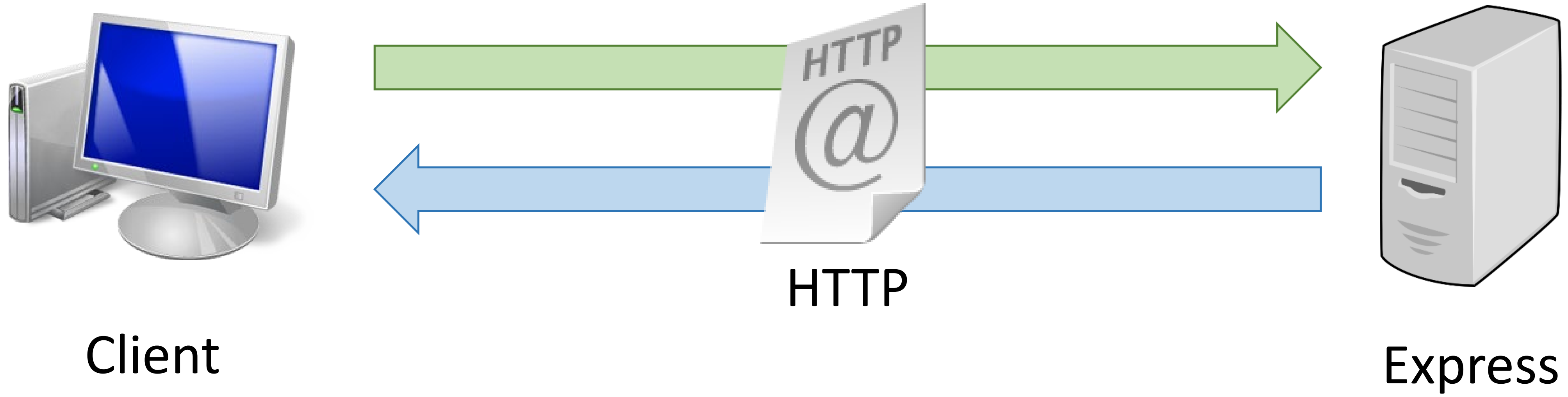
Day 36

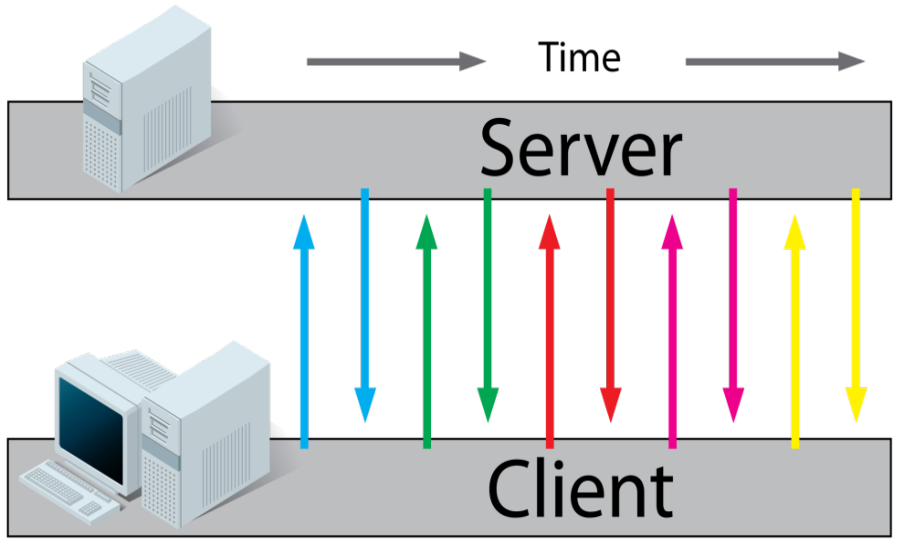
HTTP - Request/Response Protocol

- Message exchange pattern: in-out
- Client has to make a request for server to respond
  - Server cannot send unsolicited responses to client
- Cannot write realtime application: eg watching stock prices, games

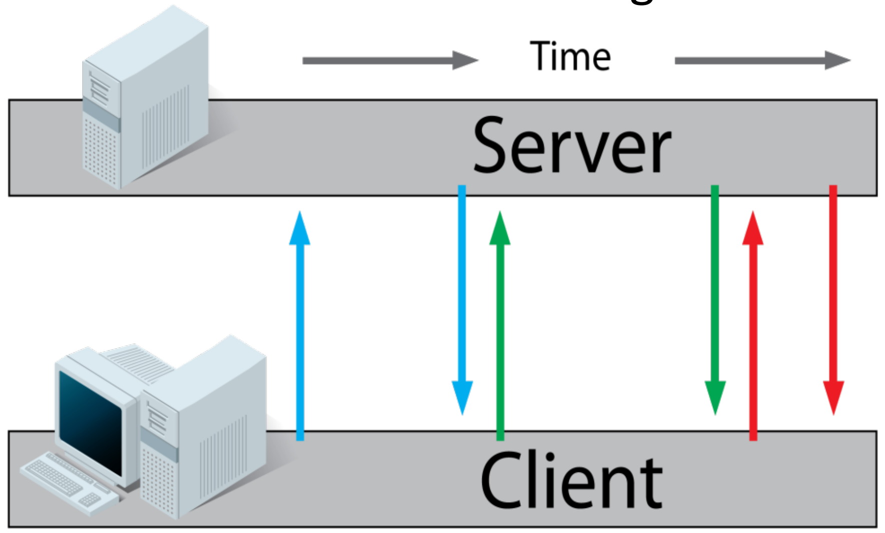


AJAX Polling and Long Poll

- AJAX polling is the technique where the client sends HTTP request at regular interval to the server
  - If there are any data, the server will return the data
  - Otherwise the server will return a no-op and closes the connection
- Long poll is a variation of AJAX polling
  - Behaves like AJAX polling if the server has data
  - But if the server does not have any data, the server will hold on to the connection until
    - New data is available
    - Passes a certain duration (timeout) where the server will send a no-op and closes the connection



AJAX Polling

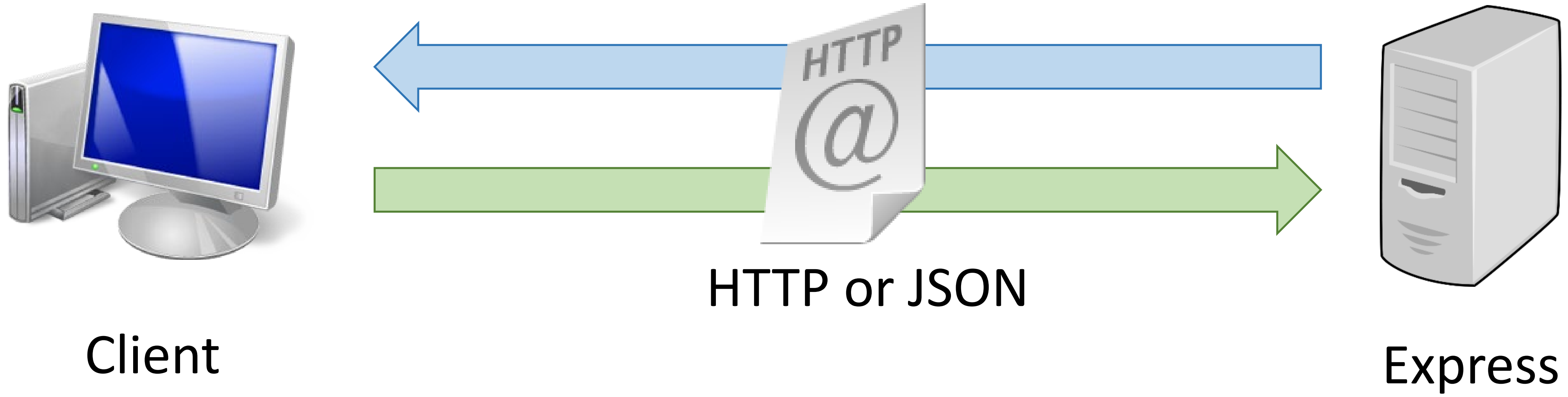


Long polling

Images from: http://stackoverflow.com/questions/11077857/what-are-long-polling-websockets-server-sent-events-sse-and-comet

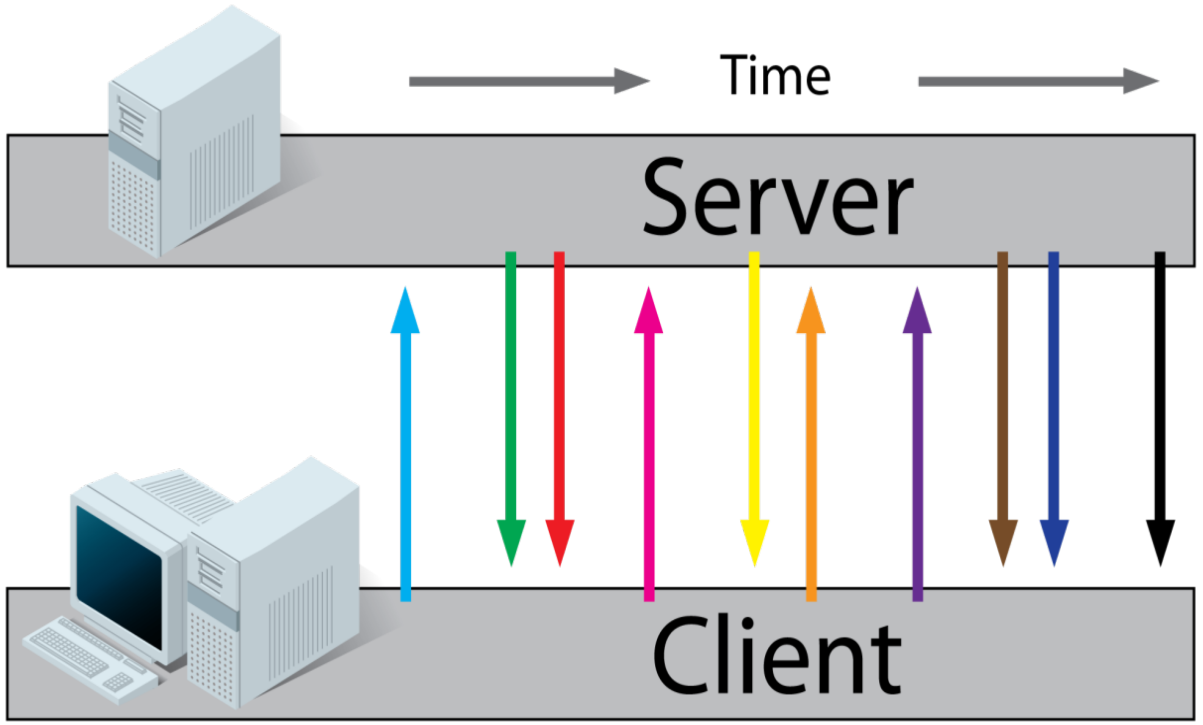
Server Push

- Ability to the server to send data to the client without the client requesting for it
- Message exchange pattern: out-in or out-only
- Use cases includes notification for breaking news, stock price alert, etc.



WebSocket

- It is a bi-directional socket connecting a client to a server
  - Eg like a TCP connection
- Its implemented in the browser
  - Connection is persistent until closed by the user unlike HTTP protocol
  - The server can push/send data whenever it like
  - Think of it as client/server with the browser as the client
- Works with existing HTTP friendly proxies and firewall
  - Unlike TCP/IP sockets
- Has lower overhead and faster
  - Than AJAX and Comet



Images from: http://stackoverflow.com/questions/11077857/what-are-long-polling-websockets-server-sent-events-sse-and-comet

WebSocket

- WebSocket is considered as part of the HTML5 platform
  - Is defined in the communication part of the HTML5 specification
- WebSocket connections are established by upgrading from a HTTP GET request
- Once a connection has been established, the client and server can communicate in full-duplex
  - Data are send between the client and server in frames
  - Data can be text or binary

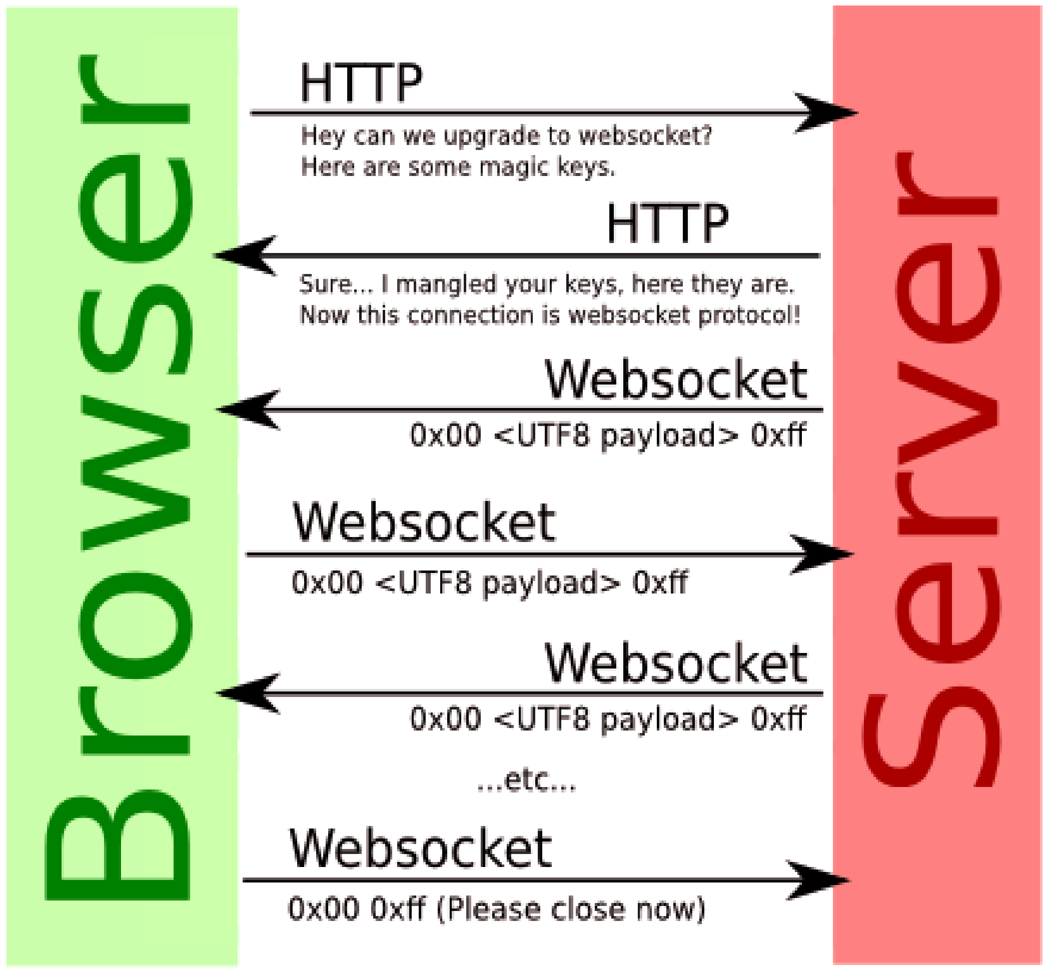
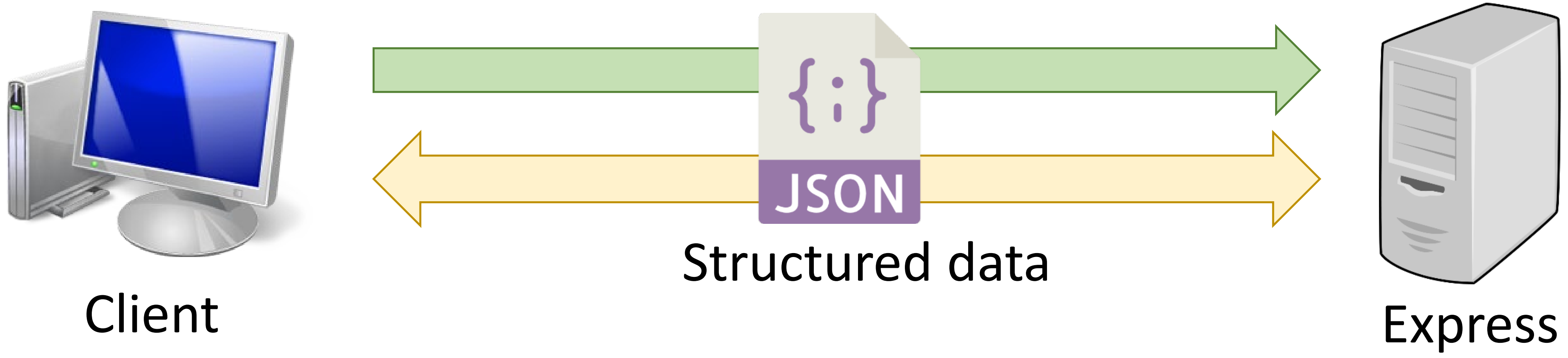

Browser
Server
HTTP
Hey can we upgrade to websocket?
Here are some magic keys.
HTTP
Sure... I mangled your keys, here they are.
Now this connection is websocket protocol!
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
...etc...
Websocket
0x00 0xff (Please close now)

Image from http://uwe.stevelacey.net/atwd/assignment2/images/xwebsocket-lifecycle.png.pagespeed.ic.weBP3HlFSb.png

WebSocket Application



- Client connects to a WebSocket endpoint
- Endpoint may received parameters via query string or route parameters
  - Eg `GET /chat/general?name=fred`

    Route parameters

    Query string
- Client and server exchange structured data
  - Usually JSON

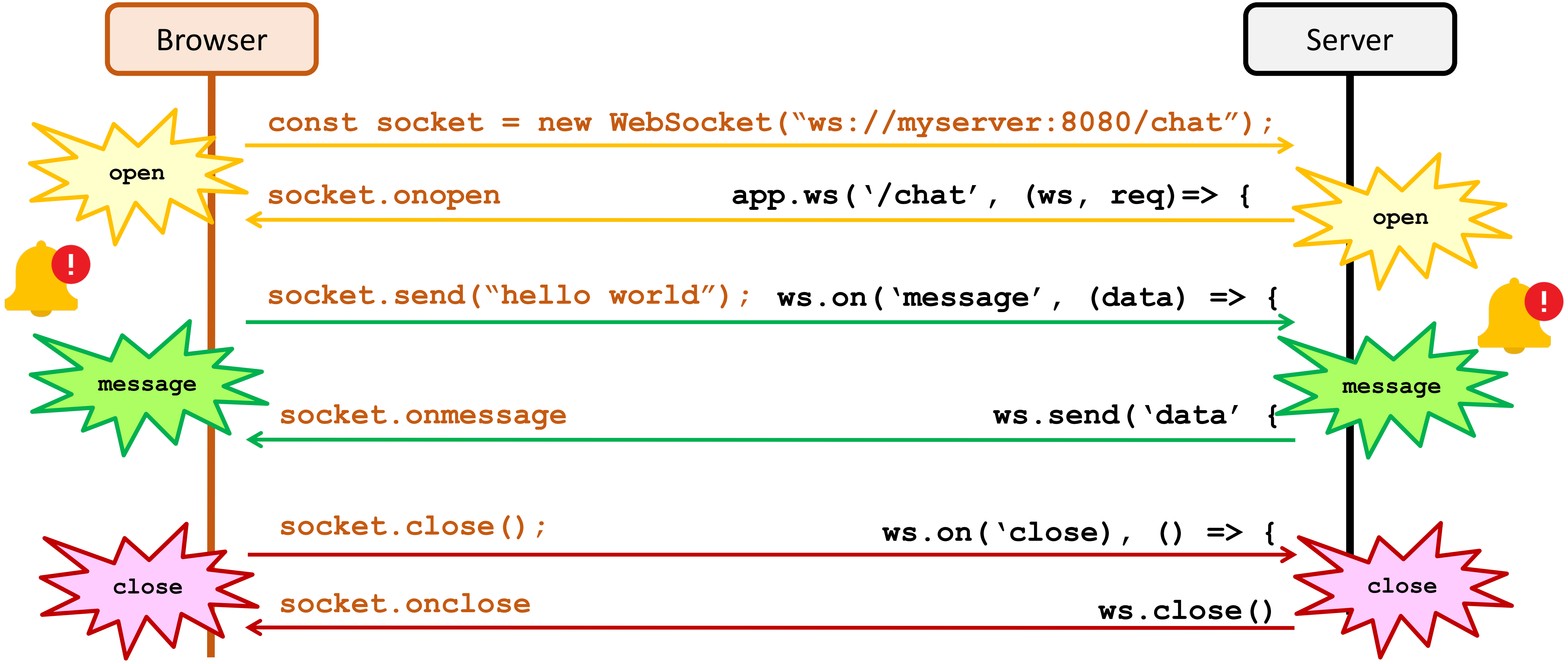
WebSocket on Express

- express-ws is an Express middleware that add WebSocket support to Express application

```
npm install --save express-ws
```

- Creates WebSocket endpoint on Express route

JavaScript WebSocket API Illustrated



Example - Chat

```
const express = require('express');

const app = express();
const appWS = require('express-ws')(app);

app.ws('/chat/:topic', (ws, req) => {
   const topic = req.params.topic
   const name = req.query.name;

   if (!name)
      return (ws.close(1000, 'Missing name'));

   ws.on('message', (payload) => {
      const data = JSON.parse(payload);

   });

})
```

Add WebSocket Express middleware to an Express application

Use `ws` to listen to WebSocket routes

The WebSocket connection is passed to the middleware. This is the active connection between the server and the client

Read the query string and route params as per Express application

Data are passed as string. If using JSON as transfer format, then parse string to JSON

Add a listener to the message event. Listener will be called whenever client sends

WebSocket Connection

WebSocket connection

```
app.ws('/chat/:topic', (ws, req) => {
```

- **`ws`** represents a stateful connection between the client and the server
  - Used by the server to communicate with that client only
  - Any events on ws relates to the client only eg message event notifies that server that the client connected at the other end has just sent some data
- Each client will have a unique instance of **`ws`**
- 'Live' connect, cannot be saved to database, have to be held in memory

Example - Chat

A data structure to hold the live connection eg. object

```
const participants = {};

const app = express();
const appWS = require('express-ws')(app);

app.ws('/chat/:topic', (ws, req) => {
  const name = req.query.name;
  ...
  if (!name)
    return (ws.close(1000, 'Missing name'));

  if (name in participants)
    participants[name].close(1000);
  participants[name] = ws;
  ...
})
```

Use the name as the key for the connection. Easily identify the connection

Close the existing connection. Assume that we only allow one login per name

Sending and Receiving Data

- WebSocket can data as
  - Text or binary (focus text)
  - Packet or stream (focus packet)
- Sending data from server to client
  - If sending JSON, must stringify object before sending

```
ws.send(JSON.stringify(jsonData));
```

- Receiving data from client to server
  - If receiving JSON, must parse string

```
ws.on(‘message’, (payload) => {
  const data = JSON.parse(payload);
  ...
});
```