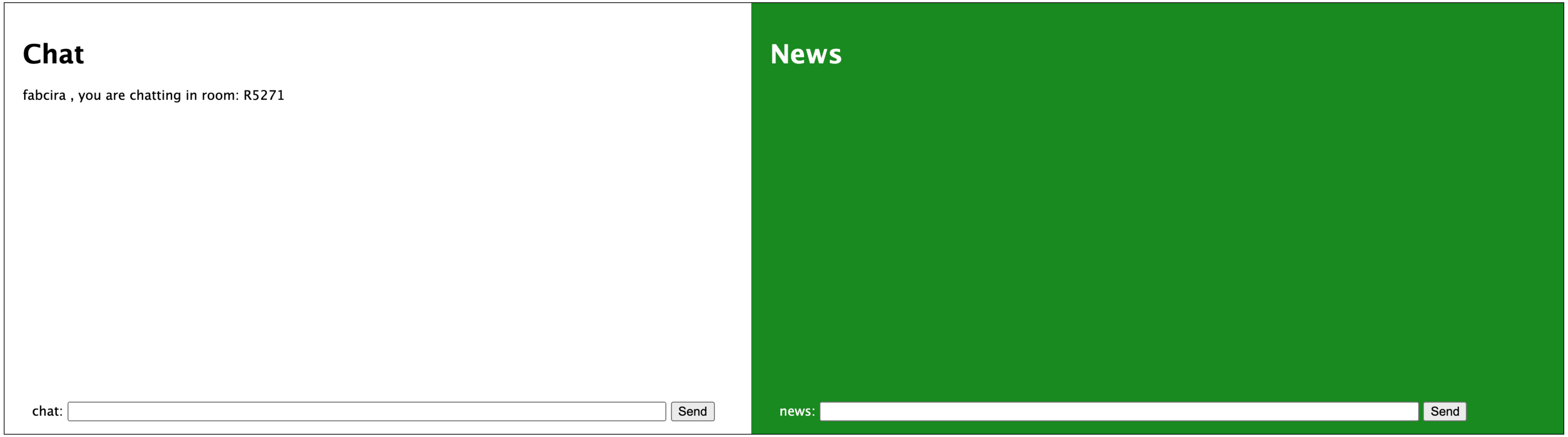

Chat

fabcira , you are chatting in room: R5271

chat: Send

News

news: Send

Lab Class Week 3.b
Learning to use socket.io

Professor Fabio Ciravegna
f.ciravegna@shef.ac.uk

© Fabio Ciravegna, University of Sheffield

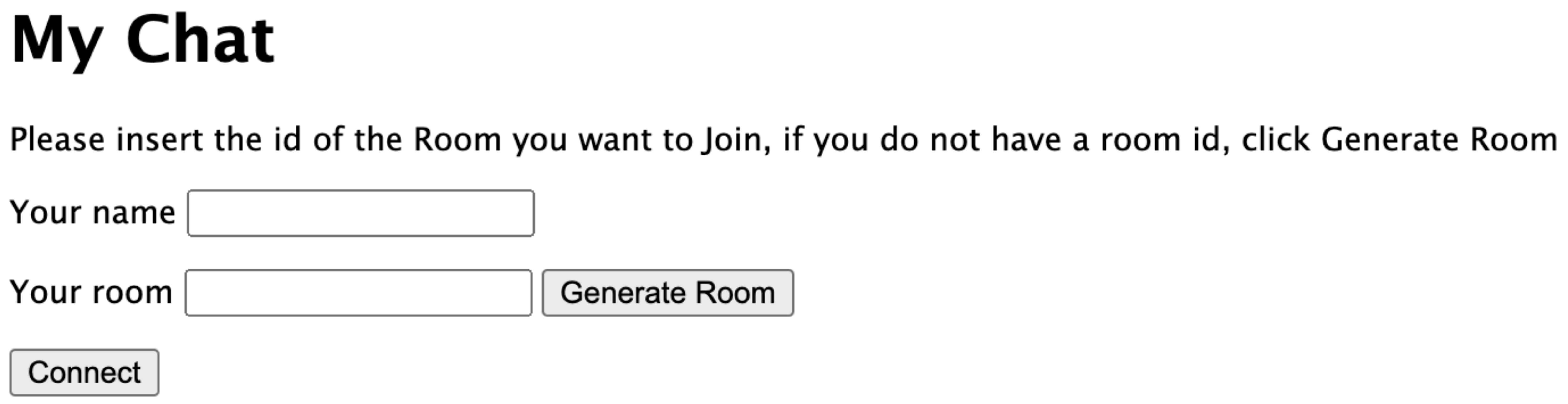
Exercise 2

- In this exercise we will see how to build a chat system using socket.io
- The exercise is divided into two parts
  - Inspecting an existing chat system to understand how socket.io works
  - Adding a namespace to the chat system
    - which will require to define a new chat system similar to the one provided
- Provided:
  - the code of an implemented chat system for you to inspect and understand
  - a new version of the code above modified to support namespaces
    - to use as a starting point to add the new namespace

© Fabio Ciravegna, University of Sheffield

The base chat system

- It implements a basic chat
- The interface has just one page
- Initially the user is asked for their name and the name of the room they want to join
  - if they do not have a room yet, they can generate a new name



My Chat

Please insert the id of the Room you want to Join, if you do not have a room id, click Generate Room

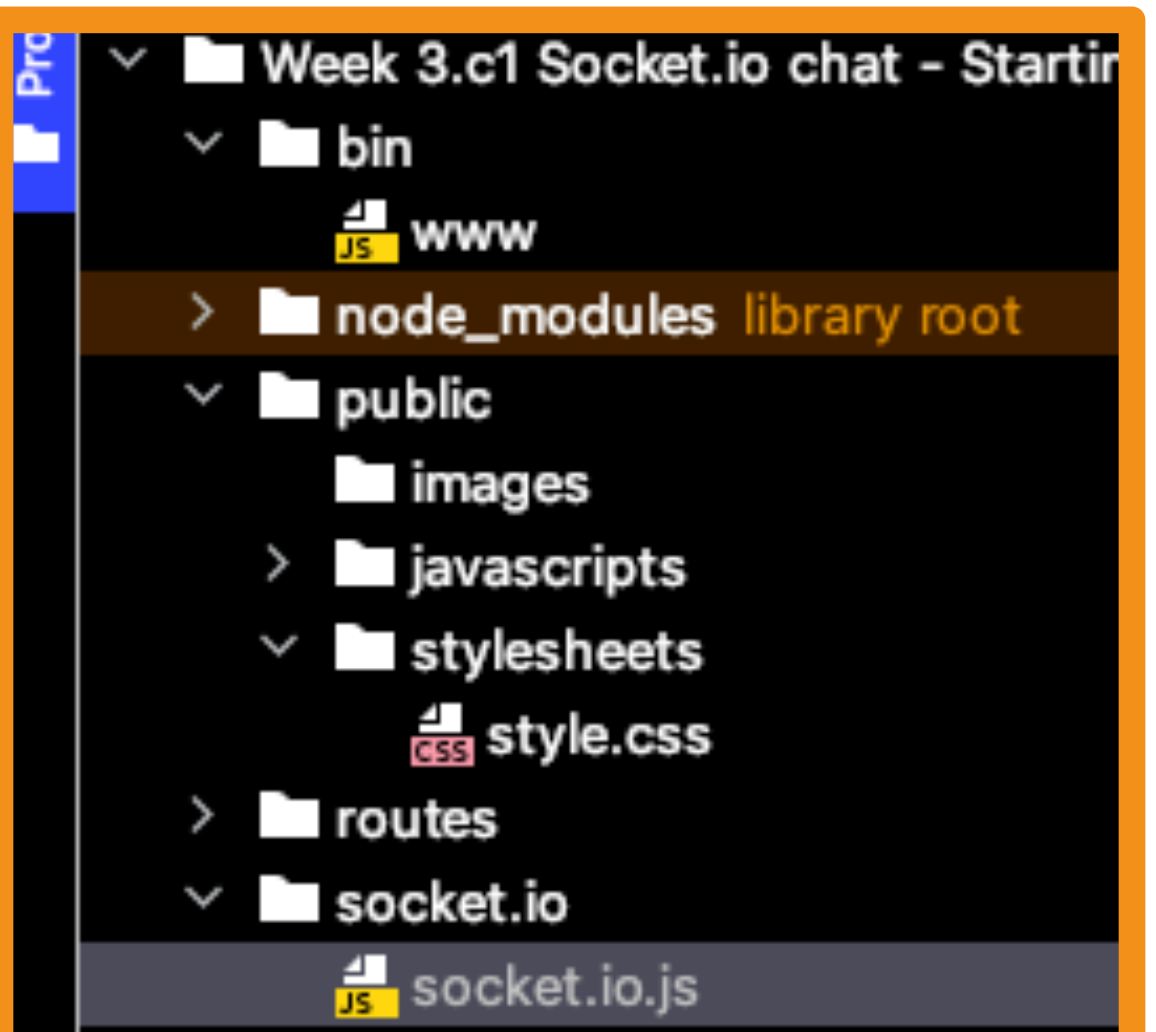
Your name

Your room

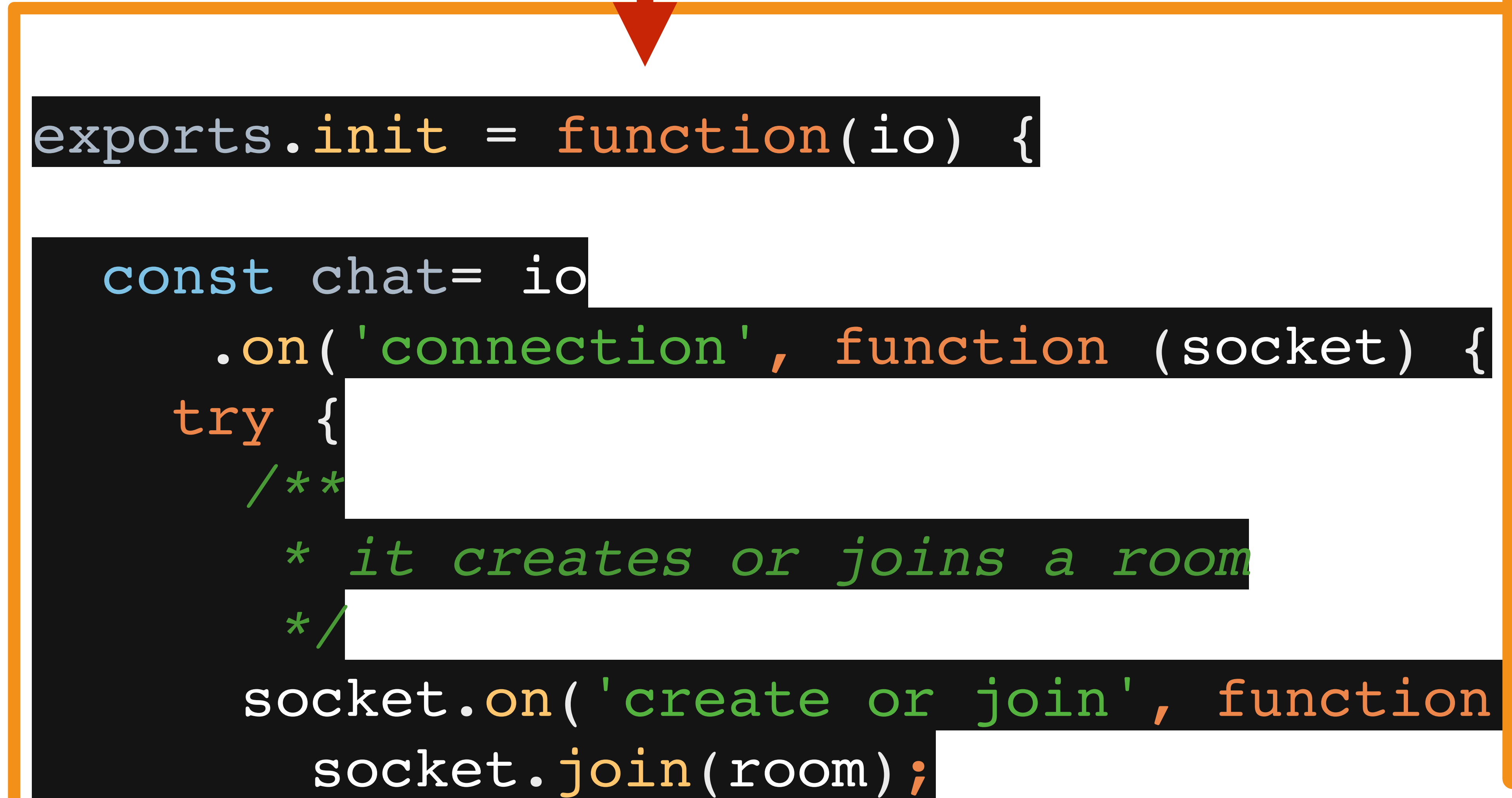
© Fabio Ciravegna, University of Sheffield

Installing socket.io

- open package.json
  - go to the bottom and start typing "socket.io", "^ (then select the top version)
    - do not forget to add the comma to the previous line!!!
    - the part "^X.X.X" will be highlighted. Right click and select "run rpm install"
- Then add folder called socket.io
  - and create a JS file called socket.io.js
    - where you will add the socket commands

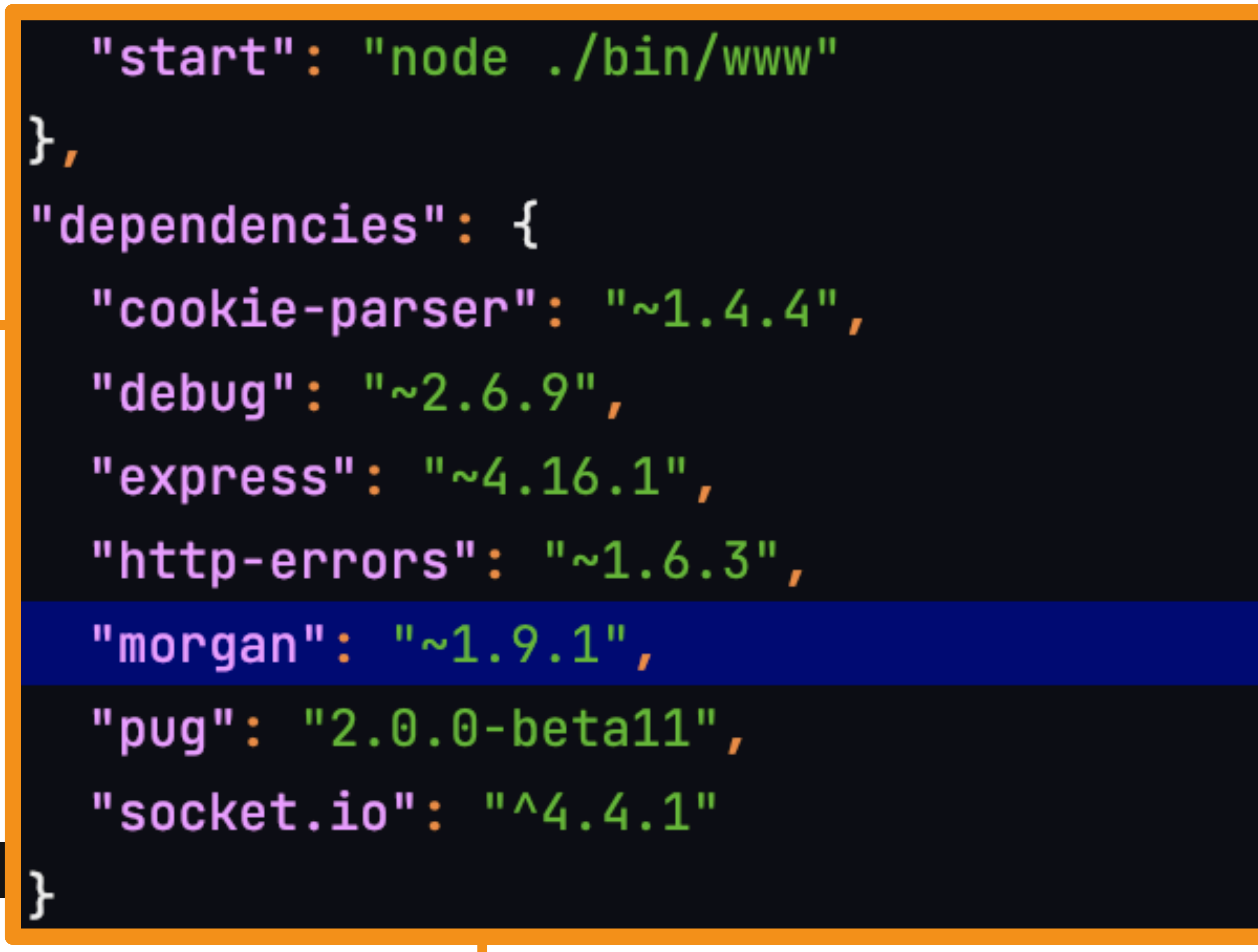


```
Week 3.c1 Socket.io chat - Startir
  bin
    www
  node_modules library root
  public
    images
    javascripts
    stylesheets
      style.css
  routes
  socket.io
    socket.io.js
```



```
exports.init = function(io) {

  const chat= io
      .on('connection', function (socket) {
    try {
      /**
       * it creates or joins a room
       */
      socket.on('create or join', function
        socket.join(room);
```



```
    "start": "node ./bin/www"
},
"dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "morgan": "~1.9.1",
    "pug": "2.0.0-beta11",
    "socket.io": "^4.4.1"
}
```

Server side

- Declare socket.io at the end of bin/www

```
const io = require('socket.io')(server, {
  pingTimeout: 60000,
});
var socket_module = require('../socket.io/socket-io');
socket_module.init(io, app);
```

© Fabio Ciravegna, University of Sheffield

socket.io operations

- In socket.io/socket-io.js We define two operations:
  - ‘create or join’ called when a room is joined
  - ‘joined’ called when someone joins the room
  - ‘chat’ called when someone sends a message
- Each of them receive at least a room and a user name
  - and will write to all participants in the room (including the sender)
    - using io.sockets.to(room).emit()

© Fabio Ciravegna, University of Sheffield

Declare operations in socket-io.js

operations server side

```
exports.init = function(io) {
  io.sockets.on('connection', function (socket) {
    try {
      /** it creates or joins a room  */
      socket.on('create or join', function (room, userId) {
        socket.join(room);
        io.sockets.to(room).emit('joined', room, userId);
      });
     socket.on('chat', function (room, userId, chatText) {
        io.sockets.to(room).emit('chat', room, userId, chatText);
      });
     socket.on('disconnect', function(){
        console.log('someone disconnected');
      });
    } catch (e) {  }});}
```

© Fabio Ciravegna, University of Sheffield

Client side socket.io

Week 3.c Socket.io chat 〉 public 〉 javascripts 〉 index.js 〉 writeOnHistory()
Project
www × package.json × socket-io.js × error.ejs × index.ejs × style.css × ro
Week 3.c Socket.io chat ~/Documents/Teachir
.idea
bin
node_modules library root
public
images
javascripts
index.js
stylesheets
routes
socket.io
socket-io.js
views
app.js
package.json
package-lock.json
Week 3.c Socket.io chat.iml
External Libraries
Scratches and Consoles
Structure

```
let name = null;
let roomNo = null;
let socket = io();


/**
 * called by <body onload>
 * it initialises the interface and the expected socket
 * plus the associated actions
 */
function init() {
    // it sets up the interface so that userId and room
    document.getElementById( elementId: 'initial_form').st
    document.getElementById( elementId: 'chat_interface').

    // called when someone joins the room. If it is som
```

declare a js file

declare socket.io and connect

create an init function

The University Of Sheffield.

Joining a room

```
<form onsubmit="return false;">
    <p><label for="name"> Your name </label>
        <input type="text" id="name" name="name">
    </p>
    <p>
        <label for="roomNo"> Your room </label>
        <input type="text" id="roomNo" name="roomNo">
        <button id="roomNoGenerator" onclick="generateRoom()">Generate Room</but
    </p>
    <button id="connect" onclick="connectToRoom()">Connect</button>
</form>
```

view/index.ejs

javascripts/index.js

```
function connectToRoom() {
    roomNo = document.getElementById('roomNo').value;
    name = document.getElementById('name').value;
    if (!name) name = 'Unknown-' + Math.random();
    socket.emit('create or join', roomNo, name);
}
```

© Fabio Ciravegna, University of Sheffield

client side: javascripts/index.js

```
let name = null;
let roomNo = null;
let socket = io();
function init() {
    // it sets up the interface so that userId and room are selected
    document.getElementById('initial_form').style.display = 'block';
    document.getElementById('chat_interface').style.display = 'none';
    // called when someone joins the room. If it is someone else it notifies
    // the joining of the room in the chat
    socket.on('joined', function (room, userId) {
        if (userId === name) {
            // if we have joined, we show the chat interface
            hideLoginInterface(room, userId);
        } else {
            // notifies that someone has joined the room
            writeOnHistory('<b>'+userId+'</b>' + ' joined room ' + room);
        }});
    // called when a message is received
    socket.on('chat', function (room, userId, chatText) {
        let who = userId
        if (userId === name) who = 'Me';
        writeOnHistory('<b>' + who + ':</b> ' + chatText);});
```

client side:

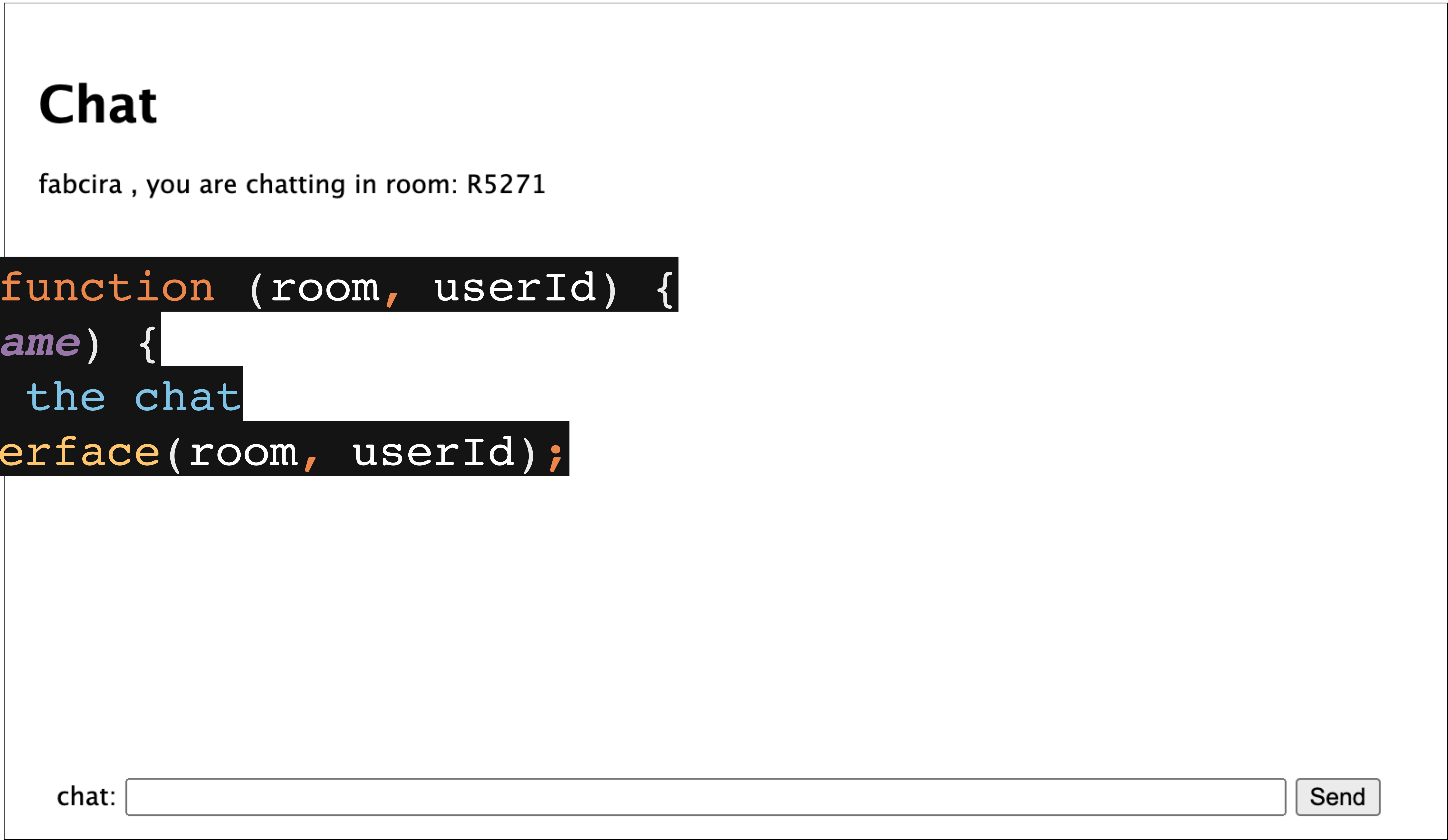
receiving a joined message

receiving a chat message

© Fabio Ciravegna, University of Sheffield

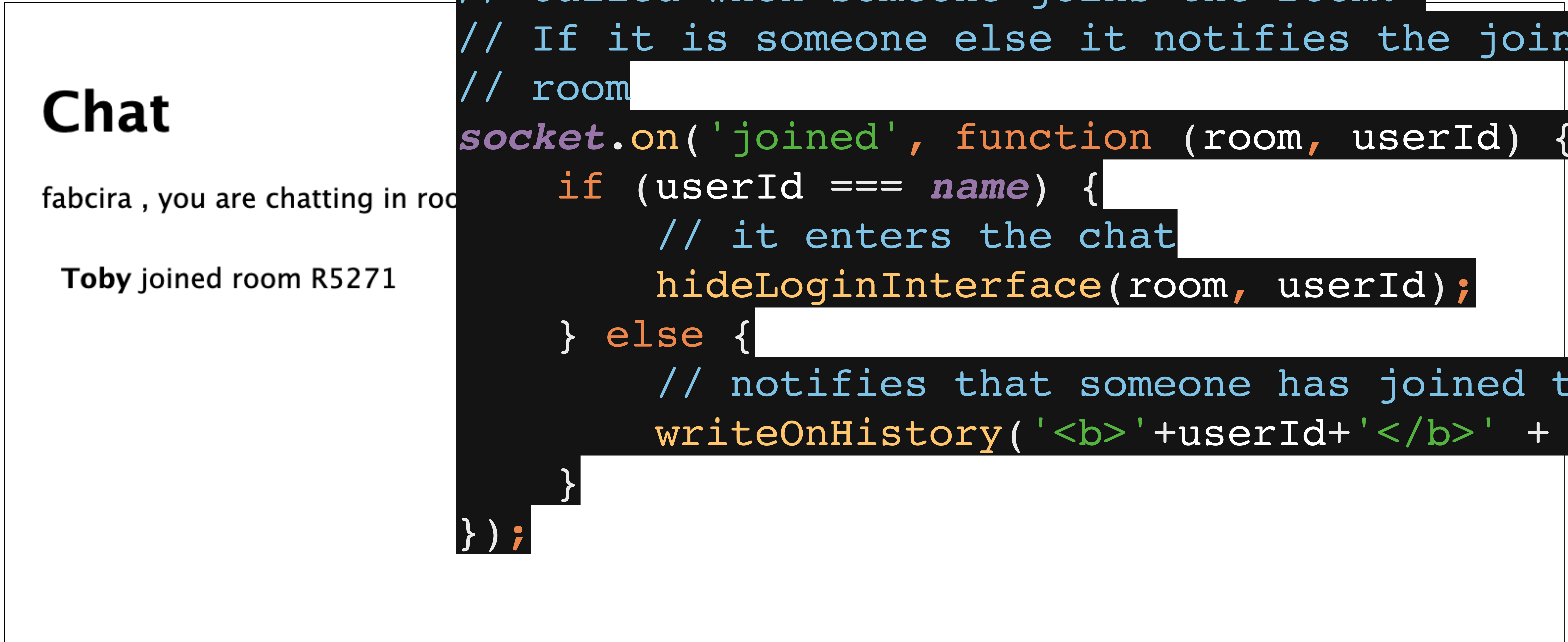
The base system

- When a room is joined the chat system will appear as follows



```
socket.on('joined', function (room, userId) {
    if (userId === name) {
        // it enters the chat
        hideLoginInterface(room, userId);
…
});
```

- When someone else joins the room, the participants in the room are notified (function writeOnHistory)


Chat

fabcira , you are chatting in roo

Toby joined room R5271

```
// called when someone joins the room.
// If it is someone else it notifies the joining of the
// room
socket.on('joined', function (room, userId) {
    if (userId === name) {
        // it enters the chat
        hideLoginInterface(room, userId);
    } else {
        // notifies that someone has joined the room
        writeOnHistory('<b>'+userId+'</b>' + ' joined roo
    }
});
```

© Fabio Ciravegna, University of Sheffield

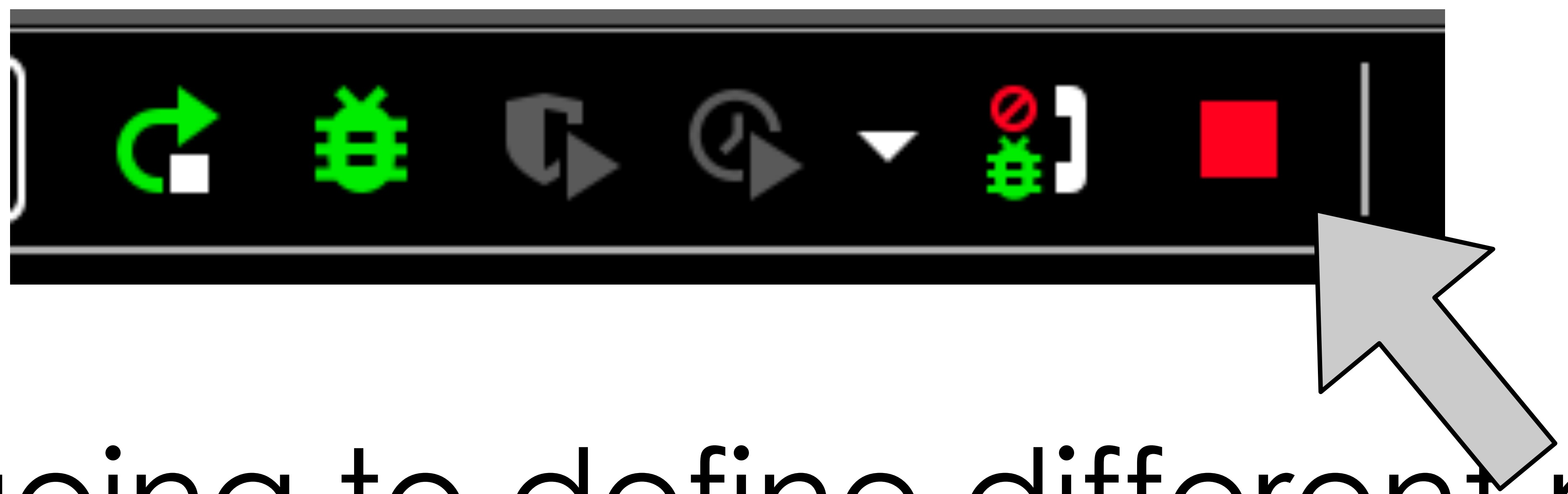
- When posting a sentence, this is shown in the history. The name of the sender is shown
  - e.g. Toby: hello!
- If it was sent by us, our name will be replaced by “Me:”
  - e.g. Me: hello!

```
// called when a message is received
socket.on('chat', function (room, userId, chatText) {
    let who = userId
    if (userId === name) who = 'Me';
    writeOnHistory('<b>' + who + ':</b> ' + chatText);
});
```

© Fabio Ciravegna, University of Sheffield

Moving to the next stage

- Make sure to understand the code
- Stop the server (red square close to the start server triangle)



- We are now going to define different name spaces where we will post on different channels
- Open the project
  - **Week 3.c1 Socket.io chat - Starting point for Solution**
  - Run the server
    - if you get a message saying that the port is already in use, you have not stopped the previous server (see above)

© Fabio Ciravegna, University of Sheffield

- the new chat has split screen with two channels corresponding to two namespaces
  - /chat and /news

Chat

fabcira , you are chatting in room: R5271

chat:
Send
News
news:
Send

- The system works as before but now the chat is executed in a new name space called /chat
  - The client side changes slightly by defining the /chat name space and use it instead of the variable socket:

```
let socket = io();
```

```
let chat= io.connect('/chat');
```

```
socket.on('joined', function (room, userId) {
socket.on('chat', function (room, userId, chatText) {
```

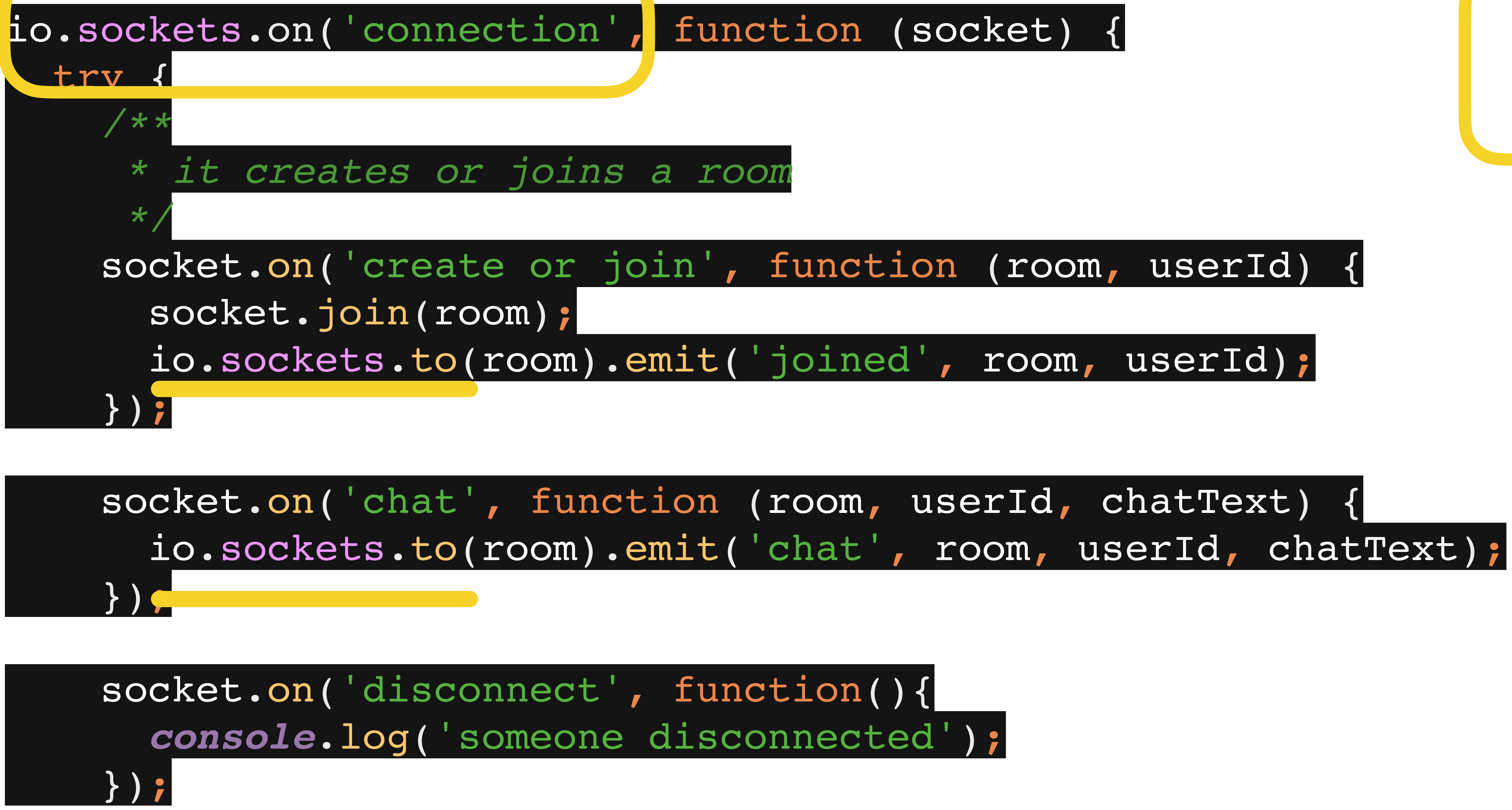
```
chat.on('joined', function (room, userId) {
chat.on('chat', function (room, userId, chatText)
```

Server side

- The server side changes by declaring the same operations now defined in a name space

© Fabio Ciravegna, University of Sheffield

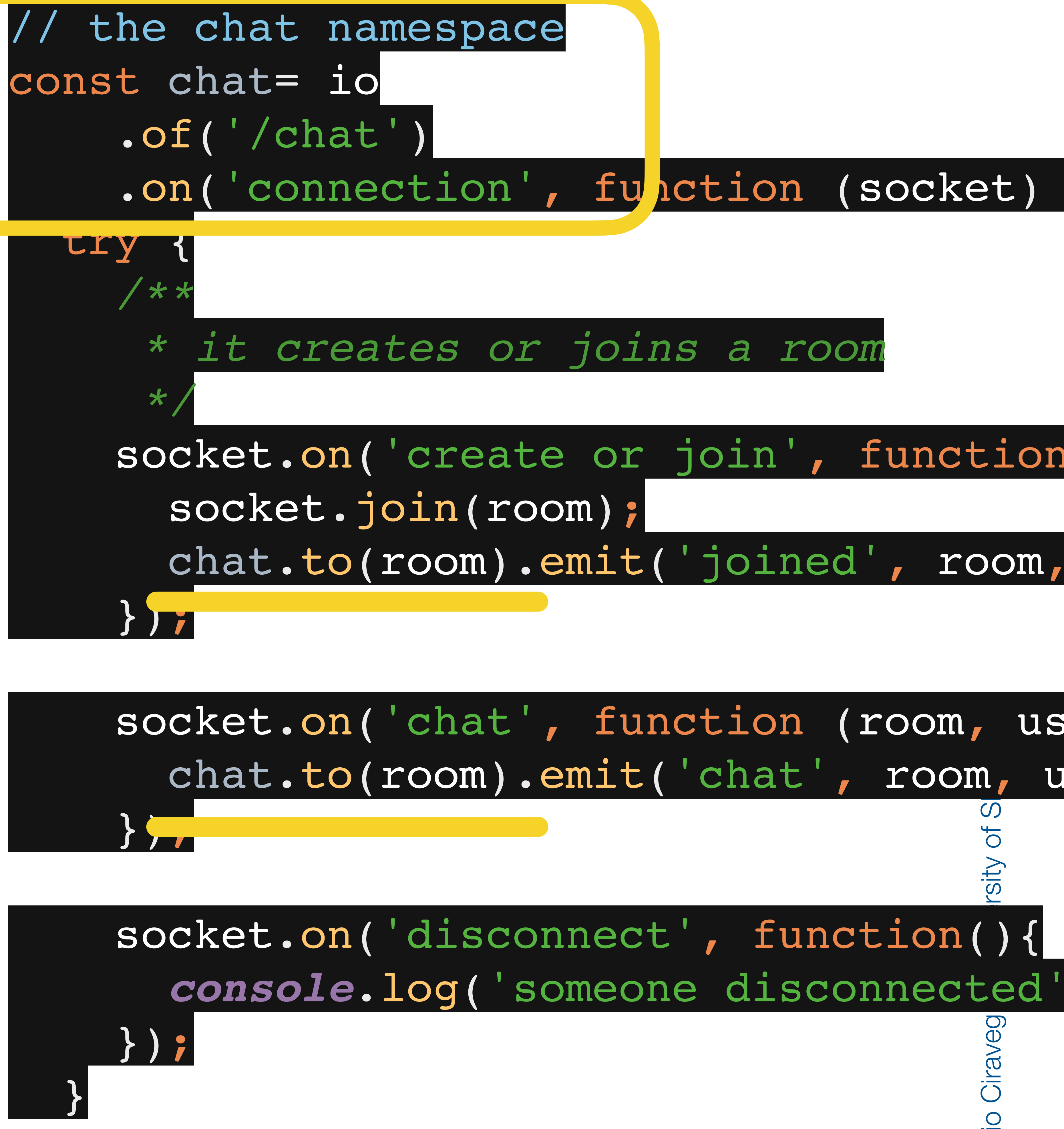
in socket.io/socket.io.js



```
io.sockets.on('connection', function (socket) {
  try {
    /**
     * it creates or joins a room
     */
    socket.on('create or join', function (room, userId) {
      socket.join(room);
      io.sockets.to(room).emit('joined', room, userId);
    });

    socket.on('chat', function (room, userId, chatText) {
      io.sockets.to(room).emit('chat', room, userId, chatText);
    });

    socket.on('disconnect', function(){
      console.log('someone disconnected');
    });
```



```
// the chat namespace
const chat= io
     .of('/chat')
     .on('connection', function (socket)
  try {
     /**
      * it creates or joins a room
      */
     socket.on('create or join', function
       socket.join(room);
       chat.to(room).emit('joined', room,
     });

     socket.on('chat', function (room, us
       chat.to(room).emit('chat', room, u
     });

     socket.on('disconnect', function(){
       console.log('someone disconnected'
     });
  }
```

© Fabio Ciraveg rsity of S

The Exercise

- Create the routes for the /news name space
- It will have the same operations as /chat both on client and on the server
  - I have left a few @todos in the code to guide you
- Hints:
  - declare namespaces and operations in socket.io.js
    - see @todo
  - declare namespaces and operations in javascripts/index.js
    - start from the function initChatSocket() (see @todo)
  - I have already defined a stub function called sendNewsText()
    - which will receive the text typed in the news form

© Fabio Ciravegna, University of Sheffield

Useful Editor tips

- To inspect where a function or variable is used or defined:
  - click on its name in the editor and hit **Command-b** on a Mac or **Control-b** on Windows
    - try it on sendNewsText now
    - if you keep hitting the key you will move between definition and uses

- To search across the entire project use Control-SHIFT-F on a Mac
  - not sure about Windows: check under Edit > Find > Find in Files
  - Try it now to search for all the occurrences of @todo

© Fabio Ciravegna, University of Sheffield

In the solution

- I have added one feature to the /news channel in the solution:
  - The news are are not copied to the author's history
    - This is to showcase the use of

      socket.broadcast.to**(room).**emit(…);

    - which sends a message to all the participants except the originating one
    - as opposed to using

      chat.to(room).emit(…);

    - which sends the message to everybody including the author
  - Note the difference: the latter
  - uses the namespace (**chat.**), while broadcast uses the socket received as parameter (**socket.**)

© Fabio Ciravegna, University of Sheffield

You should be able to do this exercise in about 40 minutes

© Fabio Ciravegna, University of Sheffield