WebSocket & Node.js

Chunki Jun
Developer @ ImageClick

Agenda

- Introduction to WebSocket / Node.js
- How to use WebSocket
- Sample project - “Parody Wave”
- Demo

WebSocket?

- Part of HTML5
- Two-way communication
  - XMLHttpRequest
  - <iframe>
  - long polling
- Work with proxy/firewall
- Less overhead

- Network Protocol
- Javascript API

Browser Support

- Chrome 4
- Firefox 4 beta
- Opera 11 beta
- Safari 5
- IE 9?
- iOS 4.2

The WebSocket Protocol

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 24, 2010

I. Hickson
Google, Inc.
May 23, 2010

The WebSocket protocol
draft-ietf-hybi-thewebsocketprotocol-00

Abstract

The WebSocket protocol enables two-way communication between a user agent running untrusted code running in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the Origin-based security model commonly used by Web browsers. The protocol consists of an initial handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g. using XMLHttpRequest or <iframe>s and long polling).

NOTE! THIS COPY OF THIS DOCUMENT IS OBSOLETE.

For an up-to-date copy of this specification, please see:

http://www.whatwg.org/specs/web-socket-protocol/

http://tools.ietf.org/id/draft-ietf-hybi-thewebsocketprotocol-00.txt

Protocol Overview

```
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1  .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4 @1  46546xW%0l 1 5
Origin: http://example.com

^n:ds[4U
```

```
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Location: ws://example.com/demo
Sec-WebSocket-Protocol: sample

8jKS'y:G*Co,Wxa-
```

0x00 ...DATA... 0xFF

W3C Editor's Draft

The WebSocket API

Editor's Draft 1 November 2010

Latest Published Version:
http://www.w3.org/TR/websockets/
Latest Editor's Draft:
http://dev.w3.org/html5/websockets/
Previous Versions:
http://www.w3.org/TR/2009/WD-websockets-20090423/
http://www.w3.org/TR/2009/WD-websockets-20091029/
Editors:
Ian Hickson, Google, Inc.

API Overview


```
Part of WebSocket Interface:
attribute Function onopen:
attribute Function onmessage:
attribute Function onerror:
attribute Function onclose:
```

- Introduction to WebSocket / Node.js
- How to use WebSocket
- Sample project - “Parody Wave”
- Demo

Node.js?

- Server-side Javascript Framework
- … to provide an **easy** way to build **scalable network programs**.
- Evented, Non-blocking
- CommonJS
- V8
- C/C++ & JavaScript
- Support Linux/Mac/Solaris/Windows/BSD

Node.js?

- Download
- ChangeLog
- Build
- About
- Links
- Contributing
- v0.3.1 docs
- v0.2.5 docs

Evented I/O for V8 JavaScript.

An example of a web server written in Node which responds with "Hello World" for every request.

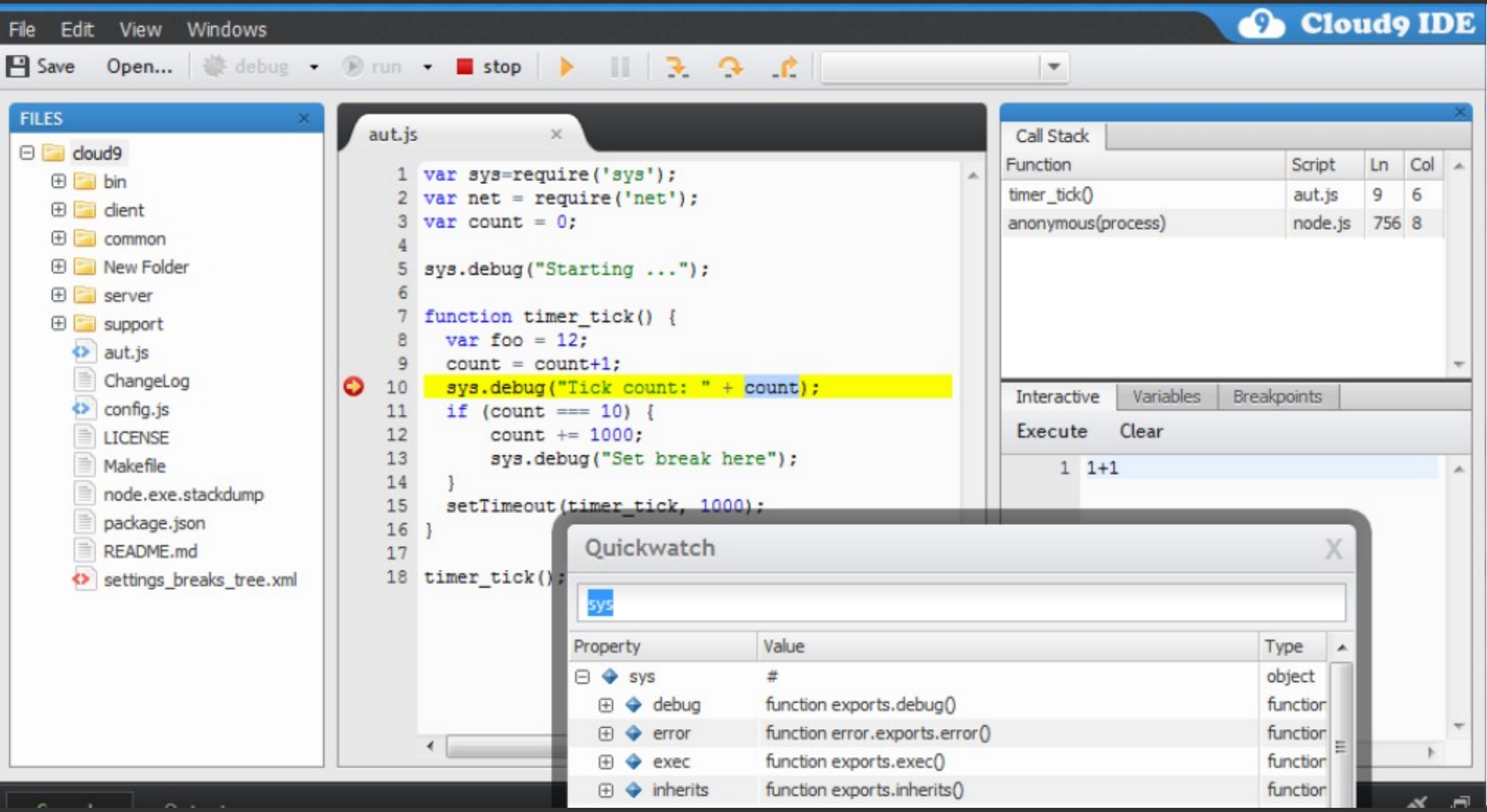
```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
```

http://nodejs.org/

Node.js is easy!

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}).listen(8124, "127.0.0.1");
console.log('Server running at http://127.0.0.1:8124/');
```

Apps Using Node.js



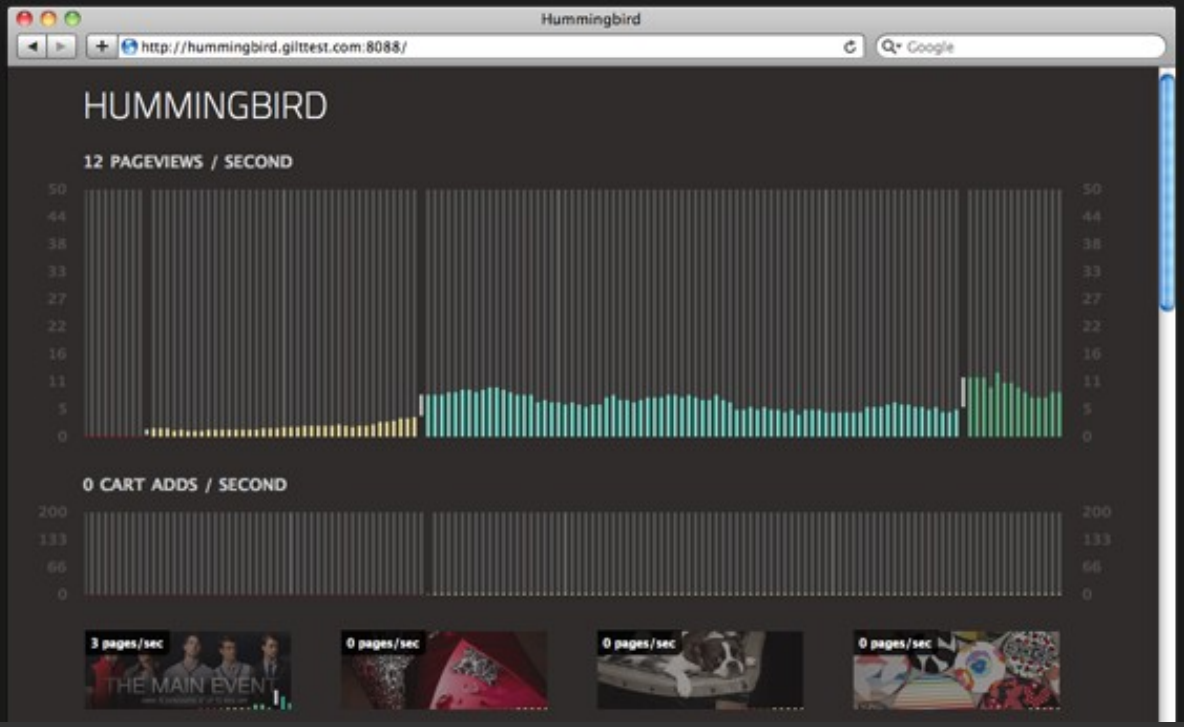
http://www.cloud9ide.com/

Apps Using Node.js

HUMMINGBIRD

Real Time Web Traffic Visualization Preview

Note: Hummingbird is pre-alpha software. While it shouldn't screw up your production environment, it's definitely not yet production ready. In fact, we're still in the process of extracting it from the original app. So check it out, but don't expect it to fulfill your greatest desires yet.



Hummingbird lets you see how visitors are interacting with your website in real time.

And by "real time" we don't mean it refreshes every 5 minutes—WebSockets enable Hummingbird to update 20 times per second.

Hummingbird is built on top of Node.js, a new javascript web toolkit that can handle large amounts of traffic and many concurrent users.

Go to the GitHub page | View the Live Demo

http://hummingbirdstats.com/

- Introduction to WebSocket / Node.js
- How to use WebSocket
- Sample project - “Parody Wave”
- Demo

WebSocket

Node.js

```
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1  .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4 @1  46546xW%0l 1 5
Origin: http://example.com

^n:ds[4U
```

'Upgrade' Event!

Implement – Client side

```
var socket = new WebSocket('ws://example.com:1204');

socket.onopen = function (event) {...};
socket.onmessage = function (event) {...};
socket.onerror = function (event) {...};
socket.onclose = function (event) {...};

socket.send('Hello WebSocket!');

socket.close();
```

Implement - Server side

```
var http = require('http');
var server = http.createServer();

server.addListener('upgrade', function (request, socket, head) {
    if (request.headers['upgrade'] == 'WebSocket') {
        var key = calcResponseKey(request.headers['sec-websocket-key1'],
                                  request.headers['sec-websocket-key2'], head);

        var response = ['HTTP/1.1 101 Web Socket Protocol Handshake',
                        'Upgrade: WebSocket',
                        'Connection: Upgrade',
                        'Sec-WebSocket-Origin: ' + request.headers['origin'],
                        'Sec-WebSocket-Location: ws://' + request.headers['host']  + request.url,
                        '',
                        ''].join('\r\n');

        socket.write(response);
        socket.write(key, "binary");

        // ...
    }
});

function calcResponseKey(key1, key2, key) { ... }

server.listen(1204);
```

Implement - Server side

```
var http = require('http');
var server = http.createServer();

server.addListener('upgrade', onUpgrade);

function onUpgrade(request, socket, head) { ... }

server.listen(1204);
```

- Introduction to WebSocket / Node.js
- How to use WebSocket
- Sample project - “Parody Wave”
- Demo

Parody Wave?

- Message Frame

  - Text
  - XML
  - JSON

Client-side Message

open	user, title
join	user, id
status	
list	
new	value
update	value
submit	value

Server-side Message

status	connections
list	threads
join	thread, [title, data]
update	id, line, value, user
error	text

- Opening WebSocket Connection
- Sending Messages to Server
- Processing Messages from Server
- Closing Connection

- Opening WebSocket Connection

```
function login() {
    if("WebSocket" in window) {
        ParodyWaveClient.conn = new WebSocket("ws://member.webxapp.com:5555");
        ParodyWaveClient.conn.onopen = function () {
            // ...
            setInterval(function() {
              ParodyWaveClient.conn.send('HB');
            }, 60000);
        }
        ParodyWaveClient.conn.onmessage = onWebSocketMessage;
        ParodyWaveClient.conn.onclose = onWebSocketClose;
    }
    else {
        alert("This web browser doesn't support WebSocket!");
    }
}
```

- Sending Messages to Server

```
function sendMsg(msg) {
    if (ParodyWaveClient.conn) ParodyWaveClient.conn.send(JSON.stringify(msg));
}
```

- Processing Messages from Server

```
function onWebSocketMessage(e) {
    var msg;
    try {
        msg = JSON.parse(e.data);
    }
    catch (e) {
        msg = {type: 'error'};
    }
    switch (msg.type) {
        case 'status':
            // ...
            break;
        case 'join':
            // ...
            break;
        case 'list':
            // ...
            break;
        case 'update':
            // ...
            break;
        case 'error':
            break;
    }
}
```

- Closing Connection

```
function onWebSocketClose() {
    // initiate text input
    // ...

    // close others
    // ...

    // initiate status
    // ...

    // unbind event
    // ...
}
```

- WebSocketServer.js
- ParodyWave.js

- “Upgrade” Event Listener
- Handshake
- Binding socket for a service

- "Upgrade" Event Listener

```
this.on('upgrade', onUpgrade);
```

WebSocketServer.js

- Handshake

```
this.WebSocketResponse76 = [
    'HTTP/1.1 101 Web Socket Protocol Handshake',
    'Upgrade: WebSocket',
    // ...
]

function onUpgrade(req, socket, head) {
    if ("sec-websocket-key1", "sec-websocket-key2" in req.headers) { // 76
        var key = calcResponseKey(req.headers["sec-websocket-key1"],
                                  req.headers["sec-websocket-key2"], head);
        // ...
        var res = socket.server.WebSocketResponse76.join('\r\n')
                    .replace(/\{origin\}/,req.headers.origin || '')
                    .replace(/\{protocol\}/,'ws')
                    .replace(/\{host\}/,req.headers.host || '')
                    .replace(/\{resource\}/,req.url || '')
                    .replace(/\{data\}/,key);
    } else {    // 75
        // ...
    }

    try {
        socket.write(res, "binary");
        // ...
    } catch (e) {
        socket.destroy();
    }
}
```

- Binding socket for a service

```
function onUpgrade(req, socket, head) {
    // ...
    try {
        socket.write(res, "binary");
        var client = initiateWebSocket(socket);
        var action = require('./ParodyWave');
        action.service(client);
        client.ready(); // "open" event for action
    } catch (e) {
        socket.destroy();
    }
}
```

- Keeping Status
- Processing Messages from Clients
- Sending Messages to Clients

- unload problem

```
window.onunload = function () {
    if (ParodyWaveClient.conn) ParodyWaveClient.conn.close();
}
```

- Timeout problem

```
setInterval(function() {
  ParodyWaveClient.conn.send('HB');
}, 60000);
```

- Let's do it!

Thanks!

Q & A