WebSockets

HTTP / AJAX / WebSockets / Socket.io

- Recall that HTTP is request/response protocol
- AJAX allows you to do HTTP request/response within a page, but it is still request/response
- A client always has to initiate communication.
  - I.e. Make the request
- What if you want the server to be able to "push" information down to the client, for example:
  - New email
  - New news
  - Updates to a shared chat / drawing canvas
  - Game events

Work-Arounds

- Workarounds have been devised
- E.g. Polling
  - Client continuously polls the server
- E.g. Long Polling
  - Client sends a request, which the server does not respond to until it has something to send
  - If something comes along to send, it completes the response and the client will immediately send a new request to hold on to for later
  - If the request times out, the client sends a new response.
  - Analogy (from the days of postal mail)…
    - Cheap friend can't afford stamps.
    - Send friend stamp, he uses it to send you a letter
    - When you receive message, immediately send him another stamp for the next time he wants to send a letter.

WebSockets

- Provides for true two-way ongoing communication between a client and server.
- Each side can "send" messages.
- Each side has listener "on message"
- WebSockets are not restricted by the Same Origin Policy
  - It is up to the server to check the Origin header and decide whether it should reply or not.

- Note:
  - Some old browsers don't implement WebSockets

WebSockets – On the browser

```
// Create a new WebSocket
var wSocket = new WebSocket("ws://www.example.com/socketserver")

// When the WebSocket is connected and ready
wSocket.onopen = function (event) {
   // Send a message to the server
   wSocket.send("Initial message to the server.");
};

// Handler for when a message arrives from the server
wSocket.onmessage = function (event) {
   console.log(event.data);
};
```

WebSockets – Server-side

- Require a WebSockets module
  - ws is very popular
  - https://www.npmjs.com/package/ws
- Create a new WebSockets Server
- Define handlers for
  - on connection
  - on message
- And *send* messages to the client

Socket.io

- Library for full-duplex communication between clients and servers
- Built on engine.io, socket.io provides an abstraction above whatever transport methods are available in the browser:
  - WebSockets
  - Long Polling

Socket.io

- Also provides a simple abstraction for event-based programming ***across the server and multiple clients***
- Clients can
  - emit events back to the server
  - listen for events from the server
- Server can
  - listen for events from clients
  - emit events back to the client
  - emit events to other clients
  - broadcast emit events to all clients

Socket.io role playing

- At each table, create a game
- An event is simply a word
  - It can optionally be accompanied with a short phrase
- One person is the Server who can:
  - listen for events from clients
  - emit an event to one client (point to one and say…)
  - broadcast emit events to all clients (point with both hands and say…)
- The rest at the table are Clients who can
  - emit events to the server
  - listen for events from the server
- Clients or the Server can generate events at any time (depending on your game-play)
- Clients cannot emit events peer-to-peer (Server must relay)

Socket.io

- on(event, function(parameter-object) { … }
- emit(event, parameter-object)
- events
  - connection
  - disconnect
  - *developer-defined events*

Server (node) side

- socket.emit(…)
  - emits back to the client
- socket.broadcast.emit(…)
  - emits to all connected clients except the socket that starts it

countPlayers

- Demo
- Code walkthrough

Homework

1. Add a welcome message to countPlayers
   - When a new player connects
   - Server emits a "welcome" event to it
     - Data of the event is {message: "Welcome player n"}
       - Where "n" is the ordinal number of the player who has joined
2. Create a simple app in which users collaborate / compete / communicate.
   - Actions by each user is broadcast to all users
   - Periodically (e.g. every one second) server broadcasts something to all users