Report on IT Lab- Assignement 2

Abstract:
A re – implementation of Websocket module(partially) using Native Node.js modules.

Name: Priyanshu Tirkey
Roll no.: 001710501005
BCSE Final Year (2017-2021)

Hosting page.

- Observation: on reload the a new get request is created.

  On a new request a new Websocket connection is created so I thought of creating a static page.

  Popular method to create static page.

```
let http_server = http.createServer((req,res) => {
    res.writeHead(200,
        {"Content-Type" : "text/html"});
    fs.createReadStream('index.html').pipe(res)
});
```

```
const port = 8080;
http_server
    .listen(port, ()=> console.log(`Server running at http://localhost:${port}.`));
```

Keeping all the connection states

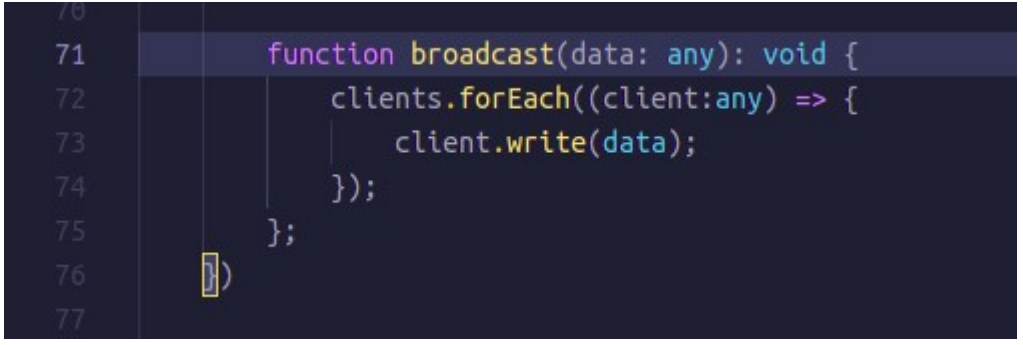
- Storing the connection states

```
11  //Storing client as they connect
12  let clients: Socket[] = [];
13
```

```
54
55          clients.push(socket);
56
```

Broadcasting.

- Sending payload one by one to each client.



Defining event callbacks. (for Http object)

- ‘upgrade’ event for the server.

```
18
19  http_server
20      .on('upgrade', (req, socket) => {
21          if(req.headers['upgrade'] !== 'websocket'){
22              socket.end('HTTP/1.1 400 Bad Request');
23              return;
24          }
25
26          /* Read the websocket key provided by the client "acceptKey"                    --1
27              Generate the response value to use in the response "generateAcceptValue"     --2
28                  Write the HTTP response into an array of response lines: "responseHeaders" --3
29                      Write the response back to the client socket, being sure to append two
30                      additional newlines so that the browser recognises the end of the response
31                      header and doesn't continue to wait for more header data             --4
32          */
```

If the upgrade request is not for websocket return with status code 400.

- Note that we have also created a ‘listen’ event for hosting static page.(slide 2)

Events callback for ‘socket object’

- ‘data’ event

```
56
57      socket.on('data', (buffer:Buffer) => {
58          const message = parseMessage(buffer);
59          if (message) {
60              // For our convenience, so we can see what the client sent
61              console.log(message);
62
63              // socket.write(constructReply(message));
64              broadcast(constructReply(message))
65          } else if (message === null) {
66              console.log('WebSocket connection closed by the client.');
67          }
68      });
69
```

Storing payload .

- Note: masking is done by browser automatically.

  We just store the data and browser does the masking part.

  - Categorize the data as text.
  - Set the payload(according to the length of payload).

```
97        // Write out the first byte, using opcode `1` to indicate that the message
98        // payload contains text data
99        buffer.writeUInt8(0b10000001, 0);
100       buffer.writeUInt8(payloadLength, 1);
103     // Write the length of the JSON payload to the second byte
104     let payloadOffset = 2;
105     if (lengthByteCount > 0) {
106       buffer.writeUInt16BE(jsonByteLength, 2); payloadOffset += lengthByteCount;
107     }
108
109     // Write the JSON data to the data buffer
110     buffer.write(json, payloadOffset);
111     return buffer;
```

Getting the payload.

- Data sent by browser will always be masked.

```
199     }
200     let maskingKey = 0x0000;
201     if (isMasked) {
202         maskingKey = buffer.readUInt32BE(currentOffset);
203         currentOffset += 4;
204     }
```

- Unmasking the data.

```
216     for (let i = 0, j = 0; i < payloadLength; ++i, j = i % 4) {
217     // Extract the correct byte mask from the masking key
218         const shift = j == 3 ? 0 : (3-j) << 3;
219         const mask = (shift == 0 ? maskingKey : (maskingKey >>> shift)) & 0xFF;
220         // Read a byte from the source buffer
221         const source = buffer.readUInt8(currentOffset++);
222         // XOR the source byte and write the result to the data
223         data.writeUInt8(mask ^ source, i);
224     }
```