



Articles » Web Development » ASP.NET » General

How to Use Gulp with ASP.NET Core MVC



saineshwar bageri, 10 Jan 2017

In this article, we are going to learn how to use Gulp.js to minify & uglify (JavaScript, cascading style sheets and HTML) and to configure Task Runner Explorer in ASP.NET Core MVC.

Download source and demo - 4.7 MB

In this article, we are going to learn how to use Gulp.js to minify & uglify (JavaScript, cascading style sheets and HTML) and to configure Task Runner Explorer in ASP.NET Core MVC.

The first question will arrive in mind is what is Gulp.js?

Gulp.js is an open source JavaScript toolkit for the automating task in the development process.

In our daily projects we develop lots of JavaScript, Cascading style sheet, HTML files, but to minify or beautify this file we go to some website there we upload files to minify or beautify files.

If we can automate this process in Visual Studio, what would be your reaction? "Wow," right?

Let's learn how to automate this process using gulp.js and enjoy.

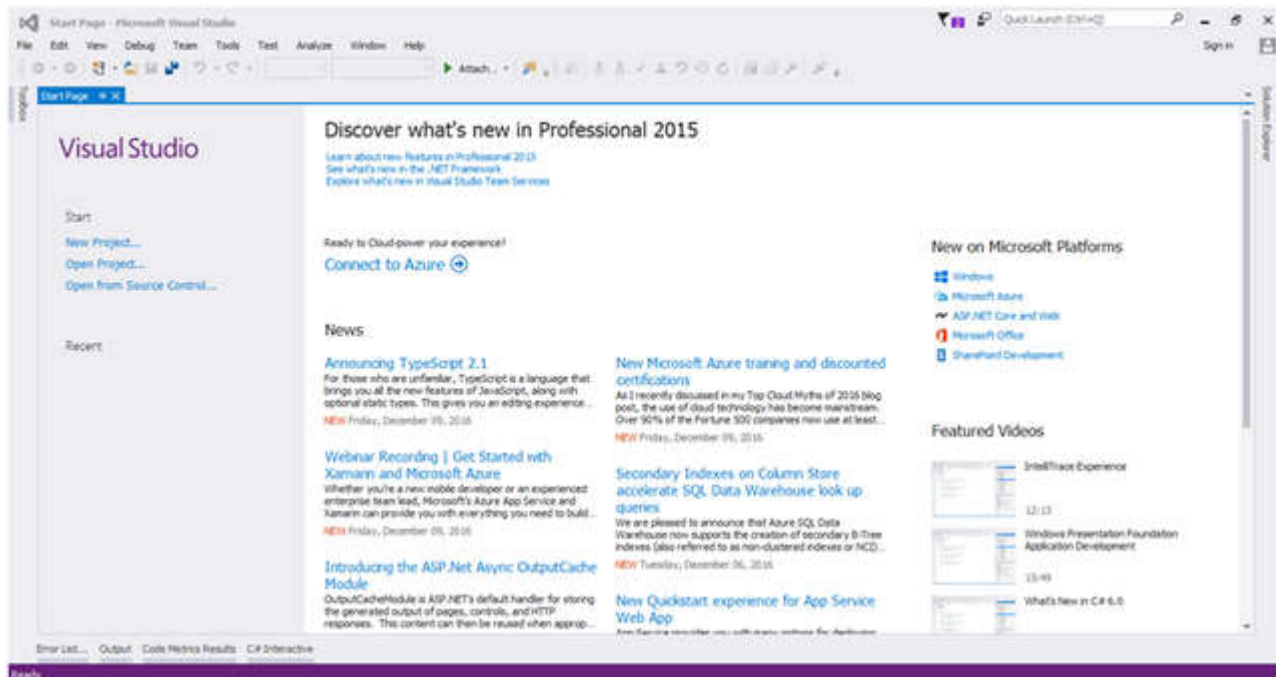


Topic

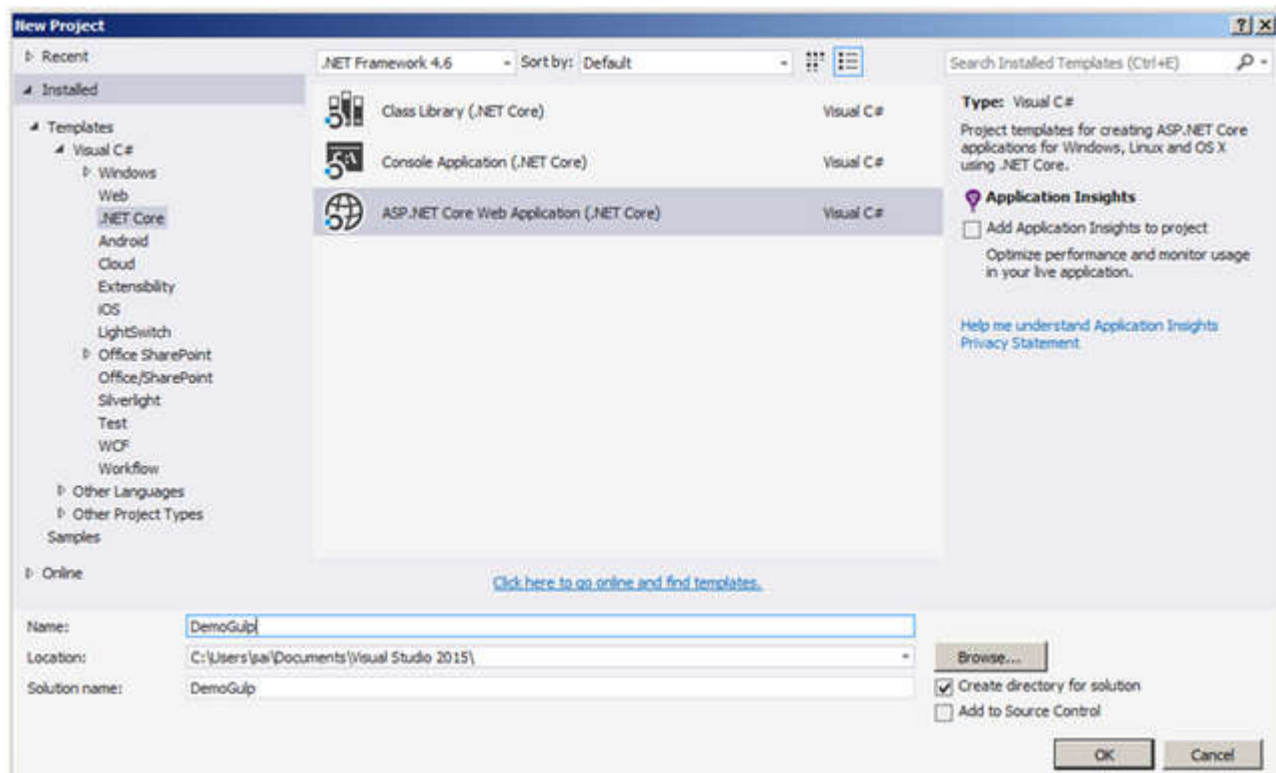
1. Creating application
2. Adding *package.json* file in the project
3. Downloading Dependencies of gulp.js
4. Adding *gulp.js* file in project
5. Writing basic task to learn how to use start using gulp.js
6. Installing "Task Runner Explorer" Extension in Visual studio
7. ; Adding HTML file to Project and writing task to minifying Html file
8. Writing task for Concatenate file JavaScript files
9. Writing task to minify JavaScript
10. Writing task to minify Cascading style sheets
11. Writing task to Change file extension of JavaScript files and Cascading style sheet
12. Automating Task

Step 1: Creating application

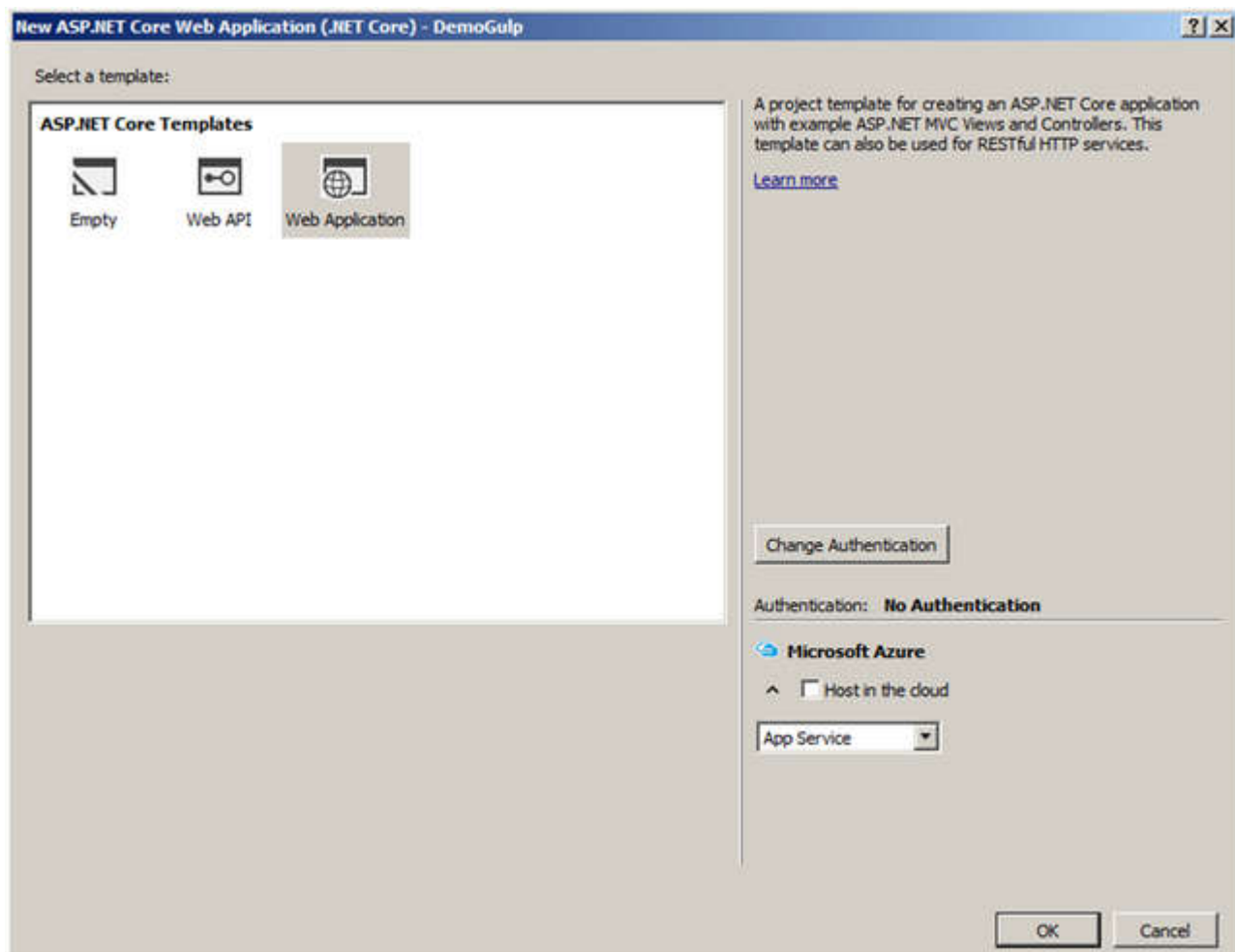
Now let's create a .NET Core Web Application Open Visual Studio IDE from start page Click on New Project link



After clicking on New Project link it will open a new dialog with Name "**New Project**" inside that from left panel choose templates inside that -> choose Visual C# -> inside that choose .NET Core template then in middle you pane you will see .NET Core project templates, from templates that choose "**ASP.NET Core Web Application (.NET Core)**" project templates.

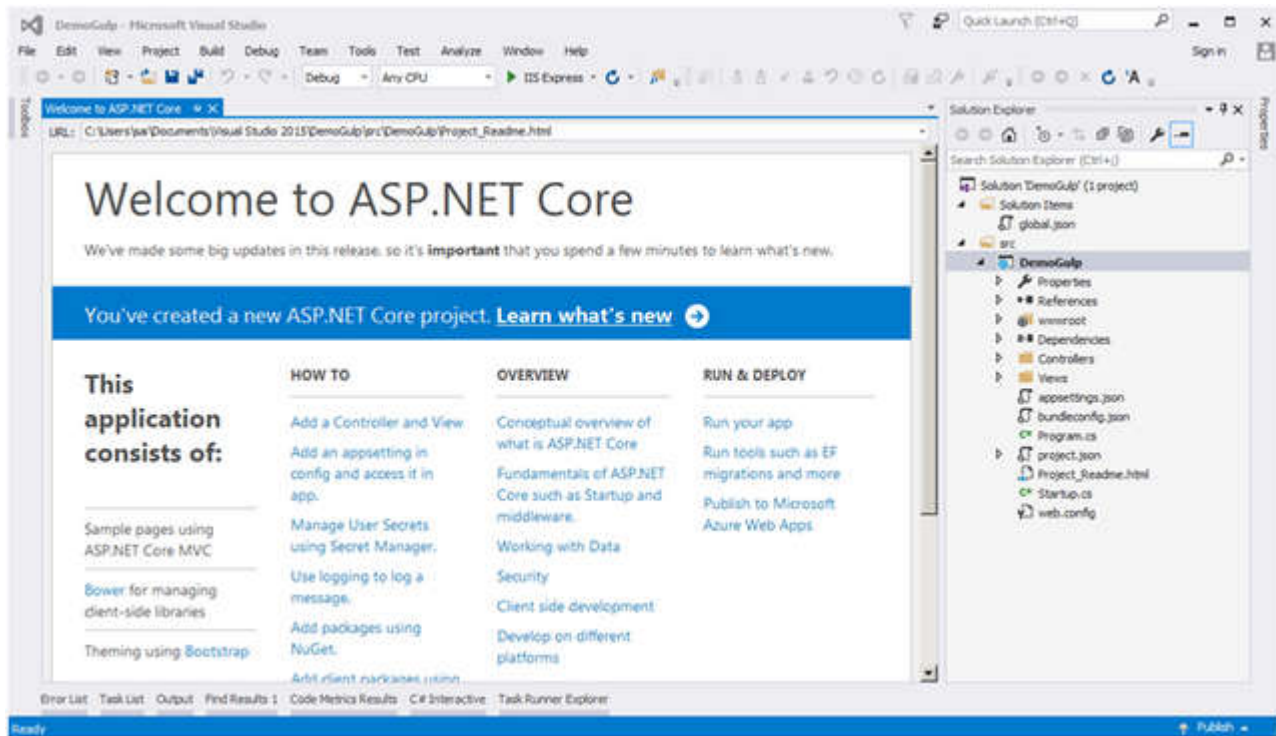


After choosing project template next we are going to name the project as "**DemoGulp**" and finally click on ok button to create a project, but it will pop up another dialog with name "**New ASP.NET Core Web Application (.NET Core)**".



Inside this dialog, we are going to choose a **"Web Application"** project template for creating **"ASP.NET Core Web Application."** Click on the OK button to create a project.

Below is a complete project view which is newly created.

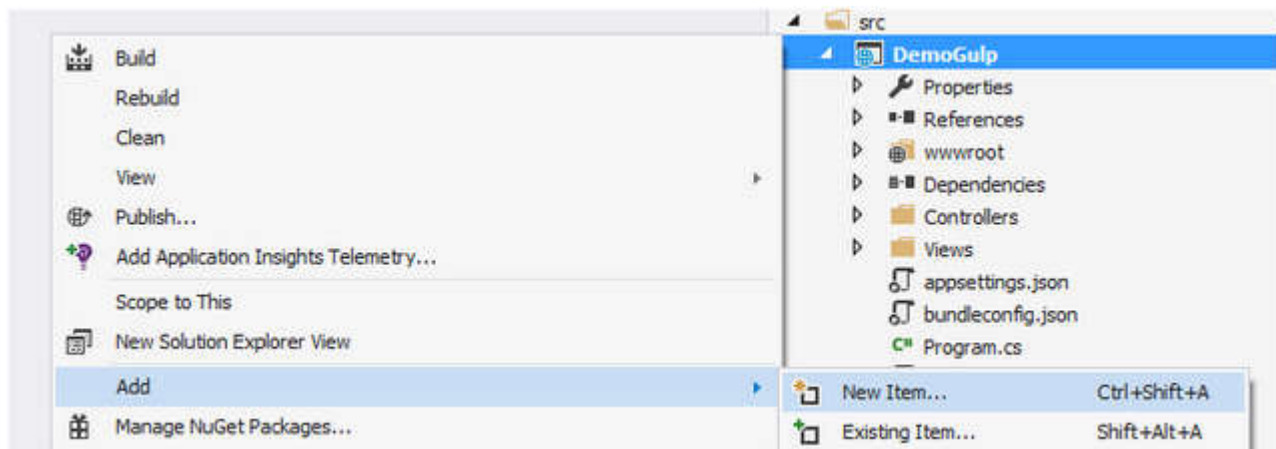


After creating a project, the next thing is to add *package.json* file to project.

Step 2: Adding package.json file in the project

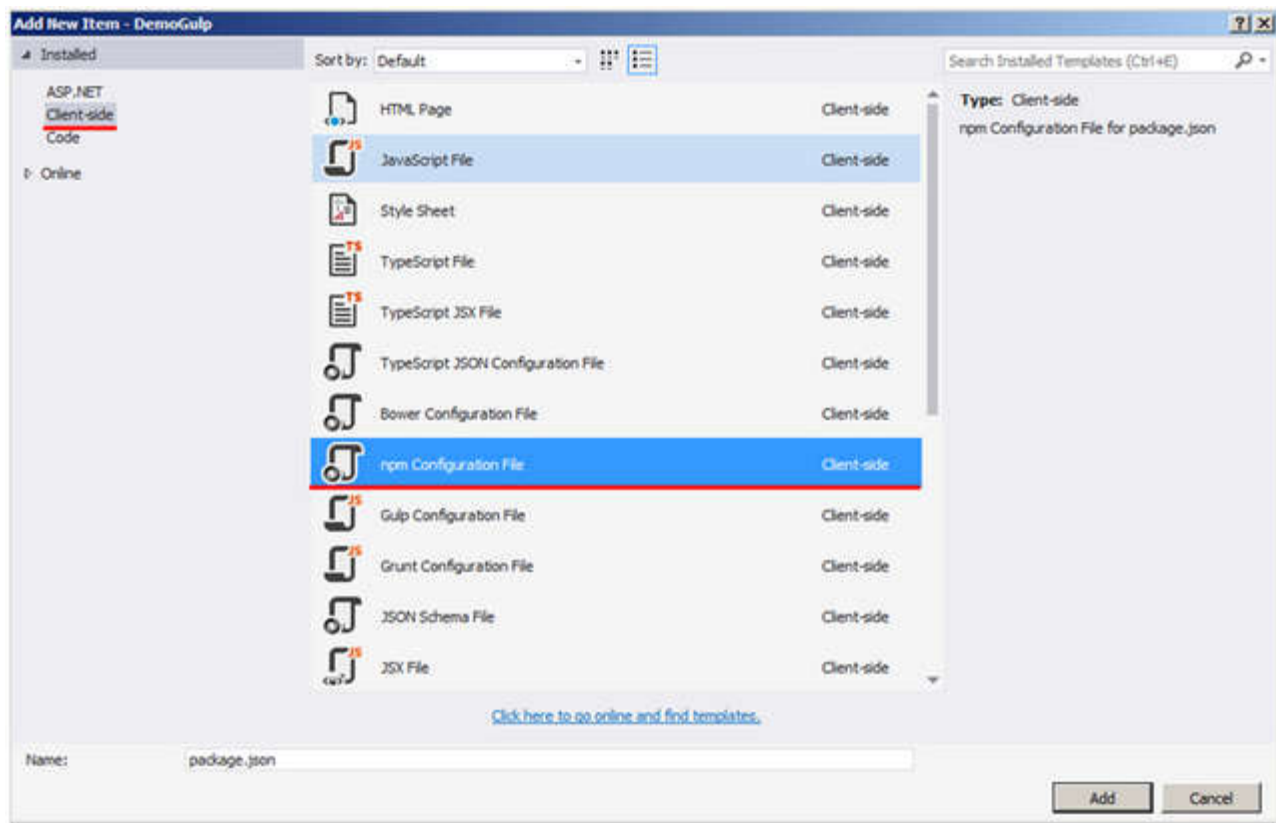
We are going to add *package.json* to the project for downloading packages from node package manager.

To add *package.json*, just right click on **"DemoGulp"** project then select Add from menu list -> then under it select New Item.

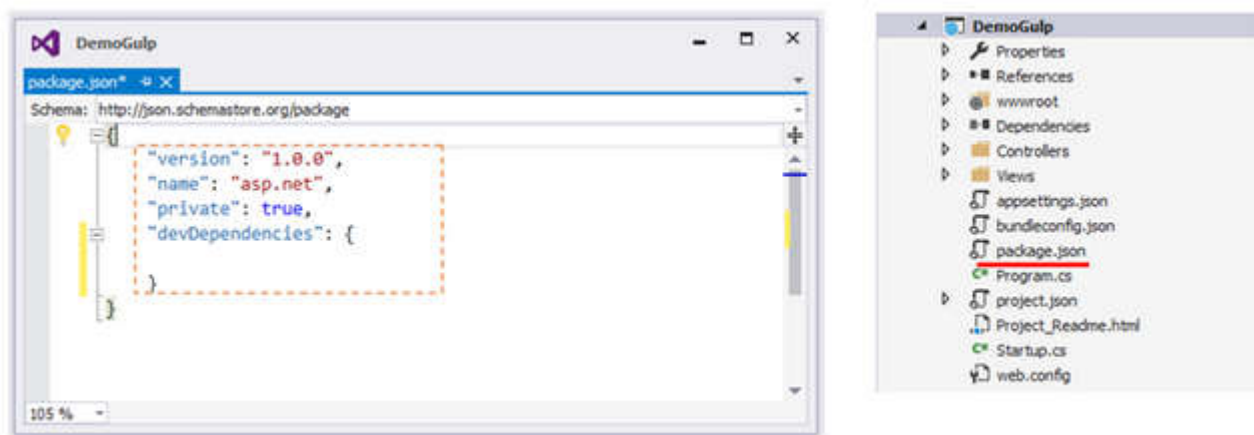


After selecting a new dialog a pop up with the name **"Add New Item"** will appear as shown below.

In this dialog, we are going to choose **"npm Configuration File"** and just click on Add button.



After clicking on Add button, you can see the *package.json* file is added to project as shown below.



After adding a *package.json* file to project, next we are going to download packages for gulp.

Step 3: Downloading gulp packages

We are going to download packages

- **gulp**
- **rimraf**
- **gulp-concat**
- **gulp-cssmin**
- **gulp-uglify**
- **gulp-minify-html**
- **gulp-ext-replace**

Details about Packages

Module Name	Description
gulp	Gulp is a streaming build system
rimraf	A deep deletion module for node
gulp-concat	Module to concat files
gulp-cssmin	Module to Minify Cascading style sheet
gulp-uglify	Module to Minify Javascript files
gulp-minify-html	Module to Minify Html files
gulp-ext-replace	Module to Change file extension

Code snippet to add in package.json

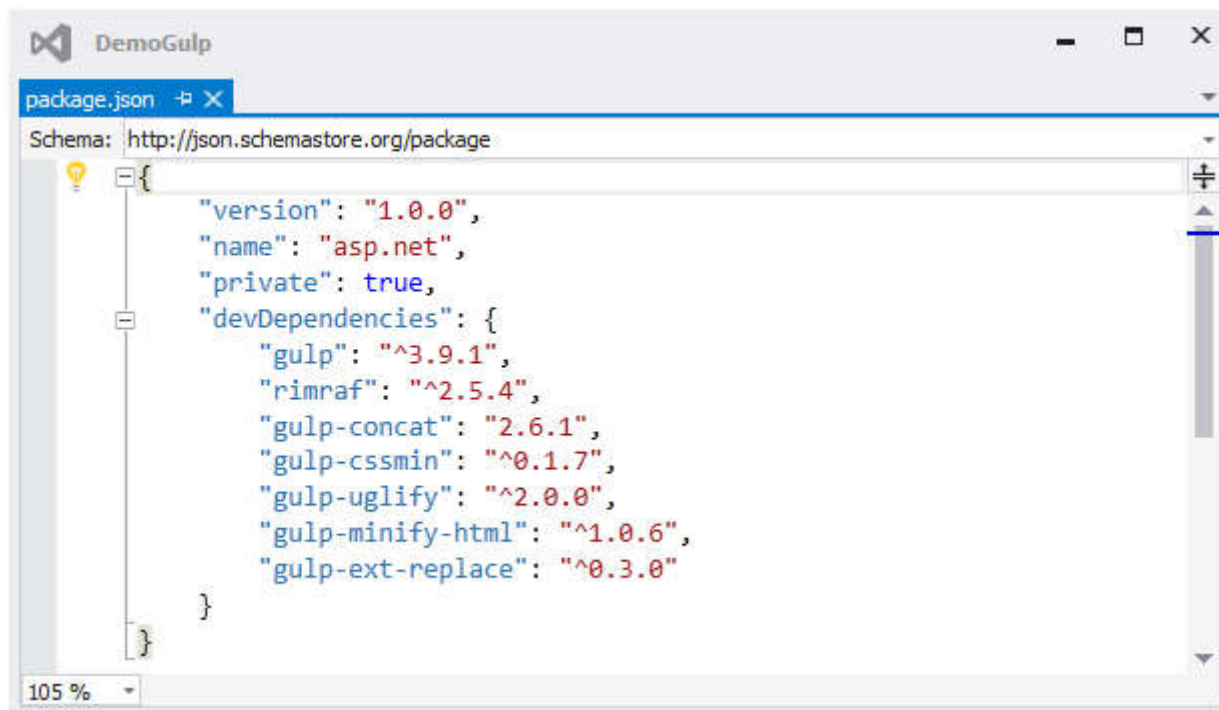
This is the code snippet which we are going to add in the *package.json* **devDependencies** part.

```
"devDependencies": {  
  "gulp": "^3.9.1",  
  "rimraf": "^2.5.4",  
  "gulp-concat": "2.6.1",  
  "gulp-cssmin": "^0.1.7",  
  "gulp-uglify": "^2.0.0",  
  "gulp-minify-html": "^1.0.6",  
  "gulp-ext-replace": "^0.3.0"  
}
```

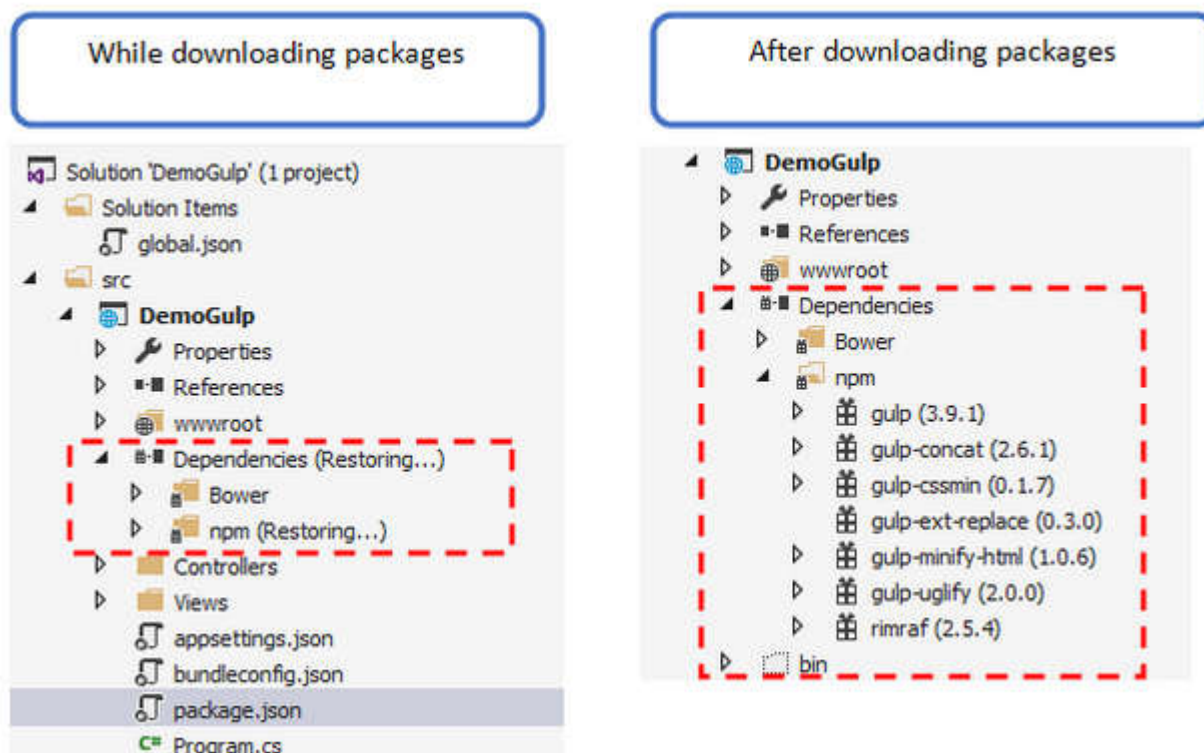
Note

- folder/*.js - will match all the JavaScript files in folder
- folder/*.css - will match all the Css files in folder
- folder/*.html - will match all the HTML files in folder

Snapshot of package.json



After adding Dependencies in the `package.json` file just saving this file or the entire project, as you save it will start downloading packages from **Node Package Manager (NPM)**

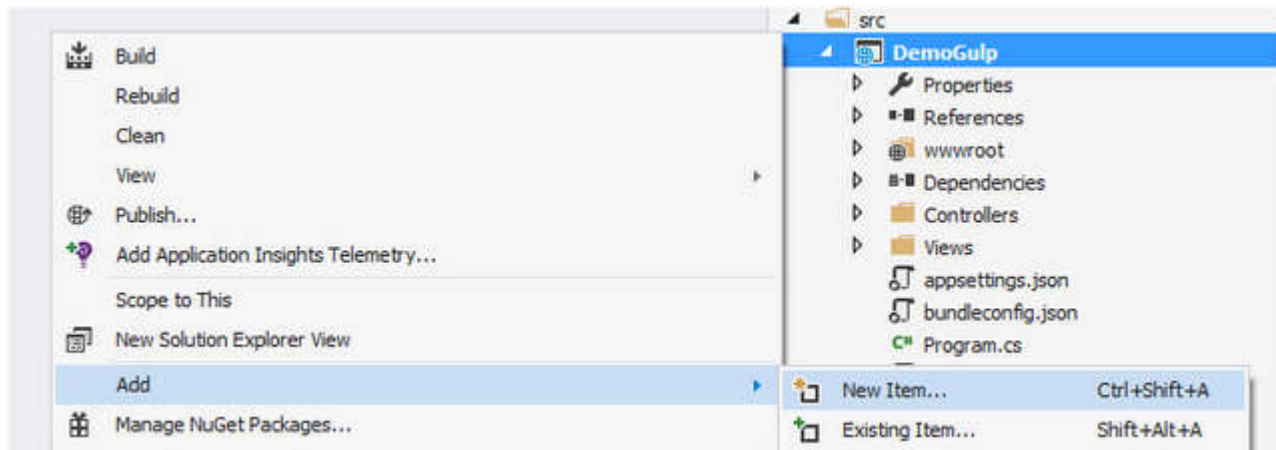


After downloading Dependencies next we are going to add a `gulpfile.js` file to the project.

Step 4: Adding `gulpfile.js` file in project

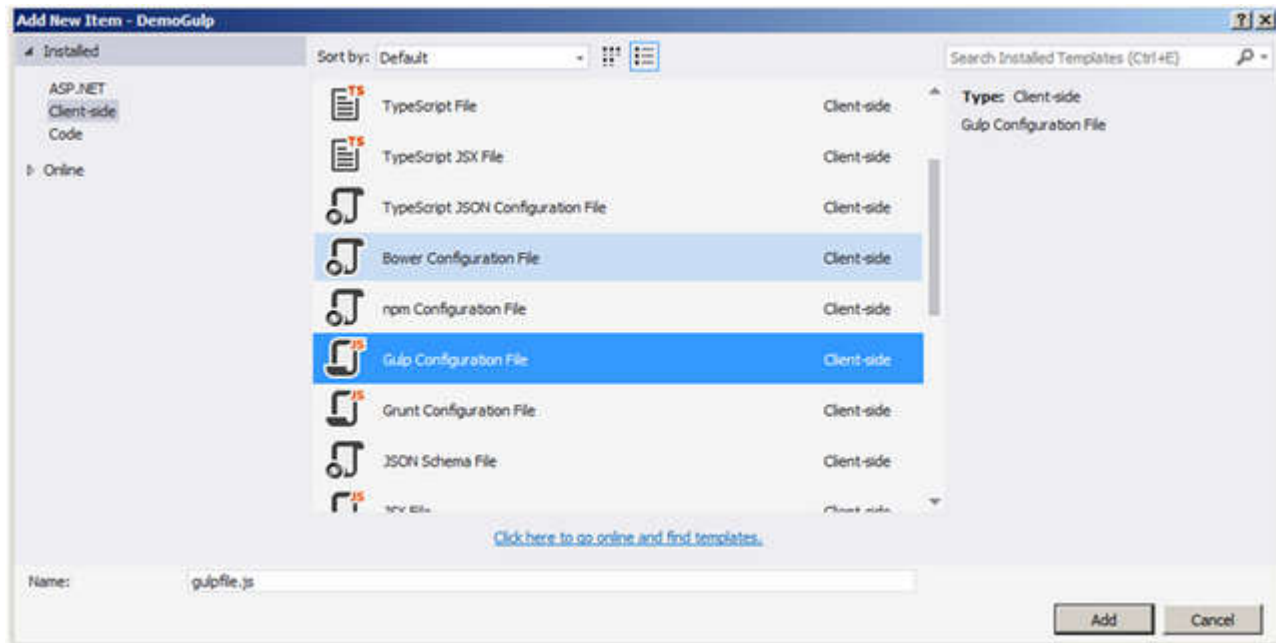
In this step, we are going to add a `gulpfile.js` file to the project.

To add *gulpfile.js* file, just right click on "**DemoGulp**" project then select Add from menu list -> then under it select New Item.



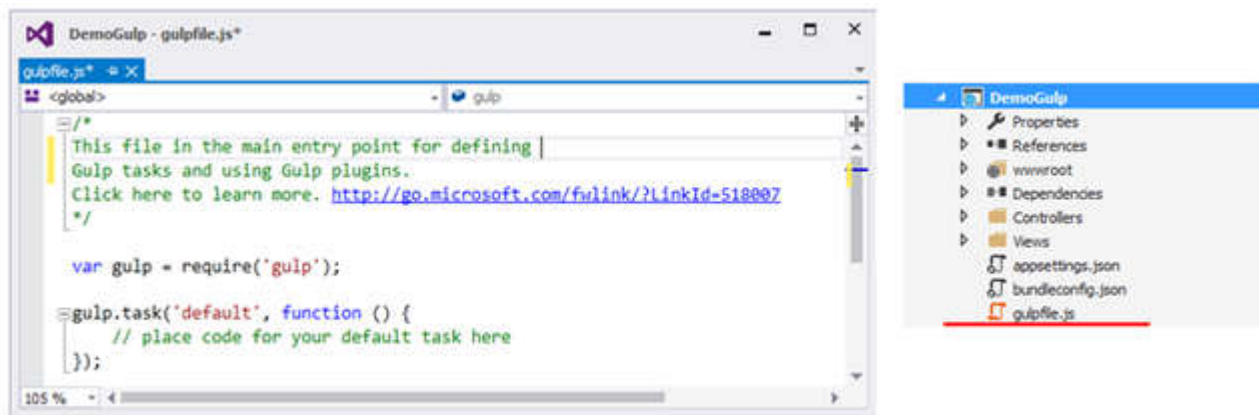
After selecting a new dialog will pop up with name "**Add New Item**" as shown below.

In this dialog, we are going to choose "**Gulp Configuration File**" and just click on Add button.



After clicking on add button it will add **Gulpfile.js** to project as shown below.

You can see some default code in *gulpfile.js* files.



Step 5: Writing basic task to learn how to start using gulp.js

1. The first step is to load module.

```
var gulp = require('gulp');
```

2. In the second step let's have a look at task how it looks like.

```
gulp.task('default', function ()
{
  // place code for your default task here
});
```

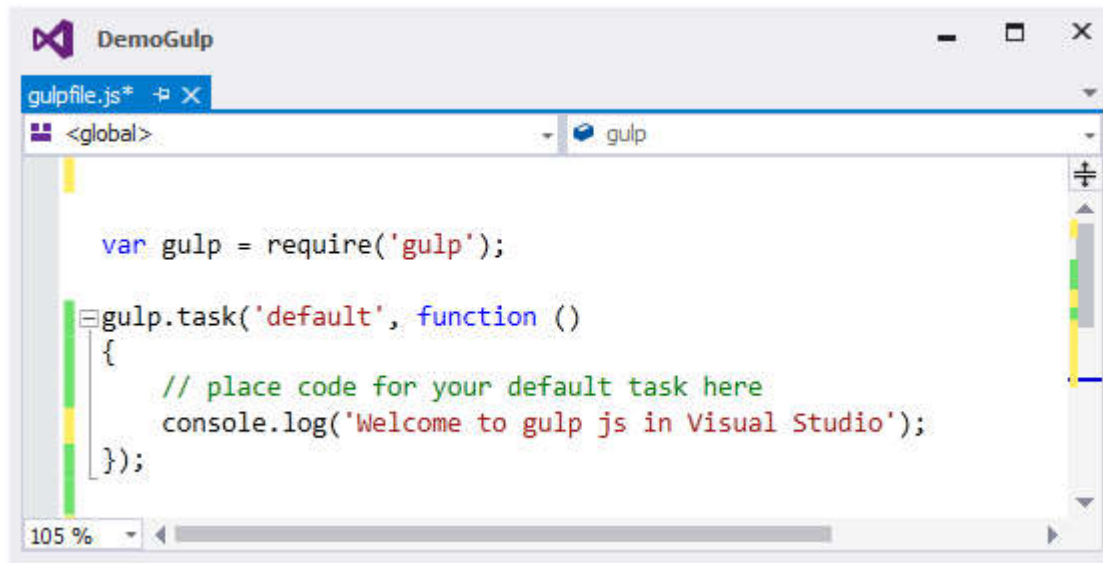
'default' is the name of the task and you can write task inside this function.

3. Now let's write simple message to display "Welcome to gulp js".

```
var gulp = require('gulp');

gulp.task('default', function ()
{
  // place code for your default task here
  console.log('Welcome to gulp js in Visual Studio');
});
```

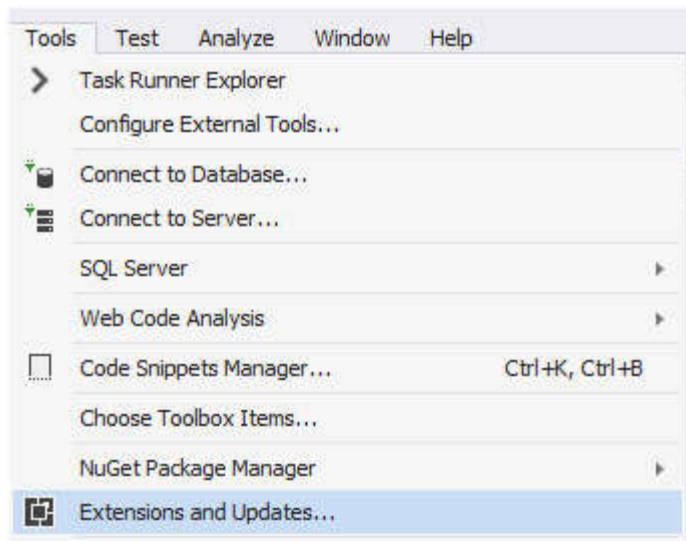
Snapshot of Gulpfile.js



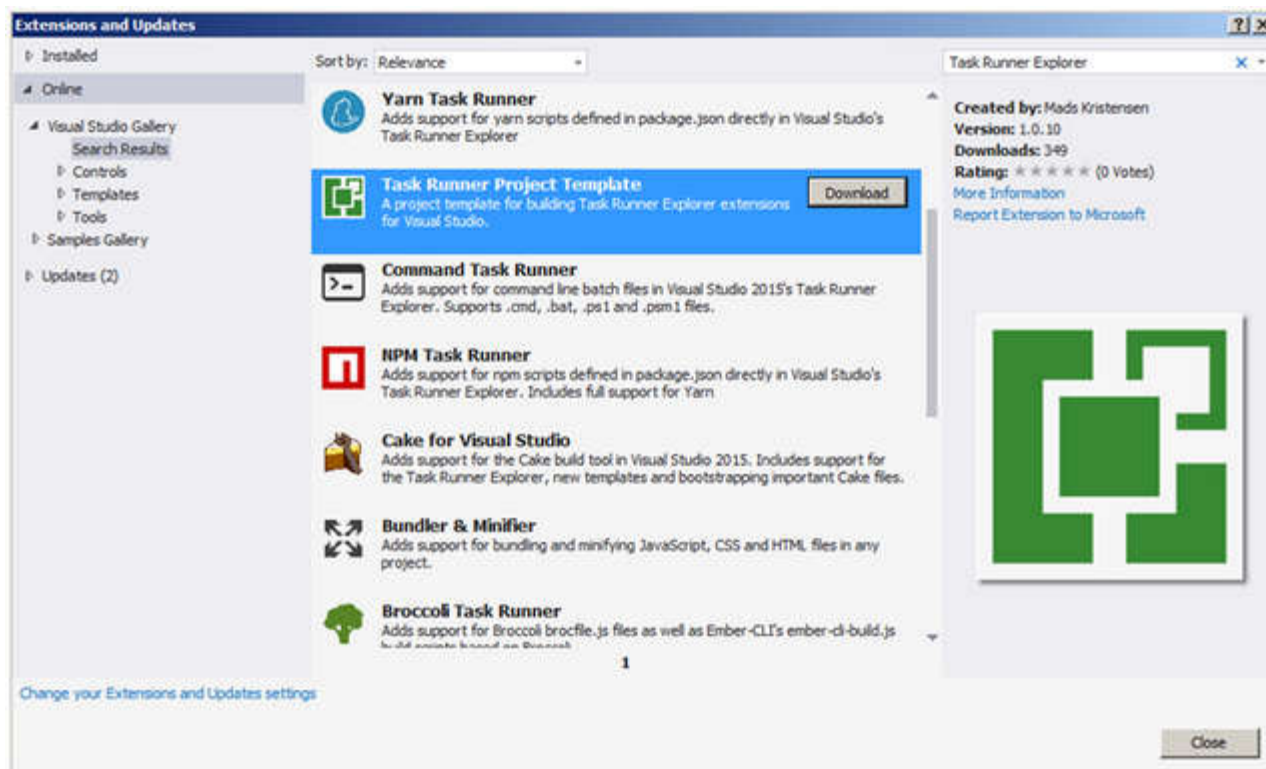
Now we have to run this task which is written in gulpfile.js for doing that we need to install extension "**Task Runner Explorer**"

Step 6: Installing "Task Runner Explorer" Extension in Visual studio

For installing "**Task Runner Explorer**" just right click on Menu bar of visual studio in that choose Tools Menu inside that -> choose **Extension and Updates**.

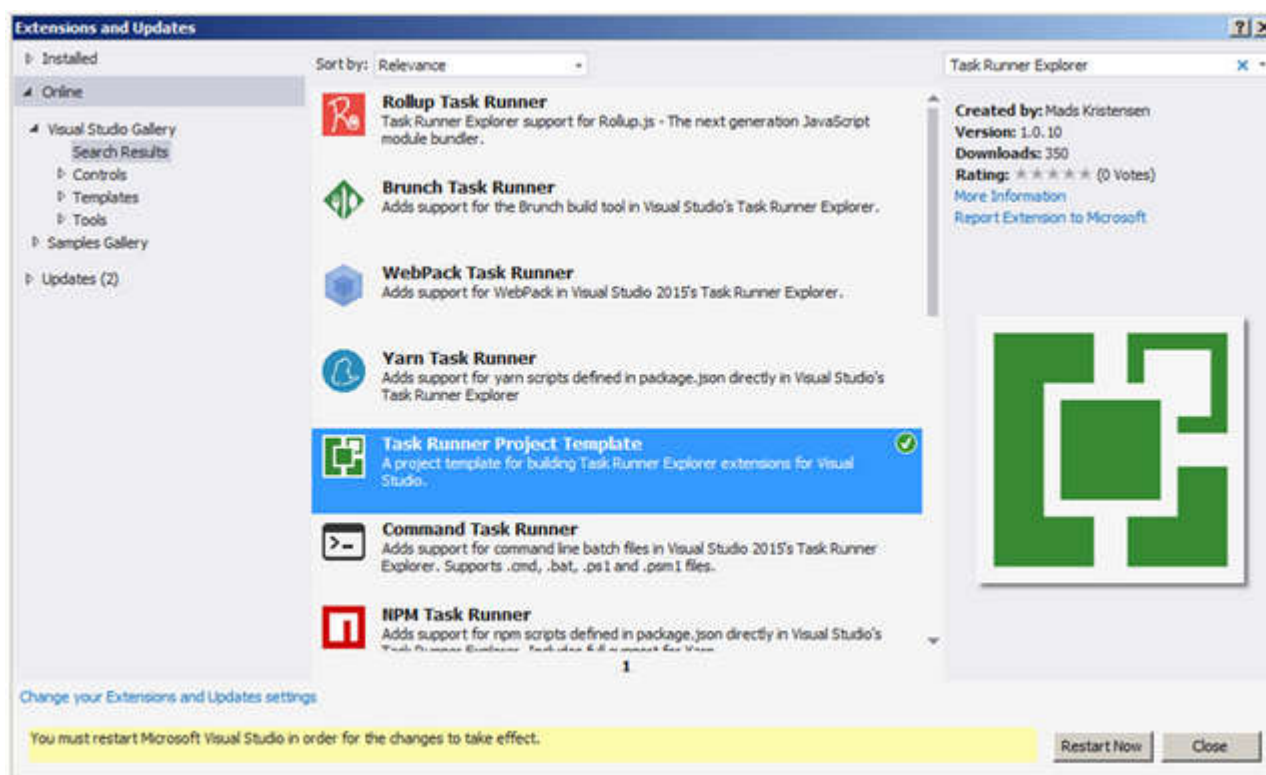


After clicking on **Extension and Updates** a new dialog will pop up as shown below in that you need to choose online tab from left, then in Search result type "**Task Runner Explorer**"



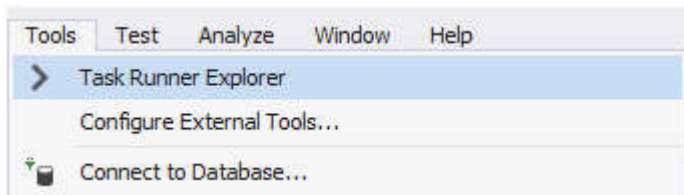
Below is snapshot after installing "Task Runner Explorer".

After installing, it will ask to restart Visual Studio.



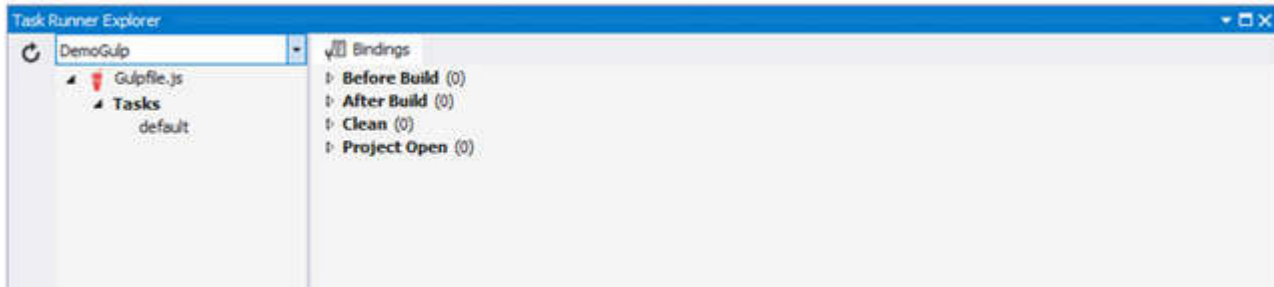
After restarting Visual Studio, next we need to open Task runner explorer.

To open Task runner explorer just click on the Tools Menu from Visual Studio and choose Task Runner Explorer from the Menu.

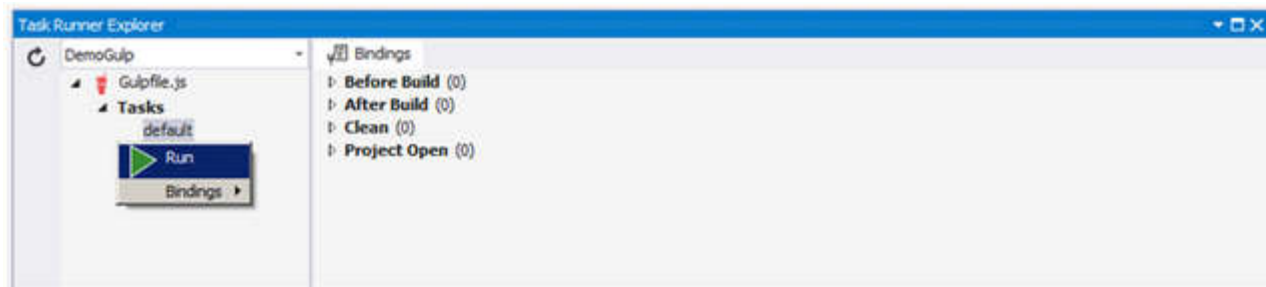


Snapshot of Task Runner explorer

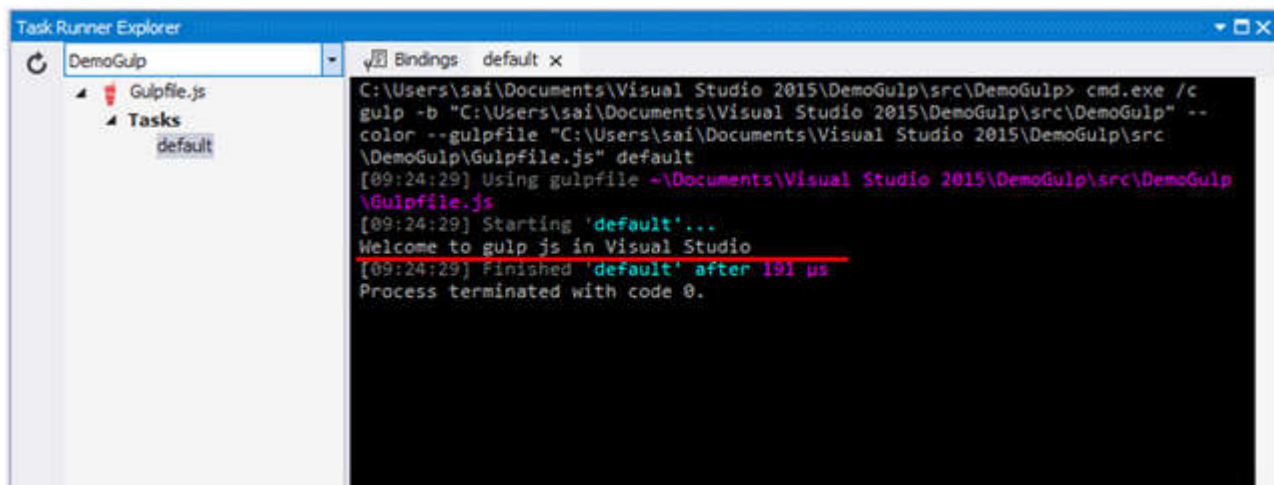
The Task Runner Explorer contains all task of Gulpfile.js and in Tasks, you will find the task which we have created **"default"** task.



Now we are going to run default task to run it just right click on Task Name (default) then two options will appear. From that, choose the Run option (option with a play button in green color).



Below is snapshot after running a task.



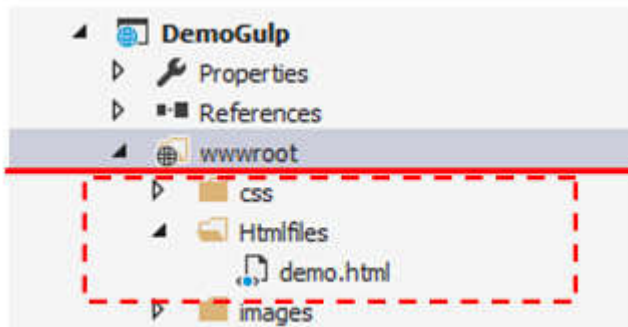
Wow, we have completed running the first task. It is easy, right?

After completing task next, we are going to learn how to minifying an HTML file using gulpfile.js

Step 7: Writing task to minifying Html file

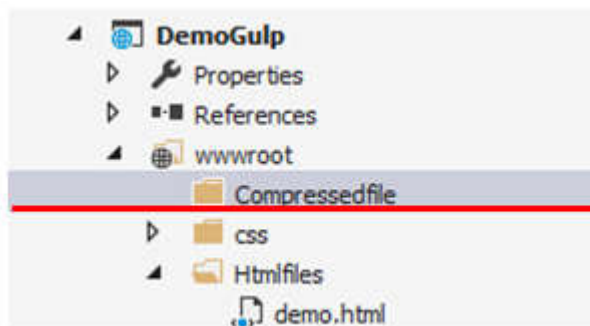
In this step, we are going to write a task for minifying html file for doing that first I am going to add a folder with name *Htmlfiles* in *wwwroot* folder add then inside that folder I am going to add an HTML file with the name *demo.html*.

Folder structure



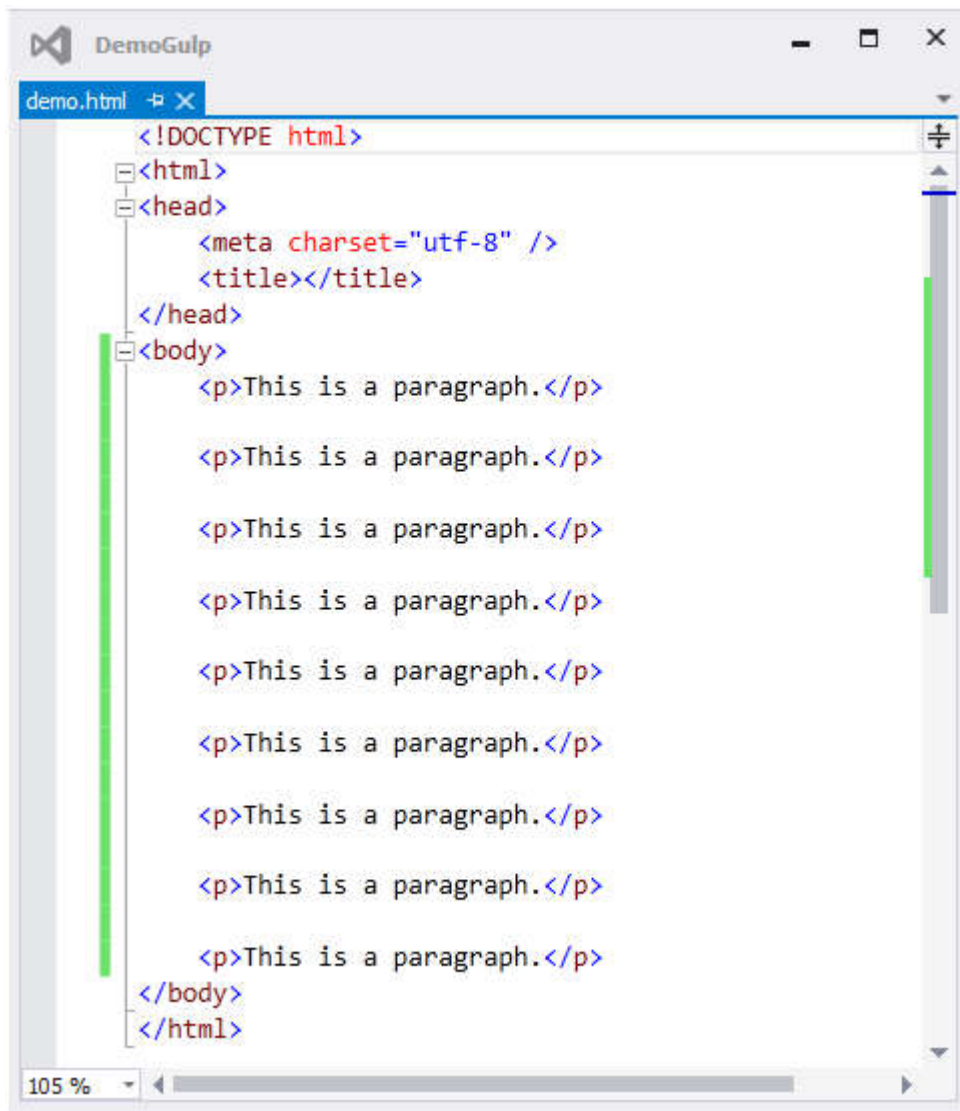
And next I am going to add another folder in same *wwwroot* folder with name *Compressedfiles*. Here we are going to store minified html files.

Folder structure



After adding the *demo.html* file now let add some text with space in this file.

Snapshot of demo.html file



After adding html file with space next we are going to write a task for minifying this html file and write it to other location (Compressed file).

Below is Complete Code Snippet of gulpfile.js

```
var gulp = require('gulp');

//Using package to minify html
minifyHtml = require("gulp-minify-html");

var paths = {
  webroot: "./wwwroot/"
};
//Getting Path of Htmlfiles to Minifying
paths.html = paths.webroot + "Htmlfiles/**/*.html";

//Path to Writing minified Files after Minifying
paths.Destination = paths.webroot + "Compressedfile";

gulp.task('minify-html', function ()
{
  // path to your files
  gulp.src(paths.html)
  // Minifying files
})
```

```
.pipe(minifyHtml())  
// Writing files to Destination  
.pipe(gulp.dest(paths.Destination));  
});  
  
// Main task to Call for Minifying html files  
gulp.task("demo", ["minify-html"]);
```

Understanding code snippet of gulpfile.js

1. In first step we are going to load module.

```
var gulp = require('gulp');  
  
//Using package to minify html  
minifyHtml = require("gulp-minify-html");
```

2. In a second step, we are going to declare Variable with name **"paths"** which is going to store the path of "wwwroot".

```
var paths = {  
  webroot: "./wwwroot/"  
};
```

3. In third step we are going declare path to get files from folder **"Htmlfiles"** folder.

```
//Getting Path of Htmlfiles to Minifying  
paths.html = paths.webroot + "Htmlfiles/**/*.html";
```

4. In the fourth step, we are going declare path to writing Compressed files in **"Compressedfile"** folder

```
//Path to Writing minified Files after Minifying  
paths.Destination = paths.webroot + "Compressedfile";
```

5. In the fifth step, we are going to write task ('minify-html') for getting all html files and use minify package to minify them and write it to destination folder which we have created.

```
gulp.task('minify-html', function ()  
{  
  // path to your files  
  gulp.src(paths.html)  
  // Minifying files  
  .pipe(minifyHtml())  
  // Writing files to Destination  
  .pipe(gulp.dest(paths.Destination));  
});
```

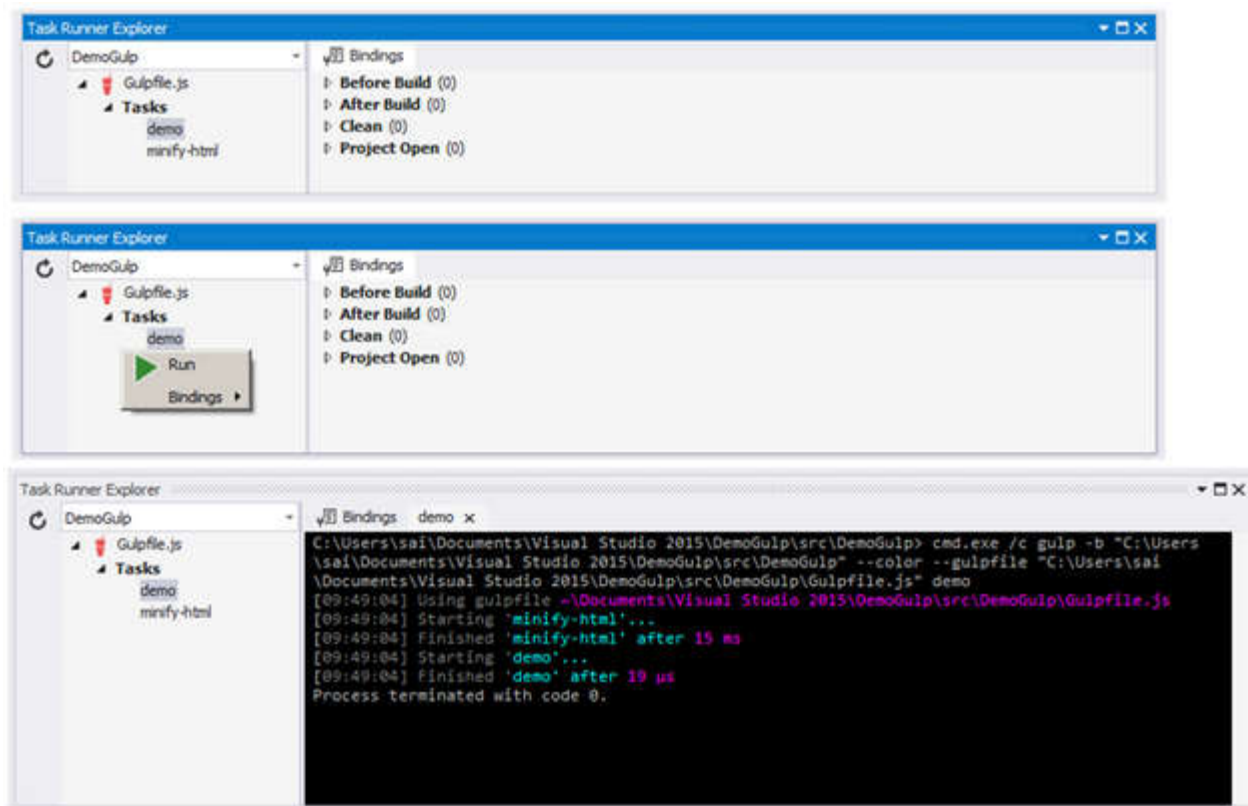
6. In the sixth step, we are going to write Main task such that we can run another task sequentially

```
gulp.task("demo", ["minify-html"]);
```

Finally, save gulpfile.js and open Task Runner Explorer.

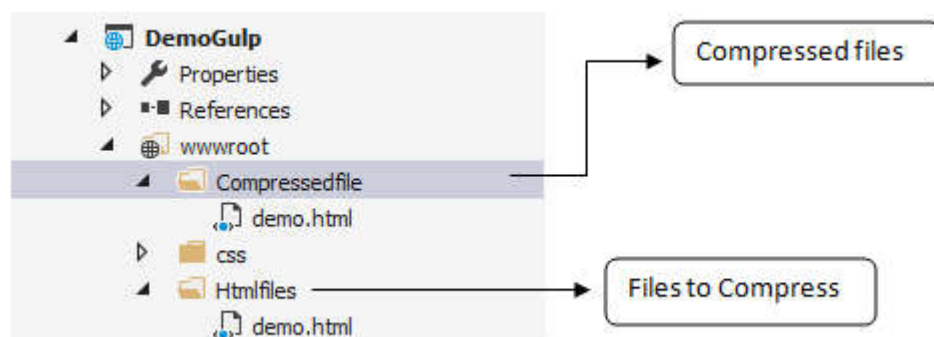
To run just right click on **"demo"** task and from the list of menu and choose Run menu.

Snapshot of Task Runner Explorer



After running task we have to see *Compressedfile* folder where minified html file is stored.

Folder structure



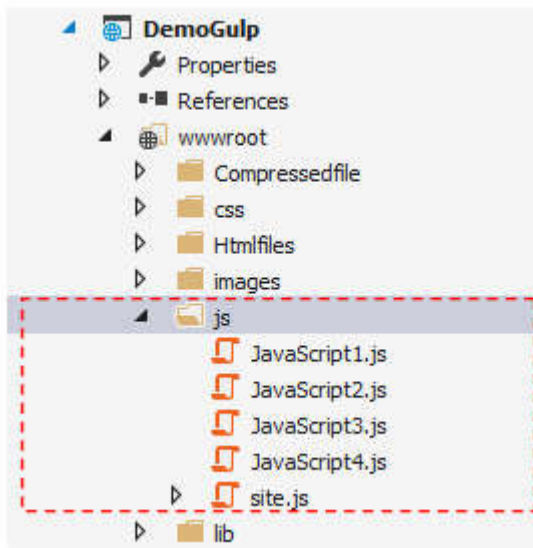
Output of Compressed file



Step 8: Writing task to concatenate file JavaScript files

In this part, we are going to learn how to concatenate JavaScript files for doing that we are going to add four JavaScript files in a folder and then we are going to concat all JavaScript files into one JavaScript file.

Folder structure



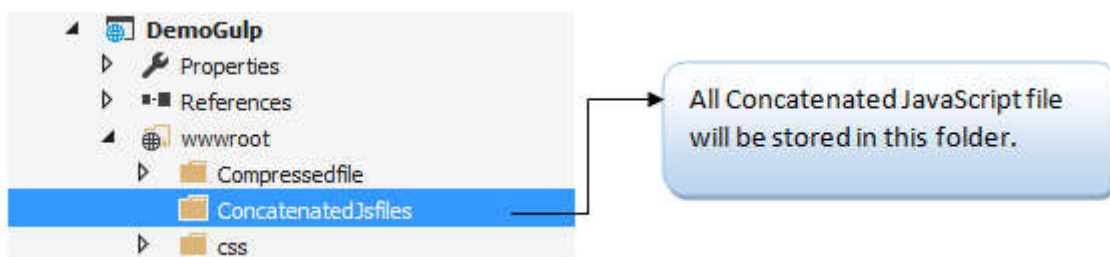
In every JavaScript file, I am going to add a single JavaScript function in it such that we can understand how it concat files.



After writing all the functions in JavaScript individual files, now let's write the task on gulpfile.js to concat all files into one single file.

One thing we have missed here is where we are going to store all Concatenated JavaScript file for storing that file I am going to create a new folder with name "ConcatenatedJsfiles" in which all JavaScript files which are concatenated will store in this folder.

Folder structure



Below is Complete Code Snippet of gulpfile.js

```
var gulp = require('gulp');

//Using package to concat files
concat = require("gulp-concat");

var paths = {
  webroot: "./wwwroot/"
};

//Getting Path of Javascript files
paths.js = paths.webroot + "js/**/*.js";

//Path to Writing concatenated Files after concatenating
paths.Destination = paths.webroot + "ConcatenatedJsfiles";

var newconcat = "concatmain.js"; // Name of new file

// Task Name [concatfiles]
gulp.task('concatfiles', function ()
{
  // path to your files
  gulp.src(paths.js)
    // concat files
    .pipe(concat(newconcat))
    // Writing files to Destination
    .pipe(gulp.dest(paths.Destination));
});

// Main task to Call for concatenating js files
gulp.task("demo", ["concatfiles"]);
```

Understanding code snippet of gulpfile.js

1. In first step we are going to load module.

```
var gulp = require('gulp');

//Using package to concat files
concat = require("gulp-concat");
```

2. In a second step, we are going to declare Variable with name "paths" which is going to store the path of "wwwroot".

```
var paths = {
  webroot: "./wwwroot/"
};
```

3. In third step we are going declare path to get JavaScript files from folder "js" folder.

```
//Getting Path of Javascript files
paths.js = paths.webroot + "js/**/*.js";
```

4. In forth step we are going declare path to store **Concatenated** files in "ConcatenatedJsfiles" folder.

```
//Path to Writing concatenated Files after concatenating
paths.Destination = paths.webroot + "ConcatenatedJsfiles";
```


5. In the fifth step, we are going to declare a variable with name **newconcat** and to this variable, we are going to assign the name of new JavaScript file (*concatmain.js*).

```
var newconcat = "concatmain.js"; // Name of new file
```

6. In sixth step we are going to write task ('**concatfiles**') for getting all JavaScript files and use concat package to concat them and write it to destination folder which we have created.

```
// Task Name [concatfiles]
gulp.task('concatfiles', function ()
{
    // path to your files
    gulp.src(paths.js)
    // concat files
    .pipe(concat(newconcat))
    // Writing files to Destination
    .pipe(gulp.dest(paths.Destination));
});
```

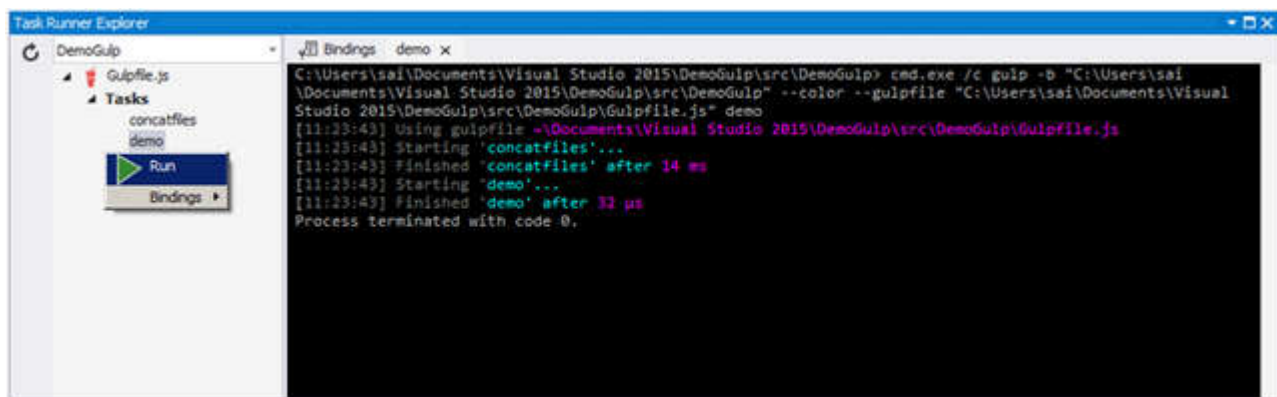
7. In the seventh step, we are going to write Main task such that we can run another task sequentially

```
gulp.task("demo", ["concatfiles"]);
```

Finally, save gulpfile.js and open Task Runner Explorer.

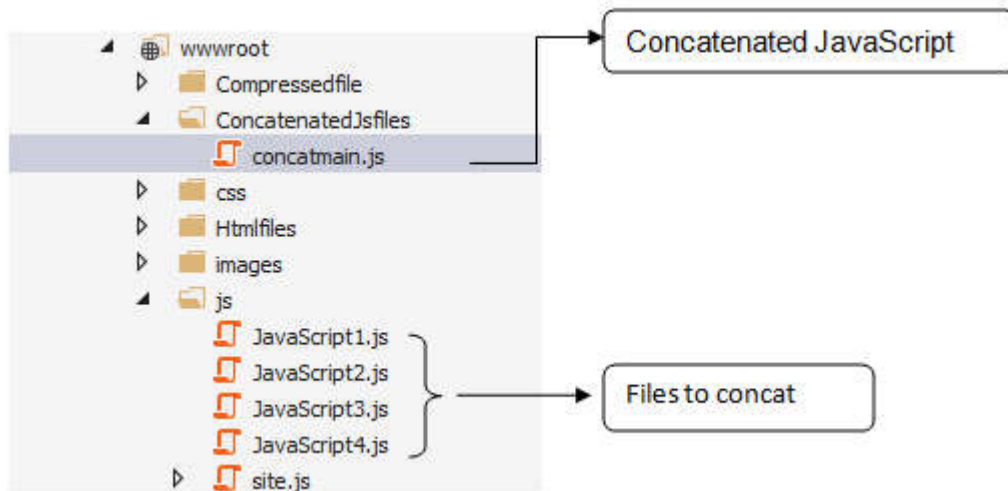
To run just right click on "**demo**" task and from the list of menu and choose Run menu.

Snapshot of Task Runner Explorer

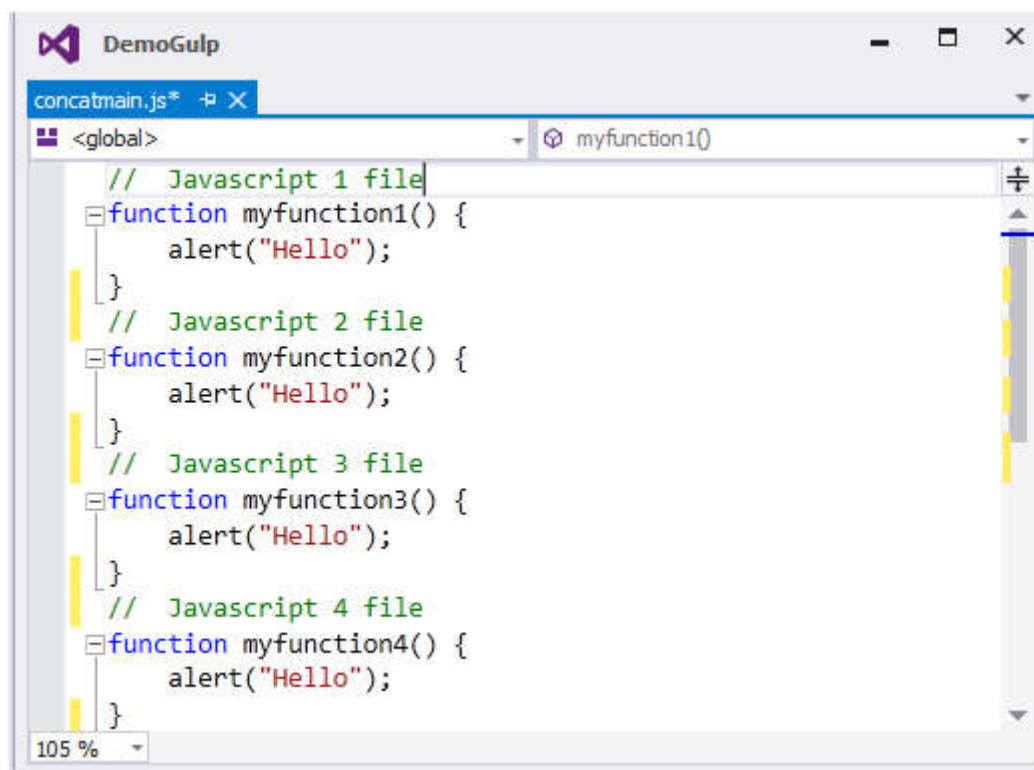


After running task we have to see "*ConcatenatedJsfiles*" folder where concatenated JavaScript file is stored.

Folder structure



Output of Concatenated Javascript files

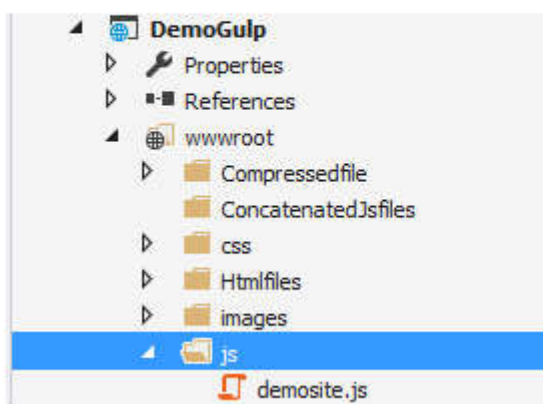


Step 9: Writing task to minify JavaScript

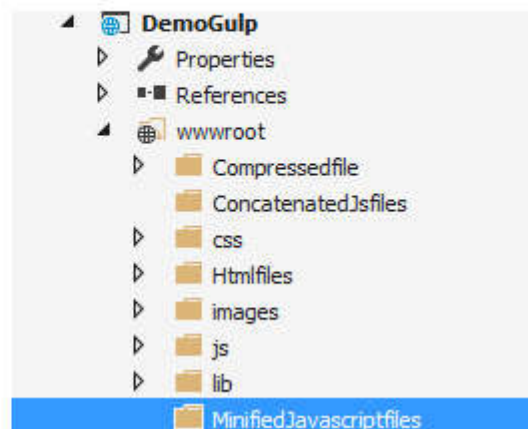
In this step we are going to write a task for minifying Javascript file for doing that. First I am going to add a folder with name *MinifiedJavascriptfiles* in the *wwwroot* folder in this folder all minified javascript files will be stored and we are going to get all javascript files from "Js" folder for minifying.

Folder structure

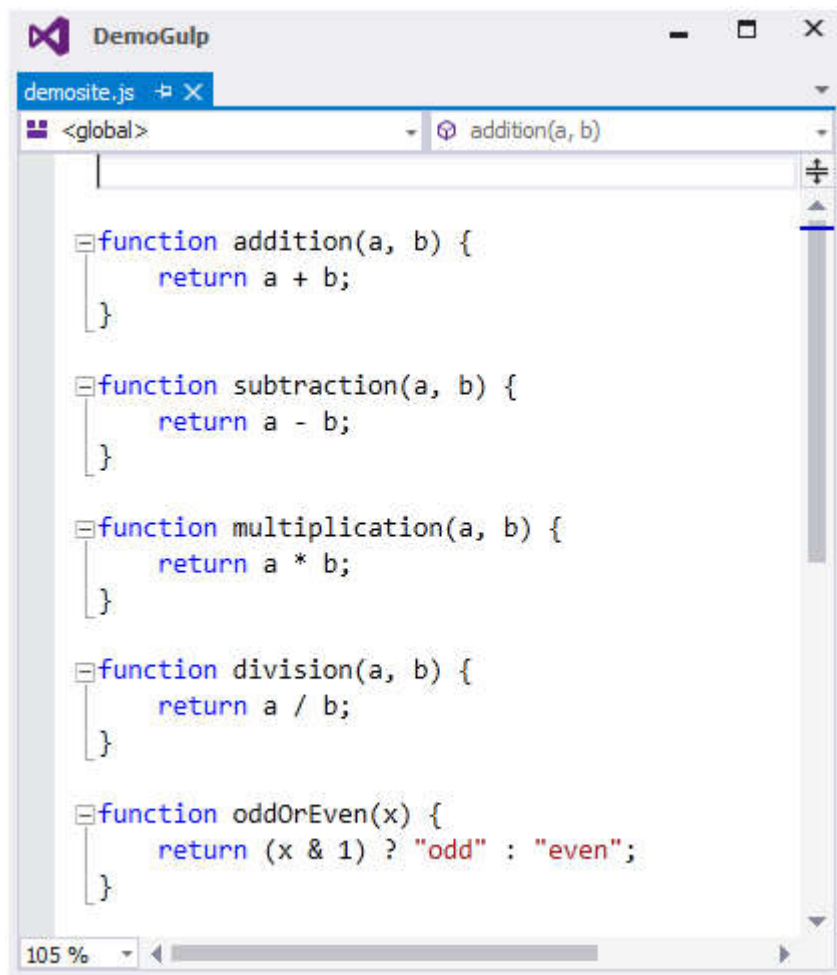
Javascript File to Minify (Js) folder



All minified files will be stored in (MinifiedJavascriptfiles) folder



Javascript File to minify



Below is Complete Code Snippet of gulpfile.js

```
var gulp = require('gulp');
```

```
//Using package to minifying files
uglify = require("gulp-uglify");

var paths = {
  webroot: "./wwwroot/"
};

//Getting Path of Javascript files
paths.js = paths.webroot + "js/**/*.js";

//Path to Writing minified Files after minifying
paths.Destination = paths.webroot + "MinifiedJavascriptfiles";

// Task Name [MinifyingTask]
gulp.task('MinifyingTask', function () { // path to your files
  gulp.src(paths.js)
    // concat files
    .pipe(uglify())
    // Writing files to Destination
    .pipe(gulp.dest(paths.Destination));
});
```

Understanding code snippet of gulpfile.js

1. In first step we are going to load module.

```
var gulp = require('gulp');

//Using package to minifying files
uglify = require("gulp-uglify");
```

2. In a second step, we are going to declare Variable with name "paths" which is going to store the path of "wwwroot".

```
var paths = {
  webroot: "./wwwroot/"
};
```

3. In third step we are going declare path to get JavaScript files from folder "Js" folder.

```
//Getting Path of Javascript files
paths.js = paths.webroot + "js/**/*.js";
```

4. In a fourth step, we are going declare the path to store **minified** files in "MinifiedJavascriptfiles" folder after minifying.

```
//Path to Writing minified Files after minifying
paths.Destination = paths.webroot + "MinifiedJavascriptfiles";
```

5. In the fifth step, we are going to write task ('MinifyingTask') for getting all JavaScript files and use uglify package to minify JavaScript files and then and write it to destination folder which we have created.

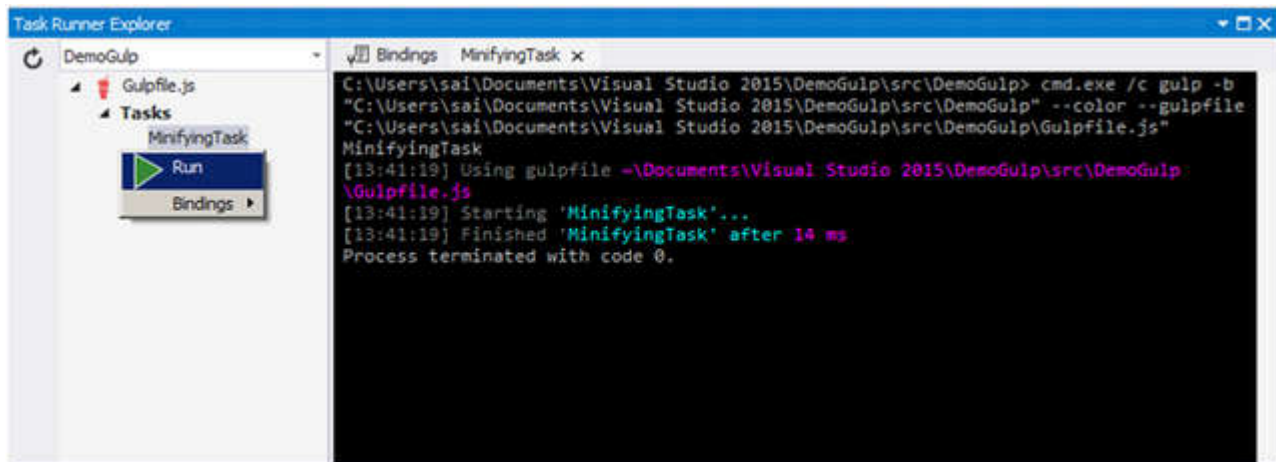
```
// Task Name [MinifyingTask]
gulp.task('MinifyingTask', function ()
{ // path to your files
  gulp.src(paths.js)
    // concat files
    .pipe(uglify())
```

```
// Writing files to Destination
.pipe(gulp.dest(paths.Destination));
});
```

Finally, save gulpfile.js and open Task Runner Explorer.

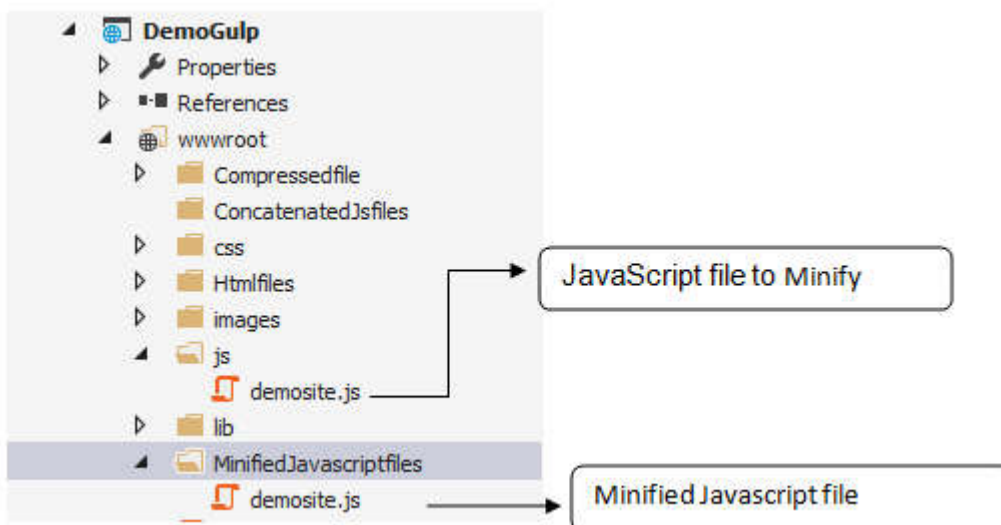
To run just right click on "**MinifyingTask**" task and from the list of menu and choose Run menu.

Snapshot of Task Runner Explorer

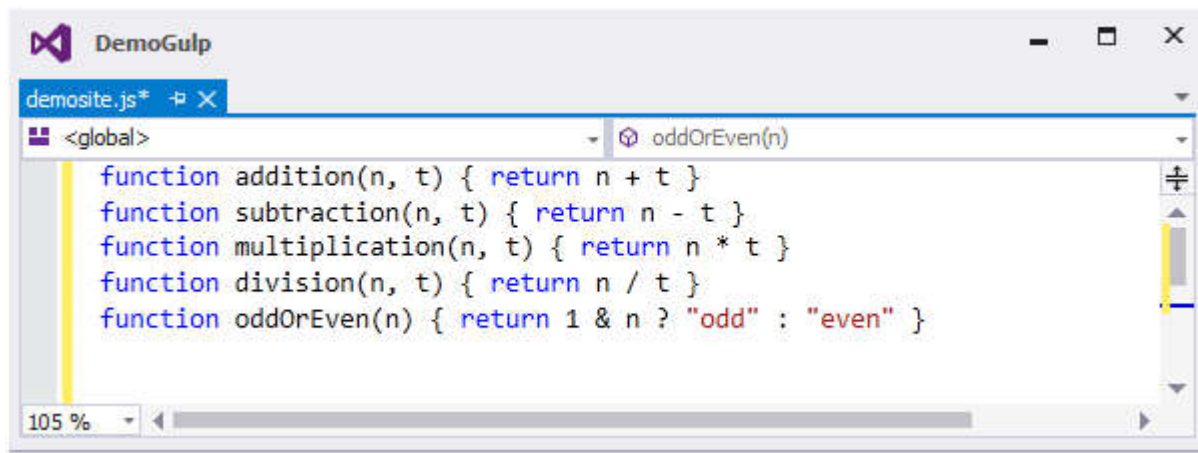


After running task we have to see "*MinifiedJavascriptfiles*" folder where minified JavaScript file is stored.

Folder structure



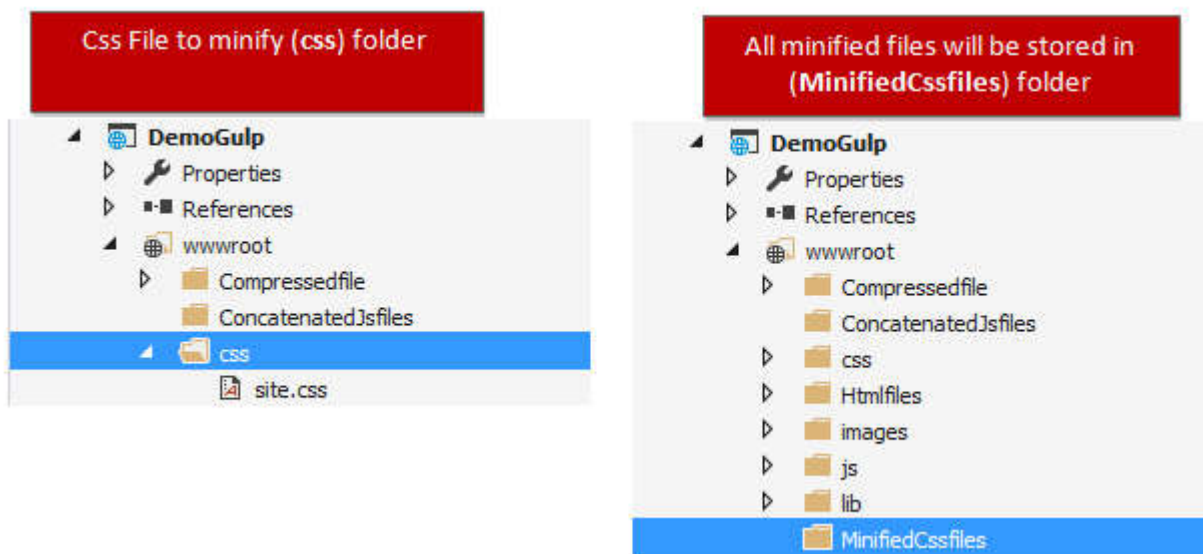
Output of Minified JavaScript files



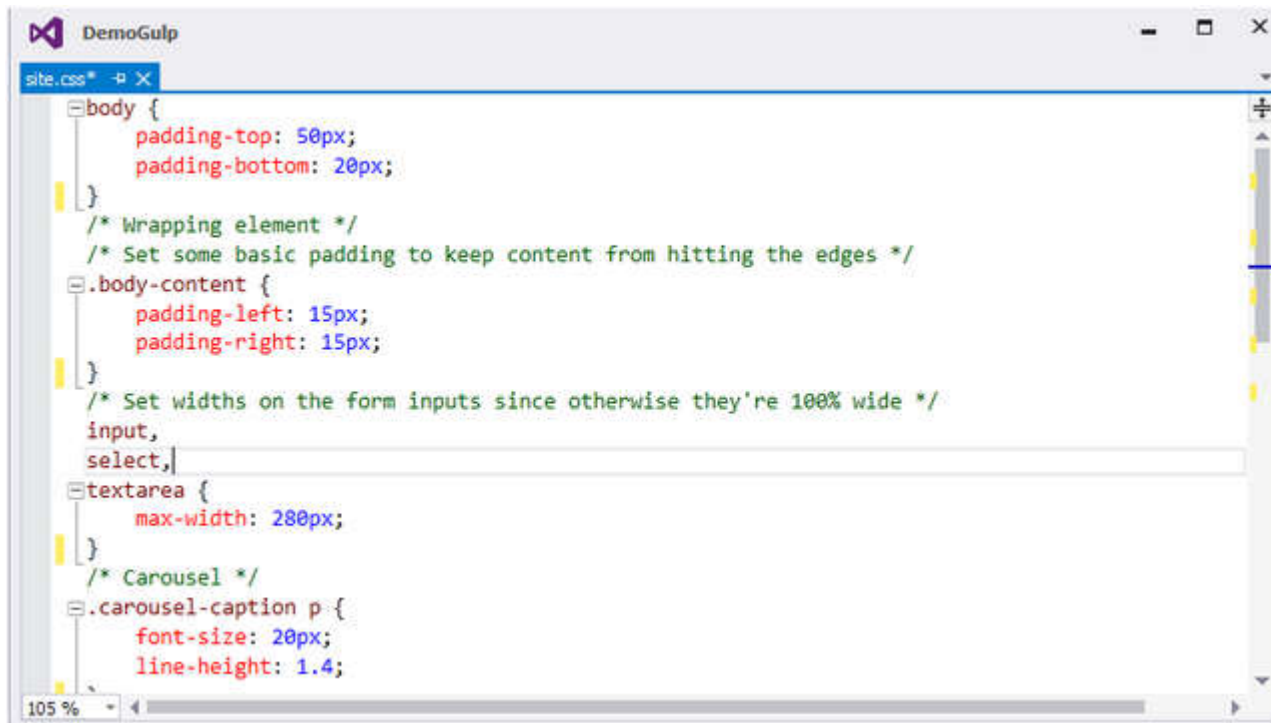
```
function addition(n, t) { return n + t }
function subtraction(n, t) { return n - t }
function multiplication(n, t) { return n * t }
function division(n, t) { return n / t }
function oddOrEven(n) { return 1 & n ? "odd" : "even" }
```

Step 10: Writing task to Minify cascading style sheets

In this step we are going to write a task for minifying cascading style sheets files. To do that, first I am going to add a folder with name *MinifiedCssfiles* in the *wwwroot* folder in this folder all minified cascading style sheets files will be stored and we are going to get all cascading style sheets files from the "css" folder for minifying.



Cascading style sheets File to minify



Below is Complete Code Snippet of gulpfile.js

```
var gulp = require('gulp');

//Using package to minifying css files
cssmin = require("gulp-cssmin");

var paths = {
  webroot: "./wwwroot/"
};

//Getting Path of Javascript files
paths.css = paths.webroot + "css/**/*.css";

//Path to Writing minified Files after minifying
paths.Destination = paths.webroot + "MinifiedCssfiles";

// Task Name [MinifyingTask]
gulp.task('Minifying-Css-Task', function () { // path to your files
  gulp.src(paths.css)
    .pipe(cssmin())
    // Writing files to Destination
    .pipe(gulp.dest(paths.Destination));
});
```

Understanding code snippet of gulpfile.js

1. In first step we are going to load module.

```
var gulp = require('gulp');

//Using package to minifying css files
cssmin = require("gulp-cssmin");
```

- In a second step, we are going to declare Variable with name "**paths**" which is going to store the path of "**wwwroot**".

```
var paths = {
    webroot: "./wwwroot/"
};
```

- In third step we are going declare path to get JavaScript files from folder "css" folder.

```
//Getting Path of cascading style sheets files
paths.css = paths.webroot + "css/**/*.css";
```

- In a fourth step, we are going declare the path to store **minified** files in "**MinifiedCssfiles**" folder after minifying.

```
//Path to Writing minified Files after minifying
paths.Destination = paths.webroot + "MinifiedCssfiles";
```

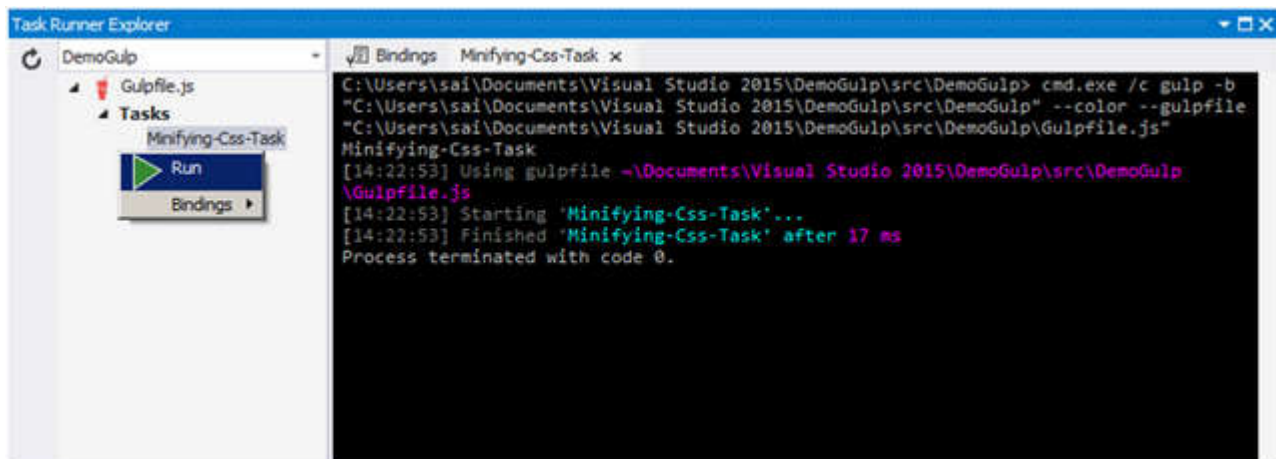
- In the fifth step, we are going to write task ('**Minifying-Css-Task**') for getting all Cascading style sheets files and use **cssmin** package to minify Cascading style sheets files and then write it to destination folder which we have created.

```
// Task Name [Minifying-Css-Task]
gulp.task('Minifying-Css-Task', function () { // path to your files
    gulp.src(paths.css)
    .pipe(cssmin())
    // Writing files to Destination
    .pipe(gulp.dest(paths.Destination));
});
```

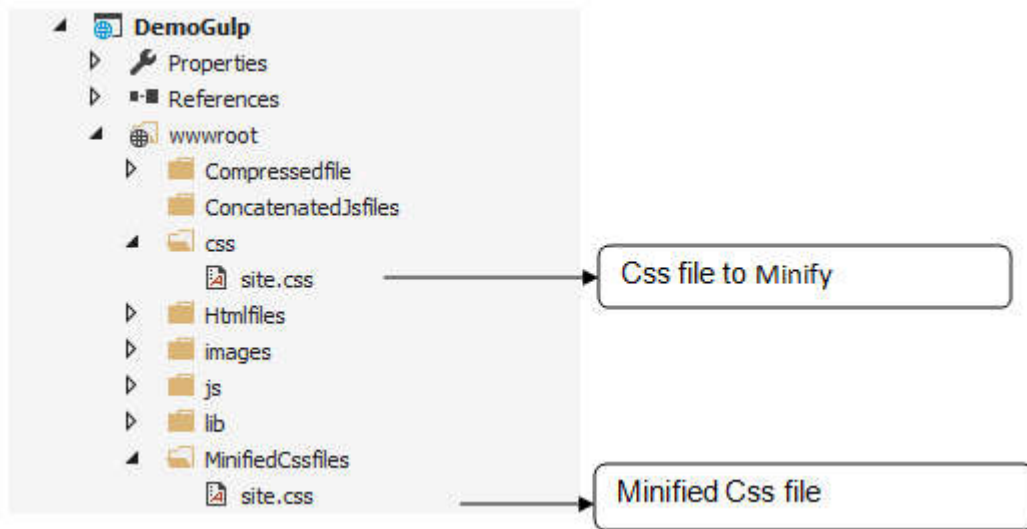
Finally, save gulpfile.js and open Task Runner Explorer.

To run just right click on "**Minifying-Css-Task**" and from the list of menu and choose Run menu.

Snapshot of Task Runner Explorer



After running task we have to see "**MinifiedCssfiles**" folder where minified cascading style sheets file is stored.



Output of Minified Javascript files

The screenshot shows a code editor window titled **DemoGulp** with a tab for `site.css*`. The minified CSS code is displayed as follows:

```
body{padding-top:50px;padding-bottom:20px}
.body-content{padding-left:15px;padding-right:15px}
input,select,textarea{max-width:280px}
.carousel-caption p{font-size:20px;line-height:1.4}
.carousel-inner .item img[src$=".svg"]
{width:100%}@media screen and (max-width:767px){.carousel-caption{display:none}}
```

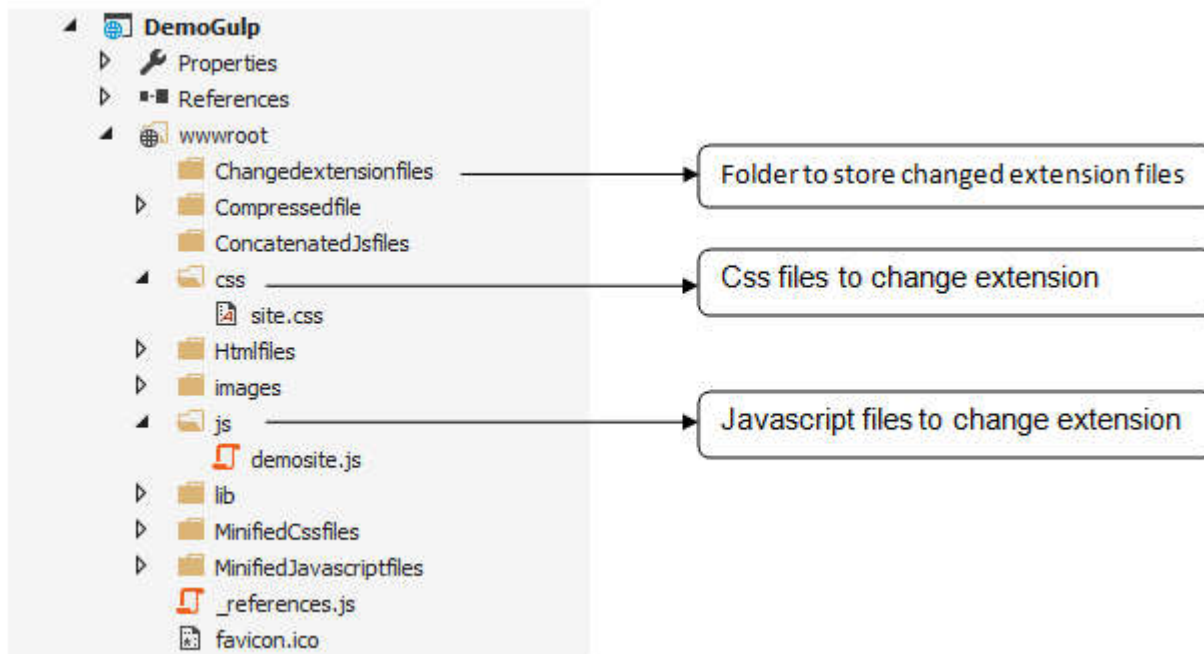
Note: Syntax If you want to run more than one task sequentially

Step 11: Writing task to Change file extension of JavaScript files and Cascading style sheet

In this step, we are going to write a task for changing extension of cascading style sheets file and javascript files. To do that, first I am going to add a folder with the name *Changextensionfiles* in *wwwroot* folder in this folder all changing extension of cascading style sheets file and javascript file will be stored.

And we are going to get cascading style sheets file from the "css" folder and the javascript files from the "js" folder.

Folder structure



Below is Complete Code Snippet of gulpfile.js

```
var gulp = require('gulp');

//Using package to minifying files
ext_replace = require("gulp-ext-replace");

var paths = {webroot: "./wwwroot/"};

//Getting Path of Javascript files
paths.js = paths.webroot + "js/**/*.js";

//Getting Path of cascading style sheets files
paths.css = paths.webroot + "css/**/*.css";

//Path to Writing Changed extensions files
paths.Destination = paths.webroot + "Changedextensionfiles";

// Task Name [Changing Extension Javascript]
gulp.task('ChangingExtensionJavascriptTask', function () {
    // path to your files
    gulp.src(paths.js)
    // change file extension from .js to .min.js
    .pipe(ext_replace('.min.js'))
    // Writing files to Destination
    .pipe(gulp.dest(paths.Destination));
});

// Task Name [Changing Extension Css]
gulp.task('ChangingExtensionCssTask', function () {
    // path to your files
    gulp.src(paths.css)
    // change file extension from css to .min.css
    .pipe(ext_replace('.min.css'))
    // Writing files to Destination
    .pipe(gulp.dest(paths.Destination));
});
```



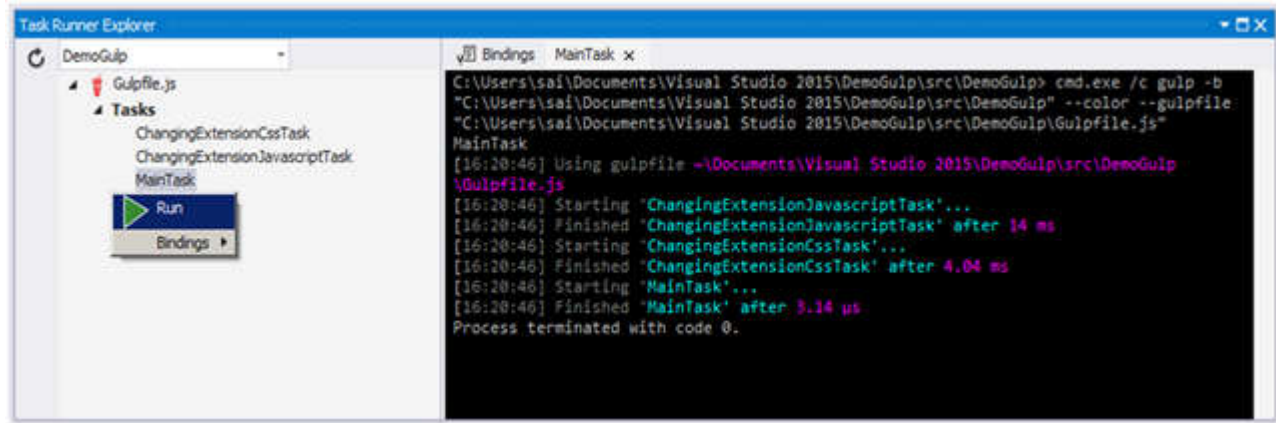
```
//Syntax If you want to run more than one task sequentially
```

```
gulp.task("MainTask", ["ChangingExtensionJavascriptTask", "ChangingExtensionCssTask"]);
```

Finally, save gulpfile.js and open Task Runner Explorer.

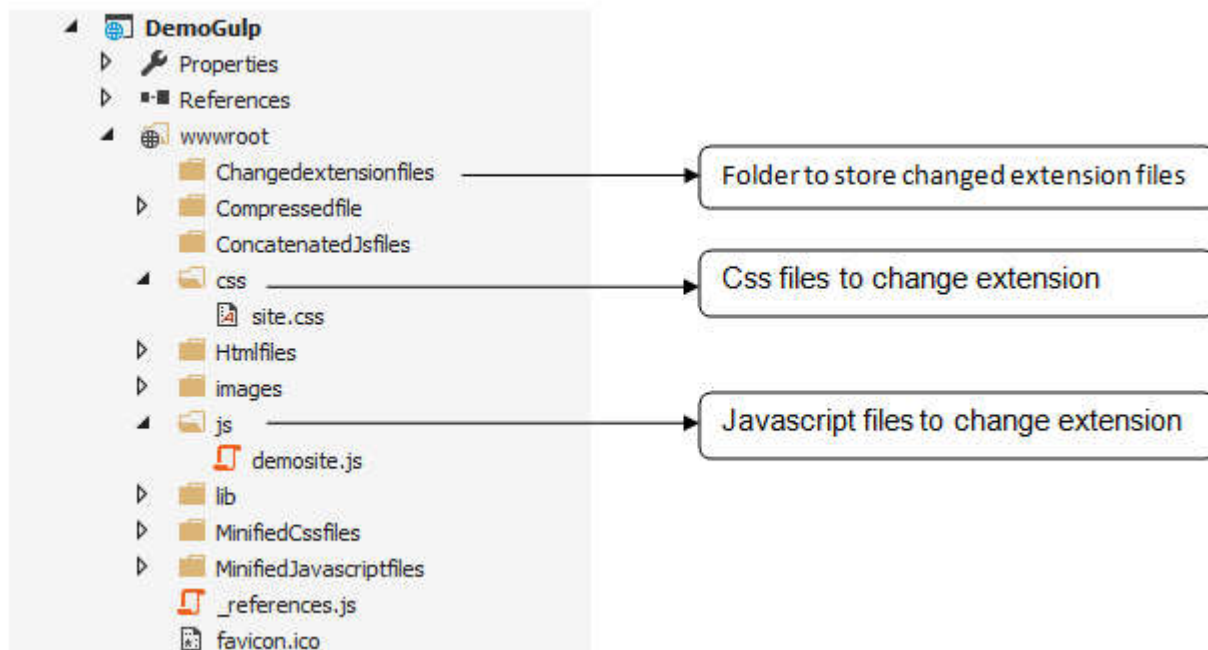
To run just right click on "**MainTask**" and from the list of menu and choose Run menu.

Snapshot of Task Runner Explorer



After running task we have to see "*Changedextensionfiles*" folder where Changed cascading style sheets file and javascript file will be stored.

Snapshot after Changing file extension

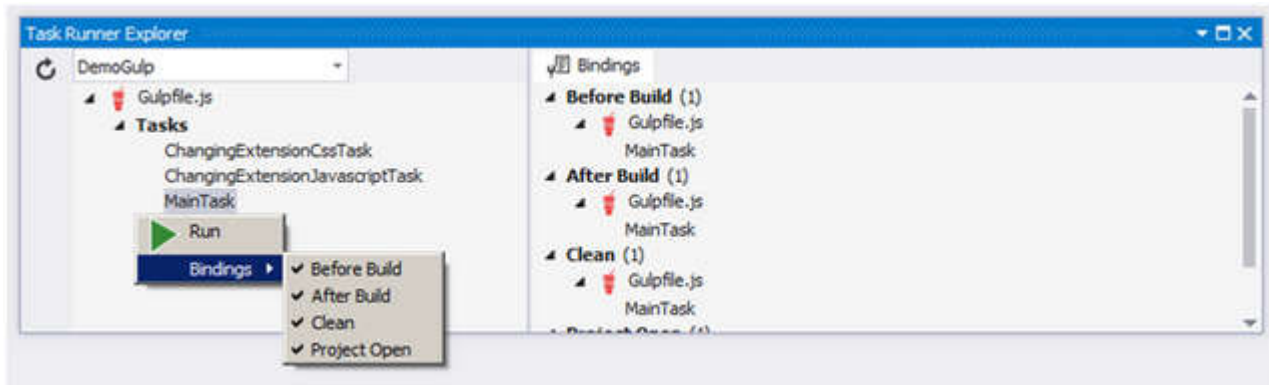


Step 12: Automating Task

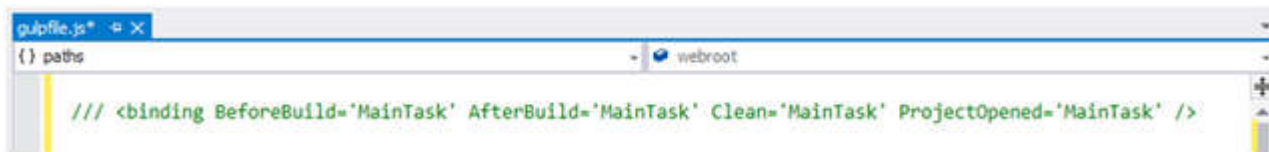
To automate this process, just right click on Bindings from the list you will find four items in it.

Choose Bindings according to your need to automate the process.

Binding	Description
Before build	The Task Run before Visual studio starts building solution
After build	The Task Run after Visual studio building solution is completed
Clean	The Task Run when we do clean solution in Visual studio
ProjectOpened	The Task Run when project is opened in Visual studio



As you choose this binding are added to gulpfile.js as shown below.



In this article, we have learned how to use gulp with ASP.NET Core MVC and along with this how to install "Task Runner Explorer" and how to minify and beautify JavaScript, cascading style sheets, and HTML.

I hope you liked my Article.

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

About the Author



saineshwar bageri



Software Developer (Senior)

India

I am Microsoft MVP | C# Corner MVP working on .Net Web Technology
[ASP.NET MVC, .Net Core, ASP.NET CORE, C#, Sqlserver, MYSQL, MongoDB, Windows]

Microsoft MVP Profile Link

<https://mvp.microsoft.com/en-us/PublicProfile/5003160?fullName=Saineshwar%20%20Bageri>

Prize.


First Prize: Best Web Dev Article of August 2016 with 10 Points to Secure Your ASP.NET MVC Applications.

Second Prize: Best Web Dev Article of April 2017 with Securing ASP.NET Web API using Custom Token Based Authentication

Second Prize:

Best Web Dev Article of February 2018 with Securing ASP.NET CORE Web API using Custom API Key based Authentication

Comments and Discussions

 **5 messages** have been posted for this article Visit <https://www.codeproject.com/Articles/1165004/How-to-Use-Gulp-with-ASP-NET-Core-MVC> to post and view comments on this article, or click [here](#) to get a print view with messages.

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Cookies](#) | [Terms of Use](#) | Mobile
Web04 | 2.8.181119.1 | Last Updated 10 Jan 2017

Article Copyright 2017 by saineshwar bageri
Everything else Copyright © [CodeProject](#), 1999-2018