Gulp Part 1

Objectives and Outcomes

In this exercise, you will learn to use Gulp, the task runner. You will install Gulp CLI and install Gulp plugins using NPM. Thereafter you will configure a Gulp 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 Gulp CLI and Gulp plugins in your project
- Configure a Gulp file with a set of tasks to build a web project from a source, and serve the built project using a server.

Installing Gulp

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

```
npm install -g gulp-cli@2.0.1
```

This will install the Gulp globally so that you can use it in all projects.

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

```
npm install gulp@3.9.1 --save-dev
```

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

Install Gulp Plugins for SASS and Browser-Sync

- Install all the Gulp plugins that you will need for this exercise. To do this, type the following at the command prompt:

```
npm install gulp-sass@3.1.0 browser-sync@2.23.6 --save-dev
```

Creating a Gulp File

- Next you need to create a Gulp file containing the tasks to be run when you use Gulp. To do this, create a file named *gulpfile.js* in the *conFusion* folder.

Loading Gulp Plugins

- Load in all the Gulp plugins by including the following code in the Gulp file:

```
'use strict';
var gulp = require('gulp'),
   sass = require('gulp-sass'),
   browserSync = require('browser-sync');
```

Adding Gulp Tasks for SASS and Browser-Sync

- Next, we will add the code for the SASS task, the Browser-Sync task and the default task as follows:

```
gulp.task('sass', function () {
 return gulp.src('./css/*.scss')
   .pipe(sass().on('error', sass.logError))
   .pipe(gulp.dest('./css'));
});
gulp.task('sass:watch', function () {
 gulp.watch('./css/*.scss', ['sass']);
});
gulp.task('browser-sync', function () {
  var files = [
     './*.html',
     './css/*.css',
     './img/*.{png,jpg,gif}',
     './js/*.js'
  ];
  browserSync.init(files, {
     server: {
        baseDir: "./"
     }
  });
});
// Default task
gulp.task('default', ['browser-sync'], function() {
   gulp.start('sass:watch');
});
```

- Save the Gulp file

Running the Gulp Tasks

- At the command prompt, if you type *gulp* it will run the default task:

```
    gulp
```

- Do a Git commit with the message "Gulp Part 1".

Conclusions

In this exercise, you learnt to use Gulp, install Gulp plugins, configure the gulpfile.js and then use Gulp to automate the web development tasks.

Gulp Part 2

Objectives and Outcomes

In this exercise, you will continue to learn to use Gulp. Thereafter you will configure a Gulp file with a set of tasks to build and serve your web project. At the end of this exercise, you will be able to:

- Configure the Gulp file with a set of tasks to build the distribution folder for the web project.

Copying the Files and Cleaning up the Dist Folder

- We will now create the tasks for copying the font files and cleaning up the distribution folder. To do this we will first install the *del* Node module and require it in the Gulp file as follows:

```
npm install del@3.0.0 --save-dev
```

```
var ...
    del = require('del'),
    …
```

- Next, we will add the code for the Clean task and the copyfonts task as follows:

```
// Clean
gulp.task('clean', function() {
    return del(['dist']);
});
gulp.task('copyfonts', function() {
  gulp.src('./node_modules/font-awesome/fonts/**/*.{ttf,woff,eof,svg}*')
  .pipe(gulp.dest('./dist/fonts'));
});
```

Compressing and Minifying Images

- We will now install the *gulp-imagemin* plugin and configure the *imagemin* task. To do this we install the plugin and require it as follows:

```
npm install gulp-imagemin@4.1.0 --save-dev
var ...
    imagemin = require('gulp-imagemin'),
    ...
```

- Next, we create the *imagemin* task as follows:

```
// Images
gulp.task('imagemin', function() {
 return gulp.src('img/*.{png,jpg,gif}')
    .pipe(imagemin({ optimizationLevel: 3, progressive: true, interlaced: true }
        ))
    .pipe(gulp.dest('dist/img'));
});
```

Preparing the Distribution Folder and Files

- We now install the gulp-usemin and other related Gulp plugins and require them as follows:

```
npm install gulp-uglify@3.0.0 gulp-usemin@0.3.29 gulp-rev@8.1.1 gulp-clean-css@3
  .9.3 gulp-flatmap@1.0.2 gulp-htmlmin@4.0.0 --save-dev

var ...
    uglify = require('gulp-uglify'),
    usemin = require('gulp-usemin'),
    rev = require('gulp-rev'),
    cleanCss = require('gulp-clean-css'),
    flatmap = require('gulp-flatmap'),
    htmlmin = require('gulp-htmlmin');
```

- We configure the usemin and the build task as follows:

```
gulp.task('usemin', function() {
 return gulp.src('./*.html')
 .pipe(flatmap(function(stream, file){
     return stream
        .pipe(usemin({
            css: [ rev() ],
            html: [ function() { return htmlmin({ collapseWhitespace: true })} ]
               ,
            js: [ uglify(), rev() ],
            inlinejs: [ uglify() ],
            inlinecss: [ cleanCss(), 'concat' ]
        }))
   }))
   .pipe(gulp.dest('dist/'));
});
gulp.task('build',['clean'], function() {
   gulp.start('copyfonts','imagemin','usemin');
});
```

- Save the Gulp file

Running the Gulp Tasks

- At the command prompt, if you type *gulp build* it will run the build task:

```
    gulp build
```

- Do a Git commit with the message "Gulp Part 2"

Conclusions

In this exercise, you learnt to use Gulp, install Gulp plugins, configure the gulpfile.js and then use Gulp to automate the web development tasks.