# Computer Programming – CS 6011
# Lecture 14: Web Sockets

Fall 2023

# Topics

- Web Sockets
  - Web Calculator Application – In Class

# Recap: Client – Server Java implementation

- Server side
  - Server waits for client connections – on a specific port Number
  - Create a **Server Socket** and wait for the Client requests – the constructor takes a port Number
  - Use sockets I/O streams to perform communication with Client
  - Close sockets
- Client side
  - Client connects to a server – that needs an IP and port
  - Create a **Socket** Object – the constructor takes an IP address and a port Number
  - Use sockets I/O streams to perform communication with the server
  - Close the socket when done

# Chat Client Communication Approach

- Each time a user enters a new message, how could we update the web page?
- Using AJAX / XMLHttpRequest?
  - For AJAX, you (the client) must make an *HTML* request…
  - Ask the server for any new messages…
  - How often? Every second? Every minute?
  - This method of communication is called *polling.*
    - Is there anything new?
    - Is there anything new?
    - Is there anything new?
    - …

# Communication Method For Chat App…

- Can we do this with AJAX?
  - No – AJAX is a client-side approach.
- In an ideal world, how would we know there are new messages?
  - The *Server* would tell us (because it knows when it receives a new message).
  - We need a method for **bi-direction** communication that allows two-way communication whenever either side wants to talk to the other.
    - Clients and Server can send messages back and forth – whenever they want.
    - A connection that stays open until closed by one of the endpoints.
    - Built on top of HTTP (at least to start) – So that all web servers are capable of at least receiving the request.

# Web Socket

- Provide a way to exchange data between browser and server
- To open a websocket connection, we need to create new WebSocket using the special protocol ws in the url:

```
let socket = new WebSocket("ws://serverURL");
```

- Once the socket is created, there will be 4 events:

```
open
message
error
close
```

Then: `socket.send(data)` to send data.

# Web Socket Approach for the chat App

- Once a user types in a message and clicks send, we will send the message to the server.
- What do we expect will happen then (almost immediately)?
  - The Server should send back a message to us (and all other clients) with the message we just sent.
- Sending and receiving messages:

```
sendMessage( data );  // Send message to server.
msgData = receiveMessage( );  // Wait for the server's response
```

- When do we send a message?
  - When the user presses “enter” or clicks the “send” button.  In other words, based on an event.
- When do we display a message?
  - `onmessage` // When the onmessage is called - again based on the event of a message arriving.

# Web Sockets – JavaScript Example

```
let ws = new WebSocket( "ws://localhost:8080" )
ws.onopen = handleConnect;
ws.onclose = handleClose;
ws.onerror = handleError;
ws.onmessage = handleMessage;
ws.send( message ) // To send the message to the server.
ws.close(); // When we are completely done.
```

- `handleConnect, handle Close` are just functions we write:

  `function handleClose( ) { // clean up, etc }`
- URL: is the address of the server

# WebSocket Handshake


- WebSockets talk over port 80 (or 443) directly to the web server.
- They send header information (Similar to HTTP requests /responses).
- Then they switch to their own protocol (“language”)

▶ [illegible]

GE[illegible]/[illegible]H[illegible]/1.1

H[illegible]:[illegible]:8000

[illegible]:[illegible]

C[illegible]:[illegible]

[illegible]-[illegible]:[illegible]G[illegible]H[illegible]B[illegible]B[illegible]25[illegible]==

[illegible]-[illegible]:13

▶ [illegible]

H[illegible]/1.1 101 [illegible]

[illegible]:[illegible]

C[illegible]:[illegible]

[illegible]-[illegible]-A[illegible]:13[illegible]+[illegible]=

# WebSocket Messages

- Once the initial WebSocket handshake has occurred (using standard HTTP headers / protocol / socket) – the server switches to using an actual WebSocket and a different message protocol.
- Binary message format (binary data)
- Have to fit all header fields into the 2 byte header
  - Use bit manipulation like we have done previously.
- More on this later.

# Thursday Assignments

- HW 7 – Server-side Calculator (In-class)
- Midterm Tomorrow (Friday)
  - 1 page – Handwritten notes
- Have you turned in every lab (Up to **Lab 8** ) / homework (Up to **HW 7**)?
  - If not, turn them in as soon as you can.
  - If so, do you have grades for them?

# Server-Side Calculator Client

- Read the assignment and think about how you might approach doing it.
  - 10 Minutes – then we will implement the solution as a class.
- localhost:8080/calculate?x=5&y=9
  - Query string: Everything after the “?”.
- Run curl to verify server is running and to see result.
- You can use WireShark to see the messages being sent.
- Cross-Origin Resource Sharing (CORS) Errors
  - You are not allowed to load new data from a different server then you loaded the page from.
  - By default, AJAX requests must go back to the server you are talking to.
- Using the class implementation as an example, you must implement your own Web Calculator Client and commit it to git.