03 Building a Cart using Context

In the previous tutorial we've learned how to define and manage states with React. We can define states using the useState hook to describe the updatable elements in our UI. We also learned how to share data between components with states and props.

However, in some situation, we need to define some data that should be shared globally, such as user information and shopping cart items. It will be very tedious to pass states through props component-by-component. Therefore, React introduces a concept called **context** to manage global states.

But firstly, let's take a look at how to create a multi-page application with React since we are getting more and more pages for different functions.

1 React router

React Router is the standard library used in React to change views and navigate between pages. For a large web applicatoin, the page routing could be very complicated. In this section, we will build a shopping cart page using React Router.

Installation

Run this command in your terminal to install the React Router dependencies.

```
npm install react-router-dom
```

Routes

Now, we will use Router to route to pages based on URL. Let's modify our `App.js` to the following codes:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Navbar from "./components/Navbar";
import Home from "./pages/home/Home";
import "bootstrap/dist/css/bootstrap.min.css";

function App() {
  return (
    <BrowserRouter>
      <Navbar brand="iBookStore" />
      <Routes>
        <Route path="/" element={<Home />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

First, we wrap our `App` component with `<BrowserRouter>`. Then, we define a `<Routes>` to wrap different views for different pages. An application can have multiple `<Routes>`. Here we only need to use one. The `<Navbar>` component is placed outside the `<Routes>` so that for each page in `<Routes>` will have the same navigation bar.

Inside `<Routes>`, we can define multiple pages with different URL addresses. Here we add a `<Route>` component with a path "/" and a element of our `<Home>` component, meaning that when user access to the root address, the home page component will be shown.

Cart Page

Now, let's add a new page for our shopping cart. We need four files to complete the page:

- `Cart.js`
- `Cart.module.css`
- `CartItem.js`
- `CartItem.module.css`

For the CSS files, please reference the complete code with the GitHub link: https://github.com/lvyin1122/ecommerce-tutorial-code/tree/03-building-a-cart-using-context.

`pages/cart/Cart.js`

```
import React from "react";
import CartItem from "../../components/CartItem.js";
import classes from "./Cart.module.css";
import data from "../../productsData.js";

function Cart() {
  const cartList = data.map((item) => (
    <CartItem
      key={item.id}
      id={item.id}
      name={item.name}
      author={item.author}
      price={item.price}
      image={item.image}
    />
  ));

  return (
    <div className={classes.cart}>
      <h1>My Shopping Cart</h1>
      <ul className={classes.cartList}>{cartList}</ul>
    </div>
  );
}

export default Cart;
```

`components/CartItem.js`

```
import React from "react";
import classes from "./CartItem.module.css";

function CartItem(props) {
  return (
    <li className={classes.cartItem}>
      <img src={props.image} alt="" />
      <div className={classes.center}>
        <h3>{props.name}</h3>
        <p>By {props.author}</p>
      </div>
      <div className={classes.right}>
        <span>${props.price}</span>
        <span className={classes.remove}>Remove</span>
      </div>
    </li>
  );
}

export default CartItem;
```

Notices that we directly import all the fake data to our cart for demonstration. We will change it later.

Now, we add a new `<Route>` to `App.js` with the new path `/cart`. The codes are quite straight forward:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Navbar from "./components/Navbar";
import Home from "./pages/home/Home";
import Cart from "./pages/cart/Cart";
import "bootstrap/dist/css/bootstrap.min.css";

function App() {
  return (
    <BrowserRouter>
      <Navbar brand="iBookStore" />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/cart" element={<Cart />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Link Component

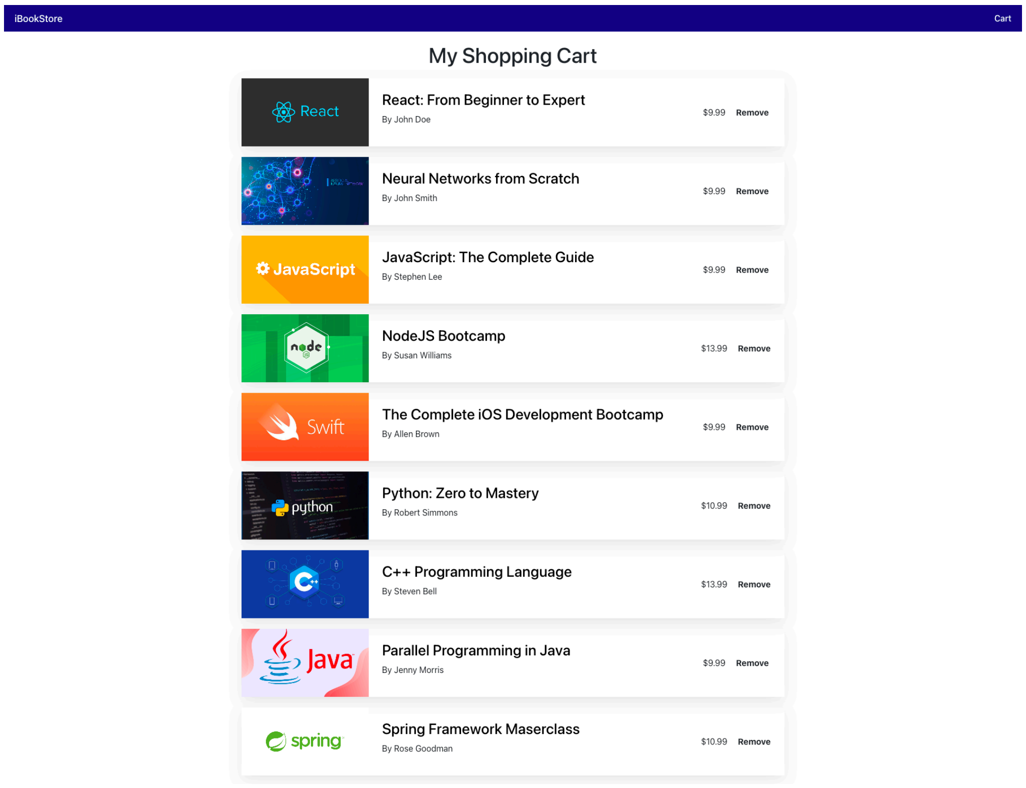
`<Link>` is a component that lets the user navigate to another page by clickling on it. We can easily improve our `Navbar.js` like this:

```
import React from "react";
import classes from "./Navbar.module.css";
import { Link } from "react-router-dom";

function Navbar(props) {
  return (
    <div className={classes.navbar}>
      <Link to="/" className={classes.brand}>
        {props.brand}
      </Link>
      <Link to="/cart" className={classes.cart}>
        Cart
      </Link>
    </div>
  );
}

export default Navbar;
```

Here, we wrap the brand name and cart text in our navigation bar with `<Link>` respectively. The `to` attribute tells where it will direct to after the user clicks on it.

Now, when we click on the brand, we can navigate to the home page. When we click on the "Cart", we will move to the cart page.

Currently, our cart page looks like this:


iBookStore Cart

My Shopping Cart

React: From Beginner to Expert
By John Doe
$9.99 Remove

Neural Networks from Scratch
By John Smith
$9.99 Remove

JavaScript: The Complete Guide
By Stephen Lee
$9.99 Remove

NodeJS Bootcamp
By Susan Williams
$13.99 Remove

The Complete iOS Development Bootcamp
By Allen Brown
$9.99 Remove

Python: Zero to Mastery
By Robert Simmons
$10.99 Remove

C++ Programming Language
By Steven Bell
$13.99 Remove

Parallel Programming in Java
By Jenny Morris
$9.99 Remove

Spring Framework Maserclass
By Rose Goodman
$10.99 Remove

2 useContext

For now, the data in your shopping cart is hard-coded. We are gonna fix it with context and the `useContext` hook, which allows us to store and manage global information accross components.

The concepts involved in the following sections will be a little bit hard to understand. You are highly recommended to go with this tutorial and do it step-by-step.

The first thing to do is create a context called `CartContext` with the method `React.createContext()`. Let's create a file called `cart-context.js` and put it under a new directory `src/store`. The code is like this:

`src/store/cart-context.js`

```
import React from "react";

// Create the cart context
const CartContext = React.createContext();

// Export CartContext
export default CartContext;
```

Next, we should define what kind of data we want to share with the context. We can simpliy initialize the data as a JavaScript variable. But generally, we should define a state with the `useState` hook to make it changable.

Another concept is called **provider**. A provider of a context is a wrapper component that makes every **child components** of it be able to access the data stored in the context. Let's define the provider for `CartContext`:

```
import { createContext, useState } from "react";

// Create the cart context
const CartContext = createContext();

// Define initial state for cart items, this hard-coded data is just for demonstration
const initialCartState = [
  {
    id: 0,
    name: "React: From Beginner to Expert",
    author: "John Doe",
    price: 9.99,
    image:
      "https://cdn.thenewstack.io/media/2022/05/600b72f9-react-1024x680.png",
  },
];

// Define CartProvider component
export function CartProvider(props) {
  // Define a state called cartState with useState hook
  // The initial value is defined as initialCartState
  const [cartState, setCartState] = useState(initialCartState);

  // Pass the state to the provider with the value attribute
  return (
    <CartContext.Provider value={cartState}>
      {props.children}
    </CartContext.Provider>
  );
}

// Export CartContext
export default CartContext;
```

Here, we first define the state called `cartState` with a initial value. Then, we pass the state to the context provider with the attribute `value`.

For our `CartProvider` component, we should define it with the `CartContext.Provider` component. The `props.children` inside the component refers to the children components of it, so that when we wrap the whole application with the provider, the components inside the provider can be shown.

To let every component access to this global state, we should wrap all the components with our provider.

`App.js`

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Navbar from "./components/Navbar";
import Home from "./pages/home/Home";
import Cart from "./pages/cart/Cart";
import { CartProvider } from "./store/cart-context";
import "bootstrap/dist/css/bootstrap.min.css";

function App() {
  return (
    <CartProvider>
      <BrowserRouter>
        <Navbar brand="iBookStore" />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/cart" element={<Cart />} />
        </Routes>
      </BrowserRouter>
    </CartProvider>
  );
}

export default App;
```

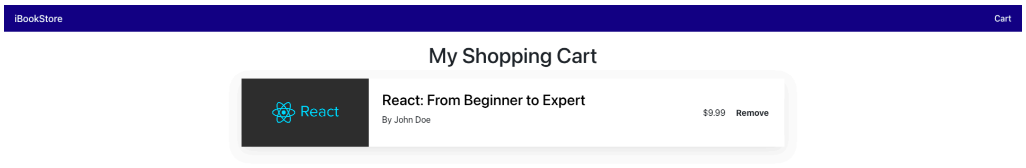
Now, to access the `cartState` data, we should call the `useContext` hook in the component. Let's udpate our `Cart` component as follows:

`Cart.js`

```
import React from "react";
import CartItem from "../../components/CartItem.js";
import classes from "./Cart.module.css";
import CartContext from "../../store/cart-context.js";
// Import the useContext hook
import { useContext } from "react";

function Cart() {
  // Import cartState using the useContext hook
  const cartState = useContext(CartContext);

  // Create cartList with data in cartState
  const cartList = cartState.map((item) => (
    <CartItem
      key={item.id}
      id={item.id}
      name={item.name}
      author={item.author}
      price={item.price}
      image={item.image}
    />
  ));

  return (
    <div className={classes.cart}>
```

```
      <h1>My Shopping Cart</h1>
      <ul className={classes.cartList}>{cartList}</ul>
    </div>
  );
}

export default Cart;
```

We use the `useContext` hook to import the value stored in `CartContext`. In this example, we get a list of products, which is exactly the `cartState` we just defined. Now, your cart page should show the sample product data on the cart page `http://localhost:3000/cart`.



My Shopping Cart

React: From Beginner to Expert

By John Doe

$9.99

Currently, our shopping cart is still not working since the data in cart context is static. We cannot make any changes to it. Remember when we define the cart context, we use `useState` to generate a update function for cart state. We should allow other component to use it as well. Update our `cart-context.js` like this:

`cart-context.js`

```
import { createContext, useState } from "react";

// Create the cart context
const CartContext = createContext();

// Define initial state for cart items, which should be an empty list
const initialCartState = [];

// Define CartProvider component
export function CartProvider(props) {
  // Apply useState hook to generate a update function
  const [cartState, setCartState] = useState(initialCartState);

  // Define a addCartItem function which can add an item to the cart list
  const addCartItem = (item) => {
    // Use concat method to add a new element to the list
    setCartState((prevCart) => prevCart.concat(item));
  };

  // Define a cartContext object to include the state and the addCartItem function
  const cartContext = {
    cartState: cartState,
    addCartItem: addCartItem,
  };

  return (
    <CartContext.Provider value={cartContext}>
      {props.children}
    </CartContext.Provider>
  );
}

export default CartContext;
```

There add three changes to the program:

1. We define a `addCartItem` that uses `setCartState` function to add a product item to the cart
2. Next, we create a object called `cartContext` containing both `cartState` and `addCartItem` function
3. Finally, we pass `cartContext` as the value of the context to the provider

Now, we have to modify how we use the context value in `Cart.js`:

`Cart.js`

```
const { cartState } = useContext(CartContext);
```

This will only import the value of `cartState` inside the `cartContext` object value.

Let's move on to apply the `addCartItem` function we just defined. Remember that for each of our product item, we have a "Add to cart" button. We can now make use of them. Let's change the code in `ProductItem.js`:

`ProductItem.js`

```
import React from "react";
import classes from "./ProductItem.module.css";
import CartContext from "../store/cart-context";
import { useContext } from "react";

function ProductItem(props) {
  // Import the addCartItem function from CartContext
  const { addCartItem } = useContext(CartContext);
  // Define a handler function that add the current item to the cart
  const addCartItemHandler = () => {
    addCartItem({
      id: props.id,
      name: props.name,
      price: props.price,
      image: props.image,
      author: props.author,
    });
  };

  // Assign the handler function to the Add to cart button
  return (
    <div className={classes.productItem}>
      <img src={props.image} alt="" />
      <div className={classes.productInfo}>
        <h1>{props.name}</h1>
        <div className={classes.infoBottom}>
          <div className={classes.infoLeft}>
            <span className={classes.author}>{props.author}</span>
            <span className={classes.price}>{props.price}</span>
          </div>
          <button onClick={addCartItemHandler}>Add to cart</button>
        </div>
      </div>
    </div>
  );
}

export default ProductItem;
```

For each `ProductItem` shown on our home page, its "Add to cart" button will call a `addCartItemHandler` function that will add the product information to our global cart state through the `addCartItem`. You can click on these button and check the results on the cart page.

Exercise

We have implemented a very simple shopping cart which is still incomplete. You should have the knowledge to add the following features on your own. Don't be afraid to give it a try and check your answer with the final codes in our project repository.ß

1. The "Remove" button in the `CartItem` componenet is not working. Try to add a `removeCartItem` in `cart-context` and import it in the `CartItem`. The function should receive an `id` argument and remove the corresponding product with that id from the cart state.
2. The `addCartItem` function in `cart-context` will keep adding products that already exists in the cart. Try to modify the function to avoid duplication.

   Tip: You may use the `filter()` method to manipulate the cart list.

The complete code of Tutorial 3: https://github.com/lvyin1122/ecommerce-tutorial-code/tree/03-building-a-cart-using-context