GRUNT

Automated Task Management

Common Front End Tasks

- Create/configure a development web server
- Pre-processing (coffee to javascript)
- Versioning
- Linting
- Minification
- Concatenation
- Optimizing assets
- Kicking off Unit or E2E testing
- Deployment

Back End Tasks

- Pre-processing
- Linting
- Testing
- Deployment

GRUNT.JS

- Built on top of Node.js
- Cross-platform (Windows,OSX, LINUX)
- Provides framework of cli & api objects to get up and running quickly.
- Extensible

Things you will need

- Node & npm
- grunt-cli
- A package.json file
- The grunt node module + plug-ins
- Gruntfile.js

Getting started

- Install the grunt client

```
$> sudo npm install -g grunt-cli
```

- Install the grunt module and plug-ins

```
$> npm install –save-dev grunt

$> npm install –save-dev grunt-contrib-xxx
```

package.json

```
{
   “name”: “my-project”,
   “version” : “1.0.0”,
   “devDependencies”: {
      “grunt”: “~0.4.2”,
      “grunt-contrib-jshint”: “~0.6.3”,
      "grunt-c6-util": "git+ssh://
git@github.com:cinema6/c6-grunt-util"
   }
}
```

- Grunt related modules are saved as devDependencies
- You can import your own custom plug-ins with npm + git.

Gruntfile.js

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

  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json),
    uglify: {
        options: {
            banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %> */\n'
        },
        build: {
            src: 'src/<%= pkg.name %>.js',
            dest: 'build/<%= pkg. name %>.min.js'
        }
    }
  });

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

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

};
```

Export Wrapper

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

}
```

- Grunt uses node's require to load the Gruntfile.js
- Gruntfile.js returns a function that receives the grunt object.
- The grunt object === the Grunt API

Configuration

```
grunt.initConfig({
   pkg: grunt.file.readJSON('package.json),
   uglify: {
       options: {
           banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %> */\n'
       },
       build: {
           src: 'src/<%= pkg.name %>.js',
           dest: 'build/<%= pkg. name %>.min.js'
…
```

- initConfig initializes a global configuration object available to all of the grunt tasks.
- configuration can be mix of general configuration + task specific
- grunt.config.get - Supports ruby style string templating
- options can be defaulted for a task and overridden within the task targets

File Handling

- Grunt comes with a lot of helpers for dealing with files
  - Standards for specifying files in configuration
  - Globbing and reg-ex
  - Methods for normalizing lists of files

Loading Tasks

- Loading 3rd party tasks from npm

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

- Loading local tasks

```
grunt.loadTasks('tasks');
```

grunt.registerTask

- Define a default task

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

- Group together (alias) a set of tasks to implement a workflow

```
grunt.registerTask('build', ['jshint','test:unit','uglify']);
```

- Implement new task via code

grunt.registerMultitask

- Define a task with multiple targets
- Run a task with a specific target

```
$> grunt s3:production
```

```
grunt.registerTask('build-prod',['s3:production'])
```

- All tasks will be run if no target specified

Multi Task Configuration

```
s3 :  {
    options: {
        key:    '<%= settings.aws.accessKeyId %>',
        secret: '<%= settings.aws.secretAccessKey %>',
        access: 'public-read'
    },
    test: {
        options: {
            bucket: '<%= settings.s3.test.bucket %>'
            CacheControl: 'max-age=0'
        },
        upload: [
            {
                src: '<%= settings.distDir %>/index.html',
                dest: '<%= settings.s3.test.app %><%= _version %>/index.html',
            },
    },
    production: {
        options: {
            bucket: '<%= settings.s3.production.bucket %>'
            CacheControl: 'max-age=31556926'
        },
        upload: [
            {
                src: '<%= settings.distDir %>/index.html',
                dest: '<%= settings.s3.production.app %><%= _version %>/index.html',
            }
        ]
    }
}
```

Aliasing Task Example

```
grunt.registerTask('build', 'build app into distDir', function(){
    grunt.config('withMaps',grunt.option('with-maps'));
    grunt.task.run('test:unit');
    grunt.task.run('git_last_commit');
    grunt.task.run('clean:build');
    grunt.task.run('copy:dist');
    grunt.task.run('ngtemplates:dist');
    grunt.task.run('htmlmin:dist');
    grunt.task.run('sed:main');
    grunt.task.run('sed:html');
    grunt.task.run('cssmin:dist');
    grunt.task.run('uglify:dist');
    if (grunt.option('with-maps')){
        grunt.task.run('copy:raw');
        grunt.task.run('sed:app_map');
    }
});
```

The Grunt API

- **grunt.config** - getting/setting configuration settings
- **grunt.file** - helpful wrappers around file/io and globbing
- **grunt.log** - logging in color / debug options
- **grunt.option** - share params across tasks, access command line arguments
- **grunt.util** - grab bag of helper functions + used for spawning processes

Inside Task API

- Grunt provides helper properties and methods by decorating the “this” object.
- Inside All Tasks
  - this.async() - returns async done function
  - this.args - array of task args
    - task:arg1:arg2 => this.args === [‘arg1’,’arg2’]
  - this.options([defaults]) - returns task options from configuration
- Inside Multi Tasks
  - this.files - normalized list of files configured through grunt’s Files Array file format
  - this.target - the current target being executed by the multi task
  - this.data - the configuration associated with the multi task target

Plug-ins

- Plugins are node modules containing grunt tasks
  - project & 3rd party contributed plug-ins (>2700)
  - NPM modules (> 68K)
- Where to find them:
  - Grunt.js: gruntjs.com/plugins
  - Github: github.com/gruntjs/grunt-contrib

Plugins for Testing/QA

- grunt-contrib-jshint
    - A helpful wrapper around the jshint linting tool.
- grunt-karma (karma-jasmine,karma-*)
    - Plug-in wrapper for the karma test runner (used by Angular.js)
- grunt-jasmine-node
    - Plug-in wrapper for the jasmine-node cli. Run jasmine BDD tests for your node project

Plugins for Development

- matchdep
    - Node module (not a grunt plugin), helpful for loading lots of grunt tasks
- grunt-contrib-watch
    - Configure the plug-in to watch selected files. Runs tasks when files are added/removed/changed.
- grunt-contrib-connect (connect-livereload)
    - Run your own local web server!
    - In conjunction with watch/livereload your sources can be auto-refreshed when code/html/css changes.
    - With open option - Launches your favorite (or default) web browser.

Plugins for Builds

- grunt-contrib-clean / grunt-contrib-copy
    - Use configuration to specify files/dirs to remove or copy
- grunt-contrib-htmlmin / grunt-contrib-cssmin / grunt-contrib-uglify
    - Minify your html / css / javascript
- grunt-sed
    - Search and replace text in your files based on configuration - useful for versioning your builds

Custom Plugins

- s3 - Uploading files to AWS S3
- versionator - Rename files based on md5 “fingerprint” of the file contents
- git_last_commit - Gets most recent git commit hash for the current project, puts in config
- launch_instances / shutdown_instances - Tasks for launching / starting / stopping / terminating AWS EC2 instances
- e2e_tests - Kick off end-to-end testing using local/remote hosts
- create_snapshots / clear_snapshot - Manage snapshots of AWS disk images