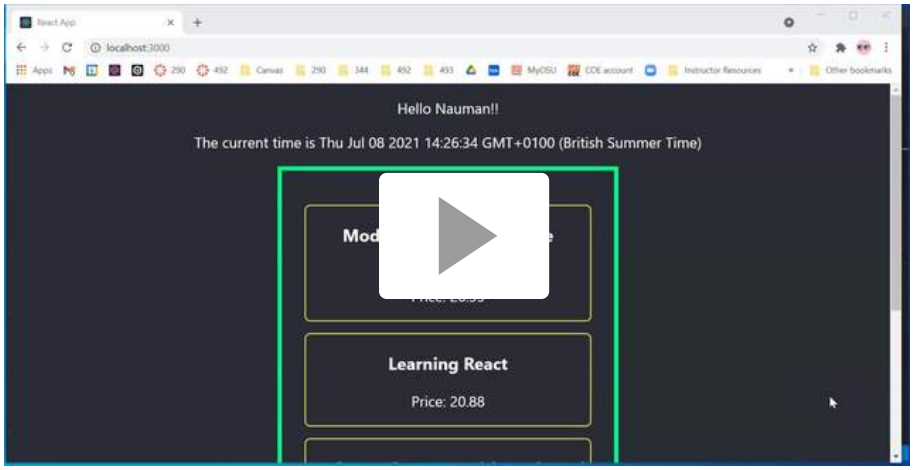
Exploration — State & React Hooks

Introduction

In this exploration, we will study state variables and their use in React applications. State variables in a React application hold data that when changed can cause the DOM tree displayed in the browser to be updated. In fact, when writing React applications, we do not write code to directly update the DOM tree. Instead, we change the value of state variables and the React framework then makes the needed changes to the DOM. React applications use state for many things, including the following:

- Giving visual feedback based on user actions. Examples include:
  - Showing and hiding items in a shopping cart.
  - Changing the icon when a bookmark is added or removed from an item.
  - Modifying a selected menu item.
- Updating the page based on data retrieval, for example, from a file, from a database, or from a web service.
- Updating the page based on a timer.

Example: Bookmark Component



> The bookmarked-books React app is available in this file: **bookmarked-books.zip (https://canvas.oregonstate.edu/courses/1879154/files/93831939?wrap=1)** ↓ (https://canvas.oregonstate.edu/courses/1879154/files/93831939/download?download_frd=1)

We will explain the concept and use of state by creating a new component named `Bookmark`, and adding it to our previous example which displays a list of books. The `Bookmark` component will provide the functionality to bookmark items displayed in an app. By using this component, our app will now have the following additional functionality:

- Each book will now display an icon.
- The icon displayed for a book will be one of two icons, with the choice determined by whether the book is currently bookmarked or not.
- Initially, all books will not be bookmarked and will show the corresponding icon.
- When someone clicks the icon, the icon will change:
  - If the book is currently not bookmarked, then the click adds a bookmark and the icon changes to the bookmark icon.
  - If the book is currently bookmarked, then the clicks removes the bookmark and the icon changes to the one that indicates the book is not bookmarked.

Here is the code for the component `Bookmark`.

```
import React, { useState } from 'react';
import { MdBookmark, MdBookmarkBorder } from 'react-icons/md';

function Bookmark() {
  const [isBookmarked, setBookmarked] = useState(false);
  const toggleBookmark = () => setBookmarked(!isBookmarked);
  return (
    <>
      {isBookmarked
        ? <MdBookmark onClick={toggleBookmark} />
        : <MdBookmarkBorder onClick={toggleBookmark} />
      }
    </>
  );
}

export default Bookmark;
```

We save this component in a file `Bookmark.js` in the `src` directory within the `book-list` directory. Let us now dissect and understand this component.

> **Note:** The Bookmark component uses the Conditional (or Ternary) Operator that was discussed in **Exploration — Conditionals and Loops (https://canvas.oregonstate.edu/courses/1879154/pages/exploration-conditionals-and-loops)**. If needed, review the relevant section in that exploration.

Using React Icons

We are using two icons from **the library react-icons** (https://react-icons.github.io/react-icons/). `react-icons` is a library containing hundreds of icons that we can use as React components. These icons in turn come from more than 20 other sources. The two icons we will use come from the **Material Design icons** (http://google.github.io/material-design-icons/). These are:

- `MdBookmark` which displays a selected bookmark, and
- `MdBookmarkBorder` which displays the border of a bookmark, i.e., an unselected bookmark.

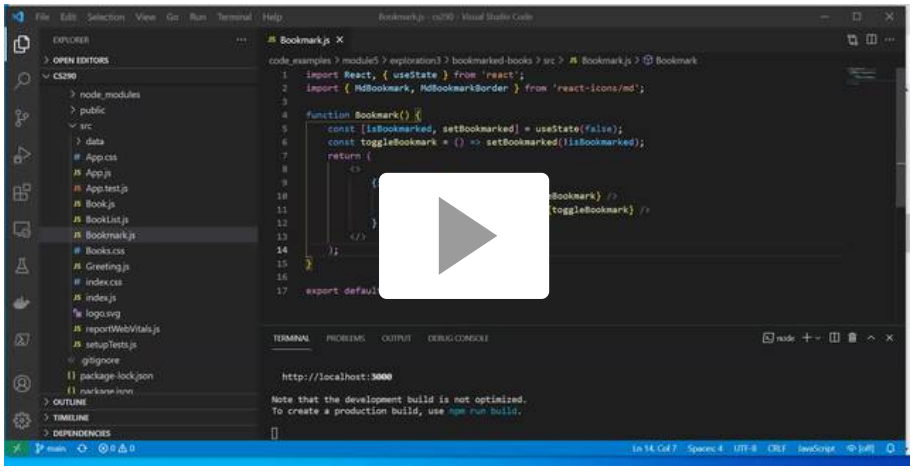
To use the React components for these icons, we install the library `react-icons`. To do this, go the directory `book-list` and run the following command:

```
npm i react-icons --save
```

In the file for our `Bookmark` component, we import the React components for these icons using the following statement:

```
import { MdBookmark, MdBookmarkBorder } from 'react-icons/md';
```

Creating State Variables



We create state variables using the function `useState()`. This function is available in the `react` library and we import it using the following statement:

```
import React, { useState } from 'react';
```

The function `useState` takes one argument, which is the initial value of the state variable, and returns an array:

- The first element in the array is the state variable that models the state of our component.
- The second element in the array is a function that takes one argument and sets the state variable to this argument. We must use this function to update the value of the state variable. Otherwise, React will not automatically render this component even when its state variable changes.

We call `useState` as follows in our example component `Bookmark`:

```
const [isBookmarked, setBookmarked] = useState(false);
```

In this statement we are using array destructuring to destructure the array returned by `useState`:

- The first element, i.e., the state variable, is assigned to the variable `isBookmarked`.
- The second element, i.e., the state update function, is assigned to the variable `setBookmarked`.

If we think about the state of the component `Bookmark`, we see that we can model it as a Boolean value since at any given time it has one of two values:

- It is bookmarked. We model this as the value `true`.
- It is not bookmarked. We model this as the value `false`.

Since on initial display we want all books to be unbookmarked, we pass the value `false` to `useState` which means that the initial value of `isBookmarked` is set to false.

Changing the State Variable

> **Note:** We use function expression and arrow function syntax in our Bookmark component. If needed, review this topic in **Exploration — Functions and Functional Programming (https://canvas.oregonstate.edu/courses/1879154/pages/exploration-functions-and-functional-programming)** .

As stated earlier, we want the following behavior for our component:

- When someone clicks the icon, the icon will change:
  - If the book is currently not bookmarked, then the click adds a bookmark and the icon changes to the bookmark icon.

- If the book is currently bookmarked, then the clicks removes the bookmark and the icon changes to the one that indicates the book is not bookmarked.

We can achieve this by changing the state of the variable `isBookmarked` on a click as follows:

- If `isBookmarked` is false, then it should be set to true.
- If `isBookmarked` is true, then it should be set to false.

The following function `toggleBookmark` has this functionality:

```
const toggleBookmark = () => setBookmarked(!isBookmarked);
```

Let's understand this statement

- We are defining a function `toggleBookmark` using a function expression which is to the right of the = sign
- The function doesn't take any arguments which is indicated by the empty parenthesis `()`.
- The function is defined using the arrow syntax.
- The function body consists of only one expression, so do not need to enclose the body in `{}` and don't need a return statement.

Note that this function does not directly update the value of the variable `isBookmarked`. Instead it calls `setBookmarked` with the new value to set for the state variable `isBookmarked`. The call to `setBookmarked` triggers a re-rendering of the component by the React framework.

Handling Events

In order to invoke `toggleBookmark` on a user click on the icon displayed by `Bookmark`, we need to register it as an event handler. In the component `Bookmark`, we register it as the handler for click events on the components `MdBookmark` and `MdBookmarkBorder` as follows:

```
  <MdBookmark onClick={toggleBookmark} />
  <MdBookmarkBorder onClick={toggleBookmark} />
```

As we can see, event handling in React is fairly similar to regular HTML and JavaScript. We register event handler code using properties that are based on the name of the event. However, there are a few differences worth noting from regular HTML and JavaScript as listed below:

1. Name of property is in camelCase.
   - In React the name of the property to register a handler for an event uses camelCase, whereas in regular HTML the property name is all lower case.
   - For example, to register a handler for the `click` event on an element, in React the property name is `onClick`, whereas in regular HTML it is `onclick`, i.e., `onClick={}` versus `onclick=""`.
2. Use curly braces to insert the function or the function name.

- In React, the event handler is registered using a JSX expression. So we need to wrap it inside `{}`.

3. Do not put parentheses after the function given as the value of the event handler.
   - In React, the JSX expression is evaluated before the value of the property is passed to the component.
   - If we put parentheses after the function, then the function will be executed and its return value will be passed as the value of the handler, rather than the function itself!

Conditional Rendering of Components

At a given time, one of the two icons is displayed. We achieve this in `Bookmark` using the following piece of code:

```
return (
  <>
    {isBookmarked
        ? <MdBookmark onClick={toggleBookmark} />
        : <MdBookmarkBorder onClick={toggleBookmark} />
    }
  </>
);
```

Here we use JavaScript's **conditional or ternary operator (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator)** to return:

- The component `MdBookmark` if `isBookmarked` is true, or
- The component `MdBookmarkBorder` if `isBookmarked` is false.

Note that when a user clicks on the displayed icon, the value of `isBookmarked` is toggled, i.e.,

- If it were `true` it is set to `false`, and
- If it were `false`, it is set to `true`.

Thus, when the component `Bookmark` is re-rendered the other icon will be displayed.

Exercise: Counter Component

Create a React component named `Counter` that models the control provided in web pages to select the quantity of an item in a shopping cart.

- Use a variable `count` to model the state of the component.
- Use the icons `MdAdd` from **Material Design Icons (https://react-icons.github.io/react-icons/icons?name=md)** .
- Clicking `MdAdd` should increment the count.

Hooks

The `useState` function is part of a React feature called **Hooks** which was introduced in React version 16.8 released in February 2019. React Hooks are code logics that are used to connect functionality to our components. Most importantly, the functionality in a React Hook can cause a component to be re-rendered. React provides many other Hooks out-of-the box. In addition, React support defining Custom Hooks. We are going to study and use additional Hooks later in the course.

Summary

In this exploration, we studied the concept of state. We saw how using state variables we can trigger updates to the DOM tree without having to directly call any direct DOM API. Instead, changes to state variable are detected by the React framework and it takes care of efficiently re-rendering the relevant part of the DOM tree.

Although, the value of state is maintained as the component is re-rendered, the state is still temporary in that it is held in a variable in the browser. If we wanted the updated state to be persisted across reloads of the page, we would need to store that data in a persistent store, such as a database, and then load it in the component. This is a topic we will study later in the course.

Additional Resources

Here are some references to learn more about the topics we discussed in this exploration.

- A lot more information on React Hooks is available in React's official docs. Start with the **Introduction (https://reactjs.org/docs/hooks-intro.html)** and you can continue on with many other topics related to Hooks.
- For more discussion of `useState` see **this page (https://reactjs.org/docs/hooks-state.html)**.