Building an Angular and Express App Part 1

ck Your Next Idea A Quick, Vital Tip for New Mor

Posted on 12 April
by J Cole Morrison

This is somewhat of an update to one of my earlier posts, but delves into setting up a better front end workflow and also using Express 4.x. It will also continue into a series of posts on building out an actual app

Note: This has been updated for generator-angular 0.9. Please tell me if you run into any problems!

Part 2 - Here

Disclaimer: I don't claim to be an expert and would appreciate any helpful tips to improve this! I have put out production apps on this build though, so it's not just for pet projects!

Today folks, we'll be scaffolding out the beginnings of an application that uses the following:

1. Angular JS for the Front End
2. Yeoman + Generator Angular for developer happiness
3. Express + Node for the backend
4. Git for version control
5. And Deploying to either Digital Ocean or Modulus.io.... maybe both?

The app will do the following:

1. Use Express as a RESTful service
2. Handle user creation and accounts
3. Accept payments through Stripe
4. Implement some security

The above are a rough outline of all the things this will likely entail, but we'll add them as we go. I don't have a well defined plan for this at the moment, but this will probably be better for the learning process anyway.

This is going to be FAR more helpful to readers who have a basic grasp on the listed technologies above. I'll try and explain the "whys" and what not as I go, but I'll be skipping the most basic of things.

Without further ado...

Getting the Tools

First make sure you have Node.js installed. They have a nice package installer for all systems.

Go to your command line (that should be using bash, so Mac OSX on terminal and Windows on... Powershell?). Do the following:

```
$ npm install -g express
```

Express is a low level server framework. Read more about it here.

```
$ npm install -g express-generator
```

This is going to help us scaffold out the server side of things. Don't worry, it's made by the actual express people.

```
$ npm install -g yo
```

Yeoman is a beastly beast. It's going to make life much easier by putting front end files in the right place and optimize our distribution versions.

```
$ npm install -g generator-angular
```

Generator Angular is a partner that works with Yeoman. Don't worry both Yeoman and Generator Angular are built, supported, and maintained by some REALLY talented folks at Google. So you won't have to worry about it being unsupported anytime soon.

```
$ npm install -g bower
```

Bower is a nice package manager for front-end assets.

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

Nodemon is an awesome package that will keep our node server from having to be restarted each time we change the server side code.

Scaffolding Time!

On your command line, navigate to where you want this to happen.

```
$ cd ~/Documents/Code_Learnings/
```

The above is obviously just an example, go to wherever you want. Make the root project directory and two sub directories `client` and `server` to start. Also jump into the new `client` directory.

```
$ mkdir my_new_project && cd my_new_project
$ mkdir server && mkdir client && cd client/
```

In the client do:

```
$ yo angular
```

At this point you'll be asked a variety of different options. You want to use SASS / Compass because you're not a masochistic front end developer. You also want to use Bootstrap and Bootstrap sass. You also want to use all the extra angular goodies (routes, cookies, etc.) ALL of them. Make sure the little box next to them is green. At this point, NPM will do it's vomit of code.

Note: If you run into problems here try doing `npm cache clean` and a `bower cache clean`. If you're using these regularly, the two managers will try and cache old packages and mess up.

Now test that it's working by running:

```
$ grunt serve
```

This will result in a nice pretty webpage popping up at localhost:9000. After you've confirmed it's up, press `ctrl+c` to stop the server.

Yeoman, with all of its greatness, just set you up an awesome front end work flow and file structure. It's hard to understand just how powerfully cool it is right now, but once you start using it...it'll be REALLY hard to stop.

Setting up the Front End Workflow

Here we're going to tweak the way the SASS is structured, add font awesome, and alter some build settings. This is so that you can just make new SASS files and CSS away without any interference. This will also allow you to not edit the bootstrap SASS source which makes updating it later SO much easier.

1. In your client folder, open `bower_components/bootstrap-sass-official/vendor/assets/stylesheets/bootstrap.scss`
2. On line 2 comment out `@import "bootstrap/variables";`
3. On line 8 comment out `@import "bootstrap/glyphicons”;`
4. in your client folder, go to `bower_components/bootstrap-sass-official/vendor/assets/stylesheets/bootstrap/_variables.scss` and copy it and put it in `app/styles/`
5. in your client folder go to `app/styles/main.scss` get rid of the $icon-path line and put `@import "variables";` at the top.
6. in your client folder in `app/styles/` make a `partials` folder and a `vendor` folder

7. in app/index.html paste `<link href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.min.css" rel="stylesheet">` right before the end of the `<head>` tag.

8. in `app/index.html` on line 16 replace `<span class="glyphicon glyphicon-ok"></span>` with `<i class="fa fa-check"></i>`

9. in `app/index.html` on line 37 replace `<span class="glyphicon glyphicon-heart"></span>` with `<i class="fa fa-heart"></i>`

10. `ctrl+c` in your command line to stop grunt and then do `$ grunt serve` again to restart it.

The first part makes it so that you can edit your own bootstrap variables file to customize colors, fonts, and such.

7 - 9 are to get font-awesome setup and tested. The only reason we stopped the grunt server (step 10) this time is that sometimes it needs an actual reset when you tamper with bower components.

Setup the Front End Build Process

Yeoman gives us the ability to run a test version of your app and then build it with all sorts of optimized goodness. We're going to change the target of our optimized build. In `client/Gruntfile.js` on line 25, replace `dist: 'dist'` with `dist: '../server/dist'`.

Okay, now open up another command tab and navigate to your server folder in it. On mac it's `cmd+t` and then `$ cd ../server/`.

Scaffold Out the Back End

Now we're going to setup Express JS. They team that created Express also made a nice little generator (you installed it earlier with `npm install -g express-generator`) that scaffolds a basic backend up. We're going to use

that and then delete what we don't need vs. coding everything ground up. I find it faster and fewer keystrokes to do it this way (and I don't have to worry about forgetting some random setting).

The summary of these changes is that it makes it so we serve the Angular app by default and let it handle routing. We do this by leveraging Express' "static" directory ability. In addition, we also set it up so that we can serve it in a `production` mode and a `development` mode. The `production` mode will serve from our `server/dist/` folder which is where Yeoman will put our optimized app. The `development` mode will serve the test version hosted in the `client` folder.

The beauty of this setup is that it will allow us to leverage express as a RESTful service. It will also makes it so that our `server` folder is the only thing we worry about deploying.

In the command line in your `server` tab:

```
$ express
```

Some npm vomit will happen.

```
$ npm install
```

This will put any dependencies that you need in your `server` directory for you. Additionally make another directory called `dist` in your `server` folder.

```
$ mkdir dist
```

Okay, now to modify some code. Make sure you're in the `server` folder for all of this:

1. Delete the `public` folder.
2. Delete the `views` folder.
3. Open your `app.js` file and delete lines 14 & 15

```
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
```

4. Delete the line

```
app.use(express.static(path.join(__dirname, 'public')));
```

5. Also delete the two route lines:

```
app.use('/', routes);
app.use('/users', users);
```

Finally, replace everything under `app.user(cookieParser());`, EXCEPT the `module.exports = app;` bit, with the following:

```
/**
 * Development Settings
 */
if (app.get('env') === 'development') {
    // This will change in production since we'll be using the
dist folder
    app.use(express.static(path.join(__dirname, '../client')));
    // This covers serving up the index page
    app.use(express.static(path.join(__dirname,
```

```
'../client/.tmp')));
    app.use(express.static(path.join(__dirname,
'../client/app')));

    // Error Handling
    app.use(function(err, req, res, next) {
        res.status(err.status || 500);
        res.render('error', {
            message: err.message,
            error: err
        });
    });
}

/**
 * Production Settings
 */
if (app.get('env') === 'production') {

    // changes it to use the optimized version for production
    app.use(express.static(path.join(__dirname, '/dist')));

    // production error handler
    // no stacktraces leaked to user
    app.use(function(err, req, res, next) {
        res.status(err.status || 500);
        res.render('error', {
            message: err.message,
            error: {}
        });
    });
}
```

Almost there. Now for one final change that will make our development life a tad bit easier. Open up `server/package.json` and replace line 6 with the following 2 lines:

```
"start": "NODE_ENV=production nodemon ./bin/www",
"test": "NODE_ENV=development nodemon ./bin/www"
```

This makes it so that we can easily shift between a "production" test and a "development" test. It also leverages `nodemon` to make it so that when we change server code, that the server will restart automatically vs. having to be manually shutdown and restarted.

Here's a couple of gists to help out:

`server/app.js`

`server/bin/www`

`server/package.json`

9/3 Update

**Note: Newer versions of express don't include the `static-favicon` middleware by default. Therefore you'll need to use the newer replacement:

https://github.com/expressjs/serve-favicon

Make sure you serve the favicon from the correct directory. If you're keeping it in your `client` code then it'll be located in the `server/dist` directory. Otherwise you can just copy it to your `server/public` directory and serve it from there.

Thanks to @PMBernstein for catching that.

Now Fire Everything Up!

Make sure you have two command line tabs open. One that's inside of your `client` folder and one that's inside of your `server` folder.

In the `client` folder command line tab do a `$ grunt serve` and let the thing fire up on localhost:9000 in your web browser. Close that web browser tab, you don't need it.

In the `server` folder command line tab do a `$ npm test` and your express server will begin running on localhost:3000. Open that up in a web browser and you should see the same Yeoman welcome page that your `grunt serve` method gave you. The difference is... THIS one is being served by your express server!

Make it Production Ready

When you want to build things into production mode you'd shut both your `grunt` server and your express server down (so a `ctrl+c` in your `client` command line tab and your `server` command line tab). Next, in your `client` tab, run

```
$ grunt --force
```

And Yeoman will optimize/jshint/minify your client side code, CSS, images, etc, and put the fresh version in your `server/dist` directory. To test it out go to your `server` tab and run

```
$ npm start
```

And your production ready version will fire up on localhost:3000! After doing these steps, your `server` folder is what you'd deploy to your hosting

service of choice.

Note: if you used just plain `$ grunt` instead of `$ grunt --force` it would run any angular unit tests you have setup.

Git It

Back out to your root folder and do the following:

```
$ git init
$ git add -A
$ git commit -m "Express and Angular Scaffolding"
```

In between each of those you'll see some logs. This will set us up for painless experimentation later in both the build process and actual app creation.

So Why 2 Servers?

A few reasons:

1. The Grunt Server handles keeping SASS compass live for you, compiling SASS into css, adding in Angular and Bower components in a Ruby-Rails-Style, and building out your optimized production version.
2. When you need to update either the server specific code, or client specific code, it doesn't wind up in a "Oh shit, now I have to update everything and pull apart this mess I've made."
3. Current generators that package all of these things into 1 command line tool aren't officially supported by the teams that make Angular or Express. I personally find these hard to use in production since many of them often turn into abandonware.
4. The pieces are far more decoupled, and it forces you to keep your REST service available for future possiblities.
5. Because Yeoman is just so worth it. You'll see.

Awesome. Summarize?

So in a short article, we meshed together 4 of the coolest web development workflow frameworks/tools. Yeoman and Angular will make front-end development FUN since you'll have access to a Rails-style generation process. And Express and Node will give us access to the thousands upon thousands of great packages out there to extend our app infinitely.

Next up, the front end build.

About the author

J Cole Morrison

Sacramento, CA

http://jcolemorrison.com

Comments

Recommend 6 Share Sort by Best

Join the discussion…

Fadel Chafai • 2 years ago

Thank for the tutorial, just for the bower_components folder bower put it on the client folder and when i start the npm test on the server folder the browser can't load all bower files

Failed to load resource: the server responded with a status of 404 (Not Found)
http://localhost:3000/bower_components/jquery/dist/jquery.js

the solution i found is to copy the bower_components in the client/app folder (duplication)

grunt need bower_components in the client folder but the server test need bower_components on the client/app folder ????

4 • Reply • Share ›

CityLims ➚ Fadel Chafai • 8 months ago

yeah, that's a minor hiccup with this tutorial. You can also update your .bowerrc file to put the bower_components into the app dir
{
"directory" : "app/bower_components
}

• Reply • Share ›

Landry Soules • 2 years ago

Thanks a lot for this brilliant tutorial ! Just a note for Windows users: in order to make server/package.json work, you need to replace

"start": "production.sh",
"test": "test.sh"

in the same folder, create
production.sh containing: set NODE_ENV=production && nodemon ./bin/www
test.sh containing: set NODE_ENV=development && nodemon ./bin/www

3 • Reply • Share ›

Chintan Pathak ➚ Landry Soules • 2 years ago

I dont know why, but on my system, the .sh files dont work, and I found that doing "start": "set NODE_ENV=production && nodemon ./bin/www.js" works and I also had to change the app.js if statements to

if(app.get('env' === 'production ')

Apparently the set command adds and extra space, so app.get('env') returns 'production '

3 • Reply • Share ›

SKY ➚ Chintan Pathak • a year ago

Please edit comment to : if(app.get('env') === 'production ')

• Reply • Share ›

Cole Morrison Mod ➚ Landry Soules • 2 years ago

Whoa, thanks for the advice for windows!

1 • Reply • Share ›

Pascal Heitz ➚ Cole Morrison • a year ago

@Cole Morrison Thank you for your great tutorial!
@Landry Soules Thank you for the tip on Windows!

For me,
set NODE_ENV=development && nodemon ./bin/www
doesn't work. The variable `process.env.NODE_ENV`
`server/database/index.js`is `undefined`, so mongoose doesn't connect to any database...

I had to change it to
NODE_ENV=development nodemon ./bin/www
(bash style)

I believe it's because I use gitbash.

• Reply • Share ›

Piyush Rahate • 2 years ago

That was a quick starter, but I kept wondering why the index page was not styling up.
I am Windows and when I did grunt serve it threw warnings but with grunt --force it worked.
And then it was the "bulb" moment, I didn't have Ruby/Compass installed.

However, I am too having the issue of setting up the environments (the NODE_ENV thing), it simply doesn't work on Windows. I keep getting the message "cannot /GET".

For now working without that. Great article nevertheless. I was going through all the resources on net just to figure out how the project should be structured and this did it.

Thanks a ton!!

Quick Tip for Windows guys: First install Ruby and Compass.
You can get the Ruby installer from rubyinstaller.org. Once that is done simply run "gem install compass", and you are set to go.

2 • Reply • Share ›

Cole Morrison Mod ➚ Piyush Rahate • 2 years ago

Hey NP! Sorry about the problem with Windows, I haven't tried this on a PC.

2 • Reply • Share ›

Landry Soules ➚ Piyush Rahate • 2 years ago

Hi, just saw your comment. Have a look at mine, it solves the NODE_ENV thing.

• Reply • Share ›

Piyush Rahate ➚ Landry Soules • 2 years ago

Hey Landry,
Which comment are you referring to?
I do not see any other comment by you.

Can you tag me?

• Reply • Share ›

Jesús Ugueto ➚ Piyush Rahate • 2 years ago

Hi, the comment he is talking about is right on top of yours. the problem is that i did try it, and although the server runs (wich it didn't before the fix), i get the "Cannot/GET" message on localhost:3000. Could it be a problem with ./bin/www? **@Landry Soules @Piyush Rahate @Cole Morrison**

• Reply • Share ›

Cole Morrison Mod ➚ Jesús Ugueto • 2 years ago

Heya! I can't say exactly (since I don't use windows) but one of the other commenters said he found a way to work with the whole set node_env thing:

http://start.jcolemorrison.com...

• Reply • Share ›

Jesús Ugueto ➚ Cole Morrison • 2 years ago

Hey men thanks for the quick response, but sadly, like i said, i did try that solution, "test": "SET NODE_ENV=development && nodemon ./bin/www", and this get the server running but cannot get the index on the browser.

On a side note, Great tuts, hope you finish them soon, i've been trying to deploy an app on digital ocean but can't get it to work, im expecting to solve it soon thanks to your tutorial an another i found from Chris Esplin, here's the link http://www.christopheresplin.c...

I shall continue fighting with the prompt hehehe

1 • Reply • Share ›

vignesh velsubbu ➚ Jesús Ugueto • 2 years ago

"SET NODE_ENV=development && nodemon ./bin/www"

remove space between development and n&&

"SET NODE_ENV=development&& nodemon ./bin/www"

otherwise like @pathak siad add space in ln last 'production '

if(app.get('env' === 'production ')

1 • Reply • Share ›

Vinicius Rosa ➚ vignesh velsubbu • 2 years ago

Another approach is to use escaped double quotes around

Another approach is to use escaped double quotes around name=value in set:

"set \"NODE_ENV=development\" && nodemon ./bin/www"

2 • Reply • Share ›

lexan ➚ Vinicius Rosa • 2 years ago

It is fix npm test for Windows. Thanks

1 • Reply • Share ›

Wilmer E. López Rivas ➚ lexan • a year ago

how?

• Reply • Share ›

scaryguy • 2 years ago

I think you'd better add a small note about being sure to have grunt-cli and compass packages globally added. I didn't have both of them and have been struggling with trying to work it since an hour... I was getting "unhandled error" during "yo angular" and "compass error" during "grunt serve".

2 • Reply • Share ›

Alexander Wolf • 2 years ago

Great tutorial, thanks.

A tip for deploying to Heroku. For deplyoment I had to change the following to make it work:
- In package.json I modified start script to: "start": "node ./bin/www",
and set the env with $heroku config:set NODE_ENV=production
- Then push your app from the root directory with $git subtree push --prefix heroku master (of course, before you have to setup heroku and build the app with $grunt build --force and commit it.)
- Now start a dyno with: $heroku ps:scale web=1
- Finally if everything is OK, you can open your app with $heroku open

That's working for me. If anyone has a better solution than setting the environment manually, please post it.

1 • Reply • Share ›

Timo Sand • 2 years ago

I've followed your tutorial to the T and the client side works fine, but the server doesn't work after removing the view engine config. I get a 'Error: No default engine was specified and no extension was provided', whenever I try to open anything on the server

1 • Reply • Share ›

Timo Sand ➚ Timo Sand • 2 years ago

Ah, fixed it. My static routers were after the errorhandler, so it didn't register them...

1 • Reply • Share ›

Wilmer E. López Rivas • 2 years ago

Thanks, good job

1 • Reply • Share ›

Cole Morrison Mod ➚ Wilmer E. López Rivas • 2 years ago

Thanks! Sorry for any weird fall behinds, looks like a bunch of the dependencies that updated decided to change around the directory structure and naming.

• Reply • Share ›

Eric Shoberg • 11 days ago

The lines for start and test need to be as follows for windows:

"start": "set NODE_ENV=production&& nodemon ./bin/www",
"test": "set NODE_ENV=development&& nodemon ./bin/www"

Note especially the lack of a space between the environment variable and the &&.

• Reply • Share ›

cnakea • a month ago

After creating about 6 instances, for some reason all new instances compile a vendor.js file with the same ID and it can't find the angular modules. I've build them in clean directories and it still does the same thing. I've also cleared npm and bower cache. Any ideas?

• Reply • Share ›

Mateo Cuervo Taylor • 5 months ago

great tutorial hopefully the security part comes along

• Reply • Share ›

Dave Munger • 8 months ago

I know this is an old post, but it's the closest thing I could find to help me learn setting up the stack. Unfortunately, I have no directory called bower_components. Also, when I run grunt serve I get:
Warning: Error: Cannot find where you keep your Bower packages. Use --force to continue. Aborted due to warnings.

Edit: I should have mentioned, yes, I have installed the bower package before getting to this error, and having the missing directory.

Edit 2: Although I have slightly different symptoms, I read one of the comments below and realized I didn't have compass installed. Doing that solved these problems for me too.

• Reply • Share ›

lokesh jain • 8 months ago

Thanks, i was looking for this post for a long time. I wish i could have found it before.

• Reply • Share ›

Bessie Malek • 8 months ago

What I heard is that newer versions of node don't get/load/install some things that it used to. Before I could do

```
```

grunt serve

```

I needed to do these commands while I was in the client directory:

npm install -g grunt-cli

npm install -g grunt-wiredep

bower install jquery

grunt wiredep

sudo gem install sass

sudo gem install compass



**Richard Lu** • 9 months ago

When I tried a grunt serve on the client it threw

Completed in 2.178s at Sun Nov 22 2015 16:56:08 GMT-0600 (CST) - Waiting...

>> File ".tmp/styles/main.css" changed.

>> File "app/styles/main.scss" changed.

Running "compass:server" (compass) task

error app/styles/main.scss (Line 1: File to import not found or unreadable: variables.

Load paths:

/Users/richardlu/express-angular/client/bower_components

Compass::SpriteImporter

/Users/richardlu/express-angular/client/app/styles

/Library/Ruby/Gems/2.0.0/gems/compass-core-1.0.3/stylesheets)

Compilation failed in 1 files.

Warning: ↑ Used --force, continuing.

I am not exactly sure what its trying to do with the @import "variables"; statement in the main.scss. I removed that statement to continue with the tutorial; however, I would like to understand the issue if you can provide that. Thanks!



**Lautaro Silva** • 10 months ago

Thanks!!!

Please update the info for install angular-generator with node 4.2.2
```

npm install -g grunt-cli bower yo generator-karma generator-angular

Greate tutorial!

• Reply • Share ›

rambutan • 10 months ago

this is a great tutorial, but if you don't want to have to change the bootstrap sass files (and i don't), then you can achieve the same result by pasting the relevant variable lines at the top of main.css, but mind that you *remove* the __ !default __ tags.

main.scss
=======
$brand-success: #04bf9d;
$brand-info: #34d0ca;
$brand-warning: #40ad4e;
$brand-danger: #f53d54;
$body-bg: #fbfbff;

• Reply • Share ›

Aaron Melocik • a year ago

Wow. So good! Thank you!

• Reply • Share ›

James Van Leuven • a year ago

Hola,

so it's been awhile since I've been on this website, but I came back to do a quick run through and see if this worked as efficiently on OSX, Ubuntu, and CentOS and it seems it's still a great tutorial. Of course I'm using different flavours of NodeJS, MongoDB, and so on, but the Angular version is still the same. Pulling in Bootstrap/Font-Awesome of course changes but they don't have any adverse affect. I still had to install the Compass Server and it might be an idea to include that in this tute for folks who are running into that error on the grunt serve segment :)

cheers!

• Reply • Share ›

Brub ➚ James Van Leuven • 9 months ago

can you tell us how to install the Compass Server — I'm having problems getting around that.

• Reply • Share ›

Matthew Harris • a year ago

decent jobs, you failed to teach how to install compass, which was a struggle, over great work, thanks

• Reply • Share ›

Josh • a year ago

Great tutorial!!!

• Reply • Share ›

ilias • a year ago

Hi ! Really great tutorial, everything works just perfect ! I am facing some difficulties to push to prod. I am working with heroku and it doesn't even build source when i make a push... I get :
"Push rejected, no Cedar-supported app detected
HINT: This occurs when Heroku cannot detect the buildpack to use for this application automatically."
If anyone figured out how to push to heroku, I'd really appreciate some help !
(I tried following **@Alexander Wolf** comment but it is not doing any good)

• Reply • Share ›

ilias ➚ ilias • a year ago

Ok my bad, Alexander Wolf was right. Just you need to push server folder only with : git subtree push --prefix server heroku master

• Reply • Share ›

Iye • a year ago

This might help someone. I followed this awesome tutorial, but when I went to use a plugin (ui-bootstrap), glyphicons were broken. The fix is to change line 84 on _variables to:

$icon-font-path: if($bootstrap-sass-asset-helper, "/bootstrap/", "../../bower_components/bootstrap-sass-official/assets/fonts/bootstrap/") !default;

Cheers.

• Reply • Share ›

James Van Leuven • a year ago

I have to say, if it wasn't for this specific tutorial, I don't know if I would have dived into Angular with such vigor. I can't thank you enough! I'm starting to redev an app I previously built using php/sql server, and I can totally see how Angular will optimize my previously bloated jQuery ajax, dom, and module management.

I bow to you oh great jedi master! The force is strong with you!

• Reply • Share ›

Ahmed Masud • a year ago

This was awesome!

• Reply • Share ›

James Van Leuven • a year ago

Ok! As promised I said I'd give you the mod deets based upon your tute.

My environment is:

Windows 7 x64

MongoDB 3.0.1 (x64)

Node v0.10.36 (x86)

NPM v2.8.4

Node-Gyp v1.0.3

Ruby v2.4.5 (this is an important inclusion)

I also have installed the Node dependencies, and recommend you follow all the stops BUT install VSExpress v10, v12, v13 for Desktop. v12 no longer holds the binaries, but v13 references v12 and suddenly installs them (it's a Microsoft typical 'let's not add this to the README crap').

---

see more

• Reply • Share ›

Cole Morrison Mod ➚ James Van Leuven • a year ago

Wow, this is great stuff man. Thanks!

• Reply • Share ›

James Van Leuven • a year ago

I'm beginning to think that updating node to 0.12.x was a BAD move. It seems as though not a single package is updated to function with this version of node. Oddly for me though, I have mongo, yeoman, node, grunt, gulp, bower, git, node-gyp all installed without any problems... all with the last builds, so they're all up to date but...

I know this is an excellent tutorial but I am suffering from installation hell whenever I want to install a generator it just goes into the scrolling red screen of death... :(

any ideas?

• Reply • Share ›

Cole Morrison Mod ➚ James Van Leuven • a year ago

Hey James,

Sorry, I've been really busy and haven't had time to go through and update these. I plan to this month so I'll keep you posted.

My thing is... when they update for code efficiency, GREAT. But when the updates are namespacing and name changing ones... ugh.

-J Cole Morrison

• Reply • Share ›

James Van Leuven ➚ Cole Morrison • a year ago

I hear ya. I decided not to be stubborn. I just went back to node v0.10.36. I'm not stoked that I've been forced back to x86 but the more you know...

I'll let you know how that goes... I probably have to change my npm version as well... :)

• Reply • Share ›

Tyler Fowler • 2 years ago

`mkdir some_folder & cd $_`

Found this little gem a while back, create folder and cd into it without typing out the name of the directory twice.

• Reply • Share ›

Sung Won Cho • 2 years ago

Thank you so much for your tutorial. I have been looking for tutorials about using angular and express together, and this is the most helpful one I have found on the web.

• Reply • Share ›

Load more comments

ALSO ON **START.JCOLEMORRISON.COM**

Yer a Hacker Harry!

3 comments • 2 years ago•

Avat **Steve** — If you define magic as simply an arcane set of knowledge where you can create things, manipulate things, or …

Setting up an AngularJS and Rails 4.1 Project

50 comments • 2 years ago•

Avat **Abe Kinney** — fantastic!!!! don't know how many poorly written angular/rails tutorials that leave me dead in the water. this was …

Building an Angular and Express App Part 2

19 comments • 2 years ago•

Avat **Vishal Shah** — I had the same issue until I realized that the sigup.html code in the GIST is missing the controller …

How I Fight Procrastination (and worry)

15 comments • 2 years ago•

Avat **Cole Morrison** — Haha, you know what I mean. When people say "Just do it." with respect to the whole thing itself. So like,"I …

Subscribe **Add Disqus to your site Add Disqus Add** **Privacy**