

# RESTful Services in Node.js

Nick Nisi

# Nick Nisi

-  @nicknisi
-  <https://github.com/nicknisi>
-  <http://nicknisi.com>
- 



# What is Node.js?



# History

- Invented in 2009 by Ryan Dahl
- Currently maintained by Joyent

# What is node.js?

- ▶ JavaScript on the Server
- ▶ Asynchronous I/O
- ▶ Event Driven
- ▶ Built on top of Google's V8



Node.js is a platform built on **Chrome's JavaScript runtime** for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

# Node.js Pros/Cons

## Pros

- Write code in a single language for backend/frontend
- JavaScript is fun!
- Can share frontend/backend code
- No context switch between frontend/backend code

## Cons

- Non-blocking I/O is different and can be hard
- Callback hell - hard to maintain code
- missing/immatute libraries



Adam Lassek  
@alassek



Following

I enjoy writing Javascript, but I can't for the life of me fathom why someone would want to run it on the server.

Reply Retweet Favorited



why node.js



Web Images Maps Shopping More Search tools

About 4,910,000 results (0.13 seconds)

[npm](https://npmjs.org/)

<https://npmjs.org/>

**Node.js** Home · Download · About · npm Registry · Docs · Blog · Community · Logos · Jobs · Create Account | Login · npm. Node Packaged Modules ...

You visited this page on 1/1/13.

[Node.js](https://nodejs.org/)

[nodejs.org/](https://nodejs.org/)

Event-driven I/O server-side **JavaScript** environment based on V8. Includes API documentation, change-log, examples and announcements.

You've visited this page 2 times. Last visit: 8/14/12

[Made of Bugs » Why node.js is cool \(it's not about performance\)](http://blog.nelhage.com/2012/03/why-node-js-is-cool/)

[blog.nelhage.com/2012/03/why-node-js-is-cool/](http://blog.nelhage.com/2012/03/why-node-js-is-cool/)

Mar 12, 2012 – **node.js** is cool because it solves a problem shared by virtually every mainstream language. That problem is the fact that, as long as "ordinary" ...

[Why Node.JS is absolutely terrible](http://dev.hasenj.org/post/31042963934/nodejs-hate)

[dev.hasenj.org/post/31042963934/nodejs-hate](http://dev.hasenj.org/post/31042963934/nodejs-hate)



by Hasen el Judy - in 84 Google+ circles - More by Hasen el Judy  
Sep 8, 2012 – **Why Node.JS** is absolutely terrible. **Node.JS** is a terrible platform. It's terribleness stems from a very simple aspect of it, and this aspect happens ...

# Installing Node.js



# Installing Node.js

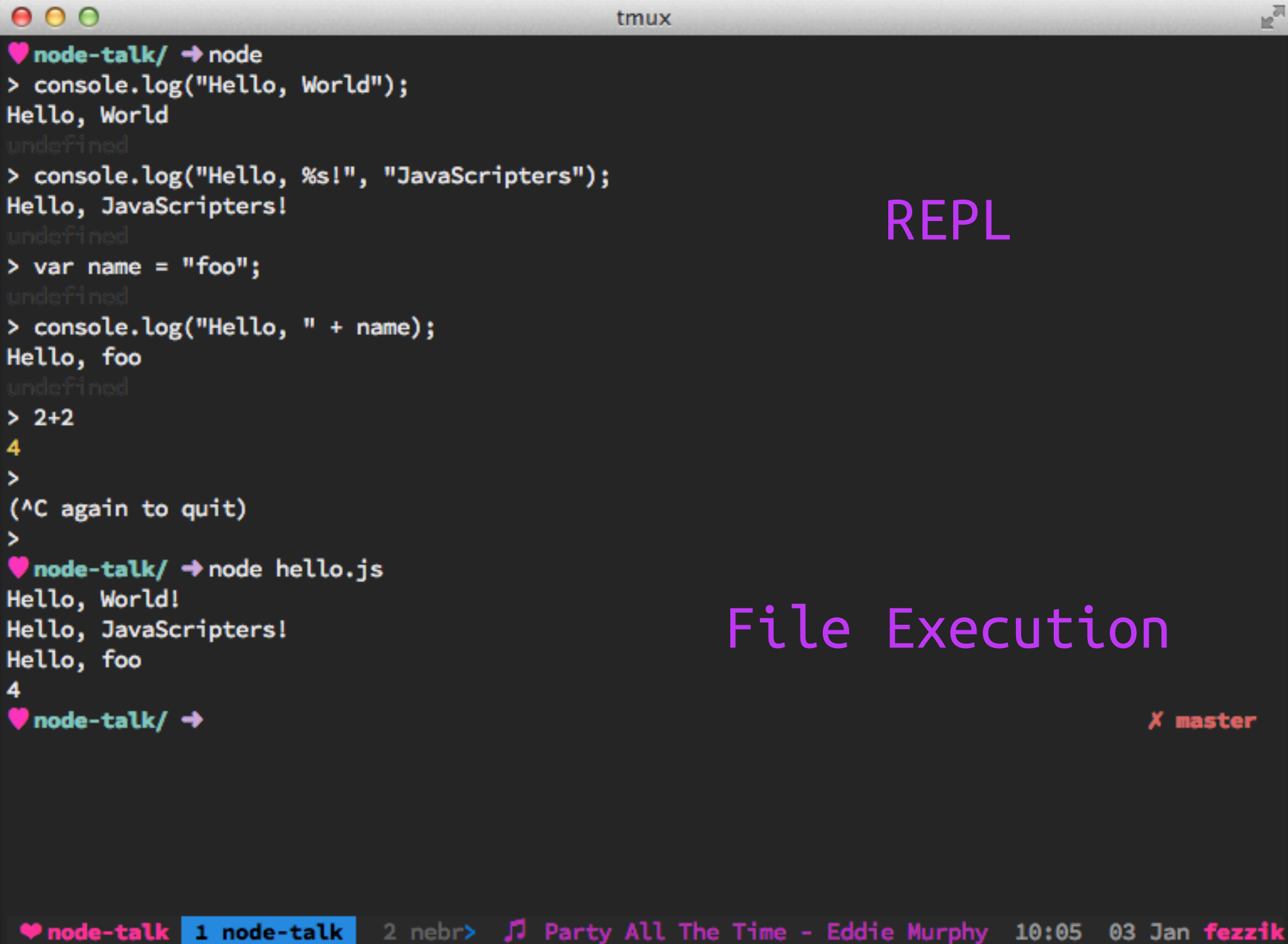
- Install binary
- Build
- homebrew (OS X)
  - `brew install node`



# Getting Started With Node.js



# Hello World



The image shows a tmux terminal window with a dark background. The title bar at the top says 'tmux'. The terminal content is as follows:

```
♥ node-talk/ → node
> console.log("Hello, World");
Hello, World
undefined
> console.log("Hello, %s!", "JavaScripters");
Hello, JavaScripters!
undefined
> var name = "foo";
undefined
> console.log("Hello, " + name);
Hello, foo
undefined
> 2+2
4
>
(^C again to quit)
>
♥ node-talk/ → node hello.js
Hello, World!
Hello, JavaScripters!
Hello, foo
4
♥ node-talk/ →
```

On the right side of the terminal, the word 'REPL' is written in large purple letters. Below it, the words 'File Execution' are written in large purple letters. In the bottom right corner of the terminal, there is a red 'X' followed by the word 'master'.

The tmux status bar at the bottom shows: ♥ node-talk | 1 node-talk | 2 nebr> | 🎵 Party All The Time - Eddie Murphy | 10:05 | 03 Jan fezzik

# Accessing Other Files

```
var User = function (firstName, lastName) {  
  this.firstName = firstName || '';  
  this.lastName = lastName || '';  
};  
  
User.prototype.getName = function () {  
  return this.firstName + ' ' +  
  this.lastName;  
};  
  
module.exports = User;
```

user.js



```
var User = require("./user");  
var user = new User("Big", "Deal");  
console.log("Hello, ", user.getName());
```

hello.js



# NPM

- Comes with Node.js
- Install packages / utilities / command line tools

`npm install PACKAGE` → `./node_modules`

`npm install -g PACKAGE` → `/usr/local`



express 3.0.0



# package.json

```
{  
  "name": "sample-app",  
  "version": "0.0.1",  
  "private": true,  
  "scripts": {  
    "start": "node app"  
  },  
  "dependencies": {  
    "express": "3.0.5",  
    "less-middleware": "*",  
    "hbs": "~2.0.1"  
  }  
}
```

npm install

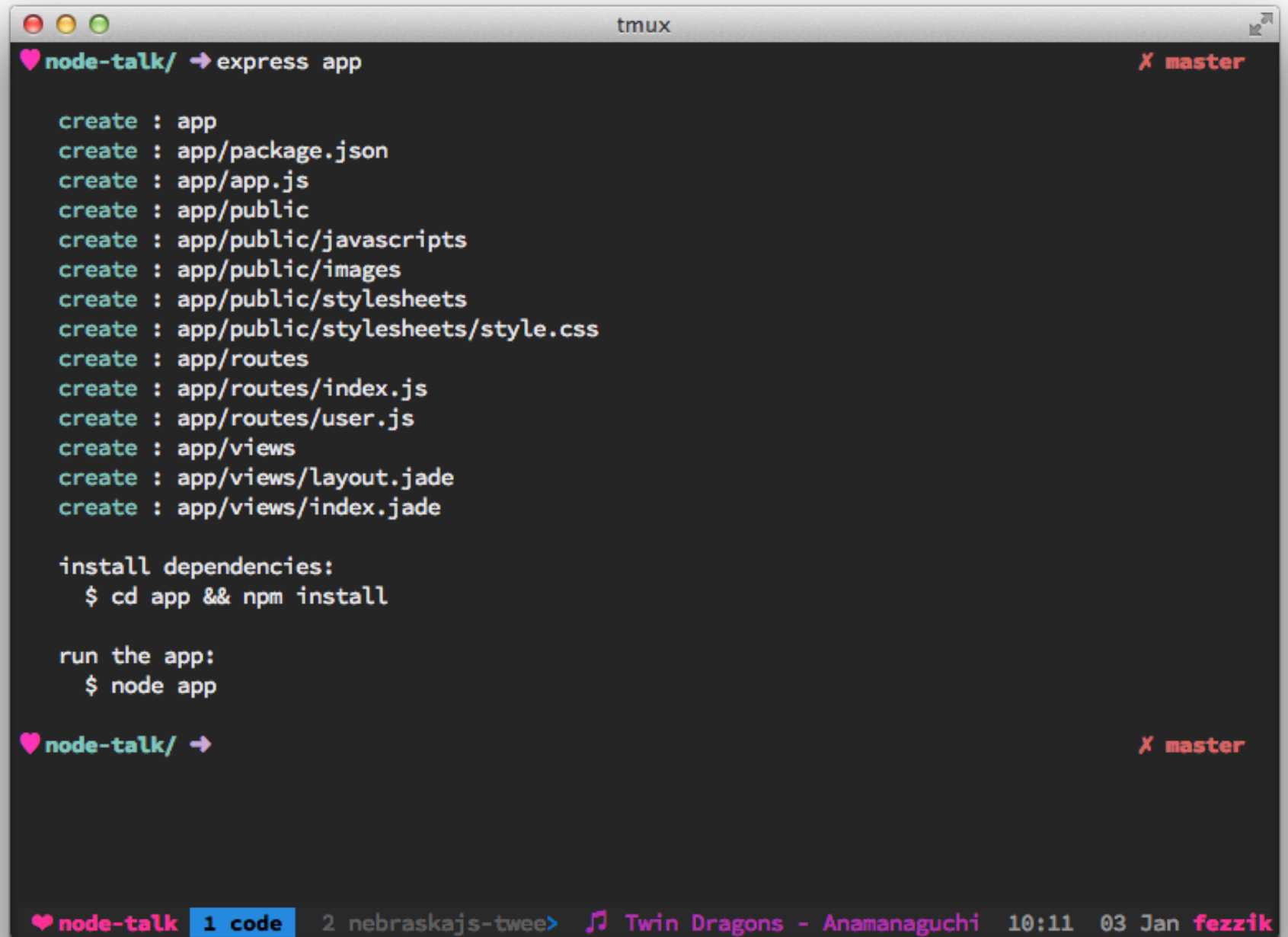
# Express



\* all code examples available at <https://github.com/nicknisi/node-talk>

# Creating A New Express App

```
npm install -g express  
express app
```



```
node-talk/ → express app master  
  
create : app  
create : app/package.json  
create : app/app.js  
create : app/public  
create : app/public/javascripts  
create : app/public/images  
create : app/public/stylesheets  
create : app/public/stylesheets/style.css  
create : app/routes  
create : app/routes/index.js  
create : app/routes/user.js  
create : app/views  
create : app/views/layout.jade  
create : app/views/index.jade  
  
install dependencies:  
$ cd app && npm install  
  
run the app:  
$ node app  
  
node-talk/ → master  
  
node-talk 1 code 2 nebraskajs-twee> Twin Dragons - Anamanaguchi 10:11 03 Jan fezzik
```

# Simple Express Server

```
var express = require('express');  
  
var app = express();  
  
app.get('/', function (req, res) {  
    res.send("Hello, World");  
});  
  
app.listen(3000);
```



# Simple Middleware

```
var express = require("express"),
    fs = require('fs');

var app = express();

app.get('/', function (req, res) {
  res.send("Hello, World");
});

var middleware = function (req, res, next) {
  // do something important
  // or don't.
  // i'm a comment, not a cop
  fs.readFile(__dirname + '/name.txt', function (err, data) {
    req.name = data;
    next();
  });
};

app.get('/foo', middleware, function (req, res) {
  res.send("Hello, " + req.name + " from the route!");
});

app.use(function (req, res) {
  res.writeHead(200, { "Content-Type": "text/plain" });
  res.end('Hello, Middleware');
});

app.listen(3000);
```

# Variable URLs

```
var express = require("express");

var app = express();

app.get('/', function (req, res) {
  res.send("Hello, World");
});

app.get('/greeting/:name/:location', function (req, res) {
  res.send("Hello, " + req.params.name + " from " + req.params.location);
});

app.listen(3000);
```

# Using Templates

```
npm install hbs --save
```

app.js

index.hbs

```
<div id="wrapper">
  <header>
    <h2>{{title}}</h2>
  </header>
  <div class="content">
    <p>Hello, {{name}}</p>
  </div>
</div>
```

```
var express = require('express');
var app = express();

app.configure(function () {
  app.set('view engine', 'hbs');
});

app.get('/', function (req, res) {
  res.render("index", { title:
    'Important Title', name: 'Bob' });
});

app.use(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain' });
  res.end('Hello, Middleware');
});

app.listen(3000);
```

# Sharing Templates Between Frontend and Backend

```
// register frontend partials, so they can be used on the backend  
var headerTpl = fs.readFileSync(__dirname + '/public/templates/header.hbs', 'utf-8');  
hbs.registerPartial('header', headerTpl);
```

```
<!DOCTYPE HTML>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <title>Server Side Templates Example</title>  
  <link rel="stylesheet" href="/stylesheets/  
style.css" />  
</head>  
<body>  
  {{> header}}  
  {{{body}}}  
</body>  
</html>
```

# Other Middleware

```
npm install less-middleware --save
```

```
var express = require('express'),
    lessMiddleware = require('less-middleware');

var app = express();

app.configure(function () {
  app.set('view engine', 'hbs');
  app.use(lessMiddleware({
    src: __dirname + '/public'
  }));
  app.use(express.static(__dirname + '/public'));
});

app.get('/', function (req, res) {
  res.render("index", { title: 'Important Title', name: 'Bob' });
});

app.get('/greeting/:name', function (req, res) {
  res.render("index", { name: req.params.name });
});

app.use(function (req, res) {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello, Middleware');
});

app.listen(3000);
```

# Callback Hell and the Promise(d) Land!



**YO DAWG, WE HEARD YOU LIKE  
CALLBACKS**

**SO WE PUT CALLBACKS IN YOUR CALLBACK SO  
YOU CAN CALLBACK WHEN YOU CALLBACK.**

memegenerator.net

```

var fs = require("fs"),
    resolve = require("path").resolve;

var fileData = [],
    interval = null,
    done = false,
    fileDir = resolve(__dirname, 'files');

fs.readdir(fileDir, function (err, files) {
    if (err) {
        console.log("error finding files: " + err);
    } else {
        files.forEach(function (file, index) {
            console.log("filename: " + file);
            fs.readFile(resolve(fileDir, file), "utf8", function (err, data) {
                if (err) {
                    console.log("error reading file: " + err);
                } else {
                    fileData.push(data);
                    if (index === files.length - 1) {
                        done = true;
                    }
                }
            });
        });
    }

    interval = setInterval(function () {
        if (done) {
            fs.writeFile(resolve(__dirname, 'out.txt'), fileData.join("\n"), "utf8", function (err) {
                if (err) {
                    console.log('Error writing file: ' + err);
                } else {
                    console.log('done');
                }
                clearInterval(interval);
            });
        } else {
            console.log('not done yet');
        }
    }.bind(this), 10);
});

```



```

var fs = require("fs"),
    comb = require("comb"),
    resolve = require("path").resolve;

var fileData = [],
    interval = null,
    done = false,
    fileDir = resolve(__dirname, 'files');

var fsp = {
  readdir: comb.wrap(fs.readdir, fs),
  readFile: comb.wrap(fs.readFile, fs),
  writeFile: comb.wrap(fs.writeFile, fs)
};

comb.async.forEach(fsp.readdir(fileDir), function (file, index) {
  var ret = new comb.Promise();
  fsp.readFile(resolve(fileDir, file)).then(function (data) {
    fileData.push(data);
    ret.callback();
  });
  return ret.promise();
}).then(function () {
  fsp.writeFile(resolve(__dirname, 'out.promise.txt'), fileData.join("\n"),
    "utf8").then(function () {
      console.log('done');
    });
}, function (err) {
  console.log("There was an error: " + err);
});

```

# Promises

- help manage callback hell
- encourages separation of success and failure logic
- error-bubbling

Comb provides plenty of methods for helping manage callbacks, as well as a million other things!

<http://c2fo.github.com/comb/index.html>

# Examples



# Example App: NebraskaJS Tweets

- written using ntwitter and socket.io modules
- First time using socket.io and connecting to twitter stream
- follows #nebraskajs tagged tweets in real time
- Try it now! tweet #nebraskajs
- Source available at [nicknisi / nebraskajs-tweets](https://github.com/nicknisi/nebraskajs-tweets)

# NebraskaJS Tweets!

