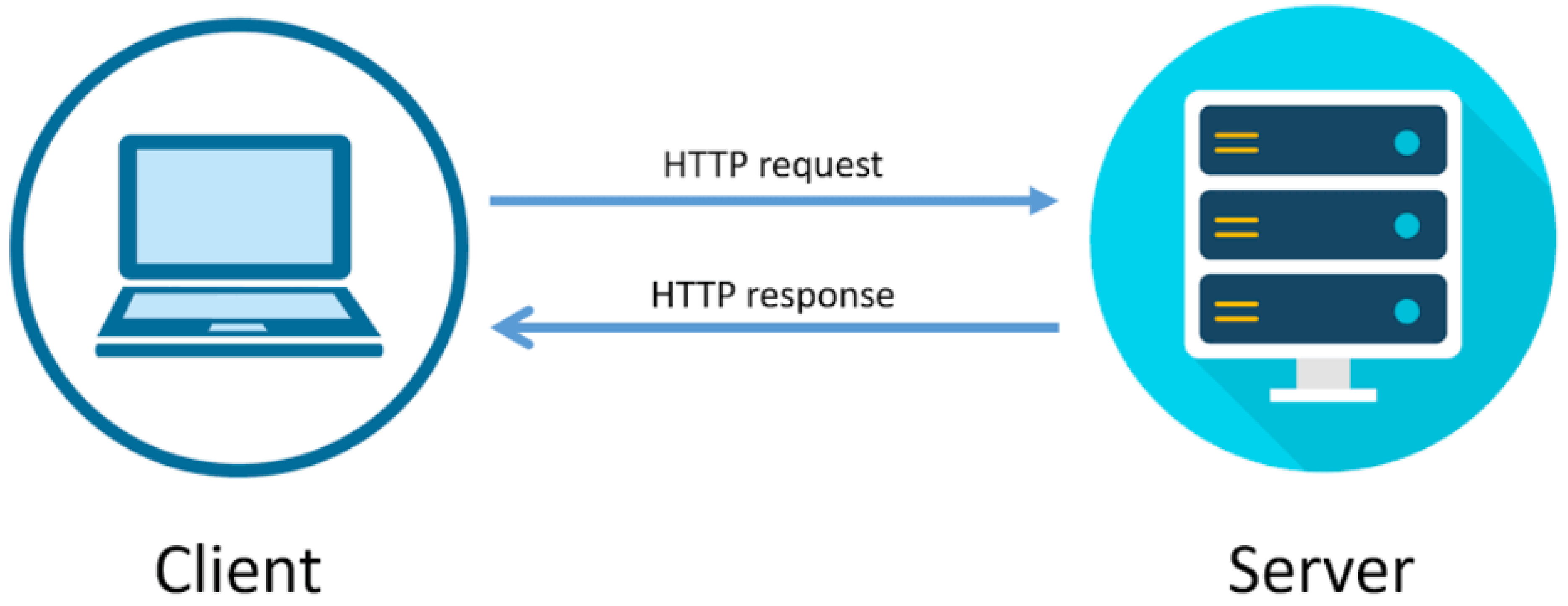
物聯網實務

10_26

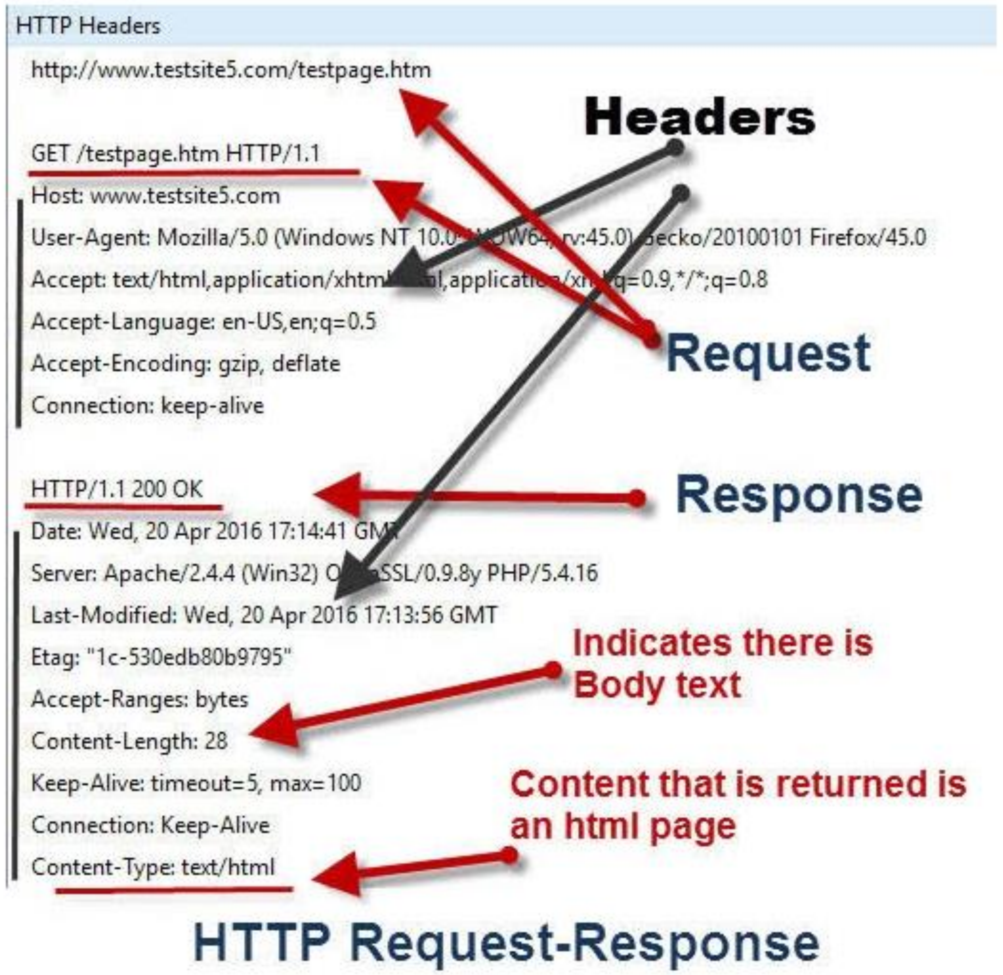
廖裕評

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol)



https://medium.com/@rohitpatil97/http-request-http-response-context-and-headers-part-iii-5c37bd4cb06b

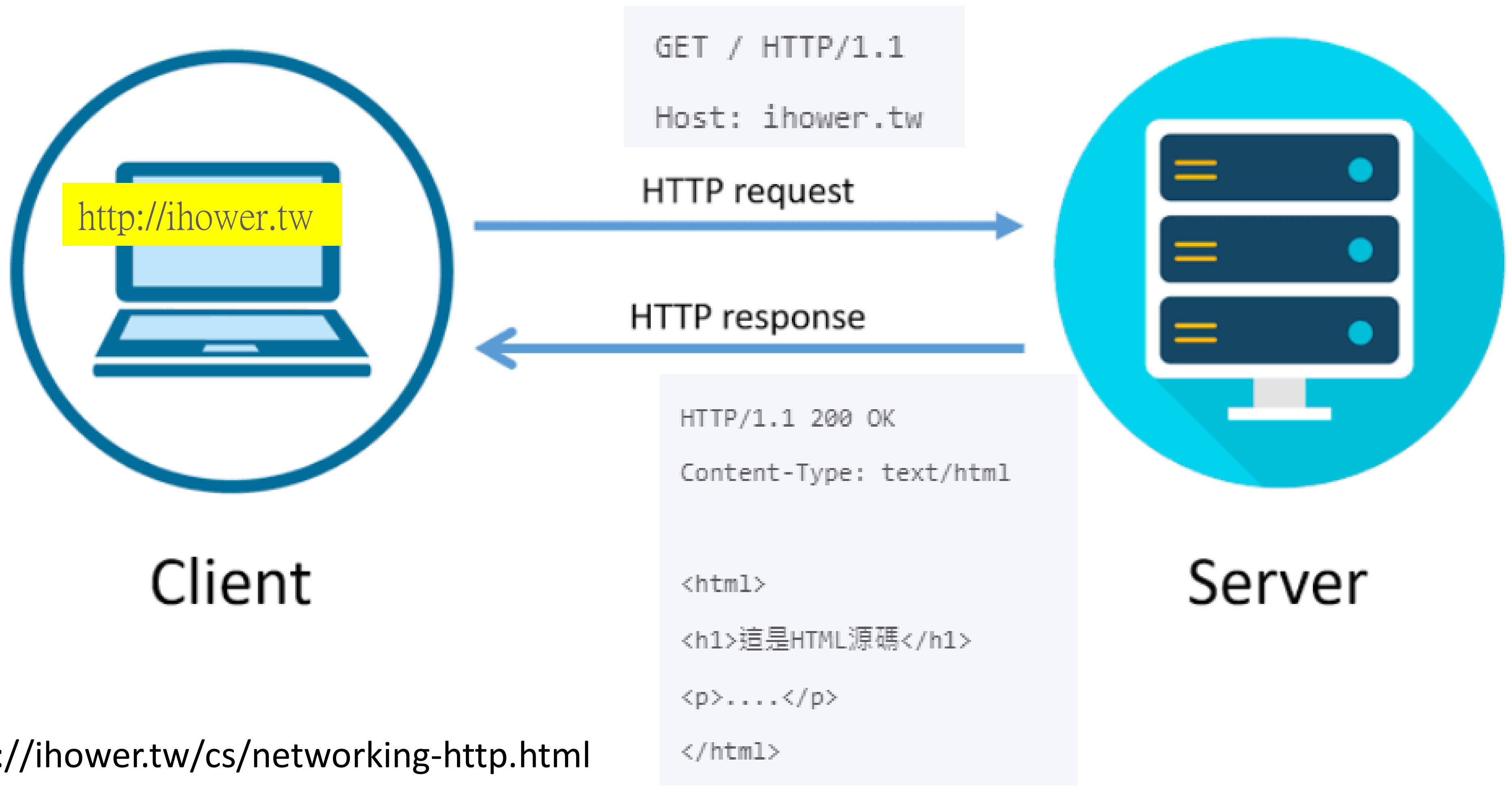
HTTP Headers


Headers
Request
Response
Indicates there is Body text
Content that is returned is an html page
HTTP Request-Response

```
HTTP Headers
http://www.testsite5.com/testpage.htm

GET /testpage.htm HTTP/1.1
Host: www.testsite5.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP/1.1 200 OK
Date: Wed, 20 Apr 2016 17:14:41 GMT
Server: Apache/2.4.4 (Win32) OpenSSL/0.9.8y PHP/5.4.16
Last-Modified: Wed, 20 Apr 2016 17:13:56 GMT
Etag: "1c-530edb80b9795"
Accept-Ranges: bytes
Content-Length: 28
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

https://medium.com/@rohitpatil97/http-request-http-response-context-and-headers-part-iii-5c37bd4cb06b

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol)

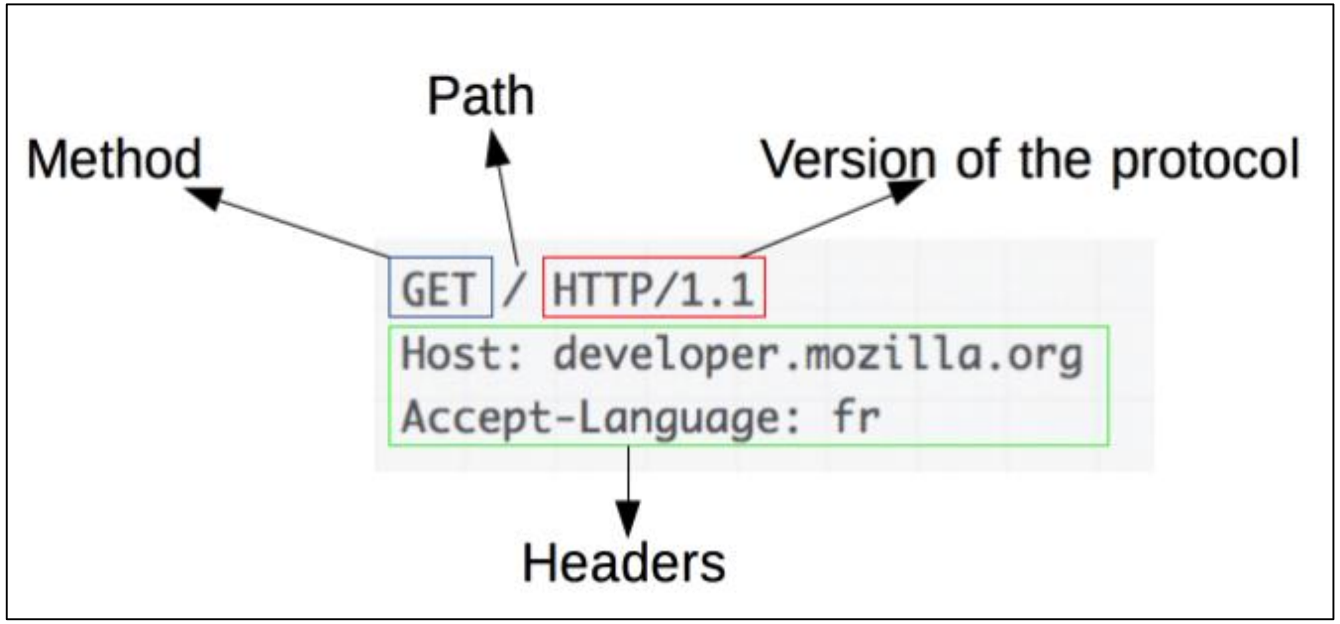


```
GET / HTTP/1.1
Host: ihower.tw
```

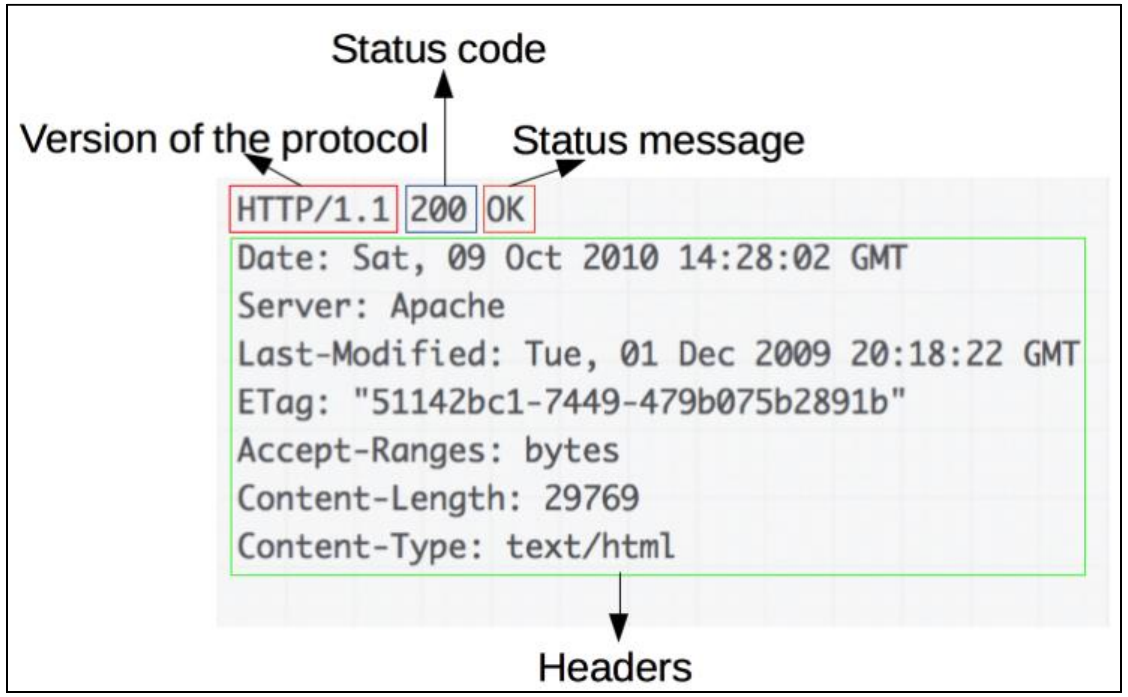
```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
<h1>這是HTML源碼</h1>
<p>....</p>
</html>
```

https://ihower.tw/cs/networking-http.html

Request & Response



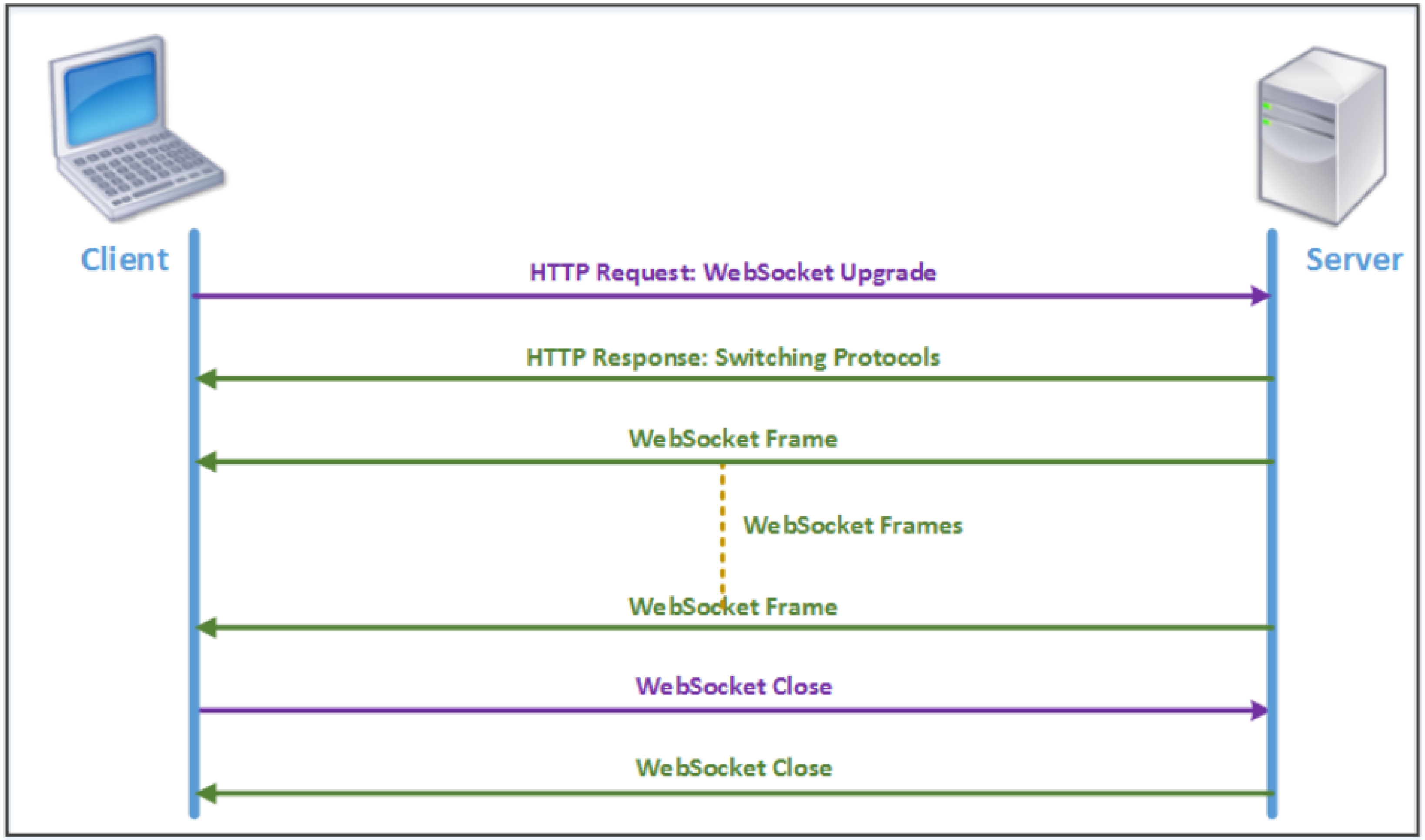
```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

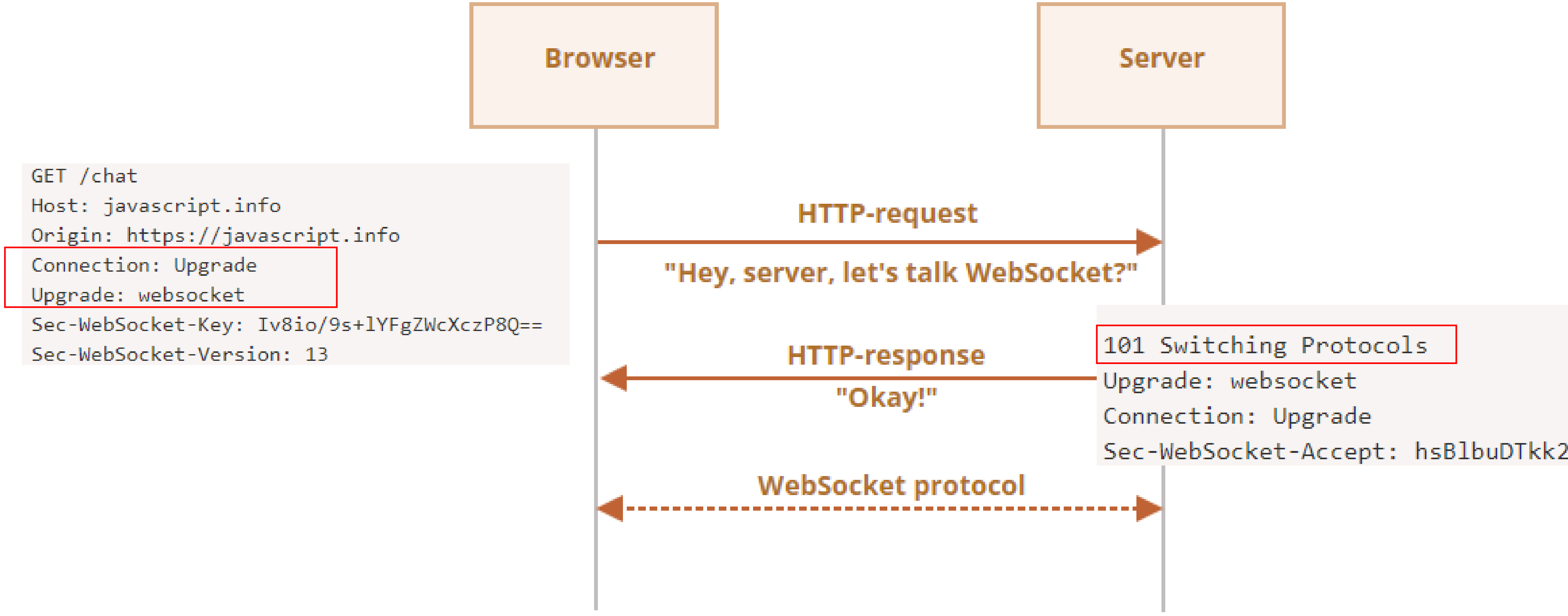


```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

HTML5 WebSocket





```
GET /chat
Host: javascript.info
Origin: https://javascript.info
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Key: Iv8io/9s+lYFgZWcXczP8Q==
Sec-WebSocket-Version: 13
```

```
101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: hsBlbuDTkk2
```

https://javascript.info/websocket

The WebSocket Protocol

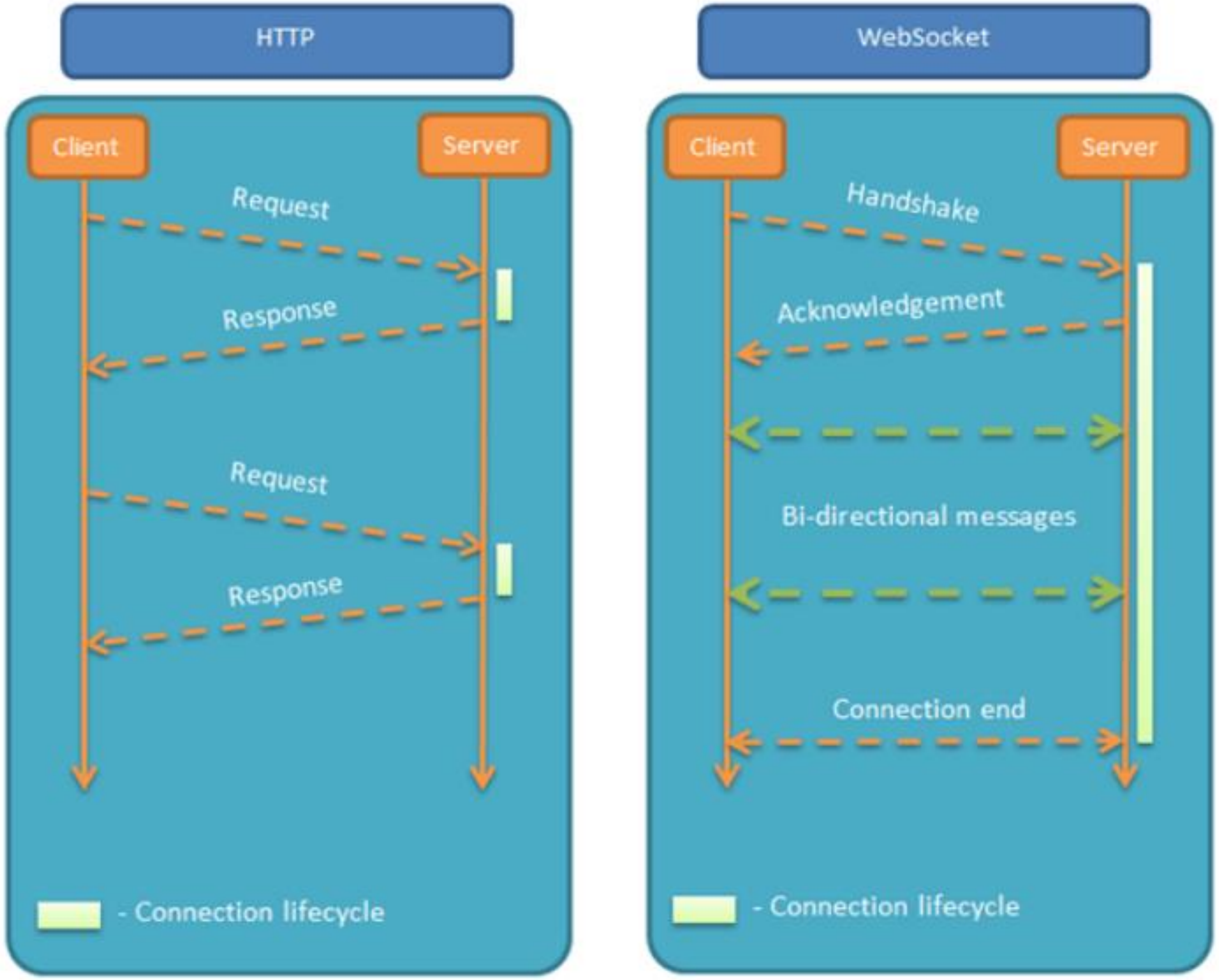
- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

- The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP.

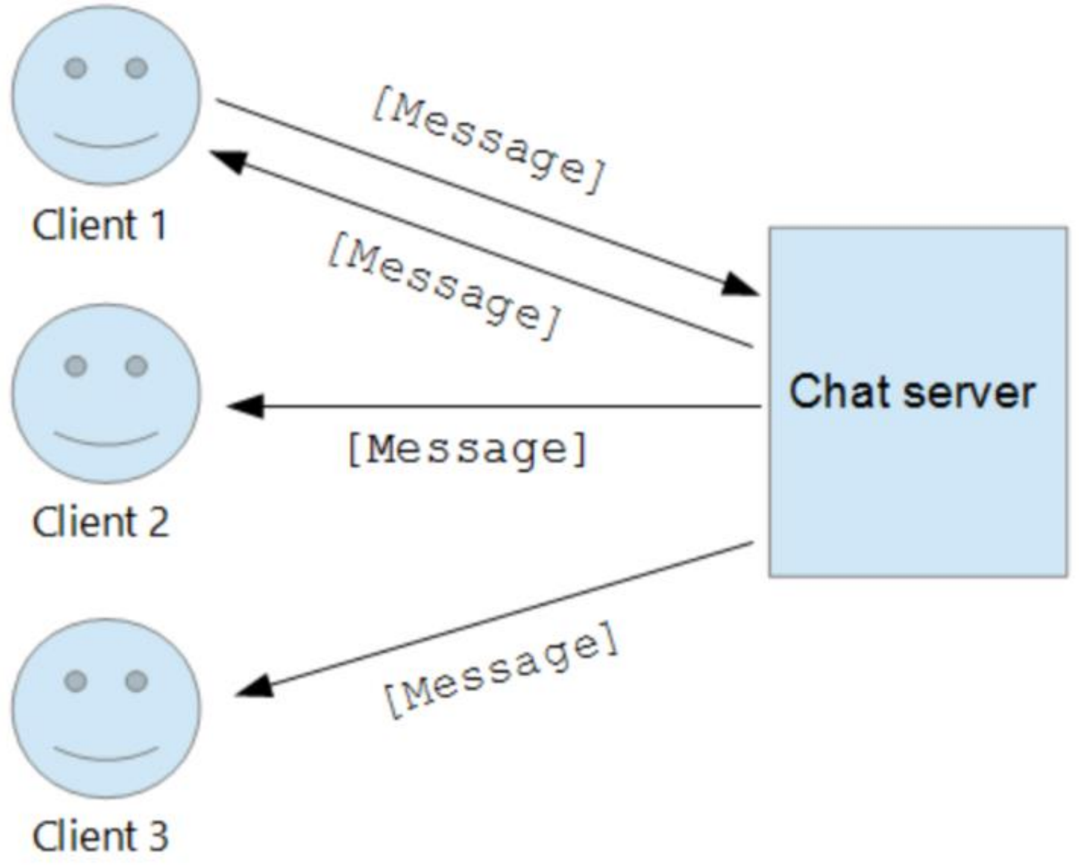
- The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections.

https://www.xoriant.com/blog/websocket-web-is-stateful-now

HTTP vs. WebSocket

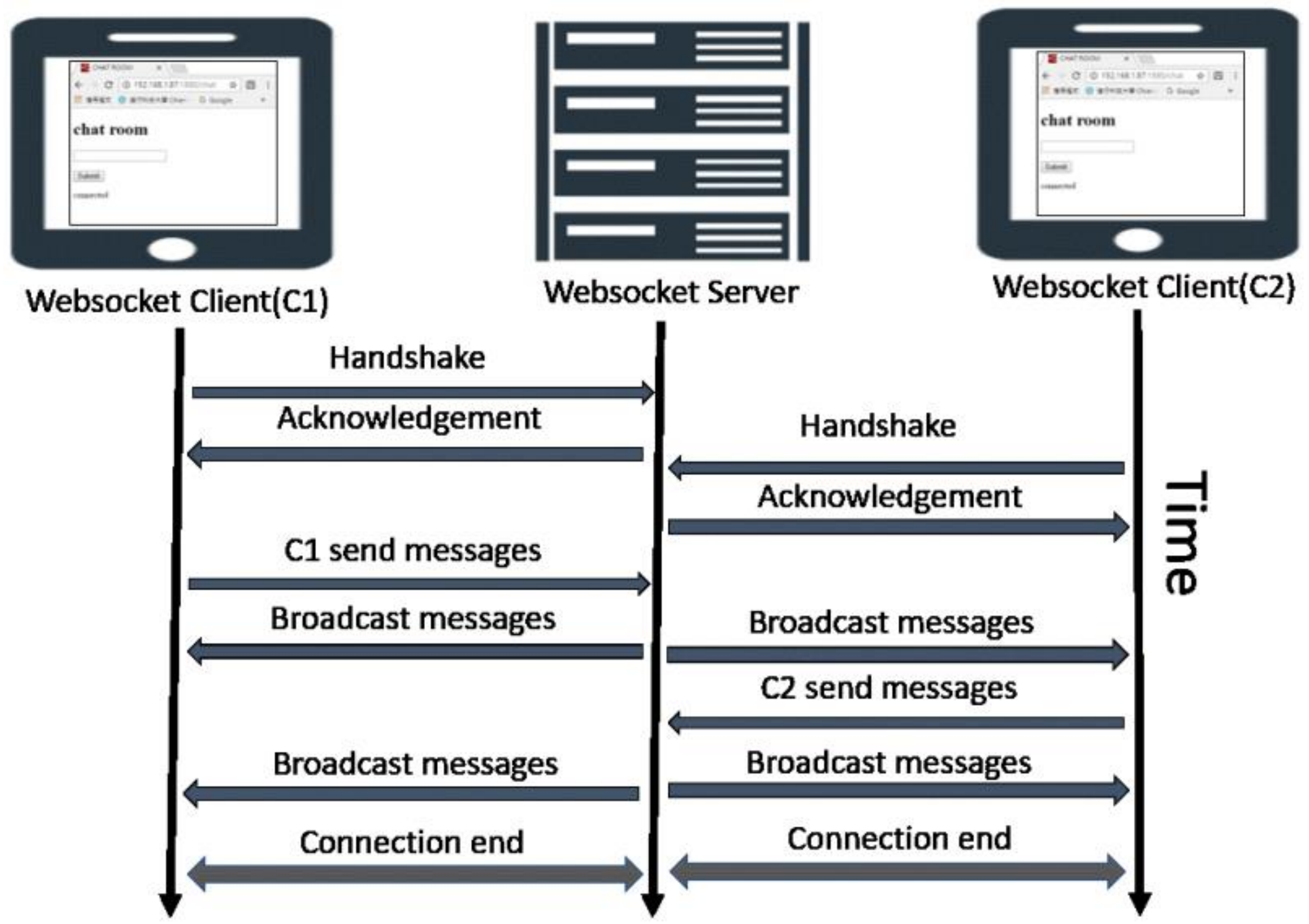


Use cases of WebSocket

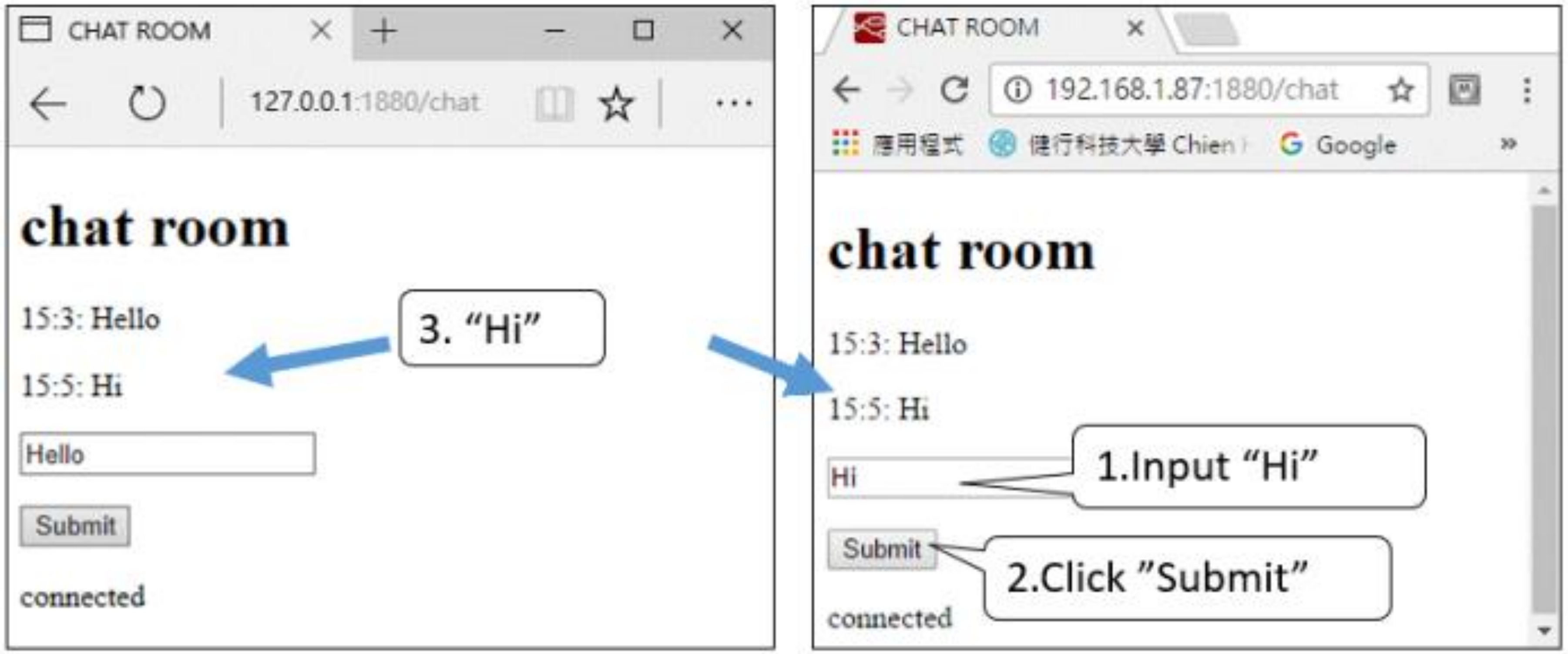


https://www.cometchat.com/tutorials/what-is-websockets

Use cases of WebSocket



Chatroom


CHAT ROOM

127.0.0.1:1880/chat

chat room

15:3: Hello

15:5: Hi

Hello

Submit

connected

CHAT ROOM

192.168.1.87:1880/chat

應用程式 健行科技大學 Chien Google

chat room

15:3: Hello

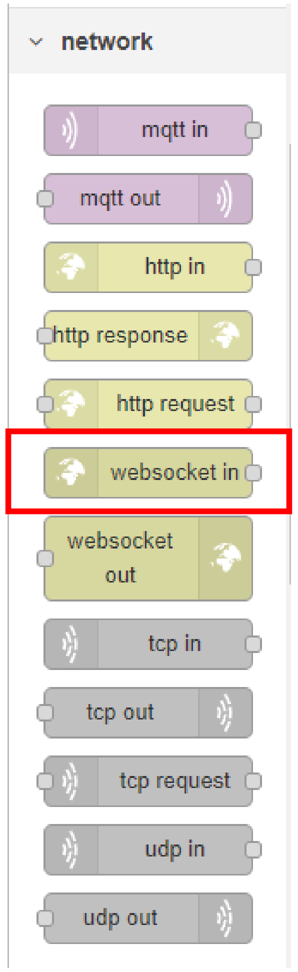
15:5: Hi

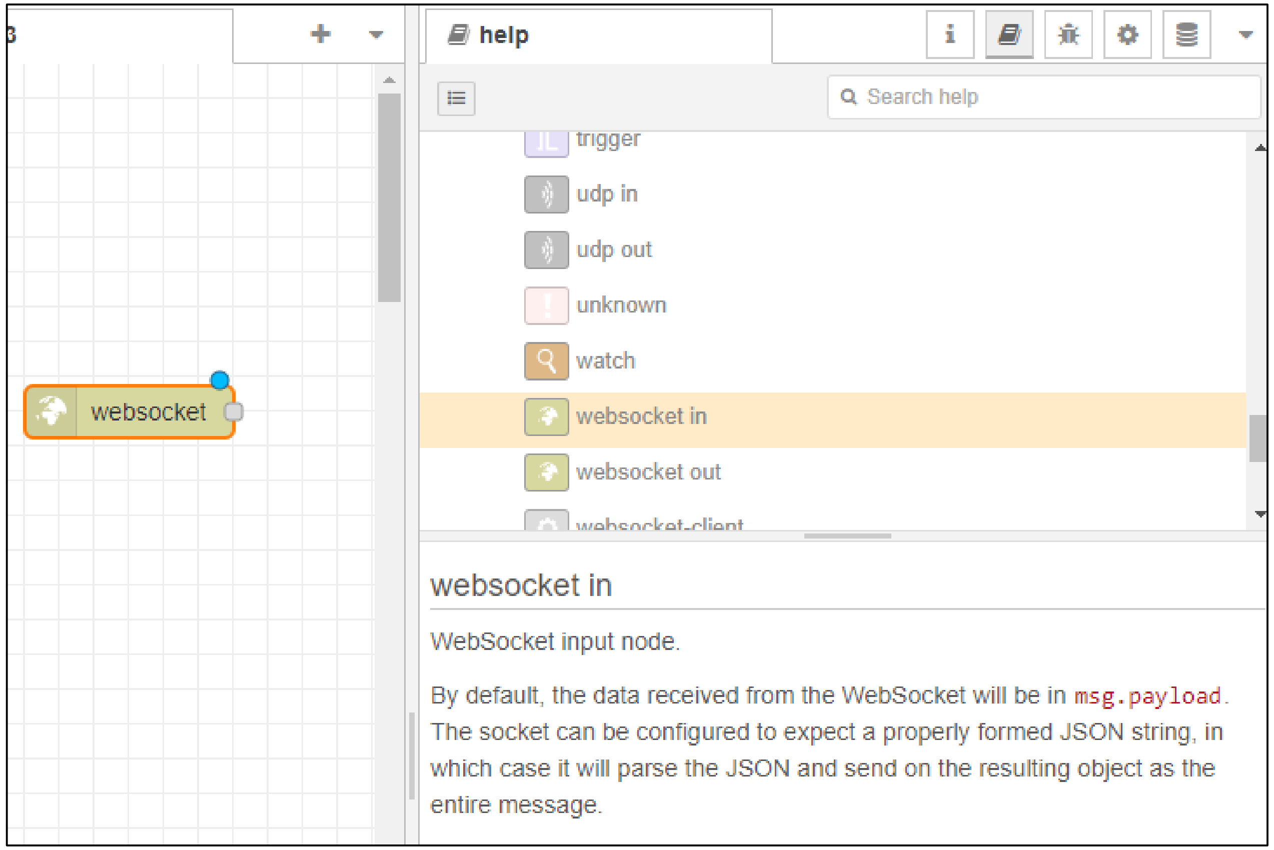
Hi

Submit

connected

websocket in Node



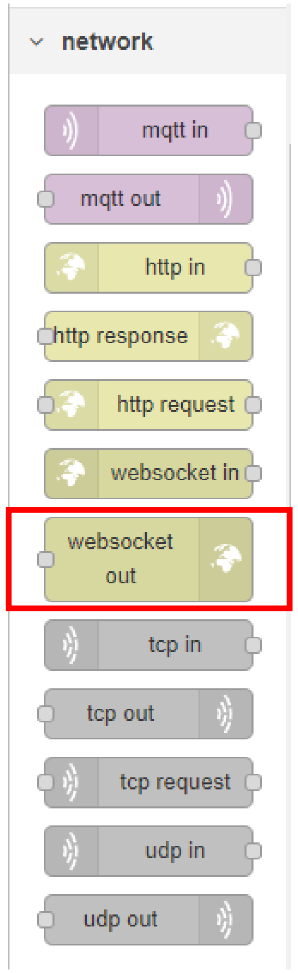


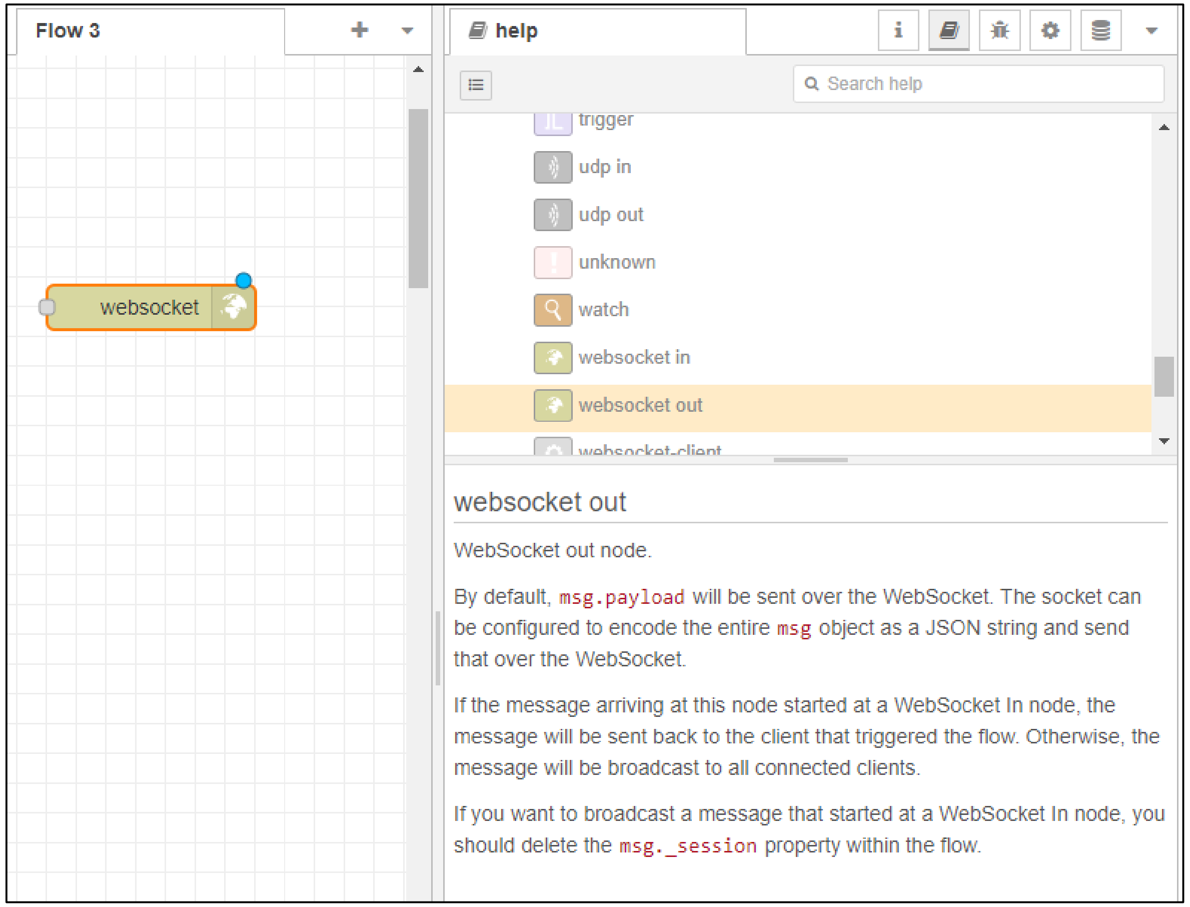
websocket in

WebSocket input node.

By default, the data received from the WebSocket will be in `msg.payload`. The socket can be configured to expect a properly formed JSON string, in which case it will parse the JSON and send on the resulting object as the entire message.

websocket out Node





websocket out

WebSocket out node.

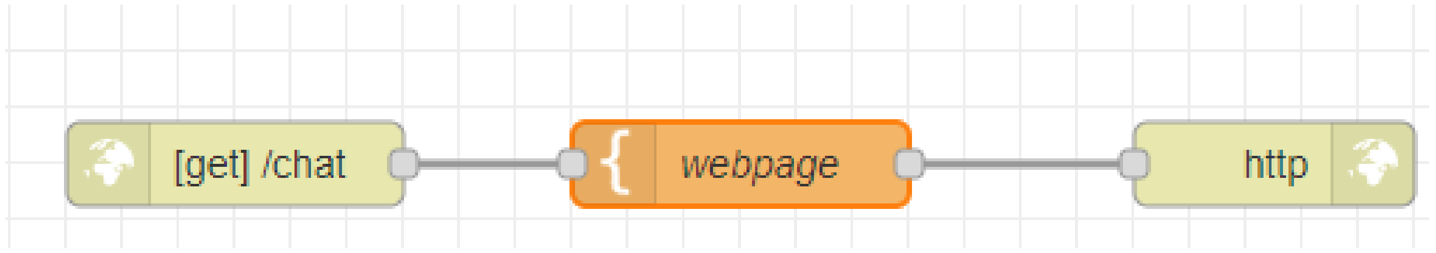
By default, `msg.payload` will be sent over the WebSocket. The socket can be configured to encode the entire `msg` object as a JSON string and send that over the WebSocket.

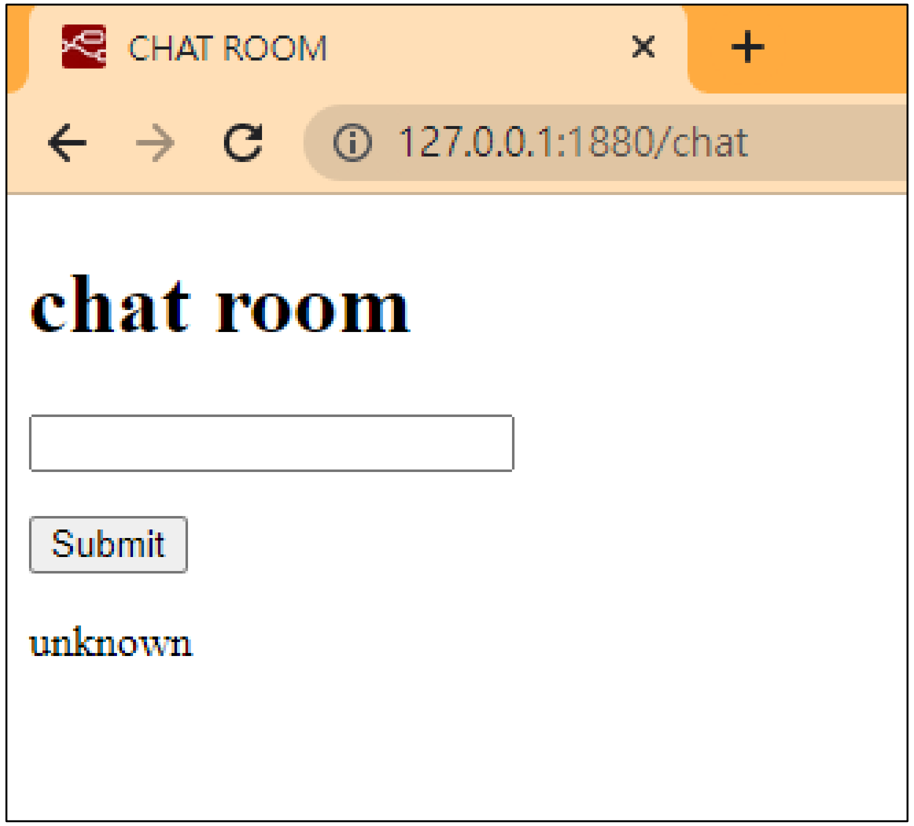
If the message arriving at this node started at a WebSocket In node, the message will be sent back to the client that triggered the flow. Otherwise, the message will be broadcast to all connected clients.

If you want to broadcast a message that started at a WebSocket In node, you should delete the `msg._session` property within the flow.

Exercise 7-1

- Create a chat room HTML

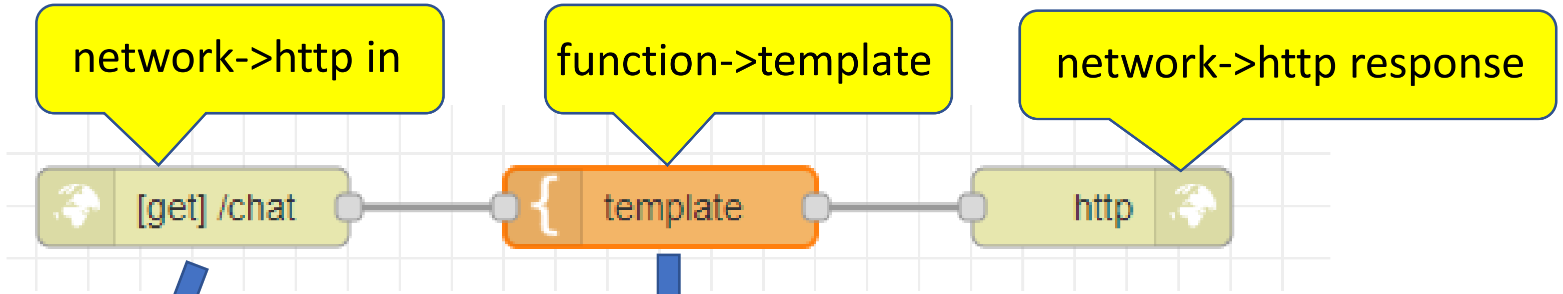


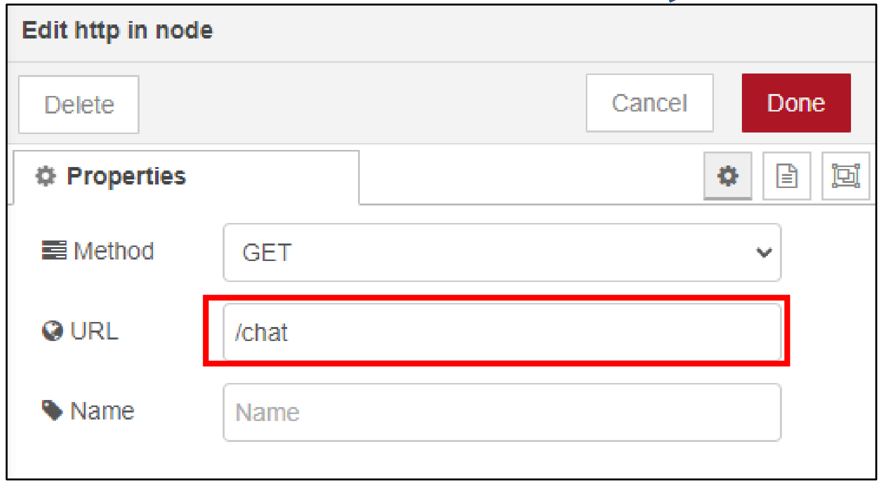


chat room

unknown

Add nodes




Edit http in node

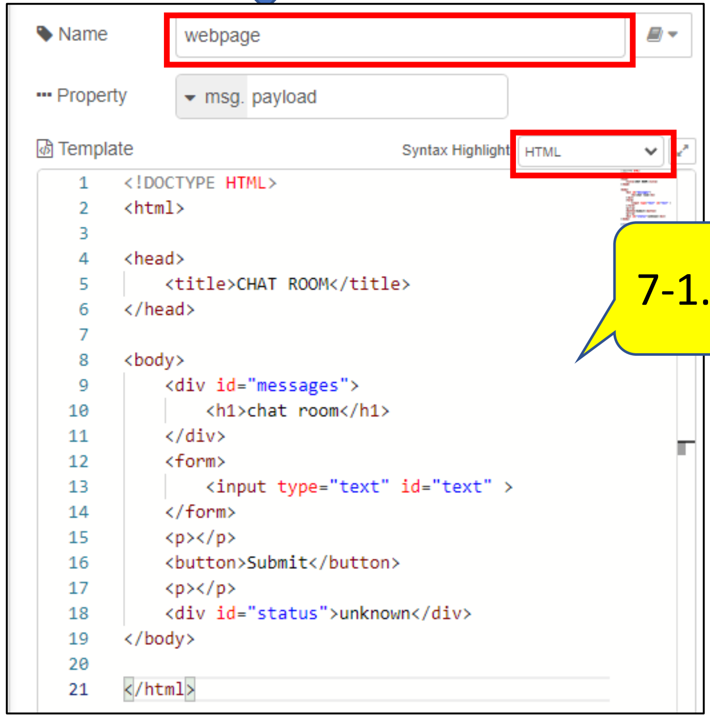
Delete Cancel Done

Properties

Method: GET

URL: /chat

Name: Name


Name: webpage

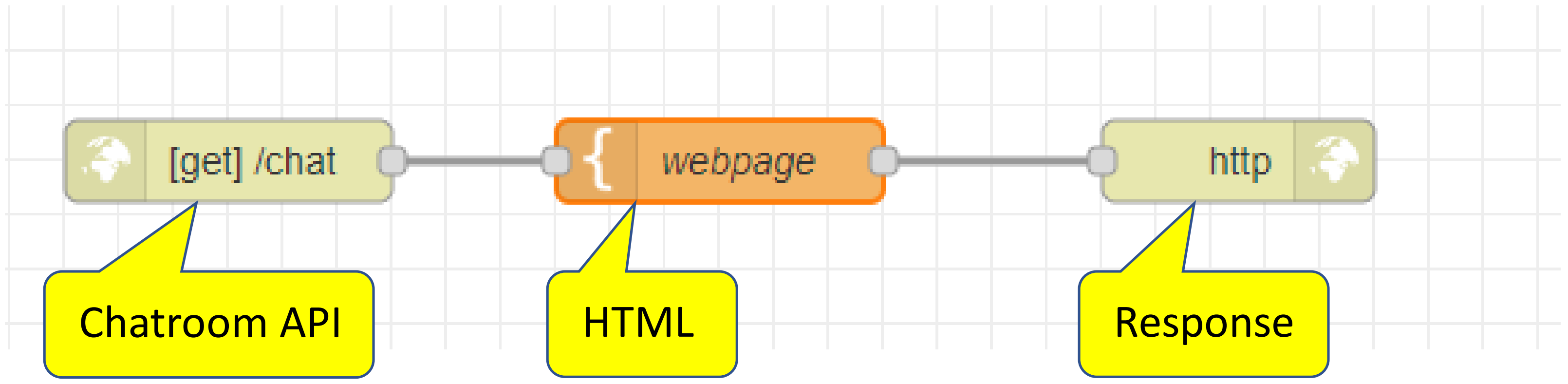
Property: msg. payload

Template — Syntax Highlight: HTML

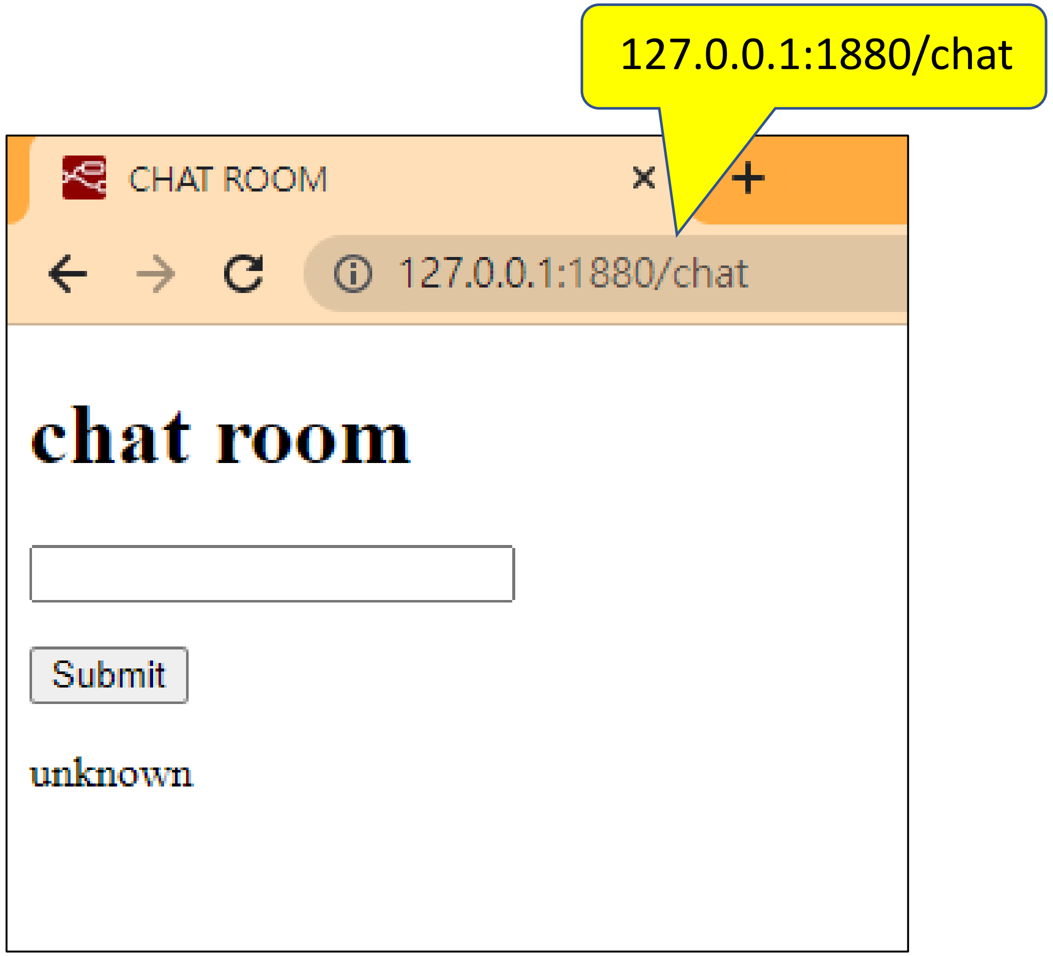
```
<!DOCTYPE HTML>
<html>

<head>
    <title>CHAT ROOM</title>
</head>

<body>
    <div id="messages">
        <h1>chat room</h1>
    </div>
    <form>
        <input type="text" id="text" >
    </form>
    <p></p>
    <button>Submit</button>
    <p></p>
    <div id="status">unknown</div>
</body>

</html>
```

7-1.txt

Chatroom API flow



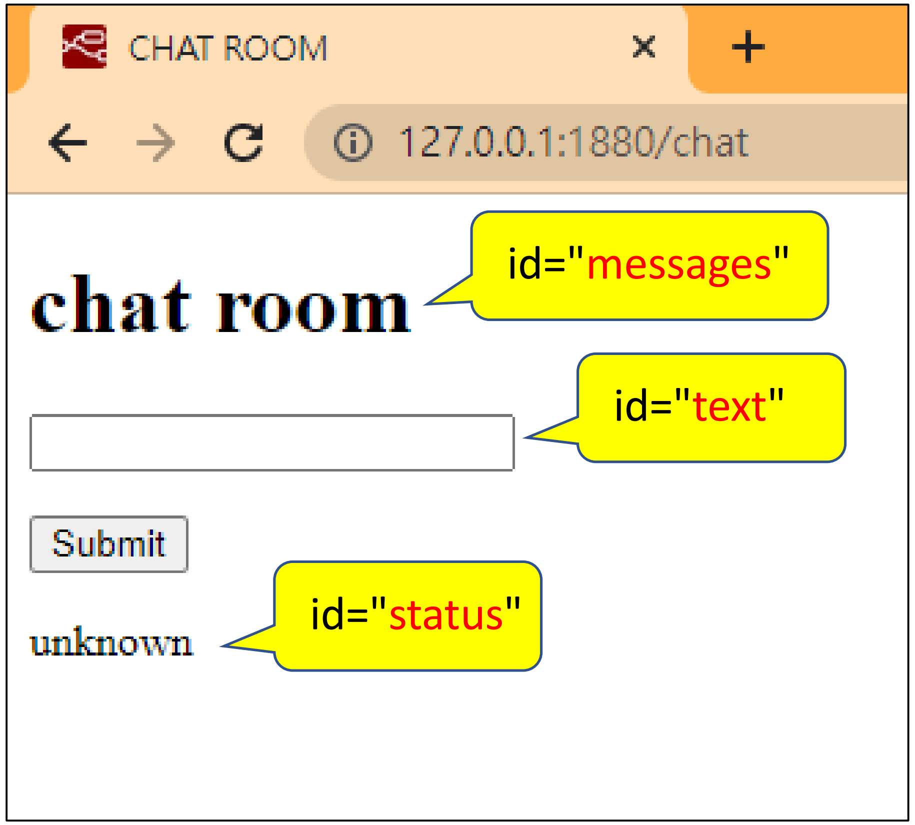
Access the website



chat room

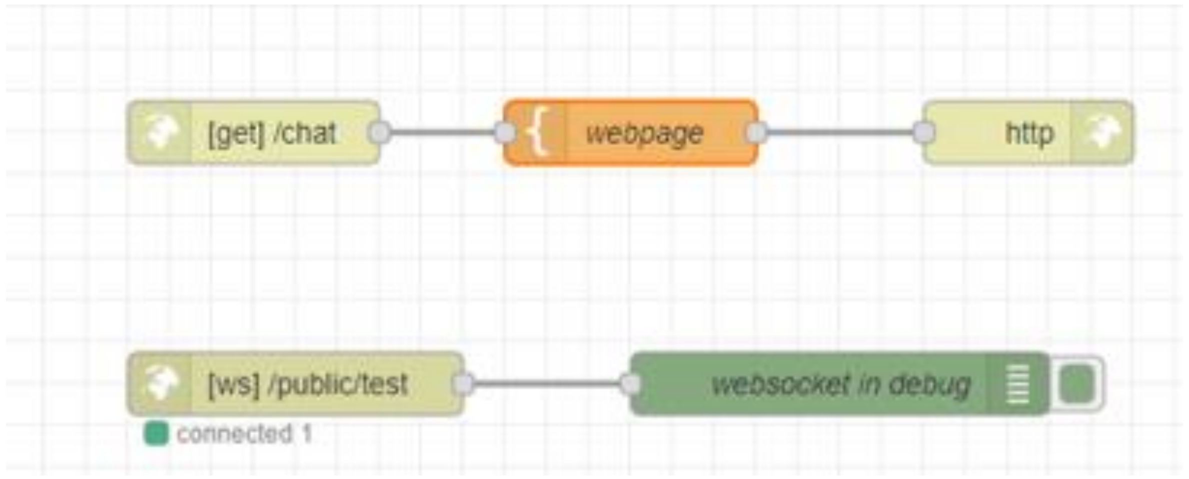
unknown

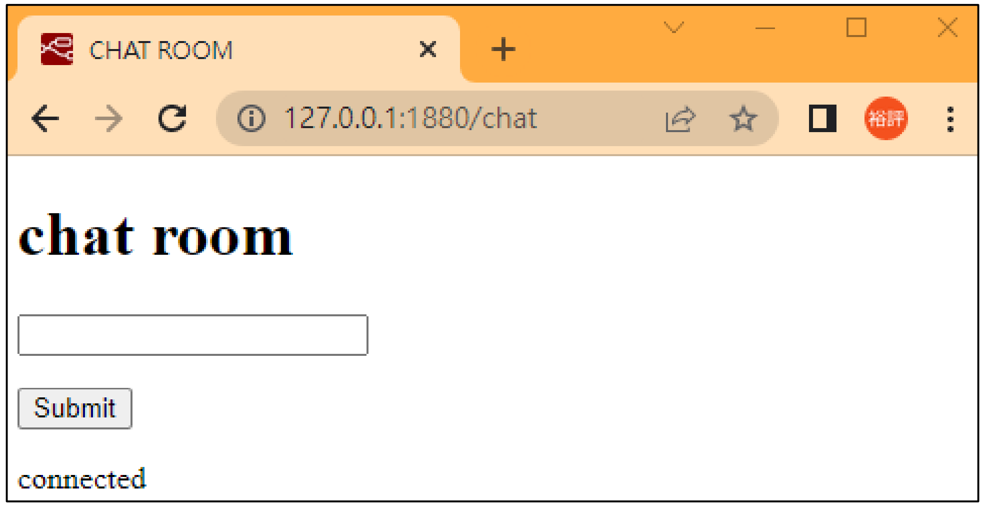
```
<!DOCTYPE HTML>
<html>
<head>
   <title>CHAT ROOM</title>
</head>
<body>
   <div id="messages">
      <h1>chat room</h1>
   </div>
   <form>
      <input type="text" id="text" >
   </form>
   <p></p>
   <button>Submit</button>
   <p></p>
   <div id="status">unknown</div>
</body>
</html>
```



Exercise 7-2

- Connect with the websocket server



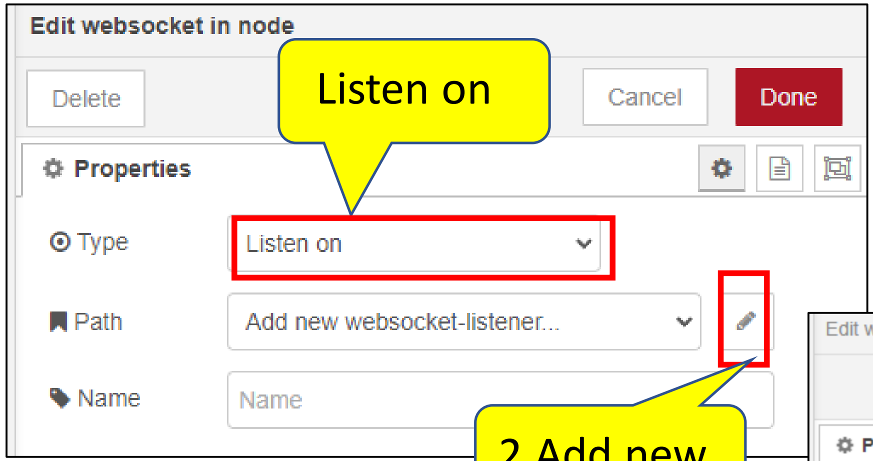


chat room

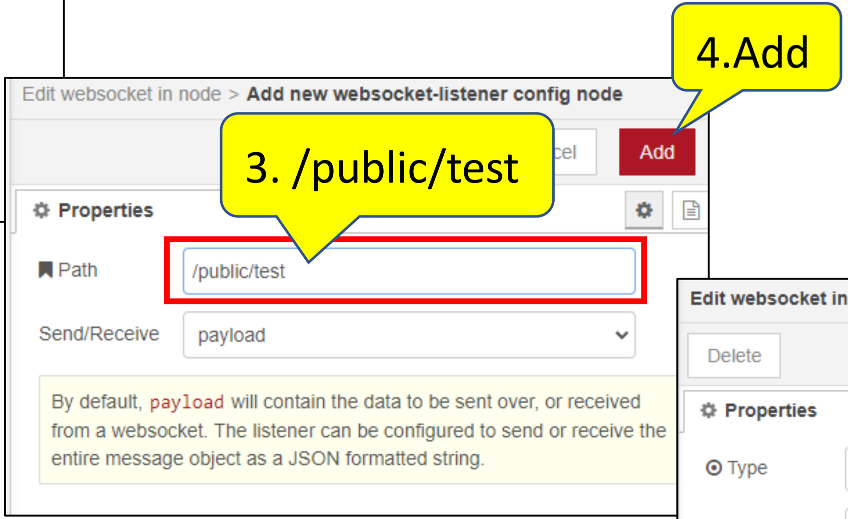
Submit

connected

Add websocket in node

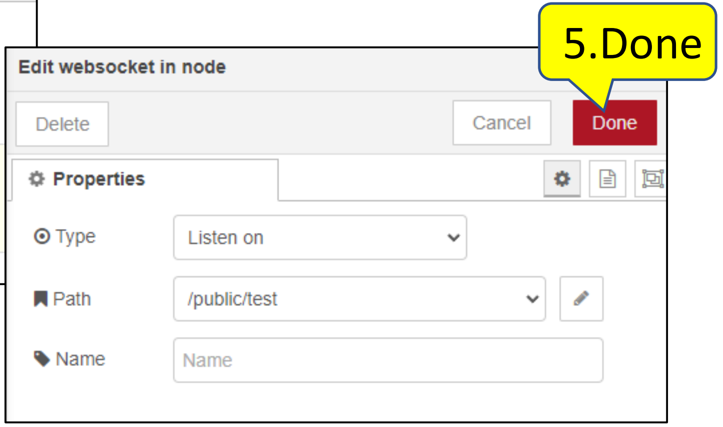


Type: Listen on
Path: Add new websocket-listener...
Name: Name

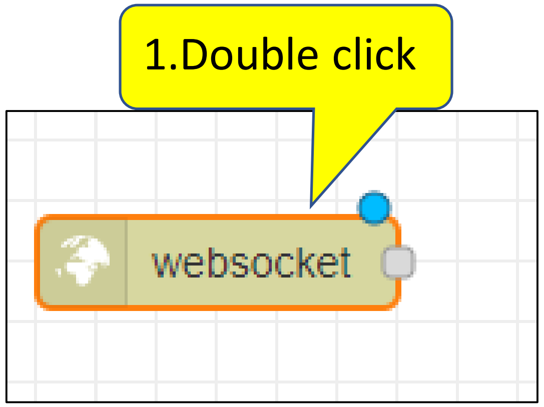

Edit websocket in node > Add new websocket-listener config node
4.Add
Add
3. /public/test
Properties

Path: /public/test
Send/Receive: payload

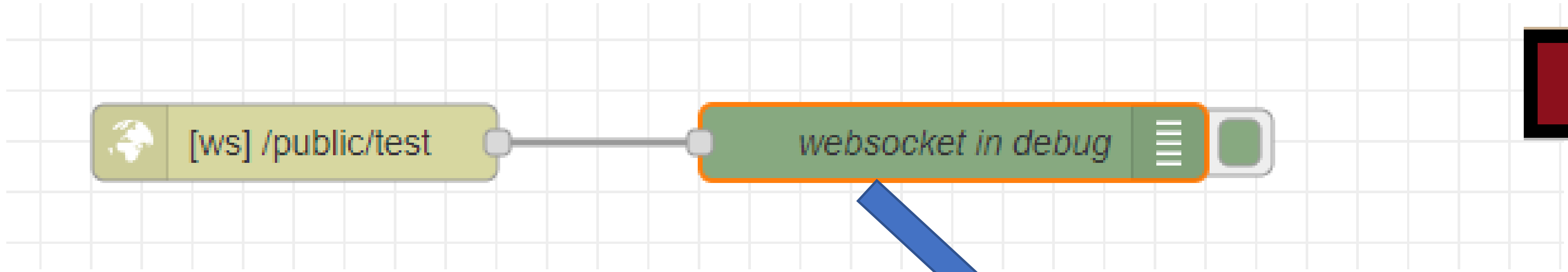
By default, `payload` will contain the data to be sent over, or received from a websocket. The listener can be configured to send or receive the entire message object as a JSON formatted string.

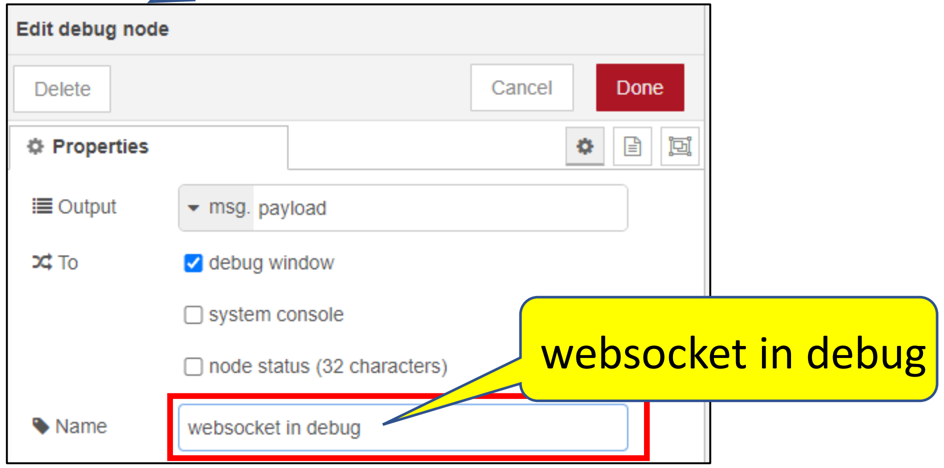


Type: Listen on
Path: /public/test
Name: Name



Add debug node





Edit debug node

Delete | Cancel | Done

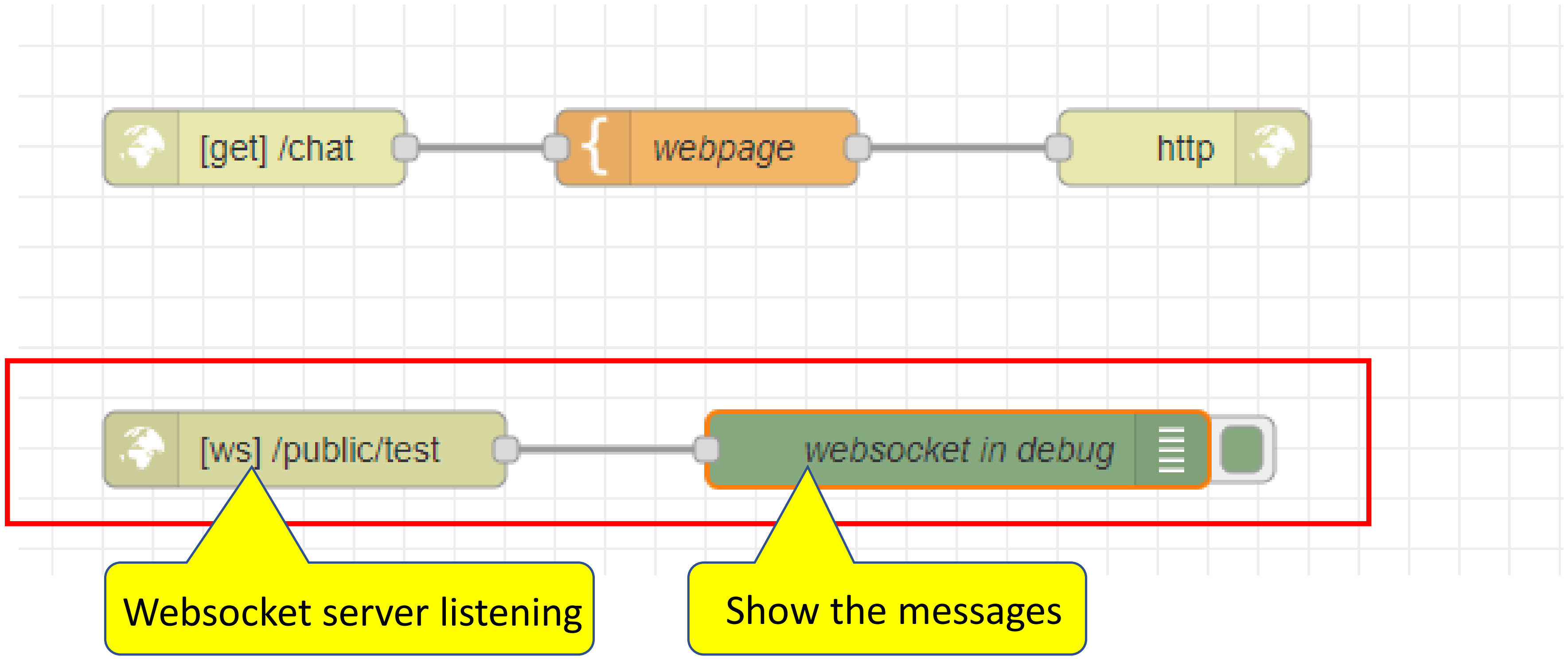
Properties

Output: msg. payload

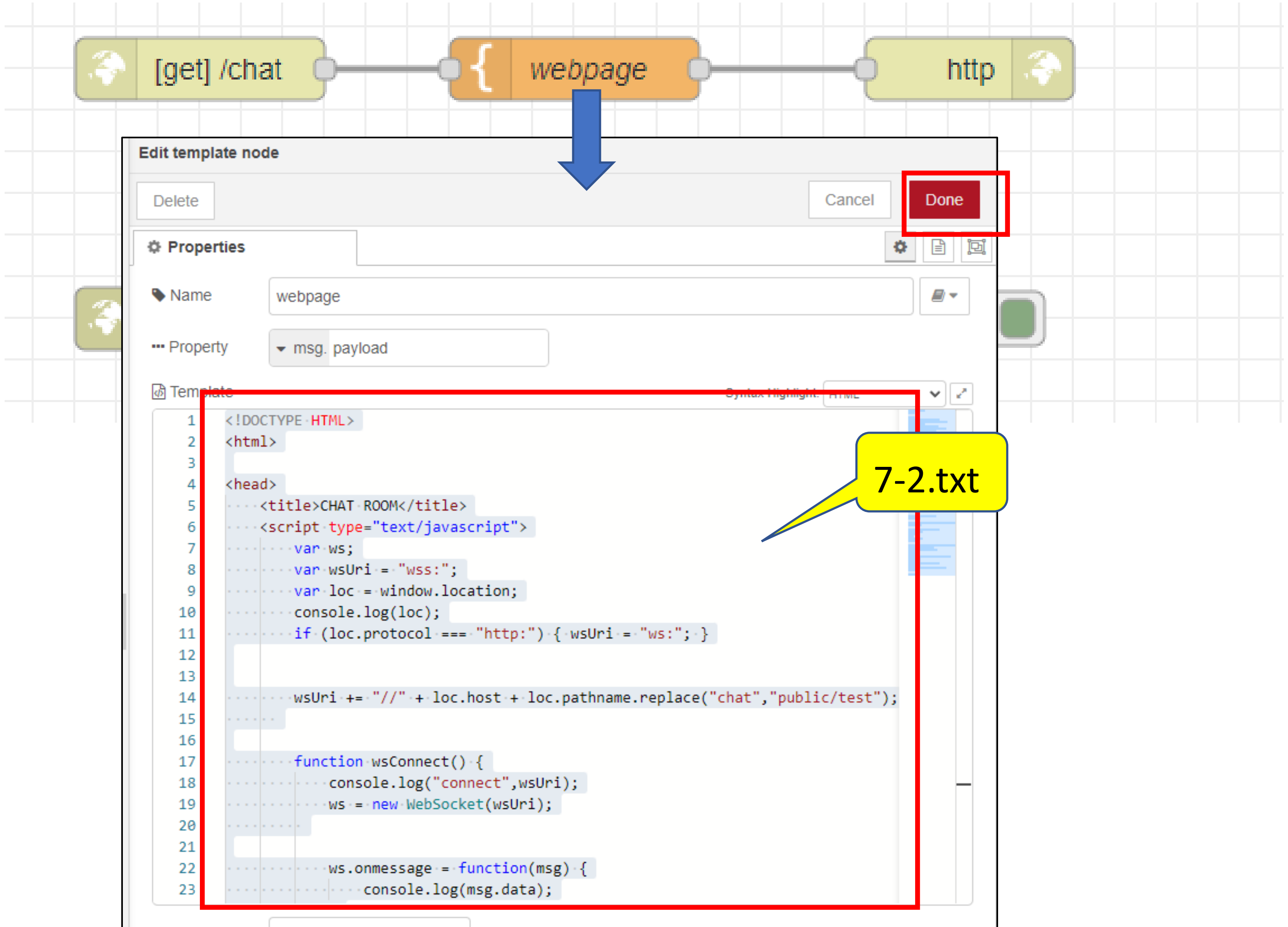
To: [✓ debug window]
system console
node status (32 characters)

Name: websocket in debug

Websocket flow



Edit HTML



Edit template node

Properties

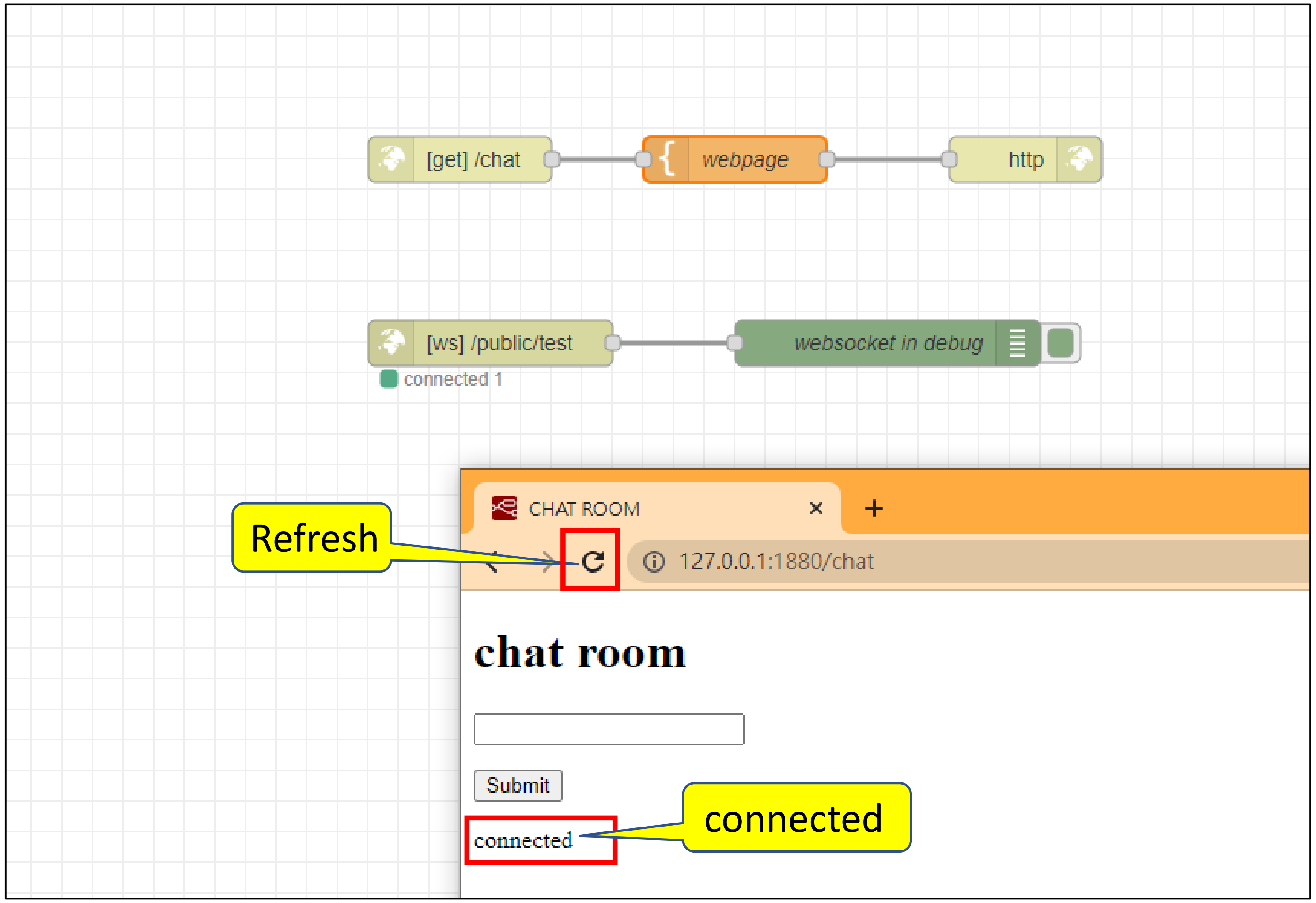
Name: webpage

Property: msg. payload

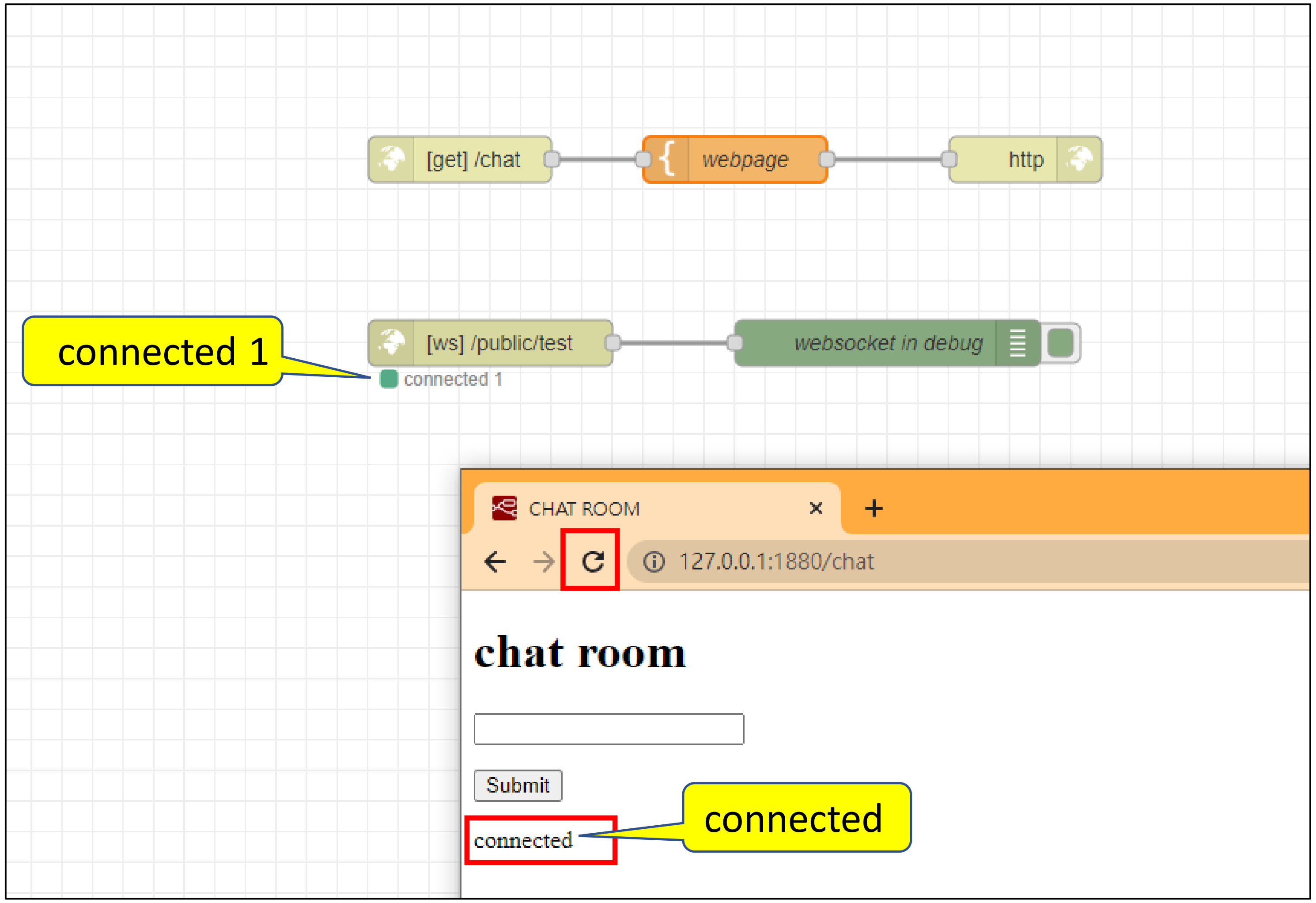
Template

```
<!DOCTYPE HTML>
<html>

<head>
    <title>CHAT ROOM</title>
    <script type="text/javascript">
        var ws;
        var wsUri = "wss:";
        var loc = window.location;
        console.log(loc);
        if (loc.protocol === "http:") { wsUri = "ws:"; }


        wsUri += "//" + loc.host + loc.pathname.replace("chat","public/test");
        

        function wsConnect() {
            console.log("connect",wsUri);
            ws = new WebSocket(wsUri);
            

            ws.onmessage = function(msg) {
                console.log(msg.data);
```

Refresh chat room web page

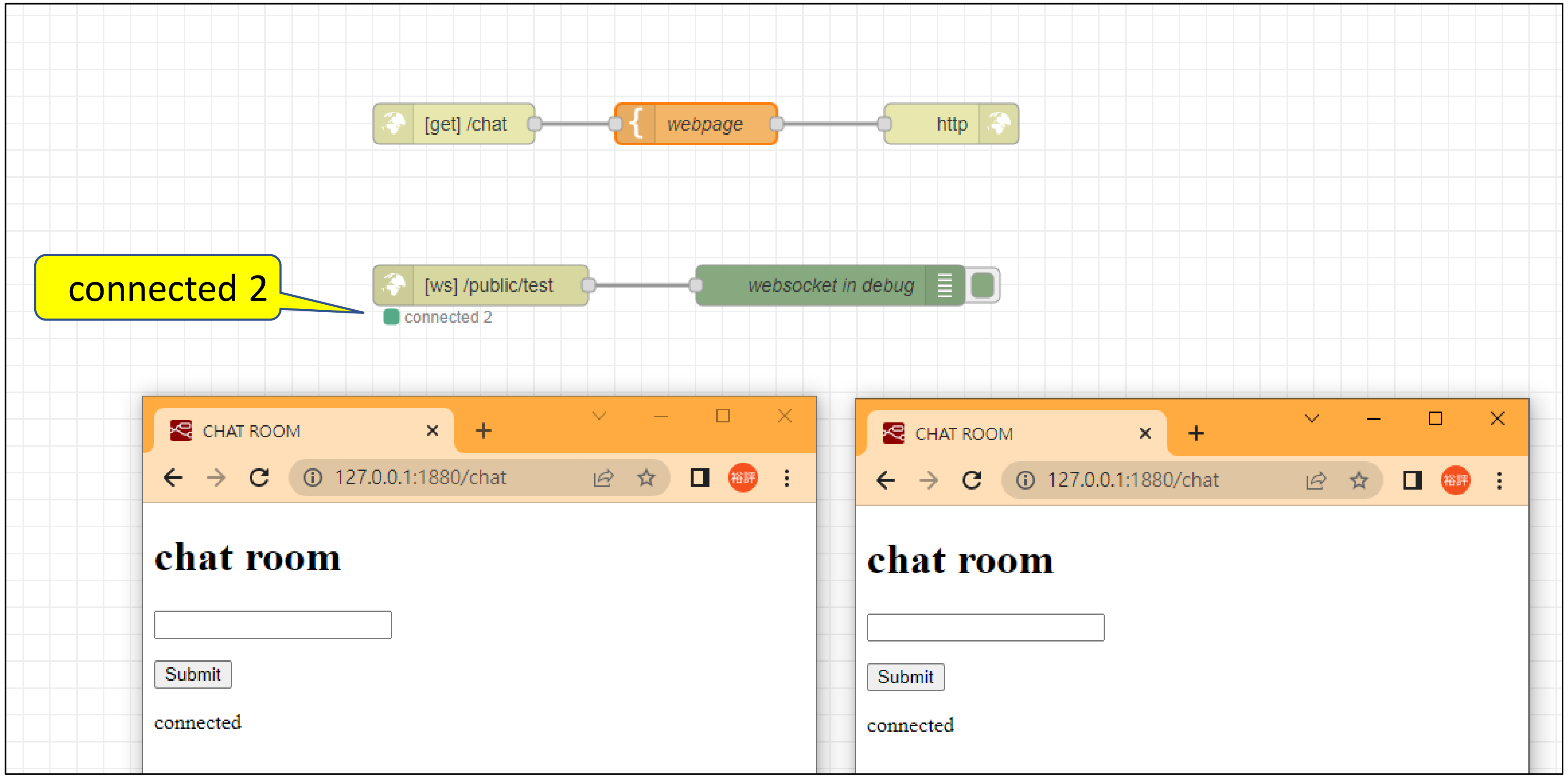


Refresh chat room web page



chat room

Create a second client



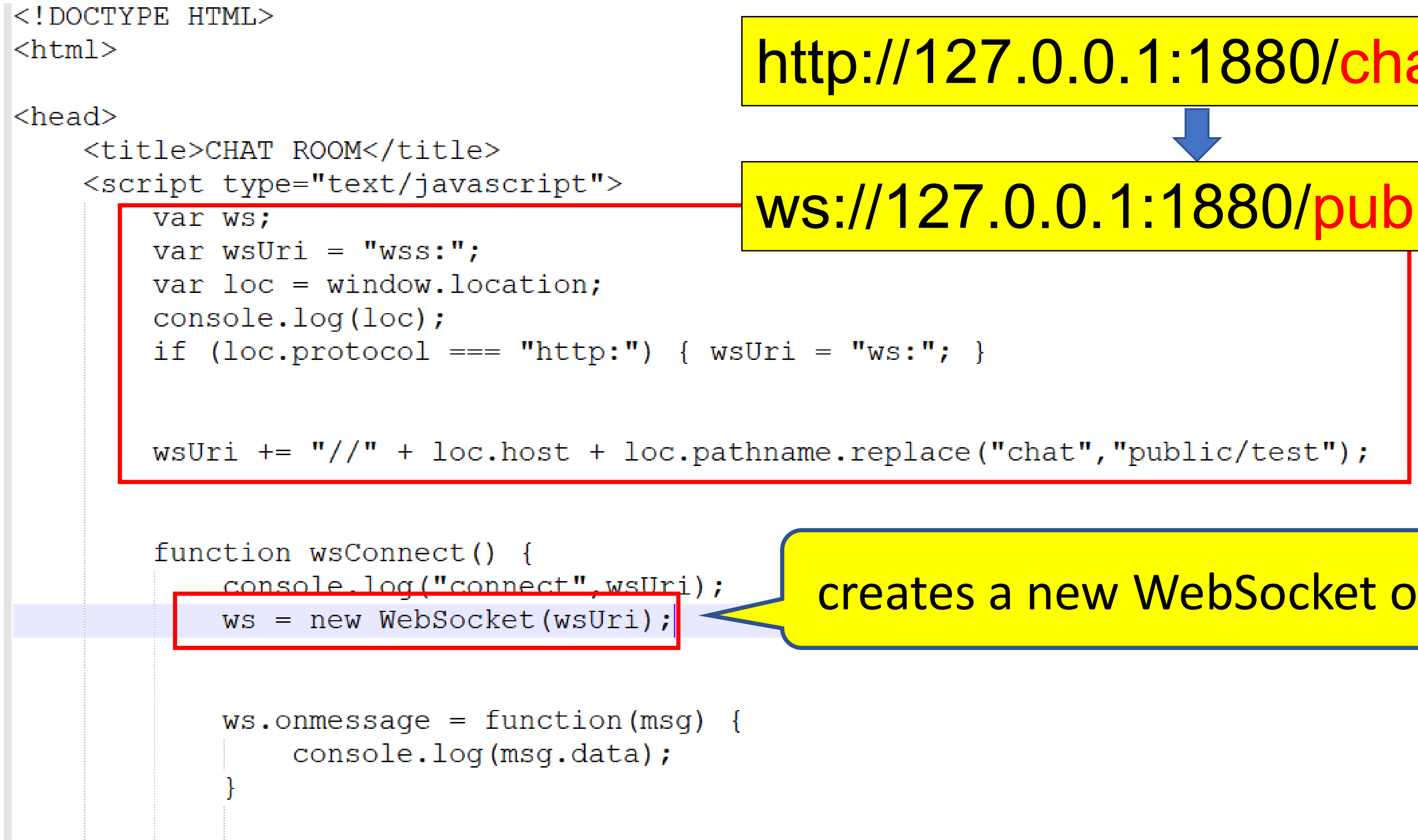
7-2.txt

Call function wsConnect()

```
<body onload="wsConnect()"          onunload="ws.onclose()" >
   <div id="messages"><h1>chat room</h1> </div>
   <form>
     <input type="text" id="text" >
   </form>
   <p></p>


   <button  onclick="sendchat()" >Submit</button>
   <p></p>
   <div id="status">unknown</div>
   </body>
```

7-2.txt

7-2.txt



```
<!DOCTYPE HTML>
<html>

<head>
    <title>CHAT ROOM</title>
    <script type="text/javascript">
        var ws;
        var wsUri = "wss:";
        var loc = window.location;
        console.log(loc);
        if (loc.protocol === "http:") { wsUri = "ws:"; }


        wsUri += "//" + loc.host + loc.pathname.replace("chat","public/test");

        function wsConnect() {
            console.log("connect",wsUri);
            ws = new WebSocket(wsUri);

            ws.onmessage = function(msg) {
                console.log(msg.data);
            }
```

http://127.0.0.1:1880/chat

ws://127.0.0.1:1880/public/test

creates a new WebSocket object

HTML5 - WebSockets

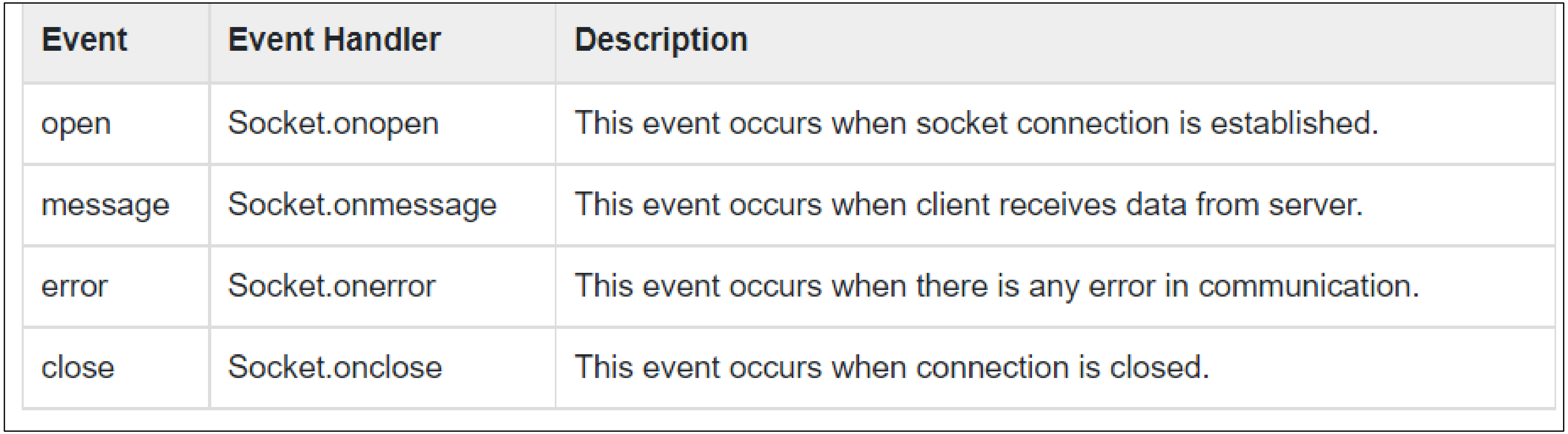
- creates a new WebSocket object:

```
var Socket = new WebSocket(url, [protocal] );
```

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful.

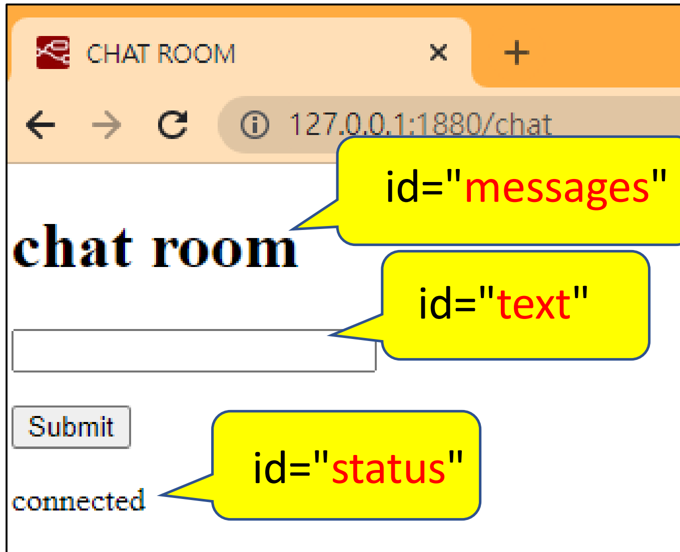
https://www.tutorialspoint.com/html5/html5_websocket.htm

WebSocket Events



Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed.

```
ws.onmessage = function(msg) {
                 console.log(msg.data);
              }

ws.onopen = function() {
              document.getElementById('status').innerHTML = "connected";
              console.log("connected");
              }

ws.onclose = function() {
              document.getElementById('status').innerHTML = "not connected";
              setTimeout(wsConnect,3000);
              }

ws.onerror = function() {
              document.getElementById('status').innerHTML = "ERROR";
              setTimeout(wsConnect,3000);
              }
```

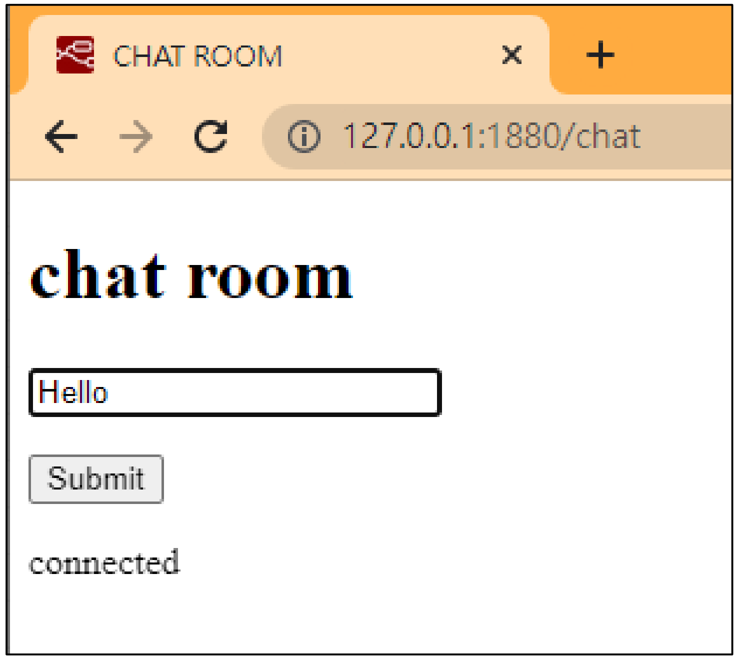


chat room

connected

Exercise 7-3

- Submit a message to websocket server

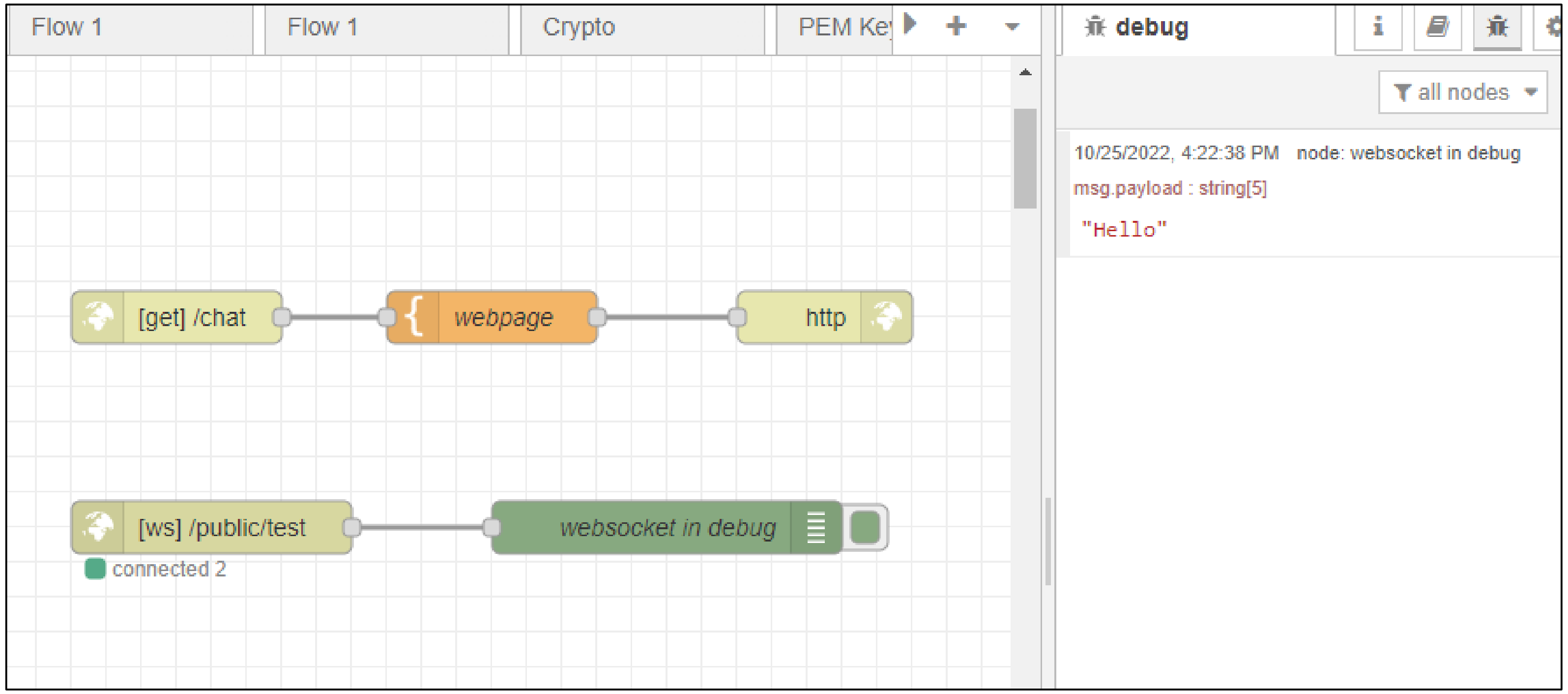

CHAT ROOM
127.0.0.1:1880/chat

chat room

Hello

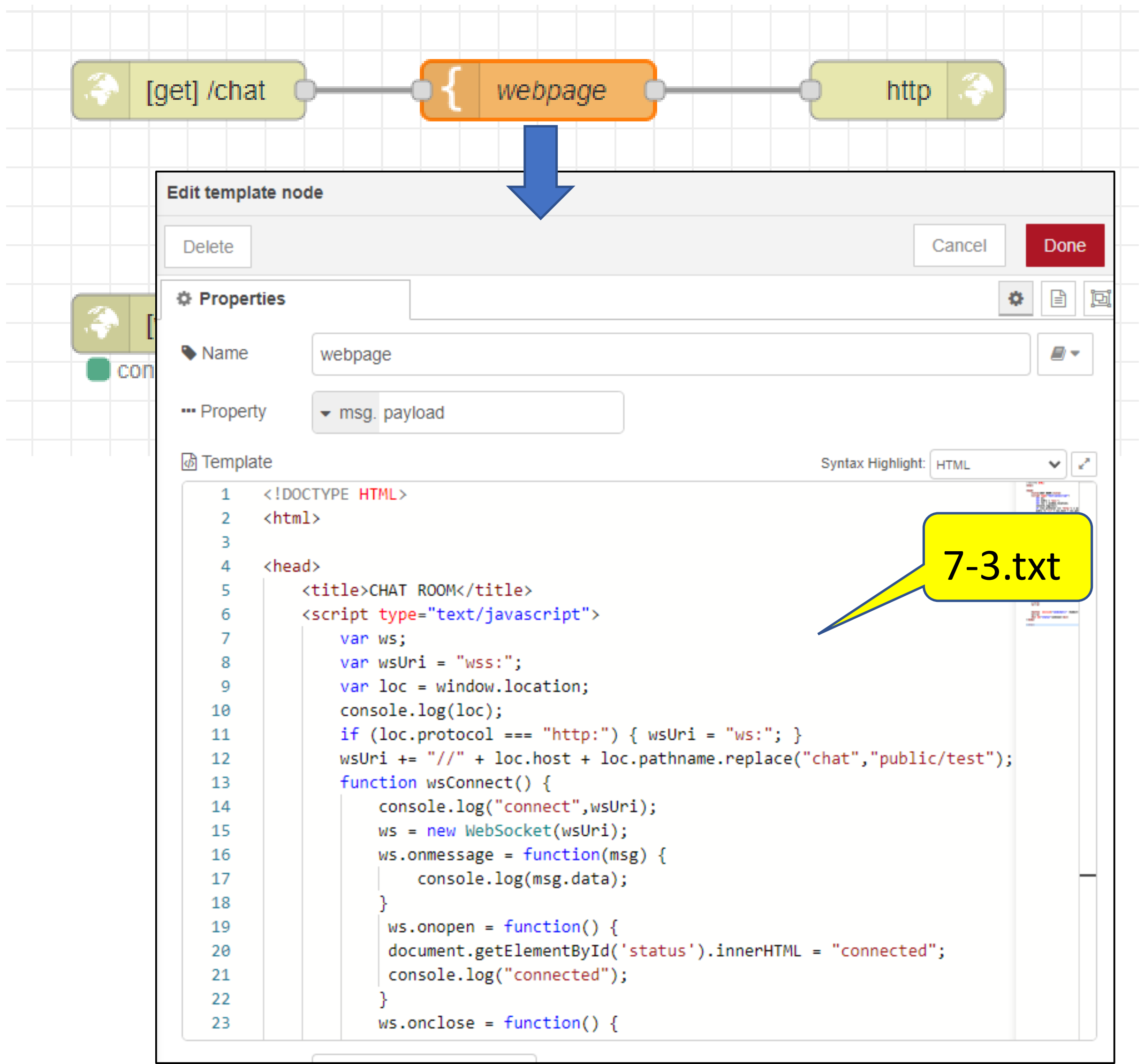
Submit

connected



10/25/2022, 4:22:38 PM node: websocket in debug
msg.payload : string[5]
"Hello"

Edit HTML



Name: webpage

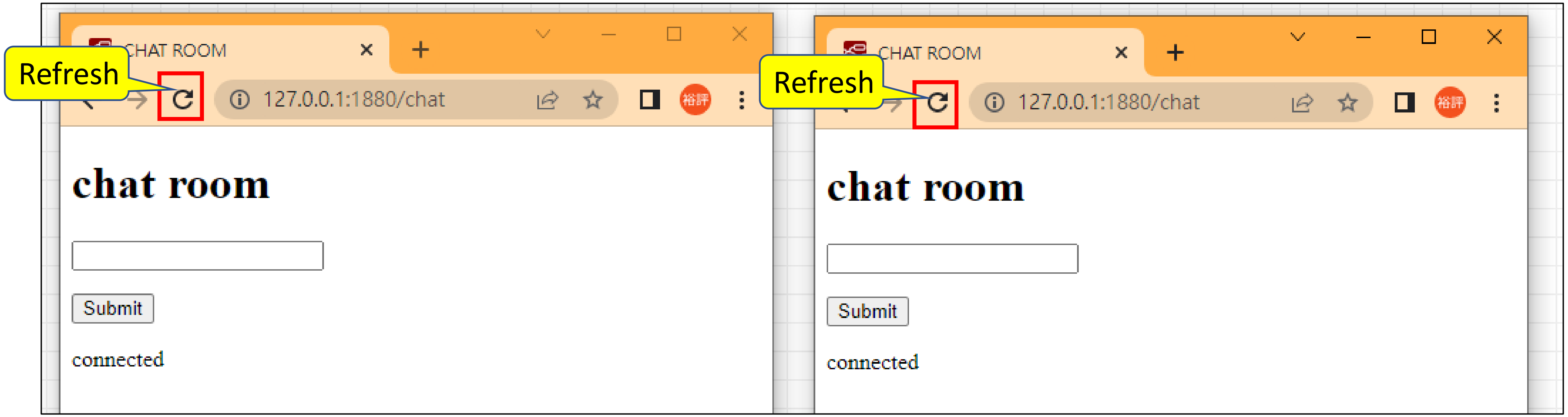
Property: msg. payload

Template

Syntax Highlight: HTML

```
<!DOCTYPE HTML>
<html>

<head>
    <title>CHAT ROOM</title>
    <script type="text/javascript">
        var ws;
        var wsUri = "wss:";
        var loc = window.location;
        console.log(loc);
        if (loc.protocol === "http:") { wsUri = "ws:"; }
        wsUri += "//" + loc.host + loc.pathname.replace("chat","public/test");
        function wsConnect() {
            console.log("connect",wsUri);
            ws = new WebSocket(wsUri);
            ws.onmessage = function(msg) {
                console.log(msg.data);
            }
             ws.onopen = function() {
             document.getElementById('status').innerHTML = "connected";
             console.log("connected");
            }
            ws.onclose = function() {
```

Refresh chat room web pages



chat room

Submit

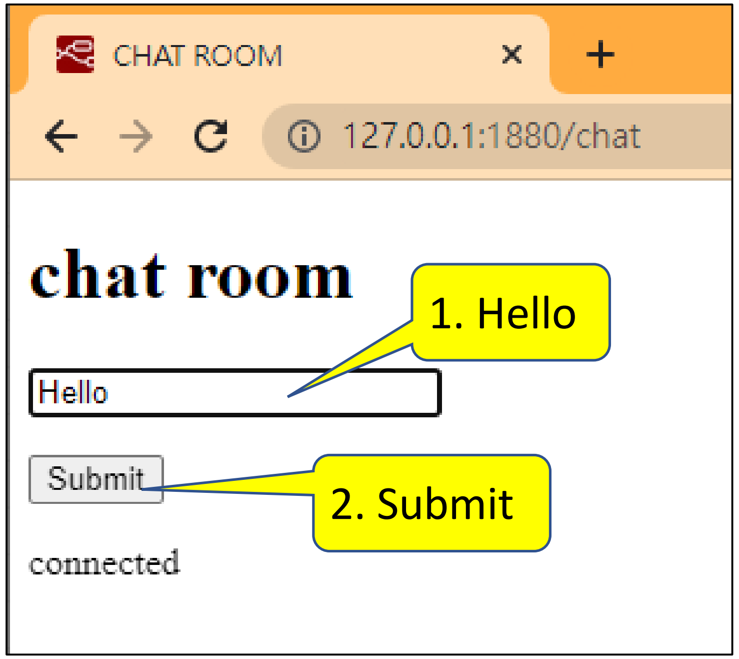
connected

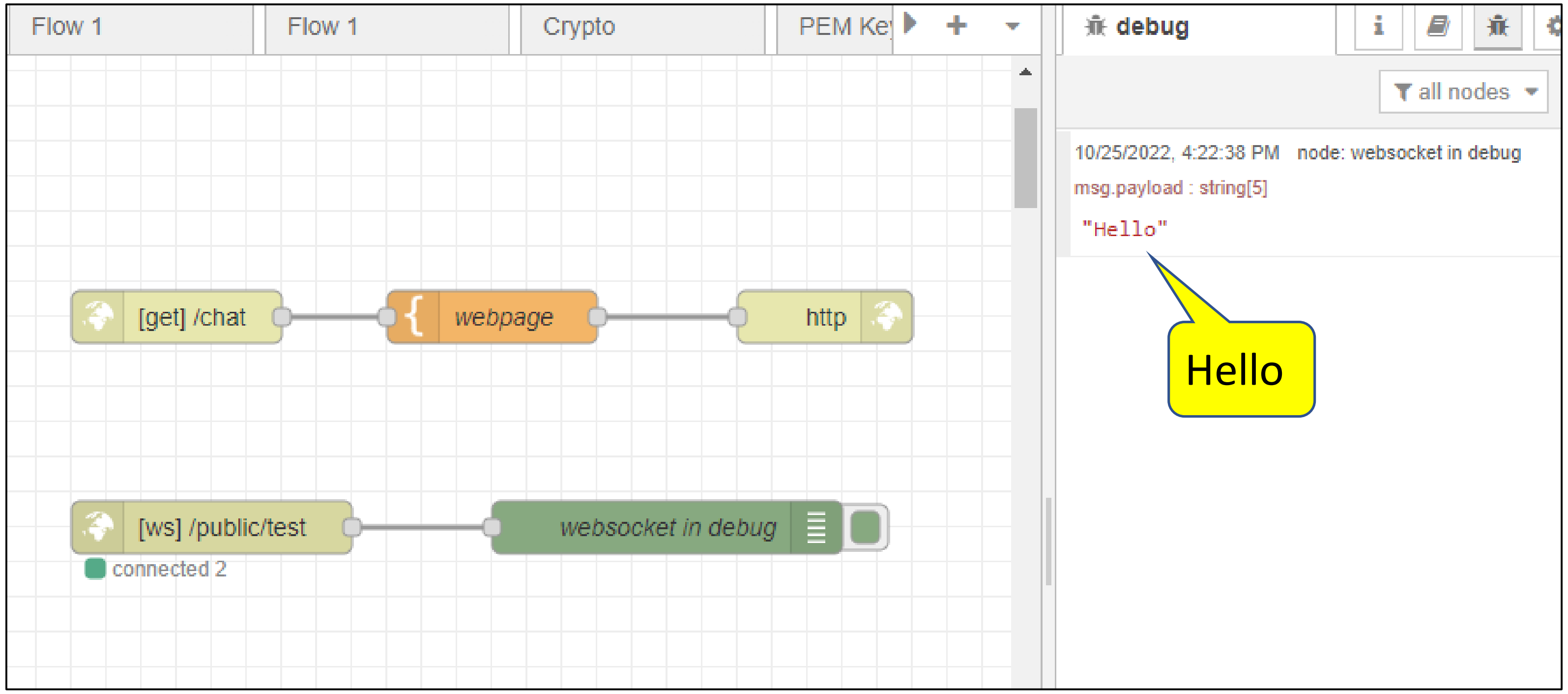
chat room

Submit

connected

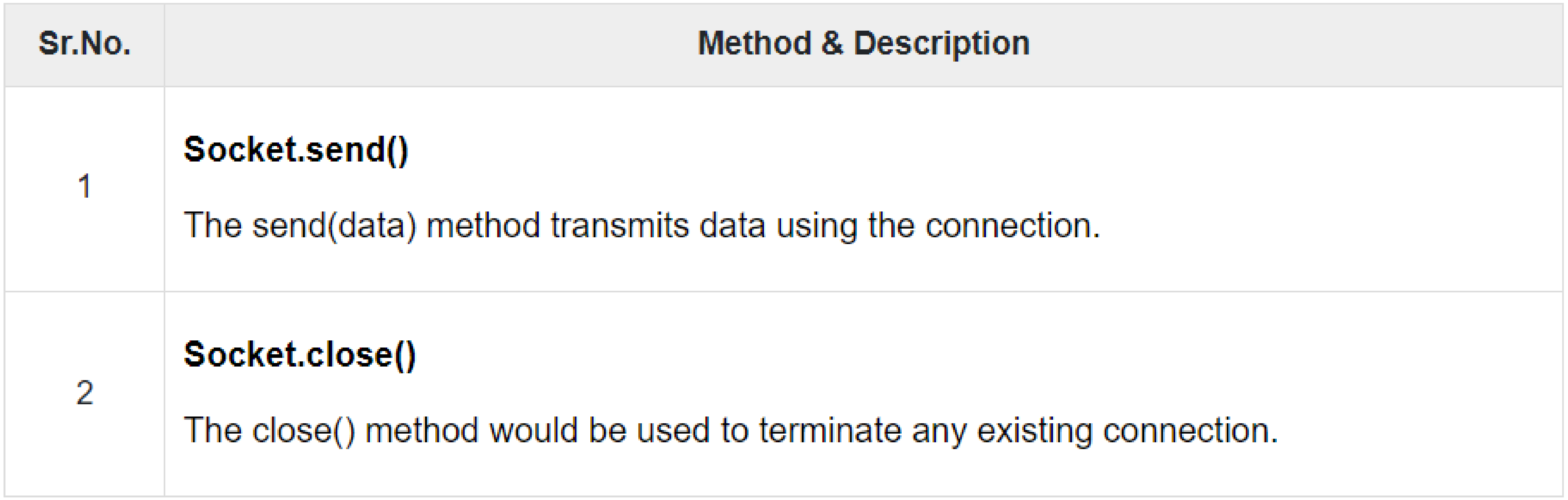
Submit a message





10/25/2022, 4:22:38 PM node: websocket in debug
msg.payload : string[5]
"Hello"

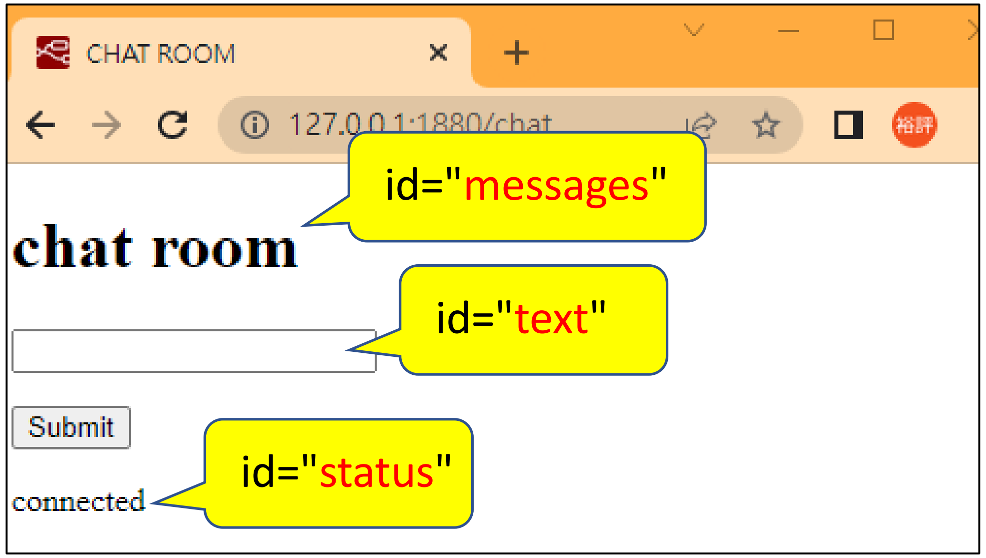
WebSocket Methods



Sr.No.	Method & Description
1	**Socket.send()** The send(data) method transmits data using the connection.
2	**Socket.close()** The close() method would be used to terminate any existing connection.

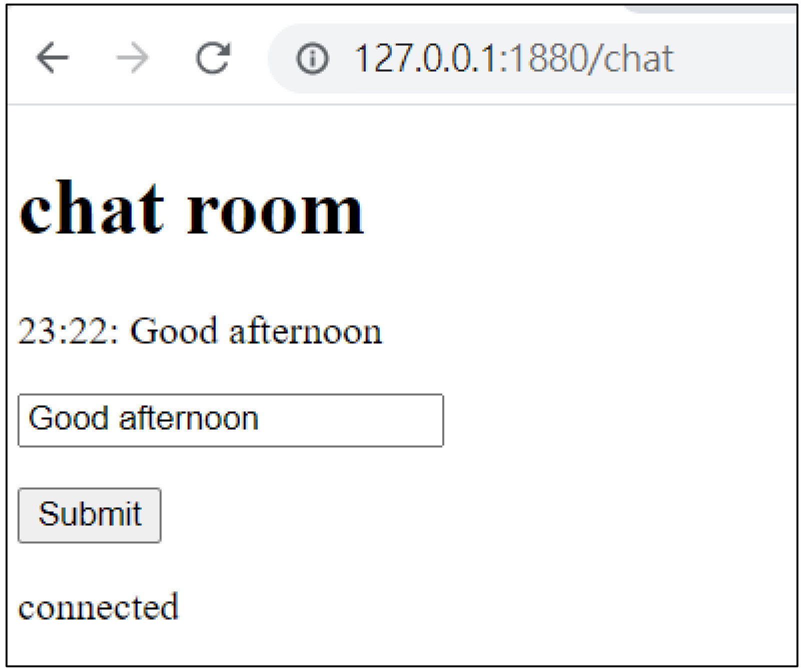
7-3.txt

```
function sendchat() {
    if (ws) {
            ws.send( document.getElementById('text').value);
     }
} //end of sendchat()
```



Exercise 7-4

- Broadcast to the connected clients.



127.0.0.1:1880/chat

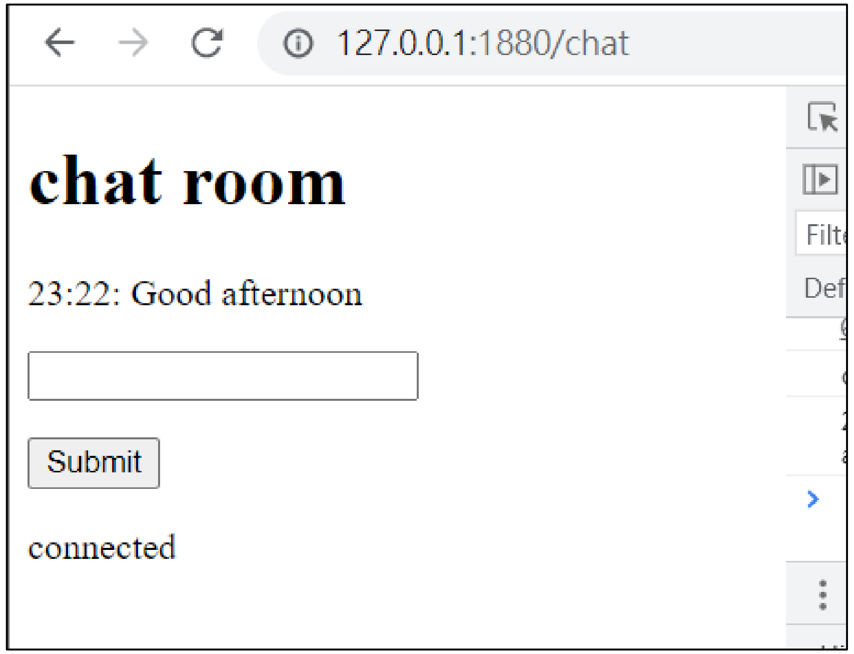
chat room

23:22: Good afternoon

Good afternoon

Submit

connected



127.0.0.1:1880/chat

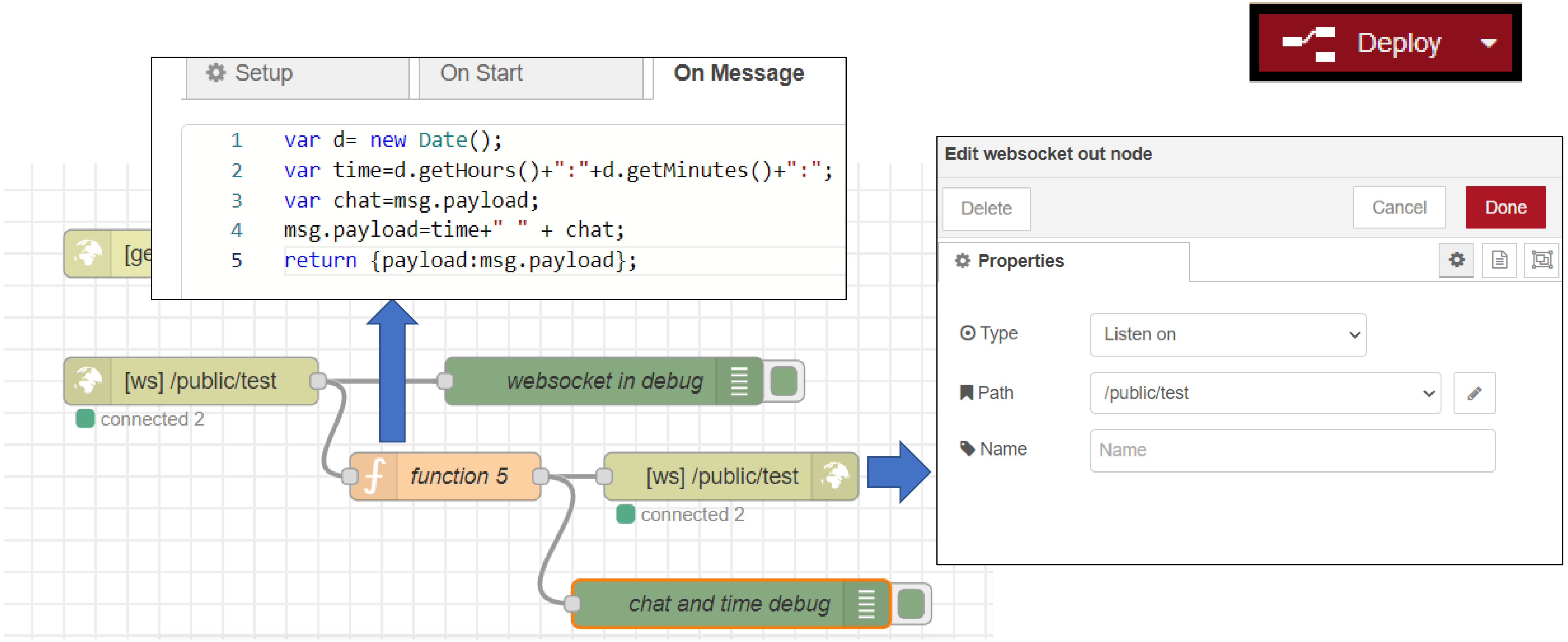
chat room

23:22: Good afternoon

Submit

connected

Add a function node and a websocket out node



```
var d= new Date();
var time=d.getHours()+":"+d.getMinutes()+":";
var chat=msg.payload;
msg.payload=time+" " + chat;
return {payload:msg.payload};
```

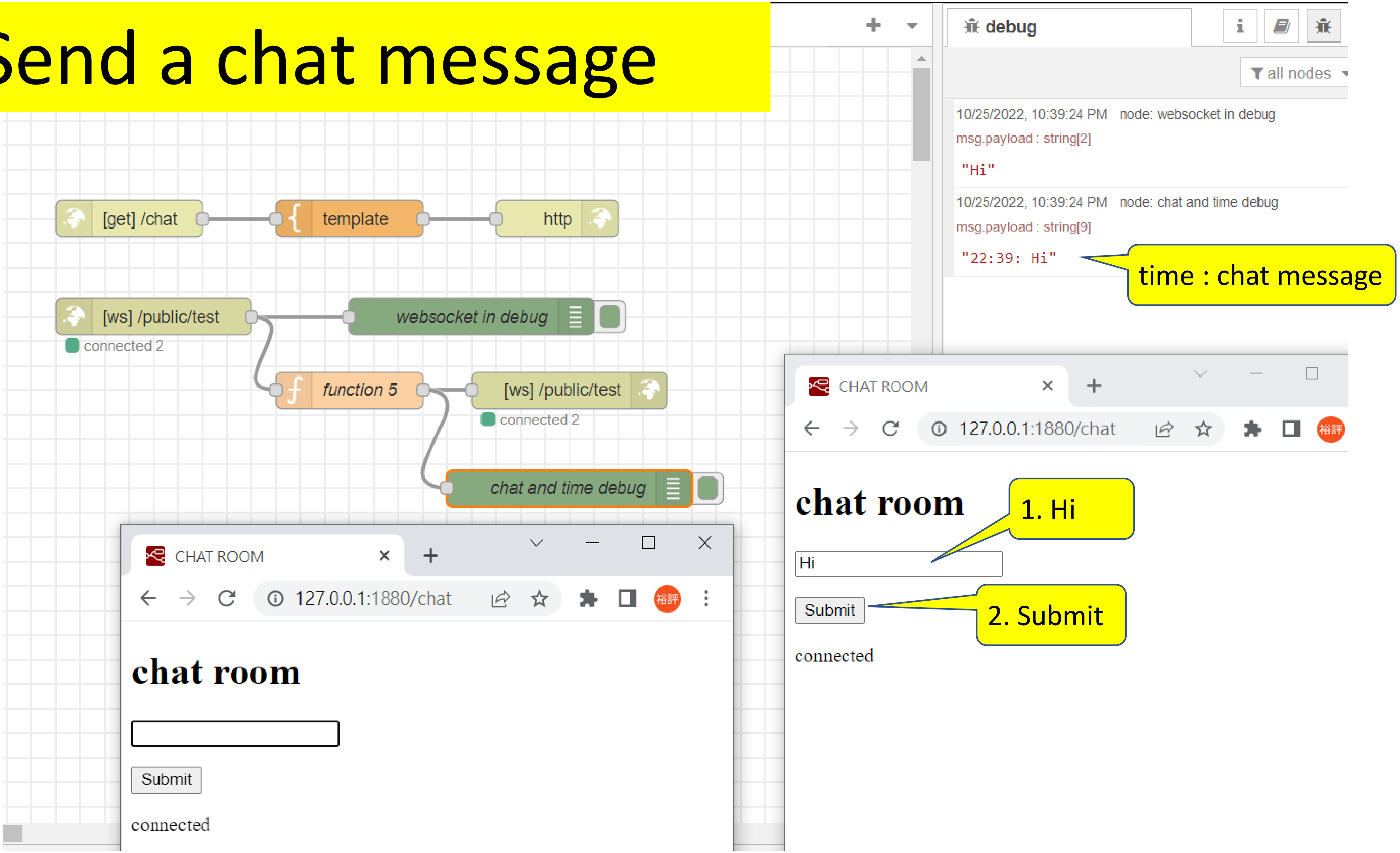
Edit websocket out node

Type: Listen on

Path: /public/test

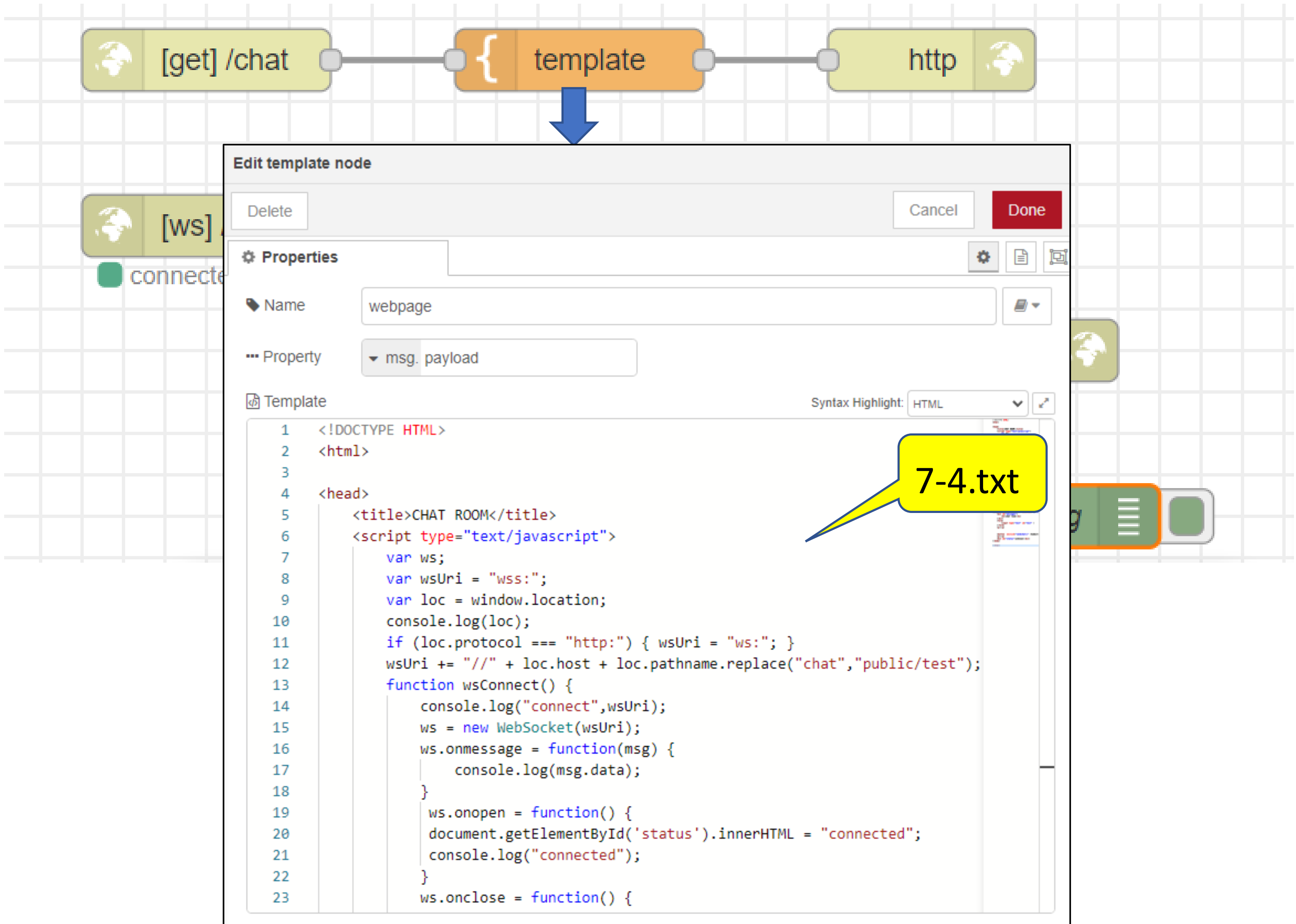
Name: Name

Send a chat message



```
10/25/2022, 10:39:24 PM   node: websocket in debug
msg.payload : string[2]
"Hi"
10/25/2022, 10:39:24 PM   node: chat and time debug
msg.payload : string[9]
"22:39: Hi"
```

Edit HTML



Name: webpage

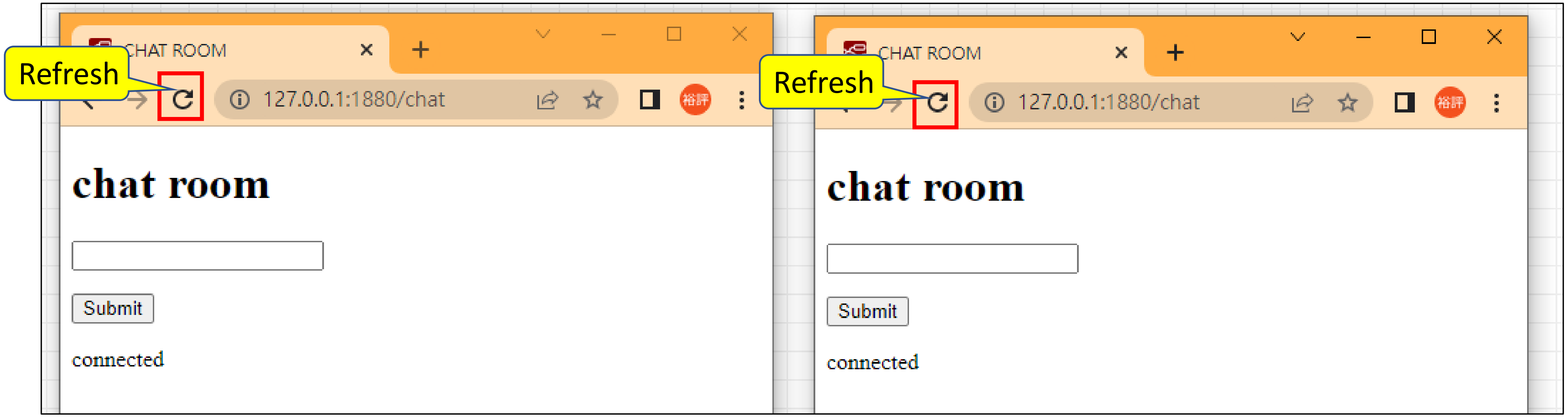
Property: msg. payload

Template

Syntax Highlight: HTML

```
<!DOCTYPE HTML>
<html>

<head>
    <title>CHAT ROOM</title>
    <script type="text/javascript">
        var ws;
        var wsUri = "wss:";
        var loc = window.location;
        console.log(loc);
        if (loc.protocol === "http:") { wsUri = "ws:"; }
        wsUri += "//" + loc.host + loc.pathname.replace("chat","public/test");
        function wsConnect() {
            console.log("connect",wsUri);
            ws = new WebSocket(wsUri);
            ws.onmessage = function(msg) {
                console.log(msg.data);
            }
             ws.onopen = function() {
             document.getElementById('status').innerHTML = "connected";
             console.log("connected");
            }
            ws.onclose = function() {
```

Refresh chat room web pages



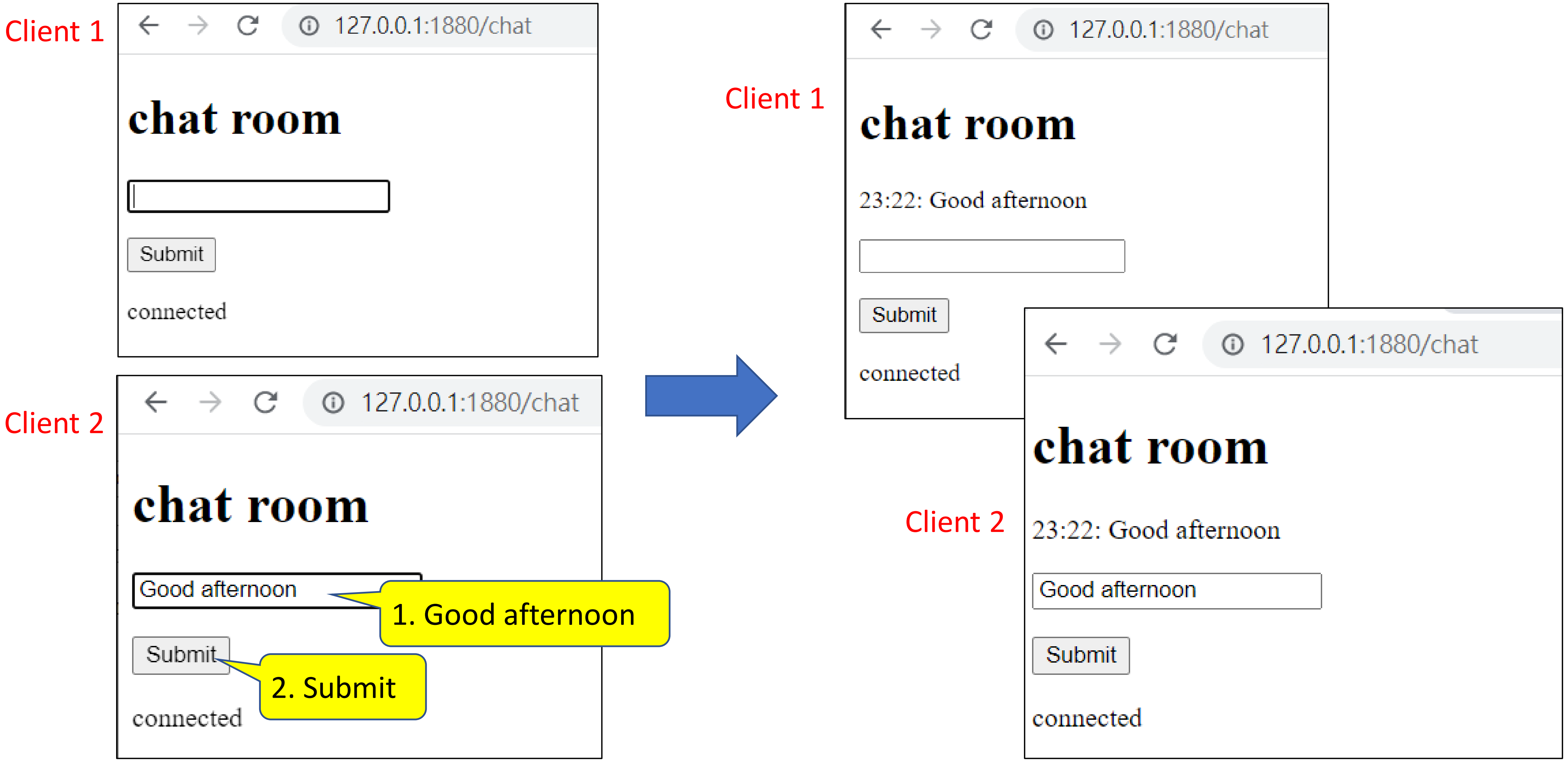
chat room

connected

chat room

connected

Send a chat message



chat room

connected

chat room

Good afternoon

connected

chat room

23:22: Good afternoon

connected

chat room

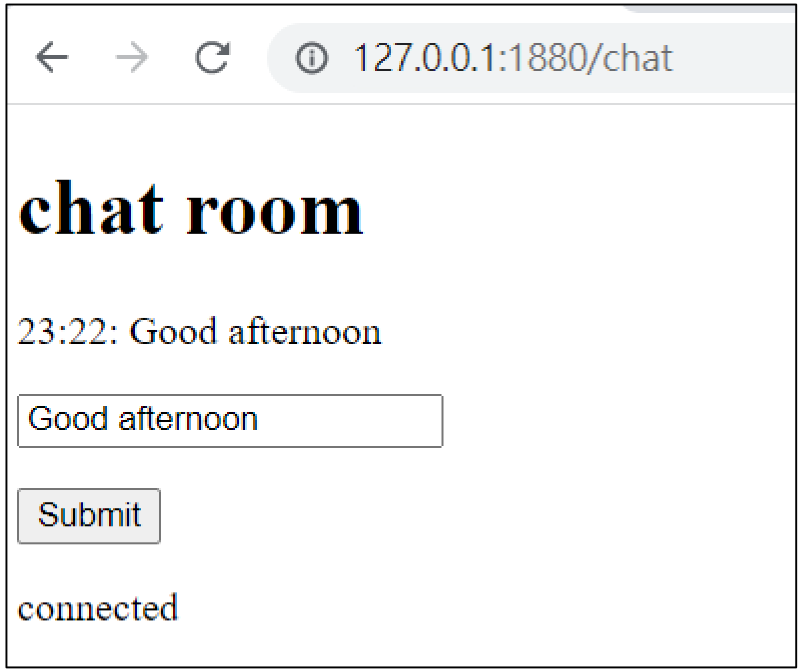
23:22: Good afternoon

Good afternoon

connected

7-4.txt

```
ws.onmessage = function(msg) {
                console.log(msg.data);
                /////
                var data = msg.data;
                var line = "";
                line += "<p>"+data+"</p>";

                document.getElementById('messages').innerHTML =
document.getElementById('messages').innerHTML+line;
                /////
            }
```



127.0.0.1:1880/chat

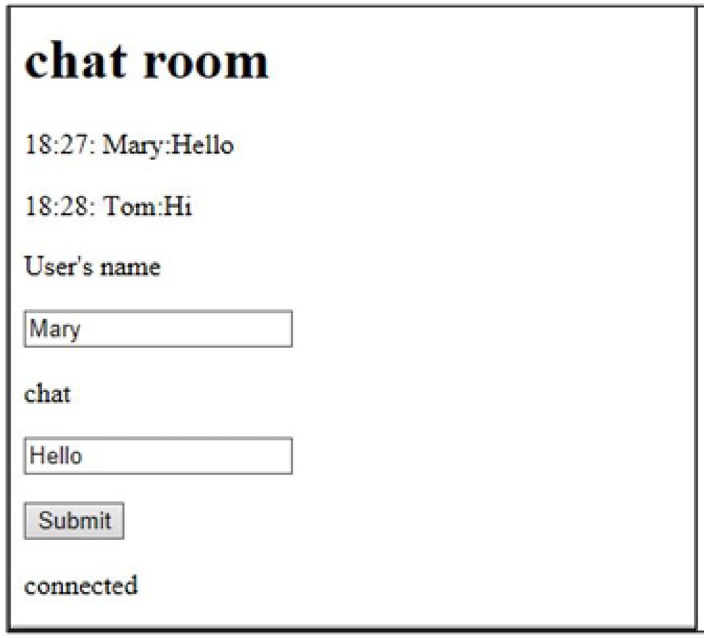
chat room

23:22: Good afternoon

connected

Exercise 7-5

- 請加上一個可以輸入用戶端代號的文字表單，讓聊天室網頁呈現進行聊天者的代號、聊天內容與聊天時間，如下
- Please add another input for user’s name. The time, the name and the chat message would be shown on the chatroom.



chat room

18:27: Mary:Hello

18:28: Tom:Hi

User's name

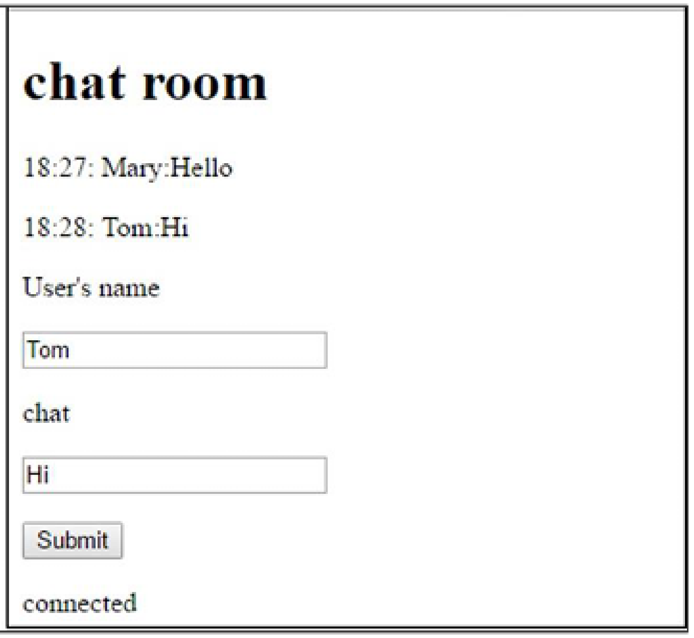
Mary

chat

Hello

Submit

connected



chat room

18:27: Mary:Hello

18:28: Tom:Hi

User's name

Tom

chat

Hi

Submit

connected