Create a WebSocket Server

5 min

WebSocket is a protocol for transmitting and receiving messages between client(s) and a server – to add WebSocket functionality to an application, you will have to implement both sides of this communication. Let's start with the server-side of our chatroom application.

While it would be possible to build a WebSocket server from scratch, it's common to use a pre-built implementation. For this lesson, we will be using the popular ws package, which may be installed using the Node Package Manager.

The ws package exports a WebSocket object with a .Server property corresponding to the WebSocketServer class. Calling this property's constructor function allows you to instantiate a new WebSocket Server, like so:

```
// server.js
const WebSocket = require('ws');

// httpServer is a premade HTTP server
const wsServer = new WebSocket.Server({ server: httpServer });
```

The WebSocket.Server() constructor function requires only one argument: an object of configuration options. Though there are a number of ways to configure this object, in the example above, we provide a premade HTTP Server called httpServer using the server key. The WebSocket server can then use this HTTP connection to establish a WebSocket connection.

***Note**: creating this httpServer is out of the scope of this lesson however you can learn more about creating an HTTP server by checking out the Setting Up a Server with HTTP lesson in our Learn Node.js course.*

With the WebSocket server in place, the application is now ready to receive WebSocket connection requests!

Instructions

Task 1

At the top of the **server.js** file you should see that the WebSocket object from ws has already been imported. Our first task is to use the WebSocket.Server class to create our WebSocket server instance.

Scroll down to the WS LOGIC section and find the comment labeled // Exercise 3 indicating where you should create your WebSocket Server.

Here, instantiate a new WebSocket.Server instance and assign it to a variable named wsServer.

Help

Make sure to pass in an options object that includes the key server corresponding to the premade HTTP server we have provided for you, also called server.

Your code should look something like this:

```
const wsServer = new WebSocket.Server({ server: server });

// or using ES6 object property naming simply

const wsServer = new WebSocket.Server({ server });
```