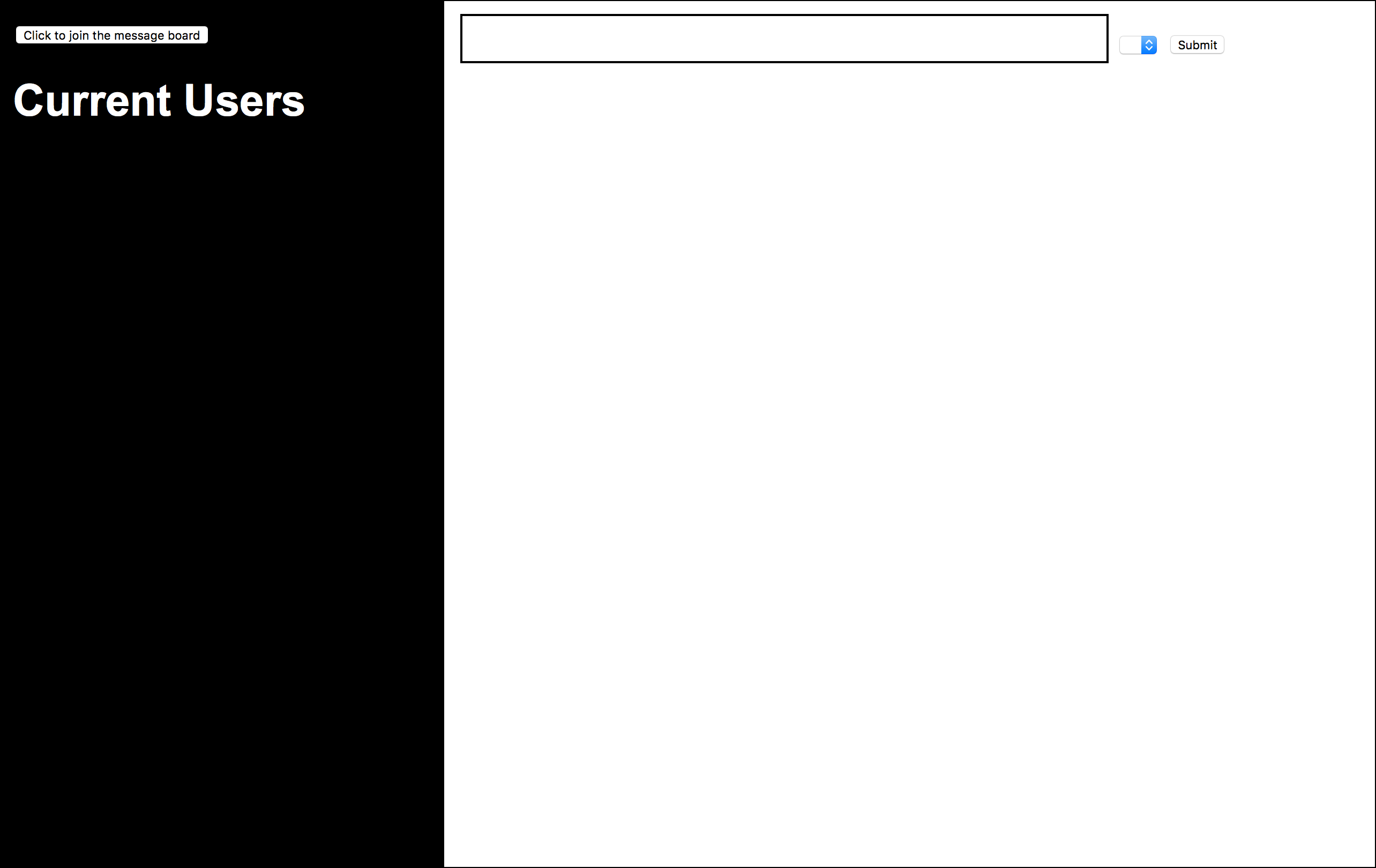
# Your Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Lab 6: due to BCIT’s “Desire to Learn” lab6 dropbox

## Description

In this lab you will create a small application that approximates the functionality of a message board/chat room. An HTML template with CSS has been provided – you must add the JavaScript code necessary to implement the following behaviours.



* New users (i.e. user objects) can join the chat room by clicking the button on the top left of the page.
* New users will be listed under the ‘Current Users’ heading in the unordered list #userlist (i.e. you must append each new user inside the <ul>, each in it’s own <li>), and will also populate the dropdown menu #user-chooser (each in an <option> inside the <select>) next to the submit button on the top right of the page.
* Once users have been added, they can be selected from the dropdown menu and message text can be typed into the input box shown top middle.
* New messages (i.e. message objects) will appear in the right-hand side of the screen, appended inside the #messages div after they have been submitted
* Each message that is posted must be wrapped in a div with class ‘message’
* The username inside the .message div must be wrapped in a paragraph tag with class ‘user’
* The message content inside the .message div must be wrapped in a paragraph tag with class ‘content’

## Instructions

Use the included HTML template (add your application code between the script tags located before the closing body tag) and create three JavaScript class files:

User.js

Message.js

MessageBoard.js

User.js must contain a definition for the User class, and a constructor that accepts one parameter. The User class must contain one property, *username*, that will be assigned via the constructor’s parameter.

Message.js must contain a definition for the Message class, and a constructor that accepts two parameters. The Message class must contain two properties, *user* (a reference to a User object) and *content* (a string representing the message text), that will be assigned via the constructor’s parameter.

MessageBoard.js must contain a definition for the MessageBoard class, and a constructor that has no parameters. The constructor for the MessageBoard class must create two arrays, *users* and *messages*, that will hold a list of current users and a list of current messages.

The MessageBoard class must also contain five methods:

1. addUser()
2. listUsers()
3. findUser()
4. addMessage()
5. listMessages()

To begin your solution, create an instance (i.e. object) of the MessageBoard class. Whenever the end-user interacts with your webpage (i.e. by creating a user or submitting a message) you must create a User object or Message object, which can then be added to/retrieved from the MessageBoard instance using the methods listed above.

**Note**: the findUser() method should search through the MessageBoard’s array of users to find the User object associated with the name chosen in the dropdown. This will be helpful in creating a Message object before it is added to the MessageBoard.

Your application code (located between the script tags located before the closing body tag) must contain at least two functions: createUser(), which is called when the ‘click to join the message board’ button is clicked; and createMessage(), which is called when the ‘submit’ button is clicked.

Make sure you include all of your files and template with your submission. Put all of your files into one folder, zip that folder, name the folder in the format firstname\_lastname\_lab3a.zip, and upload that zip file to BCIT’s Desire To Learn Lab 6 dropbox.