Grunt

NOTE: Giving up on grunt. I was able to get a good grunt default going but the watch doesn’t work. When one file changes grunt tries to runt tasks for all files so it’s very slow. Also it takes Grunt a long time to discover that a file has changed. For some reason the watch isn’t obeying my “files:” config. I tell it to ignore \*.min.js and it tries to minify them anyway, creating min.js, min.min.js, min.min.min.js, etc…

At this point I can see that Grunt is too much trouble unless you have a very simple and small application. This might not be true but I’m smart enough that if I can’t figure out how to get what I want after all this time, that’s an indicator that this is not the right tool for my team.

I’m writing this so I remember why I switched to Gulp.

Intro

Grunt Setup

Useful Grunt Commands

Mashups Grunt Implementation

NPM Packages (own blog)

# Chronological Grunt Notes

Some considerations

* Simplest solution and most bundled solution is to create a single js file.
* Another option is to allow for many smaller files
  + Drawback is there are more get requests of js files.
  + Benefit is a large js file isn’t downloaded every time the app is updated.
  + Drawback: more complex grunt implementation.

# Intro

TODO

# Grunt Setup

Download and install NodeJS or IoJS. IoJS is a fork of NodeJS by many in the community who wanted more responsive Node advancement. Theoretically you should be able to use either for the Mashup.

<https://nodejs.org/>

<https://iojs.org/en/index.html>

At this point Grunt can be installed with the following command. Notice the “-g” command. This causes the npm package to be deployed globally.

npm install -g grunt-cli

Before you can use grunt you must create the “package.json” and “Gruntfile.json files. These must exist in the root of your project file.

Create a basic package.json file

npm init

Installing Grunt

npm install -g grunt-cli

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

module.exports = function(grunt) {

grunt.initConfig({

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

});

// task setup

grunt.registerTask('default', []);

};

# Useful Grunt Commands

## Installing Grunt

npm install -g grunt-cli

## Retrieve Grunt 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

## Grunt Tips

**What your grunt tasks run. This is useful when troubleshooting your grunt config.**

Grunt --v

**Every grunt file needs a default task that runs when you type**

grunt

**It’s useful to run specific tasks that you have configured**

grunt [task name]

**Get a list of grunt commands**

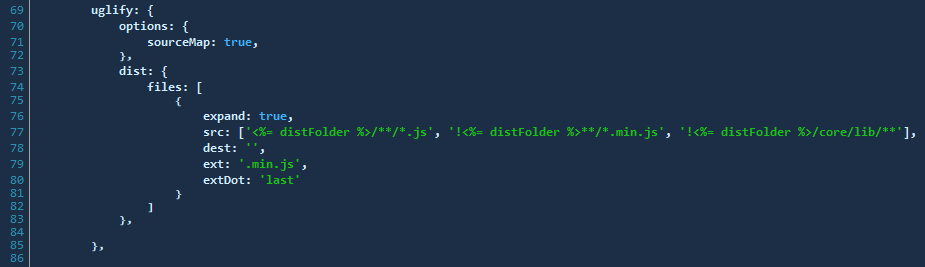
Grunt –help

**Last dot**

When creating new files and adding a new extension Grunt gets confused about which dot is the last dot designating the extension. If your file has multiple dots then you won’t get the result you’re looking for when adding “min.js” for example, to minified JavaScript files.

my.JavaScript.js would be renamed to my.js.

Add to your configuration: extDot: ‘last’



## Glob Tips

“dir/\*” – includes all files

“dir/\*\*” – includes all files and directories

# MashupJS Grunt Implementation

The Mashup will maintain both Grunt and Gulp. You can use whichever works best for your workflow.

Grunt is a task runner used in the development and the build processes. All files are processed and distributed to the “/dist” directory. Everything needed for deployment should be found here.

To make this work there are a few things you need to do. If you are using VS2015 then some of this will happen auto-magically.

This document will walk you through the basic setup and usage of Grunt, how Grunt is configured for the Mashup, and how to use Grunt as part of the Mashup workflow.

TODO: Explain Mashup’s Grunt implementation

# NPM Packages

**Commands**

* TODO – command that updates the package.json version number. Required before npm publish

**How to configure for 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. IE: If you have good automated testing it might be safe to allow the latest versions. If not then you might want to choose a more deliberate approach to npm versioning.

## 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



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

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

## 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



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/>