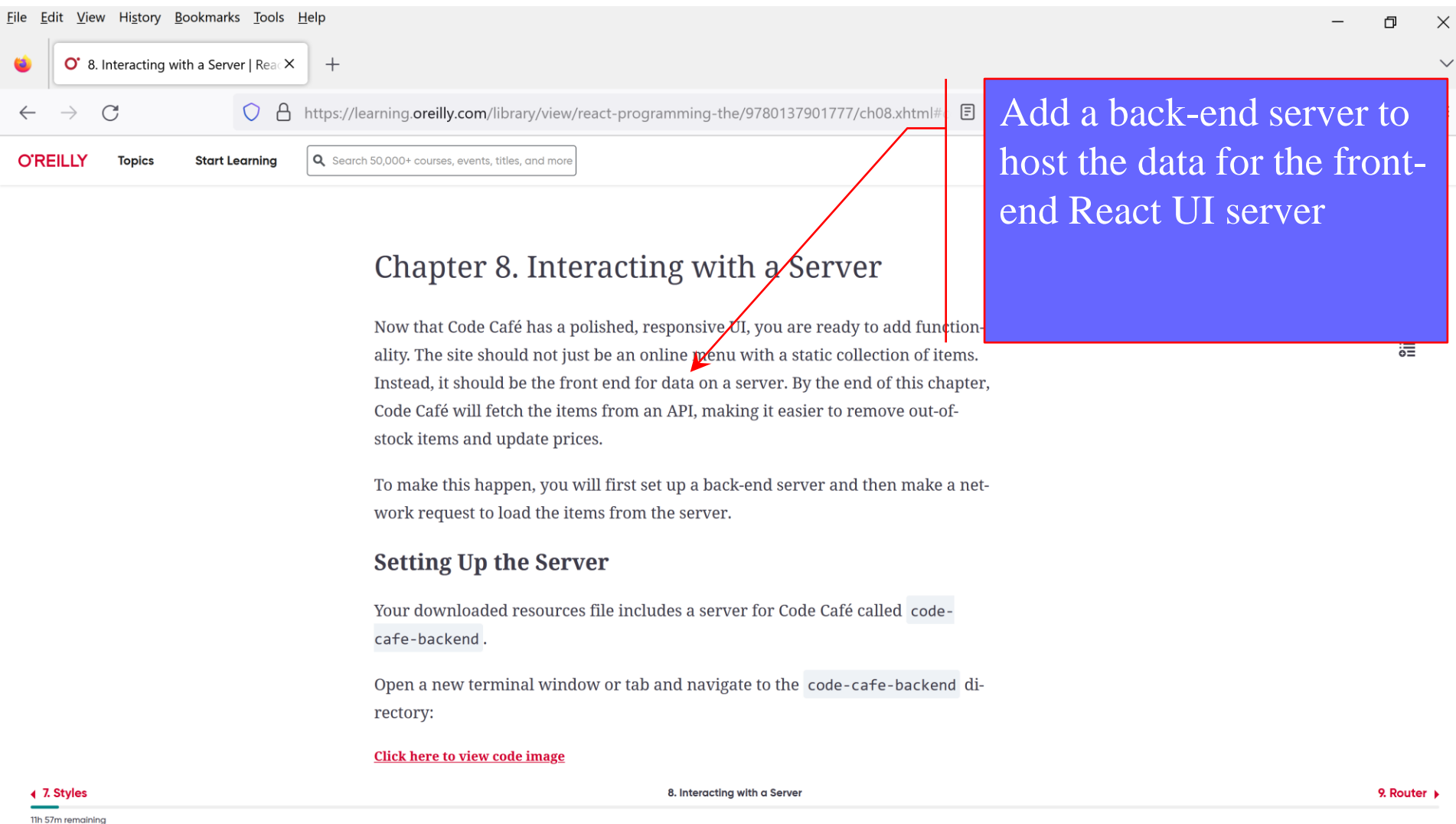


React

Code Café React App

Dr. Atef Bader

Code Café React App



File Edit View History Bookmarks Tools Help

8. Interacting with a Server | React

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch08.xhtml#

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

Chapter 8. Interacting with a Server

Now that Code Café has a polished, responsive UI, you are ready to add functionality. The site should not just be an online menu with a static collection of items. Instead, it should be the front end for data on a server. By the end of this chapter, Code Café will fetch the items from an API, making it easier to remove out-of-stock items and update prices.

To make this happen, you will first set up a back-end server and then make a network request to load the items from the server.

Setting Up the Server

Your downloaded resources file includes a server for Code Café called `code-cafe-backend`.

Open a new terminal window or tab and navigate to the `code-cafe-backend` directory:

[Click here to view code image](#)

7. Styles 8. Interacting with a Server 9. Router

11h 57m remaining

Code Café React App

File Edit View History Bookmarks Tools Help

8. Interacting with a Server | React

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch08.xhtml# 67%

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

Chapter 8. Interacting with a Server

Now that Code Café has a polished, responsive UI, you are ready to add functionality. The site should not just be an online menu with a static collection of items. Instead, it should be the front end for data on a server. By the end of this chapter, Code Café will fetch the items from an API, making it easier to remove out-of-stock items and update prices.

To make this happen, you will first set up a back-end server and then make a network request to load the items from the server.

Setting Up the Server

Your downloaded resources file includes a server for Code Café called `code-cafe-backend`.

Open a new terminal window or tab and navigate to the `code-cafe-backend` directory:

[Click here to view code image](#)

7. Styles 8. Interacting with a Server 9. Router

11h 57m remaining

The back-end server will be in the *code-cafe-backend* directory

Code Café React App

File Edit View History Bookmarks Tools Help

8. Interacting with a Server | React

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch08.xhtml# 67%

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

What's New

Your downloaded resources file includes a server for Code Café called `code-cafe-backend`.

Open a new terminal window or tab and navigate to the `code-cafe-backend` directory:

[Click here to view code image](#)

```
cd YOUR_PATH/resources/code-cafe-backend
```

Next, install the dependencies required for the server to run:

```
npm install
```

Then start the server:

```
npm start
```

The console prints `Listening on http://localhost:3030` to let you know the server is up and running ([Figure 8.1, "The server is listening"](#)).

Figure 8.1. The server is listening

7. Styles 8. Interacting with a Server 9. Router

11h 57m remaining

Code Café React App

code-café-backend > JS app.js > [e] app

```
1  const express = require('express');
2  const http = require('http');
3  const bodyParser = require('body-parser');
4  const cookieParser = require('cookie-parser');
5
6  const config = require('./config');
7  const routes = require('./routes');
8  const websocketServer = require('./websocketServer');
9
10 const app = express();
11 const { port } = config;
12
13 app.use(bodyParser.json());
14 app.use(bodyParser.urlencoded({ extended: true }));
15 app.use(cookieParser());
16
17 // Basic logging middleware
18 app.use((req, res, next) => {
19   console.log(`Accessing ${req.path}`);
20   next();
21 });
22
23 app.get('/', (req, res) => {
24   res.json({ message: 'Hi!' });
25 });
26
27 app.use('/', routes);
28
29 // 4 params are required for the error handler even though we don't use them all
30 // eslint-disable-next-line no-unused-vars
31 app.use((err, req, res, next) => {
32   console.error(err);
33   res.status(500).send('Something broke!');
34 });
35
36 const server = http.createServer(app);
37
38 server.on('upgrade', (request, socket, head) => {
39   // Skip auth since we don't have the cookie and would have to reconfigure the proxy to get it
40   // (request, socket, head) => {
```

App.js has default route

Ln 10, Col 8 (1 selected) Spaces: 2 UTF-8 LF {} JavaScript

Code Café React App

code-café-backend > JS app.js > app

```
1  const express = require('express');
2  const http = require('http');
3  const bodyParser = require('body-parser');
4  const cookieParser = require('cookie-parser');
5
6  const config = require('./config');
7  const routes = require('./routes');
8  const WebSocketServer = require('./WebSocketServer');
9
10 const app = express();
11 const { port } = config;
12
13 app.use(bodyParser.json());
14 app.use(bodyParser.urlencoded({ extended: true }));
15 app.use(cookieParser());
16
17 // Basic logging middleware
18 app.use((req, res, next) => {
19   console.log(`Accessing ${req.path}`);
20   next();
21 });
22
23 app.get('/', (req, res) => {
24   res.json({ message: 'Hi!' });
25 });
26
27 app.use('/', routes);
28
29 // 4 params are required for the error handler even though we don't use them all
30 // eslint-disable-next-line no-unused-vars
31 app.use((err, req, res, next) => {
32   console.error(err);
33   res.status(500).send('Something broke!');
34 });
35
36 const server = http.createServer(app);
37
38 server.on('upgrade', (request, socket, head) => {
39   // Skip auth since we don't have the cookie and would have to reconfigure the proxy to get it
40   // (request, socket, head) => {
41     // ...
42   }
43 });
```

App.js uses accepts the routes defines

Code Café React App

App.js will accept
/api/items route

code-café-backend > routes > JS items.js > itemRoutes.get('/') callback

```
1  const { Router } = require('express');
2
3  const itemRoutes = Router();
4
5  itemRoutes.get('/', (req, res) => {
6    setTimeout(
7      () => res.json([
8        {
9          itemId: 'coffee',
10         imageId: 'coffee',
11         title: 'Coffee',
12         price: 0.99,
13         description: '',
14         salePrice: 0,
15       },
16       {
17         itemId: 'cookie',
18         imageId: 'cookie',
19         title: 'Cookie',
20         price: 1,
21         description: 'May contain nuts.',
22         salePrice: 0.50,
23       },
24       {
25         itemId: 'croissant',
26         imageId: 'croissant',
27         title: 'Croissant',
28         price: 2.50,
29       },
30       {
31         itemId: 'cupcake',
32         imageId: 'cupcake',
33         title: 'Cupcake',
34         price: 3,
35       },
36       {
37         itemId: 'french-press',
38         imageId: 'french-press',
39         title: 'French Press',
40         price: 1.75,
```

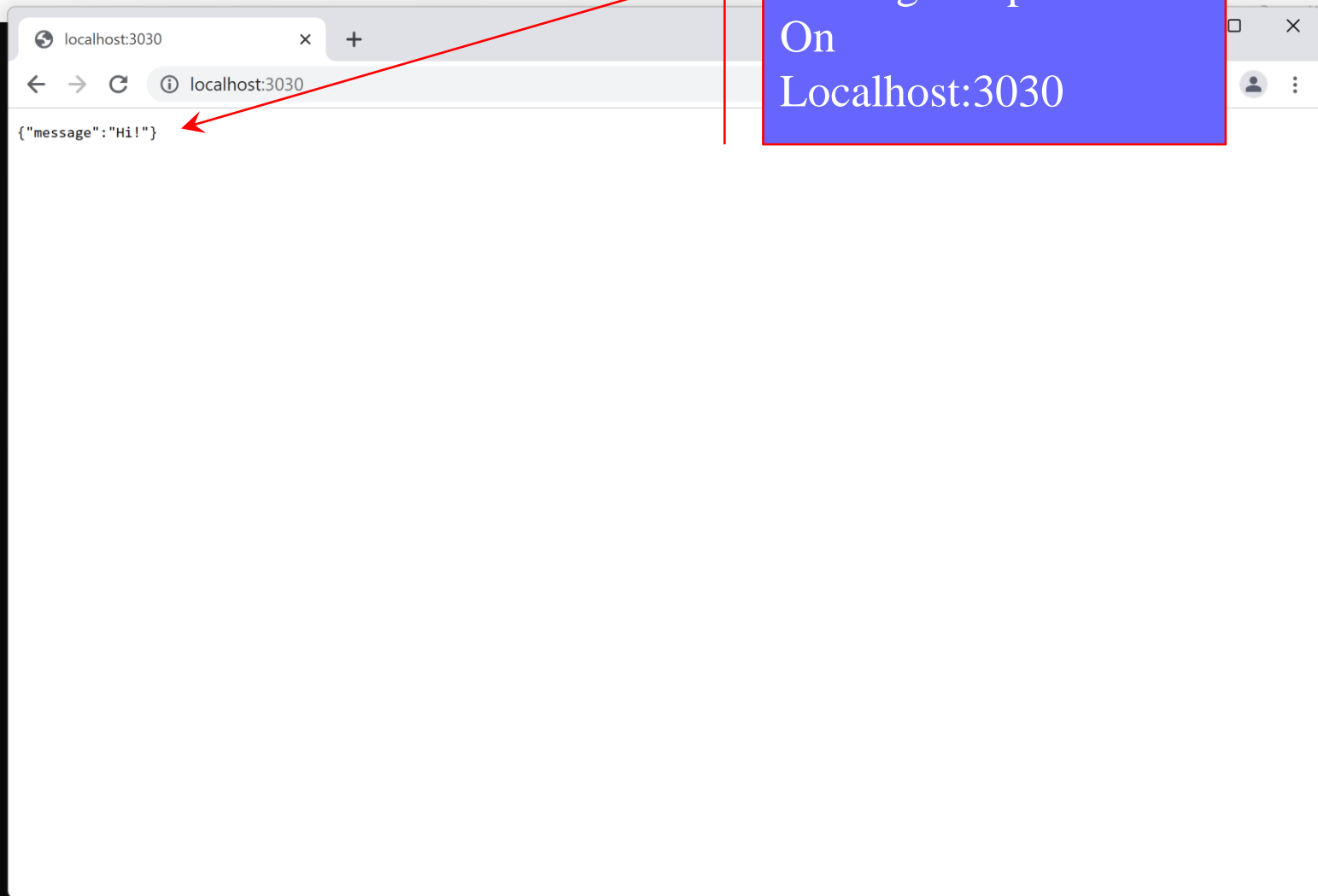
Code Café React App

```
npm start  
> node app.js  
Listening on http://localhost:3030
```

npm start

Code Café React App

```
npm start
> node app.js
Listening on http://localhost:3030
Accessing /
Accessing /favicon.ico
```



Message Hi printed
On
Localhost:3030

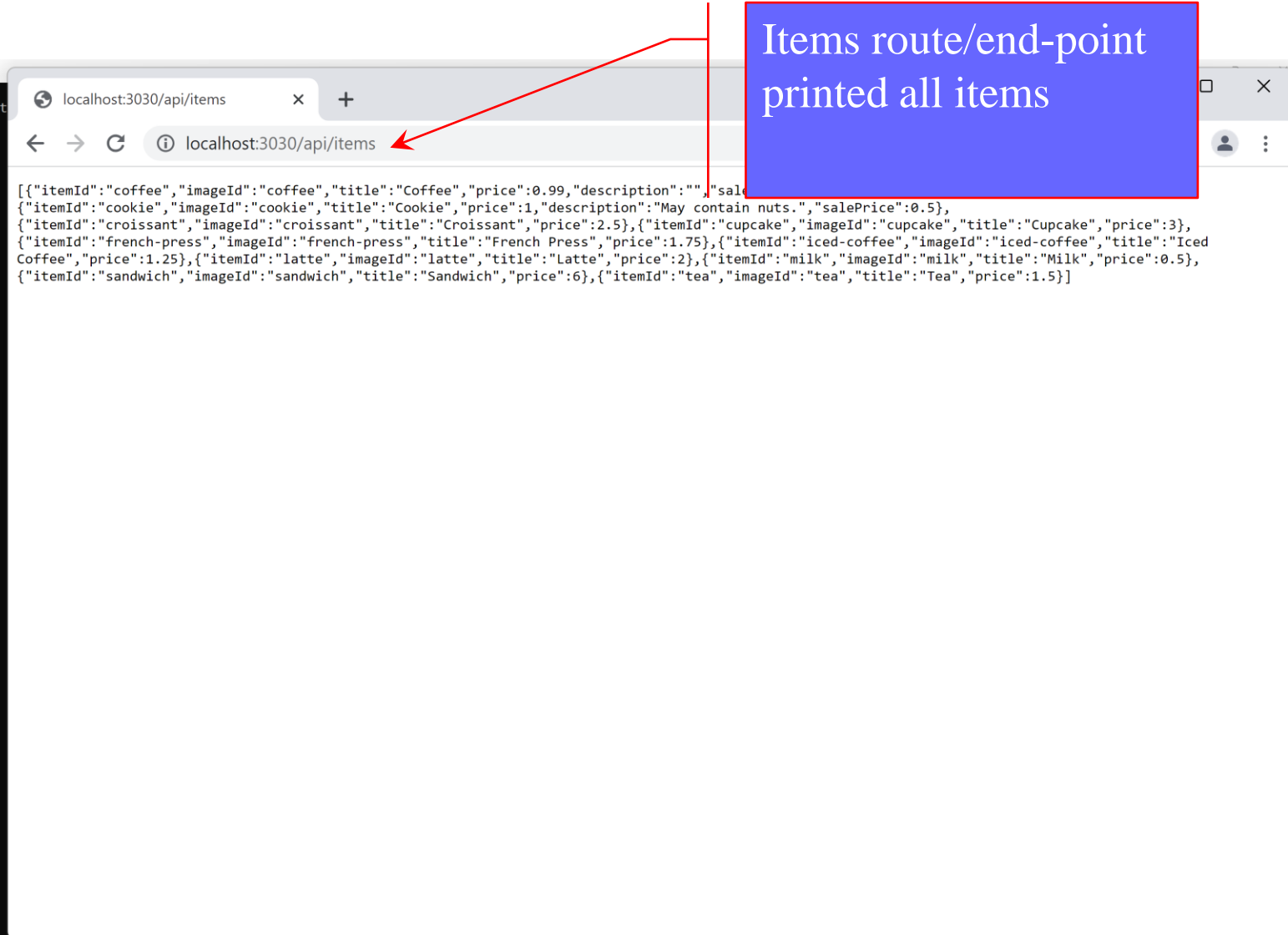
Code Café React App

npm start

C:\bader\Clark\MSCS 3025\Lectures\Lect

```
> rest-api@0.0.0 start  
> node app.js
```

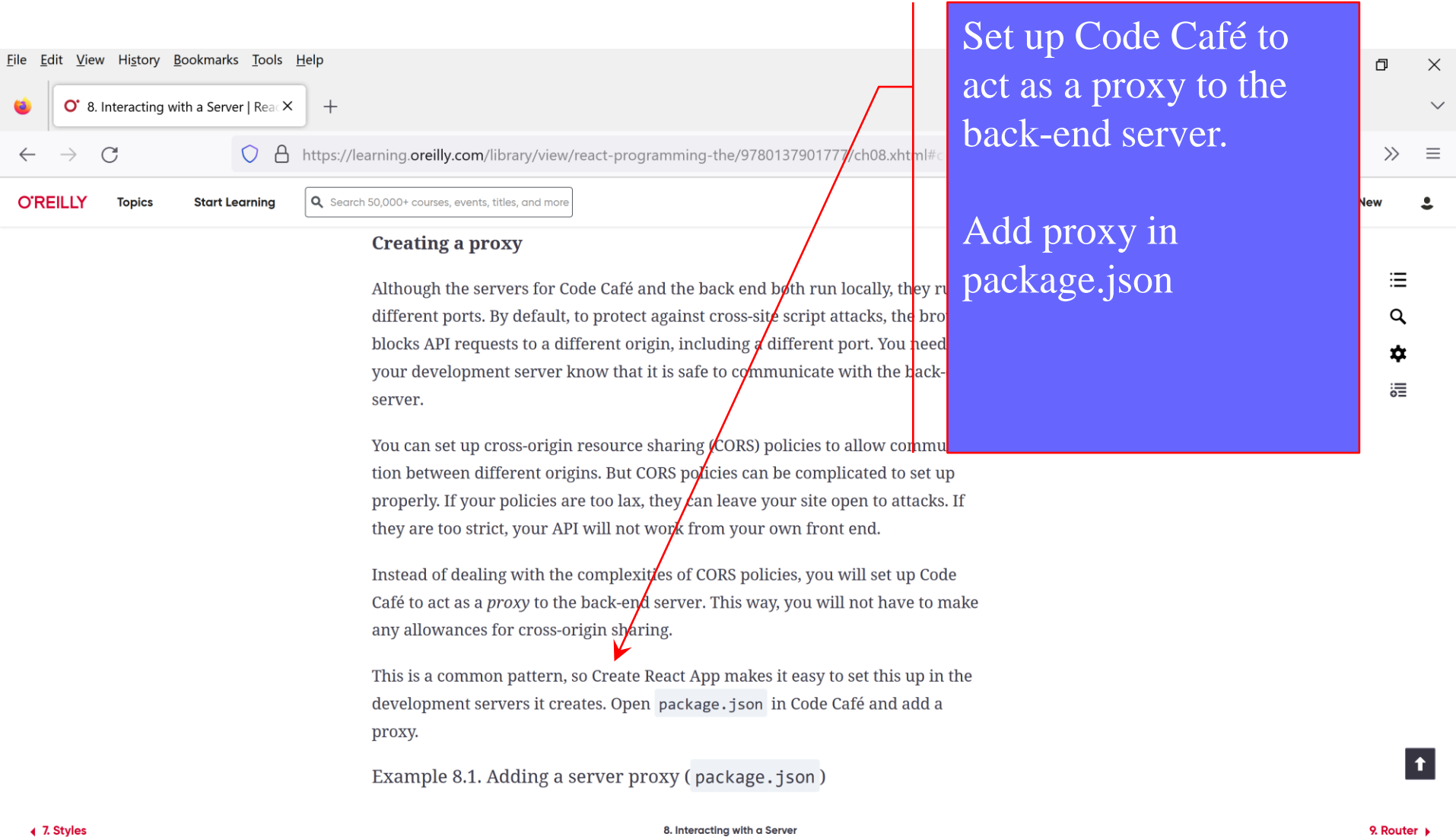
Listening on http://localhost:3030
Accessing /api/items



Items route/end-point printed all items

```
[{"itemId":"coffee","imageId":"coffee","title":"Coffee","price":0.99,"description":"","salePrice":0.99}, {"itemId":"cookie","imageId":"cookie","title":"Cookie","price":1,"description":"May contain nuts.","salePrice":0.5}, {"itemId":"croissant","imageId":"croissant","title":"Croissant","price":2.5}, {"itemId":"cupcake","imageId":"cupcake","title":"Cupcake","price":3}, {"itemId":"french-press","imageId":"french-press","title":"French Press","price":1.75}, {"itemId":"iced-coffee","imageId":"iced-coffee","title":"Iced Coffee","price":1.25}, {"itemId":"latte","imageId":"latte","title":"Latte","price":2}, {"itemId":"milk","imageId":"milk","title":"Milk","price":0.5}, {"itemId":"sandwich","imageId":"sandwich","title":"Sandwich","price":6}, {"itemId":"tea","imageId":"tea","title":"Tea","price":1.5}]
```

Code Café React App



File Edit View History Bookmarks Tools Help

8. Interacting with a Server | React

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch08.xhtml#

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

Creating a proxy

Although the servers for Code Café and the back end both run locally, they run on different ports. By default, to protect against cross-site script attacks, the browser blocks API requests to a different origin, including a different port. You need to let your development server know that it is safe to communicate with the back-end server.

You can set up cross-origin resource sharing (CORS) policies to allow communication between different origins. But CORS policies can be complicated to set up properly. If your policies are too lax, they can leave your site open to attacks. If they are too strict, your API will not work from your own front end.

Instead of dealing with the complexities of CORS policies, you will set up Code Café to act as a *proxy* to the back-end server. This way, you will not have to make any allowances for cross-origin sharing.

This is a common pattern, so Create React App makes it easy to set this up in the development servers it creates. Open `package.json` in Code Café and add a proxy.

Example 8.1. Adding a server proxy (`package.json`)

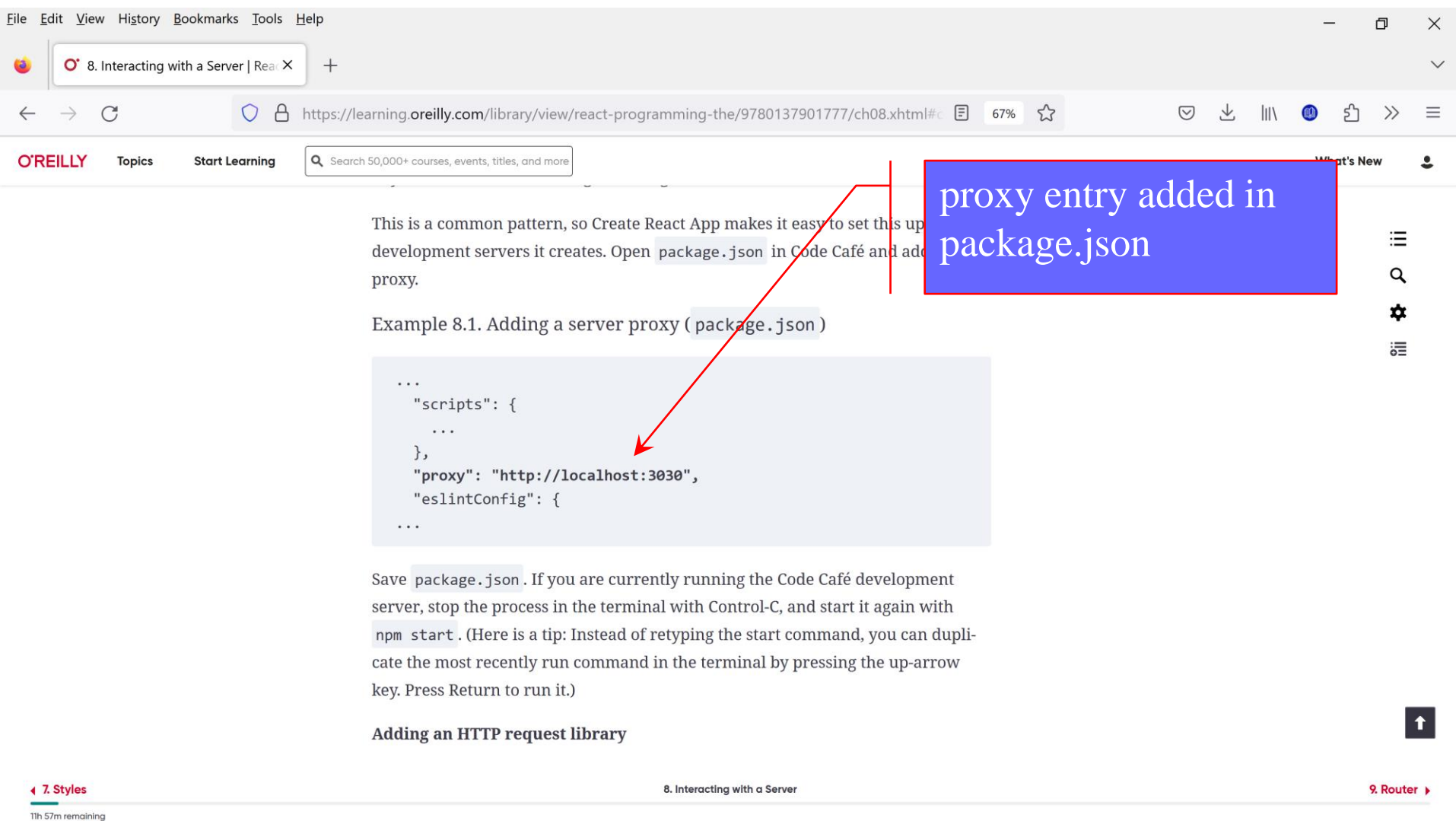
Set up Code Café to act as a proxy to the back-end server.

Add proxy in `package.json`

7. Styles 8. Interacting with a Server 9. Router

11h 57m remaining

Code Café React App



File Edit View History Bookmarks Tools Help

8. Interacting with a Server | React

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch08.xhtml# 67%

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

What's New

This is a common pattern, so Create React App makes it easy to set this up on development servers it creates. Open `package.json` in Code Café and add a proxy.

Example 8.1. Adding a server proxy (`package.json`)

```
...
"scripts": {
  ...
},
"proxy": "http://localhost:3030",
"eslintConfig": {
  ...
}
```

Save `package.json`. If you are currently running the Code Café development server, stop the process in the terminal with Control-C, and start it again with `npm start`. (Here is a tip: Instead of retyping the start command, you can duplicate the most recently run command in the terminal by pressing the up-arrow key. Press Return to run it.)

Adding an HTTP request library

7. Styles 8. Interacting with a Server 9. Router

11h 57m remaining

Code Café React App

Install Axios package to make HTTP requests.

File Edit View History Bookmarks Tools Help

8. Interacting with a Server | React

https://learning.oreilly.com/library/view/react-programming-the/9780137901777/ch08.xhtml#

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

npm start. (Here is a tip: instead of retyping the start command, you can use the up arrow key to cycle through the most recently run command in the terminal by pressing the up-arrow key. Press Return to run it.)

Adding an HTTP request library

There is one more step to setting up Code Café's back-end server: You need a way to make HTTP requests.

There are many tools you can use to make HTTP requests in React applications, including the browser's built-in Fetch API. React is not opinionated about this. We like the popular Axios library, because it is a lightweight package that provides a simple syntax for making network calls, working with JSON data, and handling errors.

Go ahead and install Axios. In a terminal separate from the two servers you have running, navigate to your code-cafe directory. Then install Axios by running

```
npm install --save axios@0.27.2
```

useEffect

7. Styles 8. Interacting with a Server 9. Router

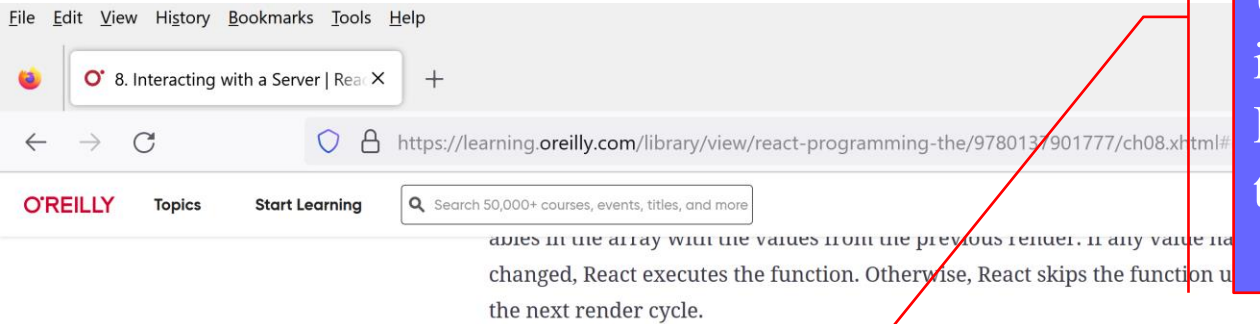
11h 57m remaining

axios

Highlight All Match Case Match Diacritics Whole Words 1 of 16 matches

Code Café React App

Use the `useEffect` hook in `App.js` to make an HTTP request to get the list of items

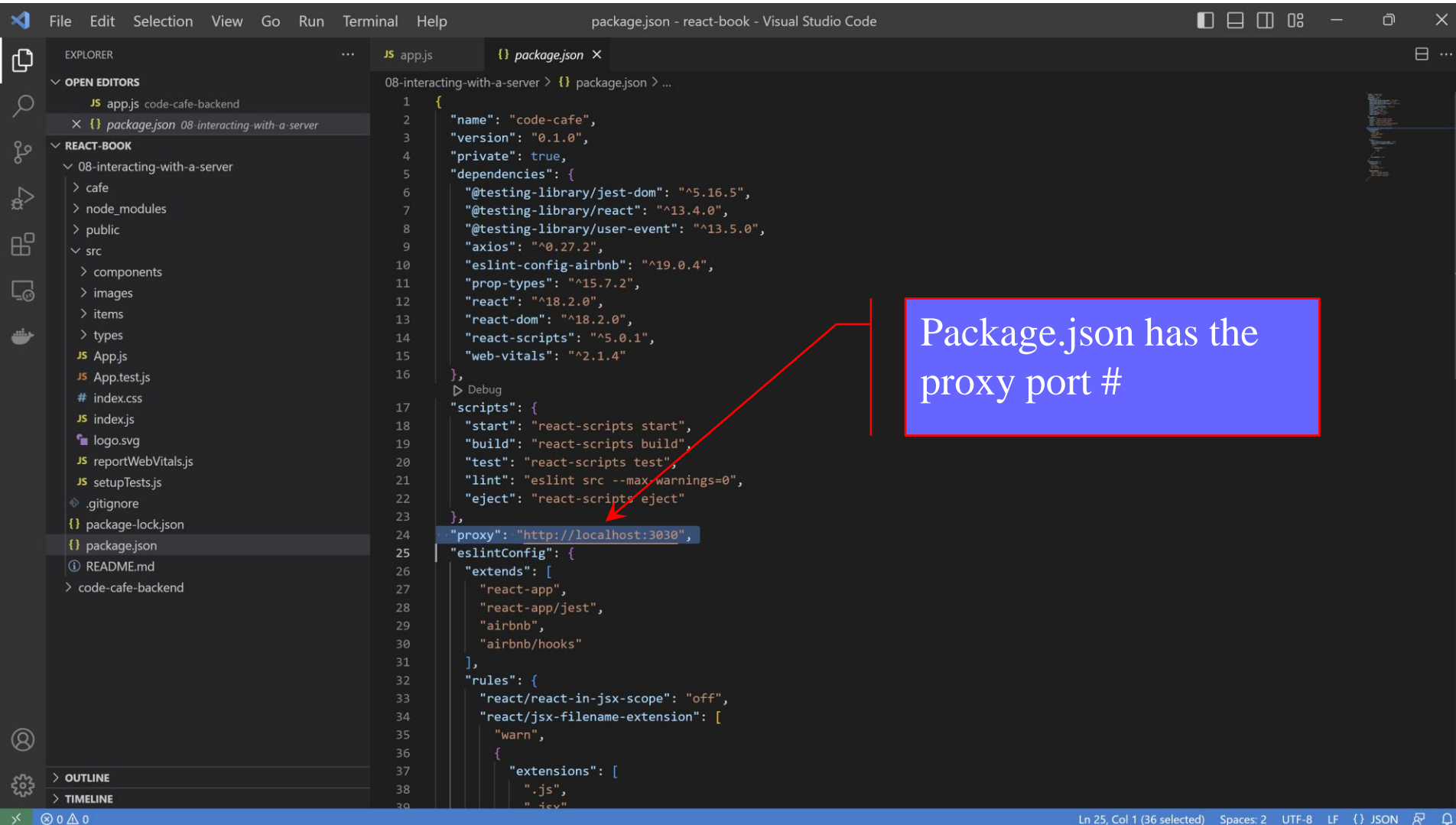


```
import axios from 'axios';
import { useEffect } from 'react';
import Header from './components/Header';
...
function App() {
  useEffect(() => {
    axios.get('/api/items');
  }, []);

  return (
    ...
  )
}
```

The first argument you supply to `useEffect` is the callback function. This function uses the Axios library to make a GET request to the API.

Code Café React App



The screenshot shows the Visual Studio Code interface with the `package.json` file open. The file contains the following content:

```
1 {
2   "name": "code-café",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^5.16.5",
7     "@testing-library/react": "^13.4.0",
8     "@testing-library/user-event": "^13.5.0",
9     "axios": "^0.27.2",
10    "eslint-config-airbnb": "^19.0.4",
11    "prop-types": "^15.7.2",
12    "react": "^18.2.0",
13    "react-dom": "^18.2.0",
14    "react-scripts": "^5.0.1",
15    "web-vitals": "^2.1.4"
16  },
17  "scripts": {
18    "start": "react-scripts start",
19    "build": "react-scripts build",
20    "test": "react-scripts test",
21    "lint": "eslint src --max-warnings=0",
22    "eject": "react-scripts eject"
23  },
24  "proxy": "http://localhost:3030",
25  "eslintConfig": {
26    "extends": [
27      "react-app",
28      "react-app/jest",
29      "airbnb",
30      "airbnb/hooks"
31    ],
32    "rules": {
33      "react/react-in-jsx-scope": "off",
34      "react/jsx-filename-extension": [
35        "warn",
36        {
37          "extensions": [
38            ".js",
39            ".jsx"
40          ]
41        }
42      ]
43    }
44  }
45 }
```

A red arrow points from the text box to the `"proxy": "http://localhost:3030"` line in the `package.json` file.

Package.json has the proxy port #

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/ch08.xhtml#`. The page content discusses asynchronous operations and promises. Two blue callout boxes with red borders and arrows point to specific text on the page:

- The first callout box, located on the right side, contains the text: "How to handle Asynchronous operations?". It points to the paragraph starting with "moving to the next operation until the previous operation is complete."
- The second callout box, located below the first, contains the text: "Axios returns a promise for the request". It points to the highlighted sentence: "The library you are using to make your network requests, Axios, automatically returns a promise when you make a request."

The page content includes the following text:

moving to the next operation until the previous operation is complete. When it comes to making network requests, such as the GET request you just made, this presents a problem. Because you make your request to an external resource, you have no control over when the request will be fulfilled. So how can you know when your app has received the items?

One way to handle asynchronous operations, such as requests to an API, is to use *promises*. A promise is a JavaScript object that represents a future value. When the asynchronous operation completes, the promise is considered fulfilled. If there is an error, the promise is considered rejected.

The library you are using to make your network requests, Axios, automatically returns a promise when you make a request. To specify the action that occurs after the promise is fulfilled, you use the promise construct `then`. Code inside the `then` block does not execute until the promise is complete. If there is an error, the `catch` block executes.

How promises work

For many developers, promises are one of the most confusing aspects of JavaScript

At the bottom of the browser window, a search bar contains the text "axios" and shows "1 of 16 matches".

Code Café React App

Use the hook `useState` to add new state value: `items`

Promise for `axios.get`

```
1 import axios from 'axios';
2 import { useEffect, useState } from 'react';
3 import Header from './components/Header';
4 import Home from './components/Home';
5
6 function App() {
7   const [items, setItems] = useState([]);
8
9   useEffect(() => {
10     axios.get('/api/items')
11       .then((result) => setItems(result.data))
12       .catch(console.error);
13   }, []);
14
15   return (
16     <div>
17       <Header />
18       <Home items={items} />
19     </div>
20   );
21 }
22
23 export default App;
```

Code Café React App

The image displays the development workflow for the Code Café React application. On the left, two terminal windows show the execution of commands to start the back-end server and the front-end React application. On the right, a web browser window shows the running application at localhost:3000, which features a grid of food and drink items. Red arrows and numbered callouts highlight the key steps in the process.

```
npm start
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\bader\Projects\React\Tutorial_4\react-book\code-cafe-backend>npm start

> rest-api@0.0.0 start
> node app.js

Listening on http://localhost:3030
Accessing /api/items
Accessing /api/items
Accessing /api/items
```

1. Startup back-end server

```
Windows PowerShell
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\bader\Projects\React\Tutorial_4\react-book\08-interacting-with-a-server>npm start

> code-cafe@0.1.0 start
> react-scripts start

Browserslist: caniuse-lite is outdated. Please run:
  npx update-browserslist-db@latest
  Why you should do it regularly: https://github.com/browserslist/update-db#readme
(node:24548) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' is deprecated. Please use the 'setupMiddlewares' option.
(node:24548) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view code-cafe in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.0.222:3000

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

webpack compiled successfully
```

2. Startup react

3. Code Café started

Code Café

localhost:3000

Coffee

Cookie

Croissant

Cupcake

French Press

Iced Coffee

Milk

Sandwich

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 (2nd 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 (6th Edition)*. Essex, England: Pearson Education Publications. [ISBN: 978-1292153919]
 - Flanagan, D. 2020. *JavaScript: The Definitive Guide (7th 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]
-