嵌入式系統實驗

實驗一 JavaScript and web programming basics

B02901061 鄧郁璇　B02901080 董皓文

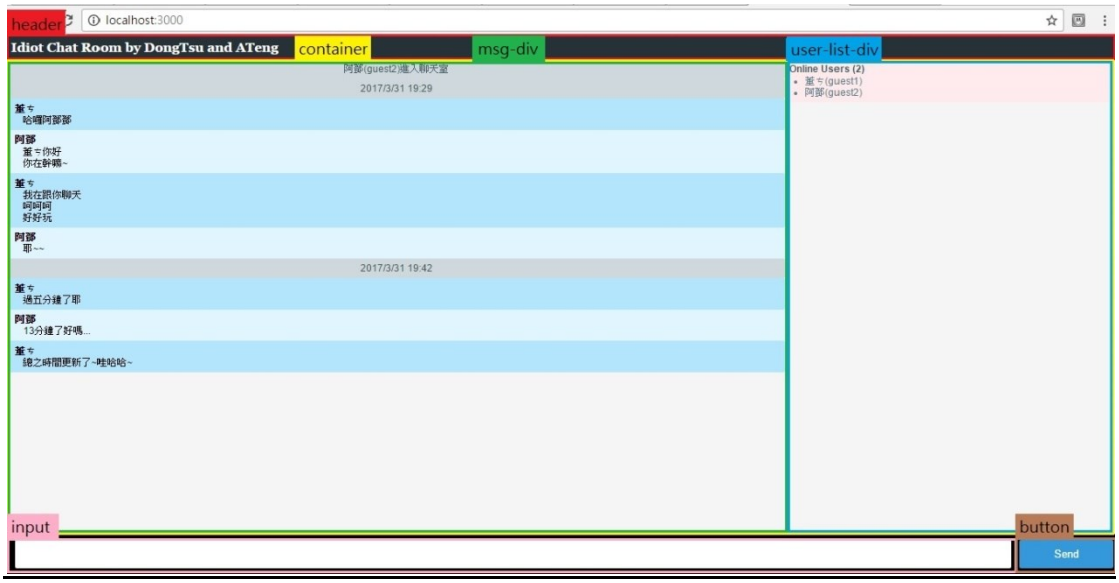
❖ 功能

- 公開聊天室
    - 所有上線用戶皆可傳送及接收訊息
- 上線用戶顯示
    - 頁面右側列出所有上線用戶名稱
- 訊息泡泡
    - 建立不同的背景顏色，以明顯區分自己與其他用戶傳送的訊息
- 系統提示
    - 有用戶上線或離線時，會在訊息列中顯示
    - 當兩則訊息傳送時間差距超過 5 分鐘，會在訊息列顯示傳送時間
- 註冊及登入系統
    - 輸入 guest 以 guest 身分登入
    - 輸入 new 以註冊新帳戶
    - 輸入已註冊帳號及密碼登入（不同 client 登入同一帳號可同步）
        - 以利未來開發歷史訊息讀取功能
        - 以及一對一對話的實現

❖ 系統架構

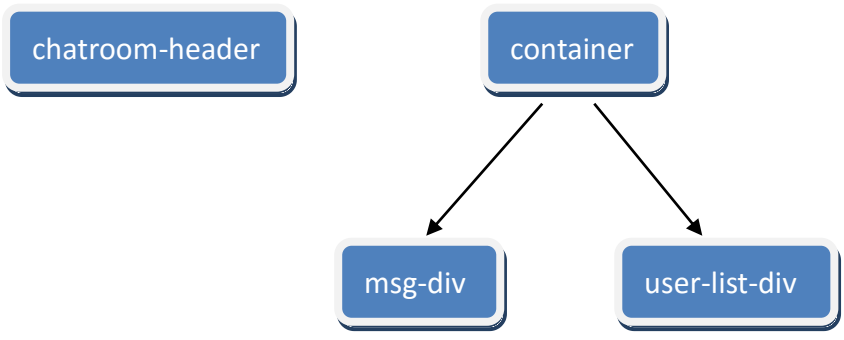
1. Client 端

聊天室介面




```
<body>
    <h3 id="chatroom-header">Idiot Chat Room by DongTsu and ATeng</h3>
    <div id="container">
        <div id="msg-div"></div>
        <div id="user-list-div">
            <h4 id="user-list-header">Online User (1)</h4>
            <ul id="user-list-ul"></ul>
        </div>
    </div>
    <form action="">
        <input id="msg-input" autocomplete="off" />
        <button id="send">Send</button>
    </form>
```

** **



msg-div：訊息顯示區

user-list-div：上線用戶列表

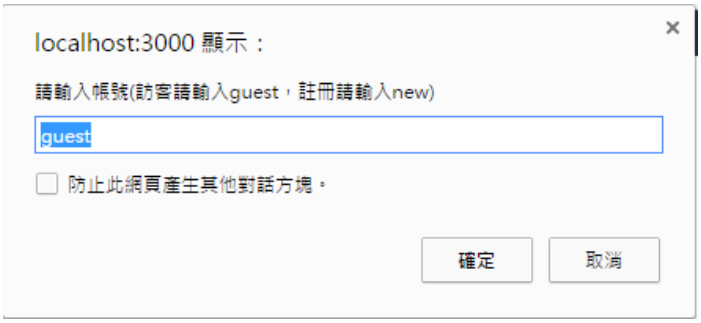
<input>

訊息輸入欄

<button>

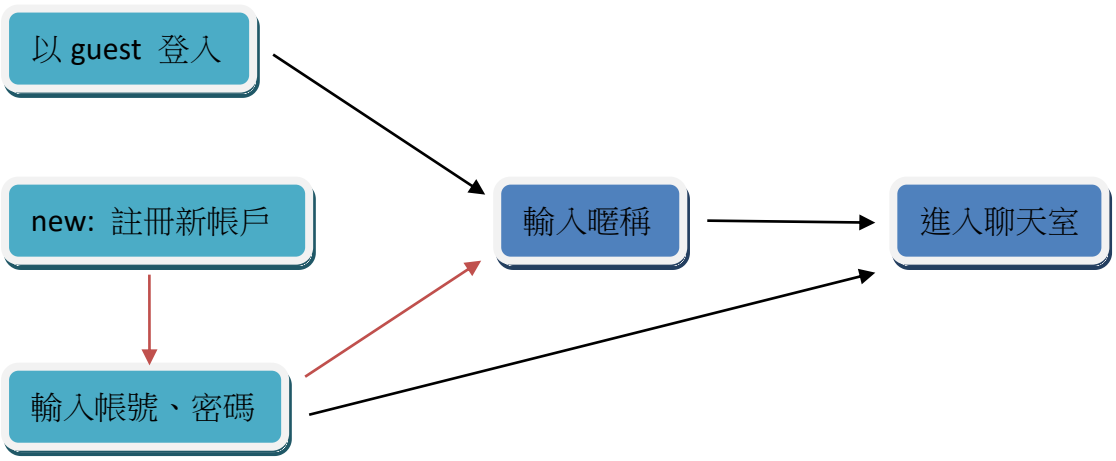
訊息傳送按鈕

註冊及登入系統



使用 prompt 功能，在連線建立時，以彈出視窗讓用戶登入。

流程圖：



```
<script>
    $(function () {
        // open connection
        var socket = io();

        // ask user information for login attempt
        var userInfo = askUserInfo();
        socket.emit('login attempt', userInfo);

        // if login failed, ask user information again
        socket.on('login failed', function(data) {
            if(data.accountFound) {
                alert("密碼輸入錯誤，請重新登入！");
            } else{
                alert("帳號並未註冊，請重新登入！");
            }
            userInfo = askUserInfo();
            socket.emit('login attempt', userInfo);
        });

        // if register failed, ask user information again
        socket.on('register failed', function(data) {
            alert("相同帳號已被註冊，請重新登入！");
            userInfo = askUserInfo();
            socket.emit('login attempt', userInfo);
        });

        // once login succeeded, user can start chatting
        socket.on('login successfully', function(myUsername) {
            userInfo.username = myUsername;
```

```
function askUserInfo() {
    var userInput = prompt("請輸入帳號(訪客請輸入guest，註冊請輸入new)", "guest");
    if(userInput=="new") {
        var userRegisterInfo = askUserRegisterInfo();
        userRegisterInfo.newUser = true;
        return userRegisterInfo;
    } else if(userInput == "guest" || userInput=="" || userInput==null) {    // considered as guest
        var nickname = askNickname();
        return { account:"guest", password:undefined, nickname:nickname, newUser:false };
    } else {    // registered user
        var password = prompt("請輸入密碼");
        return { account:userInput, password:password, nickname:undefined, newUser:false };
    }
}

function askUserRegisterInfo() {
    do {
        do {
            var account = prompt("請輸入註冊帳號");
            var password = prompt("請輸入註冊密碼");
            var passwordDoubleCheck = prompt("請再次輸入註冊密碼");
            if(password != passwordDoubleCheck) { alert("密碼輸入錯誤，請重新註冊！"); }
        } while(passwordDoubleCheck != password)
    }   while( !(confirm("確定註冊帳號: " + account + " ?")) )

    var nickname = askNickname();
    return {account:account, password:password, nickname:nickname}
}
```

```
function askNickname() {
    do {
        do {
            var nickname = prompt("請輸入暱稱");
            if(nickname=="" || nickname==null) { alert("暱稱請勿空白！"); }
            if(nickname.includes("guest")) { alert("暱稱不可為guest！"); }
        }
        while( nickname=="" || nickname==null || nickname.includes("guest") )

    } while( !(confirm("確定使用暱稱: " + nickname + " ?")) )
    return nickname;
}
```

2. Server 端

利用 socket.io 完成各種功能