CSS Critical Path

I've been looking at Critical Path CSS again as I work trying to improve the performance of this blog.

The idea behind Critical Path CSS (from now on, just Critical Path) is that we inline the CSS needed to render the first screen of content, what's called above the fold (borrowing a term from printed media) and load the remaining CSS later.

I know that inlining CSS or Javascript can negatively affect performance but in this case, reducing the number of HTTP request needed to render and only adding the smallest amount of CSS possible to render above the fold content improves performance, rather than degrade it.

Tools like Sitelocity's Critical Path Generator gives you both the critical path CSS and a script to load the remaining CSS for the page once the content has loaded.

You can also incorporate the critical path generation into your build process using libraries like critical.

Install it in your project like any other NPM module

```
npm i -D critical
```

Once it is installed, you can run the following task to generate the Critical Path CSS for two different screen resolutions and inline the resulting CSS in the HTML documents.

```
const gulp = require('gulp');
const log = require('fancy-log');
const critica'fancy-log'('critical').stream;

'critical'// Generate & Inline Critical-path CSS('critical', () => {
  return gulp
    .src('dist/*.html')
'critical'/*.html')
```

```
    .pipe(
      critical({
        base: 'dist/',
        inline: true,
        css: ['dist/styles/components.css', 'dist/styles/main.css'],
        dimensions: [
        {
          height: 200,
          width: 500,
        },
        {
          height: 900,
          width: 1200,
        },
      ],
      })
    )
    .on('error', err => {
      log.error(err.message);
    })
    .pipe(gulp.dest('dist'));
});
```

It is important that you test the results to make sure that the CSS you inline is enough to render the content in an acceptable way.

Loading CSS Asynchronously

We have the inline CSS needed to render the above-the-fold crtical path content but how do we load the rest of our CSS?

According to Scott Jehl, you can use the following link to load your CSS asynchronously as described in The Simplest Way to Load CSS Asynchronously

```
<link
  rel="preload"
  href="/path/to/my.css"
```

```
  as="style">
<link
  rel="stylesheet"
  href="/path/to/my.css"
  media="print"
  onload="this.media='all'">
```

This works with modern browsers but if you want to work with older browsers where the techniques above don't work, you will need something like loadCSS() from the Filament Group.

How about WordPress

Because WordPress runs on PHP it is not possible to run tools like Critical. However there are plugins that allow you to use third party remote tools to create the Critical CSS and use it on your pages.

Right now I'm leaning towards Autoptmize plus another plugin to use Critical or one of the critical path generators available.

There are libraries that provide PHP wrappers like Automatic critical css with Twig and PHP that give a good starting point on how to create a Critical CSS library in PHP without Node dependencies. Unfortunately, this particular example uses Twig to define what's the fold of the page so it wouldn't work out of the box with WordPress.

If you use the same or similar layout on your pages you can also use a third party tool like Sitelocity's (discussed earlier in the post) to generate your CSS and then paste it on the appropriate section of Autoptimize.

One final suggestion is to check if your cache plugin does any kind of performance optimizations and how those optimizations may conflict with existing plugins and functionality.

For example, placing all scripts on the footer may impact plugins that must remain on the header of the page. This happened to me with Automattic's AMP Plugin; it will error out if it's not placed on the header. There are ways to fix this issue but when I first started the scripts to footer plugin had no way to exclude plugins from moving to the footer so that caused errors. So be careful and test all

the changes you make to improve performance.

Further readings

- Understanding Critical CSS — Smashing Magazine
- Extract Critical CSS — web.dev
- Critical — Addy Osmani
- Critical Path CSS in WordPress — WP Speed Matters