Contents

[Using LESS CSS 1](#_Toc401049589)

[Compiling 1](#_Toc401049590)

[Variable Declarations 2](#_Toc401049591)

[Mixins 2](#_Toc401049592)

[Nested Rules 2](#_Toc401049593)

[Functions & Operations 3](#_Toc401049594)

[Using Grunt 3](#_Toc401049595)

[Searching 3](#_Toc401049596)

[Uglify and packages 4](#_Toc401049597)

[Using Less with Grunt 5](#_Toc401049598)

# Using LESS CSS

## Compiling

1. Install Node.js and the make sure it is on your PATH environmental by creating a variable called NODE, then add it to the PATH so that it can be run on the command prompt.
2. You should run the following on the command line if you are having proxy setting such as being behind a proxy. Changing the ip to the ip of your proxy.

npm config set https-proxy "http://10.64.1.95:8080/"

npm config set proxy "http://10.64.1.95:8080/"

npm config set strict-ssl false

npm config set registry http://registry.npmjs.org/

1. Next install the less via the following command

npm install –g less

1. This will install it in the roaming directory so just copy it out and put it near your Node .js directory.
2. In this case is added the following to my PATH environmental variable C:\Users\andrew.powell\AppData\Roaming\npm
3. Now create an environmental variable called LESS and point it to the Less directory, then add it to your PATH environmental variable.
4. Now go to the bootstrap src code directory and run the following command

lessc ./less/bootstrap.less > bootstrap.css

1. This will create a bootstrap.css file; you can also use the –compress or the -x argument on the command to make it a bit smaller.

## Variable Declarations

Variables are define as follows with the @ followed by the name and then the CSS

@black: #000;

It can then be used in a .less file as follows

#header {

color: @black;

}

## Mixins

Mixins allow you to embed classes in classes and can behave as functions taking parameters, this reduces the need for duplicate code.

.rounded-corners {

border-radius: @radius; border-radius: 5px;

}

#header {

.rounded-corners;

}

#footer {

rounded-corners(10px);

}

## Nested Rules

This allows you to next the CSS instead having sometimes confusing inheritance maps

#header {

h1 {

font-size: 20px;

}

p {

font-size: 12px;

a {

text-decoration: none;

}

}

}

## Functions & Operations

This allows you to do calculations on the CSS values as follows; the below will create a border 4 times as big as the defined variable, it will also add the 2 colours together, but I don’t really see much use for this at the moment.

@standard-border: 1px;

@standard-color: #ff2200;

#header {

border: (@standard-border \* 4);

color: (@standard-color + #003000);

}

# Using Grunt

1. To install Grunt Command Line Interface (CLI) you can run the following, note the –g specifies to install it globally. Note that if we only wanted a local copy then you wouldn’t use the –g option.

npm install –g grunt-cli

1. This will put the grunt command on your environmental PATH variable.
2. You will then need a package.json file and a Gruntfile.js
3. The package.json will define what grunt dependencies and meta data used.

## Searching

1. You can search for a package by typing the following

npm search uglify

1. This will index for the first time and then return a list of packages that contain the text after the search keyword
2. You can then view the package details by typing

npm info grunt-contrib-uglify

1. To view the web page you can enter the following

npm repo grunt-contrib-uglify

## Uglify and packages

**package.json**

Note the tilde (~) in the version number as this will only get the latest version for the number defined. e.g. it will only ever get the latest 0.4.x *patch* version.

{

"name": "project-name",

"version": "0.1.0",

"author": "Your Name",

"devDependencies": {

"grunt": "~0.4.1",

"**grunt-contrib-uglify**": "~0.2.2"

}

}

1. If you want to update the package version number that you are using then you can also enter the following to update your **package.json** **dependencies** section.

npm install --save grunt

1. If you want to update **devDependencies** then you would use the following command instead

npm install –save-dev grunt

1. You then need to define the Gruntfile.js which defines the tasks to be performed.

**Gruntfile.js**

module.exports = function(grunt) {

grunt.initConfig({

pkg: grunt.file.readJSON('package.json'),

uglify: {

dist: {

files: [{

expand: true,

cwd: 'js/',

src: '\*\*/\*.js',

dest: 'build/',

ext: '.min.js',

extDot : 'last'

}]

}

},

});

// Load the uglify plugin

grunt.loadNpmTasks('grunt-contrib-uglify');

// this would be run by typing "grunt test" on the command line

grunt.registerTask('test', ['uglify']);

}

1. Once created you can run the following command to install the plugins to the local directory.

This should create a **node\_modules** directory with the dependencies defined in the **package.json** file.

npm install

1. Otherwise the dependencies can be installed individually as follows

npm install –grunt-contrib-uglify

1. To run this you then just need to type the following

grunt test

# Using Less with Grunt

1. Create your **Gruntfile.js** and **package.json** files

The package,json file should look the following

{

"name": "project-name",

"version": "0.1.0",

"author": "Your Name",

"devDependencies": {

"grunt": "~0.4.1",

"grunt-contrib-less": "~0.11.4",

}

}

The Gruntfile.json should contain the following declaration

less: {

development: {

options: {

compress: false,

optimization: 2

},

files: {

"css/test.css" : "less/test.less"

}

}

},

1. As well as the following; the registerTask is used so that when you run “grunt l” on the command line it will run the less task as above.

grunt.loadNpmTasks('grunt-contrib-less');

grunt.registerTask("l", ["less"]);

# JSHint

Using this grunt task we can check out JavaScript against possible problems.

This list of warnings can be found here <https://github.com/jshint/jshint/blob/2.1.4/src/shared/messages.js>

## JQuery $ undefined

To correct this you need to add the following

/\* This will stop jshint from complaining that $ is not defined \*/

/\*global $:false \*/

## Ignore Line

To ignore you can add this comment to the end of the line

error: function (xhr, textStatus, errorThrown) { // jshint ignore:line

## Underscore variable and function names

To turn this off you need to add the following to the block of code that you want to ignore, or alternatively just add the –W030 declaration and don’t add the +W030

/\* This will turn off jshint warnings for underscore named functions \*/

/\* jshint -W030 \*/

Block of code not to check for this warning

/\* jshint +W030 \*/

## Array notation

To turn of the warning that you should use the [] notation instead of new Array() you can do the following.

/\* This will turn off jshint warnings for using the array literal notation \*/

/\* jshint -W009 \*/

Block of code not to check for this warning

/\* jshing +W009 \*/