REACT HOOKS (III)

YT Course by The Net Ninja / June 2019

Mind map by Ana-Maria Dobre

Part2

Next, inside BookList.js, use useContext.
Caution: if you try to use hooks within a CC, you get an error such as (1)

Here is the current definition of BookList.js (2)
// that we have wrongly added hooks to!

Convert BookList to FC and add useContext (3)

Dealing with errors
a) Warning: Each child in a list should have a unique "key" prop.
// fix by adding key property as such (4)

b) books.map is not a function
// Check BookContextProvider where you've defined books, you may have forgotten to add the square brackets indicative of an array

c) if, by mistake, you place useContext inside callback:
React Hook "useContext" cannot be called inside a callback. React Hooks must be called in a React function component or a custom React Hook function
// fix by placing it outside callback

(1)
react-dom.development.js:14906

Uncaught Error: Invalid hook call. Hooks can only be called inside of the body of a function component.
This could happen for one of the following reasons:
1. You might have mismatching versions of React and the renderer (such as React DOM)
2. You might be breaking the Rules of Hooks
3. You might have more than one copy of React in the same app

```
(2)
import React, {Component, useContext} from 'react';
import { BookContext } from '../contexts/BookContext';
import { ThemeContext } from '../contexts/ThemeContext';

class BookList extends Component
{
  render() {
    return(
      <ThemeContext.Consumer>{(context2) => {
        const { isLightTheme, light, dark } = context2;
        const { books } = useContext(BookContext);
        const theme = isLightTheme ? light : dark;
        return(
          <div className='book-list' style = {{color: theme.syntax, background: theme.bg}}>
            <ul>
              {books.map(book => {
                return(
                <li>{book.title}</li>
                );
              })}
              <li>hardcoded book title1</li>
              <li>hardcoded book title2</li>
              <li>hardcoded book title3</li>
            </ul>
          </div>
        );
      }}
      </ThemeContext.Consumer>
    );
  }
}

export default BookList;

(3)
import React, {Component, useContext} from 'react';
import { BookContext } from '../contexts/BookContext';
import { ThemeContext } from '../contexts/ThemeContext';

const BookList = () => {
  const { books } = useContext(BookContext);    // good placement for useContext
  return (
    <ThemeContext.Consumer>{(context2) => {
      const { isLightTheme, light, dark } = context2;
      const theme = isLightTheme ? light : dark; // wrong placement for useContext at this level (inside cb)
      return(
        <div className='book-list' style = {{color: theme.syntax, background: theme.bg}}>
          <ul>
            {books.map(book => {
              return(
              <li>{book.title}</li>
              );
            })}
          </ul>
        </div>
      );
    }}
    </ThemeContext.Consumer>
  );
}
export default BookList;

(4)
<li key={book.id}>{book.title}</li>
```

```
(1)
import React, {Component} from 'react';
import { ThemeContext } from '../contexts/ThemeContext';

class BookList extends Component
{
  render() {
    return(
      <ThemeContext.Consumer>{(context2) => {
        const { isLightTheme, light, dark } = context2;
        const theme = isLightTheme ? light : dark;
        return(
          <div className='book-list' style = {{color: theme.syntax, background: theme.bg}}>
            <ul>
              <li>Corazon salvaje</li>
              <li>Poetry by Rumi</li>
              <li>Three men in a boat</li>
            </ul>
          </div>
        );
      }}
      </ThemeContext.Consumer>
    );
  }
}

export default BookList;

(2)
import React, {createContext, useState} from "react";

export const BookContext = createContext();

const BookContextProvider = (props) => {
  const [books, setBooks] = useState([
    {title: 'Corazon salvaje', id: 1},
    {title: 'Poetry by Rumy', id: 2},
    {title: 'Three men in a boat', id: 3}
  ]);
  return (
    <BookContext.Provider value={{books}}>
      {props.children}
    </BookContext.Provider>
  );
}

export default BookContextProvider;

(3)
<BookContextProvider>
  <BookList />
</BookContextProvider>
```

Creating context with FC (Part1)

Note: Looking inside the BookList.js component (1), we are outputting a hardcoded list of books.

Idea: create a context for that data (2)
// may call it BookContext.js
- import React, as well as hooks: createContext, useState
- export const BookContext and set it equal to createContent()
// that creates our context
- sfc TAB shortcut
- FC may be called BookContextProvider, has props
// when we want to output the children inside the provider, we want to use props.
In a CC, the props are attached to this; but, in a FC, we take them in as a param.
- Inside of the Provider, use useState to define some data; that data is going to be an array of books.
// remember useState returns 2 things inside an array:
1) the data itself 2) a fct to update the data
- return a provider // BookContext.Provider
attaching the value property so we can pass down the books
- wrap the children inside the provider // {props.children}

Next: import the BookContextProvider into App.js, so we can wrap the BookList component (3)

useContext HOOK

Random tip: How to install snippets
View > Extensions (Ctrl+Shift+X) > search "Simple React Snippets" > Install
=> sfc (stateless FC) followed by TAB will generate code snippets.

Have a look at the BookList CC, **before adding useContext (1)**

Here is BookList converted into a FC, **after adding useContext (2)**

Notes:
- both (1) and (2) return the same JSX
- for approach (2) import useContext, use a hook to get the context object and destructure different properties from that object
- matter of preference, but doesn't (2) seem cleaner / nicer than (1)?

```
(1)
import React, {Component} from 'react';
import {AuthContext} from '../contexts/AuthContext';

class BookList extends Component
{
  static contextType = AuthContext;
  render() {
    const {isAuthenticated, anotherProperty} = this.context;
    return (
      <div className='book-list'>
        {isAuthenticated? 'Logged in': 'Logged out'} <br/>
        Book list: <br/>
        <ul>
          <li>Three men in a boat</li>
          <li>Rumi's book of poetry</li>
          <li>{anotherProperty}</li>
        </ul>
      </div>
    );
  }
}

(2)
import React, {useContext} from 'react';
import {AuthContext} from '../contexts/AuthContext';

const BookList = () => {
  const {isAuthenticated, anotherProperty} = useContext(AuthContext);
  return (
    <div className='book-list'>
      {isAuthenticated? 'Logged in': 'Logged out'} <br/>
      Book list: <br/>
      <ul>
        <li>Three men in a boat</li>
        <li>Rumi's book of poetry</li>
        <li>{anotherProperty}</li>
      </ul>
    </div>
  );
}

export default BookList;
```

Practice (1)

Practice by converting CC ThemeToggle (1) into a FC that uses useContext (2)

```
(1)
import React, { Component } from "react";
import { ThemeContext } from "../contexts/ThemeContext";

class ThemeToggle extends Component
{
  static contextType = ThemeContext;
  render() {
    const { toggleTheme } = this.context;
    return(
      <button onClick={toggleTheme}>Toogle the theme</button>
    );
  }
}

export default ThemeToggle;

(2)
import React, { useContext } from "react";
import { ThemeContext } from "../contexts/ThemeContext";

const ThemeToggle = () => {
  const {toggleTheme} = useContext(ThemeContext);
  return (
    <button onClick={toggleTheme}>Toogle the theme</button>
  );
}
export default ThemeToggle;
```

Practice with multiple contexts using hooks

Note: You can use useContext as many times as needed within a FC to consume different contexts!

```
(1)
import React, {Component} from 'react';
import { AuthContext } from '../contexts/AuthContext';
import { ThemeContext } from '../contexts/ThemeContext';

class Navbar extends Component
{
  render() {
    return (
      <AuthContext.Consumer>{(authContext) => (
        <ThemeContext.Consumer>{(themeContext) => {
          console.log(themeContext);
          console.log(authContext);
          const { isAuthenticated, toggleAuth } = authContext;
          const { isLightTheme, light, dark } = themeContext;
          const theme = isLightTheme ? light : dark;
          return(
            <nav style = {{ background: theme.ui, color: theme.syntax }}>
            <h1>Context App</h1>
            <div onClick={toggleAuth}>
              { isAuthenticated? 'Logged in': 'Logged out'}
            </div>
            <ul>
              <li>Home</li>
              <li>About</li>
              <li>Contact</li>
            </ul>
          </nav>
          );
        }}
        </ThemeContext.Consumer>
      )}</AuthContext.Consumer>
    );
  }
}
export default Navbar;

(2)
import React, { useContext } from 'react';
import { AuthContext } from '../contexts/AuthContext';
import { ThemeContext } from '../contexts/ThemeContext';

const Navbar = () => {
  const {isLightTheme, light, dark} = useContext(ThemeContext);
  const theme = isLightTheme ? light : dark;
  const {isAuthenticated, toggleAuth} = useContext(AuthContext);

  return (
  <nav style = {{ background: theme.ui, color: theme.syntax }}>
    <h1>Context App</h1>
    <div onClick={toggleAuth}>
      {isAuthenticated? 'Logged in': 'Logged out'}
    </div>
    <ul>
      <li>Home</li>
      <li>About</li>
      <li>Contact</li>
    </ul>
  </nav>
  );
}

export default Navbar;
```

Legend

FC: functional components
CC: class components
cb: callback