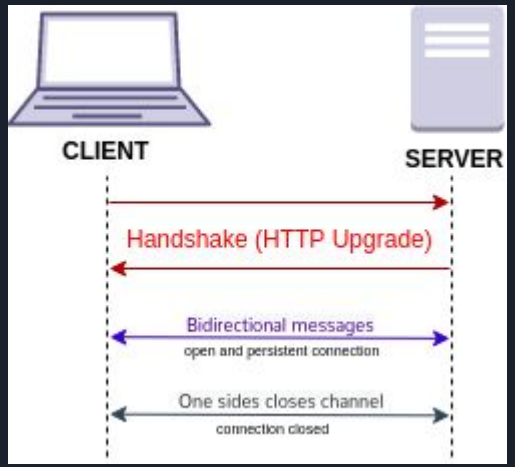
WebSocket

Adrian Velonis

Background



- Protocol over TCP
- Distinct from HTTP, but compatible with it
- Allows for efficient client–server communication
  - Request > Handshake > WebSocket connection
  - Bidirectional/“full-duplex”
  - Faster than HTTP polling
  - Allows byte streams
- URI is `ws` (WebSocket) or `wss` (WebSocket Secure) rather than `http` or `https`
- Connection remains open until client or server terminates (“stateful”)
- Useful for transmitting real-time data (stock trading, social media chats)
  - To process old data with an application, better to use HTTP
- WebSocket API uses interfaces `WebSocket`, `CloseEvent`, and `MessageEvent`
  - `WebSocket`: send/receive data
  - CloseEvent: event sent by WS object when connection closes
  - MessageEvent: event sent by WS object when client receives message from server

Methods

- WebSocket.close(int code, String reason) – Closes the connection.
  - code (optional): a close code, typically 1000
  - reason (optional): reason for closure

- WebSocket.send(data) – Enqueues data to be transmitted.
  - data (required): the data to send to the server. Permitted types:
    - USVString
    - ArrayBuffer
    - Blob
    - ArrayBufferView

serv.js

Static page server:

```
const express = require('express')
const app = express()
const port = 30012

app.listen(port, function(error){
    if(error) throw error
    console.log("Server created Successfully")
})

app.use("/", express.static(__dirname))
```

websock.js

WebSocket server:

```
let WebSocketServer = require('ws');
let wss = new WebSocketServer.Server({ port: 30011 });
let messageCounter = 0;
let connectionID = 0;
wss.on('connection', function (ws) {
    let thisConnectionID = connectionID++;
    let connectionCounter = 0;
    ws.on('message', function incoming(message) {
        messageCounter++;
        connectionCounter++;
        console.log('rec %s %d %d', message, thisConnectionID,
connectionCounter);
        ws.send(`echoooo ${message} ${messageCounter} ${thisConnectionID}
${connectionCounter}`);
    });
    ws.send(`Setup ${messageCounter}`);
});
```

wser.html

Static web page to send/receive WebSocket data:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
    </head>
    <body>
        <table>
            <tr>
                <td id="time">AA</td>
                <td id="count">BB</td>
            </tr>
        </table>
        <br>
        <button id="ss" type="submit" onclick="sendMessage();">Send
Message</button>
        <button id="ss" type="submit" onclick="closeSocket();">Close
comms</button>
        <button id="ss" type="submit" onclick="openSocket();">Open
comms</button>
    <script>
    let ws = null;

    function openSocket() {
        console.log("Open socket button hit");
        if (ws==null) {
            ws = new WebSocket("ws://165.106.10.170:30011");
            ws.onopen = function(e) {
                console.log("[open] Connection established");
                ws.send("My name is John");
            };

            ws.onmessage = function(event) {
                console.log(`[message] Data received from server:
${event.data}`);
            };

    ws.onclose = function(event) {
      if (event.wasClean) {
        console.log(`[close] Connection closed cleanly,
code=${event.code} reason=${event.reason}`);
      } else {
        // e.g. server process killed or network down
        // event.code is usually 1006 in this case
        console.log('[close] Connection died');
      }
    };
    ws.onerror = function(error) {
      alert(`[error] ${error.message}`);
    };
        } else {
            console.log("Websocket already opened");
        }
    }
    function closeSocket() {
        if (ws!=null) {
            ws.close();
            ws = null;
        }
    }
function sendMessage() {
    console.log("sending hello2");
    ws.send('Hello!!!');
}
    function jsonToString(jjj) {
        let rtn = "";
        for (const [ky,v] of Object.entries(jjj)) {
            rtn = `${rtn} ${ky}:${v}`
        }
        return rtn;
    }
    </script>
    </body>
</html>
```

Sources

- Wikipedia: WebSocket
- Mozilla: The WebSocket API
- GeeksForGeeks: What is WebSocket?