9–11 June with Remy Sharp

# NODE!

- I'm **Remy**
- Questions later **@rem** or remy@leftlogic.com
- I <3 JavaScript
- Questions: **ask & discuss**

Left Logic

## The Plan

1. Node & JavaScript
2. Patterns (callbacks, events, etc)
3. Debugging
4. Production
5. Your problems/questions/discussion

## Remember...

Questions?

Ask & discuss.

# Me & Node

- Was witness to the unveiling in 2009
- JAVASCRIPT!
- Now our entire code stack is Javascript

- jsbin.com
- nodemon
- 5minfork.com
- Full Frontal

# You?

# Node

## I like it because...

- V8 JavaScript engine on the server
- Event driven. Like your browser
- Non blocking. Like your browser
- JavaScript. Like your browser
- Cross platform. (Mostly) like your browser

**node**
**Runtime for JavaScript**

**npm**
**Package manager**

V8

## Create hello.js

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

```
// to run:
$ node hello.js
```

## Create hello.js

```
setInterval(function () {
  console.log('Hello world');
}, 200);

  // to run:
  $ node hello.js
```

# CLI & REPL

- node -p "..." (print result)
- node -e "..." (execute)
- node (as REPL)

## References

- nodejs.org/api/
- https://groups.google.com/forum/#!forum/nodejs
- #node.js@freenode.irc
- More: nodejs.org/community/

## The good stuff

- .trim, .forEach, .map, .bind, etc.
- It's not browser land, make use of the tools
- ES6 is coming, but you can play using node --harmony-generators

## Environment

```
$ PORT=8000 node app.js

// app.js
server.listen(process.env.PORT || 80);
```

# Environment

```
$ PORT=8000 node app.js

// app.js
server.listen(process.env.PORT || 80);

$ NODE_ENV=production node app.js

// app.js
if (process.env.NODE_ENV == 'production') {
  // load production settings
}
```

require

## Modules

```
var path = require('path'),
    fs = require('fs'),
    http = require'('http');
```

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

## Anatomy of a module

```
function squared(n) {
  return n * n;
}

module.exports = squared;
```

---

```
var maths = require('squared');
console.log(maths(3)); // 9
```

# Anatomy of a module

```
function squared(n) {
  return n * n;
}
```

Privately scoped to the module

```
module.exports = squared;
```

---

```
var maths = require('squared');
console.log(maths(3)); // 9
```

## Anatomy of a module

```
function squared(n) {
  return n * n;
}
```

Module as singleton

```
module.exports = squared;
```

---

```
var maths = require('squared');
console.log(maths(3)); // 9
```

```
function squared(n) {
  return n * n;
}

function times2(n) {
  return n * 2;
}

module.exports = {
  squared: squared,
  times2: times2
};
```

Module with API

module.exports
is a simple object

## Module loading

1. require('http'); // core node module
2. require('express'); // 3rd party module
3. require('./lib/myapp'); // my module (*note no .js*)

# Module loading

```
.
├── one.js                require('./one');
├── two
│   └── index.js          require('./two');
├── three
│   ├── module.js
│   └── package.json      require('./three');
└── node_modules
    └── four.js           require('four');
```

## Module loading

```
var foo = require('bar');
```

```
/home/remy/project/node_modules/bar.js

/home/remy/node_modules/bar.js

/home/node_modules/bar.js

/node_modules/bar.js
```

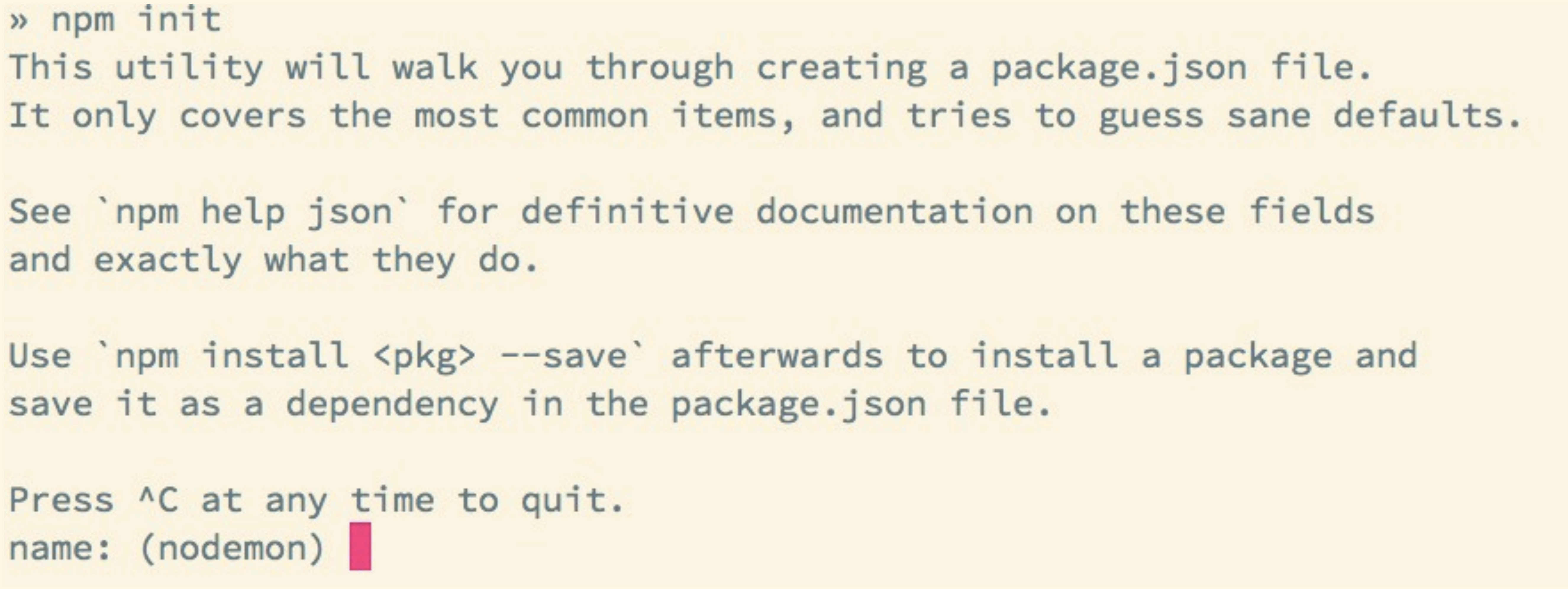
## Task

- Module: talk.js which will reply commands and console.log, *ie. talk.say, talk.shout, talk.think*
- Main: app.js which will load talk module

## Task

- Require in (any) module
- Try requiring in a module that doesn't exist
- Require in some JSON
- Create a module that requires in JSON

# npm init



```
» npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sane defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (nodemon)
```

## Create a module

- npm init
- Enter details in package
- Publish to npm
- Bump the version number*

# semver

- major.minor.patch
- patch = bug fixes
- minor = feature
- major = back incompatible changes
- 0.1.0 is first release - anything can change
- 1.0.0 is first "unstable" release
- Node employs: even major: stable, odd: unstable

## package.json

- Can be private
- Can be a utility and include a "bin" path
- Includes your dependancies
- Lots of useful info (like test command, scripts, authors, software license, git repo, etc)

# npm ftw

https://www.npmjs.org

```
$ npm install <package>[@x.x.x]
$ npm install --save <package>
$ npm install --save-dev <pkg>
```

## Automatic versions

```
"dependencies": {
  "connect": ">=1.8.2",
  "underscore": ">=1.0.0"
}, // ...
```

## Sane versions

```
"dependencies": {
  "connect": "1.8.x",
  "underscore": "^1.0.2"
}, // ...
```

- Global & local installs
  *Global is for utilities*
  *Local is for your app dependancies*
- Deploy your packages
- Automatic dependency management

package.json

+

git hooks

=

#win

# git pull

```
remy@jsbin:/WWW/jsbin$ git pull
remote: Counting objects: 19, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 11 (delta 8), reused 7 (delta 4)
Unpacking objects: 100% (11/11), done.
From git://github.com/remy/jsbin
   a7bf4f8..f99beff  master     -> origin/master
 * [new branch]      feature/menu-update -> origin/feature/menu-update
Merge made by the 'recursive' strategy.
 test/loop_detection_test.js |   12 +++++++++++-
 1 file changed, 11 insertions(+), 1 deletion(-)
rebuild jsbin.js? [Y/n] y
building public js
Running "concat:dist" (concat) task
File "public/js/prod/jsbin-3.4.8.js" created.

Running "concat:runner" (concat) task
File "public/js/prod/runner-3.4.8.js" created.

Running "uglify:dist" (uglify) task
Source Map "public/js/prod/jsbin.map.json" created.
File "public/js/prod/jsbin-3.4.8.min.js" created.

Running "uglify:runner" (uglify) task
File "public/js/prod/runner-3.4.8.min.js" created.

Done, without errors.
restart jsbin? [Y/n] y
restarting forever
info:    Forever restarted processes:
data:        uid  command             script forever pid   logfile                      uptime
data:    [0] 9QJb /usr/local/bin/node run.js 14763   22123 /home/remy/.forever/9QJb.log 0:0:11:18.741
done
remy@jsbin:/WWW/jsbin$
```

## git pull

```
#!/bin/sh

npm install
echo -n 'rebuild jsbin.js? [Y/n] '
read word < /dev/tty

if [ "$word" = "" ] || [ "$word" = "y" ] || [ "$word" = "Y" ]; then
   echo 'building public js'
   grunt build
fi

echo -n 'restart jsbin? [Y/n] '
read word < /dev/tty
if [ "$word" = "" ] || [ "$word" = "y" ] || [ "$word" = "Y" ]; then
   echo 'restarting forever'
   forever restartall;
fi
echo 'done'
```

Typically,

.gitignore prevents node_modules from saving in your git repo.

## Dependency locking

1. *Simplest*: commit node_modules to repo
2. *Smartest*: npm shrinkwrap
3. *Most secure*: custom package manager

asynchronous

## Node

```
function send(req, res) {
  res.writeHead(200, {
    'content-type': 'text/event-stream',
    'cache-control': 'no-cache'
  });

  var sendMsg = function () {
    var t = Date.now();
    var msg = 'time: ' + t;

    res.write('id: ' + t + '\n');
    res.write('data: ' + msg + '\n');
    res.write('\n');

    setTimeout(sendMsg, 1000);
  };

  sendMsg();
}
```

```php
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

function sendMsg() {
  // $t is the time, but also the message
  // id (as it's numerical);
  $t = time();

  // our message: "time: 12:34:23"
  $msg = 'time: ' . date("h:i:s", $t);
  echo "id: $t" . PHP_EOL;
  echo "data: $msg" . PHP_EOL;
  echo PHP_EOL;
  ob_flush();
  flush(); // send to client

  // wait 1 second, then send the time again
  sleep(1);

  // repeat
  sendMsg();
}

sendMsg();
```

PHP

## Node

```
function send(req, res) {
  res.writeHead(200, {
    'content-type': 'text/event-stream',
    'cache-control': 'no-cache'
  });

  var sendMsg = function () {
    var t = Date.now();
    var msg = 'time: ' + t;

    res.write('id: ' + t + '\n');
    res.write('data: ' + msg + '\n');
    res.write('\n');

    setTimeout(sendMsg, 1000);
  };

  sendMsg();
}
```

## Task

- Require fs and read a file
- Get the total size:
  - Read a directory of files
  - Make sure they're files (and not directories)
  - Get the file size
  - Echo out the total sum of all files
  - Exit script
- Bonus points for recursing subdirectories