Minify CSS

- Loading unminified CSS
- Measure
- CSS Minification with webpack
- Verify
- Next steps and resources

CSS files can contain unnecessary characters, such as comments, whitespaces, and indentation. In production, these characters can be safely removed, to reduce file size without affecting how the browser processes the styles. This technique is called **minification**.

Loading unminified CSS

Take a look at the following CSS block:

```
body {
  font-family: "Benton Sans", "Helvetica Neue", helvetica, arial, sans-serif;
  margin: 2em;
}

/* all titles need to have the same font, color and background */
h1 {
  font-style: italic;
  color: #373fff;
  background-color: #000000;
}

h2 {
  font-style: italic;
  color: #373fff;
  background-color: #000000;
}
```

This content is easy to read, at the cost of producing a larger than necessary file:

- It uses spaces for indentation purposes and contains comments, which are ignored by the browser, so they can be removed.
- The `<h1>` and `<h2>` elements have the same styles: instead of declaring them separately: "`h1 {...} h2 {...}`" they could be expressed as "`h1, h2{...}`".
- The **background-color**, `#000000` could be expressed as just `#000`.

After making these changes, you would obtain a more compact version of the same styles:

```
body{font-family:"Benton Sans","Helvetica Neue",helvetica,arial,sans-serif;margin:2em}h1,h2{font-style:italic;color:#373fff;background-color:#000}
```

You probably don't want to write CSS like that. Instead, you can write CSS as usual, and add a minification step to your build process. In this guide, you'll learn how to do it by using a popular build tool: webpack.

Measure

You'll apply CSS minification to a site that has been used in other guides: Fav Kitties. This version of the site uses a cool CSS library: animate.css, to animate different page elements when a user votes for a cat □□.

As a first step, you need to understand what would be the opportunity after minifying this file:

1. Open the measure page.
2. Enter the URL: `https://fav-kitties-animated.glitch.me` and click **Run Audit**.
3. Click **View report**.
4. Click on **Performance** and go the **Opportunities** section.

The resulting report shows that up to **16 KB** can be saved from the **animate.css** file:

{% Img src="image/admin/RFMk5OMAIvOlkUZJTsh4.png", alt="Lighthouse: Minify CSS opportunity.", width="800", height="172", class="screenshot" %}

Now inspect the content of the CSS:

1. Open the Fav Kitties site in Chrome. (It might take a while for Glitch servers to respond the first time.) {% Instruction 'devtools-network', 'ol' %}
2. Click the **CSS** filter.
3. Select the **Disable cache** checkbox. {% Instruction 'reload-app', 'ol' %}

{% Img src="image/admin/WgneNAyftk8jneyXxMih.png", alt="DevTools CSS unoptimized trace", width="800", height="138" %}

The page is requesting two CSS files, of **1.9KB** and **76.2KB** respectively.

1. Click **animate.css**.
2. Click the **Response** tab, to see the file contents.

Note that the stylesheet contains characters for whitespaces and indentation:

{% Img src="image/admin/UEB5Xxe5IHhGtMx3XfKD.png", alt="DevTools CSS unoptimized response", width="800", height="286" %}

Next, you'll add some webpack plugins to your build process to minify these files.

{% Aside 'note' %} **Note:** the previous Lighthouse report only lists `animate.css` as an opportunity for minification. Minifying `style.css` will also save some bytes, but not enough for Lighthouse to consider it a significant savings. However, minifying CSS is a general best practice; so it makes sense to minify all of your CSS files. {% endAside %}

CSS Minification with webpack

Before jumping into the optimizations, take some time understanding how build process for the Fav Kitties site works:

{% Glitch { id: 'fav-kitties-animated', path: 'webpack.config.js', previewSize: 0 } %}

By default, the resulting JS bundle that webpack produces would contain the content of the CSS files inlined. Since we want to maintain separate CSS files, we are using two complementary plugins:

- mini-css-extract-plugin will extract each style sheet into its own file, as one of the steps of the build process.
- webpack-fix-style-only-entries is used to correct an issue in wepback 4, to avoid generating an extra JS file for each CSS file listed in **webpack-config.js**.

You will now make some changes in the project:

1. Open the Fav Kitties project in Glitch. {% Instruction 'source', 'ol' %} {% Instruction 'remix', 'ol' %} {% Instruction 'console', 'ol' %}

To minify the resulting CSS, you'll use the optimize-css-assets-webpack-plugin:

1. In Glitch console, run `npm install --save-dev optimize-css-assets-webpack-plugin`.
2. Run `refresh`, so the changes are synchronized with the Glitch editor.

Next, go back to the Glitch editor, open the **webpack.config.js** file, and make the following modifications:

Load the module at the beginning of the file:

```
const OptimizeCSSAssetsPlugin = require("optimize-css-assets-webpack-plugin");
```

Then, pass an instance of the plugin to the **plugins** array:

```
  plugins: [
    new HtmlWebpackPlugin({template: "./src/index.html"}),
    new MiniCssExtractPlugin({filename: "[name].css"}),
    new FixStyleOnlyEntriesPlugin(),
    new OptimizeCSSAssetsPlugin({})
  ]
```

After making the changes a rebuild of the project will be triggered. This is how the resulting **webpack.config.js** will look like:

{% Glitch { id: 'fav-kitties-animated-min', path: 'webpack.config.js', previewSize: 0 } %}

Next, you'll check the result of this optimization with performance tools.

Verify

{% Instruction 'preview' %}

If you got lost in any previous step, you can click here, to open an optimized version of the site.

To inspect the size and content of the files:

{% Instruction 'devtools-network', 'ol' %}

1. Click the **CSS** filter.
2. Select the **Disable cache** checkbox if it isn't already. {% Instruction 'reload-app', 'ol' %}

{% Img src="image/admin/id5kWwB3NilmVPWPTM59.png", alt="DevTools CSS unoptimized response", width="800", height="130" %}

You can inspect these files, and see that the new versions don't contain any whitespaces. Both files are much smaller, in particular, the animate.css has been reduced in **~26%**, saving **~20KB**!

As a final step:

1. Open the measure page.
2. Enter the URL of the optimized site.
3. Click **View report**.
4. Click on **Performance** and find the **Opportunities** section.

The report doesn't show "Minify CSS" as "Opportunity" anymore, and has now moved to "Passed Audits" section:

{% Img src="image/admin/zegn2qIHYYK58w1GhgYd.png", alt="Lighthouse Passed Audits for optimized page.", width="800", height="163" %}

Since CSS files are render-blocking resources, if you apply minification on sites that use large CSS files, you can see improvements on metrics like First Contentful Paint.

Next steps and resources

In this guide, we've covered CSS Minification with webpack, but the same approach can be followed with other build tools, like gulp-clean-css for Gulp, or grunt-contrib-cssmin for Grunt.

Minification can also be applied to other types of files. Check out the Minify and compress network payloads guide to learn more about tools to minify JS, and some complementary techniques, like compression.

□□□ #2
□ □□□ □□□ 2026-10-31 22:36:58 UTC
□ □□□ □□□ 2026-10-31 22:36:59 UTC