Minifying your CSS, JS & HTML files using Gulp

Amardeep Rai Following
Nov 29, 2017 · 5 min read ★

When building a website it is important to consider the build process of CSS, JS & HTML files. Manual processes are not only slow, but they can be a cause of mistakes made in code, so it is always a good idea to automate the processes as much as possible.

In this article, I will be describing how to use Gulp to automate the minification of CSS, JS & HTML files.

What is Gulp?

Gulp is an open-source JavaScript toolkit, built on Node.js and npm, for automating painful or time-consuming tasks in front-end web development workflows.

Minifying files using Gulp

1. Get your computer ready

To begin, you will need to make sure that you have Node.js installed on your computer. You can download and install Node.js by visiting: https://nodejs.org/

2. Create a 'package.json' file

Once you have installed Node.js on your computer, you need to create a file called **package.json** and declare the npm packages that will be used for the minification process. Copy and paste the code below into your package.json file:

```
{
  "name": "local-minifier",
  "version": "0.1.0",
  "description": "A local environment to minify CSS, JS, & HTML
files",
  "license": "Apache-2.0",
  "author": "Amardeep Rai",
  "repository": {
    "type": "git",
    "url": "https://github.com/amardeeprai/local-minifier"
  },
  "devDependencies": {
    "del": "^2.0.2",
    "gulp": "^3.9.0",
    "gulp-autoprefixer": "latest",
    "gulp-csso": "^2.0.0",
    "gulp-htmlmin": "^3.0.0",
    "gulp-sass": "^2.0.0",
    "gulp-uglify": "^2.0.0",
    "run-sequence": "^1.2.2"
  },
  "engines": {
    "node": ">=0.12.0"
  }
}
```

3. Create a 'gulpfile.js' file and add the required code

Next, create a file called gulpfile.js. Once this is done we can start adding the code that will automate the minification process.

Declare the required npm modules

The first thing that you will have to do is declare the npm modules that will be required by Gulp to automate processes. This can be done by inserting the below code into the gulpfile:

```
'use strict';

var autoprefixer = require('gulp-autoprefixer');
var csso = require('gulp-csso');
var del = require('del');
var gulp = require('gulp');
var htmlmin = require('gulp-htmlmin');
var runSequence = require('run-sequence');
var sass = require('gulp-sass');
var uglify = require('gulp-uglify');
```

Declare supported browsers

When creating your CSS files, it is always good practice to make sure that you add prefixes to any styles for cross-browser compatibility. By adding the below code, you are declaring the browsers that you want to support (which will be used later when you create the gulp task to minify the CSS files):

```
// Set the browser that you want to support
const AUTOPREFIXER_BROWSERS = [
  'ie >= 10',
  'ie_mob >= 10',
  'ff >= 30',
  'chrome >= 34',
  'safari >= 7',
  'opera >= 23',
  'ios >= 7',
  'android >= 4.4',
  'bb >= 10'
];
```

Minifying CSS files

To create the CSS files, there are two options that are available based on:

i. if you are using SASS

ii. if you are using pure CSS

i. If you are using SASS

If you are using SASS files you can use the code below to compile and minify the final CSS file. The below code first declares a new gulp task, '**gulp styles**', which runs the following sequence:

1. the source SASS file is declared
2. the SASS is compiled
3. prefixes are added for cross-browser compatibility
4. the code is minified
5. the final CSS file is output to the declared destination.

```
// Gulp task to minify CSS files
gulp.task('styles', function () {
  return gulp.src('./src/sass/styles.scss')
    // Compile SASS files
    .pipe(sass({
      outputStyle: 'nested',
      precision: 10,
      includePaths: ['.'],
      onError: console.error.bind(console, 'Sass error:')
    }))
    // Auto-prefix css styles for cross browser compatibility
    .pipe(autoprefixer({browsers: AUTOPREFIXER_BROWSERS}))
    // Minify the file
    .pipe(csso())
    // Output
    .pipe(gulp.dest('./dist/css'))
});
```

ii. If you are using pure CSS

If you are using pure CSS files you can use the code below to compile and minify the final CSS file. This gulp task runs the following sequence:

1. the source CSS file is declared
2. prefixes are added for cross-browser compatibility
3. the code is minified
4. the final CSS file is output to the declared destination.

```
// Gulp task to minify CSS files
gulp.task('styles', function () {
  return gulp.src('./src/css/styles.css')
    // Auto-prefix css styles for cross browser compatibility
    .pipe(autoprefixer({browsers: AUTOPREFIXER_BROWSERS}))
    // Minify the file
    .pipe(csso())
    // Output
    .pipe(gulp.dest('./dist/css'))
});
```

Minifying JS files

The below code first declares a new gulp task, '**gulp scripts**', which runs the following sequence:

1. the source directory for JS files is declared
2. any JS files in the declared folder are minified
3. the minfied JS files are output to the declared destination.

```
// Gulp task to minify JavaScript files
gulp.task('scripts', function() {
  return gulp.src('./src/js/**/*.js')
    // Minify the file
    .pipe(uglify())
    // Output
```

```
    .pipe(gulp.dest('./dist/js'))
});
```

Minifying HTML files

The below code first declares a new gulp task, **'gulp pages',** which runs the following sequence:

1. the source directory for HTML files is declared
2. any HTML files in the declared folder are minified
3. the minfied HTML files are output to the declared destination.

```
// Gulp task to minify HTML files
gulp.task('pages', function() {
  return gulp.src(['./src/**/*.html'])
    .pipe(htmlmin({
      collapseWhitespace: true,
      removeComments: true
    }))
    .pipe(gulp.dest('./dist'));
});
```

Create a single task to minify CSS, JS, & HTML

The below code first declares a new gulp task, **'gulp clean',** which deletes everything in the output folder (this is to ensure that only the required files are in the output folder).

The code then declares a task, **'gulp default',** which runs the minification tasks in sequence:

1. 'gulp clean' — to delete everything in the destination folder
2. 'gulp styles' — to minify the CSS file
3. 'gulp scripts' — to minify any JS files
4. 'gulp pages' — to minify any HTML files

```
// Clean output directory
gulp.task('clean', () => del(['dist']));

// Gulp task to minify all files
gulp.task('default', ['clean'], function () {
  runSequence(
    'styles',
    'scripts',
    'pages'
  );
});
```

4. Initialise your local environment

Once you have created the two files, package.json and gulpfile.js, you are almost ready to go! Before you can begin minifying files, you must download the npm modules that are declared in your package.json file. This can be done by:

1. opening a terminal window
2. changing into the directory that you have saved the package.json and gulpfile.js files
3. running the following command:

```
npm install
```

If this does not work you may need administrator permissions on your computer, you can run the following command in that instance:

```
sudo npm install
```

A new folder will appear in the same directory called '**node_modules**'.

5. Setup your file structure

Now that the Gulp tasks are declared, you need to create a new folder named '**src**', and within that folder, you need to create two additional folders: '**js**' and '**css**'.

In the '**js**' folder you can add as many JS files as you want, with any name. In the '**css**' folder you should add either the file '**styles.scss**' or '**styles.css**' depending on how you decided to minify your CSS in step 3.

6. Start minifying

When you're at this step, and you've added your HTML files, you are ready to start minifying. Open your terminal window and change into the root directory of your project and run the command:

```
gulp
```

This should start compiling all of your CSS, JS, and HTML files and output them into a folder called '**dist**'.

You can also run the following commands to compile CSS, JS, and HTML files individually:

```
gulp styles

gulp scripts

gulp pages
```

7. Enjoy your minified files

That's all! You should now have optimised and minified files which will mean faster page-speed and improvements in SEO.

You can grab a copy of the code above by visiting:
https://github.com/amardeeprai/local-minifier

. . .

This article was originally published on amardeeprai.com