Develop Chat Room Application with Node.js and Socket.IO

Ade Yusuf Rahardian - 13514079
Informatics Undergraduate Program
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Jl. Ganesha 10 Bandung 40132, Indonesia
13514079@std.stei.itb.ac.id

***Abstract*—Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js becomes famous because its asynchrounous property and very fast in code execution. Socket.IO enables real-time bidirectional event-based communication. By using these two technologies, build realtime applications would be easy. This paper explains how chat room can be built with asynchrounous development by using Node.js and Socket.IO**

Keywords—node js; socket io; asynchronous; chat room; javascript

I. INTRODUCTION

Using popular stack technology like LAMP (Linux, Apache, MySQL, PHP) to build real-time application is very hard, because it should keep track of timestamps and poll the server for changes. Besides that, the final application would be very slow. Socket is one of the solution for real-time application, providing a two-way communication channel between a client and a server.

Nowadays, we often hear about Node.js. It's very popular among web developers as well as Internet of Things specialists. Based on http://nodejs.org, "Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js package ecosystem, npm, is the largest ecosystem of open source libraries in the world."

You can easily bring the JavaScript techniques you've learned to Node.js. Node.js is fast, event-driven, and lightweight but the greatest benefit of using Node.js is asynchronous programming.

Socket.IO is a javascript library for realtime web applications, enabling two-way communication from the server side and the client side. Some of its uses are to create chat application, screen sharing, webRTC, and many more.

The purpose of this paper is to explain how asynchrounous development using Node.js by creating a simple event-driven chat room application that uses Socket.IO to provide a layer of abstraction over WebSocket and other transports for both Node.js and client-side Javascript.

II. LITERATURE STUDY

A. *Asynchrounous Programming*

Asynchrounous programming is not about threading, it is about the form of execution. Asynchrounous programming is a form of input/output processing that permits other processing to continue before the transmission has finished. Figure 1. shows how asynchrounous programming is different from synchrounous programming.

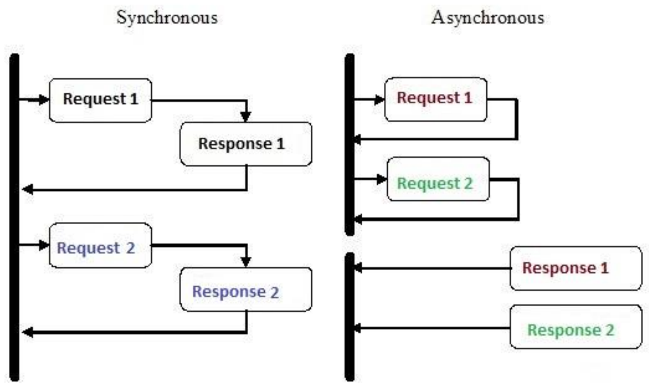


Figure 1. Synchrounous vs Asynchrounous[1]

B. *Callback Function*

In some programming languages when we do function A, we have to wait until function A is finished before we can do function B. With Node.js, it can handle this situation differently. It will do the same things and later when it is done, it will call the callback function. This callback function will see if A's task has been completed or not, and it won't wait for task A to finish to do task B. In other words, Node.js can handle multiple requests simultaneously.

III. THE PROPOSED METHOD

A. *Serving HTTP and WebSocket*

HTTP will be used to deliver the client-side things (HTML, CSS, and client-side Javascript). It's needed to set things up in the user's browser. Node can easily handle simultaneously handling HTTP and WebSocket within a single application, illustration can be seen at Figure 2.

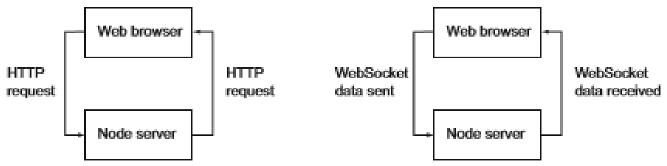


Figure 2. Serving HTTP and WebSocket using a single TCP/IP port[2]

From Figure 2., the left figure happens when user arrives at the chat application website. Meanwhile, the right figure happens repeteadly while user chats.

B. *Application File Structure*

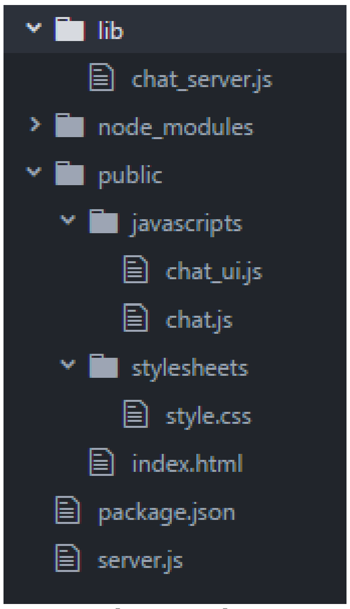


Figure 3. Application File Structure

In this project (see Appendix A), the main application file will go directly in the root of directory. Server-side logic (*chat_server.js*) will be placed in the *lib* subdirectory. Client-side logic will be placed in the *public* subdirectory. Within public subdirectory, there are *javascripts* and *stylesheets* subdirectory. All application's dependencies will be placed in *node_modules* subdirectory.

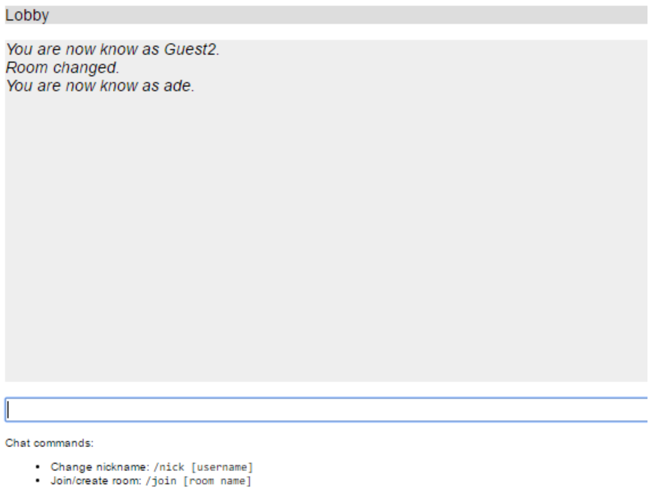
C. *Application Events and Scenarios*

There are some application events and scenarios that should be handled by helper function in this application. These are types of scenarios and events:

1. Guest name assignment
   When a user first connects to the server, the user needs a name to distinguish them from other member.
2. Room-change requests
   With this helper function, user should be able to know in which room they are and let them know what other users are in the room and let other users know that the user join in the room.
3. Name-change requests
   In this application, user is allowed to request a name-change.
4. Sending chat messages
   When a user sends a chat, the user will emit an event indicating the room where the message will be sent and the chat text. After that, the server will broadcast the message to all users in that room.
5. Room creation
   With this helper function, user is allowed to join an existing room or create a room (if the room doesn't exist)
6. User disconnection.
   In this application, user will be removed from server's data if the user leaves the chat application.

IV. EXPERIMENT RESULTS

To start the main application file, we need to run *server.js* in the root of directory by typing `node server.js` in the command prompt. When starting the application, a user is automatically assigned a guest name, but they can change it by entering a command.



Lobby

```
You are now know as Guest2.
Room changed.
You are now know as ade.
```

Chat commands:

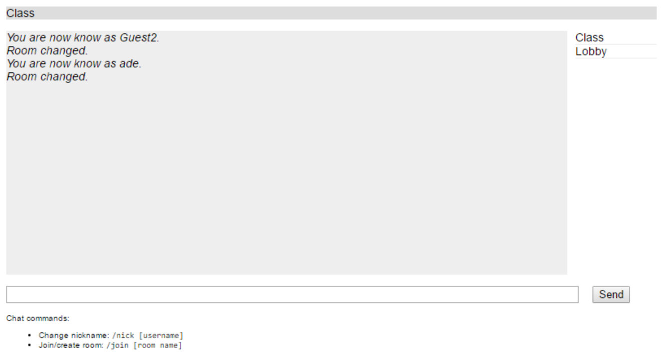
- Change nickname: /nick [username]
- Join/create room: /join [room name]

Figure 4. Starting the Application

There are two commands:

1. `/nick [username]` to change current username
2. `/join [room name]` to join spesific room (if exists) or create a room (if the room doesn't exist).

When joining or creating a room, the new room name will be shown in the horizontal bar at the top of the chat application. The room will also be included in the list of available rooms to the right of the chat message area.



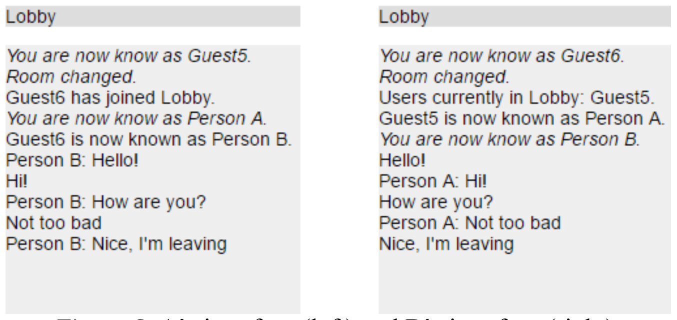
```
Class

You are now know as Guest2.
Room changed.
You are now know as ade.
Room changed.
```

Class
Lobby
Send
Chat commands:
Change nickname: /nick [username]
Join/create room: /join [room name]

Figure 5. Overall Application Interface

Figure 6. shows how the user interface is shown for person A interacting with person B and vice versa.



```
Lobby

You are now know as Guest5.
Room changed.
Guest6 has joined Lobby.
You are now know as Person A.
Guest6 is now known as Person B.
Person B: Hello!
Hi!
Person B: How are you?
Not too bad
Person B: Nice, I'm leaving
```

```
Lobby

You are now know as Guest6.
Room changed.
Users currently in Lobby: Guest5.
Guest5 is now known as Person A.
You are now know as Person B.
Hello!
Person A: Hi!
How are you?
Person A: Not too bad
Nice, I'm leaving
```

Figure 6. A's interface (left) and B's interface (right)

V. CONCLUSION

The author have performed some experiments and the results was impressive. The application shows how Node can simultaneously serve conventional HTTP data (like static files) and real-time data (chat messages) quickly. It also shows how Node applications can be organized easily.

VI. APPENDIx

Appendix A – Chat Room with Node.js and Socket.IO
Source code can be seen at
https://github.com/adeyura/multiroom-chat

VII. ACKNOWLEDGEMENT

The author would like to express his gratitude to Mr. Rinaldi Munir, Mrs. Ayu Purwarianti, and Mrs. Dessi Puji Lestari for their guidances in Socio-Informatics and Professionalism course. The author would like to thank his friend who supported him to finish this paper. Last but not least, much gratitude is given to all of those who helped and supported the author either directly or indirectly.

REFERENCES

[1] http://blogs.quovantis.com/wp-content/uploads/2015/08/Synchronous-vs.-asynchronous.jpg, accessed on May 4th 2017.

[2] Cantelon, M., Harter, M., Holowaychuk, TJ., Rajlich, N. (2014). *Node.js IN ACTION*. Shelter Island, New York: Mannings.

DECLARATION

I hereby declare that this paper is my own, not an adaption or a translation of someone else's paper, and not plagiarism.

Bandung, May 5th 2017

[illegible]

Ade Yusuf Rahardian (13514079)