

# **Code Café React App**

**Dr. Atef Bader**

# Code Café React App

The screenshot shows a web browser window with the address bar displaying `https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch14.xhtml`. The page title is "14. Local Storage and useRef | Re X". The O'Reilly logo and navigation links are visible. The main content area shows the chapter title "Chapter 14. Local Storage and useRef" and introductory text. A search bar at the bottom contains the text "localStorage" and shows "1 of 5 matches". A blue callout box on the right contains two bullet points. A red arrow points from the first bullet point to the text "to persist" in the main content.

File Edit View History Bookmarks Tools Help

14. Local Storage and useRef | Re X

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch14.xhtml 67%

O'REILLY Topics Start Learning Search 50,000+ courses, events, titles, and more

## Chapter 14. Local Storage and useRef

In the last chapter, you built a form for users to input their information and submit it. You also used controlled components to manage the state of each form element.

In this chapter, you will use local storage to persist the values in the cart in case the user closes or refreshes the browser. You will also use the form you created to learn about another React hook: `useRef`.

The `useRef` hook creates and stores a mutable JavaScript object that persists through re-renders. This means you can use `useRef` to store values that are not intended to trigger a re-render of the component, such as a reference to a timer. A more common use case for this hook is storing a reference to a React element, giving you direct access to the DOM node that the element creates. This chapter will show you examples of both.

`useRef` is a specialized hook, so most production applications use it only in a couple of cases:

- use local storage to persist the values in the cart in case the user closes or refreshes the browser
- useRef Hook

13. Forms 14. Local Storage and useRef 15. Submitting Orders

5h 29m remaining

localStorage ^ v Highlight All Match Case Match Diacritics Whole Words 1 of 5 matches

# Code Café React App

The screenshot shows the Visual Studio Code editor with the file `App.js` open. The Explorer sidebar on the left shows the project structure, including the `src` directory and the `components` folder. The `App.js` file is selected in the Explorer. The main editor area displays the code for `App.js`, which includes imports for `axios`, `useEffect`, `useReducer`, `useState`, `BrowserRouter`, `Routes`, `Route`, `DetailItem`, `Cart`, `Details`, `Header`, `Home`, `NotFound`, `cartReducer`, `CartTypes`, and `initialCartState`. The code defines a function `App()` that uses `useState` to manage `items` and `useReducer` to manage the `cart`. It also includes a `useEffect` hook that updates the `localStorage` with the current `cart` state. Two red arrows point from text boxes to specific lines of code: one points to `localStorage.getItem(storageKey)` in the `try` block, and the other points to `localStorage.setItem(storageKey, JSON.stringify(cart))` in the `useEffect` block.

```
1 import axios from 'axios';
2 import { useEffect, useReducer, useState } from 'react';
3 import {
4   BrowserRouter as Router,
5   Routes,
6   Route,
7 } from 'react-router-dom';
8 import DetailItem from './components/DetailItem';
9 import Cart from './components/Cart';
10 import Details from './components/Details';
11 import Header from './components/Header';
12 import Home from './components/Home';
13 import NotFound from './components/NotFound';
14 import { cartReducer, CartTypes, initialCartState } from './reducers/cart';
15
16 const storageKey = 'cart';
17
18 function App() {
19   const [items, setItems] = useState([]);
20   const [cart, dispatch] = useReducer(
21     cartReducer,
22     initialCartState,
23     (initialState) => {
24       try {
25         const storedCart = JSON.parse(localStorage.getItem(storageKey));
26         return storedCart || initialState;
27       } catch (error) {
28         console.log('Error parsing cart', error);
29         return initialState;
30       }
31     }
32   );
33   const addToCart = (itemId) => dispatch({ type: CartTypes.ADD, itemId });
34
35   useEffect(() => {
36     localStorage.setItem(storageKey, JSON.stringify(cart));
37   }, [cart]);
38 }
```

useReducer that fetches the value of cart from local storage using `localStorage.getItem`

use the setter `localStorage.setItem` to store the cart value

# Code Café React App

File Edit Selection View Go Run Terminal Help

Cart.js - react-book - Visual Studio Code

EXPLORER

OPEN EDITORS

JS App.js 14-local-storage-and-useRef\src

JS Cart.js 14-local-storage-and-useRef\src\components

REACT-BOOK

14-local-storage-and-useRef

node\_modules

public

src

components

Cart.css

JS Cart.js

JS CartRow.js

DetailItem.css

JS DetailItem.js

Details.css

JS Details.js

Header.css

JS Header.js

Home.css

JS Home.js

NotFound.css

JS NotFound.js

Thumbnail.css

JS Thumbnail.js

images

items

reducers

types

JS App.js

JS App.test.js

index.css

JS index.js

logo.svg

OUTLINE

TIMELINE

14-local-storage-and-useRef > src > components > JS Cart.js > ...

```
41 if (digits.length === 6 && newNumber[7] === '-') {
42   formatted = `${formatted}-`;
43 } else if (digits.length > 6) {
44   formatted = `${formatted}-${digits.substring(6, 10)}';
45 }
46
47 if (digits.length === 10) {
48   zipRef.current.focus();
49 }
50 setPhone(formatted);
51 };
52
53 const onNameChange = (newName) => {
54   setName(newName);
55   if (debounceRef.current) {
56     clearTimeout(debounceRef.current);
57   }
58   debounceRef.current = setTimeout(() => {
59     axios
60       .get(`/api/employees/isEmployeeOfTheMonth?name=${newName}`)
61       .then((response) => setIsEmployeeOfTheMonth(
62         response?.data?.isEmployeeOfTheMonth,
63       ))
64       .catch(console.error);
65   }, 300);
66 };
67
68 return (
69   <div className="cart-component">
70     <h2>Your Cart</h2>
71     {cart.length === 0 ? (
72       <div>Your cart is empty.</div>
73     ) : (
74       <>
75         <table>
76           <thead>
77             <tr>
78               <th>Quantity</th>
79               <th>Item</th>
```

userref Aa ab\_\* ? of 3 ↑ ↓ ≡ ×

Using setTimeout with useRef

Check for Employee of the month

Loren and Ashley are the current employees of the month for Code Café

Ln 3, Col 16 (6 selected) Spaces: 2 UTF-8 LF {} JavaScript

# Code Café React App

code-café-backend > routes > JS employees.js > ...

```
1 const { Router } = require('express');
2
3 const employeeRoutes = Router();
4
5 const currentEmployeesOfTheMonth = ['ashley', 'loren'];
6
7 employeeRoutes.get('/isEmployeeOfTheMonth', (req, res) => {
8   const name = req?.query?.name?.toLowerCase() || '';
9   const result = currentEmployeesOfTheMonth.includes(name);
10  res.json({ isEmployeeOfTheMonth: result });
11 });
12
13 module.exports = employeeRoutes;
```

Code-café-backend server has:

Loren and Ashley as the current employees of the month for Code Café

# Code Café React App

- Startup your backend server on port 3030

[illegible]

# Code Café React App

Select Windows PowerShell

Note that the development build is not optimized.  
To create a production build, use `npm run build`.


webpack compiled **successfully**

- Startup React server on  
port 3000




# Code Café React App

Code Café x +

localhost:3000/cart

 **Code Café**

## Your Cart

Quantity	Item	Price
1	French Press	\$1.75 
1	Cookie	\$0.50 
1	Croissant	\$2.50 

Subtotal: \$0.00  
Tax: \$0.00  
Total: \$0.00

## Checkout

**Name**

**Phone Number**

**ZIP Code**

**Order Now**

- Loren entered in Name
- Free Orders for Employee of the month
- Total is 0



# Code Café React App

File Edit View History Bookmarks Tools Help

15. Submitting Orders | React Pro X

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch15.xhtml 67%



O'REILLY Topics Start Learning Search 50,000+ courses, events, titles, and more

## Chapter 15. Submitting Orders

Although Code Café's users can build an order, they cannot actually send it to the café.

By the end of this chapter, you will store orders on the server, ready for the café staff to fulfill (**Figure 15.1, "Orders submitted to the server"**).

Figure 15.1. Orders submitted to the server

**Code Café**

### Your Cart

Quantity	Item	Price
1	Sandwich	\$6.00
1	Tea	\$1.50
Subtotal:		\$7.50
Tax:		\$0.30
Total:		\$7.80

### Checkout

Network

Filter: ☐ Preserve log ☐ Disable cache No throttling

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

20 ms 40 ms 60 ms 80 ms 100 ms

Name: **orders**

General

Request URL: http://localhost:3000/api/orders

Request Method: POST

Status Code: 201 Created

Remote Address: 127.0.0.1:3000

Referer Policy: strict-origin-when-cross-origin

14. Local Storage and useRef 5h 29m remaining 15. Submitting Orders 16. Component Composition

localStorage Highlight All Match Case Match Diacritics Whole Words 1 of 5 matches

We need to store orders on the backend server

# Code Café React App

File Edit Selection View Go Run Terminal Help

Cart.js - react-book - Visual Studio Code

EXPLORER

JS App.js 14-local-storage-and-useRef/src

JS Cart.js 15-submitting-orders/src/components

REACT-BOOK

14-local-storage-and-useRef

15-submitting-orders

node\_modules

public

src

components

Cart.css

JS Cart.js

JS CartRow.js

DetailItem.css

JS DetailItem.js

Details.css

JS Details.js

Header.css

JS Header.js

Home.css

JS Home.js

NotFound.css

JS NotFound.js

Thumbnail.css

JS Thumbnail.js

images

items

reducers

types

JS App.js

JS App.test.js

index.css

JS index.js

OUTLINE

TIMELINE

15-submitting-orders > src > components > JS Cart.js > ...

```
17 const subTotal = isEmployeeOfTheMonth ? 0 : cart.reduce((acc, item)
18   const detailItem = items.find((i) => i.itemId === item.itemId);
19   const itemPrice = detailItem.salePrice ?? detailItem.price;
20   return item.quantity * itemPrice + acc;
21 }, 0);
22
23 const taxPercentage = parseInt(zipCode.substring(0, 1) || '0', 10) + 1;
24 const taxRate = taxPercentage / 100;
25 const tax = subTotal * taxRate;
26 const total = subTotal + tax;
27 const isValid = zipCode.length === 5 && name.trim();
28
29 const submitOrder = async (event) => {
30   event.preventDefault();
31   setIsSubmitting(true);
32   try {
33     await axios.post('/api/orders', {
34       items: cart,
35       name,
36       phone,
37       zipCode,
38     });
39     console.log('Order Submitted');
40   } catch (error) {
41     console.error('Error submitting the order', error);
42   } finally {
43     setIsSubmitting(false);
44   }
45 };
46
47 const setFormattedPhone = (newNumber) => {
48   const digits = newNumber.replace(/\D/g, '');
49   let formatted = digits.substring(0, 3);
50   if (digits.length === 3 && newNumber[3] === '-') {
51     formatted = `${formatted}-`;
52   } else if (digits.length > 3) {
53     formatted = `${formatted}-${digits.substring(3, 6)}`;
54   }
55   if (digits.length === 6 && newNumber[7] === '-') {
```

Use Axios to make a POST request to the API that includes the order items and the user details.

Ln 1, Col 1 Spaces: 2 UTF-8 LF {} JavaScript

# Code Café React App

- Startup your backend server on port 3030

```
> npm start
> node app.js

Listening on http://localhost:3030
Accessing /api/items
Accessing /api/items
Accessing /
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /main.69b8defce96165e7e728.hot-update.json
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
Accessing /api/items
```

# Code Café React App

Select Windows PowerShell

Note that the development build is not optimized.  
To create a production build, use `npm run build`.

webpack compiled **successfully**

- Startup React server on  
port 3000

# Code Café React App

The screenshot displays the Code Café React App interface. The top navigation bar includes the Code Café logo and a shopping cart icon with a '3' badge. The main content area is divided into two sections: 'Your Cart' and 'Checkout'.

**Your Cart**

Quantity	Item	Price
1	French Press	\$1.75
1	Cookie	\$0.50
1	Croissant	\$2.50

Subtotal: \$4.75  
Tax: \$0.33  
Total: \$5.08

**Checkout**

**Name**

**Phone Number**

**ZIP Code**

**Order Now**

**3. Order Submitted**

**1. Fill out the form**

**2. Click Order Now**

The right sidebar shows the browser's developer console with the message 'Order Submitted' and the file path 'Cart.js:39'.

# Code Café React App

The screenshot shows a web browser window with the address bar displaying `localhost:3030/api/orders`. Below the address bar, a REST client interface shows the JSON response for the GET request. The response is a list of two orders, each with a customer object and an array of items.

```
{
  "0": {
    "id": "1",
    "name": "Sam SMith",
    "phone": "987-979-8797",
    "zipCode": "67670",
    "items": [
      {
        "0": {
          "itemId": "cookie",
          "quantity": 1
        },
        "1": {
          "itemId": "iced-coffee",
          "quantity": 1
        },
        "2": {
          "itemId": "tea",
          "quantity": 1
        }
      }
    ]
  },
  "1": {
    "id": "2",
    "name": "Ana Rodkin",
    "phone": "565-757-6575",
    "zipCode": "46343",
    "items": [
      {
        "0": {
          "itemId": "iced-coffee",
          "quantity": 4
        },
        "1": {
          "itemId": "tea",
          "quantity": 2
        }
      }
    ]
  }
}
```

A red arrow points from the text box on the right to the URL bar.

You check the list of orders  
the backend server has

# Code Café React App

The screenshot shows a web browser window with the address bar displaying `https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch16.xht`. The page title is "16. Component Composition | ReX". The O'Reilly logo and navigation links are visible. The main heading is "Chapter 16. Component Composition". The text describes component composition and lists two scenarios for using the alert component. A red box on the right contains the text: "When the order submission is successful, display a thank-you message." A red arrow points from this box to the first bullet point in the list.

File Edit View History Bookmarks Tools Help

16. Component Composition | ReX

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch16.xht 67%

O'REILLY Topics Start Learning Search 50,000+ courses, events, titles, and more

## Chapter 16. Component Composition

In this chapter, you will use *component composition* to build a reusable alert component. Component composition allows you to pass child components as props to another component, giving you flexible control over what renders. Component composition is a dynamic, powerful React tool that developers frequently use in component libraries.

You will use the alert component to display messages in two scenarios:

- When the order submission is successful, you will display a thank-you message.
- When the order submission fails, you will display an error message.

By the end of this chapter, your cart page will be complete ([Figure 16.1, "Completed cart page"](#)).

Figure 16.1. Completed cart page

15. Submitting Orders 16. Component Composition 17. Context

5h 29m remaining

localStorage ^ v Highlight All Match Case Match Diacritics Whole Words 1 of 5 matches

# Code Café React App

Alert.js - react-book - Visual Studio Code

EXPLORER

- JS App.js
- JS Cart.js
- JS Alert.js

16-component-composition > src > components > JS Alert.js > Alert > constructor

```
1 import PropTypes from 'prop-types';
2 import './Alert.css';
3
4 const BACKGROUND_COLORS = {
5   success: '#adc6a8',
6   error: '#f5c6cb',
7 };
8
9 function Alert({
10   children,
11   visible,
12   type,
13 }) {
14   return (
15     <div
16       className={`alert-component ${visible && 'visible'}`}
17       role="alert"
18       hidden={!visible}
19       style={{ backgroundColor: BACKGROUND_COLORS[type] }}
20     >
21       {children}
22     </div>
23   );
24 }
25
26 Alert.propTypes = {
27   children: PropTypes.node.isRequired,
28   visible: PropTypes.bool.isRequired,
29   type: PropTypes.oneOf(['success', 'error']).isRequired,
30 };
31
32 export default Alert;
33
```

- Create Alert Component

Ln 12, Col 8 Spaces: 2 UTF-8 LF {} JavaScript



# Code Café React App

File Edit Selection View Go Run Terminal Help

Cart.js - react-book - Visual Studio Code

EXPLORER

- JS App.js
- JS Cart.js
- JS Alert.js

OPEN EDITORS

- JS App.js 14-local-storage-and-useRef/src
- JS Cart.js 16-component-composition/src/components
- JS Alert.js 16-component-composition/src/components

REACT-BOOK

- 14-local-storage-and-useRef
- 15-submitting-orders
- 16-component-composition
  - node\_modules
  - public
  - src
    - components
      - Alert.css
      - Alert.js
      - Cart.css
      - Cart.js
      - CartRow.js
      - DetailItem.css
      - DetailItem.js
      - Details.css
      - Details.js
      - Header.css
      - Header.js
      - Home.css
      - Home.js
      - NotFound.css
      - NotFound.js
      - Thumbnail.css
      - Thumbnail.js
    - images
    - items
    - reducers
    - types
- OUTLINE
- TIMELINE

```
71 };
72
73 const onChange = (newName) => {
74   setName(newName);
75   if (debounceRef.current) {
76     clearTimeout(debounceRef.current);
77   }
78   debounceRef.current = setTimeout(() => {
79     axios
80       .get(`/api/employees/isEmployeeOfTheMonth?name=${newName}`)
81       .then((response) => setIsEmployeeOfTheMonth(
82         response.data?.isEmployeeOfTheMonth,
83       ))
84       .catch(console.error);
85   }, 300);
86 };
87
88 return (
89   <div className="cart-component">
90     <Alert visible={showSuccessAlert} type="success">
91       Thank you for your order.
92     </Alert>
93     <Alert visible={!!apiError} type="error">
94       <p>There was an error submitting your order.</p>
95       <p>{apiError}</p>
96       <p>Please try again.</p>
97     </Alert>
98     <h2>Your Cart</h2>
99     {cart.length === 0 ? (
100       <div>Your cart is empty.</div>
101     ) : (
102       <>
103         <table>
104           <thead>
105             <tr>
106               <th>Quantity</th>
107               <th>Item</th>
108               <th>Price</th>
109             </tr>
```

- Add Alert Component to Cart Component

Ln 81, Col 53 Spaces: 2 UTF-8 LF {} JavaScript

# Code Café React App

- Startup your backend server on port 3030

```
> npm start  
 > node app.js
```

Listening on http://localhost:3030

Accessing /api/items  
Accessing /api/items  
Accessing /  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /main.69b8defce96165e7e728.hot-update.json  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items  
Accessing /api/items

# Code Café React App

Select Windows PowerShell

Note that the development build is not optimized.  
To create a production build, use `npm run build`.

webpack compiled **successfully**

- Startup React server on  
port 3000

# Code Café React App

The screenshot shows a web browser window with the URL `localhost:3000/cart`. The page title is "Code Café". The main content area is titled "Your Cart" and displays a table of items in the cart. The table has three columns: "Quantity", "Item", and "Price". There are four items: French Press (\$1.75), Cookie (\$0.50), Croissant (\$2.50), and Tea (\$1.50). Each item has a small "X" icon next to its price. Below the table, the subtotal is \$6.25, tax is \$0.63, and the total is \$6.88. A red arrow points from the "Checkout" section to the "Order Now" button. The "Checkout" section contains three input fields: "Name" (filled with "Sam Smith"), "Phone Number" (filled with "987-987-9979"), and "ZIP Code" (filled with "98797"). A blue box with a red border contains the instructions: "Fill out the form" and "Click Order Now".

**Your Cart**

Quantity	Item	Price
1	French Press	\$1.75
1	Cookie	\$0.50
1	Croissant	\$2.50
1	Tea	\$1.50

Subtotal: \$6.25  
Tax: \$0.63  
Total: \$6.88

**Checkout**

**Name**


**Phone Number**

**ZIP Code**

**Order Now**

- Fill out the form
- Click Order Now

# Code Café React App



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/cart'. The page header includes the 'Code Café' logo and a shopping cart icon with a '0' badge. A green notification bar at the top contains the text 'Thank you for your order.' A red arrow points from this message to a blue box on the right. Below the notification bar, the section 'Your Cart' is displayed with the text 'Your cart is empty.'

Thank you for your order.

Your Cart

Your cart is empty.

- Alert message printed of successful order submission

# References

- Rosson, M. B., John M. 2001. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction* (1st Edition). San Francisco, CA: Morgan Kaufmann Publishers. [ISBN: 978-1558607125]
  - Tidwell, J., Brewer, C., Valencia, A. 2020. *Designing Interfaces* (3rd Edition). Sebastopol, CA: O'Reilly Media. [ISBN: 978-1492051961]
  - Klingman, L. 2023. *React Programming: The Big Nerd Ranch Guide*. Indianapolis, IN: Pearson Technology Group. [ISBN: 978-0137901692]
  - De Voil, N. 2016. *User Experience Foundations*. Swindon, UK: BCS Learning & Development. [ISBN: 978-1780173498]
  - Nielsen, J. 1994. *Usability Engineering*. Cambridge, MA: Morgan Kaufmann. [ISBN: 978-0125184069]
  - Ritter, M., Winterbottom, C. 2017. *UX for the Web*. Birmingham, UK: Packt Publishing. [ISBN: 978-1787128477]
  - Nielsen, J., Loranger, H. 2006. *Prioritizing Web Usability*. Berkeley, CA: New Riders. [ISBN: 978-0321350312]
  - Yablonski, J. 2020. *Laws of UX*. Sebastopol, CA: O'Reilly Media. [ISBN: 978-1492055310]
  - MacDonald, D. 2019. *Practical UI Patterns for Design Systems: Fast-Track Interaction Design for a Seamless User Experience*. New York, NY: Springer Science. [ISBN: 978-1484249376]
  - Timms, S. 2016. *Mastering JavaScript Design Patterns (2<sup>nd</sup> Edition)*. Birmingham, UK: Packt Publishing. [ISBN: 978-1785882166]
  - Larsen, J. 2016. *React Hooks in Action*. Shelter Island, NY: Manning Publications. [ISBN: 978-1617297632]
  - Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., Diakopoulos, N. 2016. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (6<sup>th</sup> Edition)*. Essex, England: Pearson Education Publications. [ISBN: 978-1292153919]
  - Flanagan, D. 2020. *JavaScript: The Definitive Guide (7<sup>th</sup> Edition)*. Sebastopol, CA: O'Reilly Media. [ISBN: 978-1491952023]
  - Leventhal, Li., Julie Barnes, J. 2007. *Usability Engineering: Process, Products & Examples* (1st Edition). Essex, England: Pearson Education Publications. [ISBN: 978-0131570085]
-