

“Building” Modern Web Applications

Darrell Pratt, Architect Leader, GBS

Executive Summary

With the increase in popularity of one page web applications, the JavaScript lying underneath these applications has grown more advanced. The number of libraries that are used with these applications and the manner that they can be packaged and deployed has advanced a great deal in the last year. This paper will detail some of the more popular build chains that are used throughout the JavaScript developer community and give details as to how these tools can increase developer productivity and deliver more reliable applications.

Problem Statement

Modern web applications include a wide variety of assets which must be packaged in various ways in order to deliver a good experience to the end user. From source minification, dynamic module loaders, SASS and LESS compilers, CoffeeScript transpilation and dependency management, the build process and application lifecycle management has become increasingly complex.

Dependency Management

Bower

A few tools exist now that provide a solution for dependency management in a way similar to Maven in Java. The primary tool in this space is Bower. Bower is used by many other build frameworks such as Twitter’s bootstrap and Google’s Yeoman. Bower uses a simple manifest file to specify a set of libraries and their versions which are required by the application. Bower uses GitHub to house the libraries not unlike Maven with its central repository.

Bower uses a file named bower.json in the root directory of the project to specify the dependencies. This file has a simple format shown below.

Configuration Sample

```
{
  "name": "client",
  "version": "0.0.0",
  "dependencies": {
    "angular": "1.2.6",
    "json3": "~3.2.6",
    "es5-shim": "~2.1.0",
    "jquery": "~1.10.2",
    "sass-bootstrap": "~3.0.2",
    "angular-resource": "1.2.6",
    "angular-cookies": "1.2.6",
    "angular-sanitize": "1.2.6",
    "angular-route": "1.2.6",
    "underscore": "1.6"
  },
  "devDependencies": {
    "angular-mocks": "1.2.6",
    "angular-scenario": "1.2.6"
  }
}
```

The above configuration lists information about the application in the first object block. This can be used for documentation purposes and build information if wanted. The dependencies section list the libraries by name and version that are needed for the application to function at runtime in the browser. The devDependencies section list libraries which are used during testing of the application.

With this configuration file, the developer has a central location that manages the dependencies of the project and this file itself can be managed under any version control system.

Installation

Installation of bower depends upon Node Package Manager (npm). It would be beyond the scope of this paper to cover the installation of node and npm, but my other whitepaper on nodejs does cover this subject.

```
npm install -g bower
```

All dependencies required by bower are installed by npm and the -g flag will make the command available to all users.

General Usage

Bower installs via npm (Node Package module) and provides a command line tool which gives the user a set of useful operations.

- **bower install** - will install any packages listed in the bower.json file
- **bower init** - will create a new scaffolded bower.json file for a project
- **bower search** - will look for a named package in the public repository
- **bower update** - will update any packages installed locally to the latest versions

```

$ bower help

Usage:
  bower <command> [<args>] [<options>]

Commands:
  cache      Manage bower cache
  help       Display help information about Bower
  home       Opens a package homepage into your favorite browser
  info       Info of a particular package
  init       Interactively create a bower.json file
  install    Install a package locally
  link       Symlink a package folder
  list       List local packages
  lookup     Look up a package URL by name
  prune      Removes local extraneous packages
  register   Register a package
  search     Search for a package by name
  update     Update a local package
  uninstall  Remove a local package
  version    Bump a package version

Options:
  -f, --force      Makes various commands more forceful
  -j, --json       Output consumable JSON
  -l, --log-level  What level of logs to report
  -o, --offline    Do not hit the network
  -q, --quiet      Only output important information
  -s, --silent     Do not output anything, besides errors
  -V, --verbose    Makes output more verbose
  --allow-root     Allows running commands as root

See 'bower help <command>' for more information on a specific command.

```

Bower Command Line Options

Build Systems

Grunt.js

Grunt.js is described as a task runner that can automate common build tasks. Many plugins are available and the community is quite large. Plugins range from minification, jshinting, test runners, css compilers to custom plugins which can deploy to cloud providers.

Installation

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

This command will install of the necessary dependencies and will include the grunt command line tool for all users.

Configuration

Grunt relies on a JavaScript based configuration file (Gruntfile.js) to run a set of tasks. Generally tasks all take the same form and are strung together much like Ant or Gradle commands. A configuration file is shown below.

```
module.exports = function(grunt) {

  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    concat: {
      options: {
        separator: ';'
      },
      dist: {
        src: ['src/**/*.js'],
        dest: 'dist/<%= pkg.name %>.js'
      }
    },
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("dd-mm-yyyy") %> */\n'
      },
      dist: {
        files: {
          'dist/<%= pkg.name %>.min.js': ['<%= concat.dist.dest %>']
        }
      }
    },
    qunit: {
      files: ['test/**/*.html']
    },
    jshint: {
      files: ['Gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
      options: {
        // options here to override JSHint defaults
        globals: {
          jQuery: true,
          console: true,
          module: true,
```

```

        document: true
    }
}
},
watch: {
    files: ['<%= jshint.files %>'],
    tasks: ['jshint', 'qunit']
}
});

grunt.registerTask('test', ['jshint', 'qunit']);

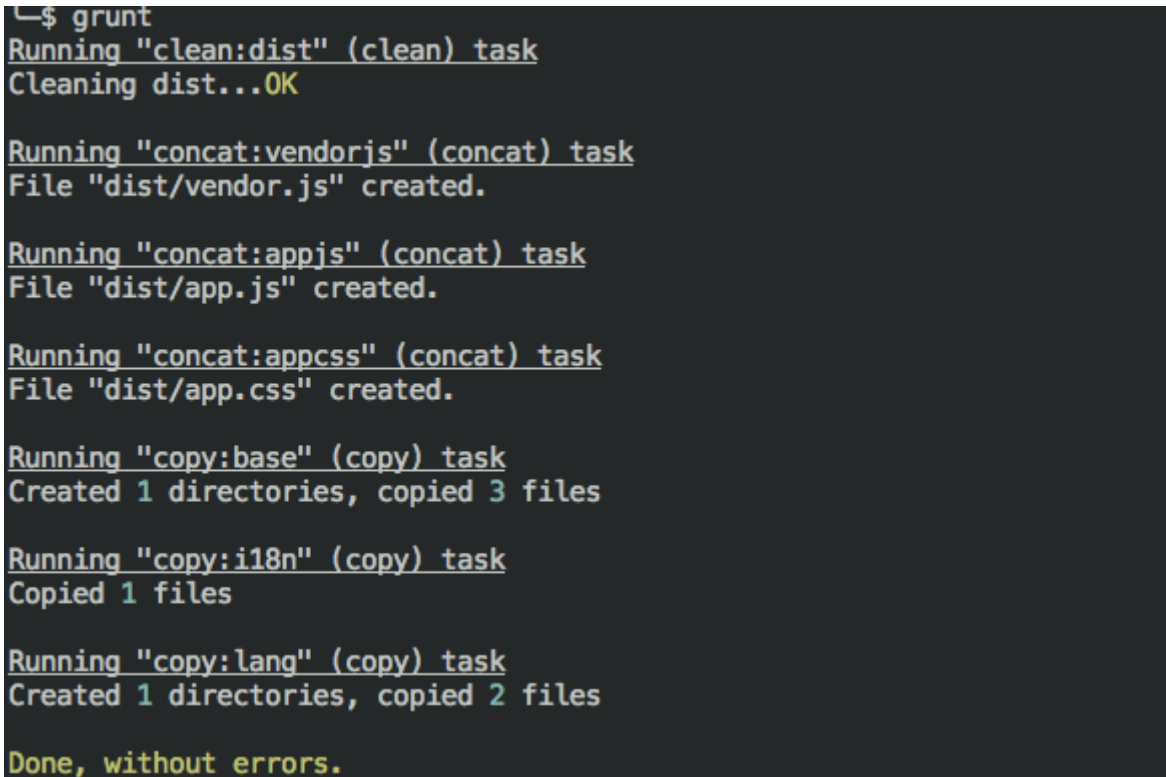
grunt.registerTask('default', ['jshint', 'qunit', 'concat', 'uglify']);

};

```

Command Line View

The Grunt command is quite similar to Ant or Gradle in that it primarily represents whatever tasks are present in the configuration file. Running the command with no option will run the default task; however, running the command with a task name will run only that command and its predecessors.



```

$ grunt
Running "clean:dist" (clean) task
Cleaning dist...OK

Running "concat:vendorjs" (concat) task
File "dist/vendor.js" created.

Running "concat:appjs" (concat) task
File "dist/app.js" created.

Running "concat:appcss" (concat) task
File "dist/app.css" created.

Running "copy:base" (copy) task
Created 1 directories, copied 3 files

Running "copy:i18n" (copy) task
Copied 1 files

Running "copy:lang" (copy) task
Created 1 directories, copied 2 files

Done, without errors.

```

Grunt Command

Gulp.js

Gulp.js is a similar build system to Grunt. It is a newer framework and it has one major difference with Grunt. Gulp.js makes use of Nodejs streams for its entire build system. Rather than using tasks as units of work as Grunt does, Gulp uses the stream technique to pipe one set of files through a pipeline of tasks. This is best illustrated with a look at a gulp configuration file.

Installation

```
npm install -g gulp
```

To install gulp as a dev dependency for your application, run the following command in your project root directory.

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

Configuration

```
// Load plugins
var gulp = require('gulp'),
    sass = require('gulp-ruby-sass'),
    autoprefixer = require('gulp-autoprefixer'),
    minifycss = require('gulp-minify-css'),
    jshint = require('gulp-jshint'),
    uglify = require('gulp-uglify'),
    imagemin = require('gulp-imagemin'),
    rename = require('gulp-rename'),
    clean = require('gulp-clean'),
    concat = require('gulp-concat'),
    notify = require('gulp-notify'),
    cache = require('gulp-cache'),
    livereload = require('gulp-livereload');

// Styles
gulp.task('styles', function() {
    return gulp.src('src/styles/main.scss')
        .pipe(sass({ style: 'expanded', }))
        .pipe(autoprefixer('last 2 version', ''))
        .pipe(gulp.dest('dist/styles'))
        .pipe(rename({ suffix: '.min' }))
        .pipe(minifycss())
        .pipe(gulp.dest('dist/styles'))
        .pipe(notify({ message: 'Styles task complete' }));
});
```

```

// Scripts
gulp.task('scripts', function() {
  return gulp.src('src/scripts/**/*.js')
    .pipe(jshint('.jshintrc'))
    .pipe(jshint.reporter('default'))
    .pipe(concat('main.js'))
    .pipe(gulp.dest('dist/scripts'))
    .pipe(rename({ suffix: '.min' }))
    .pipe(uglify())
    .pipe(gulp.dest('dist/scripts'))
    .pipe(notify({ message: 'Scripts task complete' }));
});

// Clean
gulp.task('clean', function() {
  return gulp.src(['dist/styles', 'dist/scripts', 'dist/images'], {read: false})
    .pipe(clean());
});

// Default task
gulp.task('default', ['clean'], function() {
  gulp.start('styles', 'scripts', 'images');
});

// Watch
gulp.task('watch', function() {

  // Watch .scss files
  gulp.watch('src/styles/**/*.scss', ['styles']);

  // Watch .js files
  gulp.watch('src/scripts/**/*.js', ['scripts']);

  // Watch image files
  gulp.watch('src/images/**/*', ['images']);

  // Create LiveReload server
  var server = livereload();

  // Watch any files in dist/, reload on change
  gulp.watch(['dist/**']).on('change', function(file) {
    server.changed(file.path);
  });

});

```

Based upon the different file sytaxes the reader can determine which system they might prefer.

There is considerable weight behind the Gulp.js system currently with its use of streams and the overall direction of the direction of nodejs with its emphasis on streams as well.

Scaffolding Systems

Yeoman

Yeoman is a tool that is built to automate front-end web workflows. It includes tools which we have already covered (Grunt and Bower) but adds a new tool called Yo. The focus of Yeoman is to scaffold out an application based upon inputs to its build system. This is not unlike the methodology in Ruby On Rails applications.

Yeoman includes several generators written by both the core team and community members and there is a well documented API provided to create new generators.

The framework makes use of these generators to scaffold an application structure for the user. By selecting a few options, a user can create a well documented and well structured front-end web application. Some of the most popular generators are included for AngularJS, Backbone, Polymer, Ember and test suites such as Karma and Jasmine.

Installation

```
npm install -g yo
```

To install any given Yeoman generator run the following command.

```
// basic web app generator
npm install -g generator-webapp
// angular generator
npm install -g generator-angular
```

Typical Usage

The simplest way to use Yeoman would be to generate a scaffold for a new front-end web application. In order to perform this action, the user would run the following commands.


```

$ yo
[?] What would you like to do?
  Run the Gulp-ng generator (1.0.1)
  Run the Gulp-webapp generator (0.1.0)
  Run the Guppy generator (0.2.4)
  Run the Webapp generator (0.4.9)
  Run the Karma generator (0.7.0)
  Run the Mocha generator (0.1.3)
  Update your generators
(Move up and down to reveal more choices)

```

Yeoman Generator Command

```

Make sure you're in the directory you want to scaffold into.
This generator can also be run with: yo webapp

  _____
  |  (o)  |
  |_____|
  |  'U'  |
  |_____|
  |  A    |
  |_____|
  |  ~    |
  |_____|
  |  I    |
  |_____|
  |  Y    |
  |_____|

Welcome to Yeoman,
ladies and gentlemen!

Out of the box I include HTML5 Boilerplate, jQuery, and a Gruntfile.js to build your app.
[?] What more would you like?
  ● Bootstrap
  ● Sass
  > ● Modernizr

```

Yeoman Command Continued

At this point, Yeoman is complete and the user only needs to run bower to install the dependencies.

```

I'm all done. Running bower install for you to install the required dependencies. If this fails, try
and yourself.

bower not-cached  git://github.com/visionmedia/mocha.git#~1.14.0
bower resolve     git://github.com/visionmedia/mocha.git#~1.14.0
bower not-cached  git://github.com/chaijs/chai.git#~1.8.0
bower resolve     git://github.com/chaijs/chai.git#~1.8.0
bower download    https://github.com/chaijs/chai/archive/1.8.1.tar.gz
bower download    https://github.com/visionmedia/mocha/archive/1.14.0.tar.gz
bower extract     mocha#~1.14.0 archive.tar.gz
bower extract     chai#~1.8.0 archive.tar.gz
bower resolved    git://github.com/chaijs/chai.git#1.8.1
bower mismatch    Version declared in the json (1.12.0) is different than the resolved one (1.14.0)
bower resolved    git://github.com/visionmedia/mocha.git#1.14.0
bower install     chai#1.8.1
bower install     mocha#1.14.0

chai#1.8.1 bower_components/chai
mocha#1.14.0 bower_components/mocha

Bye from us! Chat soon.

```

Bower Configuration

Conclusion

This paper has detailed a list of tools that will make the front-end web application developer more efficient in their build processes. These tools are all fairly new, but have very active communities around them and with applications like GulpJS evolving the build tools, this is a space that is worth exploring going forward.

I believe the Yeoman framework with its generators could also be very useful to Nielsen as engineering could create their own set of generators that are specific to their application needs.