3.2 Grunt Plugins

This section will guide you to:

* Work with Grunt Plugins

This lab has ten subsections, namely:

3.2.1 Configuring tasks on an example using Autoprefixer

3.2.2 Watching for changes: contrib-watch

3.2.3 Automatically compressing CSS files: contrib-cssmin

3.2.4 Automatically compressing and merging JS files: grunt-contrib-uglify

3.2.5 Loading multiple grunt tasks at once

3.2.6 Image Optimization

3.2.7 Compressing images only if a new one is added: grunt-newer

3.2.8 JShint: finding bugs in your Javascript code instantly

3.2.9 Running multiple tasks at once and improving the speed: grunt-concurrent

3.2.10 Pushing the code to your GitHub repositories

**Step 3.2.1:**  Configuring tasks on an example using Autoprefixer

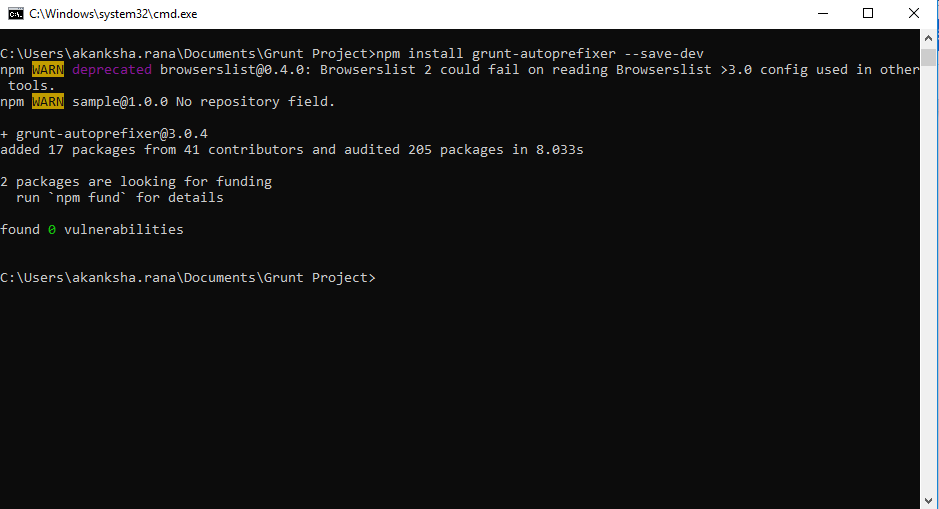
We will configure a task to add a browser as the default supported browser using autoprefixer plugin.

Before you can configure tasks, you will have to install the Autoprefixer plugin. Use the following steps to install Autoprefixer:

* Open the command-line interface and type in the commands:

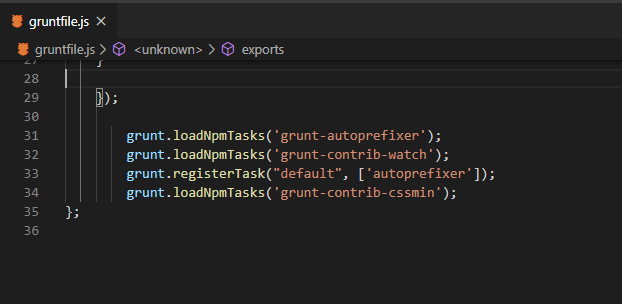
**Cd <path of your project folder>**

**npm install grunt-autoprefixer --save-dev**



* To confirm the autoprefixer installation, check and make sure that autoprefixer version is mentioned in the package.json file.
* To enable the autoprefixer plugin, copy and paste the following code line in the gruntfile.js as shown in the following screenshot:

**grunt.loadNpmTasks('grunt-autoprefixer');**

****

* Configure autoprefixer by using the following code snippet. Copy and paste it in grunt.initCongif() method as shown in the screenshot below:

**autoprefixer: {**

**options: {**

**// Task-specific options go here.**

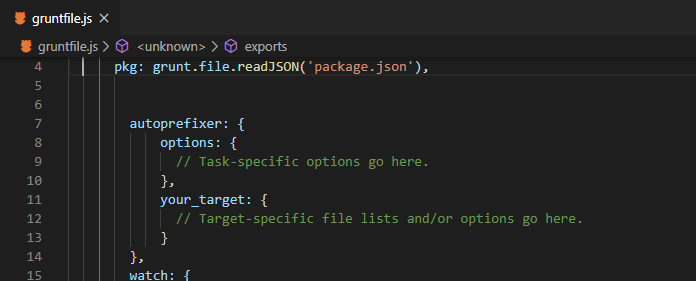
**},**

**your\_target: {**

**// Target-specific file lists and/or options go here.**

**}**

**}**



* We will use the options attribute in the **gruntile.js** file to add Mozilla Firefox browser in the list of supported browsers. Copy and paste the following code snippet in options attribute in the gruntfile.js:

**browsers: ['last 2 versions', 'firefox > 3']**

* Open styleprefixed.css file. You should see moz (Mozilla Firefox) browser mentioned along with default browsers

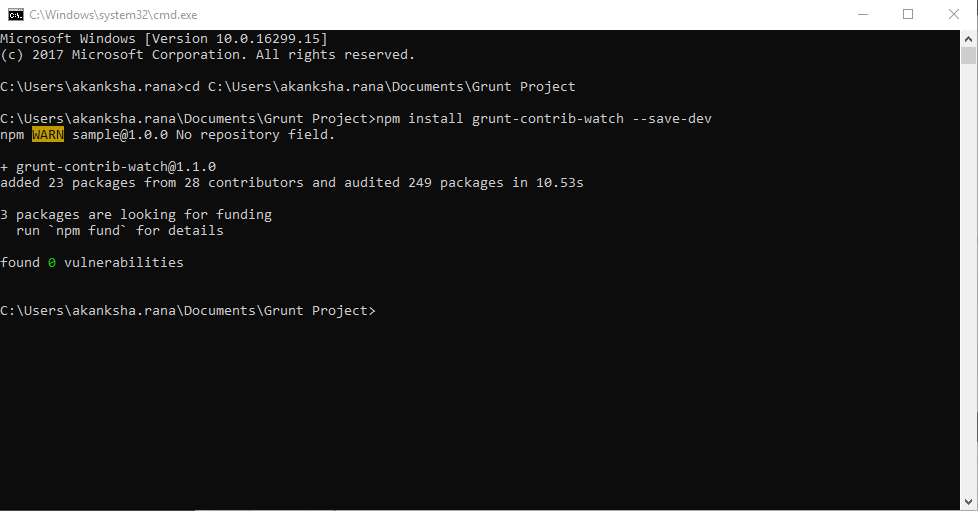
**Step 3.2.2:** Watching for Changes: contrib-watch

We will use **grunt contrib-watch** plugin to watch all the changes being made to any of the files in our project. We will also configure a task to replicate the changes to other files using this plugin.

* Use the following steps to install **grunt contrib-watch** plugin:
  + Open the command-line interface and type in the commands:

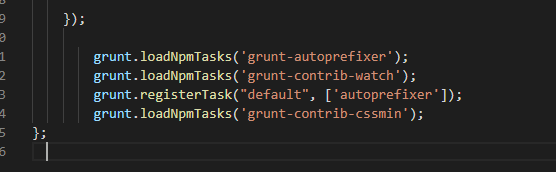
**Cd <path of your project folder>**

**npm install grunt-contrib-watch --save-dev**



* To verify grunt-contrib-watch installation, check and make sure that the contrib-watch version is mentioned in the **package.json** file.
* To enable the plugin, you will have to load it using the following code line. Copy and paste it in gruntfile.js as shown in the screenshot below:

**grunt.loadNpmTasks('grunt-contrib-watch’);**

****

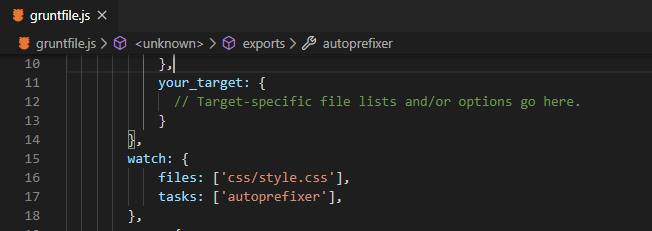
* Add the following code snippet in the gruntfile.js as shown in the screenshot below:

**watch:**

**{ files: ['css/style.css'],**

**tasks: [‘autoprefixer’]**

**}**



This code snippet specifies that **style.css** file will be watched for changes and if any changes are made to the specified file, then autoprefixer task will be executed.

* Add the following code snippet in the autoprefixer variable in the gruntfile.js file as shown in the screenshot below:

**copy: {**

**src: 'css/style.css',**

**dest: 'css/styleprefixed.css'**

**}**

* Register the task using the following code line:

**grunt.registerTask(“default”, ['autoprefixer']);**

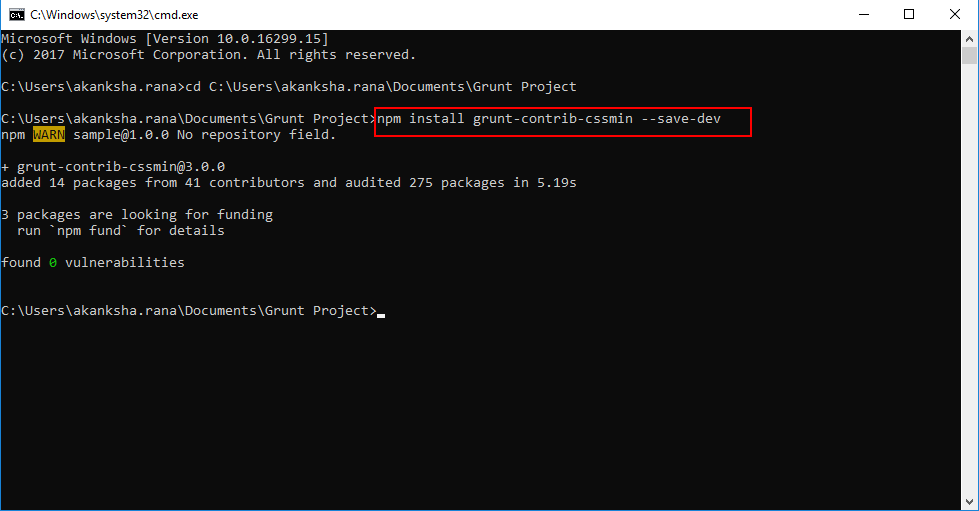
**Step 3.2.3:** Automatically compressing CSS files: contrib-cssmin

We will use **grunt contrib-cssmin** plugin to compress our css files. We will configure a cssmin task to compress any css file if there are any changes made in that file.

* Use the following steps to install grunt contrib-cssmin plugin:
  + Open the command-line interface and type in the commands:

**Cd <path of your project folder>**

**npm install grunt-contrib-cssmin --save-dev**

****

* To verify grunt-contrib-cssmin installation, check and make sure that the contrib-cssmin version is mentioned in the **package.json** file.
* To enable the plugin, you need to load it in the gruntfile.js file using the following code line:

**grunt.loadNpmTasks('grunt-contrib-cssmin’);**

* Copy and paste the following code snippet in the gruntfile.js:

**cssmin:**

**{**

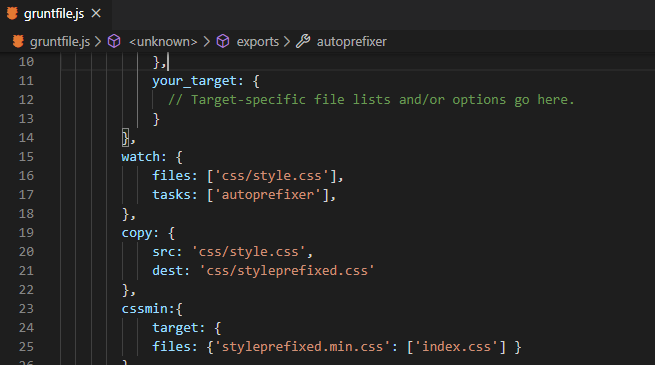
**target: {**

**files: {**

**'css/styleprefixed.min.css': ['css/styleprefixed.css'] }**

**}**

**}**



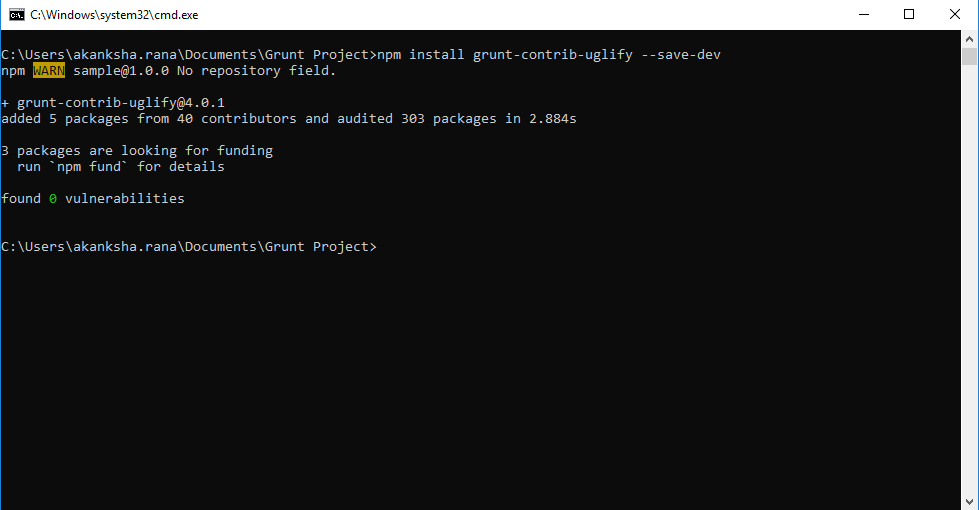
**Step 3.2.4:** Automatically compressing and merging JS files( grunt-contrib-uglify)

We will use grunt contrib-uglify plugin to compress js files in our project.

* Use the following steps to install grunt contrib-uglify plugin:
  + Open command-line interface and type in the commands:

**Cd <path of your project folder>**

**npm install grunt-contrib-uglify --save-dev**

****

* To verify grunt-contrib-uglify installation, check and make sure that the contrib-uglify version is mentioned in the **package.json** file.
* To enable the plugin, you need to load it in the gruntfile.js file using the following code line:

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

* Create a folder named js in your project folder. Create a new file inside the js folder and paste the following code snippet in the file. Save the file as script.js.

**$("document").ready(function(){**

**alert("test from jquery as a sdas");**

**});**

* Copy and paste the following code snippet in the gruntfile.js as shown in the screenshot below:

**uglify:**

**{**

**target: {**

**files: {**

**'js/output.min.js': ['js/\*.js'] }**

**}**

**}**

**Step 3.2.5:** Load multiple grunt tasks at once

We can use load-grunt-tasks plugin to load multiple grunt tasks or plugins at the same time.

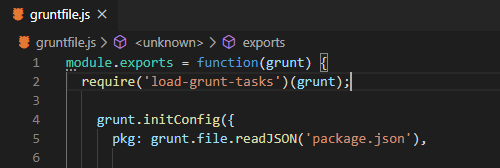
* Use the following steps to install load-grunt-tasks plugin:
  + Open command-line interface and type in the commands:

**cd <path of your project folder>**

**npm install --save-dev load-grunt-tasks**

* To verify the installation, check and make sure that the version for load-grunt-tasks plugin is mentioned in the package.json file.
* Copy and paste the following code line in gruntfile.js file as shown in the screenshot below:

**require('load-grunt-tasks')(grunt);**

****

* Comment out all the loadNpmTasks() methods.
* Verify whether grunt tasks are still available or not. Right click on gruntfile.js file on the left pane and click on grunt tasks.

**Step 3.2.6:** Image Optimization

* Use the following steps to Install the imagemin plugin:
  + Open the command-line interface and type in the command:

**cd <path of your project folder>**

**npm install grunt-contrib-imagemin --save-dev**

* Create a new folder named **images** in the root project folder. Save some images in the images folder.
* Copy and paste the following code snippet in the gruntfile.js file as shown in the screenshot below:

**imagemin:**

**{**

**dynamic:**

**{**

**files: [**

**{**

**expand: true,**

**cwd: 'images/',**

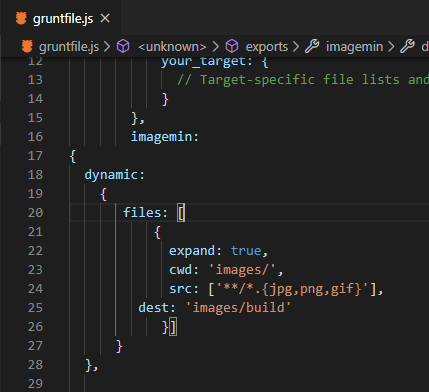
**src: ['\*\*/\*.{jpg,png,gif}'],**

**dest: 'images/build'**

**}]**

**}**

**}**



* Right click on the gruntfile.js file on the left pane and click on grunt tasks > imagemin.
* This task will run without any errors and compressed images will be stored in the build folder.

**Step 3.2.7:** Compressing images only if a new one is added (grunt-newer)

We will use grunt newer plugin to compress a newly added image. Grunt newer plugin will automatically pick up the new image and skip the images that have been already compressed.

* Use the following steps to install the grunt newer plugin:
  + Open the command-line interface and type in the commands:

**cd <path of your project folder>**

**npm install grunt-newer --save-dev**

* Copy and paste the following code snippet in the gruntfile.js file:

**grunt.registerTask("minifyNewImages", 'newer:imagemin');**

* Add a new image in the images folder. Right click on the gruntfile.js file in the left pane and click on grunt tasks > minifyNewImages. The task will run without any errors and only the newly added images will be compressed.

**Step 3.2.8:** JShint: finding bugs in your Javascript code instantly

We will use grunt contrib-jshinit plugin to find bugs in the javascript code

* Use the following steps to install the grunt-contrib-jshinit plugin
  + Open the command-line interface and type in the commands:

**cd <path of your project folder>**

**npm install grunt-contrib-jshinit --save-dev**

* Copy and paste the following code snippet in the gruntfile.js file as shown in the below screenshot:

**jshint:**

**{**

**options: {**

**"bitwise": true,**

**"camelcase": true,**

**"curly": true,**

**"latedef": true,**

**"newcap": true,**

**"nonew": true,**

**"undef": true,**

**"unused": true,**

**"esnext": true,**

**"sub": true,**

**"browser": true,**

**"node": true,**

**"jquery": true,**

**"devel": true,**

**"strict": true**

**},**

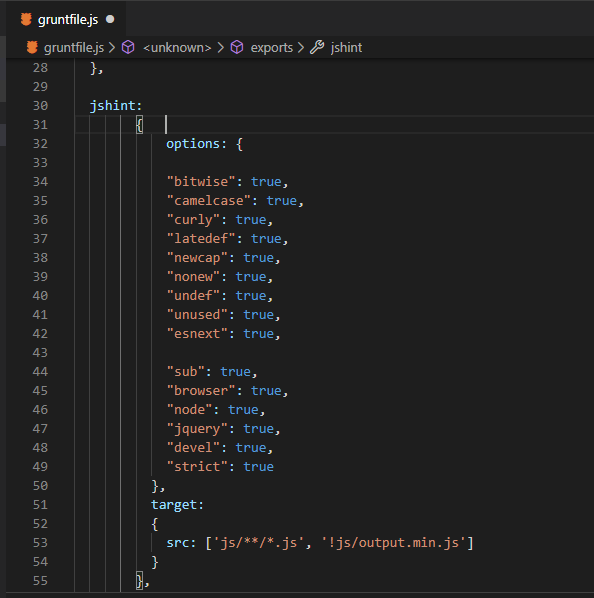
**target:**

**{**

**src: ['js/\*\*/\*.js', '!js/output.min.js']**

**}**

**}**

****

This code snippet will give suggestions about the bugs in all the files of .js extension residing in the js folder.

* Right click on the gruntfile.js file on the left pane and click on grunt tasks > jshint

**Step 3.2.9:** Running multiple tasks at once and improving the speed (grunt-concurrent)

* Use the following steps to install the grunt-concurrent plugin:
  + Open the command-line interface and type in the commands:

**cd <path of your project folder>**

**npm install --save-dev grunt-concurrent**

* Copy and paste the following code snippet in the gruntfile.js file as shown in the screenshot below:

**concurrent:**

**{**

**target:**

**{**

**tasks:["watch", "autoprefixer", "imagemin"],**

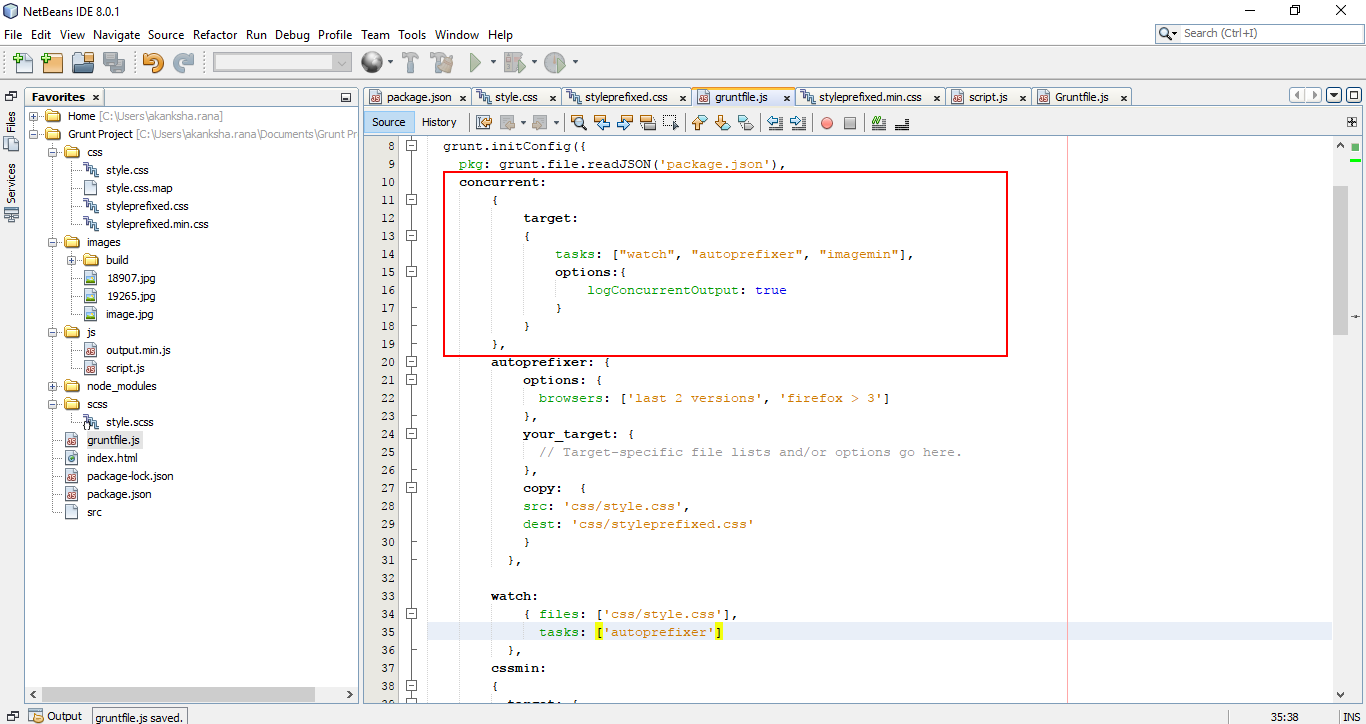
**options:{**

**logConcurrentOutput: true**

**}**

**}**

**},**

****

This code snippet will run watch, autoprefixer, and imagemin tasks concurrently.

* To run these tasks, right click on the gruntfile.js in the left pane and click on grunt tasks > concurrent.

**Step 3.2.10:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

Initialize your repository using the following command:

**git init**

Add all the files to your git repository using the following command:

**git add .**

Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

Push the files to the folder you created initially using the following command:

**git push -u origin master**