Optimizing CSS Extraction in Webpack 5

Improving Performance with MiniCssExtractPlugin

Kabui, Charles

2025-04-01

Table of Contents

Read at ToKnow.ai

Webpack 5[1] has `MiniCssExtractPlugin`[2], a powerful tool for extracting CSS from JavaScript modules. This optimization serves multiple purposes:

- **Improved Performance:** By extracting CSS, browsers can cache styles independently of JavaScript.
- **Prevents Flash of Unstyled Content (FOUC):** Ensuring styles load before rendering reduces layout shifts.

[1] Webpack 5
[2] MiniCssExtractPlugin

- **Optimized First Content Paint (FCP):** Critical styles load faster, improving perceived performance.
- **Modular CSS Bundling:** Different stylesheets can be generated per JavaScript module.

A basic implementation looks like this:

```
const MiniCssExtractPlugin = require("mini-css-extract-plugin");

module.exports = {
  plugins: [new MiniCssExtractPlugin()],
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: [MiniCssExtractPlugin.loader, "css-loader"],
      },
    ],
  },
};
```

The Problem: Too Many CSS Files

By default, `MiniCssExtractPlugin`[3] generates multiple CSS files—one per output chunk, based on how webpack handles code splitting [4]. This means each must be manually injected into the HTML document or handled using a plugin like `HtmlWebpackPlugin`[5].

The Solution: CSS Splitting Strategies

Instead of numerous render-blocking stylesheets, we can:

1. Generate one small critical CSS file (render-blocking)
2. Defer loading of larger stylesheets (non-render-blocking)

[3] MiniCssExtractPlugin
[4] GitHub Issue #42: Why Extract CSS?
[5] HtmlWebpackPlugin

Making CSS Non-Render Blocking

Normally, stylesheets block rendering:

```
<link rel="stylesheet" href="styles.css" />
```

To load styles asynchronously, use the `media="print"` trick:

```
<link
  rel="stylesheet"
  href="styles.css"
  media="print"
  onload="this.media='all'"
/>
```

This prevents blocking while ensuring styles apply once loaded.

Controlling CSS Splitting

Basic Approach: Single Bundle (Not Ideal)

This setup merges all styles into one file, reducing HTTP requests but may make the initial load slower:

```
module.exports = {
  optimization: {
    splitChunks: {
      cacheGroups: {
        styles: {
          name: "styles",
          type: "css/mini-extract",
          chunks: "all",
          enforce: true,
        },
      },
    },
  },
};
```

Per Entry File Splitting (Limited Control)

This splits styles based on entry points but lacks fine-grained control:

```
module.exports = {
  entry: {
    foo: "./src/foo",
    bar: "./src/bar",
  },
  optimization: {
    splitChunks: {
      cacheGroups: {
        fooStyles: {
          type: "css/mini-extract",
          name: "styles_foo",
          chunks: (chunk) => chunk.name === "foo",
          enforce: true,
        },
        barStyles: {
          type: "css/mini-extract",
          name: "styles_bar",
          chunks: (chunk) => chunk.name === "bar",
          enforce: true,
        },
      },
    },
  },
};
```

Advanced: Splitting by File Location

This method separates application styles from third-party vendor styles:

```
module.exports = {
  optimization: {
    splitChunks: {
      cacheGroups: {
        appCss: {
          name: "app.css",
          chunks: "all",
          enforce: true,
          test: (module) =>
            module.type === "css/mini-extract" &&
```

```
            !module.issuer?.resource?.includes("node_modules"),
        },
        vendorCss: {
          name: "vendor.css",
          chunks: "all",
          enforce: true,
          test: (module) =>
            module.type === "css/mini-extract" &&
            module.issuer?.resource?.includes("node_modules"),
        },
      },
    },
  },
};
```

> Tip
>
> *The key here is `module?.issuer?.resource`! You can also use `module?.resource`, but this is mostly null for webpack chunks.

This could be useful for [6] :

- Generating multiple theme-specific bundles.
- Separating vendor styles (Bootstrap, etc.) from application styles.
- Ensuring CSS Modules remain scoped correctly.

However, although this works, webpack documentation warns:

> "Note that `type` should be used instead of `test` in Webpack 5, or else an extra `.js` file may be generated besides the `.css` file."

Conclusion

Extracting CSS in Webpack 5 is essential for performance optimization. By intelligently splitting stylesheets, we can reduce render-blocking requests and improve First Content Paint.

[6] Support multiple instances of MiniCssExtractPlugin #45

Further Reading

- Webpack Mini CSS Extract Plugin - Why Extract CSS? [7]
- Handle CSS in Webpack

Disclaimer: *For information only. Accuracy or completeness not guaranteed. Illegal use prohibited. Not professional advice or solicitation.* ***Read more:*** */terms-of-service*

[7] GitHub Issue #42: Why Extract CSS?