



GULP

Quick guide to getting up
and running *today*

Robert Dunaway

Dedication

This book is dedicated to my soul mate and wife, Amy Dunaway. She has been a constant source of love, motivation, inspiration, and support.

TABLE OF CONTENTS

CONTENTS

Gulp Tutorials

[GULP Tutorial Part 1 – Reasons for Build Tools like Gulp](#)

[Productivity](#)

[GULP Tutorial Part 2 – Setup](#)

[GULP Tutorial Part 3 – Adding Plugins](#)

[GULP TUTORIAL PART 4 – Sequence and Parallel task processing](#)

[GULP TUTORIAL PART 5 – Handling errors with Plumber](#)

[GULP Tutorial Part 6 – Optimizing JavaScript/TypeScript](#)

[Annotation](#)

[Clean out ‘dist’](#)

[Copy all src files to ‘dist’](#)

[CONCATENATION](#)

[Compress and Minify JavaScript](#)

[GULP Tutorial Part 7 – Optimizing CSS](#)

[GULP Tutorial Part 8 – Optimizing HTML](#)

[GULP Tutorial Part 9 – Images](#)

[GULP Tutorial Part 10 – JSON, calling grunt from gulp](#)

[GULP Tutorial Part 11 – JSHINT](#)

[GULP Tutorial Part 12 – TypeScript](#)

[GULP Tutorial Part 13 – SASS](#)

[GULP Tutorial Part 14 – Watch](#)

[GULP Tutorial Part 15 – Useful Gulp Commands & Tips](#)

[GULP Tutorial Part 16 – Glob Tips](#)

[GULP TUTORIAL PART 22 – Useful NPM Packages/Commands](#)

[Commands Cheat Sheet](#)

[PowerShell \(primer\)](#)

[Syntax](#)

[Adding and removing files](#)

[Installing NodeJS and NPM Packages](#)

[NPM Version updates](#)

GULP

GULP TUTORIAL PART 1 – REASONS FOR BUILD TOOLS LIKE GULP

PRODUCTIVITY

The reason for a build system is always productivity. Otherwise we wouldn't invest time in it.

Build systems perform housecleaning work, allowing you to focus on code. Before build systems, if you were lucky, you could right click and select “minify” in your IDE. As lucky as this might have been, minification might not have been worth the additional development effort required. Build systems address this problem.

Build systems perform tasks with a level of precision humans are incapable of. For Continuous Integration and Continuous Delivery to work, a build system must be used to keep the human element out. Continuous Delivery requires automation at all levels, including testing, to mitigate common deployment defects.

There are thousands of plugins to perform just about any task imaginable. Here are a few.

Performance/Optimization

- Minification of JavaScript files
- Minification of CSS files
- Slimming down CSS classes that are not used
- Concatenating many JavaScript files to reduce get requests
- Creation of MAP files for debugging at run-time

Deployment

- Files can be optimized then copied to a folder to isolate deployment from development
- A zip file can be generated for deployment
- Automated tests can be executed
- Deployments can be created with a particular purpose; e.g., an app can be built for mobile.

Static analysis

- Linters can be executed against your code producing advice
- Cyclomatic complexity and other measures can be generated.

Documentation

- Documentation can be generated from code into readable formats.
- HTML documents can be generated from Markdown, a popular text format.

Additional resources

<https://www.youtube.com/watch?v=XJ5F-Auhato>

GULP TUTORIAL PART 2 – SETUP

Here we need to install NodeJS, pull our tutorial project from GitHub, create our NPM package configuration file, and install Gulp both globally and locally.

Installing NodeJS

Download and install NodeJS.

<https://nodejs.org/>

Get tutorial project from GitHub

Code for this tutorial can be found at

<https://github.com/MashupJS/gulp-tutorial>

On the GitHub repo page you'll see an option to "Download ZIP".

Download the ZIP file and extract it where you can work with it.

To see the end result, go to this repository.

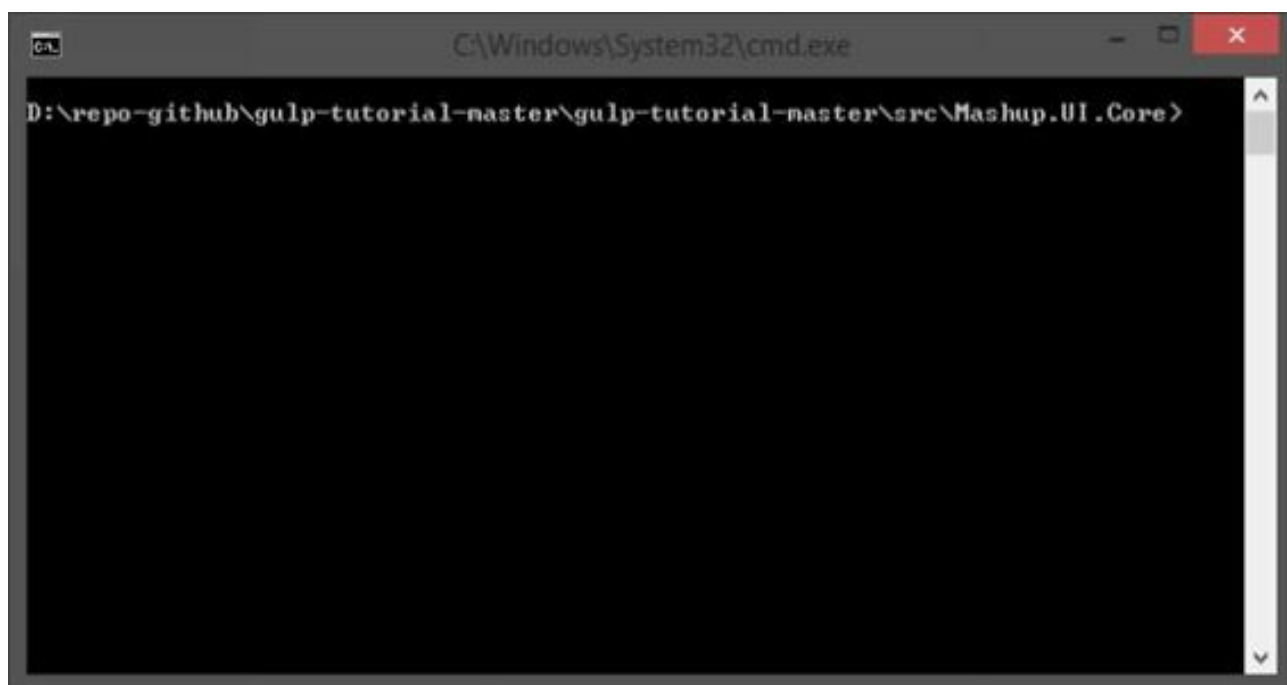
<https://github.com/MashupJS/gulp-tutorial-end-result>

Setup the NPM Project Configuration File

NPM packages are defined in the **package.json** file.

To create a package.json file, open a command prompt in the root of your client folder.

For this tutorial, open a command line to the Mashup.UI.Core folder.

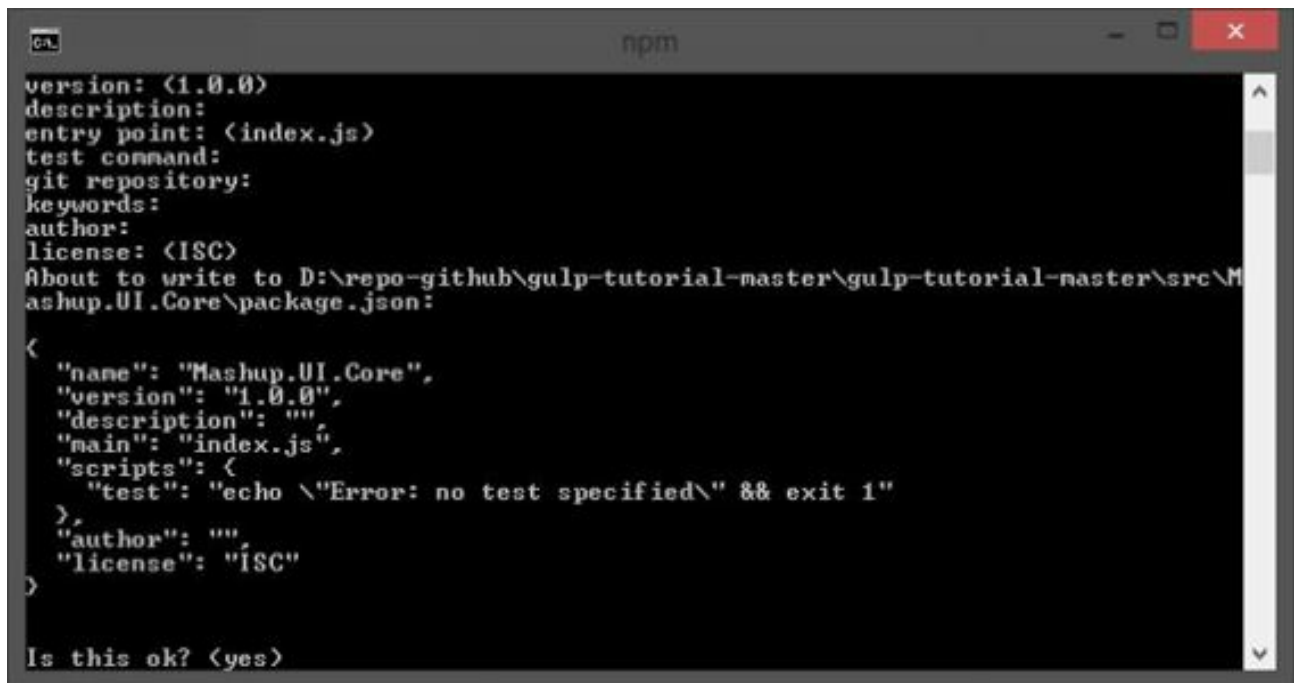
A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\System32\cmd.exe". The command prompt shows the current directory path: "D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>". The rest of the window is empty, indicating that no command has been entered yet.

At the command-line type

```
npm init
```

An empty package.json is created. Now we can begin installing NPM packages for use by Gulp.

You will be prompted for several configuration options. For the purposes of this tutorial I've skipped these and just pressed Enter for each prompt until complete.



```
npm
version: <1.0.0>
description:
entry point: <index.js>
test command:
git repository:
keywords:
author:
license: <ISC>
About to write to D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\M
ashup.UI.Core\package.json:
<
  "name": "Mashup.UI.Core",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
Is this ok? <yes>
```

And now your package.json is born.

Mashup.UI.Core.csproj	6/18/2015 5:00 PM	Visual C# Project F...	52 KB
Mashup.UI.Core.csproj.user	6/18/2015 5:00 PM	Visual Studio Proj...	2 KB
package.json	6/18/2015 5:12 PM	JSON File	1 KB
Web.config	6/18/2015 5:00 PM	XML Configuratio...	1 KB

For more information about NPM packages
<https://docs.npmjs.com/files/package.json>

Installing NPM packages

First, let's install the bower NPM module. We'll need this to pull client side scripts of which we have many.

From the command-line

Npm install bower -g

Retrieve all bower scripts from the command-line

Bower install

Installing Gulp

At this point, Gulp can be installed with the following command. Notice the “-g” command. This causes the NPM package to be installed globally.

Installing Gulp (add “-g” to install globally)

```
npm install -g gulp
```

Gulp must also be installed locally for your project. The global install allows you to execute Gulp commands from the command-line by providing a Command-Line Interface or CLI. The local Gulp install is a plug-in to NodeJS and gets access to Gulp plugins via NPM.

```
Npm install gulp --save-dev
```

If you're new to NPM, then just know that a screen that looks like this is completely normal.

```
C:\Windows\System32\cmd.exe
D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>npm
install -g gulp
C:\Users\rober_000\AppData\Roaming\npm\gulp -> C:\Users\rober_000\AppData\Roamin
g\npm\node_modules\gulp\bin\gulp.js
gulp@3.9.0 C:\Users\rober_000\AppData\Roaming\npm\node_modules\gulp
├── pretty-hrtime@1.0.0
├── interpret@0.6.2
├── deprecated@0.0.1
├── archy@1.0.0
├── minimist@1.1.1
├── v8flags@2.0.7 (user-home@1.1.1)
├── semver@4.3.6
├── tildify@1.1.0 (os-homedir@1.0.0)
├── chalk@1.0.0 (escape-string-regexp@1.0.3, ansi-styles@2.0.1, supports-color@1
.3.1, strip-ansi@2.0.1, has-ansi@1.0.3)
├── orchestra@0.3.7 (stream-consume@0.1.0, sequencify@0.0.7, end-of-stream@0.
1.5)
├── liftoff@2.1.0 (extend@2.0.1, rechoir@0.6.1, flagged-respawn@0.3.1, resolve@1
.1.6, findup-sync@0.2.1)
├── gulp-util@3.0.5 (array-differ@1.0.0, array-uniq@1.0.2, lodash._reinterpolate
@3.0.0, lodash._reevaluate@3.0.0, lodash._reescape@3.0.0, object-assign@2.1.1, b
eeper@1.1.0, replace-ext@0.0.1, vinyl@0.4.6, lodash.template@3.6.1, through2@0.6
.5, multipipe@0.1.2, dateformat@1.0.11)
├── vinyl-fs@0.3.13 (graceful-fs@3.0.8, strip-bom@1.0.0, vinyl@0.4.6, defaults@1
.0.2, mkdirp@0.5.1, through2@0.6.5, glob-stream@3.1.18, glob-watcher@0.0.6)
└── gulp-util@3.0.5 (array-differ@1.0.0, array-uniq@1.0.2, beeper@1.1.0, lodash
._reevaluate@3.0.0, lodash._reescape@3.0.0, lodash._reinterpolate@3.0.0, object-a
ssign@2.1.1, replace-ext@0.0.1, vinyl@0.4.6, lodash.template@3.6.1, through2@0.6
.5, multipipe@0.1.2, dateformat@1.0.11)

D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>npm
install gulp --save-dev
npm WARN package.json Mashup.UI.Core@1.0.0 No description
npm WARN package.json Mashup.UI.Core@1.0.0 No repository field.
npm WARN package.json Mashup.UI.Core@1.0.0 No README data
gulp@3.9.0 node_modules\gulp
├── pretty-hrtime@1.0.0
├── interpret@0.6.2
├── deprecated@0.0.1
├── archy@1.0.0
├── minimist@1.1.1
├── tildify@1.1.0 (os-homedir@1.0.0)
├── v8flags@2.0.7 (user-home@1.1.1)
├── semver@4.3.6
├── chalk@1.0.0 (escape-string-regexp@1.0.3, ansi-styles@2.0.1, supports-color@1
.3.1, strip-ansi@2.0.1, has-ansi@1.0.3)
├── orchestra@0.3.7 (stream-consume@0.1.0, sequencify@0.0.7, end-of-stream@0.
1.5)
├── liftoff@2.1.0 (extend@2.0.1, rechoir@0.6.1, flagged-respawn@0.3.1, resolve@1
.1.6, findup-sync@0.2.1)
├── vinyl-fs@0.3.13 (graceful-fs@3.0.8, strip-bom@1.0.0, defaults@1.0.2, vinyl@0
.4.6, mkdirp@0.5.1, through2@0.6.5, glob-stream@3.1.18, glob-watcher@0.0.6)
└── gulp-util@3.0.5 (array-differ@1.0.0, array-uniq@1.0.2, beeper@1.1.0, lodash
._reevaluate@3.0.0, lodash._reescape@3.0.0, lodash._reinterpolate@3.0.0, object-a
ssign@2.1.1, replace-ext@0.0.1, vinyl@0.4.6, lodash.template@3.6.1, through2@0.6
.5, multipipe@0.1.2, dateformat@1.0.11)

D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>
```

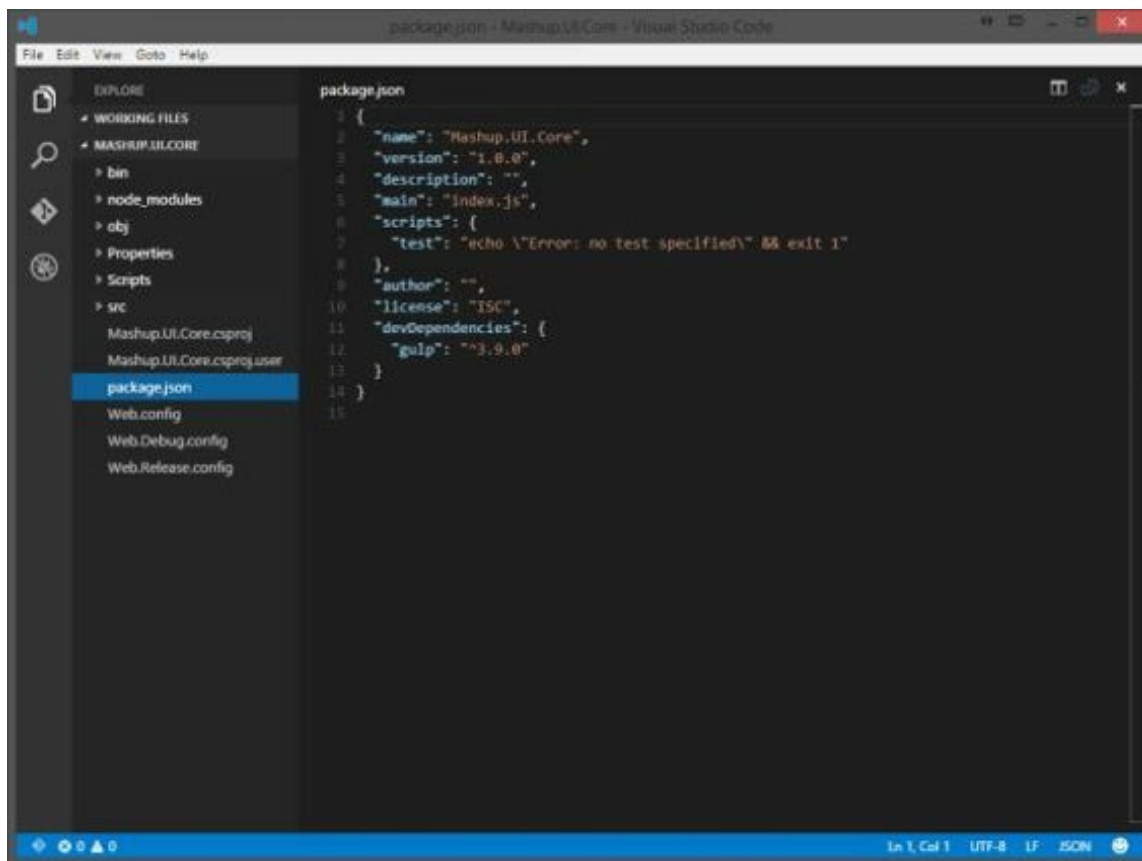
TIP: Visual Studio Code

Often I want to view a file without having to open an entire Integrated Development Environment like Visual Studio .NET. It's just way more than I need to quickly view a file and in fact, often you can't quickly view a file because VS .NET is so big.

Download and install Visual Studio Code. You can use programs like Notepad++ or Sublime as well.

In this case we just installed Gulp globally and locally. This tutorial is a learning tool so in the spirit of learning, each time I perform an action I'm going to poke around and see what changed. After installing Gulp I want to see what has changed.

Right click the package.json file and open it with your favorite code editor and see the change.



Package.json stores its information as JSON. Notice the first several attributes. These are the values we opted to provide, or, in my case, not provide.

When we applied the “ --save-dev ” options to the NPM install statement, the config was added to the “devDependencies” section.

The package.json file is a part of your code base and should be included/checked-in to source control. The “node_modules” folder created by NPM should not be checked in to source control.

When setting up the development environment on a new machine, simply open a command-line to the folder where package.json resides and type:

```
npm install
```

Or

```
npm update
```

NPM will then go download and install all the packages specified in the package.json.

For more information on installing NPM packages

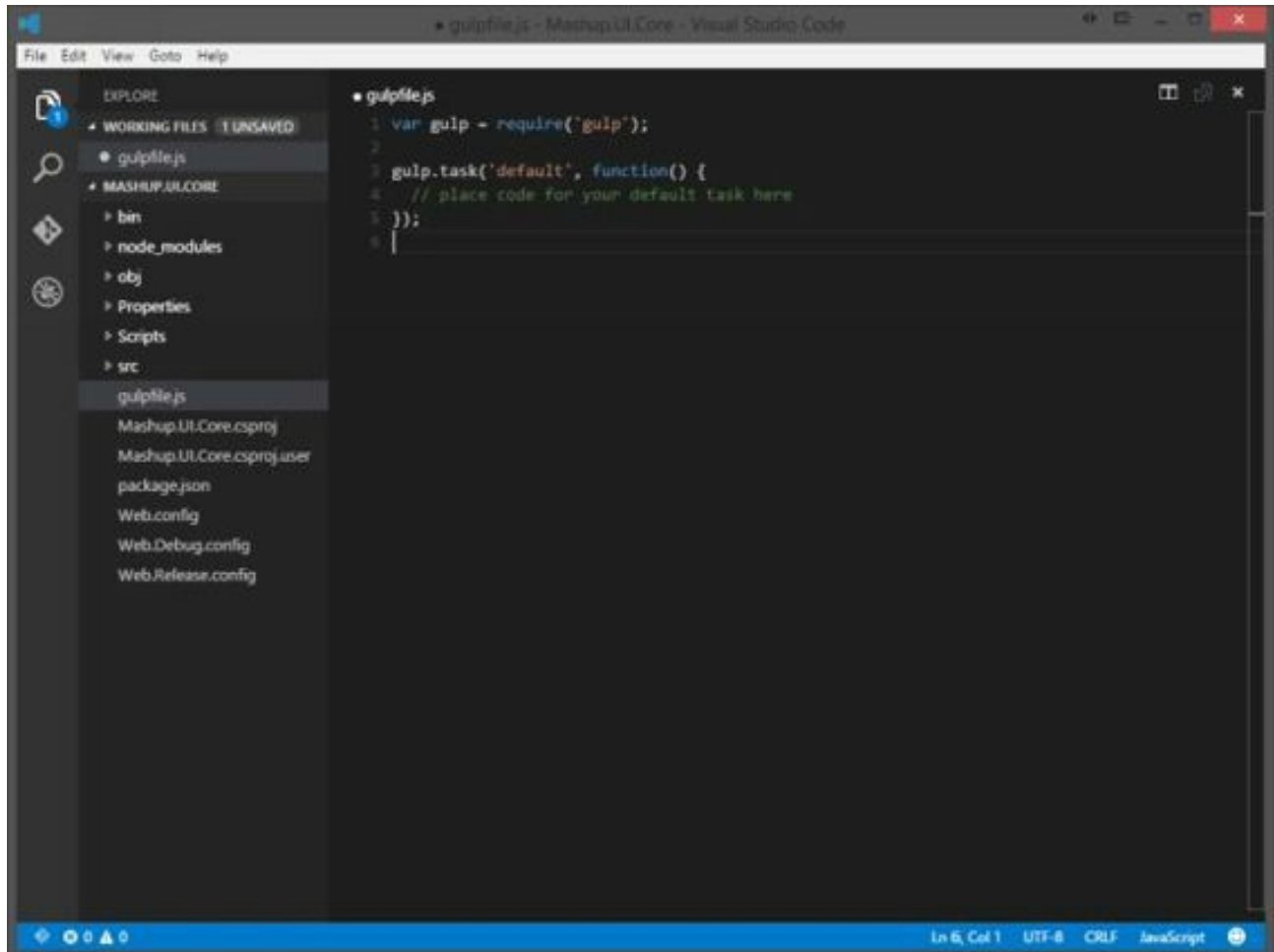
<https://docs.npmjs.com/getting-started/installing-npm-packages-locally>

Creating the gulpfile.json

Create a text file named **gulpfile.js** with the following content, in the root of your project. Add the following scaffolding to the new gulpfile.js file.

```
var gulp = require('gulp');  
gulp.task('default', function() {  
  // place code for your default task here
```

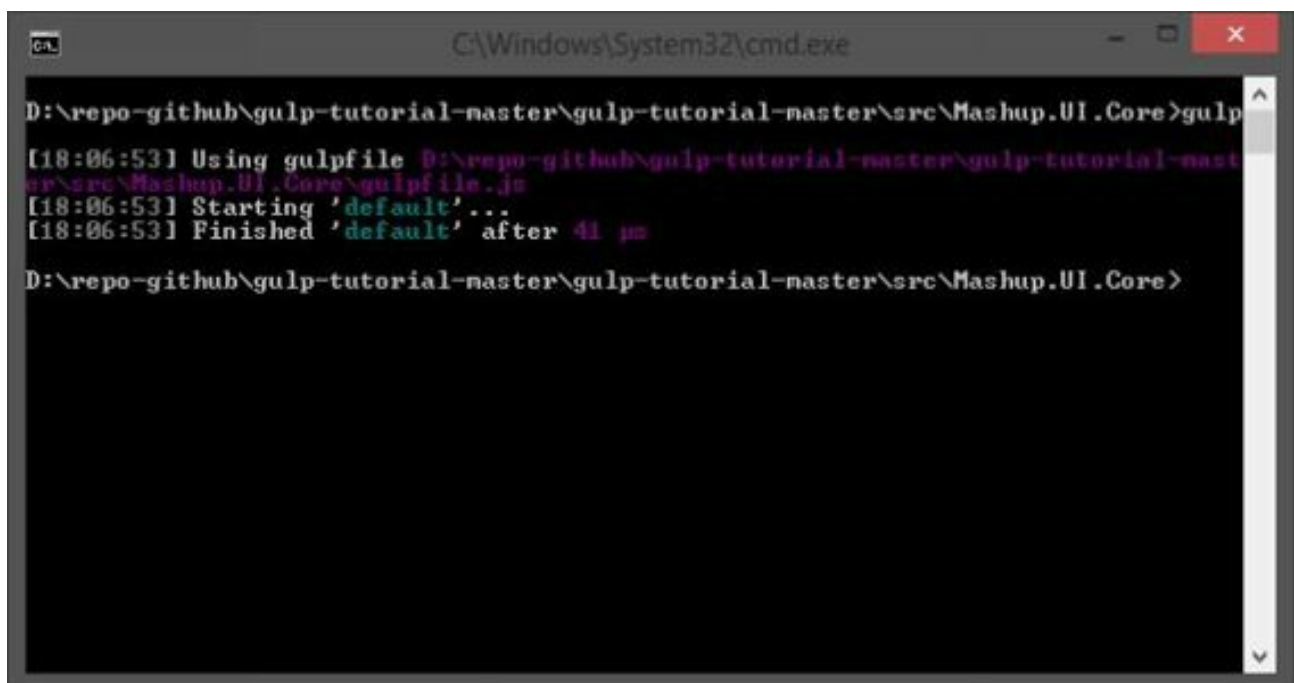
```
});
```



At this point you can execute Gulp from the command-line. There are no tasks in the “default” task, but Gulp will run.

At the command-line type “gulp” and press enter.

```
gulp
```



GULP TUTORIAL PART 3 – ADDING PLUGINS

Plugins provide function to the task runner.

Tip: When searching for plugins, consider the number of downloads the module has on NPM and activity on Github. These are indicators of how active the community is and how much support you can expect. You might find a dozen JavaScript minifiers. Choose the one with the most downloads and most recent activity on Github.

You can search for plugins here

<http://gulpjs.com/plugins>

Once you've found a plugin, navigate to the plugins page. Here you'll find general information on how to use the plugin and usually a couple examples to get you started.



Install a few useful plugins from the commandline of the root of your project. Notice “ --save-dev ”. This option includes the plugin in the package.json file.

You've already installed Gulp.

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

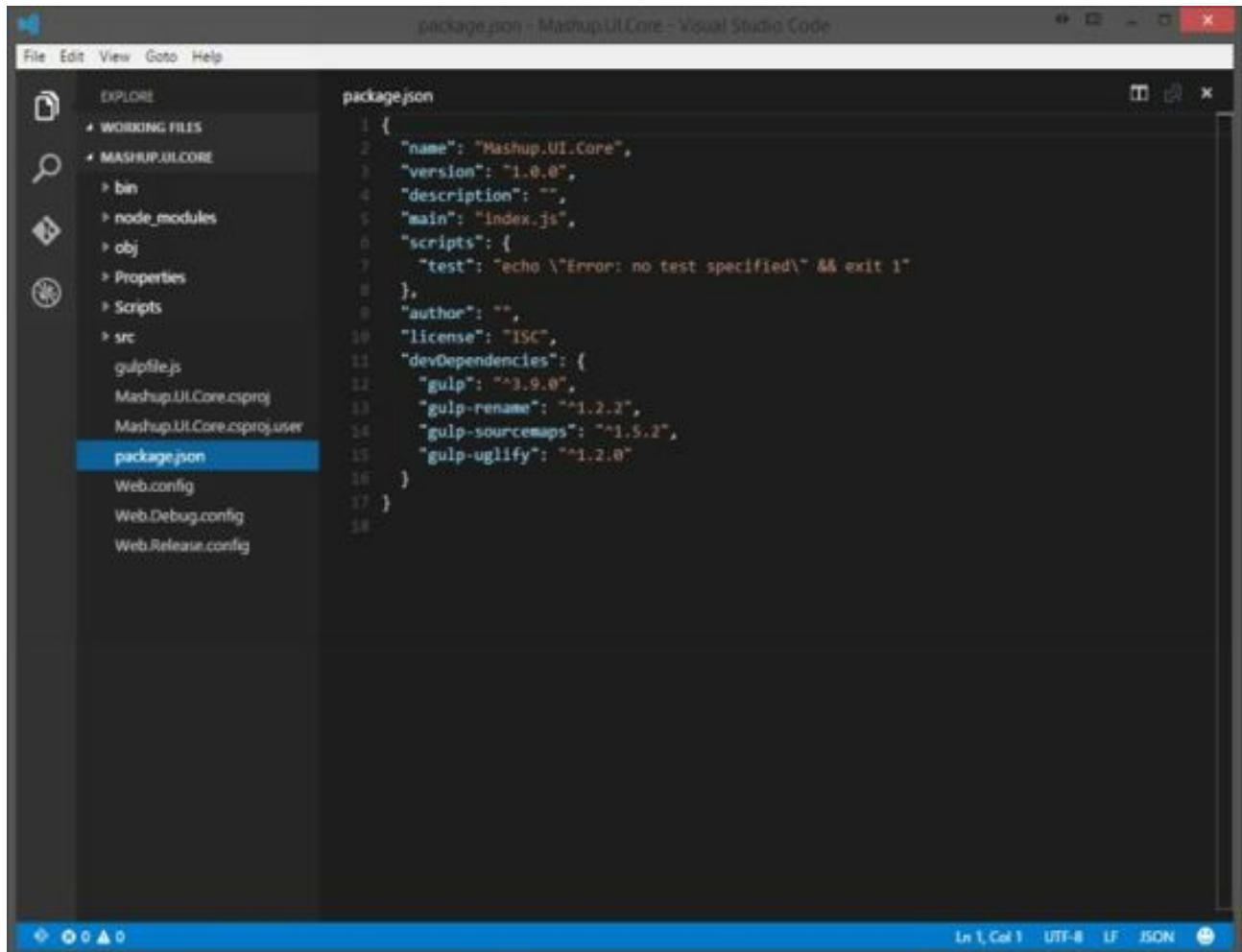
Go ahead and install a couple more. (*Just for fun*)

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

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

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

Viewing the package.json with VSC, you'll notice the new plugin configurations are saved.



```
1 {
2   "name": "Mashup.UI.Core",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "",
10  "license": "ISC",
11  "devDependencies": {
12    "gulp": "^3.9.0",
13    "gulp-rename": "^1.2.2",
14    "gulp-sourcemaps": "^1.5.2",
15    "gulp-uglify": "^1.2.0"
16  }
17 }
```

To add these plugins to your gulp implementation, add the new plugins to your gulpfile.js.

```
var gulp = require('gulp')
    , uglify      = require('gulp-uglify')
    , rename      = require('gulp-rename')
    , sourcemaps  = require('gulp-sourcemaps')
;
gulp.task('default', function() {
  // place code for your default task here
});
```

Syntax for creating a task

```
Gulp.task([task-name], function() {  
    Return gulp.src([glob-array]  
        .pipe([your-plugin])  
        .pipe([another-plugin])  
        .pipe(gulp.dest(dist));  
});
```

Notice the “function” keyword. One of the more significant differences between Gulp and Grunt is configuration versus code. Gulp subscribes to a “code” approach while Grunt subscribes to “configuration”.

GULP TUTORIAL PART 4 – SEQUENCE AND PARALLEL TASK PROCESSING

The ability to execute Gulp tasks in sequence and parallel is still a moving target. By default, Gulp leans toward executing all tasks in parallel because that is the more performant approach.

The option I chose was “run-sequence”. I chose this after battling with the other options. The Gulp 4 release should resolve many of the issues I struggled with.

We have not yet created tasks but in preparation, install this plug-in.

From the command-line install

```
npm install run-sequence --save-dev
```

Add the module to the gulp file

```
, runSequence = require('run-sequence')
```

All tasks created in this tutorial will have no other dependencies except the tasks we execute via the Gulp default task. The **runSequence** function will manage our dependencies. Creating dependencies is a simple option provided by Gulp but this causes tight coupling between tasks.

For instance, before optimizing files, we will clean out the distribution folder and then copy optimized files back to it. During development, however, cleaning the distribution folder breaks when all files are not copied and optimized back to it. An attempt to copy and optimize every file would cause a delay during development.

To resolve this, we will not tightly couple tasks, allowing all tasks to be executed when Gulp is run, but only the changes files are executed against when watching files during development.

Another example where sequence matters is how JavaScript is optimized. Some JavaScript is found in *.js files while other JavaScript is found after the compilation of TypeScript down to JavaScript. To minify and optimize JavaScript effectively, it's better to perform the JavaScript **Uglify** after the *.ts, TypeScript, files have been transpiled. So a dependency exists between TypeScript and JavaScript.

We can accomplish this with **runSequence()**.

Here is what a default task might look like when using the runSequence function to manage tasks.

(Don't add this code)

```
gulp.task('default', function() { runSequence('clean-dist',
```



```
        'annotate',
        'copy',
        ['coreservices', 'routeconfig', 'sass', 'tscompile', 'libs', 'grunt-merge-
json:menu',
        'tslint', 'jshint', 'minifyhtml', 'minifyimage'],
        ['uglifyalljs', 'minifycss'],
        'watch'
    );
});
```

Here is the sequence of execution

1 clean-dist

2 annotate

3 copy

4 (run in parallel) coreservices, routeconfig, sass, tscompile, libs, grunt-merge-
json:menu, tslint, jshint, minifyhtml, minifyimage

5 uglifyalljs, minifycss

6 watch

These are all tasks you will have created by the end of this multi-part tutorial.

Optimizing task performance

Notice the number of tasks executed in step 4. The more tasks you can run in parallel, the faster your process will be. It's important to optimize your process as much as possible so you can change a piece of code and immediately execute the optimized version without delay.

Other options

Gulp 4.0 will have new methods `series()` and `parallel()`. This will be the preferred approach once released.

Orchestrator – is an NPM module that supports series and parallel processing.

GULP TUTORIAL PART 5 – HANDLING ERRORS WITH PLUMBER

From the command-line install

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

Add the module to the Gulp file

```
, plumber = require('gulp-plumber');
```

Add this to the top of your script file. Our **plumber** will use this function for logging errors to the console.

```
var onError = function(err) {  
  console.log(err);  
};
```

Here is what a task might look like with the **plumber()** function.
(Don't add this code)

```
// -----  
// Watch specific tasks. This is to support the use of newer.  
// -----  
gulp.task('watch:annotate', function () {  
  return gulp.src(['src/index.controller.js', 'src/core/**/*.*js', 'src/apps/**/*.*js',  
    'src/core/lib/**/*.*', '!/**/*.*.min.js'], { base: 'src/.' })  
    .pipe(plumber({  
      errorHandler: onError  
    })))  
    .pipe(newer('src/.'))  
    .pipe(ngAnnotate())  
    .pipe(gulp.dest('src/.'));  
});
```

The following tutorials will implement this **plumber** function with each task we create.

Currently, your **gulpfile.js** should look like this:

• gulpfile.js

```
1
2 var onError = function(err) {
3   console.log(err);
4 };
5
6
7 var gulp = require('gulp')
8   , uglify      = require('gulp-uglify')
9   , rename      = require('gulp-rename')
10  , sourcemaps  = require('gulp-sourcemaps')
11  , runSequence = require('run-sequence')
12  , plumber     = require('gulp-plumber')
13 ;
14
15
16 gulp.task('default', function() {
17   // place code for your default task here
18 });
19
```

GULP TUTORIAL PART 6 – OPTIMIZING JAVASCRIPT/TYPESCRIPT

In this part of the tutorial, we are setting up our first Gulp task.

ANNOTATION

Before we concatenate and minify, let's make sure our Angular code is in good shape. To make a long story short... Angular is based, largely, on the ability to directly inject dependencies. This ability is made possible because the name of the injected dependency is interpreted. As soon as a file is minified, function names are changed to 'a' or 'b' or whatever is the next available small variable name. This breaks dependency injection.

Passing the dependency name as a string corrects this problem. Static string names are not minified. You can do this yourself or let Gulp run a task to do this for you. Even if you decided to handle this while writing code, it's a good idea to run an annotation task as a precaution.

From the command-line install

```
npm install gulp-ng-annotate --save-dev
```

Add the module to the gulp file

```
, ngAnnotate = require('gulp-ng-annotate')
```

Add task to Gulp file

```
gulp.task('annotate', function () {
  return gulp.src(['src/index.controller.js', 'src/core/**/*.js', 'src/apps/**/*.js',
    '!src/core/lib/**/*.min.js'], { base: 'src/.' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(ngAnnotate())
    .pipe(gulp.dest('src/.'));
});
```

The challenge presented by this task is updating the original files. In most cases, the original files are left alone. In this case, we can do the same, but since annotation is actually a correction to code, we will update the original code files. To make this work, we are using the `{ base: 'src/.' }` option.

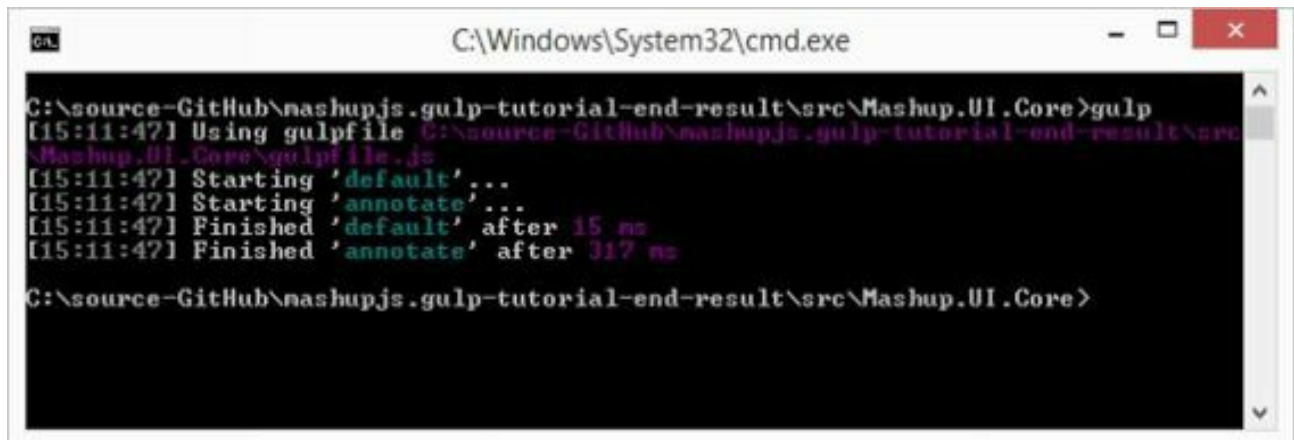
TIP: To keep your tasks running fast, eliminate unnecessary processing by telling the task to ignore your JavaScript libraries, i.e., `!src/core/lib/**/*.min.js`

Add the new task to the default task

```
gulp.task('default', function () {
  runSequence('annotate');
});
```

Run the default task

gulp



```
C:\Windows\System32\cmd.exe
C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[15:11:47] Using gulpfile C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[15:11:47] Starting 'default'...
[15:11:47] Starting 'annotate'...
[15:11:47] Finished 'default' after 15 ms
[15:11:47] Finished 'annotate' after 317 ms
C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

For more information

<http://christian.fei.ninja/DRY-dependency-injection-in-Angular-with-gulp-ng-annotate/>

CLEAN OUT 'DIST'

Now, let's clean out our 'dist' folder so we're starting afresh. Execute the following from PowerShell.

From the command-line install

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

Add the module to the Gulp file

```
, clean = require('gulp-clean')
```

Add the task to the Gulp file

```
gulp.task('clean-dist', function () {
  return gulp.src('dist', { read: false })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(clean());
});
```

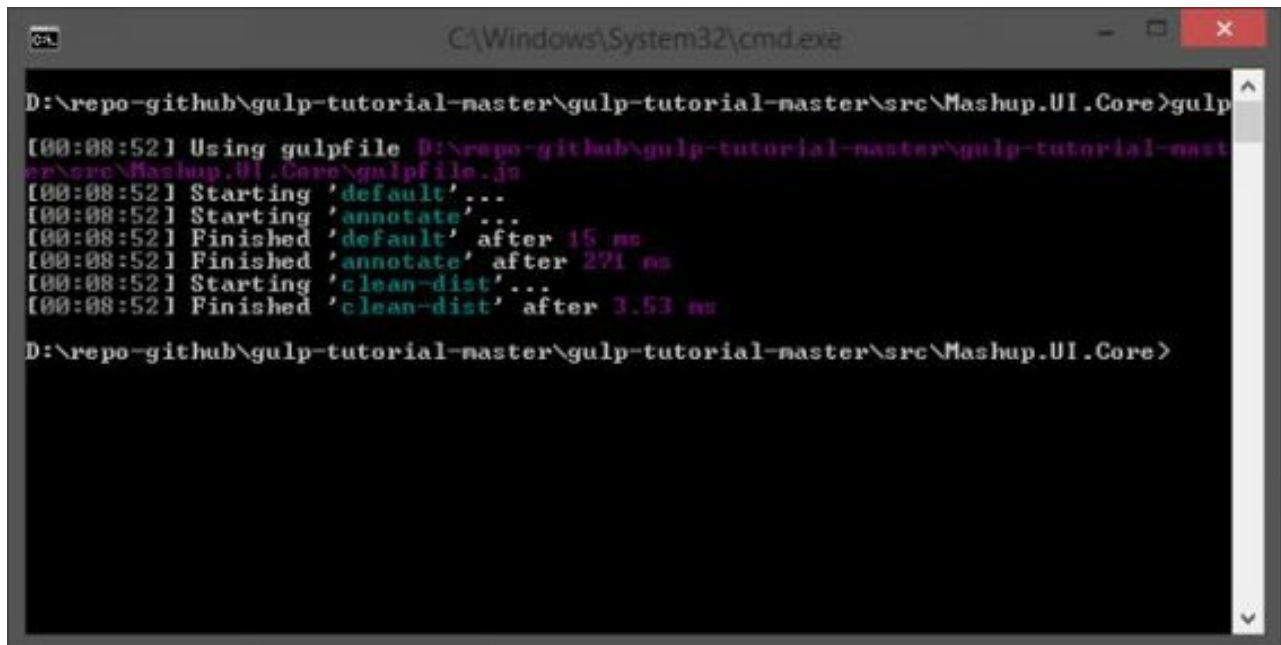
Add the new task to the default task

This task will run in sequence after the annotate task completes.

```
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist');
});
```

Run the default task

gulp



```
C:\Windows\System32\cmd.exe
D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>gulp
[00:08:52] Using gulpfile D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core\gulpfile.js
[00:08:52] Starting 'default'...
[00:08:52] Starting 'annotate'...
[00:08:52] Finished 'default' after 15 ms
[00:08:52] Finished 'annotate' after 271 ms
[00:08:52] Starting 'clean-dist'...
[00:08:52] Finished 'clean-dist' after 3.53 ms
D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>
```

For more information

<https://www.npmjs.com/package/gulp-clean>

COPY ALL SRC FILES TO 'DIST'

We will keep our source ('src') code separate from our distribution code so we don't pollute our development environment. When executing the application we'll set the **index.html** file of the 'dist' folder as the startup. This approach might seem to introduce challenges because when debugging you'll need the ability to read the compressed and minified versions of JavaScript and CSS. Gulp will give us that ability.

Now that we've cleaned out the 'dist' folder, in preparation for new files, let's go ahead and copy all our source code to 'dist'. Once we've copied the source code we can begin running tasks to optimize our code.

The newer module can be used by any task to ensure only the changed file is streamed to the task. This increases performance especially during development when only a single files is changed.

From the command-line install

```
Npm install gulp-newer --save-dev
```

Add the module to the Gulp file

```
, newer = require('gulp-newer')
```


Add the task to the Gulp file

```
gulp.task('copy', function () {
  return gulp.src('src/**/*')
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(newer('dist'))
    .pipe(gulp.dest('dist'));
});
```

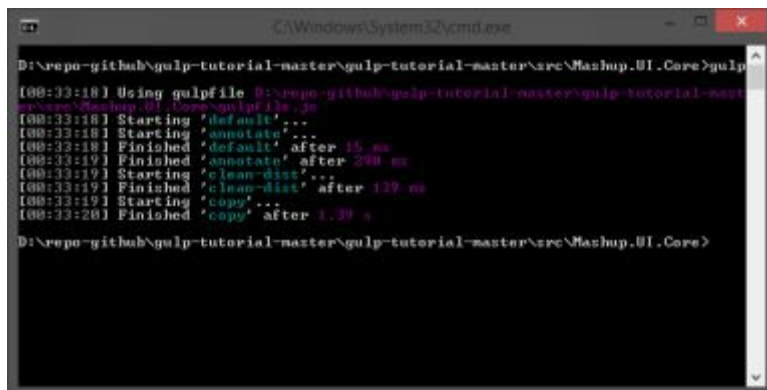
Add the new task to the default task

```
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy');
});
```

Run the default task

The Gulp command and the dist folder will be deleted and rebuilt.

Gulp



```
C:\Windows\System32\cmd.exe
D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.01.Core>gulp
[08:33:18] Using gulpfile D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.01.Core\gulpfile.js
[08:33:18] Starting 'default'...
[08:33:18] Starting 'annotate'...
[08:33:18] Finished 'annotate' after 15 ms
[08:33:19] Starting 'clean-dist'...
[08:33:19] Finished 'clean-dist' after 139 ms
[08:33:19] Starting 'copy'...
[08:33:20] Finished 'copy' after 1.39 s
D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.01.Core>
```

For more information

<https://www.npmjs.com/package/gulp-newer>

CONCATENATION

Before we minify our JavaScript files, let's see if there are any files we wish to combine. We could simply minify all JavaScript files, then concatenate them together, but then we would lose the ability to map minified code to source code for debugging purposes. We will concatenate any code we desire and then execute a general minification task.

From the command-line install

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

Add the module to the Gulp file

```
, concat = require('gulp-concat')
```

Add the task to the Gulp file

Create a task that combines all the **core/common** files into one.

```
gulp.task('coreservices', function () {  
  return gulp.src('src/core/common/**/*')  
    .pipe(plumber({  
      errorHandler: onError  
    })))  
  .pipe(concat('core.services.js'))  
  .pipe(gulp.dest('./dist/'));  
});
```

Create a task that combines all the **route.config.js** files together.

```
gulp.task('routeconfig', function () {
  return gulp.src(['src/core/config/route.config.js', 'src/apps/**/route.config.js'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(concat('route.config.js'))
    .pipe(gulp.dest('./dist/'));
});
```

NOTE: This capability to combine the route config is what makes the drop-in application style of MashupJS work.

Now copy all your **bower libraries** together. Typically, you'd combine as many of these files as possible. For now, we will work with third party libraries as separate files.

```
gulp.task('libs', function () {
  return gulp.src(['bower_components/**/bootstrap/dist/js/bootstrap.min.js'
    , 'bower_components/**/normalize.css/normalize.css'
    , 'bower_components/**/fontawesome/css/font-awesome.min.css'
    , 'bower_components/**/fontawesome/fonts/*.*'
    , 'bower_components/**/jquery/dist/jquery.min.js'
    , 'bower_components/**/angular/*.min.js'
    , 'bower_components/**/angular-route/angular-route.min.js'
    , 'bower_components/**/angular-sanitize/angular-sanitize.min.js'
    , 'bower_components/**/angular-bootstrap/ui-bootstrap-tpls.min.js'
    , 'bower_components/**/lodash/lodash.min.js'])
    .pipe(plumber({
      errorHandler: onError
    }))
    // .pipe(concat('libs.js'))
    .pipe(gulp.dest('dist/core/lib/bower/'));
});
```

Add the new tasks to the default task

```
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy',
    ['coreservices', 'routeconfig', 'libs']);
});
```

For additional documentation on gulp-concat:

<https://github.com/wearefractal/gulp-concat>

Run the default task

gulp

```
C:\Windows\System32\cmd.exe
D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>gulp
[00:47:59] Using gulpfile D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core\gulpfile.js
[00:47:59] Starting 'default' ...
[00:47:59] Starting 'annotate' ...
[00:47:59] Finished 'default' after 15 ms
[00:47:59] Finished 'annotate' after 263 ms
[00:47:59] Starting 'clean-dist' ...
[00:47:59] Finished 'clean-dist' after 138 ms
[00:47:59] Starting 'copy' ...
[00:48:00] Finished 'copy' after 1.37 s
[00:48:00] Starting 'coreservices' ...
[00:48:00] Starting 'routeconfig' ...
[00:48:00] Starting 'libs' ...
[00:48:00] Finished 'libs' after 2.73 ms
[00:48:00] Finished 'routeconfig' after 17 ms
[00:48:00] Finished 'coreservices' after 25 ms
D:\repo-github\gulp-tutorial-master\gulp-tutorial-master\src\Mashup.UI.Core>
```

For more information

<https://www.npmjs.com/package/gulp-concat>

COMPRESS AND MINIFY JAVASCRIPT

It's finally time to minify and compress JavaScript files. A normal solution would be to concatenate all *.js files into a single file name, "app.js" or "all.app.js" or something similar. The MashupJS is built to scale so large that a single concatenated *.js file might become too large and lazy loading will be desired.

Here we will minify and compress individual JavaScript files. Source maps will be created for troubleshooting and debugging. Source maps link between the original source code and the optimized code.

From the command-line install

```
npm install gulp-rename --save-dev
npm install gulp-uglify --save-dev
npm install gulp-sourcemaps --save-dev
```

Add the new modules to the Gulp file

```
, rename      = require('gulp-rename')
, uglify      = require('gulp-uglify')
, sourcemaps  = require('gulp-sourcemaps')
```

Add the task to the Gulp file

```
gulp.task('uglifyalljs', function () {
  //gulp.task('uglifyalljs', ['copy', 'coreservices', 'routeconfig', 'tscompile'], function () {
  return gulp.src(['dist/**/*.js', '!/**/*.min.js', '!dist/core/lib/**/*.', '!dist/core/common/**/*.'],
    { base: 'dist/.' })
    .pipe(plumber({
```

```

    errorHandler: onError
  )))
  .pipe(sourcemaps.init())
  // .pipe(newer('dist/.'))
  .pipe(uglify())
  .pipe(rename({
    extname: '.min.js'
  }))
  .pipe(sourcemaps.write('./'))
  .pipe(gulp.dest('dist/.'));
});

```

Add the new task to the default task

```

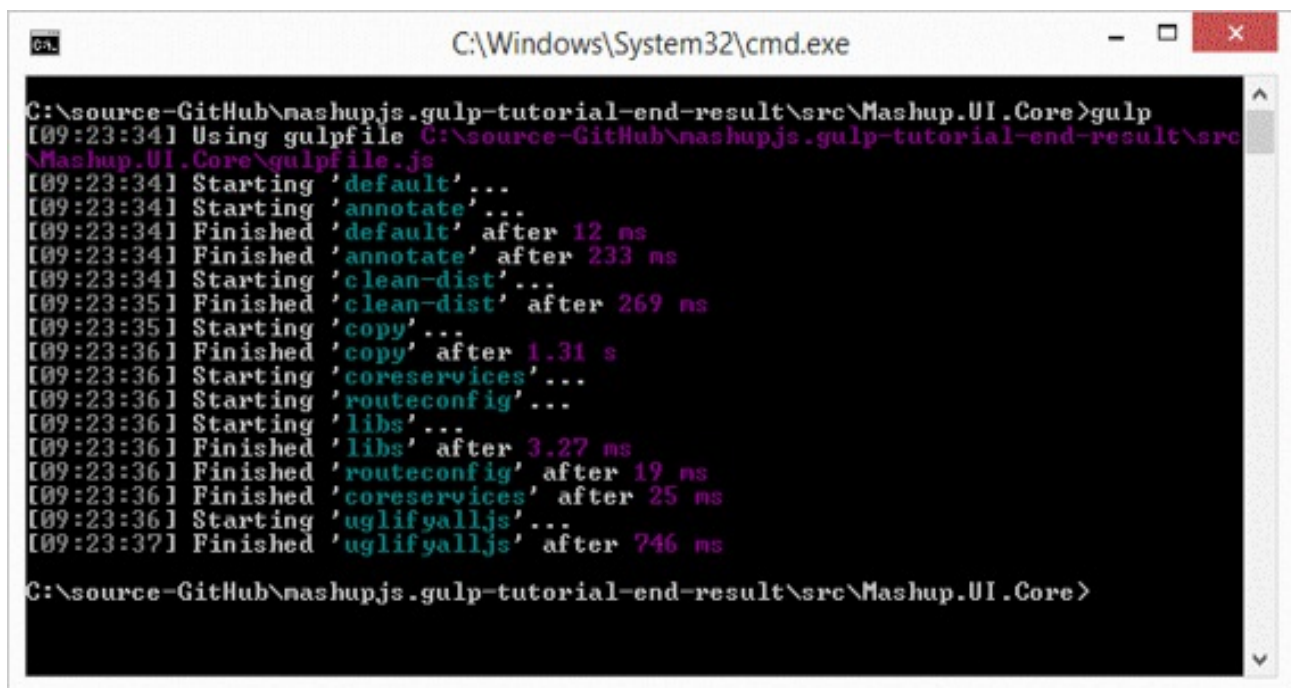
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy',
    ['coreservices', 'routeconfig', 'libs'],
    ['uglifyalljs']);
});

```

Notice the “uglifyalljs” task is placed in sequence to execute only after the previous array of tasks executes. This is to allow the “coreservices” and “routeconfig” JavaScripts to be created. Then they can be optimized by “uglifyalljs”.

Run the default task

gulp



For more information

<https://www.npmjs.com/package/gulp-rename>

<https://www.npmjs.com/package/gulp-uglify>

<https://www.npmjs.com/package/gulp-sourcemaps>

Current state of code

package.json

```
{
  "name": "Mashup.UI.Core",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "gulp": "^3.9.0",
    "gulp-clean": "^0.3.1",
    "gulp-concat": "^2.5.2",
    "gulp-newer": "^0.5.1",
    "gulp-ng-annotate": "^1.0.0",
    "gulp-plumber": "^1.0.1",
    "gulp-rename": "^1.2.2",
    "gulp-sourcemaps": "^1.5.2",
    "gulp-uglify": "^1.2.0",
    "run-sequence": "^1.1.1"
  }
}
```

gulpfile.js

```
var onError = function(err) {
  console.log(err);
};

var gulp = require('gulp')
  , uglify      = require('gulp-uglify')
  , rename     = require('gulp-rename')
  , sourcemaps = require('gulp-sourcemaps')
  , runSequence = require('run-sequence')
  , plumber    = require('gulp-plumber')
  , ngAnnotate = require('gulp-ng-annotate')
  , clean      = require('gulp-clean')
  , newer     = require('gulp-newer')
  , concat    = require('gulp-concat')
  , rename    = require('gulp-rename')
  , uglify    = require('gulp-uglify')
  , sourcemaps = require('gulp-sourcemaps')
;

gulp.task('annotate', function () {
  return gulp.src(['src/index.controller.js', 'src/core/**/*.*.js', 'src/apps/**/*.*.js', '!src/core/lib/**/*.*', '!src/**/*.*.min.js'], { base: 'src/.' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(ngAnnotate())
    .pipe(gulp.dest('src/.'));
});

gulp.task('clean-dist', function () {
```

```

    return gulp.src('dist', { read: false })
      .pipe(plumber({
        errorHandler: onError
      }))
      .pipe(clean());
  });

gulp.task('copy', function () {
  return gulp.src('src/**/*')
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(newer('dist'))
    .pipe(gulp.dest('dist'));
});

gulp.task('coreservices', function () {
  return gulp.src('src/core/common/**/*')
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(concat('core.services.js'))
    .pipe(gulp.dest('./dist/'));
});

gulp.task('routeconfig', function () {
  return gulp.src(['src/core/config/route.config.js', 'src/apps/**/route.config.js'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(concat('route.config.js'))
    .pipe(gulp.dest('./dist/'));
});

gulp.task('libs', function () {
  return gulp.src(['bower_components/**/*.*.js'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(concat('libs.js'))
    .pipe(gulp.dest('dist/core/lib/'));
});

gulp.task('uglifyalljs', function () {
  //gulp.task('uglifyalljs', ['copy', 'coreservices', 'routeconfig', 'tscompile'], function () {
  return gulp.src(['dist/**/*.*.js', '!/**/*.*.min.js', '!dist/core/lib/**/*.*', '!dist/core/common/**/*.*'], { base: 'dist/' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(sourcemaps.init())
    // .pipe(newer('dist/'))
    .pipe(uglify())
    .pipe(rename({
      extname: '.min.js'
    }))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest('dist/'));
  });

// -----
// Default Task
// -----
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy',

```

```
    ['coreservices', 'routeconfig', 'libs'],  
    ['uglifyalljs']);  
});
```


GULP TUTORIAL PART 7 – OPTIMIZING CSS

CSS gives us another opportunity for optimization. SASS is a higher-level language for CSS that can be transpiled down to CSS similar to how TypeScript is transpiled down to JavaScript. Later in this tutorial, we will add a task for transpiling SASS to CSS.

In this task, we will optimize all existing CSS files.

From the command-line install

```
npm install gulp-minify-css --save-dev
```

Add the module to the Gulp file

```
, minifycss = require('gulp-minify-css')
```

Add the task to the Gulp file

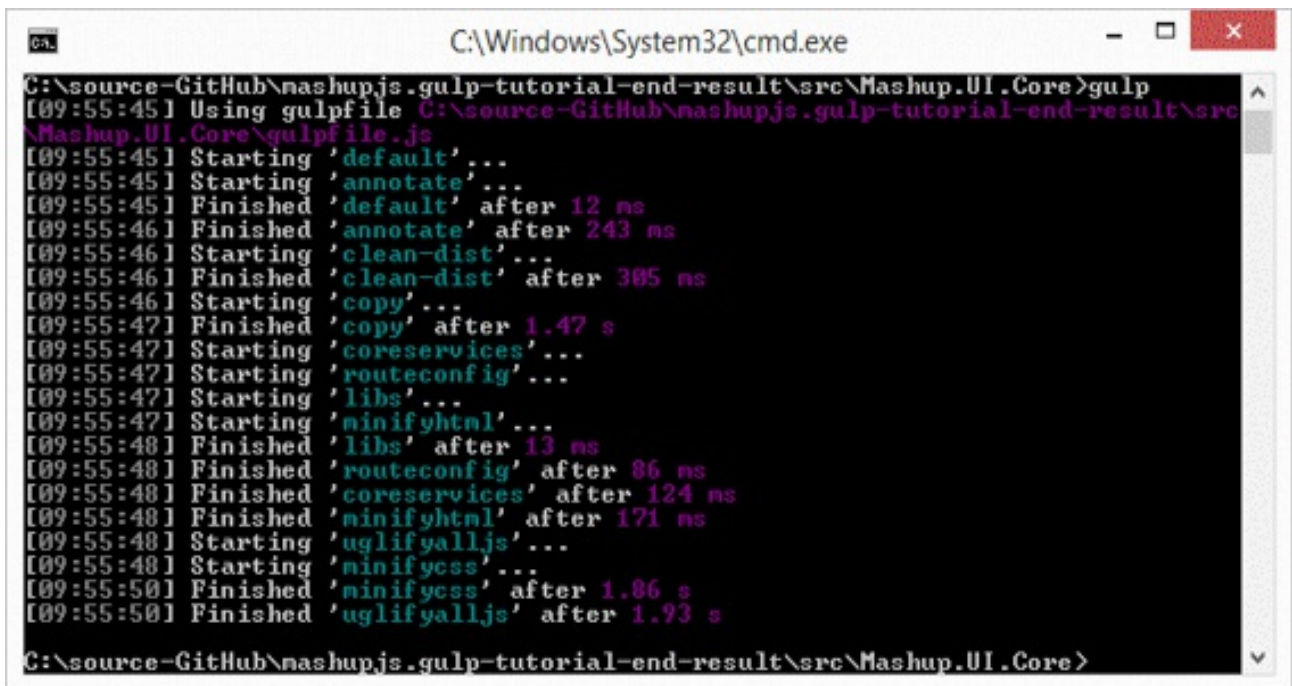
```
gulp.task('minifycss', ['copy'], function () {  
  return gulp.src(['dist/**/*.css', '!/**/*.min.css', '!dist/core/lib/**/*.css'], { base:  
    'dist/.' })  
    .pipe(sourcemaps.init())  
    .pipe(minifycss())  
    .pipe(rename({  
      extname: '.min.css'  
    })))  
    .pipe(sourcemaps.write('./'))  
    .pipe(gulp.dest('dist/.'));  
});
```

Add the new task to the default task

```
gulp.task('default', function () {  
  runSequence('annotate', 'clean-dist', 'copy',  
    ['coreservices', 'routeconfig', 'libs'],  
    ['uglifyalljs', 'minifycss']);  
});
```

Run the default task

gulp



```
C:\Windows\System32\cmd.exe  
C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp  
[09:55:45] Using gulpfile C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src  
Mashup.UI.Core\gulpfile.js  
[09:55:45] Starting 'default' ...  
[09:55:45] Starting 'annotate' ...  
[09:55:45] Finished 'default' after 12 ms  
[09:55:46] Finished 'annotate' after 243 ms  
[09:55:46] Starting 'clean-dist' ...  
[09:55:46] Finished 'clean-dist' after 385 ms  
[09:55:46] Starting 'copy' ...  
[09:55:47] Finished 'copy' after 1.47 s  
[09:55:47] Starting 'coreservices' ...  
[09:55:47] Starting 'routeconfig' ...  
[09:55:47] Starting 'libs' ...  
[09:55:47] Starting 'minifyhtml' ...  
[09:55:48] Finished 'libs' after 13 ms  
[09:55:48] Finished 'routeconfig' after 86 ms  
[09:55:48] Finished 'coreservices' after 124 ms  
[09:55:48] Finished 'minifyhtml' after 171 ms  
[09:55:48] Starting 'uglifyalljs' ...  
[09:55:48] Starting 'minifycss' ...  
[09:55:50] Finished 'minifycss' after 1.86 s  
[09:55:50] Finished 'uglifyalljs' after 1.93 s  
C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

Rather than concatenating CSS files, we are simply minifying them in place and creating maps.

Later, when transpiling from SASS, we won't need concatenation because the “@import” statement will pull multiple source files together for us.

For more information

<https://www.npmjs.com/package/gulp-minify-css>

GULP TUTORIAL PART 8 – OPTIMIZING HTML

HTML presents another opportunity for optimization. With every “bit” we can squeeze out of the code, the healthier is the application. HTML optimization is more noticeable as the HTML file becomes larger.

From the command-line install

```
npm install gulp-minify-html --save-dev
```

Add the module to the Gulp file

```
, minifyhtml = require('gulp-minify-html')
```

Add the task to the Gulp file

```
gulp.task('minifyhtml', function () {  
  return gulp.src(['dist/**/*.html', '!/**/*.min.html', '!dist/core/lib/**/*.html'], { base:  
    'dist/.' })  
    .pipe(plumber({  
      errorHandler: onError  
    }))  
    .pipe(sourcemaps.init())  
    .pipe(minifyhtml())  
    .pipe(rename({  
      extname: '.min.html'  
    }))  
    .pipe(sourcemaps.write('./'))  
    .pipe(gulp.dest('dist/.'));  
});
```

Small templates won't realize much improvement with HTML minification, but every little bit helps. Larger HTML files will benefit, but while we're at it, let's just minify all HTML files.

Add new task to the default task

```
gulp.task('default', function () {  
  runSequence('annotate', 'clean-dist', 'copy',  
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml'],  
    ['uglifyalljs', 'minifycss']);  
});
```

Run the default task

```
gulp
```

```
C:\Windows\System32\cmd.exe
C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[09:55:45] Using gulpfile C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src
\Mashup.UI.Core\gulpfile.js
[09:55:45] Starting 'default'...
[09:55:45] Starting 'annotate'...
[09:55:45] Finished 'default' after 12 ms
[09:55:46] Finished 'annotate' after 243 ms
[09:55:46] Starting 'clean-dist'...
[09:55:46] Finished 'clean-dist' after 305 ms
[09:55:46] Starting 'copy'...
[09:55:47] Finished 'copy' after 1.47 s
[09:55:47] Starting 'coreservices'...
[09:55:47] Starting 'routeconfig'...
[09:55:47] Starting 'libs'...
[09:55:47] Starting 'minifyhtml'...
[09:55:48] Finished 'libs' after 13 ms
[09:55:48] Finished 'routeconfig' after 86 ms
[09:55:48] Finished 'coreservices' after 124 ms
[09:55:48] Finished 'minifyhtml' after 171 ms
[09:55:48] Starting 'uglifyalljs'...
[09:55:48] Starting 'minifycss'...
[09:55:50] Finished 'minifycss' after 1.86 s
[09:55:50] Finished 'uglifyalljs' after 1.93 s
C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

For more information on gulp-minify-html.

<https://www.npmjs.com/package/gulp-minify-html>

GULP TUTORIAL PART 9 – IMAGES

Install the **gulp-imagemin** and **imagemin-pngquant** plugins. The `gulp-imagemin` comes with several optimizers for different image types, but does very little for PNG files. The `imagemin-pngquant` will focus on optimizing PNG files.

I've decided to leave the third party libraries alone, but you might choose to optimize them as well. This demo is focused on optimizing our own images.

From the command-line install

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

```
npm install imagemin-pngquant --save-dev
```

Add the modules to the Gulp file

```
, imagemin      = require('gulp-imagemin')  
, pngquant     = require('imagemin-pngquant')
```

Add the task to the Gulp file

```
gulp.task('minifyimage', function () {  
  return gulp.src(['dist/**/*.{png,jpg,gif,ico}', '!dist/core/lib/**/*.*',  
    '!dist/core/css/**/*.*'])  
    .pipe(plumber({  
      errorHandler: onError  
    })))  
    .pipe(imagemin({ progressive: true, optimizationLevel: 7, use: [pngquant()]  
    })))  
    .pipe(gulp.dest('dist/.'));  
});
```

Add the new task to the default task

```
gulp.task('default', function () {  
  runSequence('annotate', 'clean-dist', 'copy',  
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml', 'minifyimage'],  
    ['uglifyalljs', 'minifycss']);  
});
```

Run the default task

```
gulp
```

```
C:\Windows\System32\cmd.exe
C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[10:01:22] Using gulpfile C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[10:01:22] Starting 'default'...
[10:01:22] Starting 'annotate'...
[10:01:22] Finished 'default' after 16 ms
[10:01:22] Finished 'annotate' after 251 ms
[10:01:23] Starting 'clean-dist'...
[10:01:23] Finished 'clean-dist' after 329 ms
[10:01:23] Starting 'copy'...
[10:01:24] Finished 'copy' after 1.55 s
[10:01:24] Starting 'coreservices'...
[10:01:24] Starting 'routeconfig'...
[10:01:24] Starting 'libs'...
[10:01:24] Starting 'minifyhtml'...
[10:01:24] Starting 'minifyimage'...
[10:01:24] Finished 'libs' after 12 ms
[10:01:24] gulp-imagemin: Minified 0 images
[10:01:24] Finished 'minifyimage' after 134 ms
[10:01:24] Finished 'routeconfig' after 148 ms
[10:01:24] Finished 'coreservices' after 188 ms
[10:01:24] Finished 'minifyhtml' after 238 ms
[10:01:24] Starting 'uglifyalljs'...
[10:01:24] Starting 'minifycss'...
[10:01:26] Finished 'minifycss' after 1.89 s
[10:01:26] Finished 'uglifyalljs' after 1.95 s
C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

For more information

<https://www.npmjs.com/package/gulp-imagemin>

<https://www.npmjs.com/package/imagemin-pngquant>

GULP TUTORIAL PART 10 – JSON, CALLING GRUNT FROM GULP

On occasion you might need to combine JSON files. The MashupJS allows each app in the apps folder to define its own menu items. At build time we need these JSON files to be combined and saved with a specific file name so the menu.html template has access to all the menu items.

Edge case: Calling Grunt tasks from Gulp

So as it turns out, some of our favorite plugins used in Grunt are not available in Gulp. Before using Gulp I used Grunt and a plugin named “grunt-merge-json” to combine separate JSON files. Not only did it combine the JSON but merged them and eliminated duplication. I tried several Gulp plugins but nothing worked as expected. In time, a developer will build this plugin for Gulp but until then we can execute Grunt commands from our Gulp implementation.

From the command-line install

```
npm install -g grunt-cli
```

```
npm install grunt-merge-json --save-dev
```

Create a Gruntfile

Create a basic Gruntfile.js with the following content, in the root of your project.

```
module.exports = function (grunt) {  
  
  grunt.initConfig({  
    distFolder: 'dist',  
  
    pkg: grunt.file.readJSON('package.json'),  
    "merge-json": {  
  
      menu: {  
        src: ['src/apps/**/menu.json.txt'],  
        dest: '<%= distFolder %>/menu.json.txt',  
      },  
    },  
  });  
  
  // Load modules, register tasks  
  grunt.loadNpmTasks('grunt-merge-json');  
};
```

Install Gulp-Grunt from the command-line

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

For more information

<https://www.npmjs.com/package/gulp-grunt>

Add Grunt configuration to Gulp

Place this Grunt configuration code in the `gulpfile.js` just as you would with any other task.

```
// -----  
// Grunt configuration  
require('gulp-grunt')(gulp, {  
  // These are the default options but included here for readability.  
  base: null,  
  prefix: 'grunt-'  
  verbose: false  
});  
// -----
```

These are default configurations. The `base` represents the path to the `Gruntfile.js`. Because the `Gruntfile.js` is in the root folder `base` can be null.

We can now call the Grunt task the same way we would a Gulp task but with the prefix “`grunt=`”.

Test Grunt Plugin

Gulp `grunt-merge-json:menu`

Add the new Grunt task to the default task.

```
gulp.task('default', function () {  
  runSequence('annotate', 'clean-dist', 'copy',  
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml', 'minifyimage'  
      , 'grunt-merge-json:menu'],  
    ['uglifyalljs', 'minifycss']);  
});
```

Run the default task

gulp


```
C:\Windows\System32\cmd.exe
C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[10:35:18] Using gulpfile C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[10:35:18] Starting 'default'...
[10:35:18] Starting 'annotate'...
[10:35:18] Finished 'default' after 13 ms
[10:35:19] Finished 'annotate' after 250 ms
[10:35:19] Starting 'clean-dist'...
[10:35:19] Finished 'clean-dist' after 384 ms
[10:35:19] Starting 'copy'...
[10:35:21] Finished 'copy' after 1.38 s
[10:35:21] Starting 'coreservices'...
[10:35:21] Starting 'routeconfig'...
[10:35:21] Starting 'libs'...
[10:35:21] Starting 'minifyhtml'...
[10:35:21] Starting 'minifyimage'...
[10:35:21] Starting 'grunt-merge-json:menu'...
[10:35:21] Finished 'libs' after 22 ms
[10:35:21] gulp-imagemin: Minified 0 images
[10:35:21] Finished 'minifyimage' after 190 ms
[10:35:21] Finished 'routeconfig' after 215 ms
[10:35:21] Finished 'coreservices' after 255 ms
[10:35:21] Finished 'minifyhtml' after 311 ms
Running "merge-json:menu" (merge-json) task
File "dist/menu.json.txt" created.

Done, without errors.
[10:35:21] Finished 'grunt-merge-json:menu' after 754 ms
[10:35:21] Starting 'uglifyalljs'...
[10:35:21] Starting 'minifycss'...
[10:35:23] Finished 'minifycss' after 1.73 s
[10:35:23] Finished 'uglifyalljs' after 1.75 s

C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

Current state of code

package.json

```
{
  "name": "Mashup.UI.Core",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "grunt": "^0.4.5",
    "grunt-merge-json": "^0.9.5",
    "gulp": "^3.9.0",
    "gulp-clean": "^0.3.1",
    "gulp-concat": "^2.5.2",
    "gulp-grunt": "^0.5.2",
    "gulp-imagemin": "^2.2.1",
    "gulp-minify-css": "^1.1.6",
    "gulp-minify-html": "^1.0.3",
    "gulp-newer": "^0.5.1",
    "gulp-ng-annotate": "^1.0.0",
    "gulp-plumber": "^1.0.1",
    "gulp-rename": "^1.2.2",
    "gulp-sourcemaps": "^1.5.2",
    "gulp-uglify": "^1.2.0",
    "imagemin-pngquant": "^4.1.0",
    "run-sequence": "^1.1.1"
  }
}
```

gulpfile.js

```
var onError = function (err) {
  console.log(err);
};

var gulp = require('gulp')
  , uglify      = require('gulp-uglify')
  , rename      = require('gulp-rename')
  , sourcemaps  = require('gulp-sourcemaps')
  , runSequence = require('run-sequence')
  , plumber     = require('gulp-plumber')
  , ngAnnotate  = require('gulp-ng-annotate')
  , clean       = require('gulp-clean')
  , newer       = require('gulp-newer')
  , concat      = require('gulp-concat')
  , rename      = require('gulp-rename')
  , uglify      = require('gulp-uglify')
  , sourcemaps  = require('gulp-sourcemaps')
  , minifycss   = require('gulp-minify-css')
  , minifyhtml  = require('gulp-minify-html')
  , imagemin    = require('gulp-imagemin')
  , pngquant    = require('imagemin-pngquant')
;

gulp.task('annotate', function () {
  return gulp.src(['src/index.controller.js', 'src/core/**/*.js', 'src/apps/**/*.js',
    '!src/core/lib/**/*.js', '!src/**/*.min.js'], { base: 'src/.' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(ngAnnotate())
    .pipe(gulp.dest('src/.'));
});

gulp.task('clean-dist', function () {
  return gulp.src('dist', { read: false })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(clean());
});

gulp.task('copy', function () {
  return gulp.src('src/**/*')
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(newer('dist'))
    .pipe(gulp.dest('dist'));
});

gulp.task('coreservices', function () {
  return gulp.src('src/core/common/**/*')
    .pipe(plumber({
      errorHandler: onError
    }

```

```
    )))
    .pipe(concat('core.services.js'))
    .pipe(gulp.dest('./dist/'));
});
```

```
gulp.task('routeconfig', function () {
  return gulp.src(['src/core/config/route.config.js', 'src/apps/**/route.config.js'])
    .pipe(plumber({
      errorHandler: onError
    })))
    .pipe(concat('route.config.js'))
    .pipe(gulp.dest('./dist/'));
});
```

```
gulp.task('libs', function () {
  return gulp.src(['bower_components/**/*.*.js'])
    .pipe(plumber({
      errorHandler: onError
    })))
    .pipe(concat('libs.js'))
    .pipe(gulp.dest('dist/core/lib/'));
});
```

```
gulp.task('uglifyalljs', function () {
  //gulp.task('uglifyalljs', ['copy', 'coreservices', 'routeconfig', 'tscompile'], function () {
  return gulp.src(['dist/**/*.*.js', '!/**/*.*.min.js', '!dist/core/lib/**/*.*', '!dist/core/common/**/*.*'],
    { base: 'dist/.' })
    .pipe(plumber({
      errorHandler: onError
    })))
    .pipe(sourcemaps.init())
    // .pipe(newer('dist/.'))
    .pipe(uglify())
    .pipe(rename({
      extname: '.min.js'
    })))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest('dist/.'));
});
```

```
gulp.task('minifycss', function () {
  return gulp.src(['dist/**/*.*.css', '!dist/**/*.*.min.css', '!dist/core/lib/**/*.*'], { base: 'dist/.' })
    .pipe(plumber({
      errorHandler: onError
    })))
    .pipe(sourcemaps.init())
    .pipe(minifycss())
    .pipe(rename({
      extname: '.min.css'
    })))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest('dist/.'));
});
```

```
gulp.task('minifyhtml', function () {
  return gulp.src(['dist/**/*.*.html', '!/**/*.*.min.html', '!dist/core/lib/**/*.*'], { base: 'dist/.' })
    .pipe(plumber({
      errorHandler: onError
    })))
```

```

    )))
    .pipe(sourcemaps.init())
    .pipe(minifyhtml())
    .pipe(rename({
      extname: '.min.html'
    }))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest('dist/./'));
  });

gulp.task('minifyimage', function () {
  return gulp.src(['dist/**/*.{png,jpg,gif,ico}', '!dist/core/lib/**/*.*', '!dist/core/css/**/*.*'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(imagemin({ progressive: true, optimizationLevel: 7, use: [pngquant()] }))
    .pipe(gulp.dest('dist/./'));
});

// -----
// Grunt configuration
require('gulp-grunt')(gulp, {
  // These are the default options but included here for readability.
  base: null,
  prefix: 'grunt-',
  verbose: false
});
// -----

// -----
// Default Task
// -----
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy',
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml', 'minifyimage',
    , 'grunt-merge-json:menu'],
    ['uglifyalljs', 'minifycss']);
});

```

Gruntfile.js

```

module.exports = function (grunt) {

  grunt.initConfig({
    distFolder: 'dist',

    pkg: grunt.file.readJSON('package.json'),
    "merge-json": {

      menu: {
        src: ['src/apps/**/menu.json.txt'],
        dest: '<%= distFolder %>/menu.json.txt',
      },
    },
  });
};

```

```
// Load modules, register tasks
grunt.loadNpmTasks('grunt-merge-json');

};
```

GULP TUTORIAL PART 11 – JSHINT

JSHint is a static analysis tool used to analyze JavaScript code for quality and standards/style enforcement. The example below demonstrates how to analyze your code but it can also be piped into transpilation of code from TypeScript to JavaScript.

Install the **gulp-jshint**, **jshint-stylish**, and **gulp-jshint-html-reporter**.

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

```
npm install jshint-stylish --save-dev
```

```
npm install gulp-jshint-html-reporter --save-dev
```

Add the new plugins to your Gulp required list

```
, jshint          = require('gulp-jshint')
, stylish        = require('jshint-stylish')
, jshintfileoutput = require('gulp-jshint-html-reporter')
```

Add the new task to your gulpfile.js

```
gulp.task('jshint', function () {
  //gulp.task('jshint', ['copy', 'tscompile'], function () {
  return gulp.src(['./dist/**/*.js', '!dist/core/lib/**/*.*', '!**/*.min.js', '!dist/core/css/**/*.*.'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(jshint('.jshintrc'))
    .pipe(jshint.reporter(stylish))
    .pipe(jshint.reporter('gulp-jshint-html-reporter', { filename: 'jshint-output.html' }));
});
```

Creating a .jshintrc file

You can customize your own jshint rules in a file called .jshintrc. I'm starting with options recommended by John Papa.

<https://github.com/johnpapa/angular-styleguide#use-an-options-file>

.jshintrc TIP

You cannot easily create files beginning with a dot in Windows.

Steps to create the .jshintrc file:

- Create a file named **“.jshintrc.”** in the root alongside gulpfile.js and Gruntfile.js. Notice the ending dot. You must do this from Explorer.
- Windows will prompt you to confirm the extension.
- Finally Windows simply removes the ending dot for you. Now you

- have the file “.jshintrc”.
- If you are using Visual Studio .NET or some other IDE, you can now go and add the existing file to your project.

.jshintrc content

We are going to use the options promoted by John Papa.

Copy and paste this into your new .jshintrc file or visit John Papa’s page with the link above to see if any new updates have become available.

```
{
  "bitwise": true,
  "camelcase": true,
  "curly": true,
  "eqeqeq": true,
  "es3": false,
  "forin": true,
  "freeze": true,
  "immed": true,
  "indent": 4,
  "latedef": "nofunc",
  "newcap": true,
  "noarg": true,
  "noempty": true,
  "nonbsp": true,
  "nonew": true,
  "plusplus": false,
  "quotmark": "single",
  "undef": true,
  "unused": false,
  "strict": false,
  "maxparams": 10,
  "maxdepth": 5,
  "maxstatements": 40,
  "maxcomplexity": 8,
  "maxlen": 120,

  "asi": false,
  "boss": false,
  "debug": false,
  "eqnull": true,
  "esnext": false,
  "evil": false,
  "expr": false,
  "funcscope": false,
  "globalstrict": false,
  "iterator": false,
  "lastsemic": false,
  "laxbreak": false,
  "laxcomma": false,
  "loopfunc": true,
  "maxerr": false,
  "moz": false,
  "multistr": false,
  "notypeof": false,
  "proto": false,
```

```

"scripturl": false,
"shadow": false,
"sub": true,
"supernew": false,
"validthis": false,
"noyield": false,

"browser": true,
"node": true,

"globals": {
  "angular": false,
  "$": false
}
}

```

Using JSHint reporters

Reporters receive feedback from JSHint and format it into something human readable. By default, you'll get raw text from JSHint and it will be displayed at the command line. Adding the 'jshint-stylish' plugin and passing it to the `jshint.reporter(stylish)` gives you a more readable output to the command line.

```

Windows PowerShell
PS D:\Subversion-GitHub\mashupjs\src\Mashup.UI.Core> gulp jshint
[23:33:50] Using gulpfile D:\Subversion-GitHub\mashupjs\src\Mashup.UI.Core\gulpfile.js
[23:33:50] Starting 'annotate'...
[23:33:51] Finished 'annotate' after 483 ms
[23:33:51] Starting 'clean'...
[23:33:51] Finished 'clean' after 483 ms
[23:33:51] Starting 'copy'...
[23:33:51] Finished 'copy' after 483 ms
[23:33:55] Starting 'jshint'...

D:\Subversion-GitHub\mashupjs\src\Mashup.UI.Core\dist\core\config\route.config.js
  line 1 col 1 '...' is defined but never used.

  # 1 warning

D:\Subversion-GitHub\mashupjs\src\Mashup.UI.Core\dist\apps\mashup\~appConfig\route.config.js
  line 6 col 5 'routing' was defined but never used
  line 115 col 21 'userId' is defined but never used.
  line 117 col 21 'authDateTme' is defined but never used.

  # 1 error
  # 2 warnings

D:\Subversion-GitHub\mashupjs\src\Mashup.UI.Core\dist\core\common\services\base64.service.js
  line 25 col 42 Unexpected use of '++'.
  line 26 col 42 Unexpected use of '++'.
  line 27 col 42 Unexpected use of '++'.
  line 70 col 53 Unexpected use of '++'.
  line 71 col 53 Unexpected use of '++'.
  line 72 col 53 Unexpected use of '++'.
  line 73 col 53 Unexpected use of '++'.

  # 7 warnings

```

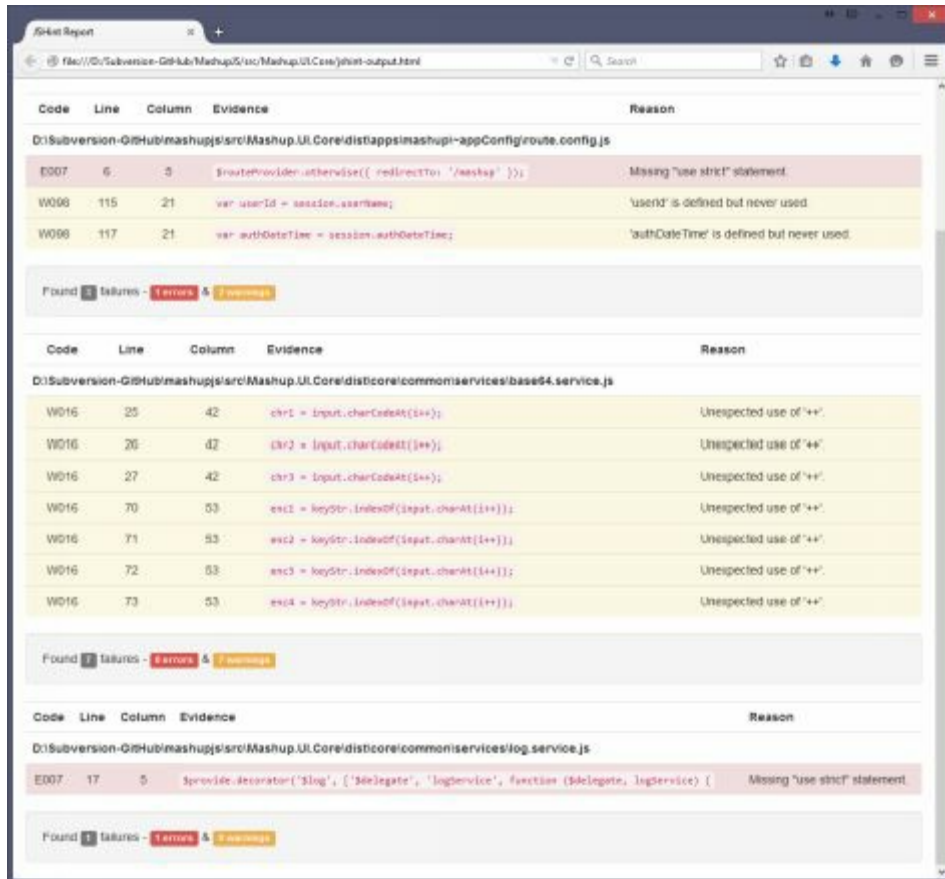
Taking this concept a step further, the 'gulp-jshint-html-reporter' generates an html document, as the name of the plugin implies.

For more information

<https://www.npmjs.com/package/gulp-jshint-html-reporter>

<https://github.com/ivan-vesely/gulp-jshint-html-reporter>

There isn't much in the way of documentation but the source code is available and you have access to the author here.



The screenshot displays a JSHint Report HTML output in a browser window. The report is organized into three sections, each corresponding to a different JavaScript file. Each section contains a table with columns for Code, Line, Column, Evidence, and Reason. The first section, for route.config.js, lists three warnings (W008) related to unused variables. The second section, for base64.service.js, lists seven warnings (W016) for the use of the ++ operator. The third section, for log.service.js, lists one error (E007) for a missing 'use strict' statement. Summary statistics at the bottom of each section indicate 0 errors and 3 warnings for the first two files, and 1 error and 0 warnings for the third.

Code	Line	Column	Evidence	Reason
D:\Subversion-GitHub\mashup\src\Mashup\UI\Core\dist\apps\mashup-appConfig\route.config.js				
E007	6	5	routerProvider.otherwise({ redirectTo: '/maskup' });	Missing "use strict" statement.
W008	115	21	var userId = session.userName;	'userId' is defined but never used.
W008	117	21	var authDateTime = session.authDateTime;	'authDateTime' is defined but never used.
Found 0 errors - 3 warnings				
Code	Line	Column	Evidence	Reason
D:\Subversion-GitHub\mashup\src\Mashup\UI\Core\dist\core\common\services\base64.service.js				
W016	25	42	chr1 = input.charCodeAt(i++);	Unspecified use of '++'.
W016	26	47	chr2 = input.charCodeAt(i++);	Unspecified use of '++'.
W016	27	42	chr3 = input.charCodeAt(i++);	Unspecified use of '++'.
W016	70	53	enc1 = keyStr.indexOf(input.charAt(i++));	Unspecified use of '++'.
W016	71	53	enc2 = keyStr.indexOf(input.charAt(i++));	Unspecified use of '++'.
W016	72	53	enc3 = keyStr.indexOf(input.charAt(i++));	Unspecified use of '++'.
W016	73	53	enc4 = keyStr.indexOf(input.charAt(i++));	Unspecified use of '++'.
Found 0 errors - 3 warnings				
Code	Line	Column	Evidence	Reason
D:\Subversion-GitHub\mashup\src\Mashup\UI\Core\dist\core\common\services\log.service.js				
E007	17	5	@provide.decorator(['\$log'], ['\$delegate', 'logService'], function (\$delegate, logService) {	Missing "use strict" statement.
Found 1 error - 0 warnings				

Options list

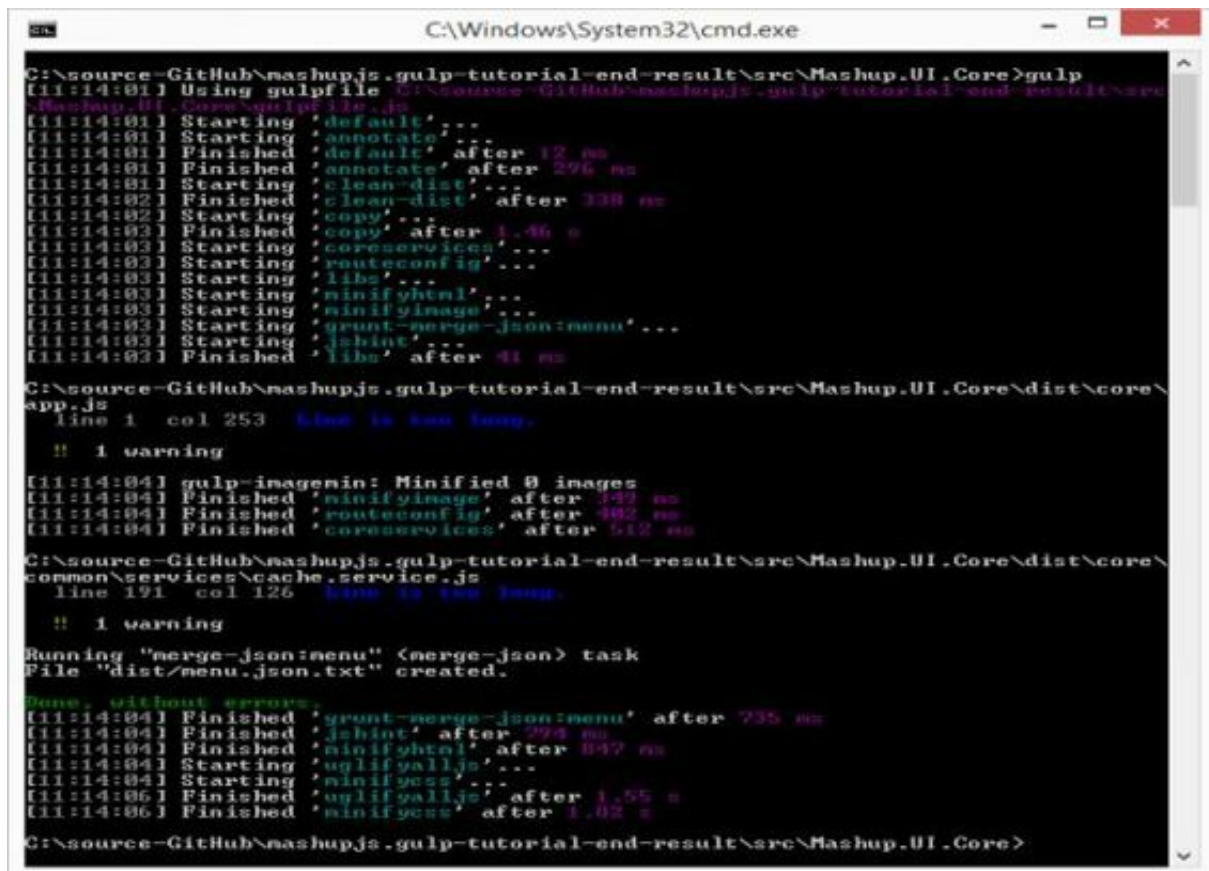
<http://jshint.com/docs/options/>

For more information

<https://www.npmjs.com/package/gulp-jshint>

Run the default task

gulp



```
C:\Windows\System32\cmd.exe
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[11:14:01] Using gulpfile C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[11:14:01] Starting 'default'...
[11:14:01] Starting 'annotate'...
[11:14:01] Finished 'default' after 12 ms
[11:14:01] Finished 'annotate' after 296 ms
[11:14:01] Starting 'clean-dist'...
[11:14:02] Finished 'clean-dist' after 338 ms
[11:14:02] Starting 'copy'...
[11:14:03] Finished 'copy' after 1.46 s
[11:14:03] Starting 'core-services'...
[11:14:03] Starting 'routeconfig'...
[11:14:03] Starting 'libs'...
[11:14:03] Starting 'minifyhtml'...
[11:14:03] Starting 'minifyimage'...
[11:14:03] Starting 'grunt-merge-json:menu'...
[11:14:03] Starting 'jshint'...
[11:14:03] Finished 'libs' after 41 ms

C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\app.js
  line 1 col 253  loop is too long.

  !! 1 warning

[11:14:04] gulp-imagemin: Minified 0 images
[11:14:04] Finished 'minifyimage' after 377 ms
[11:14:04] Finished 'routeconfig' after 482 ms
[11:14:04] Finished 'core-services' after 512 ms

C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\common\services\cache.service.js
  line 191 col 126  loop is too long.

  !! 1 warning

Running "merge-json:menu" <merge-json> task
File "dist/menu.json.txt" created.

Done, without errors.
[11:14:04] Finished 'grunt-merge-json:menu' after 735 ms
[11:14:04] Finished 'jshint' after 794 ms
[11:14:04] Finished 'minifyhtml' after 847 ms
[11:14:04] Starting 'uglifyalljs'...
[11:14:04] Starting 'minifycss'...
[11:14:04] Finished 'uglifyalljs' after 1.55 s
[11:14:06] Finished 'minifycss' after 1.82 s

C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

Tip

The jshint-output.html file is a good indicator of how your recent changes might have deviated from your standard or best practices. By checking this file into source control you can see if the file changes. If it does not, then no new errors are being reported. If it does change, then either old errors have been corrected or new errors have been detected.

GULP TUTORIAL PART 12 – TYPESCRIPT

TypeScript is a language used to build large-scale JavaScript applications. TypeScript code is transpiled down to ES5 JavaScript. Angular 2.0 is written in TypeScript and ES2015/ES2016, formerly ES6/ES7, features are available in TypeScript.

Links for installing TypeScript, for your version of Visual Studio, are here <http://www.typescriptlang.org/>

Dan Wahlin's take on TypeScript in a Gulp workflow

<http://weblogs.asp.net/dwahlin/creating-a-typescript-workflow-with-gulp>

The TypeScript file for this tutorial

The MashupJS isn't using TypeScript yet so I created a TypeScript file for this example.

I created a file in the root of the scr folder named myTypeScript.ts .

I added the following code snippet from

<http://www.typescriptlang.org/Playground> to the new file.

```
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return "Hello, " + this.greeting;
  }
}
```

```
var greeter = new Greeter("world");
```

```
var button = document.createElement('button');
button.textContent = "Say Hello";
button.onclick = function() {
  alert(greeter.greet());
}
```

```
document.body.appendChild(button);
```

Expected result

Once we've transpiled the TypeScript down to JavaScript (ES5) we will get a myTypeScript.js file and a minified version. Here is what the JS file should look like.

```
var Greeter = (function () {
  function Greeter(message) {
    this.greeting = message;
  }
});
```

```
}
Greeter.prototype.greet = function () {
    return "Hello, " + this.greeting;
};
return Greeter;
})();
var greeter = new Greeter("world");
var button = document.createElement('button');
button.textContent = "Say Hello";
button.onclick = function () {
    alert(greeter.greet());
};
document.body.appendChild(button);
```

Installing plugins

```
npm install gulp-typescript --save-dev
npm install gulp-tslint --save-dev
npm install gulp-tslint-stylish --save-dev
```

Add the new plugins to your Gulp required list

```
, ts           = require('gulp-typescript')
, tslint       = require('gulp-tslint')
, tsstylish    = require('gulp-tslint-stylish')
```

Add the new task to your gulpfile.js

```
gulp.task('tscompile', function () {
  return gulp.src(['./dist/**/*.ts', '!dist/core/lib/**/*.*', '!dist/core/css/**/*.*'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(sourceMaps.init())
    .pipe(ts({
      target: 'ES5',
      declarationFiles: false,
      noExternalResolve: true
    }))
    .pipe(rename({ extname: '.js' }))
    .pipe(gulp.dest('dist/.'));
});
```

And add the new task to your default task

```
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy',
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml', 'minifyimage',
    'grunt-merge-json:menu', 'jshint', 'tscompile'],
    ['uglifyalljs', 'minifycss']);
});
```

Executing the task

You can execute the task individually

```
gulp
```

```
C:\Windows\System32\cmd.exe
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[13:43:17] Using gulpfile C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[13:43:17] Starting 'default'...
[13:43:17] Starting 'annotate'...
[13:43:17] Finished 'default' after 13 ms
[13:43:18] Finished 'annotate' after 245 ms
[13:43:18] Starting 'clean-dist'...
[13:43:18] Finished 'clean-dist' after 288 ms
[13:43:18] Starting 'copy'...
[13:43:19] Finished 'copy' after 1.35 s
[13:43:19] Starting 'coreservices'...
[13:43:19] Starting 'routeconfig'...
[13:43:19] Starting 'libs'...
[13:43:19] Starting 'minifyhtml'...
[13:43:19] Starting 'minifyimage'...
[13:43:19] Starting 'grunt-merge-json:menu'...
[13:43:19] Starting 'jshint'...
[13:43:19] Starting 'tscompile'...
[13:43:19] Finished 'libs' after 44 ms

C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\app.js
  line 1 col 253 Line is too long.

  !! 1 warning

[13:43:19] gulp-imagemin: Minified 0 images
[13:43:20] Finished 'minifyimage' after 1.16 s
Running "merge-json:menu" (merge-json) task
File "dist/menu.json.txt" created.

Done, without errors.
[13:43:20] Finished 'grunt-merge-json:menu' after 1.16 s
[13:43:20] Finished 'tscompile' after 1.16 s
[13:43:20] Finished 'routeconfig' after 1.21 s
[13:43:20] Finished 'coreservices' after 1.31 s

C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\common\services\cache.service.js
  line 191 col 126 Line is too long.

  !! 1 warning

[13:43:21] Finished 'jshint' after 1.53 s
[13:43:21] Finished 'minifyhtml' after 1.58 s
[13:43:21] Starting 'uglifyalljs'...
[13:43:21] Starting 'minifycss'...
[13:43:23] Finished 'minifycss' after 1.89 s
[13:43:23] Finished 'uglifyalljs' after 1.9 s

C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

How it works

First the task transpiles the TypeScript code down to ES5, ECMAScript 5.

Then the new ES5 JavaScript is emitted. This file will not be used by the min.js.map. The map file will point directly to the TypeScript, “.ts”, file. The reason we are emitting the “.js” is so the TypeScript code can participate in the JSHint process. You’ll notice the ‘jshint’ task now has a dependency on the ‘tscompile’ task via the runSequence function.

Finally the TypeScript file is emitted as a minified JavaScript file with an associated map file linking it back to the TypeScript file.

jshint-output.html

If you run the new task with gulp tscompile, you’ll notice the jshint-output.html file has been updated.

If you double-click this file from explorer.exe, then you’ll notice our JavaScript, emitted by TypeScript, has a few issues.

Code	Line	Column	Evidence	Reason
D:\Subversion-GitHub\mashups\src\Mashup.UI.Core\dist\myTypeScript.js				
W109	6	25	<code>return "Hello, " + this.greeting;</code>	Strings must use singlequote.
W109	10	34	<code>var greeter = new Greeter("world");</code>	Strings must use singlequote.
W109	12	33	<code>button.textContent = "Say Hello";</code>	Strings must use singlequote.
W117	14	5	<code>alert(greeter.greet());</code>	'alert' is not defined.

Found **4** failures - **0** errors & **4** warnings

Fixing the JSHint warnings

These are pretty simple changes to make. Replace all the double quotes with single quotes and define the alert() method and JSHint is satisfied. The end result should be this.

```
/*global alert */

class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return 'Hello, ' + this.greeting;
  }
}

var greeter = new Greeter('world');

var button = document.createElement('button');
button.textContent = 'Say Hello';
button.onclick = function () {
  alert(greeter.greet());
}

document.body.appendChild(button);
```

Linting TypeScript

TypeScript has language constructs ES5 JavaScript does next so automating a TypeScript specific linter might seem redundant. It's not.

Add the following task to your gulpfile.js

```
gulp.task('tslint', ['copy'], function () {
  return gulp.src(['./dist/**/*.ts', '!dist/core/lib/**/*.ts', '!dist/core/css/**/*.ts'])
    .pipe(tslint())
    .pipe(tslint.report('verbose', {
      emitError: false,
      sort: true,
      bell: true
    }));
});
```

TSLint reporter

I was unable to find a plugin to export TSLint errors to an HTML file as we did for JSHint. If this plugin becomes available, I'll add it to this post.

TSLint configuration

Configuration information for TSLint is stored in a file named "tslint.json".

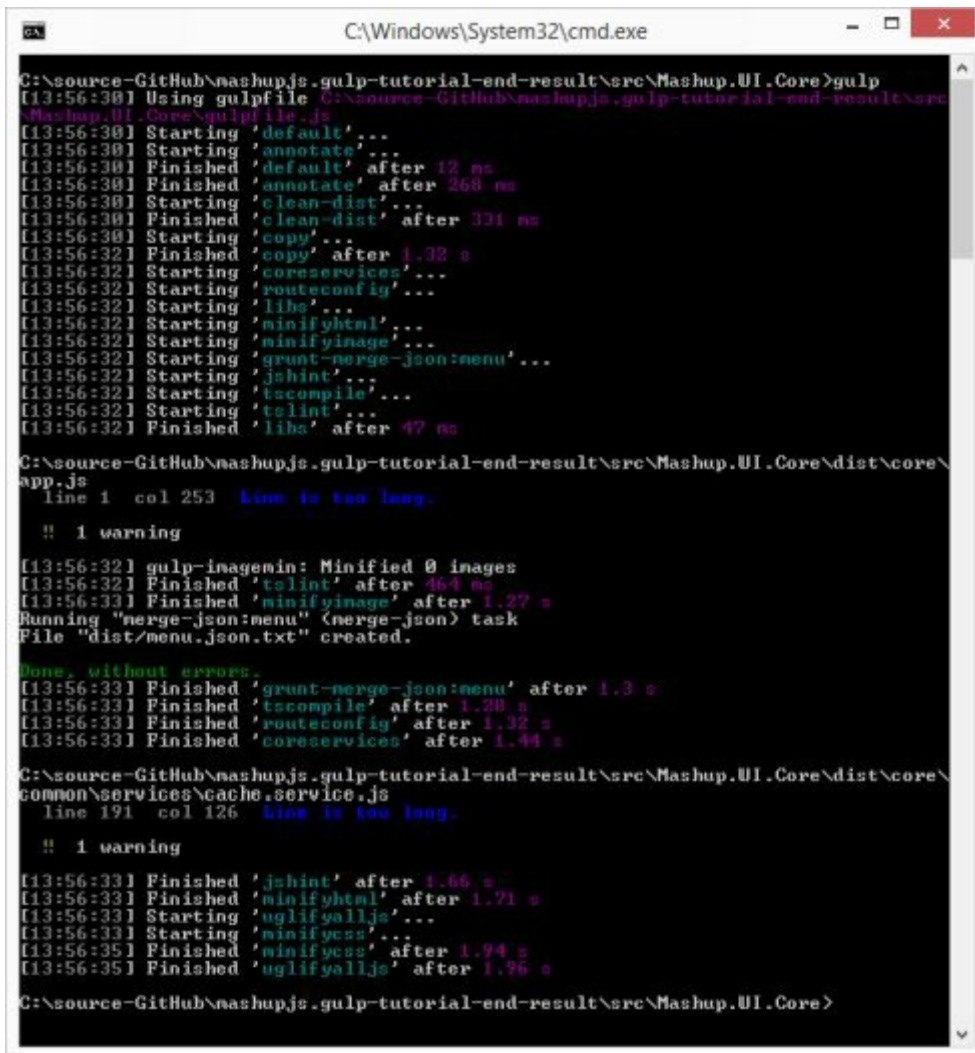
Here is a good starting point. I've borrowed this from other developer posts and it seems to be a good list.

Create a tslint.json file in the same folder as your gulpfile.js and package.json and add the following code to it.

```
{
  "rules": {
    "class-name": true,
    "curly": true,
    "eofline": false,
    "forin": true,
    "indent": [true, 4],
    "label-position": true,
    "label-undefined": true,
    "max-line-length": [true, 140],
    "no-arg": true,
    "no-bitwise": true,
    "no-console": [true,
      "debug",
      "info",
      "time",
      "timeEnd",
      "trace"
    ],
    "no-construct": true,
    "no-debugger": true,
    "no-duplicate-key": true,
    "no-duplicate-variable": true,
    "no-empty": true,
    "no-eval": true,
    "no-imports": true,
    "no-string-literal": false,
    "no-trailing-comma": true,
    "no-trailing-whitespace": true,
    "no-unused-variable": false,
    "no-unreachable": true,
    "no-use-before-declare": true,
    "one-line": [true,
      "check-open-brace",
      "check-catch",
      "check-else",
      "check-whitespace"
    ],
    "quotemark": [true, "single"],
    "radix": true,
    "semicolon": true,
    "triple-equals": [true, "allow-null-check"],
    "variable-name": false,
    "whitespace": [true,
      "check-branch",
      "check-decl",
      "check-operator",
      "check-separator"
    ]
  }
}
```

Run the default task

gulp



```
C:\Windows\System32\cmd.exe
C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[13:56:30] Using gulpfile C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[13:56:30] Starting 'default'...
[13:56:30] Starting 'annotate'...
[13:56:30] Finished 'default' after 12 ms
[13:56:30] Finished 'annotate' after 268 ms
[13:56:30] Starting 'clean-dist'...
[13:56:30] Finished 'clean-dist' after 331 ms
[13:56:30] Starting 'copy'...
[13:56:32] Finished 'copy' after 1.32 s
[13:56:32] Starting 'coreservices'...
[13:56:32] Starting 'routeconfig'...
[13:56:32] Starting 'libs'...
[13:56:32] Starting 'minifyhtml'...
[13:56:32] Starting 'minifyimage'...
[13:56:32] Starting 'grunt-merge-json:menu'...
[13:56:32] Starting 'jshint'...
[13:56:32] Starting 'tscompile'...
[13:56:32] Starting 'tallint'...
[13:56:32] Finished 'libs' after 47 ms
C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\app.js
  line 1 col 253  line is too long.

  !! 1 warning

[13:56:32] gulp-imagemin: Minified 0 images
[13:56:32] Finished 'tallint' after 464 ms
[13:56:33] Finished 'minifyimage' after 1.27 s
Running "merge-json:menu" (merge-json) task
File "dist/menu.json.txt" created.

Done, without errors.
[13:56:33] Finished 'grunt-merge-json:menu' after 1.3 s
[13:56:33] Finished 'tscompile' after 1.38 s
[13:56:33] Finished 'routeconfig' after 1.32 s
[13:56:33] Finished 'coreservices' after 1.44 s
C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\common\services\cache.service.js
  line 191 col 126  line is too long.

  !! 1 warning

[13:56:33] Finished 'jshint' after 1.66 s
[13:56:33] Finished 'minifyhtml' after 1.71 s
[13:56:33] Starting 'uglifyalljs'...
[13:56:33] Starting 'minifycss'...
[13:56:35] Finished 'minifycss' after 1.94 s
[13:56:35] Finished 'uglifyalljs' after 1.96 s
C:\source-GitHub\mashup.js.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

For more information

<https://www.npmjs.com/package/gulp-typescript>

GULP TUTORIAL PART 13 – SASS

For your SASS implementation, we will use the Gulp-SASS plugin.

From the command-line install

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

Add the module to the Gulp file

```
, sass = require('gulp-sass')
```

Add the task to the Gulp file

```
gulp.task('sass', function () {  
  gulp.src('./dist/**/*.*scss', { base: 'dist/.' })  
    .pipe(plumber({  
      errorHandler: onError  
    })))  
    .pipe(sass())  
    .pipe(gulp.dest('dist/.'));  
});
```

Add the new task to the default task

```
gulp.task('default', function () {  
  runSequence('annotate', 'clean-dist', 'copy',  
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml', 'minifyimage',  
    'grunt-merge-json:menu', 'jshint', 'tscompile', 'tslint', 'sass'],  
    ['uglifyalljs', 'minifycss']);  
});
```

Run the default task

```
gulp
```

```
C:\Windows\System32\cmd.exe
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[14:05:15] Using gulpfile C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[14:05:15] Starting 'default'...
[14:05:15] Finished 'default' after 13 ms
[14:05:16] Starting 'annotate'...
[14:05:16] Finished 'annotate' after 254 ms
[14:05:16] Starting 'clean-dist'...
[14:05:16] Finished 'clean-dist' after 355 ms
[14:05:16] Starting 'copy'...
[14:05:17] Finished 'copy' after 1.35 s
[14:05:17] Starting 'concatservice'...
[14:05:17] Starting 'routeconfig'...
[14:05:17] Starting 'libs'...
[14:05:17] Starting 'minifyhtml'...
[14:05:17] Starting 'minifyimage'...
[14:05:17] Starting 'grunt-merge-json:menu'...
[14:05:17] Starting 'jshint'...
[14:05:17] Starting 'tsc:compile'...
[14:05:17] Starting 'tshint'...
[14:05:17] Starting 'sass'...
[14:05:17] Finished 'sass' after 1.03 ms
[14:05:17] Finished 'libs' after 40 ms
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\app.js
  line 1 col 253 line is too long.
  # 1 warning
[14:05:18] gulp-imagemin: Minified 0 images
[14:05:18] Finished 'tshint' after 465 ms
[14:05:18] Finished 'minifyimage' after 1.26 s
Running "merge-json:menu" (merge-json) task
File "dist/menu.json.txt" created.
Done, without errors.
[14:05:18] Finished 'grunt-merge-json:menu' after 1.26 s
[14:05:18] Finished 'tsc:compile' after 1.27 s
[14:05:18] Finished 'routeconfig' after 1.32 s
[14:05:19] Finished 'concatservice' after 1.41 s
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\common\services\cache.service.js
  line 191 col 126 line is too long.
  # 1 warning
[14:05:19] Finished 'jshint' after 1.68 s
[14:05:19] Finished 'minifyhtml' after 1.73 s
[14:05:19] Starting 'uglifyalljs'...
[14:05:19] Starting 'minifycss'...
[14:05:21] Finished 'minifycss' after 1.94 s
[14:05:21] Finished 'uglifyalljs' after 1.96 s
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>
```

Learn more about SASS here

<http://sass-lang.com/>

The current state of code

We are starting to build up a nice collection of files. Here is the state the code should be in. If you're having trouble, then use this to level set yourself so you can move forward.

package.json

```
{
  "name": "Mashup.UI.Core",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "grunt": "^0.4.5",
    "grunt-merge-json": "^0.9.5",
    "gulp": "^3.9.0",
    "gulp-clean": "^0.3.1",
    "gulp-concat": "^2.5.2",
    "gulp-grunt": "^0.5.2",
    "gulp-imagemin": "^2.2.1",
    "gulp-jshint": "^1.11.0",
    "gulp-jshint-html-reporter": "^0.1.2",
    "gulp-minify-css": "^1.1.6",
    "gulp-minify-html": "^1.0.3",
```

```
"gulp-newer": "^0.5.1",
"gulp-ng-annotate": "^1.0.0",
"gulp-plumber": "^1.0.1",
"gulp-rename": "^1.2.2",
"gulp-sass": "^2.0.1",
"gulp-sourcemaps": "^1.5.2",
"gulp-tslint": "^3.0.2-beta",
"gulp-tslint-stylish": "^1.0.1",
"gulp-typescript": "^2.7.7",
"gulp-uglify": "^1.2.0",
"imagemin-pngquant": "^4.1.0",
"jshint-stylish": "^2.0.0",
"run-sequence": "^1.1.1"
}
}
```

gulpfile.js

```
var onError = function (err) {
  console.log(err);
};

var gulp = require('gulp')
  , uglify      = require('gulp-uglify')
  , rename      = require('gulp-rename')
  , sourcemaps  = require('gulp-sourcemaps')
  , runSequence = require('run-sequence')
  , plumber     = require('gulp-plumber')
  , ngAnnotate  = require('gulp-ng-annotate')
  , clean       = require('gulp-clean')
  , newer       = require('gulp-newer')
  , concat      = require('gulp-concat')
  , rename      = require('gulp-rename')
  , uglify      = require('gulp-uglify')
  , sourcemaps  = require('gulp-sourcemaps')
  , minifycss   = require('gulp-minify-css')
  , minifyhtml  = require('gulp-minify-html')
  , imagemin    = require('gulp-imagemin')
  , pngquant    = require('imagemin-pngquant')
  , jshint       = require('gulp-jshint')
  , stylish     = require('jshint-stylish')
  , jshinthtmlreporter = require('gulp-jshint-html-reporter')
  , ts          = require('gulp-typescript')
  , tslint      = require('gulp-tslint')
  , tsstylish   = require('gulp-tslint-stylish')
  , sass        = require('gulp-sass')

;

gulp.task('annotate', function () {
  return gulp.src(['src/index.controller.js', 'src/core/**/*.*.js', 'src/apps/**/*.*.js',
    'src/core/lib/**/*.*', '!src/**/*.*.min.js'], { base: 'src/.' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(ngAnnotate())
    .pipe(gulp.dest('src/.'));
});

gulp.task('clean-dist', function () {
  return gulp.src('dist', { read: false })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(clean());
});

gulp.task('copy', function () {
  return gulp.src('src/**/*')
    .pipe(plumber({
      errorHandler: onError
    }))

```

```
.pipe(newer('dist'))
.pipe(gulp.dest('dist'));
});
```

```
gulp.task('coreservices', function () {
  return gulp.src('src/core/common/**/*')
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(concat('core.services.js'))
    .pipe(gulp.dest('./dist/'));
});
```

```
gulp.task('routeconfig', function () {
  return gulp.src(['src/core/config/route.config.js', 'src/apps/**/route.config.js'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(concat('route.config.js'))
    .pipe(gulp.dest('./dist/'));
});
```

```
gulp.task('libs', function () {
  return gulp.src(['bower_components/**/*.*.js'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(concat('libs.js'))
    .pipe(gulp.dest('dist/core/lib/'));
});
```

```
gulp.task('uglifyalljs', function () {
  //gulp.task('uglifyalljs', ['copy', 'coreservices', 'routeconfig', 'tscompile'], function () {
  return gulp.src(['dist/**/*.*.js', '!/**/*.*.min.js', '!dist/core/lib/**/*.*', '!dist/core/common/**/*.*'],
    { base: 'dist/.' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(sourcemaps.init())
    // .pipe(newer('dist/.'))
    .pipe(uglify())
    .pipe(rename({
      extname: '.min.js'
    }))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest('dist/.'));
});
```

```
gulp.task('minifycss', function () {
  return gulp.src(['dist/**/*.*.css', '!dist/**/*.*.min.css', '!dist/core/lib/**/*.*'], { base: 'dist/.' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(sourcemaps.init())
    .pipe(minifycss())
    .pipe(rename({
      extname: '.min.css'
    }))
});
```

```
.pipe(sourcemaps.write('./'))
.pipe(gulp.dest('dist/./'));
});
```

```
gulp.task('minifyhtml', function () {
  return gulp.src(['dist/**/*.*.html', '!**/*.min.html', '!dist/core/lib/**/*.*'], { base: 'dist/./' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(sourcemaps.init())
    .pipe(minifyhtml())
    .pipe(rename({
      extname: '.min.html'
    }))
    .pipe(sourcemaps.write('./'))
    .pipe(gulp.dest('dist/./'));
});
```

```
gulp.task('minifyimage', function () {
  return gulp.src(['dist/**/*.*.{png,jpg,gif,ico}', '!dist/core/lib/**/*.*', '!dist/core/css/**/*.*'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(imagemin({ progressive: true, optimizationLevel: 7, use: [pngquant()] }))
    .pipe(gulp.dest('dist/./'));
});
```

```
// -----
// Grunt configuration
require('gulp-grunt')(gulp, {
  // These are the default options but included here for readability.
  base: null,
  prefix: 'grunt-',
  verbose: false
});
// -----
```

```
gulp.task('jshint', function () {
  //gulp.task('jshint', ['copy', 'tscompile'], function () {
  return gulp.src(['./dist/**/*.*.js', '!dist/core/lib/**/*.*', '!**/*.min.js', '!dist/core/css/**/*.*'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(jshint('.jshintrc'))
    .pipe(jshint.reporter(stylish))
    .pipe(jshint.reporter('gulp-jshint-html-reporter', { filename: 'jshint-output.html' }));
});
```

```
gulp.task('tscompile', function () {
  return gulp.src(['./dist/**/*.*.ts', '!dist/core/lib/**/*.*', '!dist/core/css/**/*.*'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(sourcemaps.init())
    .pipe(ts({
      target: 'ES5',
```



```

    declarationFiles: false,
    noExternalResolve: true
  )))
  .pipe(rename({ extname: '.js' })))
  .pipe(gulp.dest('dist/.'));
});

gulp.task('tslint', function () {
  return gulp.src(['./dist/**/*.ts', '!dist/core/lib/**/*.*', '!dist/core/css/**/*.*'])
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(tslint())
    .pipe(tslint.report('verbose', {
      emitError: false,
      sort: true,
      bell: true
    })));
});

gulp.task('sass', function () {
  gulp.src('./dist/**/*.scss', { base: 'dist/' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(sass())
    .pipe(gulp.dest('dist/.'));
});

// -----
// Default Task
// -----
gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy',
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml', 'minifyimage',
    'grunt-merge-json:menu', 'jshint', 'tscompile', 'tslint', 'sass'],
    ['uglifyalljs', 'minifycss']);
});

```

Gruntfile.js

```

module.exports = function (grunt) {

  grunt.initConfig({
    distFolder: 'dist',

    pkg: grunt.file.readJSON('package.json'),
    "merge-json": {
      menu: {
        src: ['src/apps/**/*.menu.json.txt'],
        dest: '<%= distFolder %>/menu.json.txt',
      },
    },
  });
};

```

```
// Load modules, register tasks
grunt.loadNpmTasks('grunt-merge-json');

};
```

.jshintre

```
{
  "bitwise": true,
  "camelcase": true,
  "curly": true,
  "eqeqeq": true,
  "es3": false,
  "forin": true,
  "freeze": true,
  "immed": true,
  "indent": 4,
  "latedef": "nofunc",
  "newcap": true,
  "noarg": true,
  "noempty": true,
  "nonbsp": true,
  "nonew": true,
  "plusplus": false,
  "quotmark": "single",
  "undef": true,
  "unused": false,
  "strict": false,
  "maxparams": 10,
  "maxdepth": 5,
  "maxstatements": 40,
  "maxcomplexity": 8,
  "maxlen": 120,

  "asi": false,
  "boss": false,
  "debug": false,
  "eqnull": true,
  "esnext": false,
  "evil": false,
  "expr": false,
  "funcscope": false,
  "globalstrict": false,
  "iterator": false,
  "lastsemic": false,
  "laxbreak": false,
  "laxcomma": false,
  "loopfunc": true,
  "maxerr": false,
  "moz": false,
  "multistr": false,
  "notypeof": false,
  "proto": false,
  "scripturl": false,
  "shadow": false,
  "sub": true,
```

```
"supernew": false,  
"validthis": false,  
"noyield": false,  
  
"browser": true,  
"node": true,  
  
"globals": {  
  "angular": false,  
  "$": false  
}  
}
```

tslint.json

```
{  
  "rules": {  
    "class-name": true,  
    "curly": true,  
    "eofline": false,  
    "forin": true,  
    "indent": [true, 4],  
    "label-position": true,  
    "label-undefined": true,  
    "max-line-length": [true, 140],  
    "no-arg": true,  
    "no-bitwise": true,  
    "no-console": [true,  
      "debug",  
      "info",  
      "time",  
      "timeEnd",  
      "trace"  
    ],  
    "no-construct": true,  
    "no-debugger": true,  
    "no-duplicate-key": true,  
    "no-duplicate-variable": true,  
    "no-empty": true,  
    "no-eval": true,  
    "no-imports": true,  
    "no-string-literal": false,  
    "no-trailing-comma": true,  
    "no-trailing-whitespace": true,  
    "no-unused-variable": false,  
    "no-unreachable": true,  
    "no-use-before-declare": true,  
    "one-line": [true,  
      "check-open-brace",  
      "check-catch",  
      "check-else",  
      "check-whitespace"  
    ],  
    "quotemark": [true, "single"],  
    "radix": true,  
    "semicolon": true,  
    "triple-equals": [true, "allow-null-check"],
```

```
"variable-name": false,  
"whitespace": [true,  
  "check-branch",  
  "check-decl",  
  "check-operator",  
  "check-separator"  
]  
}  
}
```

GULP TUTORIAL PART 14 – WATCH

Running the default task for Gulp, with all our tasks included, will consume more CPU and time than is required during development where files are changed one at a time. For dealing with individual files as they change, we can use the Watch plugin.

When the Watch is triggered, `gulp.watch` tasks execute.

From the command-line install

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

Add the module to the Gulp file

```
, watch = require('gulp-watch')
```

Add the task to the Gulp file

The “watch” task is large and uses many other tasks, so I place it at the bottom of the file so it is separate.

```
gulp.task('watch', function () {

  // -----
  // Watching JS files
  // -----
  // Copy all files except *.js files.
  gulp.watch(['src/**/*.!', 'src/**/*.js', '!bower_components/**/*.!'], function () { runSequence('copy'); });

  // Annotates and copies *.js files
  gulp.watch(['src/**/*.js',
    'src/core/config/route.config.js', 'src/apps/**/*.route.config.js',
    '!bower_components/**/*.js'], function () { runSequence('watch:annotate', 'copy'); });

  // routeConfig file changes.
  gulp.watch(['src/core/config/route.config.js', 'src/apps/**/*.route.config.js'], function () {
runSequence('routeconfig'); });

  // Uglify JS files
  gulp.watch(['dist/**/*.js', 'dist/**/*.min.js', 'dist/core/lib/**/*.!', 'dist/core/common/**/*.!'], function () {
runSequence('uglifyalljs'); });

  // -----
  // Watching Bower components
  // -----
  gulp.watch(['bower_components/**/*.js'], function () { runSequence('libs'); });
  // TODO: Add other bower component types like css, scss and images

  // -----
  // Watching css and scss files
  // -----
  gulp.watch(['dist/**/*.css', 'dist/**/*.min.css', 'dist/core/lib/**/*.!'], function () { runSequence('minifycss');
});
  gulp.watch(['dist/**/*.scss', 'dist/core/lib/**/*.!'], function () { runSequence('sass'); });

  // -----
  // Watching TypeScript files
  // -----
  gulp.watch(['dist/**/*.ts', 'dist/core/lib/**/*.!', 'dist/core/css/**/*.!'], function () {
runSequence('tscompile'); });

  // -----
  // Watch - Execute linters
  // -----
  gulp.watch(['dist/**/*.ts', 'dist/core/lib/**/*.!', 'dist/core/css/**/*.!'], function () { runSequence('tslint');
});
  //gulp.watch(['dist/**/*.js', 'dist/core/lib/**/*.!', 'dist/**/*.min.js', 'dist/core/css/**/*.!'], function() {
runSequence('jshint'); });

  gulp.watch(['dist/**/*.js', 'dist/core/lib/**/*.!', 'dist/**/*.min.js', 'dist/core/css/**/*.!', ['jshint']]);

  // -----
  // Watching image files
  // -----
  // unable to get this watch to ever notice a file changed. This will be handled on the initial build.
```

```

    //gulp.watch(['dist/**/*.{png,jpg,gif,ico}', '!dist/core/lib/**/*.*', '!dist/core/css/**/*.*'], function() {
runSequence('minifyimage'); });

});

```

Add another task to the gulp file

Also add this supporting Watch task. This helps improve performance with the “newer” plugin.

```

// -----
// Watch specific tasks. This is to support the use of newer.
// -----
gulp.task('watch:annotate', function () {
  return gulp.src(['src/index.controller.js', 'src/core/**/*.*.js', 'src/apps/**/*.*.js',
'!src/core/lib/**/*.*', '!**/*.*.min.js'], { base: 'src/' })
    .pipe(plumber({
      errorHandler: onError
    }))
    .pipe(newer('src/'))
    .pipe(ngAnnotate())
    .pipe(gulp.dest('src/'));
});

```

Add the new task to the default task

Add the new “watch” task to the default task. The default task will execute all the tasks in the sequence specified by the runSequence function. The last task run is the Watch task which contains 10 individual Watch tasks. This time, when you run the Gulp default task, the command line will not return control to you. To break out of this, press CTRL + C and then answer the prompt with “y”.

```

gulp.task('default', function () {
  runSequence('annotate', 'clean-dist', 'copy',
    ['coreservices', 'routeconfig', 'libs', 'minifyhtml', 'minifyimage',
    'grunt-merge-json:menu', 'jshint', 'tscompile', 'tslint', 'sass']
    , ['uglifyalljs', 'minifycss']
    , 'watch');
});

```

Run the default task

gulp

```
grunt
C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core>gulp
[14:28:52] Using gulpfile C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\gulpfile.js
[14:28:52] Starting 'default'...
[14:28:52] Starting 'annotate'...
[14:28:52] Finished 'default' after 13 ms
[14:28:52] Finished 'annotate' after 275 ms
[14:28:52] Starting 'clean-dist'...
[14:28:53] Finished 'clean-dist' after 328 ms
[14:28:53] Starting 'copy'...
[14:28:54] Finished 'copy' after 1.33 s
[14:28:54] Starting 'coreservices'...
[14:28:54] Starting 'routeconfig'...
[14:28:54] Starting 'libs'...
[14:28:54] Starting 'minifyhtml'...
[14:28:54] Starting 'minifyimage'...
[14:28:54] Starting 'grunt-merge-json:menu'...
[14:28:54] Starting 'jshint'...
[14:28:54] Starting 'tscompile'...
[14:28:54] Starting 'tslint'...
[14:28:54] Starting 'sass'...
[14:28:54] Finished 'sass' after 3.15 ms
[14:28:54] Finished 'libs' after 57 ms

C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\app.js
  line 1  col 253  line is too long.

  !! 1 warning

[14:28:54] gulp-inagemin: Minified 0 images
[14:28:54] Finished 'tslint' after 485 ms
[14:28:55] Finished 'minifyimage' after 1.33 s
Running "merge-json:menu" (merge-json) task
File "dist/menu.json.txt" created.

Done, without errors.
[14:28:55] Finished 'grunt-merge-json:menu' after 1.33 s
[14:28:55] Finished 'tscompile' after 1.33 s
[14:28:55] Finished 'routeconfig' after 1.38 s
[14:28:55] Finished 'coreservices' after 1.48 s

C:\source-GitHub\nashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\common\services\cache.service.js
  line 191  col 126  line is too long.

  !! 1 warning

[14:28:56] Finished 'jshint' after 1.73 s
[14:28:56] Finished 'minifyhtml' after 1.78 s
[14:28:56] Starting 'uglifyalljs'...
[14:28:56] Starting 'minifycss'...
[14:28:58] Finished 'minifycss' after 1.93 s
[14:28:58] Finished 'uglifyalljs' after 1.99 s
[14:28:58] Starting 'watch'...
[14:29:01] Finished 'watch' after 3.36 s
```

If you modify any files the Watch is configured to watch, then you'll see tasks run. Here is what happens after changing the index.controller.js file.


```
grunt
[14:32:38] Starting 'tallint'...
[14:32:38] Starting 'sass'...
[14:32:38] Finished 'sass' after 2.06 ms
[14:32:38] Finished 'libs' after 57 ms
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\
app.js
  line 1 col 253  line is too long.

  !! 1 warning
[14:32:38] gulp-imagemin: Minified 8 images
[14:32:38] Finished 'tallint' after 404 ms
[14:32:39] Finished 'minifyimage' after 1.31 s
Running "merge-json:menu" (merge-json) task
File "dist/menu.json.txt" created.
Done, without errors.
[14:32:39] Finished 'grunt-merge-json:menu' after 1.34 s
[14:32:39] Finished 'tcompile' after 1.31 s
[14:32:39] Finished 'routeconfig' after 1.37 s
[14:32:39] Finished 'coreservices' after 1.48 s
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\
common\services\cache.service.js
  line 191 col 126  line is too long.

  !! 1 warning
[14:32:39] Finished 'jshint' after 1.73 s
[14:32:39] Finished 'minifyhtml' after 1.78 s
[14:32:39] Starting 'uglifyalljs'...
[14:32:39] Starting 'minifycss'...
[14:32:41] Finished 'minifycss' after 1.92 s
[14:32:41] Finished 'uglifyalljs' after 1.94 s
[14:32:41] Starting 'watch'...
[14:32:45] Finished 'watch' after 3.48 s
[14:33:03] Starting 'watch:annotate'...
[14:33:03] Finished 'watch:annotate' after 88 ms
[14:33:03] Starting 'copy'...
[14:33:04] Starting 'uglifyalljs'...
[14:33:04] Starting 'jshint'...
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\
services.js
  line 334 col 126  line is too long.

  !! 1 warning
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\
app.js
  line 1 col 253  line is too long.

  !! 1 warning
C:\source-GitHub\mashupjs.gulp-tutorial-end-result\src\Mashup.UI.Core\dist\core\
common\services\cache.service.js
  line 191 col 126  line is too long.

  !! 1 warning
[14:33:05] Finished 'jshint' after 1.19 s
[14:33:05] Finished 'uglifyalljs' after 1.38 s
[14:33:05] Finished 'copy' after 1.53 s
```

First the annotate task runs against the changed JavaScript code. Then the file is copied to the dist folder and minified by the uglifyalljs task. Finally the JavaScript code is linted with the jshint task.

Tutorial end result

The completed code for this tutorial can be found here

<https://github.com/MashupJS/gulp-tutorial-end-result>

Installing Gulp

Execute both of these. The first adds Gulp locally so it can be used by NPM. The second installs Gulp globally so it can be accessed from the command line.

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

```
npm install gulp -g
```

Retrieve Gulp version

```
Grunt -version
```

Installing plugins

The syntax for Grunt plugins is

```
install [plugin-name] --save-dev
```

For example, if you want to minify and concatenate your JavaScript for performance, you would install two plugins.

Perform a quick Google search and you'll find this site

<https://github.com/gruntjs/grunt-contrib-uglify>

```
npm install grunt-contrib-uglify --save-dev
```

Perform a quick google search and you'll find this site

<https://github.com/gruntjs/grunt-contrib-concat>

```
npm install grunt-contrib-concat --save-dev
```

Retrieve Gulp version

```
Gulp --v
```

Every Gulp file needs a default task. To execute Gulp's default task

```
grunt
```

It's useful to run specific tasks that you have configured

grunt [task-name]

Get a list of Grunt commands

Grunt -help

To verify a plugin is not blacklisted

Gulp --verify

Testing tasks while building your gulpfile.js

You can type `gulp [task-name]` and your task will run. If it has any dependencies then those dependencies will run first.

`gulp [task-name]`

GULP TUTORIAL PART 16 – GLOB TIPS

Here are some of the more common Glob patterns.

“ dir/* ” – includes all files

“ dir/** ” – includes all files and folders

“**/*.js” – all JavaScript files

“!*/*.min.js” – excludes all minified JavaScript files

“!app/lib/**/*” – excludes all files in the lib folder.

<https://github.com/isaacs/node-glob>

<http://mywiki.woledge.org/glob#preview>

GULP TUTORIAL PART 17 – USEFUL NPM PACKAGES/COMMANDS

COMMANDS CHEAT SHEET

Find outdated modules

```
npm outdated --depth=0
```

```
npm outdated --json --depth=0
```

Installing a package

```
npm install grunt-contrib-uglify@* --save-dev
```

Updating local packages

When someone has added modules since your last check-in just run.

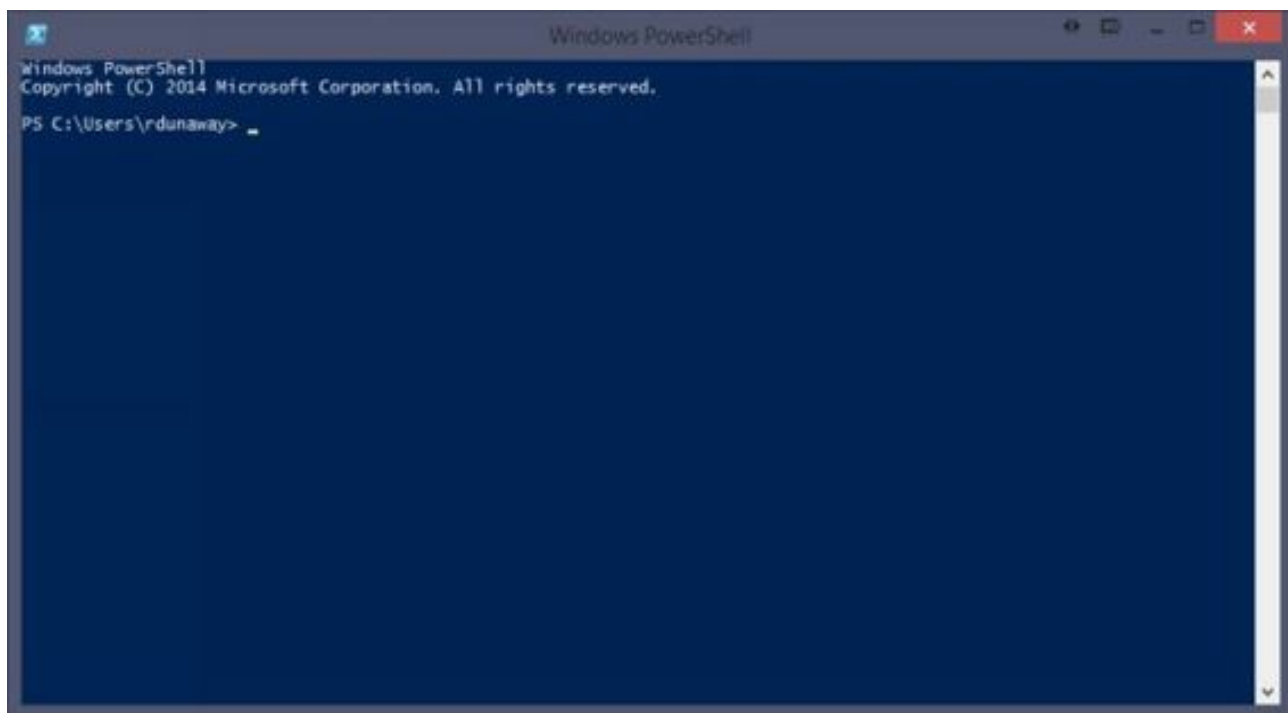
```
npm update
```

POWERSHELL (PRIMER)

Windows users can use either the Command Prompt or PowerShell.

PowerShell is pre-installed on Windows 8 or newer installations.

From Start: *Search programs and files* type “powershell”. Select “powershell.exe”.



SYNTAX

Powershell Command Syntax: **application action –flags arguments**

For help with any application add the **–h** or **–help** flags for additional instructions.

The **tab** key autocompletes your statement.

ADDING AND REMOVING FILES

To create a new item use the **ni** command. This might not seem useful with Visual Studio 2013 because any file added must also be added to your project file.

Visual Studio 2015 does not have a project file needing updates. Instead a Glob pattern is used to determine what files should and should not be included in the project. That being the case, suddenly, **ni** makes more sense.

Adding files

`ni newjsfile.js -type file`

`new-item newjsfile.js -type file`

Removing files

`rinewjsfile.js` or `remove-item newjsfile.js`

INSTALLING NODEJS AND NPM PACKAGES

Install NodeJS from:

<https://nodejs.org/>

Install NPM packages with the following syntax

`npm install [global option -g] [package-name] [options]`

Example: (You need to install Gulp both locally and globally)

`npm install gulp --save-dev`

`npm install gulp -g`

<https://docs.npmjs.com/getting-started/installing-npm-packages-locally>

NPM VERSION UPDATES

There are multiple options for keeping NPM packages up to date. The approach you choose might depend on your development workflow and automated testing solution, i.e., if you have good automated testing, it might be safe to allow the latest versions. If not, you might want to choose a more deliberate approach to NPM versioning.

VERSION UPDATES: OPTION 1 – USING NODE TOOLS

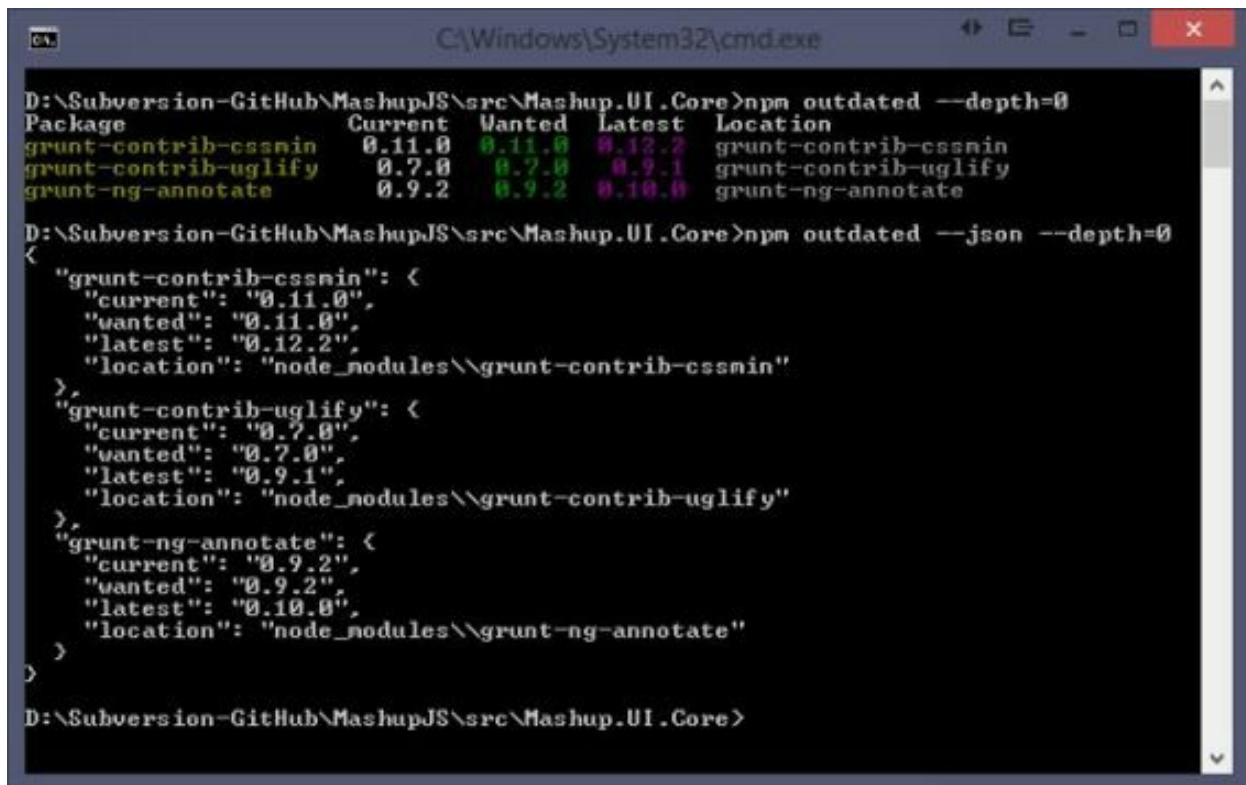
Check to see which NPM packages are out of date

Display colored rows

```
npm outdated --depth=0
```

Display in json which includes current, wanted, latest version numbers

```
npm outdated --json --depth=0
```



```
C:\Windows\System32\cmd.exe
D:\Subversion-GitHub\MashupJS\src\Mashup.UI.Core>npm outdated --depth=0
Package      Current  Wanted  Latest  Location
grunt-contrib-cssmin  0.11.0  0.11.0  0.12.2  grunt-contrib-cssmin
grunt-contrib-uglify  0.7.0   0.7.0   0.9.1   grunt-contrib-uglify
grunt-ng-annotate  0.9.2   0.9.2   0.10.0  grunt-ng-annotate

D:\Subversion-GitHub\MashupJS\src\Mashup.UI.Core>npm outdated --json --depth=0
{
  "grunt-contrib-cssmin": {
    "current": "0.11.0",
    "wanted": "0.11.0",
    "latest": "0.12.2",
    "location": "node_modules\\grunt-contrib-cssmin"
  },
  "grunt-contrib-uglify": {
    "current": "0.7.0",
    "wanted": "0.7.0",
    "latest": "0.9.1",
    "location": "node_modules\\grunt-contrib-uglify"
  },
  "grunt-ng-annotate": {
    "current": "0.9.2",
    "wanted": "0.9.2",
    "latest": "0.10.0",
    "location": "node_modules\\grunt-ng-annotate"
  }
}
```

Note: Not all your packages will be displayed. Only the outdated packages will be displayed.

Note: If you modify the command to include “-g” then you’ll get a list of your outdated global packages.

To update packages one at a time

```
npm install [package-name]@* [save?]
```

```
npm install grunt-contrib-uglify@* --save-dev
```

VERSION UPDATES: OPTION 2 – USING NPM-CHECK-UPDATES

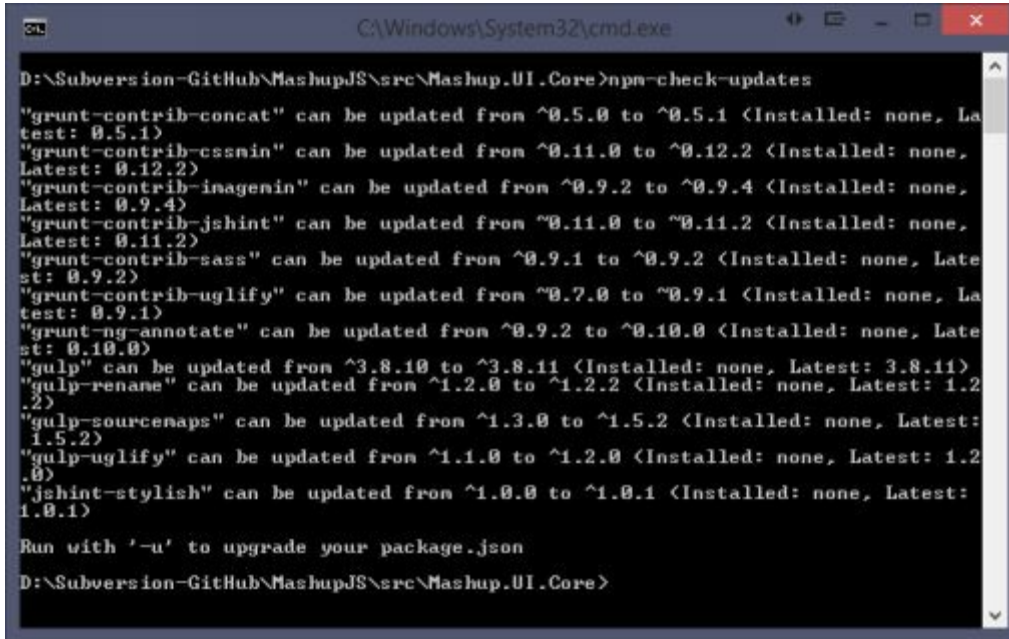
Using the npm-check-updates package, you can keep all your packages updated.

<https://www.npmjs.com/package/npm-check-updates>


```
npm install -g npm-check-updates
```

Then execute the following command to see what packages can be updated.

```
npm-check-updates
```



```
C:\Windows\System32\cmd.exe
D:\Subversion-GitHub\MashupJS\src\Mashup.UI.Core>npm-check-updates
"grunt-contrib-concat" can be updated from ^0.5.0 to ^0.5.1 <Installed: none, Latest: 0.5.1>
"grunt-contrib-cssmin" can be updated from ^0.11.0 to ^0.12.2 <Installed: none, Latest: 0.12.2>
"grunt-contrib-inagenin" can be updated from ^0.9.2 to ^0.9.4 <Installed: none, Latest: 0.9.4>
"grunt-contrib-jshint" can be updated from ^0.11.0 to ^0.11.2 <Installed: none, Latest: 0.11.2>
"grunt-contrib-sass" can be updated from ^0.9.1 to ^0.9.2 <Installed: none, Latest: 0.9.2>
"grunt-contrib-uglify" can be updated from ^0.7.0 to ^0.9.1 <Installed: none, Latest: 0.9.1>
"grunt-ng-annotate" can be updated from ^0.9.2 to ^0.10.0 <Installed: none, Latest: 0.10.0>
"gulp" can be updated from ^3.8.10 to ^3.8.11 <Installed: none, Latest: 3.8.11>
"gulp-rename" can be updated from ^1.2.0 to ^1.2.2 <Installed: none, Latest: 1.2.2>
"gulp-sourcemaps" can be updated from ^1.3.0 to ^1.5.2 <Installed: none, Latest: 1.5.2>
"gulp-uglify" can be updated from ^1.1.0 to ^1.2.0 <Installed: none, Latest: 1.2.0>
"jshint-stylish" can be updated from ^1.0.0 to ^1.0.1 <Installed: none, Latest: 1.0.1>
Run with '-u' to upgrade your package.json
D:\Subversion-GitHub\MashupJS\src\Mashup.UI.Core>
```

To upgrade all your packages

```
npm-check-updates-u [ -g option for global packages]
```

Now your package.json is updated.

Then execute an NPM install to update the package installations.

```
npm install [ -g option for global packages]
```

NPM VERSIONING SEMANTICS

<https://docs.npmjs.com/misc/semver>

<http://semver.org/>

The End

Inhaltsverzeichnis

Gulp Tutorials	5
GULP Tutorial Part 1 – Reasons for Build Tools like Gulp Productivity	6
GULP Tutorial Part 2 – Setup	7
GULP Tutorial Part 3 – Adding Plugins	13
GULP TUTORIAL PART 4 – Sequence and Parallel task processing	16
GULP TUTORIAL PART 5 – Handling errors with Plumber	18
GULP Tutorial Part 6 – Optimizing JavaScript/TypeScript Annotation	20
Clean out ‘dist’	22
Copy all src files to ‘dist’	23
CONCATENATION	25
Compress and Minify JavaScript	28
GULP Tutorial Part 7 – Optimizing CSS	33
GULP Tutorial Part 8 – Optimizing HTML	35
GULP Tutorial Part 9 – Images	37
GULP Tutorial Part 10 – JSON, calling grunt from gulp	39
GULP Tutorial Part 11 – JSHINT	46
GULP Tutorial Part 12 – TypeScript	51
GULP Tutorial Part 13 – SASS	59
GULP Tutorial Part 14 – Watch	69
GULP Tutorial Part 15 – Useful Gulp Commands & Tips	74
GULP Tutorial Part 16 – Glob Tips	76
GULP TUTORIAL PART 22 – Useful NPM Packages/Commands	76
Commands Cheat Sheet	77
PowerShell (primer)	77
Syntax	77
Adding and removing files	78
Installing NodeJS and NPM Packages	79
NPM Version updates	79