Node.js and WebSockets

Gonzalo Ayuso 2011

@gonzalo123
http://gonzalo123.wordpress.com
https://github.com/gonzalo123

Part 1. Node.js

Part 2. WebSockets

Node.js | What's node.js

- JavaScript on the server
- Written by Ryan Dahl
- Based on V8 (Google)

- Asynchronous event-driven model
- Similar in design to:
  - Event Machine (Ruby)
  - Twisted (Python)

Node.js | Why?

I/O needs to be done differently

Node.js | Why?

I/O needs to be done differently

From:

```
recordset = db.query("select * from Table");
render(recordset);
```

Node.js | Why?

I/O needs to be done differently

Q: When will you add threads to JavaScript?”
A: over your dead body

Brendan Eich (creator of JavaScript)

Node.js | Why?

I/O needs to be done differently

From:

```
recordset = db.query("select * from Table");
render(recordset);
```

To:

```
db.query("select * from Table", function (recordset) {
    render(recordset)
});
```

Node.js | Why?

I/O needs to be done differently

From:

```
recordset = db.query("select * from Table")
render(recordset)
```

To:

```
db.query("select * from Table", function (recordset) {
    render(recordset)
});
```

Design Goals

- No function should direct perform I/O.
- To read info from disk, network, ... there must be a **callback**

Node.js | Show me the code!

HTTP SERVER

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, "127.0.0.1");

console.log('Server running at http://127.0.0.1:1337/');
```

Node.js | Show me the code!

```
HTTP SERVER
var http = require('http');
var total = 0;

http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/plain'
  });
  res.end('Hi ' + total + '\n');
  tot++;
}).listen(1337, "127.0.0.1");

console.log('Server running at http://127.0.0.1:1337/');
```

Node.js | Pros and Cons

PROS

- Great I/O performance
- Just JavaScript. We all know JavaScript
- Community
- A lot of modules available https://github.com/joyent/node/wiki/modules
- npm

Node.js | Pros and Cons

PROS

- Great I/O performance
- Just JavaScript. We all know JavaScript
- Community
- A lot of modules available https://github.com/joyent/node/wiki/modules
- npm

CONS

- Hosting
- We don't really know JavaScript
- Modules too young
- One thread, one single process for all
- Windows support

Part 1. Node.js

Part 2. WebSockets

WebSockets | History

Description of the problem:

- **Real time communications in Browser**

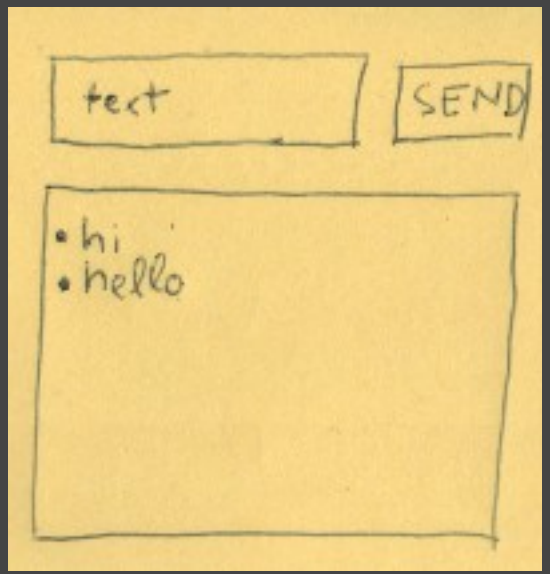
WebSockets | History

Description of the problem:

- **Real time communications in Browser**

Imagine. Let's create simple chat client (5 years ago)**:**

Trivial problem with heavy clients, hard in browsers.



WebSockets | History

COMET (Wikipedia):

Comet is a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it.

WebSockets | History

COMET (Wikipedia):
Comet is a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it.

Problem with COMET:

- Servers (keep-alive, MaxClients, ...)
- Clients

WebSockets | History

COMET (Wikipedia):
Comet is a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it.

Problem with COMET:

- Servers (keep-alive, MaxClients, ...)
- Clients

WebSockets | Short Polling / Long Polling

Short Polling

- <meta http-equiv="refresh" content="5">
- setInterval and setTimeout
- Inneficient
- Uses many resources (DB Connection, ...)

WebSockets | Short Polling / Long Polling

Short Polling

- <meta http-equiv="refresh" content="5">
- setInterval and setTimeout
- Inneficient
- Uses many resources (DB Connection, ...)

Long Pooling

- Looks like Real Time but it isn't RT

WebSockets | Short Polling / Long Polling

Short Polling

- <meta http-equiv="refresh" content="5">
- setInterval and setTimeout
- Inneficient
- Uses many resources (DB Connection, ...)

Long Pooling

- Looks like Real Time but it isn't RT

It works? **Yes**
It scales? **No** (Long Polling better but still No)
Is your sysadmin happy? **No**

WebSockets | Short Polling / Long Polling

Short Polling

- <meta http-equiv="refresh" content="5">
- setInterval and setTimeout
- Inneficient
- Uses many resources (DB Connection, ...)

Long Pooling

- Looks like Real Time but it isn't RT

It works? **Yes**
It scales? **No** (Long Polling better but still No)
Is your sysadmin happy? **No**

WebSockets | HTML5

WebSockets

This specification defines an API that enables Web pages to use the WebSocket protocol for two-way communication with a remote host.

WebSockets | HTML5

WebSockets

This specification defines an API that enables Web pages to use the WebSocket protocol for two-way communication with a remote host.

That's means:

- The solution of the problem with RT at browser side

WebSockets | HTML5


```
var ws = new WebSocket(url);
ws.onopen = function() {
    // When the connection opens
 };
ws.onmessage = function() {
    // When the server sends data
 };
ws.onclose = function() {
    // When the connection is closed
 };
ws.send('Hi all');
// later...
ws.close();
```

WebSockets | HTML5

.

Cool but ...

WebSockets | HTML5

.

Cool but ...

Not all browsers support it

WebSockets | socket.io

Is there a solution?

WebSockets | socket.io

Is there a solution?

http://socket.io/

WebSockets | socket.io

Is there a solution?

http://socket.io/

Client

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
  });
</script>
```

WebSockets | socket.io

Is there a solution?

http://socket.io/

Client

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
  });
</script>
```

Server (node.js application)

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data) {
    console.log(data);
  });
});
```

WebSockets | socket.io

.

http://socket.io/

Supported transports

- WebSocket
- Adobe® Flash® Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

WebSockets | socket.io

.

http://socket.io/

Supported transports

- WebSocket
- Adobe® Flash® Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

Supported browsers

- Internet Explorer 5.5+
- Safari 3+
- Google Chrome 4+
- Firefox 3+
- Opera 10.61+
- iPhone Safari
- iPad Safari
- Android WebKit
- WebOs WebKit

References

- http://dev.w3.org/html5/websockets/
- https://github.com/joyent/node/wiki/modules
- http://nodejs.org/
- http://socket.io/
- http://en.wikipedia.org/wiki/Push_technology
- http://www.youtube.com/watch?v=jo_B4LTHi3I
- http://gonzalo123.wordpress.com/category/technology/node-js/

The End.

Many thanks

@gonzalo123
http://gonzalo123.wordpress.com