Left Logic

09:00	Begin!
10:00	Break (15m)
12:00	Lunch (45m)
14:30	Break (15m)
17:00	Fin.

Remy / @rem

node.js

For a real-time web

- Hi, I'm Remy
- @rem
  remy@leftlogic.com
- I <3 JavaScript
- Questions: interrupt & ask!

Schedule

- Basics
- Debugging
- Modules
- Web Servers
- WebSockets
- Live tips

Me & Node

- Discovered at jsconf 2009
- Lightbulb moment :)
- JavaScript
- JAVASCRIPT!
- Built few tools
- Even some services!

YOU?

How's Node Different?

Different?

- Event driven
- High concurrency
- Non-blocking
- JavaScript

Cons?

- Young
- API could still change
- Documentation, tutorials, etc

Installing

- Manually: make
- Or nave – for multi-node
- Or other methods (n, etc)

Running

- localhost
- nodester.com
- nodejitsu
- jsapp.us
- <more>

Hello REPL

```
$ node
> foo = 12;
12
> foo += 8;
20
>
```

More REPL

```
> name = 'Remy';
'Remy'
> name
'Remy'
> console.log(name.split('').reverse().join(''))
ymeR
undefined
```

More REPL

```
> name = 'Remy';
'Remy'
> name
'Remy'
> console.log(name.split('').reverse().join(''))
ymeR
undefined
```

Getting stuck

```
> var foo = function () {
...
```

Getting stuck

```
> var foo = function () {
... .break
>
```

Hello Terminal

```
$ node -e "12+8" -p
20
```

Env values

```
$ FOO=bar node
> process.env;
{ FOO: ‘bar’,
  // etc...
```

Dev vs. Prod

```
$ NODE_ENV=prod node app.js

if (process.env.NODE_ENV=='prod'){
  // fork
}
```

Native modules

```
var fs = require('fs'),
    path = require('path'),
    // etc...
```

http://nodejs.org/docs/latest/api/

VS. PHP

```
echo "Hello";
sleep 2; // Blocks :(
echo "World";
```

Node doesn't block.

Async

```
setTimeout(function () {
  console.log('World');
}, 2000);

console.log('Hello');
```

Async

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

console.log('step 1: file exists?');

path.exists(process.argv[2], function (exists) {
  console.log('Exists? ' + exists);
});

console.log('step 2: now we wait...');
```

Let's build Web Servers

http://www.flickr.com/photos/nnova/4588857363/

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

req is the inbound request

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

res is the *outbound* response, which we will write to

Task

- Return “Writing JavaScript makes me happy”
- Change content type to HTML
- Return different content response to different url requests

nodemon

No more stop 'n start

```
$ npm install -g nodemon
```

Debugging

http://www.flickr.com/photos/knowprose/101872870

debug

```
$ node debug script.js
debug>
< debugger listening on port 5858
connecting... ok
break in callbacks.js:1
  1 var path = require('path');
  2
  3 console.log('step 1: file exists?');
```

debug

```
debug> setBreakpoint('callbacks.js', 3);
  1 var path = require('path');
  2
* 3 console.log('step 1: file exists?');
  4
  5 path.exists(process.argv[2], function
  6     console.log('Exists? ' + exists);
```

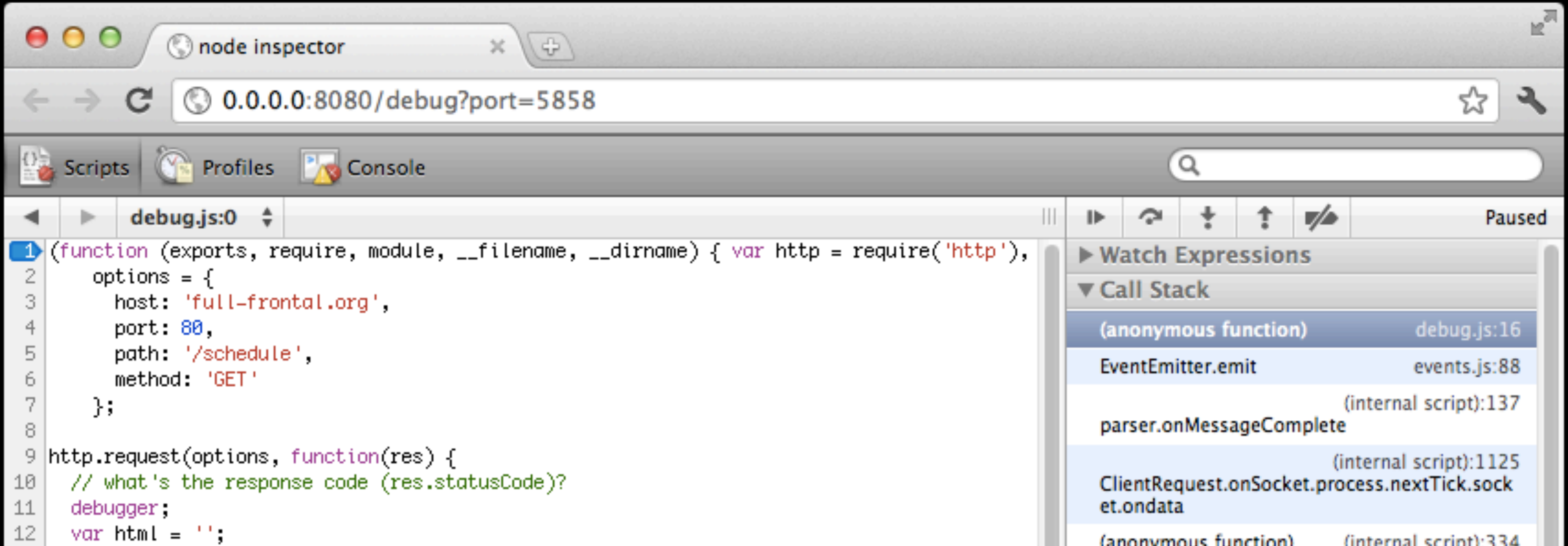
debug

```
debug> watch('exists')
debug> cont
break in callbacks.js:3
Watchers:
  0: exists = "<error>"

  1 var path = require('path');
  2
* 3 console.log('step 1: file exists?');
  4
  5 path.exists(process.argv[2], function
```

debugger

```
npm install -g node-inspector
```



```
(function (exports, require, module, __filename, __dirname) { var http = require('http'),
    options = {
      host: 'full-frontal.org',
      port: 80,
      path: '/schedule',
      method: 'GET'
    };

http.request(options, function(res) {
  // what's the response code (res.statusCode)?
  debugger;
  var html = '';
```

```
(anonymous function)                          debug.js:16
EventEmitter.emit                             events.js:88
parser.onMessageComplete          (internal script):137
ClientRequest.onSocket.process.nextTick.socket.ondata   (internal script):1125
(anonymous function)              (internal script):334
```

debugger

```
$ npm install -g node-inspector
$ node-inspector &
$ node --debug-brk script.js
```

`--debug-brk` breaks on first line. `--debug` might run before debugger is hooked.

Task

- Use http.request to get the body of http://2011.full-frontal.org/schedule
- Use debugger to check statusCode
- Find position of “node.js”
- Find what string is at 6175 characters in, 5 characters long?

Modules

```
module.exports = ...
```

```
var talk = {
  say: function (line) {
    console.log(line);
  },
  shout: function (line) {
    console.log(line.toUpperCase());
  }
};
```

```
var talk = {
  say: function (line) {
    console.log(line);
  },
  shout: function (line) {
    console.log(line.toUpperCase());
  }
};

module.exports = talk;
```

```
var talk = require('./talk');

talk.say("Hi there little fella");
```

```
var talk = require('./talk');

talk.say("Hi there little fella");
```

No .js as it's a module

```
var talk = require('./talk');

talk.say("Hi there little fella");
```

Relative to running script

Tasks

- Create a monkey module
- Add:
  - monkey.say
  - monkey.do
- Include module and call methods

- Require from ./module
- Reading from a dir – using index.js
- Reading from a dir – using package.json (main: 'module')
- Reading from ‘node_modules’ – also where npm will install
- Publishing your own npm module

NPM FTW

http://search.npmjs.org

```
$ npm install <module>
```

More than package manager

- Deploy your packages
- Automatic dependancy management
- Global & local installs

package.json

+

npm install

=

#win

Dependencies

.git/hooks/post-merge

```
#!/bin/sh
echo 'updating modules'
npm install
```

Manage expectations

```
"dependencies": {
  "connect": ">= 1.8.0",
  "underscore": ">= 1.3.0"
}, // ...
```

Bad :(

Manage expectations

```
"dependencies": {
  "connect": "1.8.x",
  "underscore": "1.x.x"
}, // ...
```

Good :)

connect

@1.8.5

Connect 2.0 is out, but isn't compatible with Express...yet.

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = http.createServer(function (req, res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = connect().createServer(function(req,res) {
  res.end('Hello World');
});

server.listen(8000);
```

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

var server = connect();

server.use(connect.static(__dirname + '/public'));

server.listen(8000);
```

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

var server = connect();

server.use(connect.static(__dirname + '/public'));

server.listen(8000);
```

serves static content on
"this directory"/public

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

var server = connect();

server.use(connect.static(__dirname + '/public'));

server.listen(8000);
```

.static returns a function
– which handles req & res

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

var server = connect();

server.use(connect.static(__dirname + '/public'));

server.listen(8000);
```

This is middleware

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

var server = connect();

server.use(connect.static(__dirname + '/public'));

server.listen(8000);
```

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

var server = connect();

server.use(connect.logger());
server.use(connect.static(__dirname + '/public'));

server.listen(8000);
```

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

var server = connect()
    .use(connect.logger())
    .use(connect.static(__dirname + '/public'))
    .listen(8000);
```

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

var server = connect()
    .use(connect.logger())
    .use(connect.static(__dirname + '/public'))
    .listen(8000);
```

code time

Tasks

- Serve up static content
- Add directory listing support
- Support favicon
- Start server on PORT environment value

Routers

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

var server = connect();

server.use(connect.logger());
server.use(connect.static(__dirname + '/public'));
server.use(connect.router(routes));

server.listen(8000);
server.use(connect.static(__dirname + '/public'));
server.use(connect.router(routes));

server.listen(8000);
```

```
var routes = function (app) {
  app.get('/adam', function (req, res) {
    // serve custom content
  });
};
```

(note: routes would be above - it's just below for slide display...sorry!)

```
var routes = function (app) {
  app.get('/adam', function (req, res) {
    // serve custom content
  });
};
```

hardcoded route

```
var routes = function (app) {
  app.get(/a.*/i, function (req, res) {
    // serve custom content
  });
};
```

Regular expressions

```
var routes = function (app) {
  app.get('/:name', function (req, res) {
    // serve custom content
  });
};
```

Placeholder: req.params.name

Query Data

POST

```
connect.bodyParser()
req.body.key
```

Query Data

POST

```
connect.bodyParser()
req.body.key
```

GET

```
connect.query()
req.query
```

Middleware

```
server.use(myMiddleWare());
server.use(connect.other());
server.use(connect.logger());
// etc
```

Middleware

```
connect.createServer(
  function (req, res, next) {
    // 1. add something to req
    req.foo = ‘bar’;
    // 2. continue to next middleware
    next();
    // 3. or throw exception
    next(new Error(‘go away’));
    // 4. or close request
    res.writeHead(404);
    res.end(‘fail :(‘);
  },
  connect.logger(),
  // etc
);
```

```
server.use(
  function (req, res, next) {
    // 1. add something to req
    req.foo = ‘bar’;
    // 2. continue to next middleware
    next();
    // 3. or throw exception
    next(new Error(‘go away’));
    // 4. or close request
    res.writeHead(404);
    res.end(‘fail :(‘);
  }
);
```

```
server.use(
  function (req, res, next) {
    // 1. add something to req
    req.foo = ‘bar’;
    // 2. continue to next middleware
    next();
    // 3. or throw exception
    next(new Error(‘go away’));
    // 4. or close request
    res.writeHead(404);
    res.end(‘fail :(‘);
  }
);
```

```
server.use(
  function (req, res, next) {
    // 1. add something to req
    req.foo = ‘bar’;
    // 2. continue to next middleware
    next();
    // 3. or throw exception
    next(new Error(‘go away’));
    // 4. or close request
    res.writeHead(404);
    res.end(‘fail :(‘);
  }
);
```

```
server.use(
  function (req, res, next) {
    // 1. add something to req
    req.foo = ‘bar’;
    // 2. continue to next middleware
    next();
    // 3. or throw exception
    next(new Error(‘go away’));
    // 4. or close request
    res.writeHead(404);
    res.end(‘fail :(‘);
  }
);
```

```
server.use(
  function (req, res, next) {
    // 1. add something to req
    req.foo = ‘bar’;
    // 2. continue to next middleware
    next();
    // 3. or throw exception
    next(new Error(‘go away’));
    // 4. or close request
    res.writeHead(404);
    res.end(‘fail :(‘);
  }
);
```

```
server.use(
  function (req, res, next) {
    // 1. add something to req
    req.foo = ‘bar’;
    // 2. continue to next middleware
    next();
    // 3. or throw exception
    next(new Error(‘go away’));
    // 4. or close request
    res.writeHead(404);
    res.end(‘fail :(‘);
  }
);
```

Tasks

- Write a middleware to count page requests
- Add CORS support to /image/ requests

Express

```
$ npm install -g express
$ express mysite
$ cd mysite
$ npm install -d
$ nodemon app.js
$ npm install -g express
$ express mysite
$ cd mysite
```

But requires some Jade knowledge

Tasks

- Create partial for Family Guy pages: name & desc minimum
- Read JSON file and support http://mysite/peter
- Add API support for peter.json and peter.json?callback=foo

command line

http://www.mongodb.org/downloads

Command line

```
$ mongod # start server

$ mongo
MongoDB shell version: 1.8.2
connecting to: test
> use family-guy
switched to db family-guy
> db.characters.find();
> use family-guy
switched to db family-guy
> db.characters.find();
```

The database name

```
> use family-guy
switched to db family-guy
> db.characters.find();
```

The collection

```
> use family-guy
switched to db family-guy
> db.characters.find();
```

Find with no criteria

```
> use family-guy
switched to db family-guy
> db.characters.find();
```

```
.find({ name: ‘foo’ })
```

```
> use family-guy
switched to db family-guy
> db.characters.find();
```

```
.findOne({ name: ‘foo’ })
```

```
> use family-guy
switched to db family-guy
> db.characters.find();
```

```
.find({ name: ‘foo’ }).length()
```

```
> use family-guy
switched to db family-guy
> db.characters.find();
```

```
.remove({ name: ‘foo’ })
```

Mongoose

Better than nested callback hell

Task

```
$ mongoimport -d family-guy -c characters
```

```
var mongoose = require('mongoose'),
    path = require('path'),
    Schema = mongoose.Schema;

mongoose.connect('mongodb://localhost/family-guy');

var Character = new Schema({
  name  :  { type: String },
  url   :  { type: String, index: true },
  image :  { type: String },
  bio   :  { type: String, trim: true }
});
```

Web Sockets

http://www.flickr.com/photos/44442915@N00/4960579336

The old =
comet =
hacks =
overhead

web sockets are the silver bullet.

Draft 75

Draft 76

Draft 10

Draft 17

http://www.flickr.com/photos/redux/5086447340/

Don’t worry, smarter developers than us have our back

In a nutshell

- Persistent connection
- Tiny chunks of data exchanged
- Bi-directional & no origin rules

Some uses

- Chat aka Hello World
- Multi-gaming state
- Google Wave remember?!

Some uses

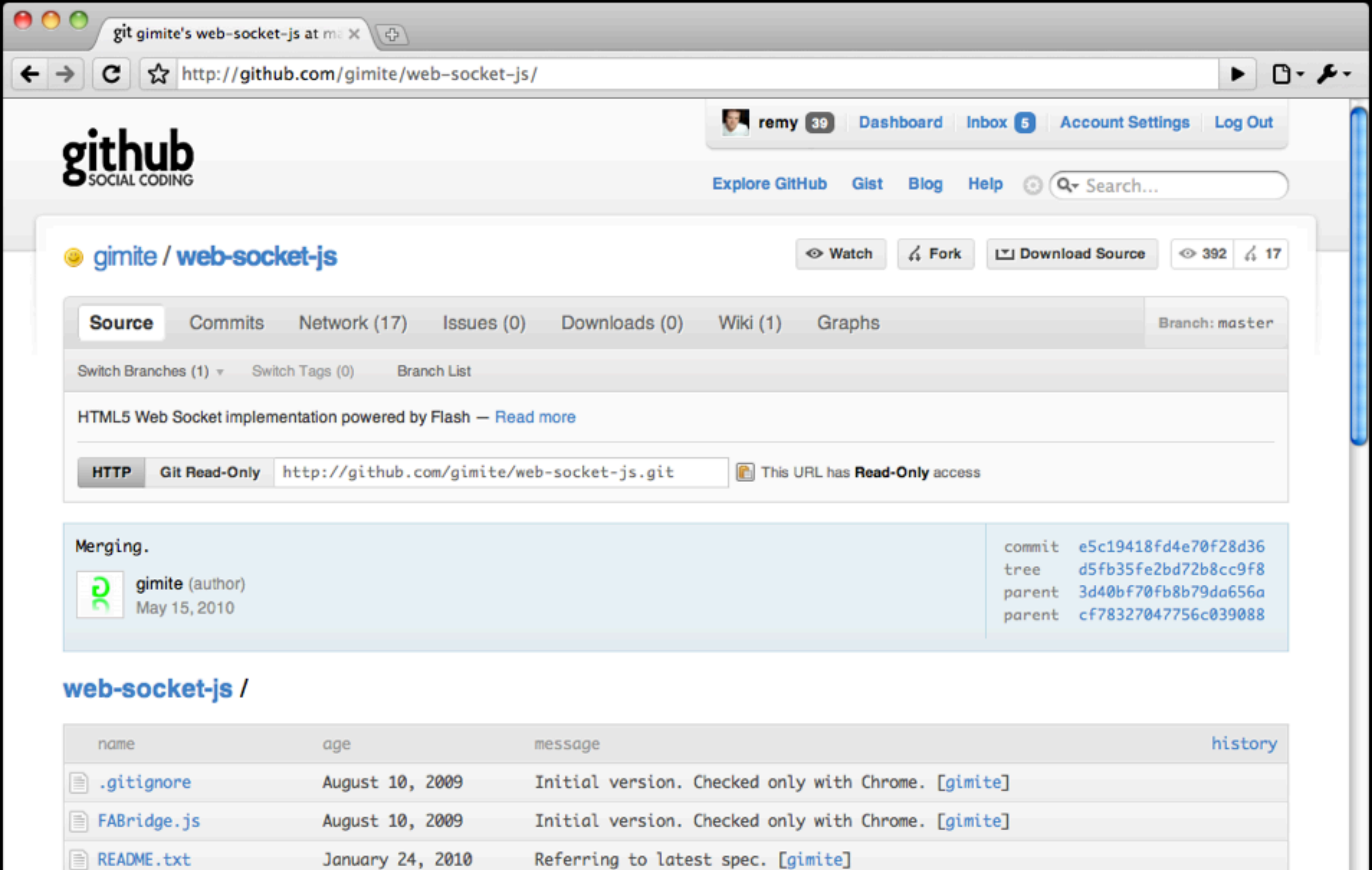
Chat aka Hello World

- Multi-gaming state
- Google Wave remember?!

Native or polyfilled

`IE6✓  :'(`

http://github.com/gimite/web-socket-js/



gimite / web-socket-js

HTML5 Web Socket implementation powered by Flash — Read more

http://github.com/gimite/web-socket-js.git — This URL has **Read-Only** access

Merging.

gimite (author)
May 15, 2010

```
commit  e5c19418fd4e70f28d36
tree    d5fb35fe2bd72b8cc9f8
parent  3d40bf70fb8b79da656a
parent  cf78327047756c039088
```

web-socket-js /

name	age	message	history
.gitignore	August 10, 2009	Initial version. Checked only with Chrome. [gimite]	
FABridge.js	August 10, 2009	Initial version. Checked only with Chrome. [gimite]	
README.txt	January 24, 2010	Referring to latest spec. [gimite]	

```
new WebSocket(url)
```

```
ws://node.remysharp.com:8000
```

```
onopen

onmessage

onclose

onerror
```

```
var data = JSON.parse(event.data);
```

Product support

- DIY - full control good thing?
- Socket.IO - provides full browser support
- Pusher - provides full redundancy support

Task

- Connect to `ws://node.remysharp.com:8000`
- Listen for messages
- Post your a personalised welcome message from you

Server side

Let’s not write this ourselves!

websocket.io

github.com/LearnBoost/websocket.io

cliché simple chat

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

```
var connect = require('connect'),
    ws = require('websocket.io');

var app = connect.createServer(
  connect.static(__dirname)
).listen(process.env.PORT || 8000);

ws.attach(app).on('connection',function(sock){
  sock.on('message', function (msg) {
    // new message in from socket
  }).on('close', function () {
    // now clean up
  });
});
```

Task

- Create simple echo server AKA chat: maintain active connections, create a broadcast function.
- Support message types: leave, join, say, etc
- Add a robot that runs special commands

Keep it up

1. **forever start example.js**
2. **forever list**
3. **forever stop example.js**
4. **forever stop 0** to stop the process with index 0 (as shown by forever list).

forever!

How I do things

1. `proxy requests`
2. `screen -S <name>`
3. `forever <script>`