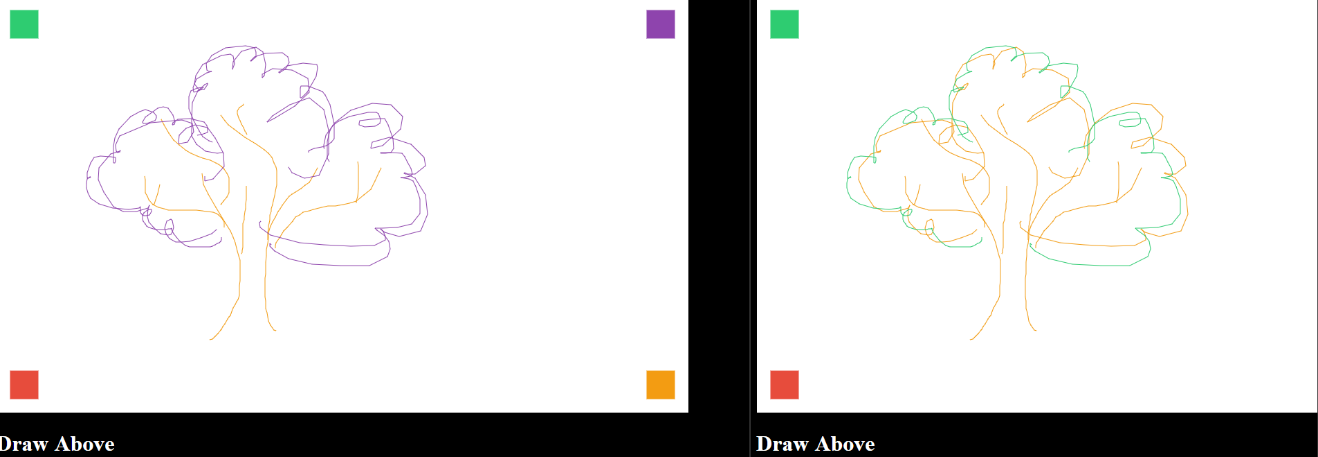


WebSockets

N00146933 | Networks & Distributed Systems | 03/04/16

# Introduction

The presented project is a socket IO whiteboard that allows multiple users to connect to a server and draw in various colors. The initial idea was to create a pong game but due to time issues, the game concept was then changed to a real time interactive whiteboard. The whiteboard still has issues with color change but other than that everything is functional.



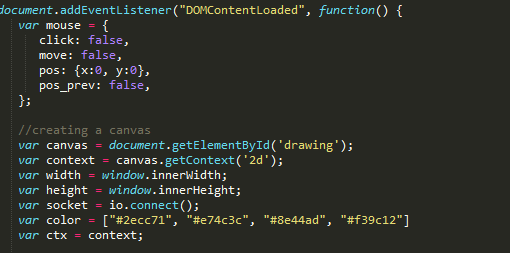
# Implementation

A basic html file was created to hold the canvas element. The html file linked all the js and css files. A css file was created to add simple style to the canvas page to make it more pleasing to the eye.

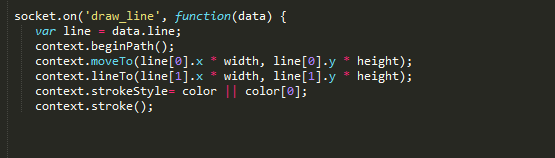
A server.js file was created, this file is the main server and is responsible for running on node. Since node was already installed on the machine there was no need to go through the setup process. Node command prompt is run and the server file is run after installing nodemon. The cmd line was nodemon server.js in the directory that the server is on.



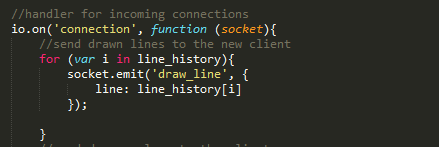
After the server was run and the console log is displayed the client js file was created. This file contains all the code that affects the canvas and the user’s interaction with it. The event listener DOMContentLoader was created which is responsible for the mouse variable. The canvas element was created and the client was connected to the server.



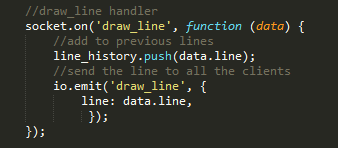
Mouse event listeners were created which check the mouse movement and actions.

A socket.on is used to listen for messages form the server. The message that this line listens to is draw\_line from the server.js file. Each line is drawn and sent to the server.

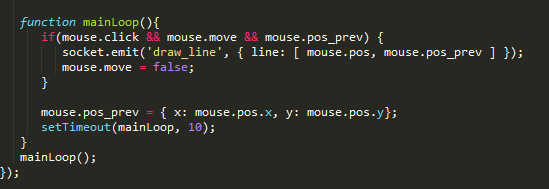
io.on('connection', function (socket) { was added to the server file which checks whenever a new client connects to the server. In this handler all the lines from a lines history variable are sent to the new client so that when they join they can still see what was drawn prior to their connection.



A draw-line handler is created for the client. Every time a line is added to the line \_history its sent to all the clients connected to the server, to be displayed on their individual canvas.



Finally in the client.js a main loop function is created that checks if the mouse is clicked every 10 milliseconds, where the mouse was and where it currently is. If the mouse was clicked and moved from one location to another draw\_line is sent to the server and the mouse movement is reset to false when it stops.



An attempt was made to make lines of different colors but dude to time the color change was never implemented on the server.js file.

# Conclusion

The created app works as it should with a bit of functionality missing. Two clients can connect to the server and the drawing appears on both canvases due to constant update of the line\_history array in the server.js file.

The code was created by following a tutorial on:

http://code-and.coffee/post/2015/collaborative-drawing-canvas-node-websocket/