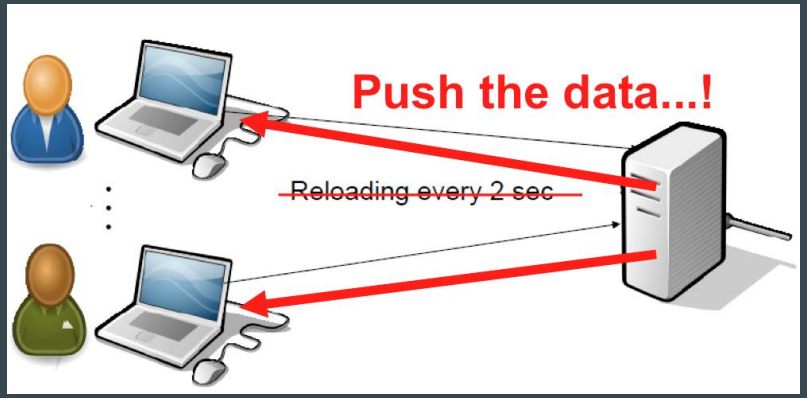
WebSockets && SockJS

Ahmedur Rahman Shovon
Codalo
shovon.sylhet@gmail.com

Different Communication Techniques in Web

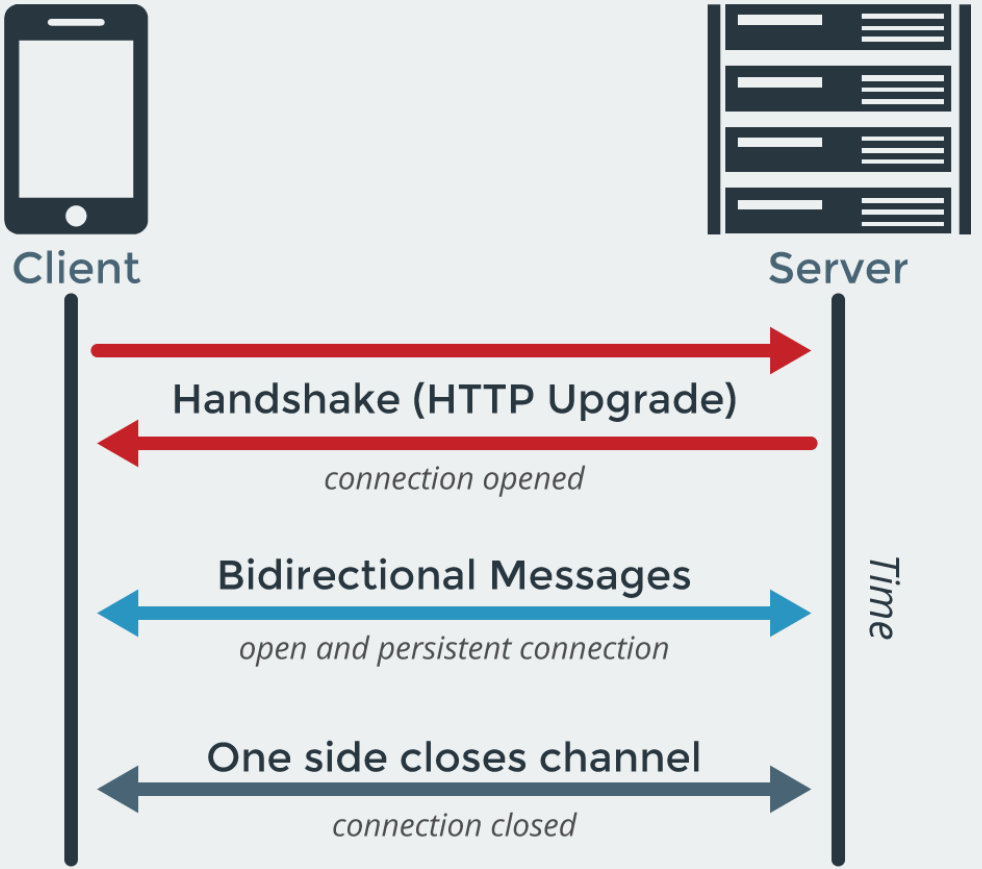
- **AJAX:** request → response. Creates connection to server, sends request headers with optional data, gets response from server, closes connection.
- **Long poll:** request → wait → response. Creates connection to server like AJAX does, but keep-alive connection open for some time (not long though).
- **WebSockets:** client ↔ server. Create TCP connection to server, and keep it as long as needed. Server or client can easily close it.



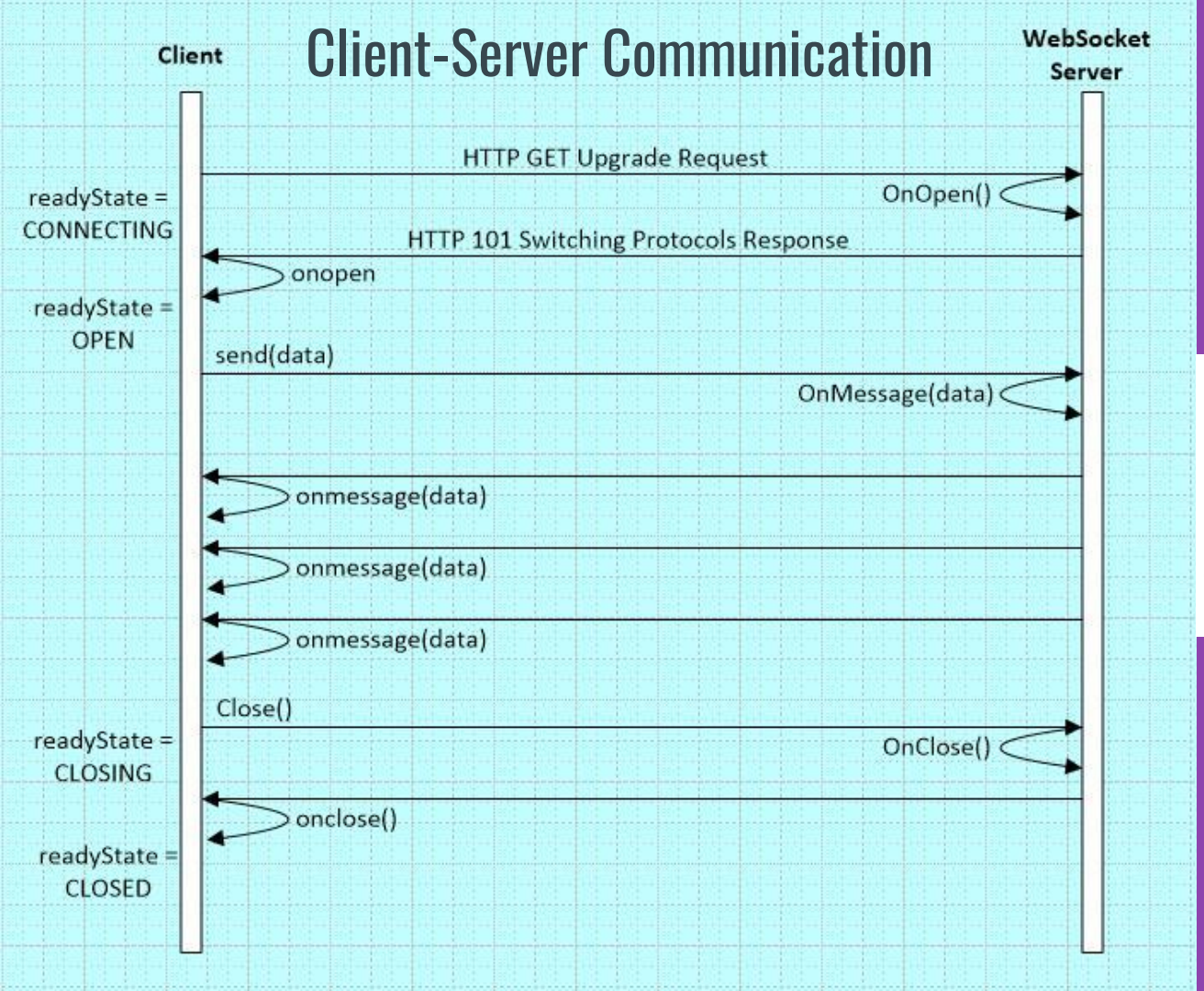
What is WebSockets?

“WebSockets is a technology for bidirectional communication over single (TCP) socket, a type of PUSH technology.”

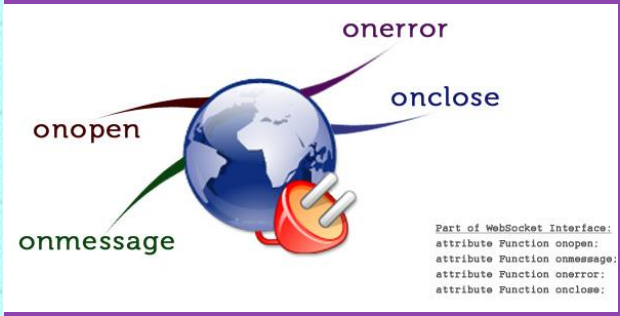
From code.tutsplus.com



Client-Server Communication



Events



Advantages

- Cross origin communication (however this poses security risks)
- Cross platform compatibility (web, desktop, mobile)
- No HTTP overhead
- Replace long-polling
- WebSockets are data typed but AJAX calls can only send String datatypes
- WebSockets are faster than AJAX

Disadvantages

- WebSockets has no success functions like AJAX
- WebSockets needs server support. Though any server will support WebSockets. Using hosting service, one may not be able to use them, specially Heroku.
- Load testing
- Scaling, redundancy, load balancing, replication

Use Cases of WebSockets

- Multiplayer online games
- Chat applications
- Live sports ticker
- Real time updating social streams

What is SockJS?

- SockJS is a browser JavaScript library that provides a WebSocket-like object.
- It gives a coherent, cross-browser, Javascript API which creates a low latency, full duplex, cross-domain communication channel between the browser and the web server.
- Under the hood SockJS tries to use native WebSockets first.
- SockJS is intended to work for all modern browsers and in environments which don't support the WebSocket protocol.

SockJS-client, SockJS-server

Remember these things please

SockJS-client JavaScript client library
SockJS-node Node.js server
SockJS-erlang Erlang server
SockJS-cyclone Python/Cyclone/Twisted server
SockJS-tornado Python/Tornado server
SockJS-twisted Python/Twisted server
Spring Framework Java client & server
vert.x Java/vert.x server
Xitrum Scala server
Atmosphere Framework JavaEE Server, Play Framework, Netty, Vert.x

What should I do to play with SockJS?

Required Setup to Wear Socks!

- SockJS client side library: sockjs-1.1.1.min.js
- Node.js (for running SockJS server, you can choose others too)
- SockJS libraries based on the SockJS server
- Node.js server library recommended by the SockJS: SockJS-node
- SockJS-node is a Node.js server side counterpart of SockJS-client browser library written in CoffeeScript.
- SockJS-node Installation: npm install sockjs

Now let’s see socketing in action!

Installing SockJS Server

At first we need to install SockJS-node as we are using Node.js as SockJS server.

- First install the Node.js installer.
- Then using Node Package Manager(NPM) install SockJS-node.

```
D:\nodePrac\chat_app\server>npm install sockjs
D:\nodePrac\chat_app\server
`-- sockjs@0.3.18
  +-- faye-websocket@0.10.0
  | `-- websocket-driver@0.6.5
  |   `-- websocket-extensions@0.1.1
  `-- uuid@2.0.3
```

Installing SockJS-node using npm

SockJS Server

Server Script

```
var http = require("http");
var sockjs = require("sockjs");

var user_list = {};

function broadcast(message){
    for(var user in user_list){
        user_list[user].write(JSON.stringify(message));
    }
}

var chat = sockjs.createServer();
chat.on("connection",function(conn){
    user_list[conn.id] = conn;
    conn.on("data", function(message){
        broadcast(JSON.parse(message));
    });
    conn.on("close",function(){
        delete user_list[conn.id];
    });
});

var node_server = http.createServer();
chat.installHandlers(node_server,{prefix:'/codalochat'});
node_server.listen(9999,'0.0.0.0');
```

Running the server

```
D:\nodePrac\chat_app\server>node codalo_chat_server.js
SockJS v0.3.18 bound to "/codalochat"
GET /codalochat/info 14ms 200
GET /codalochat/638/ccex1ess/websocket 15ms (unfinished)
GET /codalochat/info 1ms 200
GET /codalochat/387/wyd0xumb/websocket 2ms (unfinished)
```

SockJS Client Application

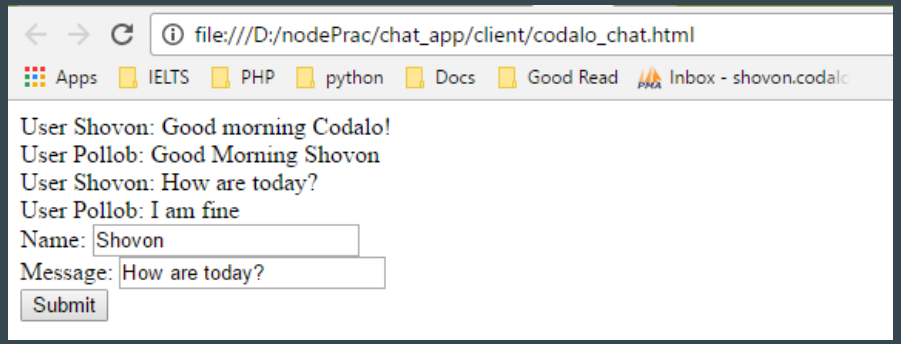
```html
<html>
    <head>
        <title>Codalo Chat</title>
    </head>
    <body>
        <div id="chat_content"></div>
        <br>

        <label for="username_text">Name:</label>
        <input type="text" id="username_text" placeholder="Enter Username" />
        <br>

        <label for="message_text">Message:</label>
        <input type="text" id="message_text" placeholder="Enter Message"  />
        <br>

        <input type="submit" id="submit_btn" value="Submit" />

        <script src="jquery-1.11.0.min.js"></script>
        <script src="sockjs-0.3.min.js"></script>
        <script src="chat_client.js"></script>
    </body>
</html>
```

Client Side Application

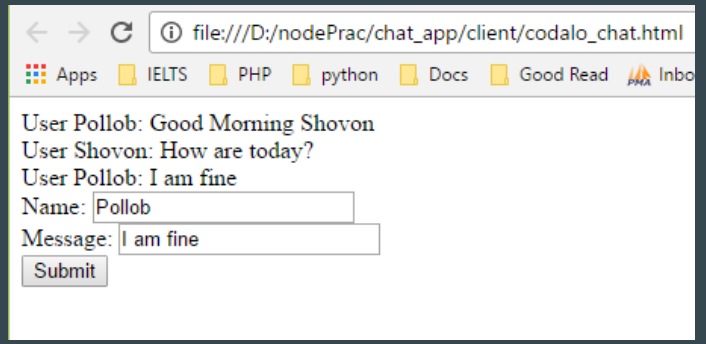
```javascript
$(document).ready(function(){
    var sock = new SockJS("http://localhost:9999/codalochat");

    sock.onopen = function(){};

    sock.onclose = function(){};

    sock.onmessage = function(e){
        var content = JSON.parse(e.data);
        $old_content = $("#chat_content").html();
        $new_line = "User "+content.username+": "+content.message+"<br>";
        $("#chat_content").html($old_content+$new_line);
    };

    $("#submit_btn").on("click",function(){
        var username = $("#username_text").val();
        var message = $("#message_text").val();
        var send_data = {username: username, message: message};

        sock.send(JSON.stringify(send_data));
    });
});
```

chat_client.js

SockJS Client Demo



User 1



User 2

And it's open source!

- SockJS-Client: https://github.com/sockjs/sockjs-client
- SockJS-Node: https://github.com/sockjs/sockjs-node
- Did not like the demo interface? Have a look at updated code here: https://github.com/arsho/real_time_chatting_wearing_socks

Note Behind

This interface does not allow for raw access to the underlying network. For example, this interface could not be used to implement an IRC client without proxying messages through a custom server.

Browser Quirks

- Pressing ESC in Firefox, before Firefox 20, closes the SockJS connection.
- *jsonp-polling* transport will show a "spinning wheel" (aka. "busy indicator") when sending data.
- Can't open more than one SockJS connection to one domain at the same time.
- From SockJS' point of view there is nothing special about SSL/HTTPS.
- Trying to connect from secure "https://" to insecure "http://" is not a good idea.

Thank you.