Group 115 (Amal Vivek , Hosam Abdeltawab)

How To Run

1. Open up terminal and change to the directory containing the program files.
2. Run the command, 'node index.js'.
3. Go to chrome and open the url, 'http://localhost:3000/'.
4. Open the url in multiple tabs to add more clients.
5. Enter a username for each - if you do not enter a username, you will be logged in as guest with the username 'guest****' (* = randomly generated number).
6. Enter message from the different tabs to have a conversation

Program Details

Language Used: Javascript, HTML, CSS
Library Used: NodeJS (Socket IO)
Project: Totally different to Portfolio 1

Checklist

A. Differentiation between sender and receiver - as well as different types of messages (broadcast vs chat)
This is observed through css changes - message sent, message received, connection/disconnection/change username messages.
B. Use of Socket IO to create a server that can handle multiple clients and show them a unique page.
C. Use of Semantic UI to create a sidebar.
D. Use of regexes to differentiate between types of messages.
E. Use of sessionStorage rather than local storage to store usernames for each tab.

Portfolio 2

An instant messenger run on localhost through Socket IO

Amal Vivek, Hosam Abdeltawab

Table of Contents

Overview

The website designed is an Instant Messenger run on localHost by using Socket IO, inspired to an extent by the first lab done in Java. I chose to do an Instant Messenger to understand how to run a server, even on localHost, and deal with multiple clients at the same time. Designing the website involved exploring multiple ways of going about hosting a server. I looked into using PHP, Comet, Websockets and NodeJS. Of all these option, I found Socket IO to have the best documentation and thought it would be best to start writing the program using that as my main base.

The most complex part of this Portfolio was understanding how Socket IO worked and how to send and receive messages. I delved further into jQuery as well adjusting how things are shown in the browser during run time - using functions like append, show, and hide.

To attain higher levels of Bloom's Taxonomy, I looked into adding multiple extra features, such as usernames, broadcast messages, and sending media content. I looked to make the instant messenger as functional as possible, and even looked into hosting it on one of my domains - however for the submission, I decided against it.

Overall, this portfolio allowed my to delve further into server/client programs and gain an adequate understanding of how these functions work. I produced a functional program that is also aesthetically pleasing with a simple user interface.

New And Complex Section

Some of the jQuery and JavaScript I used in the implementation of the Instant Messenger was new and non-trivial. This includes some more of the jQuery used to change how elements are displayed or changed after interacting with them - as done in lab 4 (code.js and code.html). Some of the new functions I used were show, hide and focus.

In addition to this, I had to find an alternative to localStorage to keep content relevant to tabs within those tabs. For this I used sessionStorage - a non-trivial element that I had never used before.

Some of the CSS I used was also new. An example of this is using 'pointer-events'. This was to assert no messages were sent without a username. The only fields clickable while the Enter Username overlay is displayed is the form in that div. Once the submit button is clicked, the 'pointer-events' is reverted to default.

The more complex aspects of the Instant Messenger was the actual set up of the server/client relationship, sending messages and setting up Semantic UI for the sidebar.

None of the web development I had done until this portfolio had any server/client relationship. This was the biggest part of the portfolio that was new and complex for me. It required reading into the documentation of Socket IO and the implementation of it through NodeJS. Socket IO has a lot of well written documentation that made it possible to understand and replicate a server/client system required for the Instant Messenger. Some of the more complex features were the recognising whether the message was sent or received and to display that through CSS. In addition, checking whether the messages are broadcast or person to person. Broadcast messages include, connections, disconnections, and username changes. I looked into sending media content such as images however did not successfully implement that in the given time frame.

Semantic UI was simpler to implement and worked similar to bootstrap with some jQuery functions to toggle the sidebar. The sidebar is there to act as a menu for the program. Being a simple program in terms of what it can do, there aren't many functions in the menu.

Overall, the portfolio was filled with new and complex things that had to be dealt with to create a functional Instant Messenger.

Bloom's Taxonomy

The goal for this portfolio for me was to understand how a server/client system works over localHost. I explored a multitude of options, including PHP, Comet, web sockets and NodeJS. In addition to this, I wanted to explore more modules. Learning about Socket IO and Semantic UI gave insight into how useful and how much work can be cut out by using these well developed modules.

Socket IO provided the base for creating a server/client system. I spent a long time understanding how it works, and decided to use it over php as the API was developed better than what I could have done on my own. It provided the basic functions for server side and client side, which allowed me to create a simple Instant Messenger which would just send and receive text using a simple form.

From this point, I had to add functionality to make it a useable program with a viable interface for any user. This included adding usernames, and correcting the user interface such that there was a distinction between messages sent and messages received. Adding in the usernames was not very difficult, however I had to consider that setting them in localStorage would affect other tabs involved in the conversation. Due to this I used sessionStorage, which worked exactly the same as localStorage however confining the data to the tab it was set in.

In order to to distinguish between message sent and message received, I had to accordingly adjust the CSS and distinguish between the two in CSS. To do this, I looked into the Socket IO documentation to see if anything could be done using the API. In the end, I found it easier to do some checking when the message is received by the client. By comparing the username with the one stored in sessionStorage, I could adjust the CSS accordingly to display the message different if it was sent or received.

In addition, using regexes, I was able to distinguish between broadcast messages and chat messages and such was able to adjust the CSS for broadcast messages. These messages include clients connecting, disconnecting and changing their username. This was slightly difficult as I had to get a grasp of regexes before I was able to implement them. Using them made me realise how powerful regexes are and why they are so commonly used in Perl and Python.

To add some more functionality to the Instant Messenger, I included a sidebar menu. At first, I looked into doing this manually using CSS and jQuery - however after some research, using a module became the obviously better way of going about it. I included the Semantic UI script and css stylesheet and then implemented a sidebar using the documentation that explained how to implement it.

Overall, the Instant Messenger is a very basic chat client that operates on localhost. It incorporates a range of skills learnt in the labs including basic jQuery and HTML, but also covers Socket IO in a lot of detail. The client provides the basic chat functions with a neatly designed user interface. However, I wish I had time to implement some other functions which time didn't allow for - for example, I looked into sending images over the connection using the NodeJS module, ‘delivery’.

The idea behind designing an Instant Messenger for the portfolio was to do something different that could be recognised as a difficult project, while at the same time providing a great learning experience. I learnt about two great modules, Socket IO and Semantic UI (even though I only used the sidebar from it).