Getting Started

6 min

Before you start building the WebSocket chat application, let’s take a moment to explore the starter code that has been provided for you.

You’ll notice that the starter code includes a file called server.js. This is where you’ll be creating your WebSocket server and implementing the server-side functionality required to create a real-time chat application.

In the last lesson, you learned how WebSocket connections are established over an HTTP server. In server.js, we’ve provided an HTTP server instance (created with node’s http package) for you to use when establishing your WebSocket server endpoint.

In /public, you’ll notice a file called index.html. This file contains the content that will display in the browser when you access the server and is where you will write your client-side WebSocket logic.

Each of the files listed above will have large comment headers separating the various concerns of the application. There will also be

Preview: Docs Loading link description

comments

 throughout indicating where you should be writing your code for each exercise. In some instances, you may be asked to write a function or a large block of code over the course of multiple exercises.

Once you’ve had a chance to explore the starter code, follow the instructions below for running the application for the first time!

Instructions

Task 1

We need to install the dependencies for the project which includes the ws package.

Open a terminal window. Then, use the command line to navigate to the root of the starter code directory, and run the following command:

```
npm install
```

Help

Use the cd command to change directories into the project code you just downloaded. Then run the command above.

Task 2

With the dependencies installed, we can now start the server. In your project directory, enter the command below into your terminal.

```
node server.js
```

If this command is successful, you should see the message Listening on: http://localhost:8080 displayed in your terminal.

If you ever need to restart the server, press Control+c to stop the server and then enter the node server.js command again.

Task 3

Open localhost:8080 in two different browser tabs. Take a moment to test the app's functionality by typing in a message and pressing the send button (or Enter). You'll notice that messages sent in one tab are displayed only in that tab. You will also notice that your browser is not yet connected to the WebSocket server.

Your task over the course of this lesson will be to use WebSockets to turn the app into a functional chat room wherein users can send and receive messages with other users, all through a WebSocket server.

Let's begin!