1. A WebSocket Server is a TCP application listening on any port of the server.
2. This provides for an efficient communication “channel” between clients and the server. The WebSocket Server can be written using any server-side language.
3. Since Node.js provides server side processing for JavaScript, we can use it for Socket Programming.
4. In Node.js we have the **ws** module for creating WebSocket Server using code. This Module provides events and functions for opening, sending, receiving and closing messages.

**WebSocket Server Implementation**

1. We will make use of Visual Studio Code as our editor.
2. Create a folder of the name **VSCodeWebSocketServer** on your hard drive. We will use this folder for storing our application files.
3. Open Visual Studio Code. Click on File > Open Folder and navigate to the **VSCodeWebSocketServer** folder. Select this folder. You should now see options for creating a new folder inside **VSCodeWebSocketServer** as shown in the following image:
4. Name this folder as **Scripts,** we will use this folder for storing our JavaScript application file. Add a new folder of the name **Pages** in the **VSCodeWebSocketServer** for storing the Html file.
5. Open the Node.js Command Prompt, (This will be available to us after installing Node.js tools). Run following commands from the command prompt:
   1. npm install -g tsd

This will install TypeScript Definition for our application so that we can install and use other Node.js modules

* 1. tsd query node --action install

This command will provide Node.js intellisense.

* 1. tsd query ws –action install

This command will install the **ws** module. This module provides APIs for creating WebSocket Server. After installing this module the following result will be displayed

1. In the Scripts folder, add a new JavaScript file of the name **app.js,** add the following code in this file.

var webSocketServer = require('ws').Server;

var http = require('http');

var fs = require('fs');

//2.

var webSocketServerObject = new webSocketServer({ port: 9060 });

//3.

webSocketServerObject.on('connection', function (socketObject) {

socketObject.on('message', function (message) {

console.log('The' + message + 'Message Received from \n from IP ' + socketObject.upgradeReq.connection.remoteAddress);

socketObject.send("Received " + message);

});

socketObject.on('close', function (c, d) {

console.log('Disconnect ' + c + ' -- ' + d);

});

});

//4.

var server = http.createServer(function (req, resp) {

fs.readFile("../Pages/Client.html", function (error, pgResp) {

if (error) {

resp.writeHead(404);

resp.write('Contents you are looking are Not Found');

} else {

resp.writeHead(200, { 'Content-Type': 'text/html' });

resp.end(pgResp);

}

});

});

//5.

server.listen(5050);

console.log('Server started');

7. The above code has the following specifications (Note: Comment numbers applying on code matches with the following numbering)

1. Load modules for creating **WebSocket** Server, **WebServer**, and the File IO.

2. Define an instance of the WebSocket and register the port 9060 for it.

3. Register the **connection** event which will be used by the WebSocket server for managing messaging through it. This event provides Socket object, using which Messages can be used for processing. The **Send ()** function is used to send message through the socket. The **close** event is used to disconnect the messaging.

4. Create a web server to read the Html file from the **Pages** folder and respond to the request made by the client.

5. Start listening on the port 5050.

8. In the Pages folder, add a new file of the name client.html. This file will contain HTML markup and HTML5 WebSocket object to manage the communication with the WebSocket Server created in the previous step. Add the following code (HTML and JavaScript) in the html page

1. Create an instance of the WebSocket by defining the URL for the Port on which WebSocket Server is ready to accept incoming messages. In our case web have use port 8080.

2. The click event for the button calls the **send()** function of the WebSocket object and sends the message entered in the textbox to the server.

3. **Onmessage** event is used to listen to the message sent by socket server.

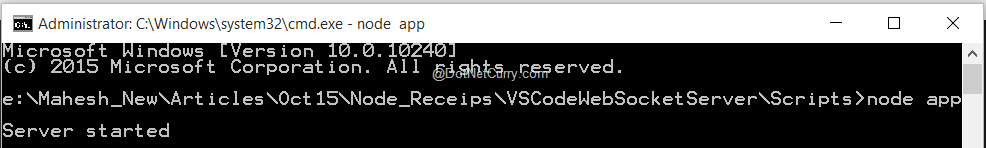
4. **Onclose** event is used to listen to the connection close event.

5. **Onerror** is used to take necessary action when an error occurs during messaging.

**9.** To execute the application, right click on app.js in Visual Studio Code and select the option **Open in Command Prompt.** This will open the command prompt. Add the following command in the Command Prompt

node app.js

The following result will be displayed:

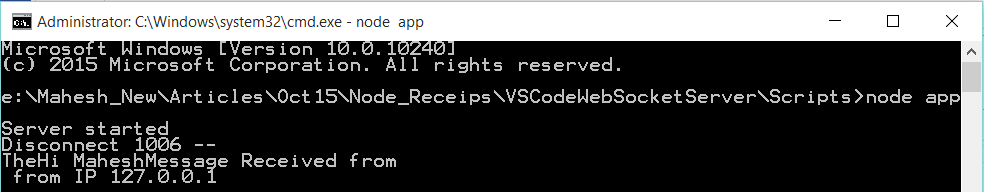


Open any browser and enter the following address in the address bar:

<http://localhost:5050/Pages/Client.html>







<http://www.dotnetcurry.com/nodejs/1220/create-web-socket-server-nodejs-for-real-time>

