SPIRIA

THINK.
CODE.
COLLABORATE.

spiria.com

Websockets Bring Light at the End of the Tunnel

Presented by

JOEL LORD

Node PDX – June 20th, 2016

Aboot me, eh?

JOEL **LORD**

SPIRIA

- Javascript junkie
- Tinkerer
- Technology enthusiast

@joel__lord #nodepdx

WEB SOCKETS

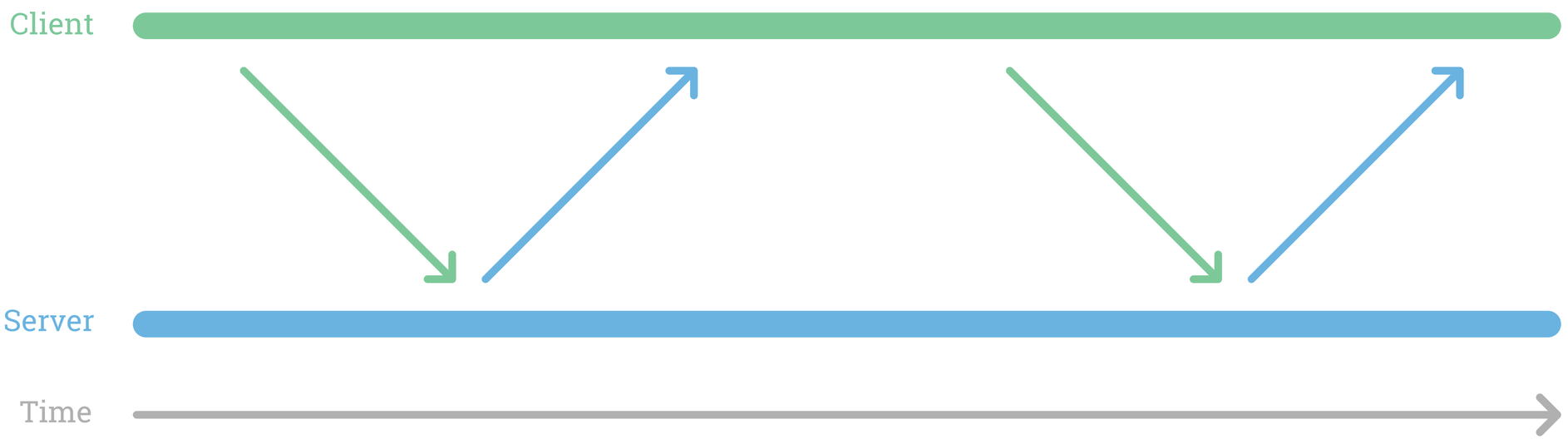
What are they?

WebSocket is a protocol providing full-duplex communications channels over a single TCP connection.

The **WebSocket** protocol was standardized by the IETF as RFC 6455 in 2011, and the **WebSocket**API in Web IDL is being standardized by the W3C.

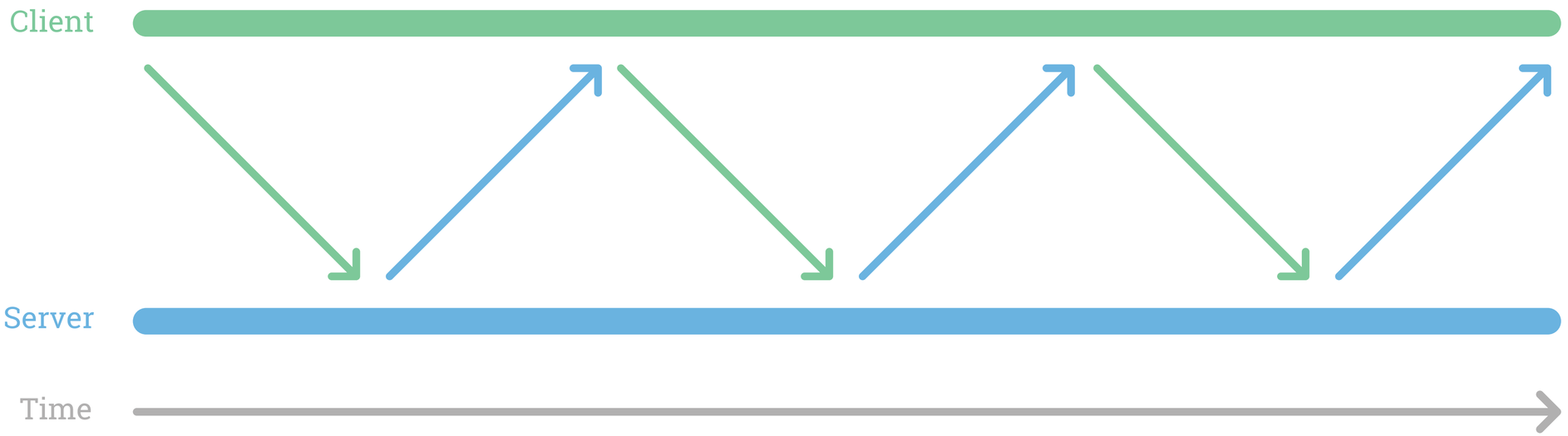
Typical real-time

NORMAL HTTP REQUESTS



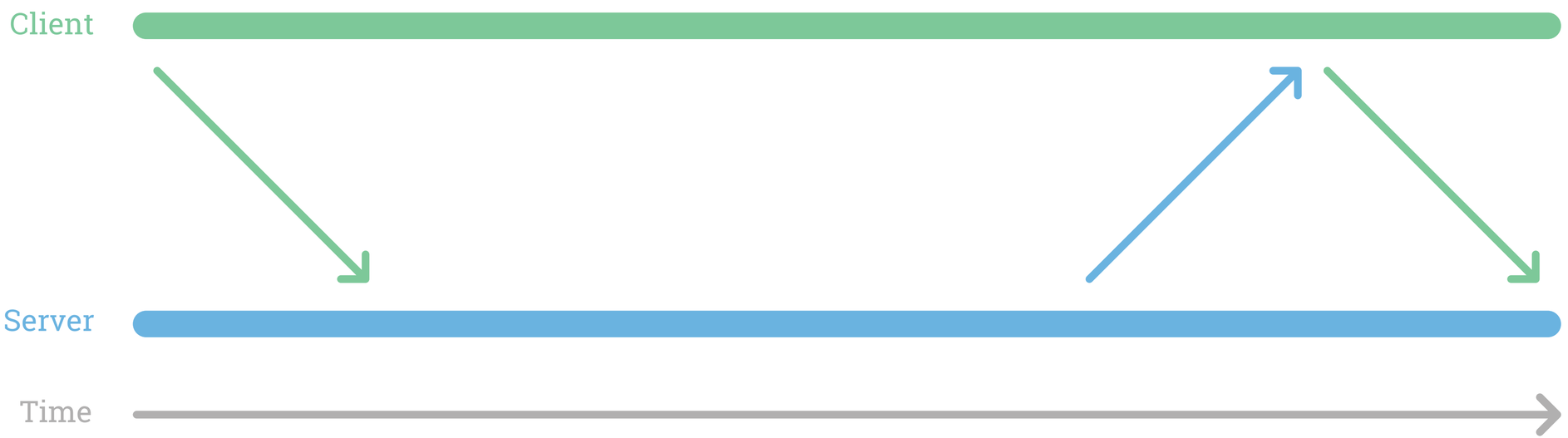
Typical real-time

POLLING



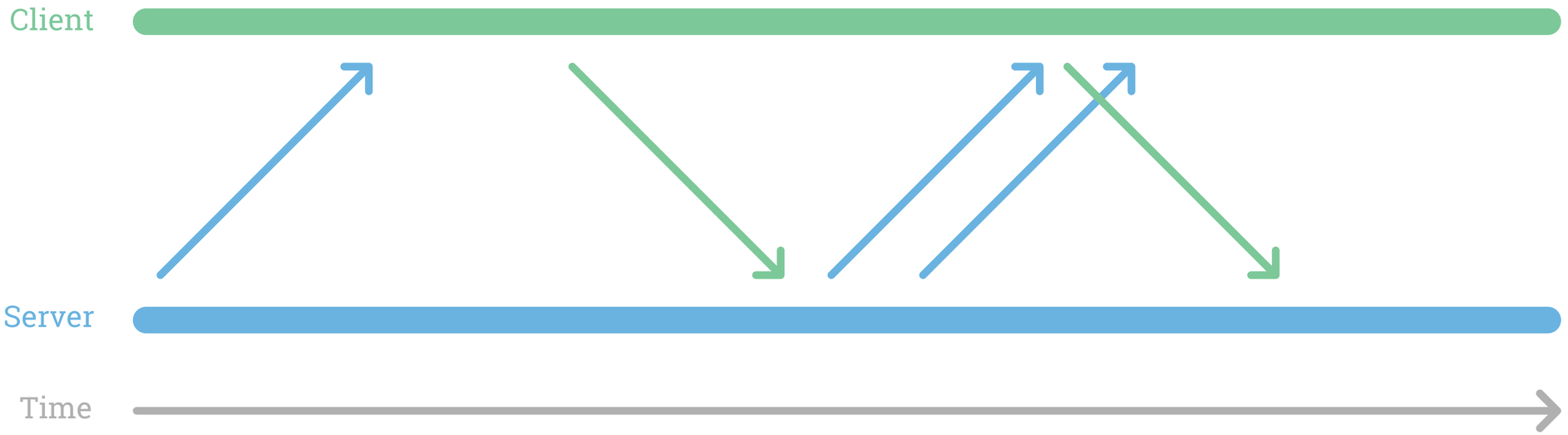
Typical real-time

LONG-POLLING



Typical real-time

WEBSOCKETS



Typical real-time

WHAT ABOUT SERVER-SIDE EVENTS?

Pros and cons.

01

PROS

Bi-directional data transfers

Fast

Low bandwith

Detects connection and disconnection

02

CONS

Lose the caching capabilities built in XHR and HTTP

Architectural changes in the application

WebSockets in the Wild

REAL WORLD CASES

- Real time synchronization of data amongst a group of users (trello.com)
- Live feeds (Twitter stream)
- Long server processes and providing ETAs to users
- Multiplayer HTML5 games
- Chat clients (Slack)

Using WebSockets with HTML5

ESTABLISHING A CONNECTION

```
var connection = new WebSocket('ws://somedomain/path/')
```

Using WebSockets with HTML5

EVENTS

```
connection.onopen = function(e) {
    console.log("Connected");
};

connection.onmessage = function(e) {
    console.log( "Received: " + e.data);
};

connection.onclose = function(e) {
    console.log("Connection closed");
};
```

Using WebSockets with HTML5

SENDING DATA TO THE SERVER

```
//Strings
connection.send('this is a string');

//Array Buffers
var image = aCanvas.getImageData(0, 0, 640, 480);
var binary = new Uint8Array(image.data.length);
for (var i = 0; i < image.data.length; i++) {
    binary[i] = image.data[i];
}
connection.send(binary.buffer);

//Blobs
var myFile = document.querySelector('input[type="file"]').files[0];
connection.send(myFile);

//JSON objects ?
var jsonObject = JSON.stringify({"data": "value"});
connection.sent(jsonObject);
```

Getting started

VARIOUS IMPLEMENTATIONS

- PubNub (cloud)
- Ratchet (php)
- Jetty (Java)
- socket.io (node.js)

Here comes Socket.io!

WHAT'S SO COOL ABOUT IT?

- Server and client-side implementation
- Falls back to long polling when necessary (IE 9 😱)
- Adds features like heartbeats, timeouts, and disconnection support not provided in WebSocket API
- Easy stuff !

Here comes Socket.io!

CLIENT SIDE

```
<body>

<input type="text" id="textField" />
<button type="button" id="sendMsg">Send</button>
<textarea id="msgBox"></textarea>

</body>
<script src="/socket.io/socket.io.js"></script>
<script type="text/javascript">
    var socket = io();

    document.getElementById("sendMsg").onclick = function() {
        socket.emit("messageFromClient", {
            text: document.getElementById("textField").value
        });
    };

    socket.on("messageFromServer", function(data) {
        document.getElementById("msgBox").value += data.text;
    });

</script>
```

Here comes Socket.io!

SERVER SIDE

```
//Express server setup
var express = require("express");
var app = express();
var server = require("http").createServer(app);
var port = 8888;

server.listen(port, function () {
    console.log("Server started on port " + port);
});

app.use(express.static(__dirname + "/../"));

//Socket setup

var io = require("socket.io")(server);
io.on("connection", function (socket) {
    socket.on("messageFromClient", function (data) {
        socket.broadcast("messageFromServer", data);
    });
});
```

LET'S GET SERIOUS

Coding time!

SPIRIA

THINK.
CODE.
COLLABORATE.

spiria.com

The end!

Presented by

JOEL LORD

@joel__lord #nodepdx

/joellord

Node PDX – June 20th, 2016