Getting Gulp installed

Step One - Install Node

Download Node

First - we need our most important requirement, Node. To install Node - simply visit http://nodejs.org and then click that big green "Install" button. Once your download completes, run that application and you should be all ready to go. The Node installer also includes npm, which we will come back to a little later.

Step Two - Get To Know Your Command Line

Now, you may not be very familiar with your command-line interface (Terminal for OSX, Command Prompt for Windows) but you should be! It may seem intimidating at first, but once you get the hang of it you will have the ability to run many different command line applications such as Sass, Yeoman and Git. All of which are very useful tools that your workflow could benefit from!

> If you are familiar with your command-line interface, then feel free to skip to step four.

As a quick example, open up your command line and we will throw a couple commands at it to ensure that Node is properly installed.

```
node -v
```

Type that and then hit enter and you should get a response on the next line with the version number of Node that you have installed.

Now, let's do the same for npm.

```
npm -v
```

Again, this should return the version number on the next line.

If you didn't get a response, then it may mean that Node didn't install correctly or you may need to restart your command line application. If this still isn't working after restarting, then simply jump back up the top and try the first step again.

Step Three - Navigate To Your Project Directory

Now that we have met our command-line interface and know how to communicate with it, our next step will be navigating it. Luckily, it only takes two commands to change directories and take a look at what is inside them. These commands are ls (or dir, for Windows) to list what is in a directory and cd to change directories.

> I suggest that you spend some time playing with these commands. Get used to your file system and be aware of where everything lives. Don't rush through this - it will save you a lot of headache later!

Once you are comfortable with the ls and cd commands, we need to navigate to our project folder. This will likely be different for each person, but as an example this is what I would type to navigate to my local project:

```
cd /Applications/XAMPP/xamppfiles/htdocs/my-project
```

It is important to note that I am working on OS X. The file system on Windows is much different so, while this example may be similar, it wont translate directly to Windows.

Once you have made it to your project directory - let's run a quick npm command to initialize our package.json file.

```
npm init
```

This will create a file in the root directory of your project called package.json which will provide information about our project and help manage our dependencies. Now we're ready to install gulp!

Step Four - Installing gulp You've met your command-line and you know how to talk to it - you even know your way around your file system. Now, it's time to get to the good stuff. Let's meet npm and install gulp!

NPM stands for Node Package Manager and it is a command line tool that will allow you to install additional Node packages to your projects. It even comes with a nifty site that allows you to browse and search through all of the available packages.

In your command-line application, type:

```
sudo npm install -g gulp
```

Let's quickly break this down.

1. sudo is the command to grant you administrator access so you can properly install files. It will simply ask you for your computer password. (This may or may not be needed - but it wont hurt. I've included it to avoid permissions issues that some folks may run into.)
2. npm is the application we are using to install our package.
3. We are running the install command on that application.
4. The -g is an optional flag used to signify that we want to install this package globally so that any project can use it.
5. And finally, gulp is the name of the package we would like to install.

Once that has run it's course check your command-line to ensure that there are no error messages. If there are none to be seen, then congratulations! You just installed gulp! Just to double check, let's refer back to our versioning commands we used above for Node and npm.

```
gulp -v
```

Like before, this should return the version number on the next line of your command-line.

Next, we also need to install gulp locally.

```
npm install --save-dev gulp
```

The only thing different here is we used the `--save-dev` flag which instructs npm to add the dependency to our devDependencies list in our package.json file that we created earlier.

Dependencies help us organize which packages are needed in our development and production environments as others contribute to or use our project. If you would like to read more about dependencies be sure to check out the package.json documentation.

Now that gulp is installed, the next step is setting up our gulpfile. We're almost done!

Step Five - Setting Up Our Gulpfile & Running Gulp

Once gulp is installed we have to give it some instruction so it knows what tasks for perform for us. But, first, we need to figure out exactly what tasks we need to run in our project. Time for… a SCENARIO.

In our Exciting Non-Generic Real World Scenario®, our boss has assigned us with the following tasks:

- Lint our JavaScript. (Seriously. Do it.)
- Compile our Sass files. (Browsers can't read that stuff…)
- Concatenate our JavaScript. (Reduce HTTP Requests!)
- Minify and rename said concatenated files. (Every little bit counts!)

I'm imagining our supervisor as the impatient, somewhat frightening type who eats interns when they don't do what the boss wants. So, let's get right to it before our lunch companion gets eaten.

Install Required Plugins

```
npm install gulp-jshint gulp-sass gulp-concat gulp-uglify gulp-rename --save-dev
```

This will install all of the plugins we will need and add them to our devDependencies in our package.json file like we did when we installed gulp.

As a reminder, if you are getting permissions errors installing these plugins you may have to prepend those commands with sudo!

Create Our gulpfile

Now that our plugins are available for us to use, we can start writing our gulpfile and instructing gulp to perform the tasks our boss assigned to us.

Before we get right into the code I think it's very important to mention that gulp only has 5 methods. These methods are as follows: task, run, watch, src, and dest. These are all you will need to write your tasks.

In the root directory of your project create a new file and name it gulpfile.js and paste the following code inside.

gulpfile.js

```
// Include gulp
var gulp = require('gulp');

// Include Our Plugins
var jshint = require('gulp-jshint');
var sass = require('gulp-sass');
var concat = require('gulp-concat');
var uglify = require('gulp-uglify');
var rename = require('gulp-rename');

// Lint Task
gulp.task('lint', function() {
    return gulp.src('js/*.js')
        .pipe(jshint())
        .pipe(jshint.reporter('default'));
});

// Compile Our Sass
gulp.task('sass', function() {
    return gulp.src('scss/*.scss')
        .pipe(sass())
        .pipe(gulp.dest('css'));
});

// Concatenate & Minify JS
gulp.task('scripts', function() {
    return gulp.src('js/*.js')
        .pipe(concat('all.js'))
        .pipe(gulp.dest('dist'))
        .pipe(rename('all.min.js'))
        .pipe(uglify())
        .pipe(gulp.dest('dist'));
});

// Watch Files For Changes
gulp.task('watch', function() {
    gulp.watch('js/*.js', ['lint', 'scripts']);
    gulp.watch('scss/*.scss', ['sass']);
});

// Default Task
gulp.task('default', ['lint', 'sass', 'scripts', 'watch']);
```

Now, let's break this down and review what each part does.

Core & Plugins

```
// Include gulp
var gulp = require('gulp');

// Include Our Plugins
var jshint = require('gulp-jshint');
var sass = require('gulp-sass');
var concat = require('gulp-concat');
var uglify = require('gulp-uglify');
var rename = require('gulp-rename');
```

This includes the gulp core and plugins associated with the tasks that we will be performing. Next, we setup each of our separate tasks. These tasks are lint, sass, scripts and default.

Lint Task

```
// Lint Task
gulp.task('lint', function() {
    return gulp.src('js/*.js')
        .pipe(jshint())
        .pipe(jshint.reporter('default'));
});
```

Our lint task checks any JavaScript file in our js/ directory and makes sure there are no errors in our code.

Sass Task

```
// Compile Our Sass
gulp.task('sass', function() {
    return gulp.src('scss/*.scss')
        .pipe(sass())
        .pipe(gulp.dest('css'));
});
```

The sass task compiles any of our Sass files in our scss/ directory into .css and saves the compiled .css file in our css/ directory.

Scripts Task

```
// Concatenate & Minify JS
gulp.task('scripts', function() {
    return gulp.src('js/*.js')
        .pipe(concat('all.js'))
        .pipe(gulp.dest('dist'))
        .pipe(rename('all.min.js'))
        .pipe(uglify())
        .pipe(gulp.dest('dist'));
});
```

The scripts task concatenates all JavaScript files in our js/ directory and saves the ouput to our dist/ directory. Then gulp takes that concatenated file, minifies it, renames it and saves it to the dist/ directory alongside the concatenated file.

Watch Task

```
// Watch Files For Changes
gulp.task('watch', function() {
    gulp.watch('js/*.js', ['lint', 'scripts']);
    gulp.watch('scss/*.scss', ['sass']);
});
```

The watch task is used to run tasks as we make changes to our files. As you write code and modify your files, the gulp watch() method will listen for changes and automatically run our tasks again so we don't have to continuously jump back to our command-line and run the gulp command each time.

Default Task // Default Task gulp.task('default', ['lint', 'sass', 'scripts', 'watch']); Finally, we have our default task which is basically a wrapper to our other tasks. This will be the task that is ran upon entering gulp into the command line without any additional parameters.

Now, all we have left to do is run gulp. Switch back over to your command-line and type:

```
gulp
```

This will call gulp and run everything we have defined in our default task. So, in other words It's the same thing as running:

```
gulp default
```

Additionally, we don't have to run the default task. We could run any of the tasks we defined at any time. Simply call gulp and then specify the task you would like to run directly afterward. For example, we can run our sass task manually at any time like so:

```
gulp sass
```

Pretty cool, eh?

Wrapping Up

Well, you've made it. We have survived our tempermental, intern-eating boss and you have heroically saved your lunch buddy. Take a moment to pat yourself on the back. As a quick recap, let's review what we have learned.

- We learned how to install Node.
- We got to know our command-line.
- We learned how to navigate our command-line.
- We learned how to use npm and install gulp.
- We learned how to write a gulpfile and run it.
- It is my hope that this introduction has made understanding task runners much easier and that you can see the real value that gulp adds to your project and your development workflow. If you have any further questions be sure to post them in the comments!

Note: http://travismaynard.com/writing/getting-started-with-gulp