Reviewing the high-level architecture of the chat application

Setting up the client and server code projects

Creating a quick, basic build system for my projects

Chat Application

Client side

- Runs in browser
  - HTML,CSS
  - Typescript
  - WebSockets

Server side

- Runs in Node.js
- Listens via WebSockets

Client > package.json

```
{
  "name": "sample",
  "version": "1.0.0",
  "description": "A sample game.",
  "main": "index.js",
  "scripts": {
    "build": "node build.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Taesik",
  "license": "ISC",
  "devDependencies": {
    "fs-extra": "^8.1.0",
    "typescript": "^3.3.3"
  },
  "dependencies": {
    "@types/websocket": "0.0.40",
    "websocket": "^1.0.28"
  }
}
```

Client > tsconfig.json

```
{
  "compilerOptions": {
    /* Basic Options */
    "target": "es5", /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017','ES2018' or 'ESNEXT'. */
    "module": "amd", /* Specify module code generation: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015', or 'ESNext'. */
    // "lib": [],                          /* Specify library files to be included in the compilation. */
    // "allowJs": true,                    /* Allow javascript files to be compiled. */
    // "checkJs": true,                    /* Report errors in .js files. */
    // "jsx": "preserve",                  /* Specify JSX code generation: 'preserve', 'react-native', or 'react'. */
    "declaration": false, /* Generates corresponding '.d.ts' file. */
    "declarationMap": false, /* Generates a sourcemap for each corresponding '.d.ts' file. */
    "sourceMap": false, /* Generates corresponding '.map' file. */
    "outFile": "./dist/index.js", /* Concatenate and emit output to single file. */
    "outDir": "./dist", /* Redirect output structure to the directory. */
    // "rootDir": "./",                    /* Specify the root directory of input files. Use to control the output directory structure with --outDir. */
    // "composite": true,                  /* Enable project compilation */
    // "removeComments": true,             /* Do not emit comments to output. */
    // "noEmit": true,                     /* Do not emit outputs. */
    // "importHelpers": true,              /* Import emit helpers from 'tslib'. */
    // "downlevelIteration": true,         /* Provide full support for iterables in 'for-of', spread, and destructuring when targeting 'ES5' or 'E
    // "isolatedModules": true,            /* Transpile each file as a separate module (similar to 'ts.transpileModule'). */
    /* Strict Type-Checking Options */
```

```
"strict": true, /* Enable all strict type-checking options. */
"noImplicitAny": false, /* Raise error on expressions and declarations with an implied 'any' type. */
"strictNullChecks": false, /* Enable strict null checks. */
// "strictFunctionTypes": true,           /* Enable strict checking of function types. */
"strictPropertyInitialization": false, /* Enable strict checking of property initialization in classes. */
// "noImplicitThis": true,                /* Raise error on 'this' expressions with an implied 'any' type. */
// "alwaysStrict": true,                  /* Parse in strict mode and emit "use strict" for each source file. */
/* Additional Checks */
// "noUnusedLocals": true,                /* Report errors on unused locals. */
// "noUnusedParameters": true,            /* Report errors on unused parameters. */
// "noImplicitReturns": true,             /* Report error when not all code paths in function return a value. */
// "noFallthroughCasesInSwitch": true,    /* Report errors for fallthrough cases in switch statement. */
/* Module Resolution Options */
// "moduleResolution": "node",            /* Specify module resolution strategy: 'node' (Node.js) or 'classic' (TypeScript pre-1.6). */
// "baseUrl": "./",                       /* Base directory to resolve non-absolute module names. */
// "paths": {},                           /* A series of entries which re-map imports to lookup locations relative to the 'baseUrl'. */
// "rootDirs": [],                        /* List of root folders whose combined content represents the structure of the project at runtime. */
// "typeRoots": [],                       /* List of folders to include type definitions from. */
// "types": [],                           /* Type declaration files to be included in compilation. */
// "allowSyntheticDefaultImports": true,  /* Allow default imports from modules with no default export. This does not affect code emit, jus
"esModuleInterop": true /* Enables emit interoperability between CommonJS and ES Modules via creation of namespace objects for al
// "preserveSymlinks": true,              /* Do not resolve the real path of symlinks. */
/* Source Map Options */
```

```
// "sourceRoot": "",                    /* Specify the location where debugger should locate TypeScript files instead of source locations. */
    // "mapRoot": "",                      /* Specify the location where debugger should locate map files instead of generated locations. */
    // "inlineSourceMap": true,               /* Emit a single file with source maps instead of having a separate file. */
    // "inlineSources": true,                /* Emit the source alongside the sourcemaps within a single file; requires '--inlineSourceMap' or '--
    /* Experimental Options */
    // "experimentalDecorators": true,        /* Enables experimental support for ES7 decorators. */
    // "emitDecoratorMetadata": true,         /* Enables experimental support for emitting type metadata for decorators. */
  },
  "include": [
    "./src/**/*.ts"
  ],
  "exclude": [
    "node_modules",
    "./**/*.d.ts"
  ]
}
```

Server

.vscode > launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "Launch Program",
            "program": "${workspaceFolder}\\dist\\app.js",
            "outFiles": [
                "${workspaceFolder}/dist/**/*.js"
            ]
        }
    ]
}
```

Server

> package.json

```
{
  "name": "sample-server",
  "version": "1.0.0",
  "description": "A sample server.",
  "main": "index.js",
  "scripts": {
    "build": "node build.js",
    "local": "node dist/app.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Travis Vroman",
  "license": "ISC",
  "devDependencies": {
    "fs-extra": "^8.0.1",
    "typescript": "^3.3.3"
  },
  "dependencies": {
    "@types/websocket": "0.0.40",
    "websocket": "^1.0.28"
  }
}
```

Server

> tsconfig.json

```
{
  "compilerOptions": {
    /* Basic Options */
    "target": "es5",                          /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017','ES2018' 
    "module": "commonjs",                     /* Specify module code generation: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015', or 'ESN
    // "lib": [],                             /* Specify library files to be included in the compilation. */
    // "allowJs": true,                       /* Allow javascript files to be compiled. */
    // "checkJs": true,                       /* Report errors in .js files. */
    // "jsx": "preserve",                     /* Specify JSX code generation: 'preserve', 'react-native', or 'react'. */
     "declaration": true,                    /* Generates corresponding '.d.ts' file. */
     "declarationMap": true,                 /* Generates a sourcemap for each corresponding '.d.ts' file. */
     "sourceMap": true,                      /* Generates corresponding '.map' file. */
    // "outFile": "./dist/index.js",          /* Concatenate and emit output to single file. */
     "outDir": "./dist",                     /* Redirect output structure to the directory. */
    // "rootDir": "./",                       /* Specify the root directory of input files. Use to control the output directory structure with --outDir. */
    // "composite": true,                     /* Enable project compilation */
    // "removeComments": true,                /* Do not emit comments to output. */
    // "noEmit": true,                        /* Do not emit outputs. */
    // "importHelpers": true,                 /* Import emit helpers from 'tslib'. */
    // "downlevelIteration": true,            /* Provide full support for iterables in 'for-of', spread, and destructuring when targeting 'ES5' or '
    // "isolatedModules": true,               /* Transpile each file as a separate module (similar to 'ts.transpileModule'). */
```

```
/* Strict Type-Checking Options */
"strict": true,                           /* Enable all strict type-checking options. */
// "noImplicitAny": true,                 /* Raise error on expressions and declarations with an implied 'any' type. */
 "strictNullChecks": false,               /* Enable strict null checks. */
// "strictFunctionTypes": true,           /* Enable strict checking of function types. */
 "strictPropertyInitialization": false,  /* Enable strict checking of property initialization in classes. */
// "noImplicitThis": true,                /* Raise error on 'this' expressions with an implied 'any' type. */
// "alwaysStrict": true,                  /* Parse in strict mode and emit "use strict" for each source file. */

/* Additional Checks */
// "noUnusedLocals": true,                /* Report errors on unused locals. */
// "noUnusedParameters": true,            /* Report errors on unused parameters. */
// "noImplicitReturns": true,             /* Report error when not all code paths in function return a value. */
// "noFallthroughCasesInSwitch": true,    /* Report errors for fallthrough cases in switch statement. */

/* Module Resolution Options */
// "moduleResolution": "node",            /* Specify module resolution strategy: 'node' (Node.js) or 'classic' (TypeScript pre-1.6). */
// "baseUrl": "./",                       /* Base directory to resolve non-absolute module names. */
// "paths": {},                           /* A series of entries which re-map imports to lookup locations relative to the 'baseUrl'. */
// "rootDirs": [],                        /* List of root folders whose combined content represents the structure of the project at runtime. */
// "typeRoots": [],                       /* List of folders to include type definitions from. */
// "types": [],                           /* Type declaration files to be included in compilation. */
// "allowSyntheticDefaultImports": true,  /* Allow default imports from modules with no default export. This does not affect code emit, jus
"esModuleInterop": true                   /* Enables emit interoperability between CommonJS and ES Modules via creation of namespace o
// "preserveSymlinks": true,              /* Do not resolve the real path of symlinks. */
```

```
/* Source Map Options */
    // "sourceRoot": "",                    /* Specify the location where debugger should locate TypeScript files instead of source locations. */
    // "mapRoot": "",                       /* Specify the location where debugger should locate map files instead of generated locations. */
    // "inlineSourceMap": true,                /* Emit a single file with source maps instead of having a separate file. */
    // "inlineSources": true,                /* Emit the source alongside the sourcemaps within a single file; requires '--inlineSourceMap' or '--

    /* Experimental Options */
    // "experimentalDecorators": true,        /* Enables experimental support for ES7 decorators. */
    // "emitDecoratorMetadata": true,         /* Enables experimental support for emitting type metadata for decorators. */
  },
  "include": [
    "./src/**/*.ts"
  ],
  "exclude": [
    "node_modules",
    "./**/*.d.ts"
  ]
}
```

Creating the client (Frontend) as a Web Page

Src > packets.ts

```
namespace Chat {
    export enum RequestType {
        LOGIN = "login",
        CHAT_SEND = "chat_send"
    }
    export enum ResponseType {
        LOGIN_RESULT = "login_result",
        CHAT_RESULT = "chat_result",
    }
    /**It deals with what kind of packet **/
    export interface IResponsePacket {
        readonly responseType: string;
    }
    export class ChatResponse implements IResponsePacket {
        public readonly responseType:string;
        public constructor(public userData:UserData, public content:string) {
            this.responseType = ResponseType.CHAT_RESULT
        }
    }
    export class LoginResponse implements IResponsePacket {
        public readonly responseType:string;
        public constructor(public success:boolean ) {
            this.responseType = ResponseType.LOGIN_RESULT;
        }
    }

}
```

Src > userData.ts

```
namespace Chat {

    export class UserData {

        public constructor(public userId: number, public name: string,public color:string) {
        }
        public static readonly SERVER:UserData = new UserData(-1, "SERVER","yellow");
    }
}
```

Src > serverConnection.ts

```
namespace Chat {
    /** Handles a connection to the chat server**/
    export class ServerConnection {
        private _connection:WebSocket;

        public constructor(public readonly host:string,
                           public readonly port:number,
                           private _connectionCallback:()=>void,
                           private _packetResponseCallback:(packet:IResponsePacket)=>void,
                           private _connectionCloseCallback: () => void) {
        }
        public connect():void {
            this._connection = new WebSocket(`ws://${this.host}:${this.port}`);
            this._connection.addEventListener("open",this._connectionCallback.bind(this));
            this._connection.addEventListener("error",()=>alert("unable to connect to server"));
            this._connection.addEventListener("message",this.onMessage.bind(this));
            this._connection.addEventListener("close",this._connectionCloseCallback.bind(this));
        }
        public login(username:string):void {
            let request = {
                requestType: RequestType.LOGIN,
                username:username,
            }
            this._connection.send(JSON.stringify(request));
        }
```

```
public disconnect():void {
            this._connection.close(1000,"User disconnect");
        }
        public send(request:any):void {
            this._connection.send(JSON.stringify(request));
        }

        private onMessage( message:MessageEvent):void {
            this._packetResponseCallback(JSON.parse(message.data) as IResponsePacket);
        }
    }
}
```

Src >documentHelper.ts

```
namespace Chat {
    export class DocumentHelper {

        public static initialize():void {
                DocumentHelper.getInputElement("username").addEventListener("keyup",DocumentHelper.userKeyup);
                DocumentHelper.getInputElement("entry").addEventListener("keyup",DocumentHelper.entryKeyup);
        }
        public static toggleLogin(showLogin:boolean):void {
            DocumentHelper.toggleForm("JoinDiv",showLogin);
            DocumentHelper.toggleForm("inputWrapper",!showLogin);
        }
        public static getInputElement(id:string):HTMLInputElement {
            return (document.getElementById(id) as HTMLInputElement);
        }
        public static userKeyup(event: KeyboardEvent):void {
            let value = DocumentHelper.getInputElement("username").value;
            let valueEmpty = value ==="";
            DocumentHelper.getInputElement("loginbutton").disabled = valueEmpty;
            /**if enter is pressed**/
            if(event.keyCode ===13) {
                if(valueEmpty) alert("A username is required.");
                else App.connect();
            }
        }
```

```
public static entryKeyup(event:KeyboardEvent):void {
    let value = DocumentHelper.getInputElement("entry").value;
    let valueEmpty = value ==="";
    DocumentHelper.getInputElement("sendbutton").disabled = valueEmpty;

    if (event.keyCode ===13 && !valueEmpty) App.sendChat();
}

public static appendMessage(userData:UserData, content:string):void {
    if(content.trim() !== "") {
        let wrapper = document.createElement("div");
        wrapper.className = "messageWrapper";
        let userSection = document.createElement("span");
        userSection.style.color=userData.color;
        userSection.innerHTML = `[${userData.name}]: `;
        let contentSection = document.createElement("span");
        contentSection.innerHTML=content;
        wrapper.appendChild(userSection);
        wrapper.appendChild(contentSection);
        let log =document.getElementById("chatlog");
        log.appendChild(wrapper);
        //Make sure to always scroll to the bottom.
        log.scrollTo(0,log.clientHeight);
    }
}
```

```
private static toggleForm(id:string, value: boolean):void {
        document.getElementById(id).style.display=value? "block":"none";
    }
  }
}
```

Src >index.ts

```
namespace Chat {
    export class App {
        private static _serverConnection:ServerConnection;

        public static  start():void {
            //Prepare a server connection.
            App._serverConnection = new ServerConnection(
                "127.0.0.1",
                5001,
                App.onConnected,
                App.processPacket,
                App.onDisconnected,
            );
            DocumentHelper.initialize();
        }
        public static connect():void {
            App._serverConnection.connect();
        }
        public static disconnect():void {
            App._serverConnection.disconnect();
            DocumentHelper.toggleLogin(true);

            //Clear the user box.
            DocumentHelper.getInputElement("username").value="";
        }
```

```
/** Sends the content in the entry box as a chat message. **/
public static sendChat(): void {
    let input = DocumentHelper.getInputElement("entry");
    let text = input.value;
    input.value = "";

    /** Special cases for commands **/
    if(text[0] == "/") {
        let command = text.substr(1).toLowerCase();
        if(command === "logout" || command === "disconnect" || command ==="quit" ) {
            App.disconnect();
        }
        return ;
    }

    let request = {
        requestType:RequestType.CHAT_SEND,
        message:text
    };
    this._serverConnection.send(request);

}
```

```
private static onConnected():void {
        let username = DocumentHelper.getInputElement("username").value;
        //Attempt login
        DocumentHelper.toggleLogin(username === undefined);
        App._serverConnection.login(username);
    }
    private static onDisconnected():void {
        DocumentHelper.toggleLogin(true);
        DocumentHelper.appendMessage(UserData.SERVER, "You have benn disconnected from the chat server");
    }
    private static processPacket(packet:IResponsePacket):void {
        switch (packet.responseType) {
            case ResponseType.CHAT_RESULT:
                //Add message to the log.
                let response = packet as ChatResponse;
                DocumentHelper.appendMessage(response.userData,response.content);
                break;
            case ResponseType.LOGIN_RESULT:
                let result = packet as LoginResponse;
                if(result.success) DocumentHelper.appendMessage(UserData.SERVER,"Welcome to the chat server");
                DocumentHelper.toggleLogin(!result.success);
                break;
        }
    }
  }
}
```

```
/** Application entry point**/
window.onload = () => {
    Chat.App.start();
}
```

```
PS C:\9781789619423_Code\Section 1\chat_application\client\dist> Set-ExecutionPolicy -Scope CurrentUser

cmdlet Set-ExecutionPolicy(명령 파이프라인 위치 1)
다음 매개 변수에 대한 값을 제공하십시오.
ExecutionPolicy: Unrestricted
PS C:\9781789619423_Code\Section 1\chat_application\client\dist>
PS C:\9781789619423_Code\Section 1\chat_application\client\dist> http-server -c -o
```

```
PS C:\9781789619423_Code\Section 1\chat_application\client\dist> http-server -c -o
Starting up http-server, serving ./

http-server version: 14.0.0

http-server settings:
CORS: disabled
Cache: true seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://192.168.56.1:8080
  http://192.168.0.102:8080
  http://127.0.0.1:8080
Hit CTRL-C to stop the server
Open: http://127.0.0.1:8080
```

Creating the Server (Backend)

Src>users.ts

```
export class UserData {
    private static GLOBAL_USER_ID: number = 0;

    public userID: number;

    public constructor(public name: string, public color?: string) {
        this.userID = UserData.GLOBAL_USER_ID++;
        this.color = color ? color : this.generateRandomColor();
    }
    /** return HTML style hexcode **/
    private generateRandomColor():string {
        return '#' + (Math.floor(Math.random()*(0xFFFFFF - 0x666666 +1) ) +0x666666).toString(16);
    }

    /** The static user data for server communications. **/
    public static readonly SERVER:UserData = new UserData("SERVER","yellow");


}
```

```
export class UserManager {
    private static _users: { [name: string]: UserData} = {}; // hashTable Key:name value:UserData

    public static getUserDataForId(id: number):UserData {
        let users = Object.values(UserManager._users).map(x=>x).filter(x=>x.userID === id);
        return users[0] ? users[0] : undefined;
    }

    public static tryAuthenticateUser(request:any):number {
        let username = request.username.toLowerCase();
        let record = UserManager._users[username];
        if(record === undefined) {
            /** If not defined, create a new record**/
            record = new UserData(username);
            UserManager._users[username] = record;
        }
        return record.userID;
    }
}
```

Src>responsePackets.ts

```
export enum ResponseType {
    LOGIN_RESULT = "login_result",
    LOGOUT_RESULT = "logout_result",
    CHAT_RESULT = "chat_result"
}
```

Src>requestPackets.ts

```
export enum RequestType {
    LOGIN = "login",
    CHAT_SEND = "chat_send",
}
```

Src>clientConnection.ts

```
import WebSocket from "websocket"; import {ResponseType} from
"./responsePackets"; import {UserManager} from "./users"; import
{RequestType} from "./requestPackets"; import {Server} from "./app";
/**Represents a single client connection.**/
export class ClientConnection {
    private static _GLOBAL_CONNECTION_ID: number = 0;
    private _server: Server;
    private _connection: WebSocket.connection;
    private _connectionID: number;
    private _userId:number = -1;
```

```
public constructor(server:Server, request:WebSocket.request) {
    this._server = server;
    // Set the connection id and increment the global counter.
    this._connectionID = ClientConnection._GLOBAL_CONNECTION_ID++;
    //Create the connection
    this._connection = request.accept(null, request.origin);
    //Setup event handlers.
    this._connection.on("message",this.onMessage.bind(this));

    this._connection.on("close", (code:number,description:string) =>{
        console.log(`Connection closed (${this._connectionID}):`,closed);
        this._server.onClientDisconnected(this);
    });
    this._connection.on("error",(error:Error)=>{
        console.log("Error:",error);
    });
    this._connection.on("error", (error:Error)=> {
        console.log("Error:",error);
    });
}
```

```
public get connectionID():number {
    return this._connectionID;
}
public get userId():number {
    return this._userId;
}
public send(response:any):void {
    this._connection.sendUTF(JSON.stringify(response));
}

public disconnect():void {
    //Tell the server about the disconnection, then actually close
    this._server.onClientDisconnected(this);
    this._connection.close();
}
```

```
private onMessage(message: WebSocket.IMessage):void {
    console.debug(message);

    //This only accepts text.
    if(message.type ===“utf8") {
        let packet = JSON.parse(message.utf8Data);
        if(this._userId ===-1) {
            if(packet.responseType !== RequestType.LOGIN) this.respondFail();
            else {
                if(this.tryAuth(packet)) this.respondSuccess();
                else this.respondFail();
            }
        }else {
            //User is authenticated and sending a chat. Broadcast is
            //Otherwise it is invalid - disconnect the user.
            if(packet.responseType === RequestType.CHAT_SEND) {
                let response = {
                    responseType:ResponseType.CHAT_RESULT,
                    content:packet.message,
                    userData:UserManager.getUserDataForId(this.userId)
                };
                Server.broadcast(response);
            }else this.respondFail();
        }
    }
}
```

```
private respondSuccess():void {
        this.send({responseType: ResponseType.LOGIN_RESULT,success:true});

        //Notify other users this one has logged in.
        this._server.onClientAuthenticated(this);
    }
    private respondFail():void {
        this.send({responseType:ResponseType.LOGIN_RESULT,success:false});
        this.disconnect();
    }

    private tryAuth(packet:any):boolean {
        this._userId=UserManager.tryAuthenticateUser(packet);
        return this._userId !== -1;
    }

}
```

Src>app.ts

```
import {ClientConnection} from "./clientConnection";
import WebSocket from "websocket";
import Http from "http";
import {UserData, UserManager} from "./users";
import {ResponseType} from "./responsePackets";

export class Server {
    //Hold all connections.
    private static _connections:{[connectionID:number]:ClientConnection}={};

    public static broadcast(response:any):void {
        Object.values(Server._connections).map(x=>x).forEach(x=>x.send(response));
    }
    public constructor(public readonly port:number) {
        //Create the http server.
        let httpServer = Http.createServer(this.listenerCallback.bind(this));

        //Create the websocket server and pass the http server to it.
        let wsServer = new WebSocket.server({httpServer:httpServer})
    }
    public static getConnectedUserIds():number[] {
        return Object.values(Server._connections).map(x=>x.userId).filter(x=>x !==-1);
    }
```

```
public onClientDisconnected(client:ClientConnection):void {
    //If client was authenticated, tell all the other clients about
    if(client.userId !== -1) {
        console.log("User disconnected: "+client.userId);
        let user = UserManager.getUserDataForId(client.userId);
        let message = (user? user.name : "Someone" ) + " has gone offline";
        let response = {
            responseType: ResponseType.CHAT_RESULT,
            userData: UserData.SERVER,

        };
        Server.broadcast(response);
    }
    console.log("Connection id disconnected: "+ client.connectionID);
    Server._connections[client.connectionID] = undefined;
    delete Server._connections[client.connectionID];
}
```

```
public onClientAuthenticated(client:ClientConnection):void {
    //If the client is authenticated, thell all the other clients about the con
    if(client.userId !==-1) {
        console.log("User signed on: "+ client.userId);
        let user = UserManager.getUserDataForId(client.userId);
        let message = (user?user.name:"Someone")+ " has come online.";
        let response = {
            responseType: ResponseType.CHAT_RESULT,
            userData:UserData.SERVER,
            content:message
        };
        Server.broadcast(response);
    }
}

private listenerCallback(message:Http.IncomingMessage, response:Http.ServerResponse):void {
    //Handle HTTP requests. This will be useful for registration.
}
private requestCallback(): void {
    console.log((new Date() )+ " - Server started and listening on port: " + this.port);
}
```

```
private onWebSocketRequest(request:WebSocket.request):void {
        console.log((new Date())+ " - New client connection:" +request.remoteAddress);
        let connection = new ClientConnection(this,request);
        Server._connections[connection.connectionID] = connection;
    }
}
//Server config
process.title = "Chat Server";
var server = new Server(5001);
```

Bringing Client and Server together

```
PS C:\9781789619423_Code\Section 1\chat_application\client\dist> http-server -c -o
Starting up http-server, serving ./

http-server version: 14.0.0

http-server settings:
CORS: disabled
Cache: true seconds
```

WebSockets Chat

WebSockets Chat

127.0.0.1:8080

동기화하지 않음

Username

Login

[SERVER]: Welcome to the chat server.
[SERVER]: testguy1 has come online.

Elements Console Sources Network

View: Group by frame Preserve log

Filter Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other

2000 ms 4000 ms 6000 ms 8000 ms 10000 ms 12000 r

Name	Status	Type	Initiator	Size	Time	Waterfall
127.0.0.1	101	web...	index.js:177	0 B	Pen...	
index.html	200	doc...	Other	1.0 ...	4 ms	
index.js	200	script	index.html	9.5 ...	8 ms	
main.css	200	styl...	index.html	1.2 ...	6 ms	

http://127.0.0.1:8080/index.html

127.0.0.1:8080/index.html

[SERVER]: Welcome to the chat server.
[SERVER]: testguy1 has come online.
[testguy1]: Hello!
[SERVER]: testguy2 has come online.

Enter text to send chat

Send

WebSockets Chat

127.0.0.1:8080/index.html

[SERVER]: Welcome to the chat server.
[SERVER]: testguy2 has come online.

Enter text to send chat

Send

Enhancing the experience with Chat attachments

Client>www>main.css

```
html, body {
    margin: 0;
    padding: 0;
    overflow: hidden;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 14pt;
    background:black;
    color:white;
}
img {
    max-width: 95%;
    margin: 5px;
}
.messageWrapper {
    display: block;
    max-width: 100%;
    min-height: 20px;
}
#chatlog {
    width:550px;
    height:300px;
    overflow:hidden;
```

```
input[type=text] {
    border:2px solid #666;
    background: #222;
    border-radius: 3px;
    outline: none;
    font-size: 14pt;
    color:white;
    width: 450px;
}
input[type=text]:active, input[type=text]:focus {
    border-color:#ff6600;
}

input[type=button] {
    border:2px solid white;
    background: #FF6600;
    border-radius: 3px;
    outline: none;
    font-size: 14pt;
    color:white;
    width: 100px;
}

input[type=button]:disabled {
    background: #666666;
    color:#ddd;
}
```

```
public static appendMessage(userData:UserData, content:string):void {
    if(content.trim() !== "") {
        let wrapper = document.createElement("div");
        wrapper.className = "messageWrapper";

        let userSection = document.createElement("span");
        userSection.style.color=userData.color;
        userSection.innerHTML = `[${userData.name}]: `;

        let contentSection = document.createElement("span");

        //[img]https://www.gstatic.com/webp/gallery/1.jpg[/img]
        //<img src="https://www.gstatic.com/webp/gallery/1.jpg"/>
        let replaceContent = content.replace(/\[img\]/g,"<img src=\"")
            .replace(/\[\/img\]/g,"\"/>");

        contentSection.innerHTML=replaceContent;
        wrapper.appendChild(userSection);
        wrapper.appendChild(contentSection);

        let log =document.getElementById("chatlog");
        log.appendChild(wrapper);

        //Make sure to always scroll to the bottom.
        log.scrollTo(0,log.clientHeight);
    }
```

change