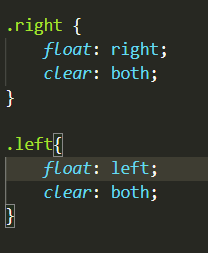
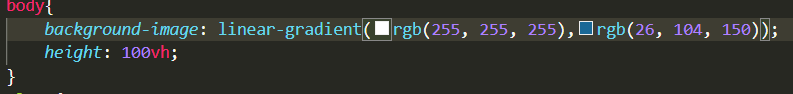
**REAL TIME CHAT APP**

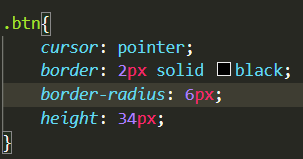
In web developments when we are posting our requests to the server as we have used the get request and post request then this connection is called as ONE WAY Connection the server will respond after getting the request.

* Server never initiates a response that take this(updates) to the client.
* Server and clients are not connected.
* One method using the HTTP is called Long Polling but we will focus on WEB SOCKETS here.
* Web sockets is basically a two way connections between a client and server which means they can send data without the need of requests. Server can push the updates to the client , and client can push to server.
* SOCKETIO: helps is fixing this problem which means that using this framework we will be able to do things in REAL TIME as when we request the server for data it will revert back not only with the data but also with the UPDATES AUTOMATICALLY so the client and server are more connected in this case than the previous GET-POST html method.
* It is used in making things like REAL TIME.
* WS , WSS: Web socket protocol, web secure socket protocol.
* **Socket.IO is a JavaScript library for Real time web applications. It enables Real time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have a nearly identical API**.
* EVENT: is something which is basically whatever the client will do the webpage, for example if the client clicks on something that is an event, if client drags something somewhere that is an event. And in response to this the backend JS that we write can CATCH those EVENETs and respond accordingly. And in the similar manner if a client gives some event then the server can catch it and respond accordingly.
* Sockets have traditionally been the solution around which most real-time chat systems are architected, providing a bi-directional communication channel between a client and a server.
* This means that the server can push messages to clients. Whenever you write a chat message, the idea is that the server will get it and push it to all other connected clients.

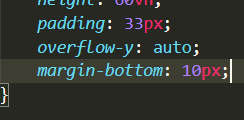


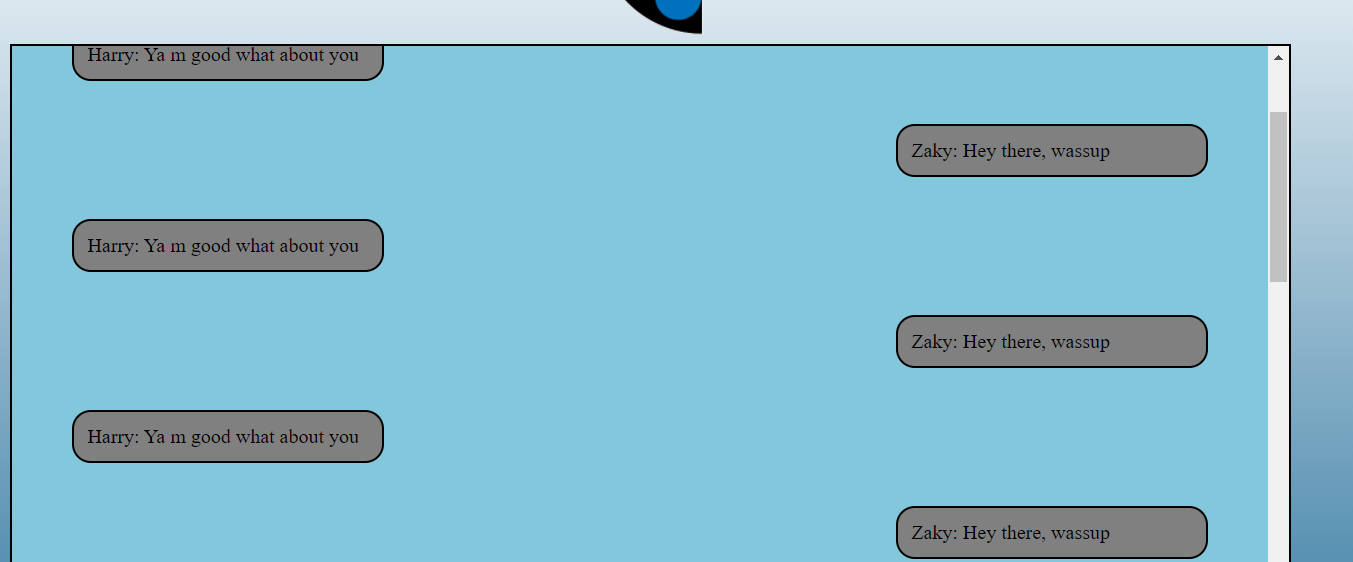


* The cursor property of CSS allows **you to specify the type of cursor that should be displayed to the user**. One good usage of this property is in using images for submit buttons on forms. By default, when a cursor hovers over a link, the cursor changes from a pointer to a hand.

 When the user write in the box and clicks send then the page reloads

* Made the overflow of y as the auto as when there will be more messages then we can scroll down , automatically the scroll button will appear.





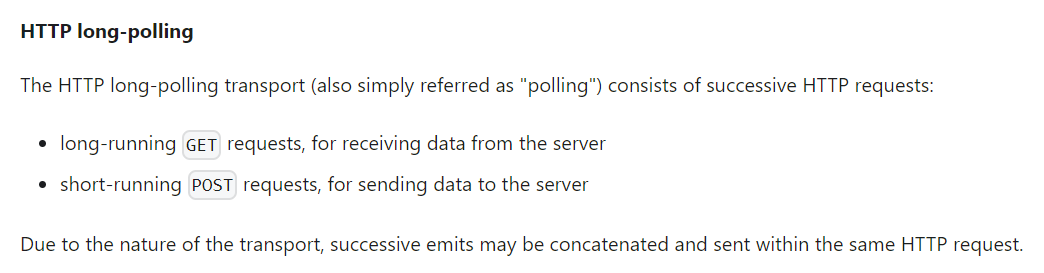
* The io is is basically requiring the socket.io which we have installed.
* Io.on is basically that event which is accountable for all the connection in the app, whereas socket.on means that when two people talking that will be handled by socket.on and entire people can be handled by the io.on.
* So we created an event “new-user-joined” . when this event happens a call back function happens which hold the name object and we will provide the user id of that socket and call it name.
* Now inside the same io.on which is responsible for the entire server we will be holding event of sending and receiving messages
* When the “send” event happens then the message call back will execute as it will broadcast the and emit the message to be received and it will be message and then the name of the user id which we have provided.
* Note: the event names mentioned above are custom, basically we will call that event by that name which is custom written by us, just like variable names.



* One simple script change makes that security error go

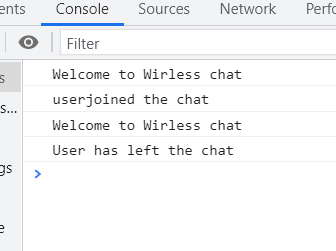
 Created a simple variable hmr, and declared it as nodemon and then ran the show.

Since the “name” was deprecated so I believed that the entire course may contain lots of bugs Therefore the best way is to make it by ourselves as I want to implement EXPRESS in this app and I want to host it on Netlify.

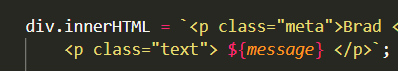




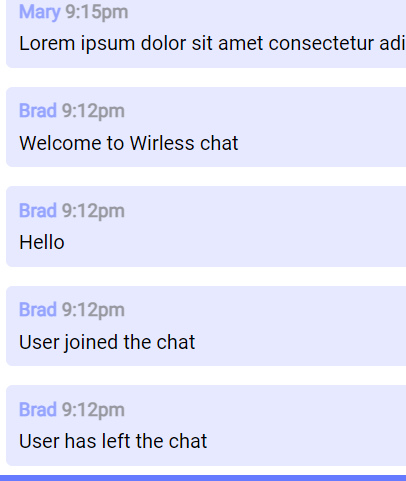




* BACKTICKS: You see this is not a single quote this is a a key I donnot what to call it it on the left top below the Esc button

 Notice beginning and ending of the <p> tag

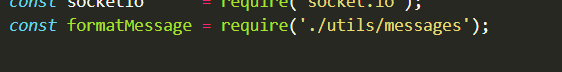
* These messages pops up when we write a message as it is catched by the console and we are using it to outputMessage



* The function we created for username, text, and live time



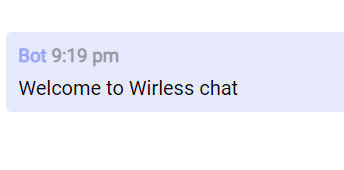
* The incorporation in the server.js



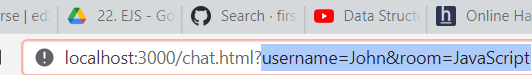
* The output message



* The result



* A library that grabs the query tags in the browser address

These 

Can be done by using the qs library

<https://cdnjs.com/libraries/qs>