Tutorial 7

Sockets

In this tutorial, you will be introduced to the concent of WebSockets. Sockets are connections made of the WebSockets protocol that allows messages to be sent bilaterally between the client and the server. Typical HTTP connections only allow for messages to be initiated from the client and sent to the server. With sockets, the server stores the connections for all the clients and is able to emit messges for any of them at any point.

Start off by `npm install` -ing the `socket.io` and wrapping the current `express` app with a socket server.

```
const server = http.createServer(app);
const { Server } = require("socket.io");
const io = new Server(server, {cors: {origin: "*"}});
```

The `cors` setting is important here for the WebSockets server as the server is hosted on a different domain. The asterisk wildcard means that any site can connect to it.

After installing and setting up the server, add an `io.on` function that prints out when a new connection is made and when a connection is disconnected. It is helpful for debugging issues.

Frontend

Moving to the frontend, add a `<script>` tag with the following contents:

```
<script src="http://localhost:8000/socket.io/socket.io.js"></script>
```

This gives us access to the `io` object which we can connect to our server as such:

```
var socket = io('http://localhost:8000');
```

With this socket object, we can send messages to the servers. The server can then setup a `socket.on` handler to handle the emit code emitted by the server. Let's emit a `newTweet` message to the server every time we post a tweet on the client. Set the value of the socket message to `true` .

When setting up the handler for this in the backend, make sure the `newTweet` broadcast value is set to `true` as well.

Upon receival of a `newTweet` message from the server, display a sort of button or banner that when clicked refreshes all the tweets. This should mimic Twitter's built in "new tweets refresh" button.