Grunt Part 1

Objectives and Outcomes

In this exercise, you will learn to use Grunt, the task runner. You will install Grunt CLI and install Grunt packages using NPM. Thereafter you will configure a Grunt file with a set of tasks to build and serve your web project. At the end of this exercise, you will be able to:

- Install Grunt CLI and Grunt packages in your project
- Configure a Grunt file with a set of tasks to build a web project from a source, and serve the built project using a server.

Installing Grunt

- At the command prompt, type the following to install Grunt command-line interface (CLI):

```
npm install -g grunt-cli@1.2.0
```

This will install the Grunt CLI globally so that you can use them in all projects.

- Next install Grunt to use within your project. To do this, go to the *conFusion* folder and type the following at the prompt:

```
npm install grunt@1.0.2 --save-dev
```

This will install local per-project Grunt to use within your project.

Creating a Grunt File

- Next you need to create a Grunt file containing the configuration for all the tasks to be run when you use Grunt. To do this, create a file named *Gruntfile.js* in the *conFusion* folder.
- Next, add the following code to Gruntfile.js to set up the file to configure Grunt tasks:

```
'use strict';
module.exports = function (grunt) {
 // Define the configuration for all the tasks
 grunt.initConfig({
 });
};
```

This sets up the Grunt module ready for including the grunt tasks inside the function above.

Compiling SCSS to CSS

- Next, we are going to set up our first Grunt task. The SASS task converts the SCSS code to CSS. To do this, you need to include some Grunt modules that help us with the tasks. Install the following modules by typing the following at the prompt:

```
npm install grunt-sass@2.1.0 --save-dev
npm install time-grunt@1.4.0 --save-dev
npm install jit-grunt@0.10.0 --save-dev
```

The first one installs the Grunt module for SCSS to CSS conversion. The time-grunt module generates time statistics about how much time each task consumes, and jit-grunt enables us to include the necessary downloaded Grunt modules when needed for the tasks.

- Now, configure the SASS task in the Gruntfile as follows, by including the code inside the function in *Gruntfile.js*:

```
'use strict';
module.exports = function (grunt) {
   // Time how long tasks take. Can help when optimizing build times
   require('time-grunt')(grunt);
   // Automatically load required Grunt tasks
   require('jit-grunt')(grunt);
   // Define the configuration for all the tasks
   grunt.initConfig({
       sass: {
           dist: {
               files: {
                   'css/styles.css': 'css/styles.scss'
               }
           }
       }
   });
   grunt.registerTask('css', ['sass']);
};
```

- Now you can run the grunt SASS task by typing the following at the prompt:

```
grunt css
```

Watch and Serve Tasks

- The final step is to use the Grunt modules watch and browser-sync to spin up a web server and keep a watch on the files and automatically reload the browser when any of the watched files are updated. To do this, install the following grunt modules:

```
npm install grunt-contrib-watch@1.0.0 --save-dev
npm install grunt-browser-sync@2.2.0 --save-dev
```

- After this, we will configure the browser-sync and watch tasks by adding the following code to the Grunt file:

```
,
       watch: {
           files: 'css/*.scss',
           tasks: ['sass']
       },
       browserSync: {
           dev: {
               bsFiles: {
                   src : [
                       'css/*.css',
                       '*.html',
                       'js/*.js'
                   ]
               },
               options: {
                   watchTask: true,
                   server: {
                       baseDir: "./"
                   }
               }
           }
       }
```

- Then add the following task to the Grunt file:

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

- Now if you type the following at the command prompt, it will start the server, and open the web page in your default browser. It will also keep a watch on the files in the css folder, and if you update any of them, it will compile the scss file into css file and load the updated page into the browser (livereload)

```
grunt
```

Grunt Part 2

Objectives and Outcomes

In this exercise, you will continue to learn to use Grunt, the task runner. You will configure the Grunt file with a set of additional tasks to build your web project. At the end of this exercise, you will be able to:

- Configure a Grunt file with a set of tasks to build your web project from a source.

Copying the Files and Cleaning Up the Dist Folder

- Next you will install the Grunt modules to copy over files to a distribution folder named dist, and clean up the dist folder when needed. To do this, install the following Grunt modules:

```
npm install grunt-contrib-copy@1.0.0 --save-dev
npm install grunt-contrib-clean@1.1.0 --save-dev
```

- You will now add the code to perform the copying of files to the dist folder, and cleaning up the dist folder. To do this, add the following code to *Gruntfile.js*. This should be added right after the configuration of the SASS task.:

```
,
       copy: {
           html: {
               files: [
               {
                   //for html
                   expand: true,
                   dot: true,
                   cwd: './',
                   src: ['*.html'],
                   dest: 'dist'
               }]
           },
           fonts: {
               files: [
               {
                   //for font-awesome
```

```
                    expand: true,
                    dot: true,
                    cwd: 'node_modules/font-awesome',
                    src: ['fonts/*.*'],
                    dest: 'dist'
                }]
            }
        },
        clean: {
            build: {
                src: [ 'dist/']
            }
        }
```

- Remember to add the comma after the end of the SASS task.

Compressing and Minifying Images

- Next we install the grunt-contrib-imagemin module and use it to process the images. To install this module type at the prompt:

```
npm install grunt-contrib-imagemin@2.0.1 --save-dev
```

- Then, configure the imagemin task as shown below in the Gruntfile:

```
,
        imagemin: {
            dynamic: {
                files: [{
                    expand: true,                  // Enable dynamic expansion
                    cwd: './',                   // Src matches are relative to
                       this path
                    src: ['img/*.{png,jpg,gif}'],   // Actual patterns to match
                    dest: 'dist/'                  // Destination path prefix
                }]
            }
        }
```

Preparing the Distribution Folder and Files

- We are now going to use the Grunt *usemin* module together with *concat*, *cssmin*, *uglify* and *filerev* to prepare the distribution folder. To do this, install the following Grunt modules:

```
npm install grunt-contrib-concat@1.0.1 --save-dev
npm install grunt-contrib-cssmin@2.2.1 --save-dev
npm install grunt-contrib-htmlmin@2.4.0 --save-dev
npm install grunt-contrib-uglify@3.3.0 --save-dev
npm install grunt-filerev@2.3.1 --save-dev
npm install grunt-usemin@3.1.1 --save-dev
```

- Next, update the task configuration within the Gruntfile.js with the following additional code to introduce the new tasks:

```
,
       useminPrepare: {
           foo: {
               dest: 'dist',
               src: ['contactus.html','aboutus.html','index.html']
           },
           options: {
               flow: {
                   steps: {
                       css: ['cssmin'],
                       js:['uglify']
                   },
                   post: {
                       css: [{
                           name: 'cssmin',
                           createConfig: function (context, block) {
                           var generated = context.options.generated;
                               generated.options = {
                                   keepSpecialComments: 0, rebase: false
                               };
                           }
                       }]
                   }
               }
           }
       },
       // Concat
       concat: {
           options: {
               separator: ';'
           },
            // dist configuration is provided by useminPrepare
           dist: {}
       },
       // Uglify
       uglify: {
           // dist configuration is provided by useminPrepare
           dist: {}
       },
       cssmin: {
           dist: {}
       },
       // Filerev
       filerev: {
           options: {
               encoding: 'utf8',
               algorithm: 'md5',
               length: 20
           },
            release: {
           // filerev:release hashes(md5) all assets (images, js and css )
           // in dist directory
               files: [{
                   src: [
                       'dist/js/*.js',
                       'dist/css/*.css',
                   ]
               }]
           }
       },
        // Usemin
       // Replaces all assets with their revved version in html and css files.
       // options.assetDirs contains the directories for finding the assets
       // according to their relative paths
       usemin: {
           html: ['dist/contactus.html','dist/aboutus.html','dist/index.html'],
           options: {
               assetsDirs: ['dist', 'dist/css','dist/js']
           }
       },
       htmlmin: {                                         // Task
           dist: {                                        // Target
               options: {                                 // Target options
                   collapseWhitespace: true
               },
               files: {                                   // Dictionary of
                  files
                   'dist/index.html': 'dist/index.html',  // 'destination':
                      'source'
                   'dist/contactus.html': 'dist/contactus.html',
                   'dist/aboutus.html': 'dist/aboutus.html',
               }
           }
       }
```

- Next, update the jit-grunt configuration as follows, to inform it that useminPrepare task depends on the usemin package:

```
require('jit-grunt')(grunt, {
 useminPrepare: 'grunt-usemin'
});
```

- Next, update the Grunt build task as follows:

```
   grunt.registerTask('build', [
       'clean',
       'copy',
       'imagemin',
       'useminPrepare',
       'concat',
       'cssmin',
       'uglify',
       'filerev',
       'usemin',
       'htmlmin'
   ]);
```

- Now if you run Grunt, it will create a dist folder with the files structured correctly to be distributed to a server to host your website. To do this, type the following at the prompt:

```
grunt build
```

Conclusions

In this exercise you have learnt how to configure a Grunt file to perform several tasks. You were able to build a distribution folder for your web project.