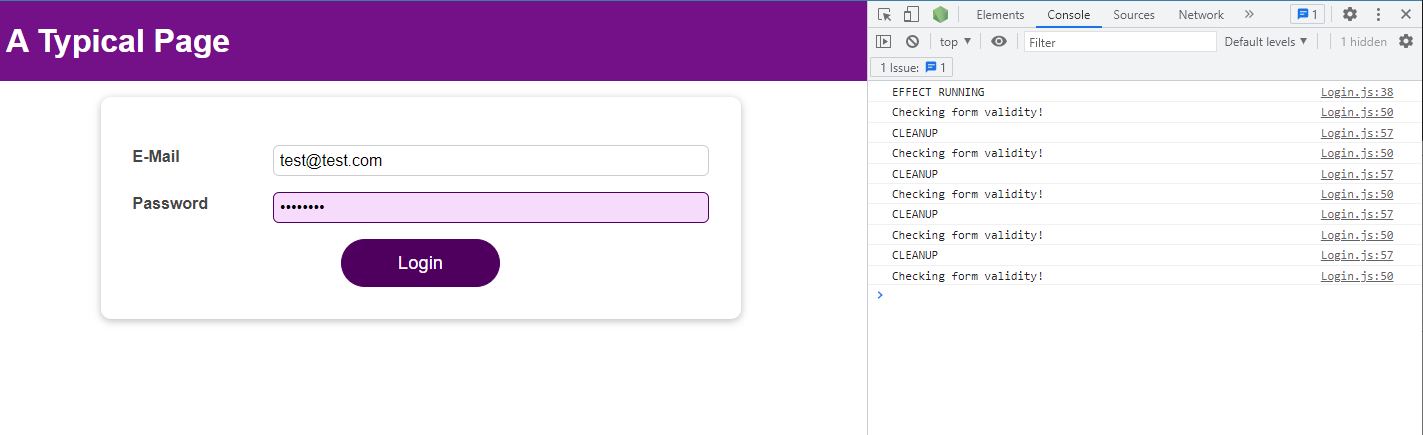
The advantage now is that I'm pulling out the isValid state here, whenever just the value changes and the validity does not change, the effect will not re-run.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { **isValid**: emailIsValid } = emailState;  const { **isValid**: passwordIsValid} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailIsValid && passwordIsValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailIsValid, passwordIsValid]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  props.onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

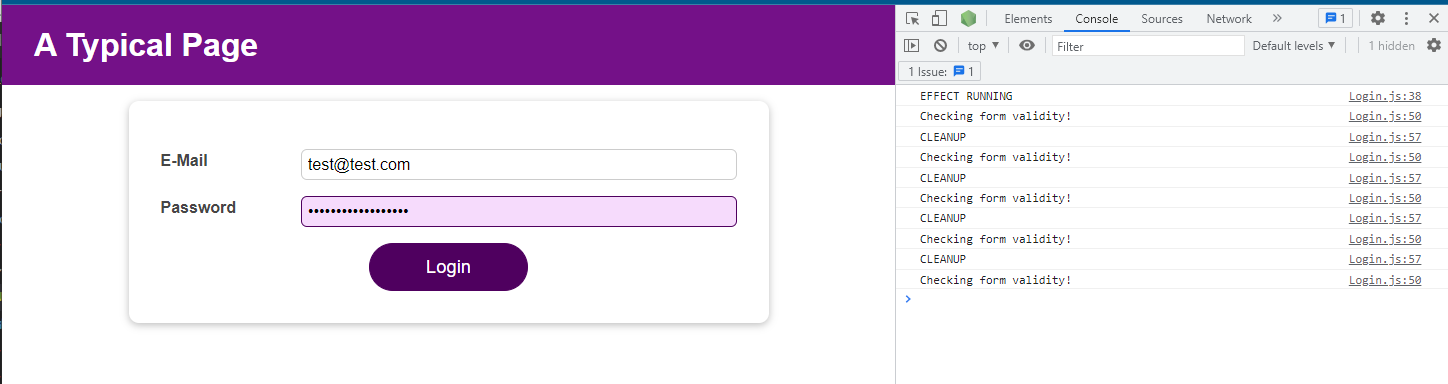
When we save and reload the browser, we can see that we get "Checking form validity" our password still hasn't met the default length requirements.



We meet the length requirement.



If we keep typing in the password field, we see that the effect does not run again.



### 118. Adding Nested Properties AS Dependencies To useEffect

We can use object destructuring to add object properties as dependencies to useEffect().

|  |
| --- |
| const { someProperty } = someObject;  useEffect(() => {  // code that only uses someProperty …  }, [someProperty]); |

The key thing is NOT that we use destructuring but that we pass specific properties instead of the entire object as a dependency.

We could also write this code and it would work in the same way.

|  |
| --- |
| useEffect(() => {  // code that only uses someProperty …  }, [someObject.someProperty]); |

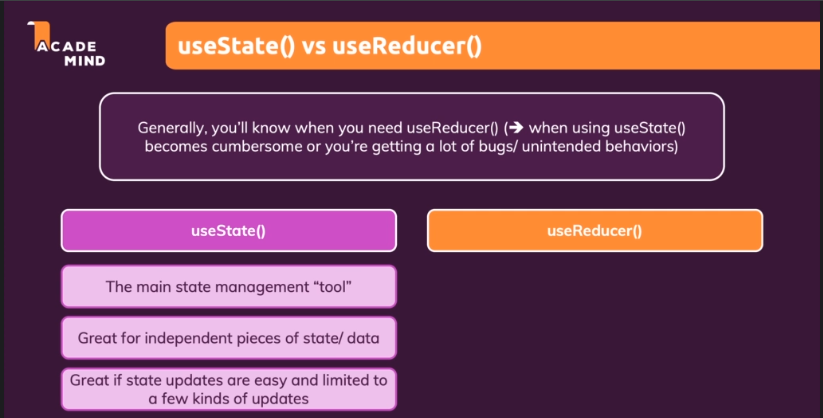
You should avoid this code:

|  |
| --- |
| useEffect(() => {  // code that only uses someProperty …  }, [someObject]) |

The reason we should avoid the snippet immediately above this one is because now the effect function would re-run whenever ANY property of someObject changes – not just the one property (someProperty in the above code snippet) our effect might depend on.

### 119. Use Reducer vs useState for State Management

You should use useReducer instead of useState when using useState becomes too cumbersome.



If you have an object as state or a more complex state, you may want to use useReducer.



### 120. Introducing React Context (Context API)

A "problem" that may happen is that you are passing a lot of data through a lot of components via props.

We use the login state in a lot of different places in the application.

It is quite common that you pass data to components through props, but it's always a problem if you forward state through multiple components. So if you basically just leverage props to forward data to another component.

In the App component I am passing isLoggedIn

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={**isLoggedIn**} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

through the isAuthenticated prop

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader **isAuthenticated**={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

to MainHeader.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <**MainHeader** isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

Also, in the App component I am passing logoutHandler

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={**isLoggedIn**} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

through the onLogout prop

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader **isAuthenticated**={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

to MainHeader as well.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <**MainHeader** isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

In the App component I am passing isLoggedIn

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader isAuthenticated={**isLoggedIn**} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

through the isAuthenticated prop

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <MainHeader **isAuthenticated**={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

to MainHeader.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  <**MainHeader** isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </React.Fragment>  );  }  export default App; |

In MainHeader, I am using neither of those two props. I am using them only to forward them again to the Navigation component because

|  |
| --- |
| src/components/MainHeader/MainHeader.js |
| import React from 'react';  import Navigation from './Navigation';  import classes from './MainHeader.module.css';  const MainHeader = (props) => {  return (  <header className={classes['main-header']}>  <h1>A Typical Page</h1>  <Navigation **isLoggedIn={props.isAuthenticated} onLogout={props.onLogout}** />  </header>  );  };  export default MainHeader; |

it's the Navigation component, which needs the isLoggedIn data to render different links and which needs the onLogout prop (and the logoutHandler therefore) to

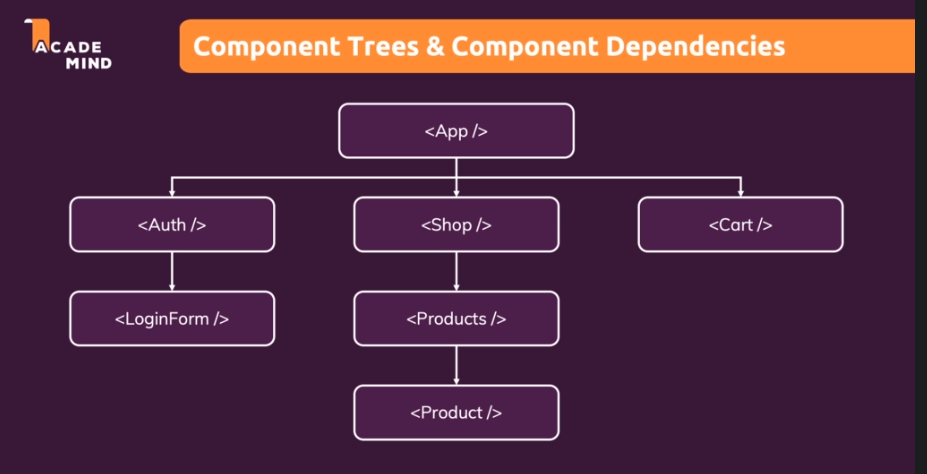
|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <nav className={classes.nav}>  <ul>  {**props.isLoggedIn** && (  <li>  <a href="/">Users</a>  </li>  )}  {**props.isLoggedIn** && (  <li>  <a href="/">Admin</a>  </li>  )}  {**props.isLoggedIn** && (  <li>  <button onClick={**props.onLogout**}>Logout</button>  </li>  )}  </ul>  </nav>  );  };  export default Navigation; |

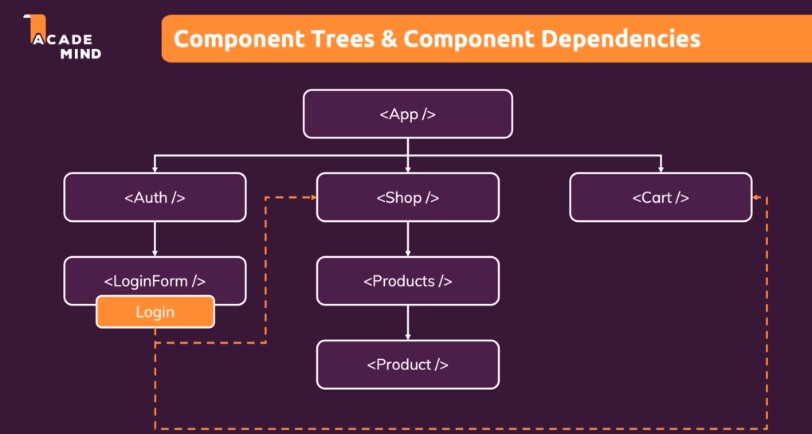
do something when the "Logout" button is clicked.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <nav className={classes.nav}>  <ul>  {props.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {props.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {props.isLoggedIn && (  <li>  **<button onClick={props.onLogout}>Logout</button>**  </li>  )}  </ul>  </nav>  );  };  export default Navigation; |

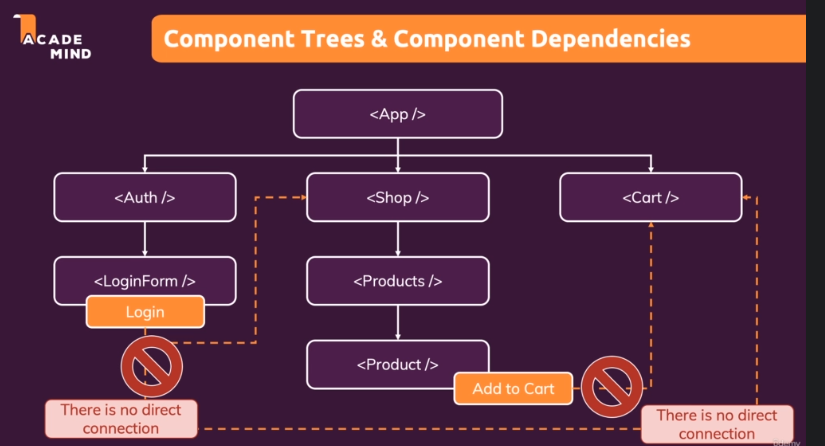
This is not necessarily a problem, but in bigger apps, that chain of forwarding might become longer and longer.

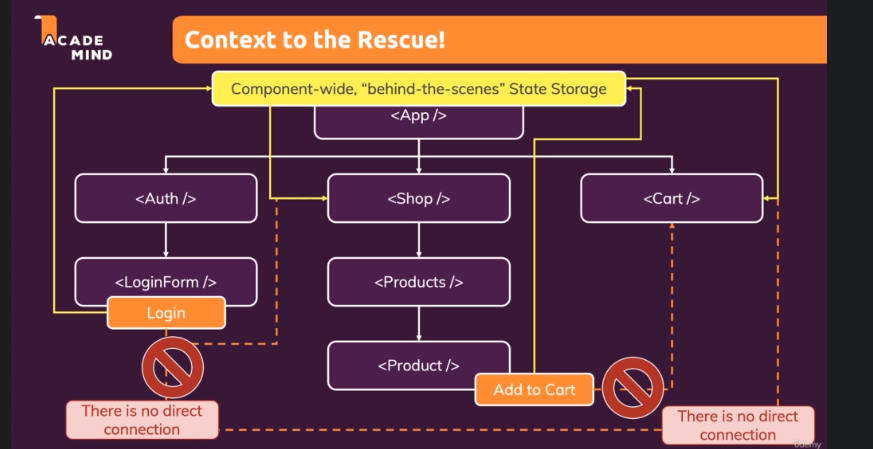
Consider a fictional app.



We might need information about the login status in different parts of the application.

We also see that here there are no direct connections





### 121. Using the React Context API

We will create a new folder next to our "components" folder called "store". We will create a new file in "store", and since we are going to manage some authentication state, we will name the file "auth-context.js". We will use kebab case to name our context files, so we don't imply that we have a component in that file, as that's not exactly what we're doing.



We import React from 'react'; and then on React we call createContext(), which does as the name implies; it creates such a context object.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  React.createContext(); |

createContext takes a default context, and context here is really just your app- or component-wide state. It's up to you what that State should be. It can be something simple like a string. If your app- or component-wide state is just some text, you could have some text as your default State.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  React.createContext(**"my state!"**); |

Often, though, your default state will be an object where in our example you manage the isLoggedIn state and set it to false.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  React.createContext(**{**  **isLoggedIn: false**  **}**); |

We get back an object that contains components from createContext(), so in this case we get back AuthContext. We name it like this because while AuthContext itself is not a component, it is an object that will contain a component.

|  |
| --- |
| src/store/auth-context.js |
| const **AuthContext** = React.createContext({  isLoggedIn: false  }); |

We don't need this object here, but we will need it in the other components, so we will export it as a default to make it available in other files.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  isLoggedIn: false  });  **export default AuthContext;** |

To use context in your application, you need to do two things:

1. You need to provide it, which basically tells React, "Hey, here's my Context." All components that are wrapped by it should have access to it.
2. Besides providing, you then need to consume it. You need to hook into it; you need to listen to it.

Providing is always the first step. Providing means that you wrap in JSX code all of the components that should be able to tap into that Context. So that should be able to listen to that Context. Any component that's not wrapped will not be able to listen.

If we know that we need AuthContext everywhere in the entire application. So in all components possibly, then we want to wrap everything in the App component with it. If we knew that we only needed it in the Login component and its child components, we would only wrap the Login component.

Therefore, in the App component, we can wrap the MainHeader and this main section

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  **<MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />**  **<main>**  **{!isLoggedIn && <Login onLogin={loginHandler} />}**  **{isLoggedIn && <Home onLogout={logoutHandler} />}**  **</main>**  </React.Fragment>  );  }  export default App; |

with AuthContext and we will also need to import it. AuthContext will not be a component. It is an object, so with a dot we can access a property on this AuthContext object that contains a component, and that is the Provider. AuthContext.Provider is a component that we can use in our JSX code, and we can wrap it around other components, and those other components and all their decedent components will now have access to the context.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <React.Fragment>  **<AuthContext.Provider>**  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  **</AuthContext.Provider>**  </React.Fragment>  );  }  export default App; |

Because we now have AuthContext.Provider as a wrapping component, we can remove React.Fragment.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

Now that we have done the first step of context. We now need to do the second step, and that is the listening part. To get access to our value, which at the moment is always false by default,

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  **isLoggedIn: false**  });  export default AuthContext; |

we need to listen. We can listen in two ways. We can listen by using AuthContext.Consumer, or by using a react hook.

Let's see how we can use AuthContext.Consumer. Let's say in Navigation, we're interested in knowing whether the user is authenticated or not. For that we can use AuthContext and wrap everything where we need data from it with that Consumer. So here, we could wrap the entire component in AuthContext.Consumer.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  **<AuthContext.Consumer>**  <nav className={classes.nav}>  <ul>  {props.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {props.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {props.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  **</AuthContext.Consumer>**  );  };  export default Navigation; |

The consumer works a bit differently, though. The consumer takes a child, which actually should be a function.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <AuthContext.Consumer>  **{() => {}}**  <nav className={classes.nav}>  <ul>  {props.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {props.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {props.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  </AuthContext.Consumer>  );  };  export default Navigation; |

As an argument to that function, you'll get your context data, ctx.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <AuthContext.Consumer>  {(**ctx**) => {}}  <nav className={classes.nav}>  <ul>  {props.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {props.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {props.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  </AuthContext.Consumer>  );  };  export default Navigation; |

So in our case, you would get this object:

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext(**{**  **isLoggedIn: false**  **}**);  export default AuthContext; |

In this function you should return your JSX code, which should have access to that data.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <AuthContext.Consumer>  {(ctx) => {  **return (**  **<nav className={classes.nav}>**  **<ul>**  **{props.isLoggedIn && (**  **<li>**  **<a href="/">Users</a>**  **</li>**  **)}**  **{props.isLoggedIn && (**  **<li>**  **<a href="/">Admin</a>**  **</li>**  **)}**  **{props.isLoggedIn && (**  **<li>**  **<button onClick={props.onLogout}>Logout</button>**  **</li>**  **)}**  **</ul>**  **</nav>**  **)**  }}  </AuthContext.Consumer>  );  };  export default Navigation; |

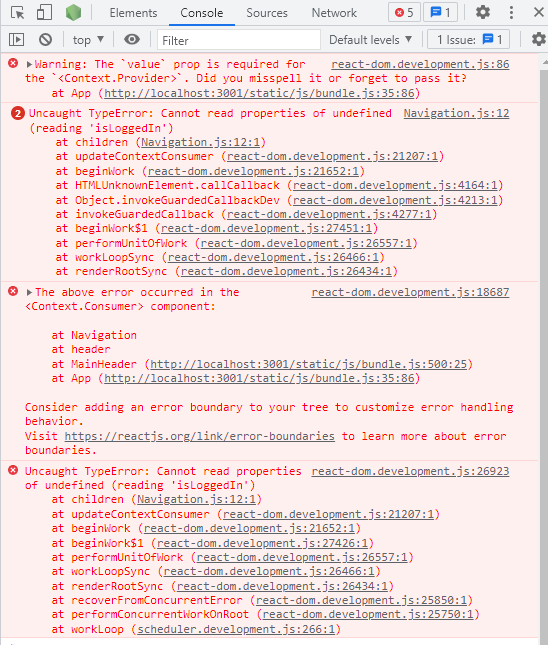
Within the function, rather instead of props, we now have access to ctx.isLoggedIn because

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <AuthContext.Consumer>  {(ctx) => {  return (  <nav className={classes.nav}>  <ul>  {**ctx**.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {props.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {props.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  )  }}  </AuthContext.Consumer>  );  };  export default Navigation; |

my context object is this object here and there I have the isLoggedIn prop.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext(**{**  **isLoggedIn: false**  **}**);  export default AuthContext; |

If we save and refresh, we see that the application crashes.



The reason that it crashes is because while we do have a default value here, the default value will only be used if

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext(**{**  **isLoggedIn: false**  **}**);  export default AuthContext; |

we consumed here

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  **<AuthContext.Consumer>**  **{(ctx) => {**  **return (**  **<nav className={classes.nav}>**  **<ul>**  **{ctx.isLoggedIn && (**  **<li>**  **<a href="/">Users</a>**  **</li>**  **)}**  **{props.isLoggedIn && (**  **<li>**  **<a href="/">Admin</a>**  **</li>**  **)}**  **{props.isLoggedIn && (**  **<li>**  **<button onClick={props.onLogout}>Logout</button>**  **</li>**  **)}**  **</ul>**  **</nav>**  **)**  **}}**  **</AuthContext.Consumer>**  );  };  export default Navigation; |

WITHOUT having a provider.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  **<AuthContext.Provider>**  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  **</AuthContext.Provider>**  );  }  export default App; |

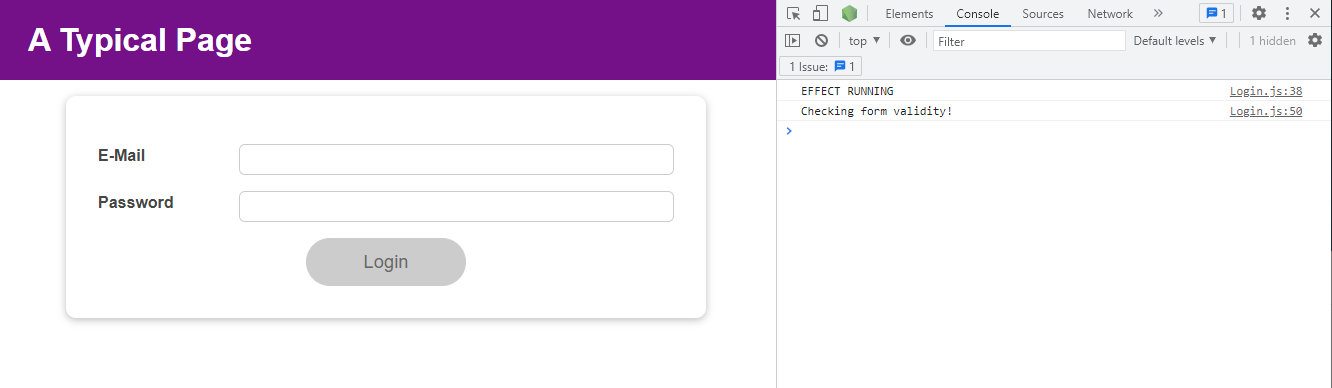
So technically the Provider is not really needed; however, you should memorize this pattern where you need the Provider. Technically, you don't need the Provider if you have a default value, but in reality, you will use context to have a value that can change, and that will only be possible with a Provider. So therefore, to make sure that this does not crash, we can go back to the Provider and there on the Provider component you can add a value prop. The value prop needs to be named value because Provider is not a component that you created.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider **value**={}>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

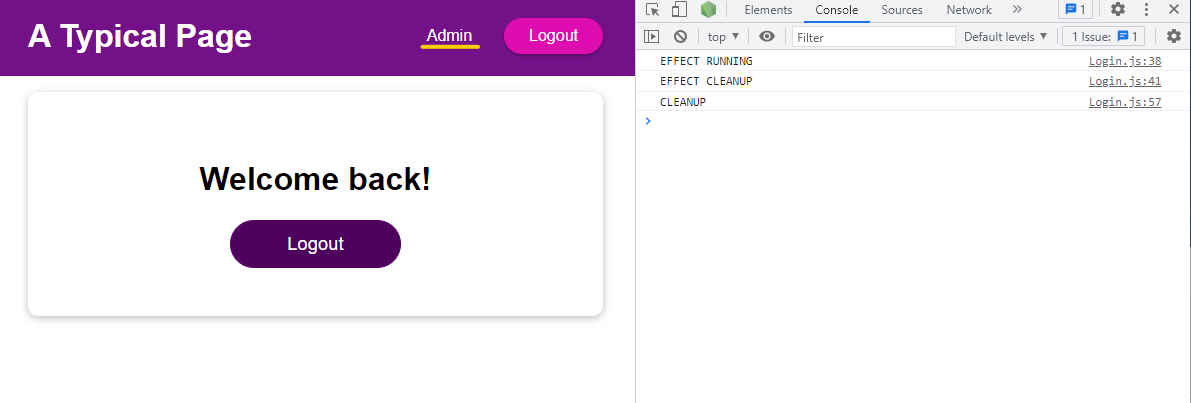
To the value, you can also pass your object. In this case, we will just copy the object from "auth-context.js", but now you would be able to change that object, for example, through state in the App component, and whenever it changes, the new value will be passed down to all consuming components.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={**{**  **isLoggedIn: false,**  **}**}>  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

If we save and reload the page, our application loads.



If we now, login, you will see that we only have one link here.



So now our "Users" link in Navigation is missing

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <AuthContext.Consumer>  {(ctx) => {  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">**Users**</a>  </li>  )}  {props.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {props.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  )  }}  </AuthContext.Consumer>  );  };  export default Navigation; |

because there I am pulling data from my context, and that at the moment is never changing.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <AuthContext.Consumer>  {(ctx) => {  return (  <nav className={classes.nav}>  <ul>  {**ctx.isLoggedIn** && (  <li>  <a href="/">Users</a>  </li>  )}  {props.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {props.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  )  }}  </AuthContext.Consumer>  );  };  export default Navigation; |

To change the context, we see that we have set up the Provider in the App component:

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  **<AuthContext.Provider**  **value={{**  **isLoggedIn: false,**  **}}**  **>**  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

And in the App component we manage the isLoggedIn state.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  **const [isLoggedIn, setIsLoggedIn] = useState(false);**  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={{  isLoggedIn: false,  }}  >  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

Rather than hardcoding the isLoggedIn value here,

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={{  isLoggedIn: **false**,  }}  >  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

we can set the isLoggedIn value to the state isLoggedIn.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={{  isLoggedIn: **isLoggedIn**,  }}  >  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

And if we do that, this value object will be updated by React

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value=**{{**  **isLoggedIn: isLoggedIn,**  **}}**  >  <MainHeader isAuthenticated={isLoggedIn} onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

whenever the isLoggedIn state changed and the new context object will be passed down to all listening components, so to all components where we consume this context. The difference to before is that we don't need to use a prop to forward that therefore. Instead we just set it on the Provider and then everywhere in all child components we could listen to that. So here we could remove isAuthenticated={isLoggedIn} from the MainHeader.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  }}  >  <MainHeader ~~isAuthenticated={isLoggedIn}~~ onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

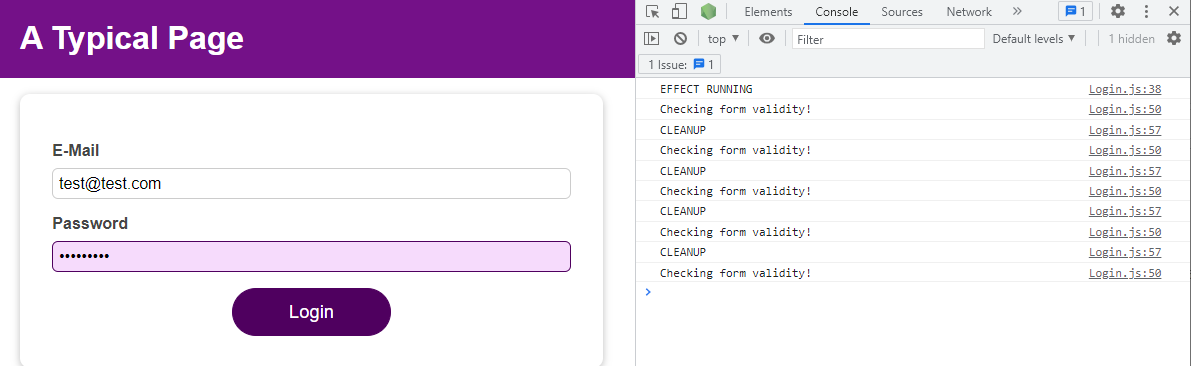
And in the MainHeader we can stop forwarding isAuthenticated to Navigation.

|  |
| --- |
| src/components/MainHeader/MainHeader.js |
| import React from 'react';  import Navigation from './Navigation';  import classes from './MainHeader.module.css';  const MainHeader = (props) => {  return (  <header className={classes['main-header']}>  <h1>A Typical Page</h1>  <Navigation ~~isLoggedIn={props.isAuthenticated}~~ onLogout={props.onLogout} />  </header>  );  };  export default MainHeader; |

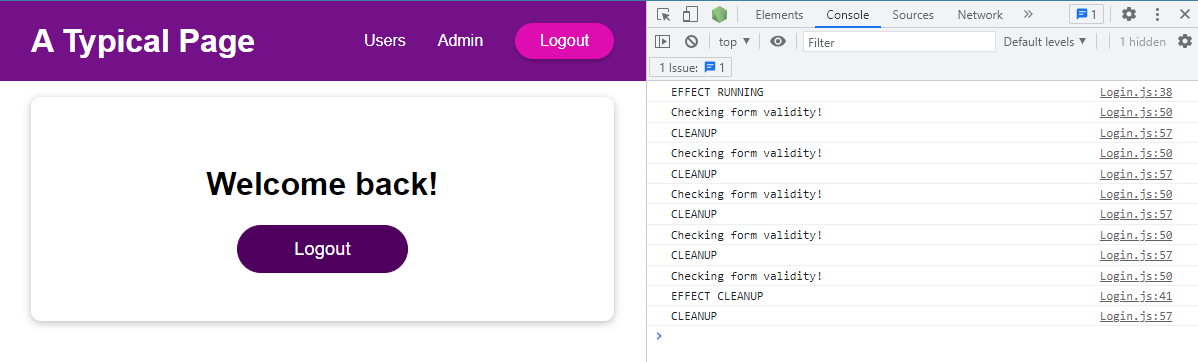
And in Navigation we can now use ctx.isLoggedIn everywhere where we need the isLoggedIn information. And that's now where we break up that props chain, where we stop forwarding that data that we don't need in the MainHeader and where we instead leverage context.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <AuthContext.Consumer>  {(ctx) => {  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {**ctx**.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {**ctx**.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  )  }}  </AuthContext.Consumer>  );  };  export default Navigation; |

So now if we refresh our page, we see that it still works and



when we Login, our links are back.



### 122. Tapping Into Context with the useContext Hook

We will explore the useContext hook and for that we will remove the Consumer:

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  ~~<AuthContext.Consumer>~~  ~~{(ctx) => {~~  ~~return (~~  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  )  ~~}}~~  ~~</AuthContext.Consumer>~~  );  };  export default Navigation; |

Now we are back to our old Navigation component but are referring to the ctx object, which doesn't exist anymore. We still are importing AuthContext, and we can use it again by

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  );  };  export default Navigation; |

importing a hook from React, and that's the useContext hook, another hook that is built into react, and it does what the name implies; it allows us to tap into the context and listen to it.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React, {**useContext**} from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  );  };  export default Navigation; |

To use useContext, you call it in your react component function and you pass a pointer at it for the context that you want to use, which in our case is AuthContext. You get back the context value, which in this case, we'll call ctx.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React, {useContext} from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  **const ctx = useContext(AuthContext);**  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <button onClick={props.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  );  };  export default Navigation; |

When we save and reload the browser, the code works like before.

### 123. Making Context Dynamic

We're using useContext to pass the isLoggedIn state to my other components. We are still forwarding the onLogout prop. We can set up a dynamic context where we don't just pass data to other components, but where we also pass functions. All we need to do is in the App component, in the AuthContextProvider where set up the value, besides passing down isLoggedIn, we can also pass down onLogout where we point at the logoutHandler.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  **onLogout: logoutHandler**,  }}  >  <MainHeader onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

If we do this, then every component that listens to our context will be able to utilize logout handler through the onLogout context value. Therefore, back in the Navigation component where we are referring to props.onLogout, we can do ctx.onLogout. This will work because

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React, {useContext} from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (props) => {  const ctx = useContext(AuthContext);  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <button onClick={**ctx**.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  );  };  export default Navigation; |

on the context object we now have the onLogout value.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  **onLogout**: logoutHandler,  }}  >  <MainHeader onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

If we save that, logging out as well as the rest of the app should work just fine. Since we're using ctx, we no longer need props in our Navigation component anymore, so we can remove it.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React, {useContext} from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = (~~props~~) => {  const ctx = useContext(AuthContext);  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <button onClick={ctx.onLogout}>Logout</button>  </li>  )}  </ul>  </nav>  );  };  export default Navigation; |

Now, in the App component, in the main section, we will still pass down the logoutHandler through onLogout, and the loginHandler through onLogin because we directly use those handlers in the Login and Home component.

|  |
| --- |
| src/App.js |
| import React, { useState, useEffect } from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  import AuthContext from './store/auth-context';  function App() {  const [isLoggedIn, setIsLoggedIn] = useState(false);  useEffect(() => {  const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');  if (storedUserLoggedInInformation === '1') {  setIsLoggedIn(true);  }  }, []);  const loginHandler = (email, password) => {  // We should of course check email and password  // But it's just a dummy/ demo anyways  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  };  const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  };  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  onLogout: logoutHandler,  }}  >  <MainHeader onLogout={logoutHandler} />  <main>  {!isLoggedIn && <**Login** **onLogin**={**loginHandler**} />}  {isLoggedIn && <**Home** **onLogout**={**logoutHandler**} />}  </main>  </AuthContext.Provider>  );  }  export default App; |

In the Login component we directly refer to props.onLogin and are not forwarding it.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid: emailIsValid } = emailState;  const { isValid: passwordIsValid} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailIsValid && passwordIsValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailIsValid, passwordIsValid]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  **props.onLogin(emailState.value, passwordState.value);**  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

In the Home component we are using props.onLogout.

|  |
| --- |
| src/components/Home/Home.js |
| import React from 'react';  import Card from '../UI/Card/Card';  import Button from '../UI/Button/Button';  import classes from './Home.module.css';  const Home = (props) => {  return (  <Card className={classes.home}>  <h1>Welcome back!</h1>  <Button onClick={props**.onLogout**}>Logout</Button>  </Card>  );  };  export default Home; |

Sure we are technically forwarding props.onLogout to our own Button component, but the Button component is a pure presentational component. We don't want to use context inside of our Button to always bind button clicks to onLogout because while it would remove the need to pass props.onLogout, it would mean that our Button would now always be logging the user out and not able to do anything else upon a click.

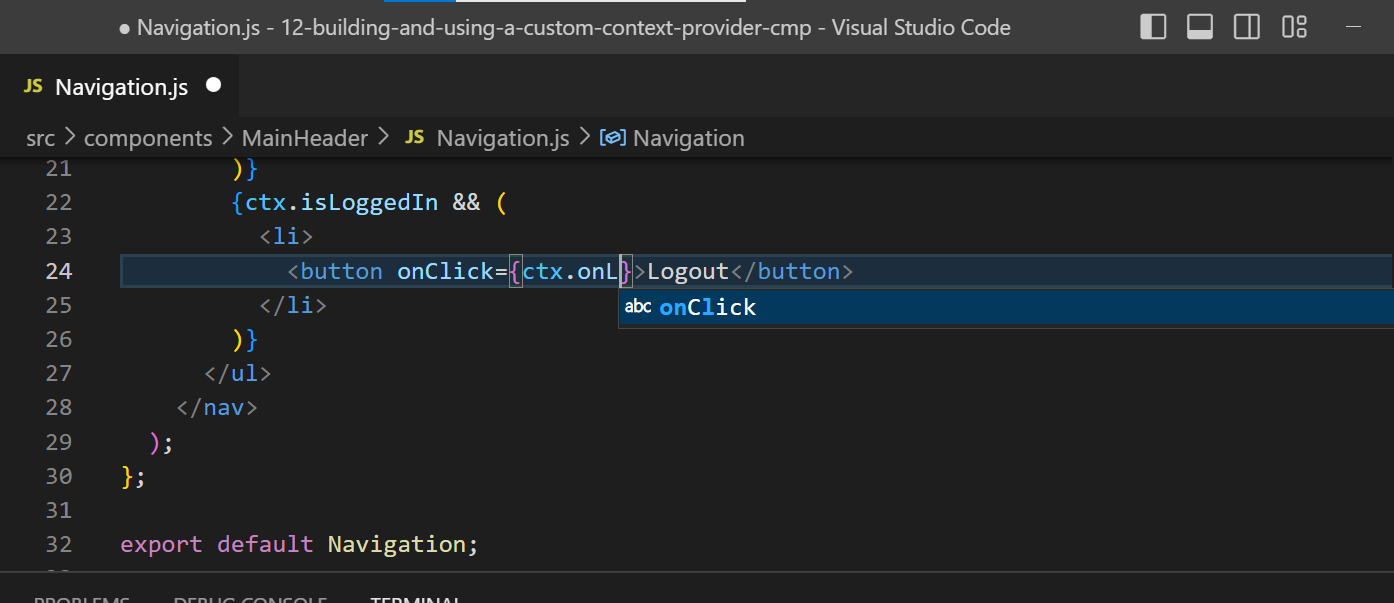
|  |
| --- |
| src/components/Home/Home.js |
| import React from 'react';  import Card from '../UI/Card/Card';  import Button from '../UI/Button/Button';  import classes from './Home.module.css';  const Home = (props) => {  return (  <Card className={classes.home}>  <h1>Welcome back!</h1>  <**Button** onClick={props**.**onLogout}>Logout</Button>  </Card>  );  };  export default Home; |

In most cases, you will use props to pass data to components because props are your mechanism to configure components and make them reusable. Only if you have something that you would forward through a lot of components, and you're forwarding it to a component that does something very specific like in the Navigation component, this button will always log the user out. In such cases, you want to consider context.

|  |
| --- |
| src/components/MainHeader/Navigation.js |
| import React, {useContext} from 'react';  import AuthContext from '../../store/auth-context';  import classes from './Navigation.module.css';  const Navigation = () => {  const ctx = useContext(AuthContext);  return (  <nav className={classes.nav}>  <ul>  {ctx.isLoggedIn && (  <li>  <a href="/">Users</a>  </li>  )}  {ctx.isLoggedIn && (  <li>  <a href="/">Admin</a>  </li>  )}  {ctx.isLoggedIn && (  **<li>**  **<button onClick={ctx.onLogout}>Logout</button>**  **</li>**  )}  </ul>  </nav>  );  };  export default Navigation; |

### 124. Building & Using a Custom Context Provider Component

It is a good idea to also add your functions like onLogout in our case to the default context when you create the context. There you can just store a dummy function, which does nothing. We do this for better completion by our IDE. Without the onLogout function in the default context object, if we're in Navigation, for example, and we try to call onLogout, we don't get a suggestion for it by our IDE. It doesn't know that there is an onLogout prop, but it does have the isLoggedIn suggestion.



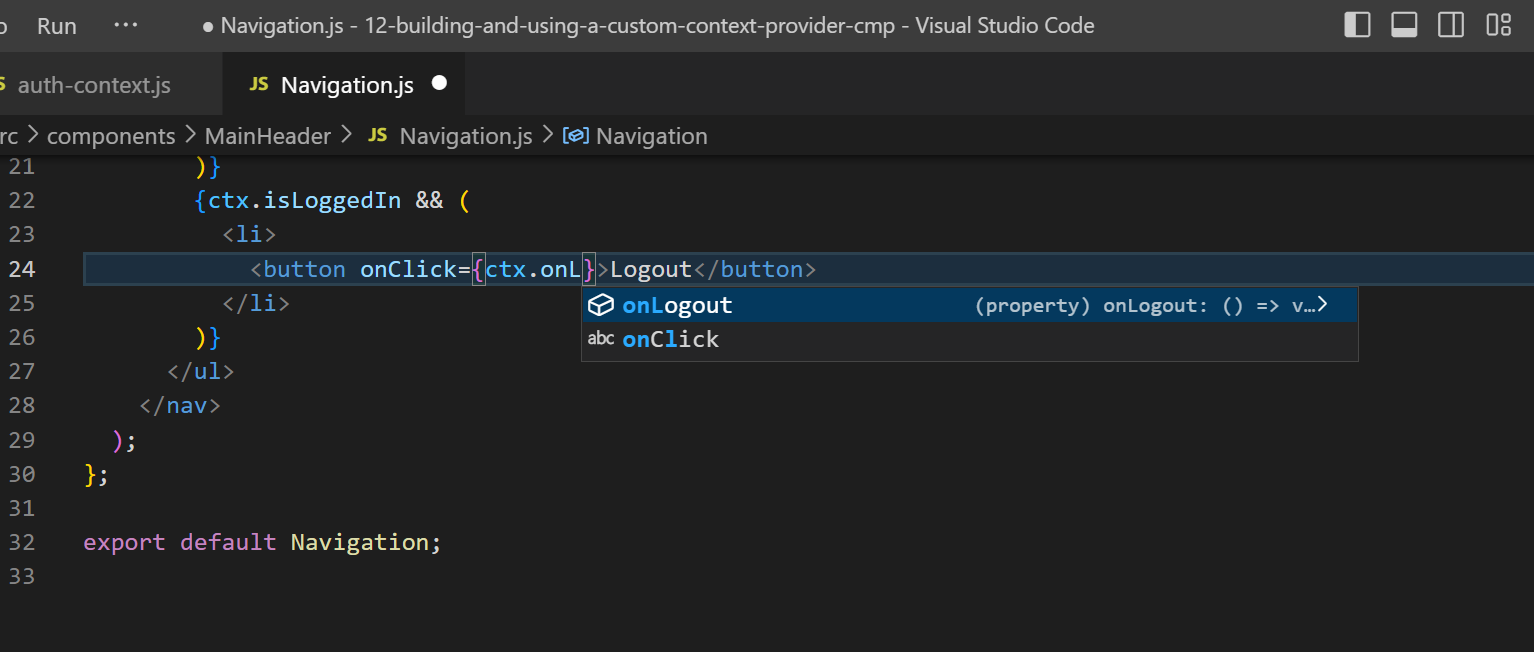
The IDE is looking at this default context object to find out what it is able to access on the context.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext(**{**  **isLoggedIn: false**  **}**);  export default AuthContext; |

So for better IDE autocompletion, it is a good idea to add the onLogout prop in the default context object just with a dummy function since we are not going to use the function's value in there any way.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  **onLogout: () => {}**  });  export default AuthContext; |

Now, in Navigation we do get this onLogout autocompletion suggestion.



Another thing that you might consider is depending on your application's structure and how you're managing your data, you might want to pull more logic out of your App component, for example, and create a separate context management component. How would this look? Well, in our "auth-context.js" file we could create const authContextProvider component.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  **const AuthContextProvider = () => {**    **}**  export default AuthContext; |

We then can return <AuthContext.Provider>, and

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  const AuthContextProvider = () => {  **return (**  **<AuthContext.Provider>**  **</AuthContext.Provider>**  **)**  }  export default AuthContext; |

we accept props.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  const AuthContextProvider = (**props**) => {  return (  <AuthContext.Provider>  </AuthContext.Provider>  )  }  export default AuthContext; |

We then simply pass whatever we get on props.children between our provider.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  const AuthContextProvider = (props) => {  return (  <AuthContext.Provider>  **{props.children}**  </AuthContext.Provider>  )  }  export default AuthContext; |

We then export AuthContext.Provider in addition to the default export as a named export.

|  |
| --- |
| src/store/auth-context.js |
| import React from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  **export** const AuthContextProvider = (props) => {  return (  <AuthContext.Provider>  {props.children}  </AuthContext.Provider>  )}  export default AuthContext; |

We do this because now we can import useState.

|  |
| --- |
| src/store/auth-context.js |
| import React, **{useState}** from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  export const AuthContextProvider = (props) => {  return (  <AuthContext.Provider>  {props.children}  </AuthContext.Provider>  )  }  export default AuthContext; |

and manage the isLoggedIn state and setIsLoggedIn here inside of AuthContextProvider.

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  export const AuthContextProvider = (props) => {  **const [isLoggedIn, setIsLoggedIn] = useState(false);**  return (  <AuthContext.Provider>  {props.children}  </AuthContext.Provider>  )  }  export default AuthContext; |

We also can provide the logoutHandler and setIsLoggedIn to false.

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  export const AuthContextProvider = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  **const logoutHandler = () => {**  **setIsLoggedIn(false);**  **}**  return (  <AuthContext.Provider>  {props.children}  </AuthContext.Provider>  )  }  export default AuthContext; |

We can now also add the loginHandler function here, so that we manage the entire authentication state in this separate provider component.

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  export const AuthContextProvider = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const logoutHandler = () => {  setIsLoggedIn(false);  }  **const loginHandler = () => {**  **setIsLoggedIn(true);**  **}**  return (  <AuthContext.Provider>  {props.children}  </AuthContext.Provider>  );  }  export default AuthContext; |

In AuthContext.Provider we then set the value to an object with a key of isLoggedIn, which is set to the isLoggedIn state, and we have the onLogout key that points at the logoutHandler.

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  export const AuthContextProvider = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const logoutHandler = () => {  setIsLoggedIn(false);  }  const loginHandler = () => {  setIsLoggedIn(true);  }  return (  <AuthContext.Provider **value={{ isLoggedIn: isLoggedIn, onLogout: logoutHandler }}**>  {props.children}  </AuthContext.Provider>  )  }  export default AuthContext; |

In addition, we then also have an onLogin key with a value pointing at the loginHandler

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {}  });  export const AuthContextProvider = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const logoutHandler = () => {  setIsLoggedIn(false);  }  const loginHandler = () => {  setIsLoggedIn(true);  }  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  onLogout: logoutHandler,  **onLogin: loginHandler,**  }}  >  {props.children}  </AuthContext.Provider>  )}  export default AuthContext; |

Therefore, we will now also add onLogin to our default here as a dummy function with the parameters of email and password just to make it clear what we're expecting but are not doing anything inside of the function.

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {},  **onLogin: (email, password) => {}**  });  export const AuthContextProvider = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const logoutHandler = () => {  setIsLoggedIn(false);  }  const loginHandler = () => {  setIsLoggedIn(true);  }  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  onLogout: logoutHandler,  onLogin: loginHandler,  }}  >  {props.children}  </AuthContext.Provider>  )  }  export default AuthContext; |

------ continue at 3:31

Now we have this standalone file, which manages the entire login state in this AuthContextProvider component and which also sets up all of the context.

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {},  onLogin: (email, password) => {}  });  export const **AuthContextProvider** = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const logoutHandler = () => {  setIsLoggedIn(false);  }  const loginHandler = () => {  setIsLoggedIn(true);  }  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  onLogout: logoutHandler,  onLogin: loginHandler,  }}  >  {props.children}  </AuthContext.Provider>  )}  export default AuthContext; |

We need to move over our local storage access from the App component into the handler functions in AuthContext.Provider.

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {},  onLogin: (email, password) => {}  });  export const AuthContextProvider = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  const logoutHandler = () => {  **localStorage.removeItem('isLoggedIn');**  setIsLoggedIn(false);  }  const loginHandler = () => {  **localStorage.setItem('isLoggedIn', '1');**  setIsLoggedIn(true);  }  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  onLogout: logoutHandler,  onLogin: loginHandler,  }}  >  {props.children}  </AuthContext.Provider>  )  }  export default AuthContext; |

We also want to move over our useEffect code to the AuthContextProvider and therefore, we also need to import useEffect

|  |
| --- |
| src/store/auth-context.js |
| import React, {useState, **useEffect**} from 'react';  const AuthContext = React.createContext({  isLoggedIn: false,  onLogout: () => {},  onLogin: (email, password) => {}  });  export const AuthContextProvider = (props) => {  const [isLoggedIn, setIsLoggedIn] = useState(false);  **useEffect(() => {**  **const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');**    **if (storedUserLoggedInInformation === '1') {**  **setIsLoggedIn(true);**  **}**  **}, []);**    const logoutHandler = () => {  localStorage.removeItem('isLoggedIn');  setIsLoggedIn(false);  }  const loginHandler = () => {  localStorage.setItem('isLoggedIn', '1');  setIsLoggedIn(true);  }  return (  <AuthContext.Provider  value={{  isLoggedIn: isLoggedIn,  onLogout: logoutHandler,  onLogin: loginHandler,  }}  >  {props.children}  </AuthContext.Provider>  )  }  export default AuthContext; |

We can now clear all of the logic out of the App component that we moved into AuthContextProvider, remove the hook imports. Remove the "auth-context" import. Also, remove AuthContext.Provider from App.

|  |
| --- |
| src/App.js |
| import React, ~~{ useState, useEffect }~~ from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  ~~import AuthContext from './store/auth-context';~~  function App() {  ~~const [isLoggedIn, setIsLoggedIn] = useState(false);~~  ~~useEffect(() => {~~  ~~const storedUserLoggedInInformation = localStorage.getItem('isLoggedIn');~~  ~~if (storedUserLoggedInInformation === '1') {~~  ~~setIsLoggedIn(true);~~  ~~}~~  ~~}, []);~~  ~~const loginHandler = (email, password) => {~~  ~~// We should of course check email and password~~  ~~// But it's just a dummy/ demo anyways~~  ~~localStorage.setItem('isLoggedIn', '1');~~  ~~setIsLoggedIn(true);~~  ~~};~~  ~~const logoutHandler = () => {~~  ~~localStorage.removeItem('isLoggedIn');~~  ~~setIsLoggedIn(false);~~  ~~};~~  return (  ~~<AuthContext.Provider~~  ~~value={{~~  ~~isLoggedIn: isLoggedIn,~~  ~~onLogout: logoutHandler,~~  ~~}}~~  ~~>~~  <MainHeader onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  ~~</AuthContext.Provider>~~  );  }  export default App; |

We then also have to bring back React.Fragment. Because now we can wrap the App component as a

|  |
| --- |
| src/App.js |
| import React from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  return (  **<React.Fragment>**  <MainHeader onLogout={logoutHandler} />  <main>  {!isLoggedIn && <Login onLogin={loginHandler} />}  {isLoggedIn && <Home onLogout={logoutHandler} />}  </main>  **</React.Fragment>**  );  }  export default App; |

whole in "index.js" by wrapping App with AuthContextProvider, which gets exported in the "auth-context" file. With that, we have one central place for the auth state management.

|  |
| --- |
| src/index.js |
| import React from 'react';  import ReactDOM from 'react-dom/client';  import './index.css';  import App from './App';  **import { AuthContextProvider } from './store/auth-context';**  const root = ReactDOM.createRoot(document.getElementById('root'));  root.render(  **<AuthContextProvider>**  <App/>  **</AuthContextProvider>**  ); |

The central place is now not the App component but a dedicated context component and a dedicated context file. In the App component we now want to import { useContext }, and pass the AuthContext object to useContext, which is our default export in the "auth-context" file. Therefore, we also add ctx to isLoggedIn and ctx to !isLoggedIn. We then also remove onLogin and onLogout props here because we are not manage the login and logout handlers in the App component now anyway.

|  |
| --- |
| src/App.js |
| import React, **{ useContext }** from 'react';  import Login from './components/Login/Login';  import Home from './components/Home/Home';  import MainHeader from './components/MainHeader/MainHeader';  function App() {  **const ctx = useContext(AuthContext);**  return (  <React.Fragment>  <MainHeader ~~onLogout={logoutHandler}~~ />  <main>  {**!ctx.isLoggedIn** && <Login ~~onLogin={loginHandler}~~ />}  {**ctx.isLoggedIn** && <Home ~~onLogout={logoutHandler}~~ />}  </main>  </React.Fragment>  );  }  export default App; |

Now we can tap into context in the Home and Login components.

In the Home component we also import { useContext} , we get access to our const authCtx = useContext(AuthContext); We can then use authCtx.onLogout

|  |
| --- |
| src/components/Home/Home.js |
| import React, **{ useContext }** from 'react';  import Card from '../UI/Card/Card';  import Button from '../UI/Button/Button';  import classes from './Home.module.css';  **import AuthContext from '../../store/auth-context';**  const Home = (props) => {  **const authCtx = useContext(AuthContext);**  return (  <Card className={classes.home}>  <h1>Welcome back!</h1>  <Button onClick={**authCtx.onLogout**}>Logout</Button>  </Card>  );  };  export default Home; |

In the Login component we can do something similar. We import { useContext }. In our component function we can set up const authCtx = useContext(AuthContext). For that make sure you are importing "auth-context". We then use authCtx when we need to login.

|  |
| --- |
| src/components/Login/Login.js |
| import React, { useState, useEffect, useReducer, **useContext** } from 'react';  import Card from '../UI/Card/Card';  import classes from './Login.module.css';  import Button from '../UI/Button/Button';  **import AuthContext from '../../store/auth-context';**  const emailReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.includes('@') };  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.includes('@')};  }  return {value: '', isValid: false};  };  const passwordReducer = (state, action) => {  if (action.type === 'USER\_INPUT') {  return { value: action.val, isValid: action.val.trim().length > 6}  }  if (action.type === 'INPUT\_BLUR') {  return { value: state.value, isValid: state.value.trim().length > 6}  }  return {value: '', isValid: false};  }  const Login = (props) => {  // const [enteredEmail, setEnteredEmail] = useState('');  // const [emailIsValid, setEmailIsValid] = useState();  // const [enteredPassword, setEnteredPassword] = useState('');  // const [passwordIsValid, setPasswordIsValid] = useState();  const [formIsValid, setFormIsValid] = useState(false);  const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null});  const [passwordState, dispatchPassword] = useReducer(passwordReducer, {value: '', isValid: null});  **const authCtx = useContext(AuthContext);**  useEffect(() => {  console.log('EFFECT RUNNING');  return () => {  console.log('EFFECT CLEANUP');  };  }, []);  const { isValid: emailIsValid } = emailState;  const { isValid: passwordIsValid} = passwordState;  useEffect(() => {  const identifier = setTimeout(() => {  console.log('Checking form validity!');  setFormIsValid(  emailIsValid && passwordIsValid  );  }, 500);  return () => {  console.log('CLEANUP');  clearTimeout(identifier);  };  }, [emailIsValid, passwordIsValid]);  const emailChangeHandler = (event) => {  dispatchEmail({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // event.target.value.includes('@') && passwordState.isValid  // );  };  const passwordChangeHandler = (event) => {  dispatchPassword({type: 'USER\_INPUT', val: event.target.value});  // setFormIsValid(  // emailState.isValid && event.target.value.trim().length > 6  // );  };  const validateEmailHandler = () => {  dispatchEmail({type: 'INPUT\_BLUR'});  };  const validatePasswordHandler = () => {  dispatchPassword({type: 'INPUT\_BLUR'});  };  const submitHandler = (event) => {  event.preventDefault();  **authCtx.**onLogin(emailState.value, passwordState.value);  };  return (  <Card className={classes.login}>  <form onSubmit={submitHandler}>  <div  className={`${classes.control} ${  emailState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="email">E-Mail</label>  <input  type="email"  id="email"  value={emailState.value}  onChange={emailChangeHandler}  onBlur={validateEmailHandler}  />  </div>  <div  className={`${classes.control} ${  passwordState.isValid === false ? classes.invalid : ''  }`}  >  <label htmlFor="password">Password</label>  <input  type="password"  id="password"  value={passwordState.value}  onChange={passwordChangeHandler}  onBlur={validatePasswordHandler}  />  </div>  <div className={classes.actions}>  <Button type="submit" className={classes.btn} disabled={!formIsValid}>  Login  </Button>  </div>  </form>  </Card>  );  };  export default Login; |

Our code now still works when we save and refresh our browser.

### 125. React Context Limitations

Context is not a replacement for component configuration. For example, the Button, if we used context, we could make sure that on a click, we logged a user out; however, that would mean that we couldn't use Button for anything else than logging users out. That might not be what you want because in this same application, we are using the Button for users to Login and trigger the form submission. That is a scenario where using context in the Button component would be bad. You want to use props in the Button component.

React Context is NOT optimized for high frequency changes, like state changes multiple times per second; then React context is not optimized for that.

React Context should also not be used to replace ALL component communications and props. props are still vital for component configuration. Components should still be configurable via props and short "prop chains" might not need any replacement.