Socket.IO

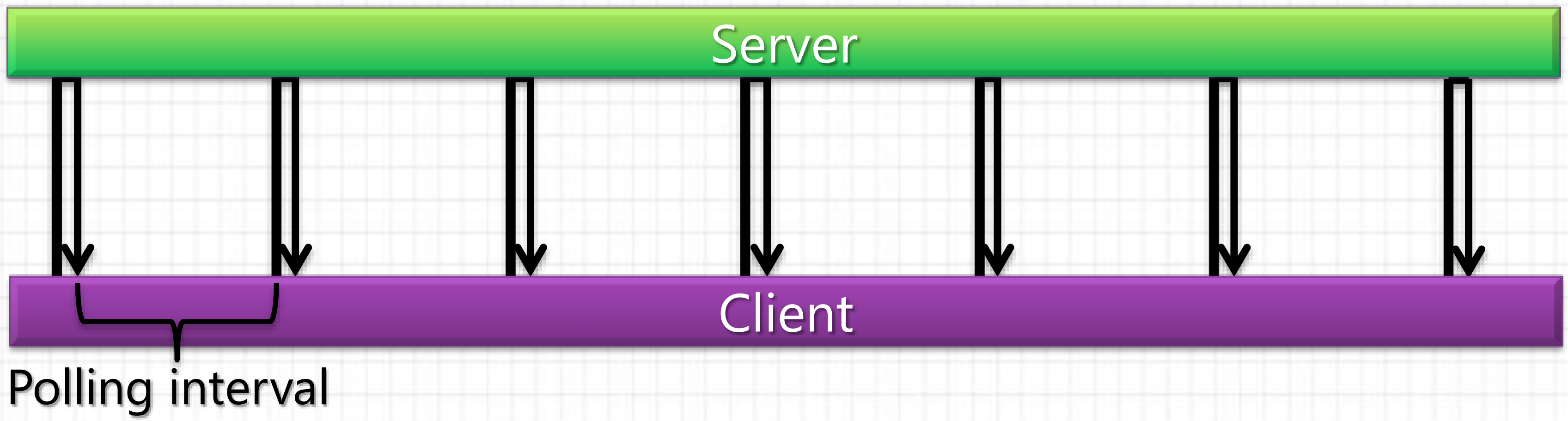
Eyal Vardi
Microsoft MVP ASP.NET
blog: eyalvardi.wordpress.com

Agenda

- Type of Communications
- WebSocket
- Socket.IO
- Angular Socket.IO

Type of Communications

Periodic polling

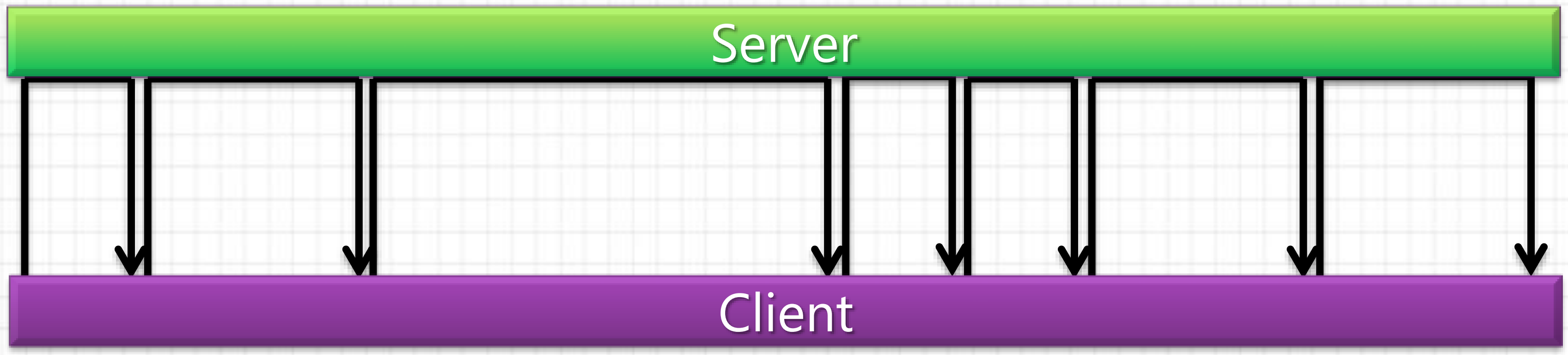


Poll from time to time using Ajax

Delay in communications due to polling interval

Wastes bandwidth & latency ☹

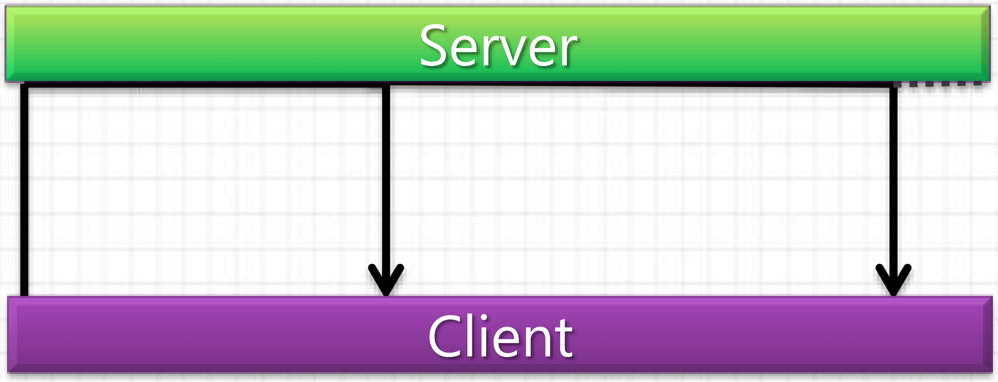
Long polling



Poll but don't respond untill there's data

Poll again after data received or after the connection times out. Consumes server threads & connection resources ☹

Forever Frame



```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked
```

```
<script>eval("... ")</script>\0
```

```
<script>eval("... ")</script>\0
```

Server tells client that response is chuncked Client keeps connection open untill server closes it Server pushes data to the client followed by \0 Consumes server threads.

WebSockets

What is WebSockets?

- WebSocket is a web technology providing for bi-directional, full-duplex communications channels over a single TCP connection.
- The communications are done over the regular TCP port number 80 or 443.
- **ws://** and **wss://** prefix to indicate a WebSocket and a WebSocket Secure connection, respectively.

WebSocket Protocol Handshake

- To establish a WebSocket connection, the client sends a WebSocket handshake request, and the server sends a WebSocket handshake response.

Client

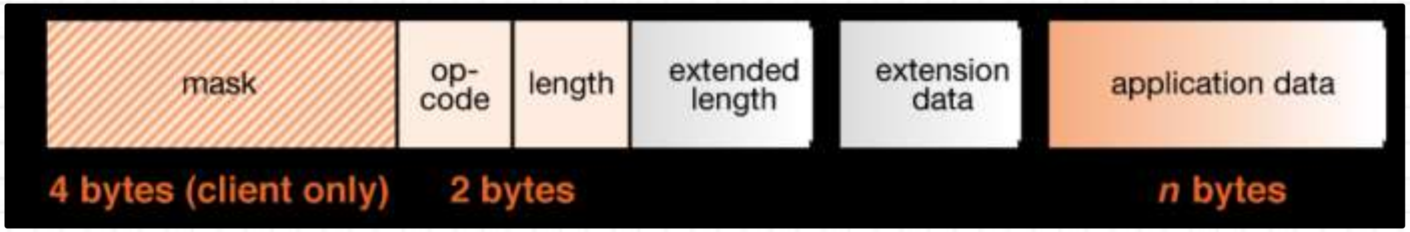
```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Server

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol
```

WebSocket Protocol Handshake

- WebSocket data frames can be sent back and forth between the client and the server in full-duplex mode.
  - Both text and binary frames can be sent in either direction at the same time.
  - The data is minimally framed with just two bytes. In the case of text frames, each frame starts with a 0x00 byte, ends with a 0xFF byte, and contains UTF-8 data in between.
  - WebSocket text frames use a terminator, while binary frames use a length prefix.



HTML5 WebSocket API

```
var myWebSocket       = new WebSocket("ws://www.websockets.org");

myWebSocket.onopen    = function(evt) { alert("Connection open ..."); };
myWebSocket.onmessage = function(evt) { alert("Received Message: " + evt.data); };
myWebSocket.onclose   = function(evt) { alert("Connection closed."); }

myWebSocket.send("Hello WebSockets!");
myWebSocket.close();
```

demo

Web Socket Echo

(http://www.websocket.org/echo.html)

Socket.IO

Socket.IO

- Socket.IO enables real-time bidirectional event-based communication.
- It works on every **platform**, **browser** or **device**, focusing equally on reliability and speed.
- Samples:
  - Real-time analytics
  - Binary streaming
  - Instant messaging and chat
  - Document collaboration

Echo Sample (Server)

```
var http = require("http");
var connect = require("connect");
var socketio = require("socket.io");
var app = connect();

app.use(connect.static("public"));
var server = http.createServer(app);

var io = socketio.listen(server);
io.on("connection", function (socket) {
    socket.on("message", function (data) {
        socket.emit("echo", data);
    });
});

server.listen(8000);
```

Echo Sample (client)

```
<!DOCTYPE html>
<html>
<head>
    <script src="/socket.io/socket.io.js"></script>
</head>
<body>
    <body>
        <script>
            var socket = io.connect("http://localhost");
            socket.emit("message", "Hello!");
            socket.on("echo", function(data) {
            document.write(data);
            });
        </script>
    </body>
</html>
```

Broadcasting

- In order to send an event to everyone, Socket.IO gives us the io.emit:
  - `io.emit('some event', { for: 'everyone' });`
  - `socket.broadcast.emit('some event');`

demo

Chat

Namespaces

- Socket.IO allows you to "namespace" your sockets, which essentially means assigning different *endpoints* or *paths*.
- We call the default namespace **/** and it's the one Socket.IO clients connect to by default, and the one the server listens to by default.

```
// Server Side
var nsp = io.of('/my-namespace');
nsp.on('connection', function(socket) {
    console.log('someone connected');
});
nsp.emit('hi', 'everyone!');
```

```
// Client Side
var socket = io('/my-namespace');
```

Rooms

- Within each namespace, you can also define arbitrary channels that sockets can join and leave.

```
io.on('connection', function (socket) {
    socket.join('some room');
});

io.to('some room').emit('some event');
```

Integration With AngularJS

- Socket.IO integration with any JavaScript application should be simple enough.

- In Angular, there are a few subtleties to address, such as the **digest cycle**.
  - Angular-socket-io by **Brian Ford** is a tiny and simple bower component that takes care of those issues

socket.io + ANGULARJS

Using angular-socket-io

- Using Your Socket Instance.

```
var mi = angular.module('myApp', ['btford.socket-io']);

mi.factory('mySocket', function (socketFactory) {
    return socketFactory();
});

mi.controller('MyCtrl', function(mySocket) {
    // ...
});
```

Using angular-socket-io

```
angular.module(‘socketioApp')
.controller('MainCtrl', function($scope, socket) {

    socket.emit(‘event-from-client’, someData)

    socket.forward(‘my-event', $scope);

    $scope.$on(‘socket:my-event',
        function (event, serverData) {
        $scope.data = serverData;
      });
}
```

Allows you to forward the events received by Socket.IO's socket to AngularJS's event system.

Thanks

eyalvardi.wordpress.com

Eyal Vardi
Microsoft MVP ASP.NET
blog: eyalvardi.wordpress.com