How to Improve GOOGLE CORE WEB VITALS for WordPress

WP ROCKET
Superior WordPress Performance

Founded in 2014 in France, WP Media is a distributed company of more than 35 WordPress lovers living in the world's four corners. We develop plugins that make the web a better place — faster, lighter, and easier to use.

Check out our two main plugins:

- WP Rocket integrates more than 80% of web performance best practices automatically to reduce your website's loading time.
- Imagify reduces image file sizes without losing quality. By compressing your images, you speed up your website and boost your SEO.

Table of content

How to Improve Google Core Web Vitals for WordPress

INTRODUCTION

Why You Can’t Ignore Core Web Vitals

The Core Web Vitals have been a hot topic for a while now. It doesn’t come as a surprise, right?

In November 2020, Google announced that the page experience signals would be included as a new ranking factor starting from May 2021. The new ranking factor is the **Page Experience Update**. And the Core Web Vitals are part of it, alongside the existing search signals: mobile-friendliness, safe-browsing, intrusive interstitial guidelines, and HTTPS security.

You should care for it.

Now, it’s not only a matter of providing users with the best user experience — which should be essential anyways.

And it’s not only about achieving a 90+ PageSpeed Insight score for the sake of it — how fast your site loads can still be a different thing.

Overall, it’s about preserving — or even improving — your site’s organic visibility and traffic.

SEO has always been a reliable and long-term medium for driving traffic and conversions. And achieving good Core Web Vitals scores and optimizing your PSI performance grade may increase your SEO traffic. It won’t likely happen overnight. Yet, the organic landscape may change very soon, for better or worse. You should get ready.

That's why the bottom line is: you can't ignore Core Web Vitals.

In this ebook, you'll find everything you need to understand what these metrics are and how to test and optimize them on WordPress. Everything in plain English, we promise!

You'll also get the cheat sheets to recap all the actions required to improve your Core Web Vitals performance and address the PageSpeed Insights recommendations.

We hope you'll find it helpful and practical — we do care about it!

CHAPTER 1

What Are Google Core Web Vitals?

Core Web Vitals are a new initiative from Google designed to measure and improve user experience on the web. Instead of focusing on generic metrics like how long it takes your entire website to load, Core Web Vitals focus on how your WordPress site's performance connects to delivering a high-quality user experience.

Users care about how fast they can start interacting with a page. That's precisely what the Core Web Vitals metrics aim to measure.

Currently, there are three Core Web Vitals: **Largest Contentful Paint** (loading performance), **Cumulative Layout Shift** (visual stability), and **First Input Delay** (interactivity).

According to Google, these metrics are the most important ones for providing a great user experience.

If you think that these names are confusing, and if you tend to mix one metric with another, don't worry! We'll explain each metric in the easiest way. We want you to understand what each Core Web Vital means and its impact on user experience.

It's the first step for improving the scores and your overall SEO and WordPress performance.

Explaining Largest Contentful Paint (LCP)

Largest Contentful Paint (LCP) measures how long it takes for the most meaningful content on your site to load – that's usually your site's hero section or featured image.

According to Google, how long it takes for a page's main content to load affects how quickly users perceive your site to load.

Practical example: you land on a page and don't see the top image fully displayed right away. You would be annoyed, right? You would even think about leaving the page right away. Here's why the Largest Contentful Paint metric is closely related to user experience — more than the overall site's loading time.

The LCP "element" is different for each site, and it's also different between the mobile and desktop versions of your site. Sometimes the LCP element could be an image, while other times, it could just be text.

If you're wondering what a good LCP time is, here are Google's thresholds:

- **Good** – Less than or equal to 2.5 seconds

- **Needs Improvement** – Less than or equal to 4.0 seconds
  Poor – More than 4.0 seconds.On a side note: LCP is very similar to First Contentful Paint (FCP), another metric included in PageSpeed Insights.

The key difference is that LCP measures when the "main" content loads. FCP is focused on just when the "first" content loads — which could be a splash screen or loading indicator, that's a less relevant user-experience element.

LEARN HOW TO IMPROVE LCP ON WORDPRESS

Explaining First Input Delay (FID)

First Input Delay (FID) measures the time between when a user interacts with something on your page (e.g., clicking a button or a link) and when their browser can begin processing that event.

Practical example: if you click on a button to expand an accordion section, how long does it take for your site to respond to that and show the content?

First Input Delay is probably the most complicated metric to understand and optimize for, also because it's heavily affected by JavaScript.

Let's say that you land on a site from mobile and click on a link, but you don't get an immediate response. It could be because your phone is busy processing a large JavaScript file from that site.

Here's how Google defines FID scores:

- **Good** – Less than or equal to 100 ms
- **Needs Improvement** – Less than or equal to 300 ms
- **Poor** – More than 300 ms.

LEARN HOW TO REDUCE FID ON WORDPRESS

Explaining Cumulative Layout Shift (CLS)

The Cumulative Layout Shift (CLS) measures how much your site's content "shifts" or "moves around" as it loads.

Practical example: you're about to click on a link or CTA, and you can't do it because your content has just gone down after being loaded. You have a terrible user experience, and that's a layout shift. The same goes when you accidentally click the wrong button because the late-loading content caused a button to shift.

Or, have you ever been on a news website where the content in the article keeps shifting around as the site loads ads, and you are unable to keep reading? That's a layout shift, too.

You can see from yourself how the cumulative layout shift is super annoying for users and how they will have a poor experience.

Here's how Google defines the CLS scores:

- **Good** – Less than or equal to 0.1 seconds
- **Needs Improvement** – Less than or equal to 0.25 seconds
- **Poor** – More than 0.25 seconds.

LEARN HOW TO IMPROVE CLS ON WORDPRESS

Do Core Web Vitals Affect SEO as a Ranking Factor?

In May 2021, Google will start using Core Web Vitals as a ranking factor – therefore, these metrics could affect your SEO performance.

Core Web Vitals will be part of **the new Page experience signals**, together with HTTPS-security, safe-browsing, mobile-friendliness, and intrusive interstitial guidelines.

Core Web Vitals will affect both mobile and desktop organic results, as well as whether or not your site appears in Top Stories. Previously, your site needed to use AMP to appear in Top Stories. That will no longer be the case when Google rolls out the change, but your site will need to meet specific minimum Core Web Vitals scores to appear in Top Stories.

What's more, it seems like **all Core Web Vitals metrics need to be met** to improve organic ranking. And the Core Web Vitals score for **no-indexed pages may matter**, too.

In short: if you care about your SEO performance, improving your Core Web Vital scores is now mandatory.

Testing Core Web Vitals on WordPress

You can test and measure the Core Web Vitals with all of Google's tools for web developers, from PageSpeed Insights to Google Search Console, and much more.

You can learn more about testing each metric in our dedicated resource.

Now it's time to find out how to improve your Core Web Vitals' performance!

CHAPTER 2

What Affects Largest Contentful Paint (LCP) and How to Improve it

For WordPress sites, three factors affect LCP: slow server response times, render-blocking JavaScript and CSS, and slow resource load time.

Slow Server Response Times

The browser makes a request to the server, but the server takes too long to send the content requested. Since the browser doesn't receive the content quickly enough, it takes a while to get something rendered on your screen. As a result, load time is not great. The LCP score gets affected.

You'll fix the issue by improving your Time to First Byte, using a CDN, and establishing third-party connections early.

Render-blocking JavaScript and CSS

The browser makes a request and gets the content from the server. At this point, the browser will render the content and show it, right? Not so fast.

To render any content, the browser has to analyze (or *parse*) the HTML of the page and make it "readable" into the HTML structure of the page – that's the DOM tree. After that, the content will be rendered and fully displayed – unless some scripts and stylesheets block the HTML parsing. These scripts and stylesheets are the render-blocking resources.

As a result of this block, the parsing is delayed. Once again, the content you requested takes a bit before being loaded. The LCP performance gets affected again.

You'll tackle these issues by deferring and removing unused JS files. Don’t worry! You’ll find all the information you need in the next section.

Slow Resource Load Times

Other files can also cause poor performance and a bad user experience: images, videos, and block-level elements like HTML and CSS files.

As you already know, LCP is related to the elements at the top of the page. And this issue comes up precisely when these files are rendered above-the-fold and take too long to load. As a result, loading time and LCP are affected once again.

You'll manage the resource load times by optimizing images, minifying and compressing CSS, JS, HTML files, and preloading critical assets.

The bottom line: how fast the browser receives and renders the content requested determines the LCP score.

Let’s understand how to fix all these issues in detail.

How to Fix a Largest Contentful Paint Longer Than 4 s

You'll learn ten proven ways to improve your LCP score and follow the PageSpeed Insights recommendations.

For each suggestion, you'll find a piece of information about its performance impact – from low to high. The highest the impact is, the highest chance that the Largest Contentful Paint score will improve after following that specific recommendation.

1. Improve the Time to First Byte and Reduce Server Response Time

Performance Impact: high

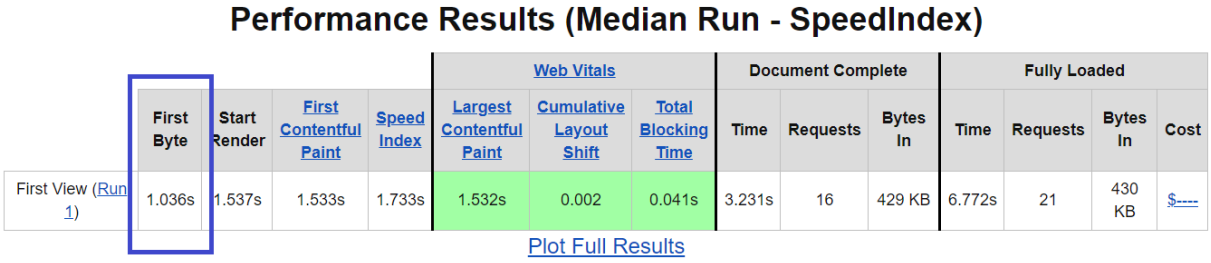
One of the main reasons for a bad LCP is a slow server response time.

You can measure your server response time by looking at the Time to First Byte (TTFB).

Every time you want to consume any piece of content, the browser sends a request to the server. The TTFB measures how long it takes for the browser to receive the first byte of content from the server.

By improving your TTFB, you'll improve your server response time and the LCP score.

Please note that a good TTFB should be under 200 ms – you can quickly check this metric by testing your URL's site on WebPageTest.



Performance Results (Median Run - SpeedIndex)

	First Byte	Start Render	First Contentful Paint	Speed Index	Web Vitals: Largest Contentful Paint	Web Vitals: Cumulative Layout Shift	Web Vitals: Total Blocking Time	Document Complete: Time	Document Complete: Requests	Document Complete: Bytes In	Fully Loaded: Time	Fully Loaded: Requests	Fully Loaded: Bytes In	Fully Loaded: Cost
First View (Run 1)	1.036s	1.537s	1.533s	1.733s	1.532s	0.002	0.041s	3.231s	16	429 KB	6.772s	21	430 KB	$----

Plot Full Results

WebPageTest example

There are two ways to fix a bad server time:

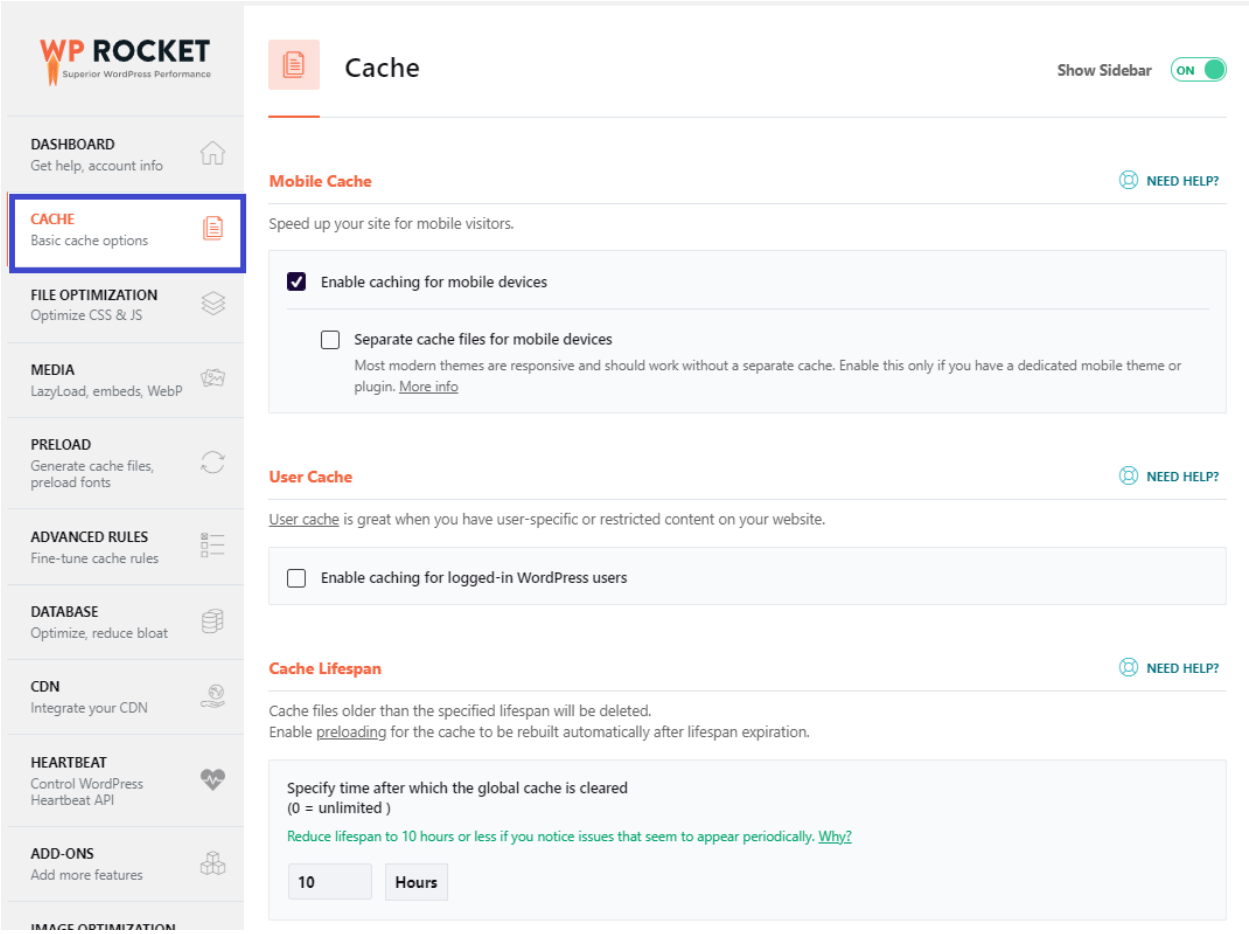
Enable Page Caching

By enabling page caching, your site's pages will be stored as HTML files on the server after the page is loaded for the first time. As a result, the content will be displayed faster. It's an easy and effective way to improve TTFB.

You can also choose one of the top WordPress hosting providers that include a server-level caching option.

WP Rocket can easily take care of page caching with no effort from your side.

A dedicated tab will allow you to enable mobile caching and set the options you prefer. WP Rocket enables 80% of web performance best practices automatically. So, if you're in doubt, you'll get covered anyway!



Cache tab

Choose a Fast Server Hosting Service

A fast hosting can make a huge difference in performance. And maybe it's time to upgrade your hosting plan!

As the first thing, your hosting provider should have servers close to the majority of your users. The closer your users are to the server, the fastest the data will be sent.

You should also choose the right server host type. A dedicated hosting server will ensure the fastest performance. Take into consideration how much traffic your site gets, and make your decision.

By enabling caching and choosing a fast hosting, you'll take care of the following PageSpeed Insights recommendations:

- Reduce server response times (TTFB)
- Serve static assets with an efficient cache policy.

2. Use a CDN

Performance Impact: medium

A CDN helps you reduce the length of time between the user request and the server response. This amount of time is the latency. The back and forth between the browser request and the server response is the round trip time (RTT).

If your users are located far from the server's location, it could take a while before all the assets (e.g., images, JS and CSS files, videos) are sent. Latency and RTT will be high and will affect loading time and the LCP score.

You already saw how the location of your server could affect your site's performance.

A CDN solves the issue thanks to a global network of servers. No matter where your users are located. Every time they request a page, they will receive the assets from the closest server. Simple as that.

RocketCDN is the best way to let your users access your site quickly and easily.

If you want to know more about the CDN benefits and the different types, you can read our article.

Choosing a CDN will help you address the following PageSpeed recommendations:

- Serve static assets with an efficient cache policy
- Enable text compression.

Please note that a CDN will address such recommendations only if properly configured. The default options might not be enough to improve performance as expected.

3. Defer JavaScript

Performance Impact: high

Render-blocking resources like JavaScript files are one of the main causes of a bad LCP score.

Deferring the JavaScript files will help you tackle the issue. In other words, you'll change the priority of the JS files being loaded.

Remember? The browser parses the HTML, builds the DOM tree, and then renders the page — unless there is any blocking-resource to slow the process down.

By deferring JavaScript, the browser will process and load the JS files only after parsing the HTML document and building the DOM tree. Since there won't be anything to block the process, rendering will be much faster — and the LCP metric will improve.

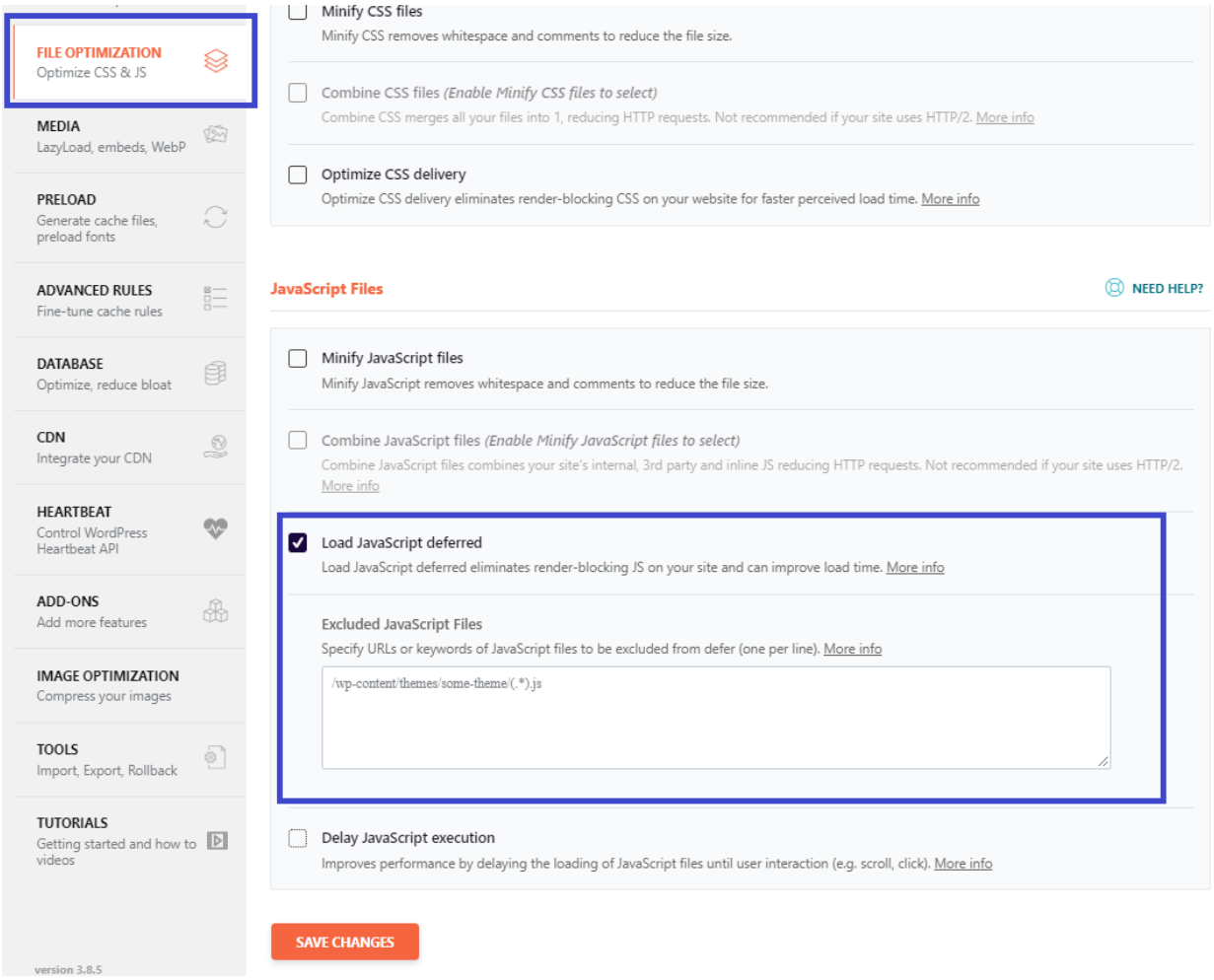
You can add the defer attribute to the JavaScript files so that the browser can detect the resources to defer. The browser will analyze the HTML and build the DOM tree with no interruption.

Here's an example of the defer attribute:

```
<script defer src="/example-js-script"></script>
```

The easiest way to manage the JavaScript resources is to take advantage of WP Rocket and its Load Javascript Deferred feature.

You can choose this option in the File optimization tab. What's more, you can easily exclude specific JS files from being deferred – in case the defer feature conflicts with any file.



File optimization Tab - Load JavaScript deferred

You'll address the "Eliminate render-blocking resources" PSI recommendation in a few clicks – even though the render-blocking resources issues don't stop here.

Let's move to the next point about tackling render-blocking resources.

4. Remove Unused JavaScript

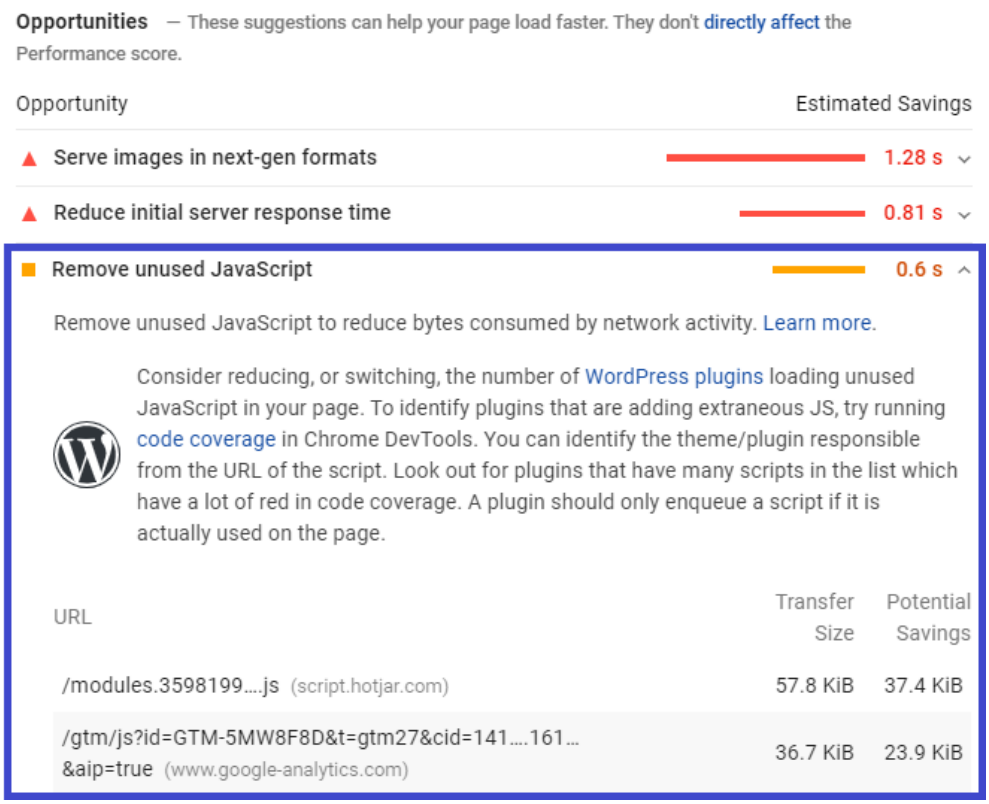
Performance Impact: medium

Another way to eliminate the render-blocking resources is to remove the JavaScript resources that are not used. They may not be used for two reasons:

- They're not used anymore on your site. They're still in the code but are completely useless.

- They aren't included in the above-the-fold content. Therefore, they're non-critical for building the DOM tree. Yet, these files have a reason to be there (e.g., Google Analytics tracking code).

You can find the list of the unused JS files in the PageSpeed report in the "Remove unused Javascript" section:



List of unused Javascript files - PageSpeed Insights Report

There are two ways to solve this issue in WordPress:

- **Load the JavaScript files only when needed**.
  For instance, the files should be loaded only on the pages that need that specific file – in any other case, the execution of JS should be disabled. You can take care of this aspect with plugins such as Perfmatters and Assets Cleanup.

- **Delay the JavaScript files.**
  The JavaScript files won't be loaded until the first user interaction (e.g., scrolling, clicking a button). If there's no user interaction, the JS files won't be loaded at all.

  By delaying JavaScript, the JS files won't be detected by Lighthouse nor listed in the "Remove unused Javascript files" recommendation.
  For instance, the Google Analytics tracking code is usually included in this

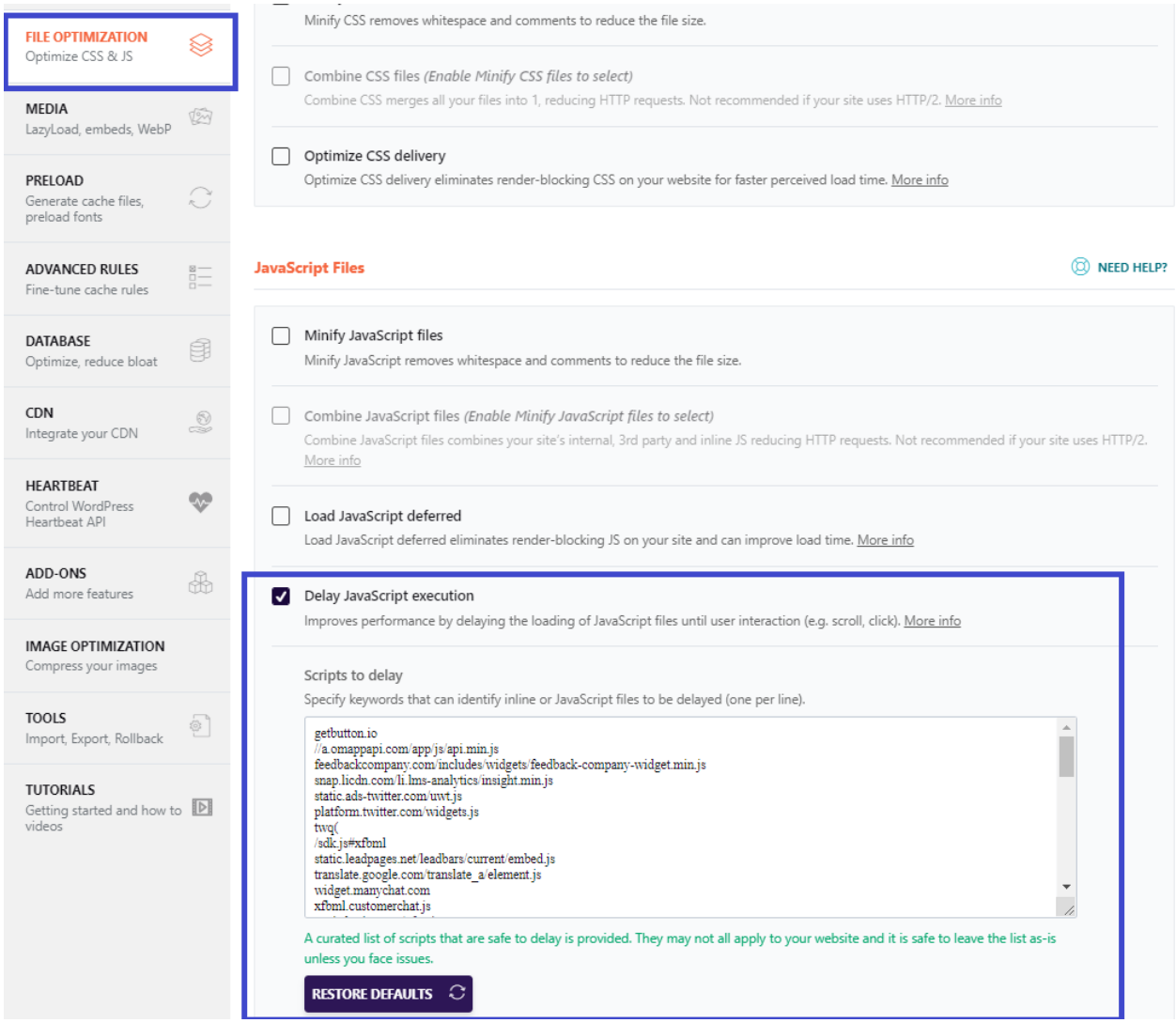
PageSpeed Insights recommendation. If you delay the JS files, the Google Analytics JS file won't be reported anymore.

Delaying JS files doesn't have the purpose of solving this PSI recommendation per se. However, it works well in addressing this PageSpeed audit and improving your LCP score. So, it's good for you to know.

Please note that not all the scripts from the PageSpeed recommendation list can be safely delayed.

If you're looking for a free plugin to delay JavaScript files, you can use Flying Scripts.

Another way to safely tackle any unused JavaScript is to take advantage of WP Rocket! The plugin allows you to delay the JavaScript execution in a few clicks from the File optimization tab. You can easily include the list of scripts to delay and let the plugin do the job for you:



File optimization tab - Delay JavaScript execution

As we mentioned, by removing unused Javascript files, you'll address the specific PageSpeed recommendation. Overall, you'll work towards "Eliminating render-blocking resources" and "Reducing javascript execution time".

Your LCP grade will get another boost.

5. Defer Non-Critical CSS and Inline Critical CSS

Performance Impact: medium

As for the JS files, you should also defer non-critical CSS — all the files not relevant for rendering the page. In other words, you should change the priority for these files, too. They will load after the browser has rendered the most relevant content on the page.

While deferring the CSS files, you should also inline critical CSS — the resources above-the-fold that need to be loaded as fast as possible. It means that you should identify the critical CSS (or *Critical Path CSS*) and inline them inside the HTML structure.

If you want to implement both actions on WordPress, here's how the process looks like:

1) First, you should extract and inline the Critical Path CSS (CPCSS) using one available generator tool. You can find one here.

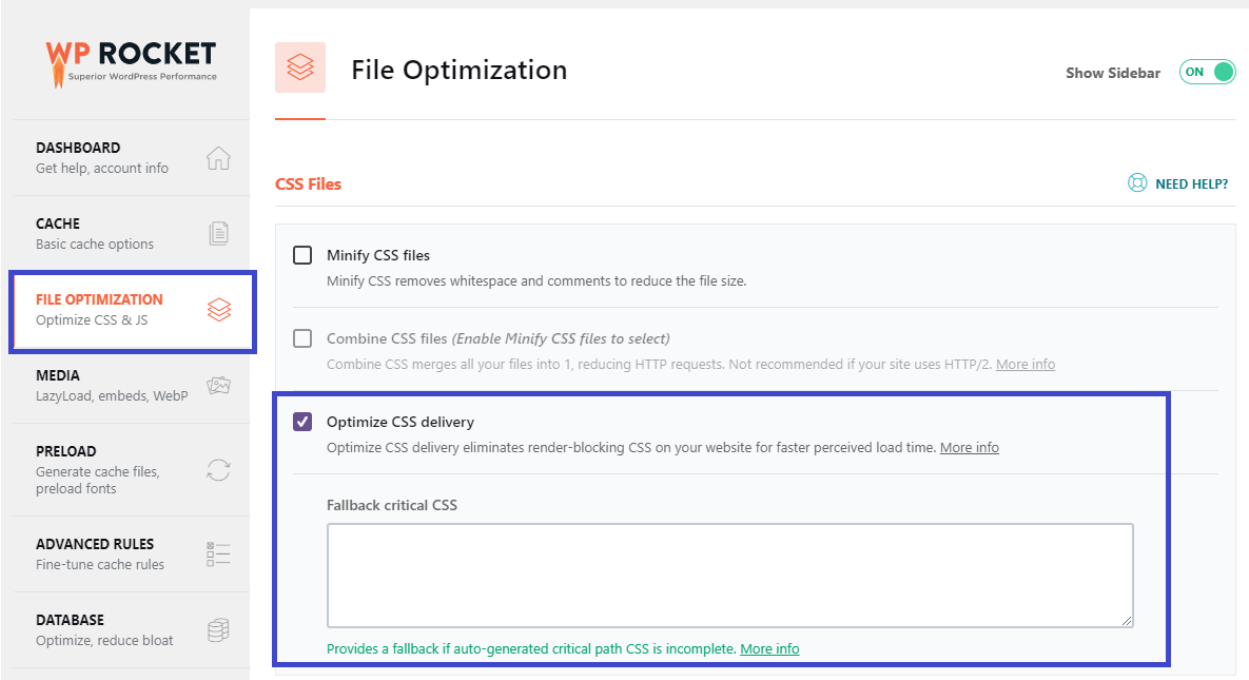
2) Then, you should load the rest of the classes asynchronously by applying the following pattern.

You can read more about the process in the dedicated Google resource.

Another tip is to avoid placing large non-critical CSS code in the <head> of the code.

If you want to take care of both critical and non-critical CSS quickly, you can take advantage of the Optimize CSS delivery feature provided by WP Rocket. This option will defer non-critical CSS and inline critical CSS — you don't need to do anything else.

In the File optimization tab, you can choose this option:



File optimization Tab - Optimize CSS delivery

By implementing these actions, you'll address the "Eliminate render-blocking resources" and "Avoid chaining critical requests" PageSpeed Insights recommendations.

6. Minify CSS and JS Files

Performance Impact: low

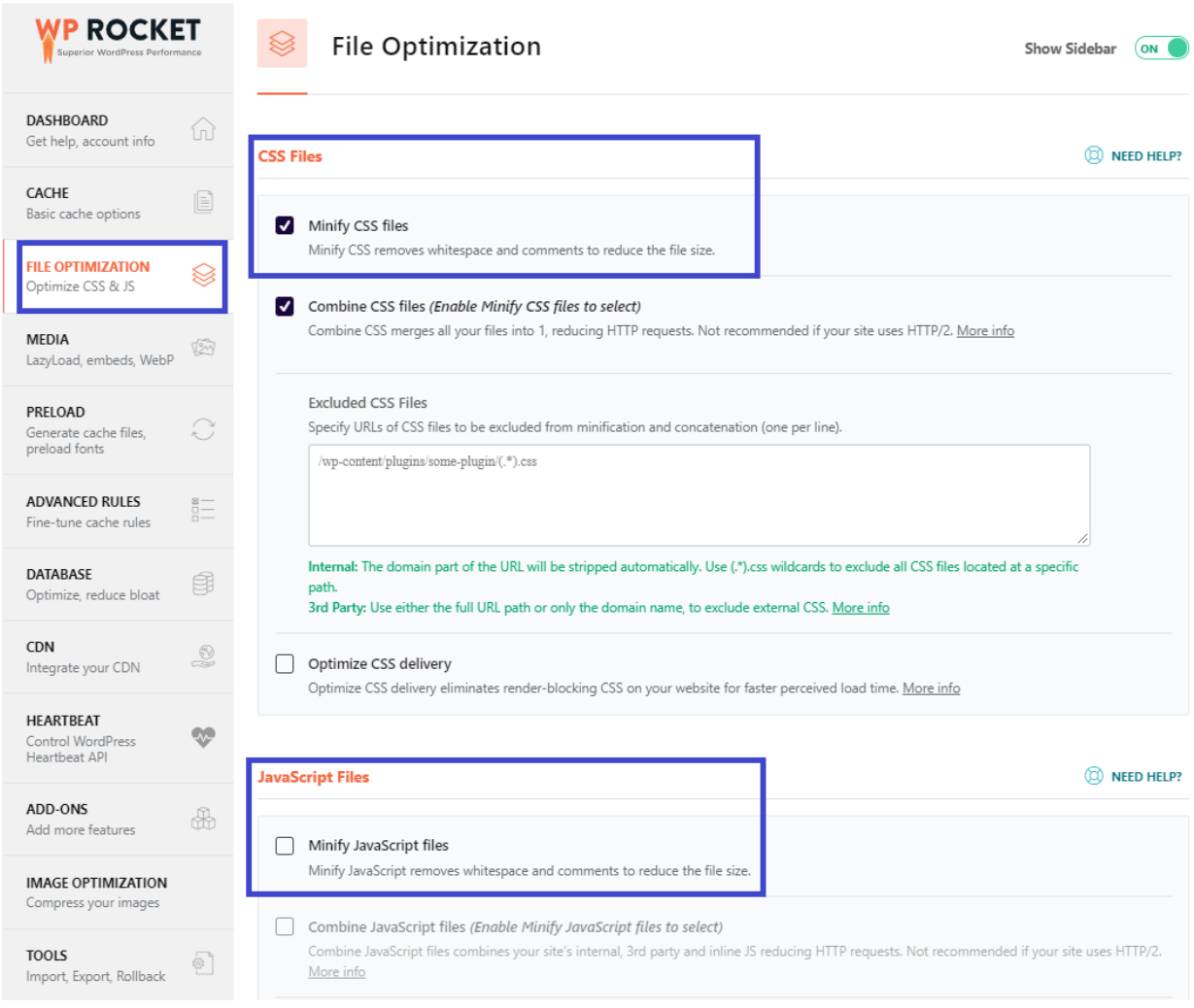
Another effective way to improve Largest Contentful Paint is to minify CSS and JS files.

Minification comes down to optimizing your code by making it more compact. It means to remove any white spaces, line breaks, and comments included in the code. As a result, minification will reduce CSS and JS files' size and make them load faster.

It sounds easy, but the reality is more complicated. It's not always simple to minify both file types and be sure to have excluded all the right resources – especially if you're not a developer. Either way, it's time-consuming.

The easiest and most effective way to take care of minification is to use a plugin like WP Rocket.

In the file optimization tab, you'll have the opportunity to minify both CSS and JavaScript files.



File optimization tab - Minifying CSS and JS files

You'll address the following PageSpeed Insights recommendations:

- Minify CSS
- Minify JS
- Avoid enormous network payloads.

7. Optimize Your Images

Performance Impact: high

Optimizing images is another relevant way to fix a bad LCP score.

Images are often the LCP element from mobile or desktop. By improving their loading time, you'll boost the Largest Contentful Paint performance.

Here's what you can do to fix any performance issues about images.

Compress and resize your images. You should reduce the file size without losing quality. The smaller the image dimension is, the faster the loading time will be.

To be clear: if you use a tool to optimize your images on the desktop, you will only optimize the original size. The images that you upload on WordPress won't be resized. As you may know, in WordPress, there are different image sizes. Unless you use a proper image optimization plugin, you won't optimize anything for performance.

For optimizing a few images, you could use a tool like ImageOptim. On the other hand, if you want to optimize more images and their thumbnails in bulk, Imagify is the perfect solution. You'll reduce your images' weight without sacrificing their quality. You'll save plenty of time!

Convert your images into new formats. Overall, Google recommends the WebP format. And that's why all WordPress image optimizer plugins now include the option to convert images to WebP. Other formats you may take into account are JPEG 2000 and JPEG XR. These formats are smaller than the JPEG, PNG, and GIF ones and help improve performance.

Use responsive images. You shouldn't use the same images' size for desktop and mobile. For instance, if the desktop image size is large, the mobile image size should be medium.

Page builders like Elementor allow users to upload different image sizes according to the device. Mobile image optimization is pretty essential, and the mobile score matters the most. Don't underestimate its impact on your LCP grade!

Exclude the LCP element from lazy-loading. While overall lazy-load helps improve loading time, it can make the LCP score worse, especially when the LCP element is an image and gets lazy-loaded. That's why excluding the LCP element from lazy-load and displaying it directly in the HTML of the page is an excellent way to improve the LCP score.

Use a static image instead of a slider. Sliders can be very heavy to load because of the code. On the other hand, a static image made by HTML code is lighter and faster.
By optimizing your images, you'll address the following PageSpeed Insights audits:

- Serve images in next-gen formats
- Properly size images
- Efficiently encode images
- Avoid enormous network payloads.

8. Compress Text Files

Performance Impact: high

You should also compress text files such as HTML, CSS, or JavaScript resources.

Compression means to “zip” your files in a smaller and lighter format so that they load faster. Once you reduce their size, the transfer between browser and server will be quicker. The browser will be able to load these resources faster. Load time and LCP will improve.

You can use compression formats such as GZIP and Brotli. On the one hand, GZIP is supported by most of the hosts. On the other one, Brotli is more performant and currently mostly recommended. You can read more about the differences in our dedicated article.

You can easily enable GZIP compression on WordPress by using a plugin. You can choose between different options, from the straightforward Enable Gzip Compression plugin to WP Rocket, which automatically includes GZIP compression. Also, some hosts enable GZIP compression by default.

Either way, you’ll address the “Enable text compression” PageSpeed recommendation.

9. Use Preload for Critical Assets

Performance Impact: low

At this point, you know how much the assets above the fold are critical for a good performance score. These critical resources can be fonts, images, videos, CSS, or JavaScript files.

To improve your LCP score, you should always make the critical assets load as fast as possible.
The Preload option comes in handy. It tells the browser to prioritize the load of these resources. In other words, the Preload prevents the browser from discovering and loading these critical files until much later.

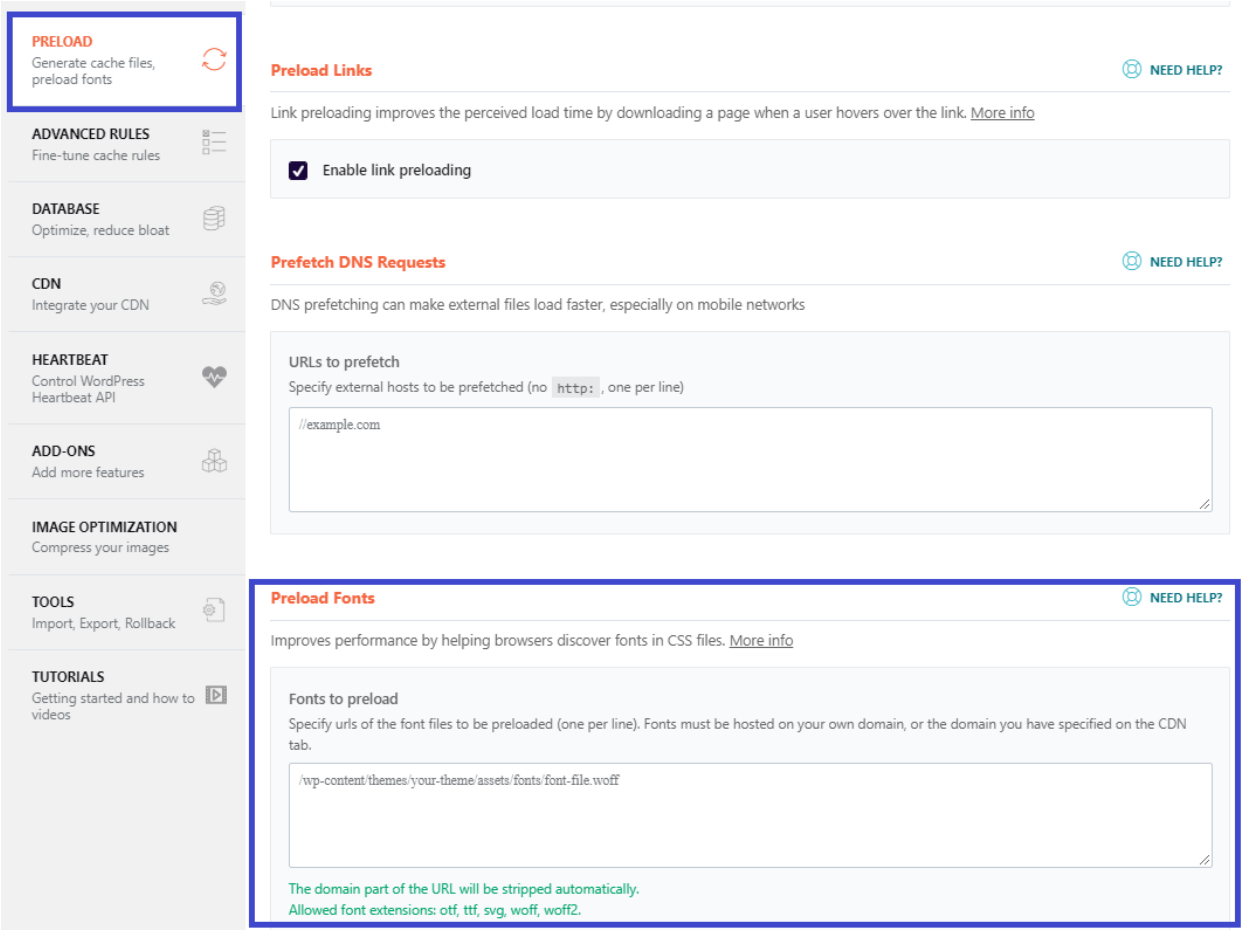
You can include the rel=”preload” in the code:

```
<link rel="preload" as="script" href="script.js">
<link rel="preload" as="style" href="style.css">
<link rel="preload" as="image" href="img.png">
<link rel="preload" as="video" href="vid.webm"
type="video/webm">
<link rel="preload" href="font.woff2" as="font"
type="font/woff2" crossorigin>
```

Source: Google

For **preloading images**, you can use a plugin like Perfmatters.

If you need to **preload fonts**, you can take advantage of the WP Rocket feature:



Preload tab - Preload fonts feature

What's more, WP Rocket also helps you **preload the CSS files** thanks to the Optimize CSS delivery feature mentioned above.

By using preload for critical assets, you'll address the "Preload key requests" PageSpeed recommendation.

10. Establish Third-party Connections Early

Performance Impact: low

Making the third-party connections faster is an additional way to optimize your LCP performance.

You should use the Preconnect option.

Let's say that there's a CSS or JS file requested from a third party, such as Facebook or Google Analytics. The browser will request the external resource.

If enabled, the Preconnect option tells the browser to establish a connection with the external domain as fast as possible. The browser will then handle the request in parallel with the ongoing rendering process.

You can include the `rel="preconnect"` in your code:

```
<link rel="preconnect" href="https://example.com">.
```

As an alternative, you can use a plugin as Perfmatters.

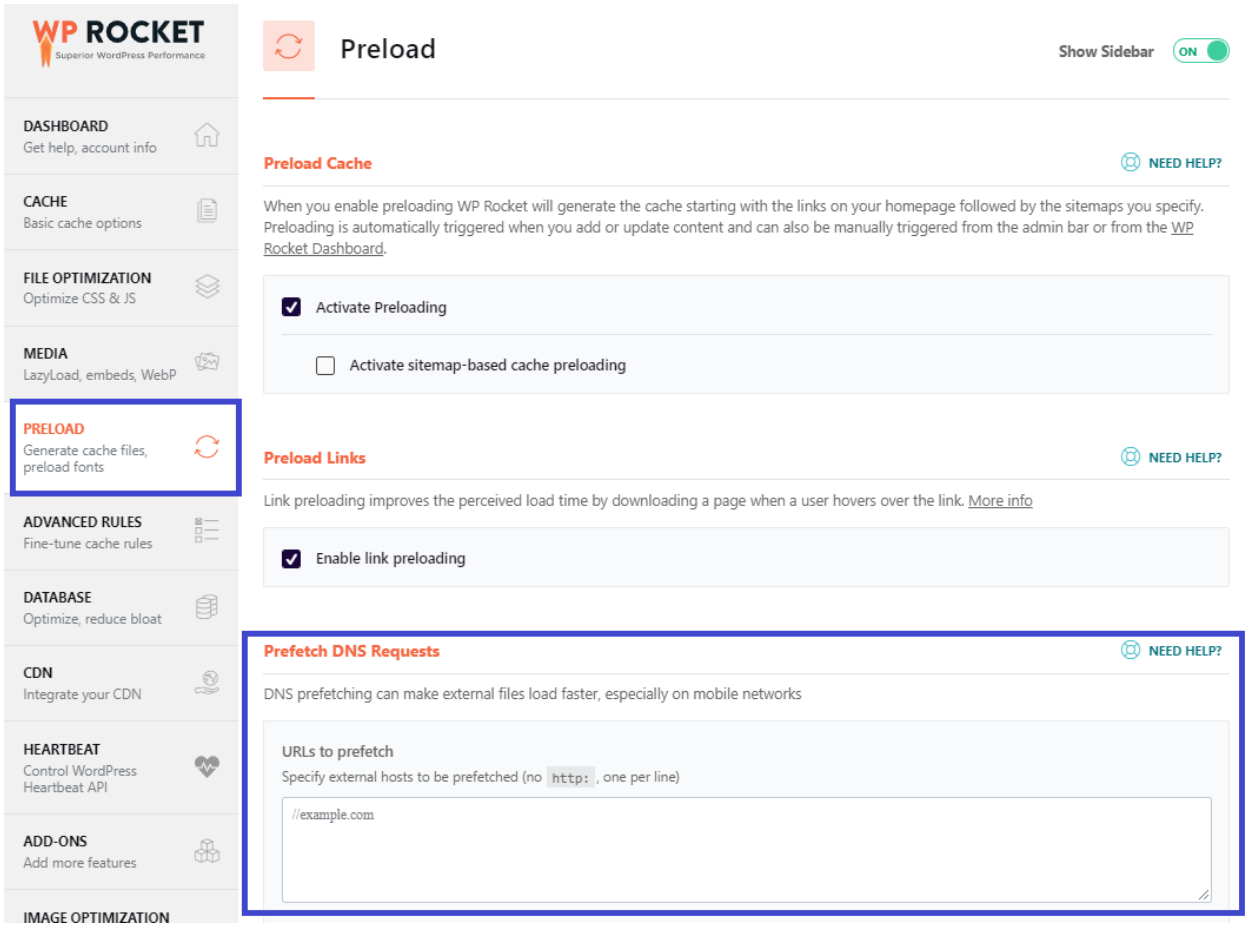
Since your browser may not support the preconnect option, it's always best to implement dns-prefetch as a fallback technique. You'll then resolve the DNS lookups faster. In other words, the external files will load more quickly, especially on mobile networks.

You can add the `rel="dns-prefetch"` to your code – as a separate tag from the preconnect attribute:

```
<head>
  …
<link rel="preconnect" href="https://example.com">
<link rel="dns-prefetch" href="https://example.com">
```

```
</head>
```

WP Rocket's Preload tab allows you to prefetch the DNS requests. You only have to specify the external hosts to be prefetched:



Preload tab - Prefetch DNS requests

By establishing third-party connections earlier, you'll improve the Time to First Byte and the server response time. You'll also address the "Preconnect to required origins" PageSpeed recommendation.

THE LCP CHEAT SHEET

ACTION	IMPACT ON PERFORMANCE	PSI RECOMMENDATION ADDRESSED	CAN WP ROCKET HELP?
Improve the Time to First Byte and Reduce Server Response Time	3	• Reduce server response times (TTFB) • Keep request counts low and transfer sizes small • Serve static assets with an efficient cache policy	✓ ✓ ✓
Defer JavaScript	3	• Eliminate render-blocking resources • Reduce the impact of third party code	✓ ✓
Optimize Your Images	3	• Serve images in next-gen formats • Properly size images • Efficiently encode images • Avoid enormous network payloads	✓ ✗ But Imagify can! ✓ ✓
Compress Text Files	3	• Enable text compression	✓
Use a CDN	2	• Serve static assets with an efficient cache policy • Enable text compression	✗ But RocketCDN can!
Remove Unused JavaScript	2	• Remove unused Javascript • Eliminate render-blocking resources • Reduce javascript execution time	✓ ✓ ✓
Defer Non-Critical CSS and Inline Critical CSS	2	• Eliminate render-blocking resources • Avoid chaining critical requests	✓ ✓
Minify CSS and JS Files	1	• Minify CSS • Minify JS • Avoid enormous network payloads	✓ ✓ ✓
Use Preload for Critical Assets	1	• Preload key requests	✓ *For fonts and CSS*
Establish Third-party Connections Early	1	• Reduce server response times (TTFB) • Preconnect to required origins	✓ *For the prefetch fallback option* ✗

CHAPTER 3

What Affects First Input Delay (FID) and How to Improve it

As we explained in Chapter 1, **FID is mainly impacted by JavaScript execution.**

When the browser is busy dealing with heavy JavaScript files, it can't process other requests, including the users' ones.
As a result, interactivity is poor; JavaScript execution times are longer; the main thread is busy and blocked. In short, the page can't respond to user inputs nor interactions.

We'll see different ways to fix these issues.

Since JavaScript is the key to improving FID, you should be aware that **having many plugins** – especially the JavaScript-based ones – could also affect the First Input Delay grade. You should avoid any unnecessary JavaScript execution on the pages where the plugin is not needed. It's also important to remove any plugin that is not essential.

Heavy WordPress themes can also affect the First Input Delay grade. They have more JS files, complex layouts, and an inefficient style that will affect the main thread – therefore, the FID performance.
That's why the less complexity the themes have, the better. And that's also why the tendency now is to simplify everything: layouts, animations, more native JS use vs. relying on complex libraries.

How to Reduce the First Input Delay on WordPress

Improving how the browser deals with JavaScript execution reduces the First Input Delay on WordPress and enhances the FID score. The goal is to make the process faster and smoother so that interactivity and responsiveness can get better.

There are several ways to improve the First Input Delay grade on WordPress:

- Reduce JavaScript execution time
- Optimize your page for interaction readiness
- Break up Long Tasks.

Let's see in detail what actions you should take and what's the performance impact.

1. Defer JavaScript

Performance Impact: high

As for optimizing the JavaScript execution, you should defer Javascript files.

By deferring JavaScript files, these render-blocking resources will be loaded after the browser has rendered the most relevant content – that is, the content needed to let users interact with the page.

As a result, the loading time will improve, as well as the FID grade.

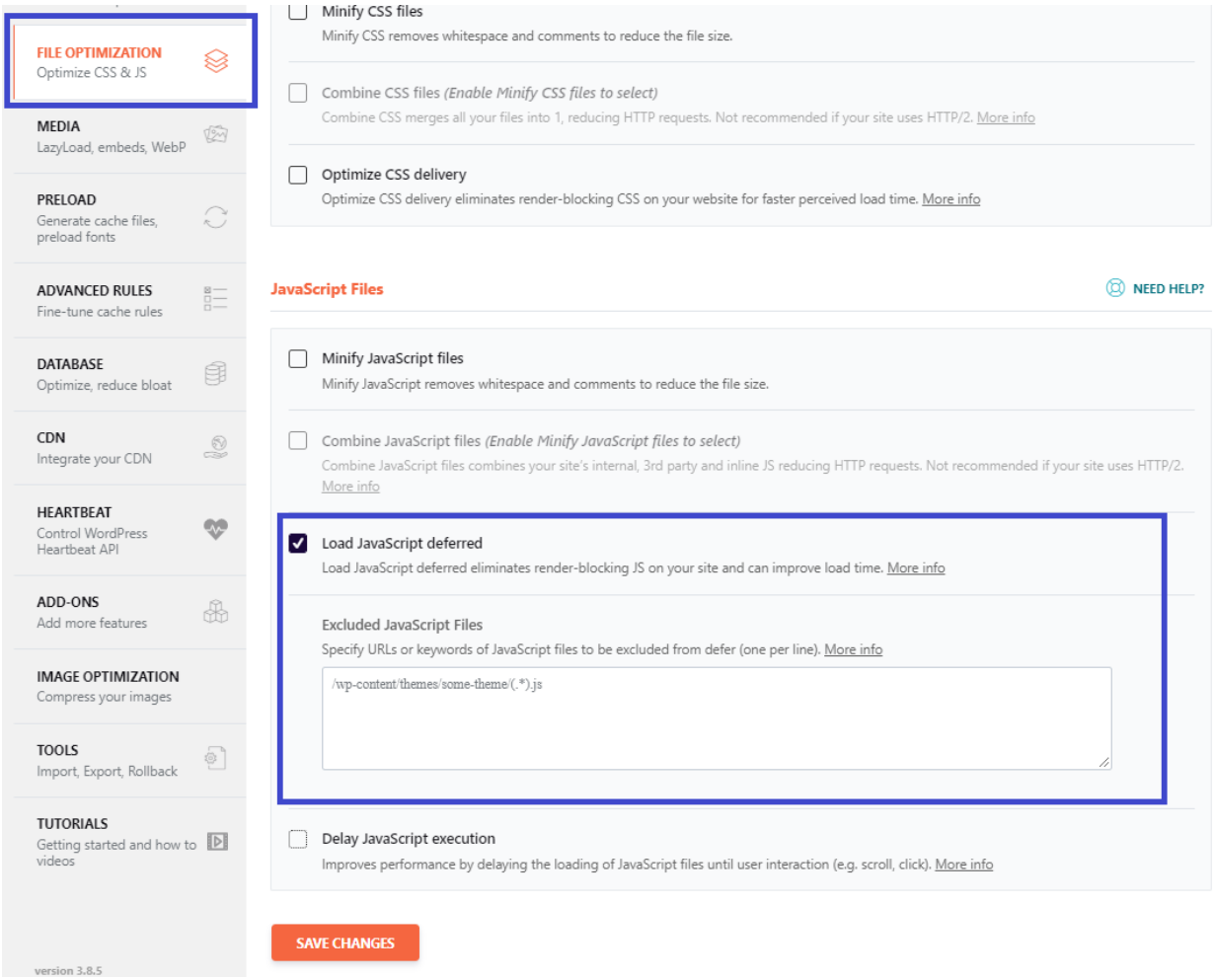
Once you have identified the JS resources to defer, you should add the defer attribute to the JavaScript files. The browser will then know which files to defer until the page rendering is complete.

Here's an example of the defer attribute:

```
<script defer src="/example-js-script"></script>
```

You can easily manage the JavaScript files' deferring with WP Rocket and its Load Javascript Deferred feature.

You'll find this option in the File optimization tab. You'll also be able to exclude specific JS files from being deferred – in case you need this option due to any conflict.



File optimization Tab - Load JavaScript deferred

You'll address the "Eliminate render-blocking resources" and "Reduce the impact of third party code" PageSpeed recommendations – even though the JS render-blocking resources issues don't stop here.

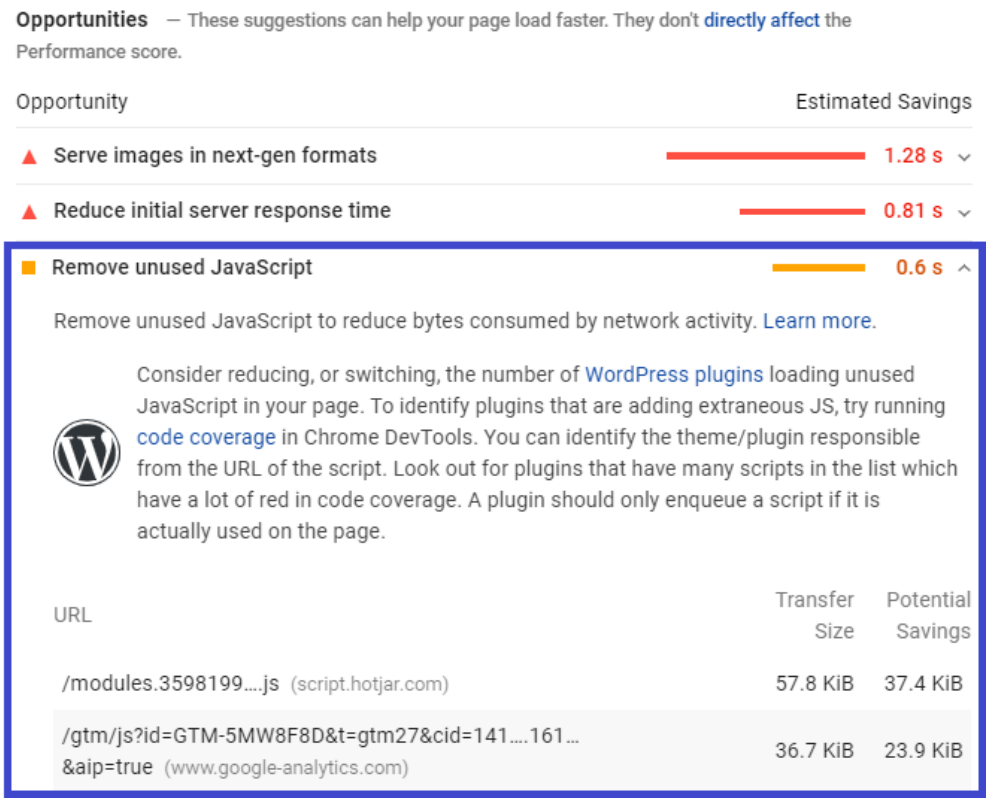
Keep reading to learn what other actions you should implement.

2. Remove Unused JavaScript

Performance Impact: medium

You can also tackle the JS issues by removing the unused JavaScript files that slow downloading time and make interactivity worse.
Unused JS files are the JavaScript resources not essential for rendering the page or not useful at all. Yet, these files are in the code, so you should manage them. Examples of unused JS files are the third-party JavaScript files such as the analytics and ads tracking codes.

The PageSpeed Insights report shows you the list of the unused JS files you should take care of:



List of unused Javascript files- PageSpeed Insights Report

You have two options to tackle unused Javascript files:

1. Load the JavaScript files only when needed.

You can use plugins such as Perfmatters and Assets Cleanup to load these files only when needed. The execution of JavaScript files should be disabled in any other case. As an additional tip, you may ask your theme and plugin developers to ensure that only the needed assets are loaded when their respective features are used.

2. Delay the JavaScript files.

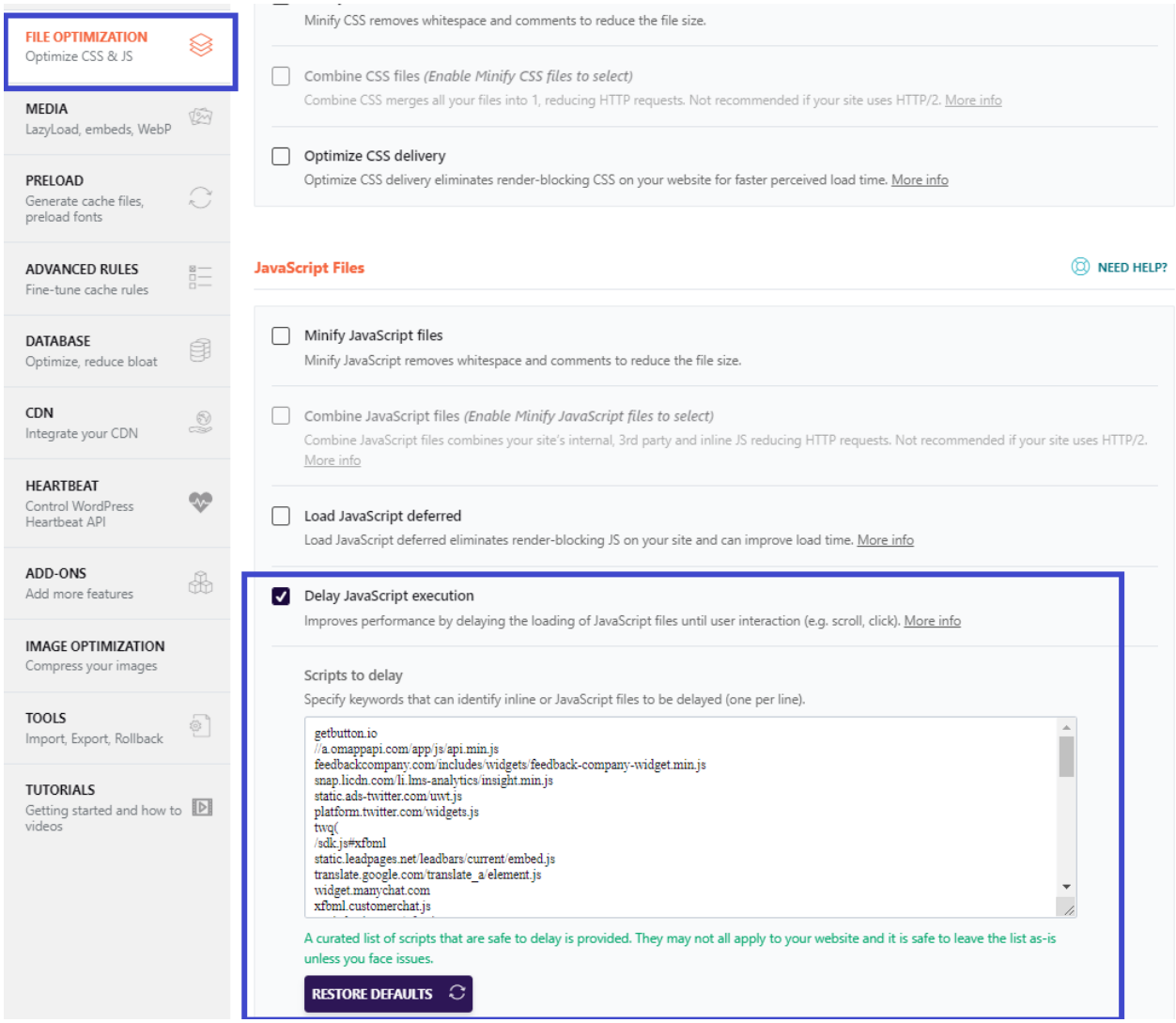
Delaying JavaScript resources means that the JavaScript files won't be loaded until the first user interaction (e.g., scrolling, clicking a button). In other words, no JS files will be loaded unless there's user interaction. It's important to keep in mind that not all the scripts from the PageSpeed recommendation list, like the one included above, can be safely delayed.

An additional advantage of delaying JavaScript is that Lighthouse won't detect any JS files. As a result, the tool won't include any of these JS resources in the "Remove unused Javascript files" recommendation.

To be clear: you shouldn't delay JS files to solve this PSI recommendation. You'll find more information about the main reason why you should delay JS in the next point. However, it's useful for you to know as an added value for improving your PSI score.

So, how can you delay JavaScript files? You can use a free plugin such as Flying Scripts.

On the other hand, you can take advantage of WP Rocket and its Delay JavaScript execution feature. The File optimization tab allows you to delay the JavaScript execution in a few clicks. You only have to include the list of scripts to delay. Simple as that!



File optimization tab - Delay JavaScript execution

Lastly, here's a list of other plugins that can help you to minimize unused JS. We recommend using them carefully:

- Plugin Organizer
- Asset CleanUp
- Gonzales

Removing unused Javascript files will address the specific PageSpeed recommendation listed in the report. You'll also address the "Eliminating render-blocking resources" and "Reducing javascript execution time" recommendations.

3. Delay JS Execution Time Until User Interaction

Performance impact: very high

You can optimize JavaScript resources and prioritize the scripts needed for interaction by delaying the JS files and their execution until user interaction.
In other words, no JavaScript files will be loaded until a user interaction, such as clicking a button or scrolling the content.

As explained above, you should delay all the JavaScript files that affect loading time and interaction for no reason, such as the unused JS files included in the previous section.

As you already know, you have different options to delay JavaScript files. You can use a free plugin such as Flying Scripts or take advantage of the Delay JavaScript execution feature option provided by WP Rocket – more details above.

This is how you'll address the "Reduce javascript execution time" PSI recommendation.

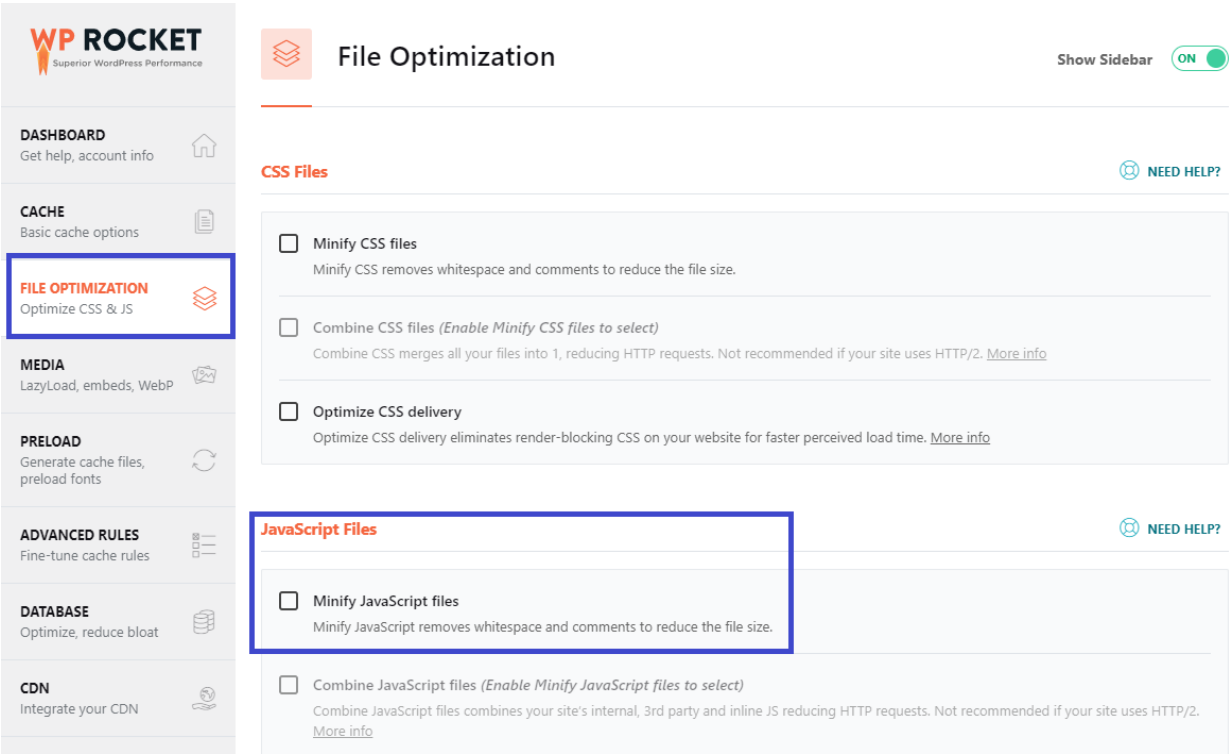
4. Minify JS

Performance impact: low

Another effective way to manage the JavaScript execution time goes through the minification of JavaScript files.

By minifying JS files, you'll remove any comments, line breaks, and white spaces included in the code. The goal is to make the files' size smaller and faster.

Minification can be time-consuming, and there's always the risk of missing out on anything. For these reasons, it's best to use a minification tool or use WP Rocket. WP Rocket is the easiest way to minify JS files in a few clicks. You only have to enable the Minify JavaScript file option in the file optimization tab.



File optimization tab - Minifying JS files

You'll address the following PageSpeed Insights recommendations:

- Minify JS
- Avoid enormous network payloads.

5. Async or Defer CSS

Performance impact: medium

The main thread work can have a significant impact on interactivity and FID performance. That's why one of the PSI recommendations is "Minimize main thread work." To address the issue and improve FID time, you should defer or async the CSS files. To be clear: defer and async mean the same.

In the Defer JavaScript section, you read why defer is essential to allow the browser to focus only on the resources essential to page rendering. The same goes for the CSS files that are render-blocking resources.

An option is to include the defer attribute to all the CSS files:

```
<script defer src="/example-css-script"></script>
```

And here's another 2-step process to make the CSS render-blocking resources load asynchronously:

1) Extract and inline the Critical Path CSS (CPCSS) using an available generator tool like this one.

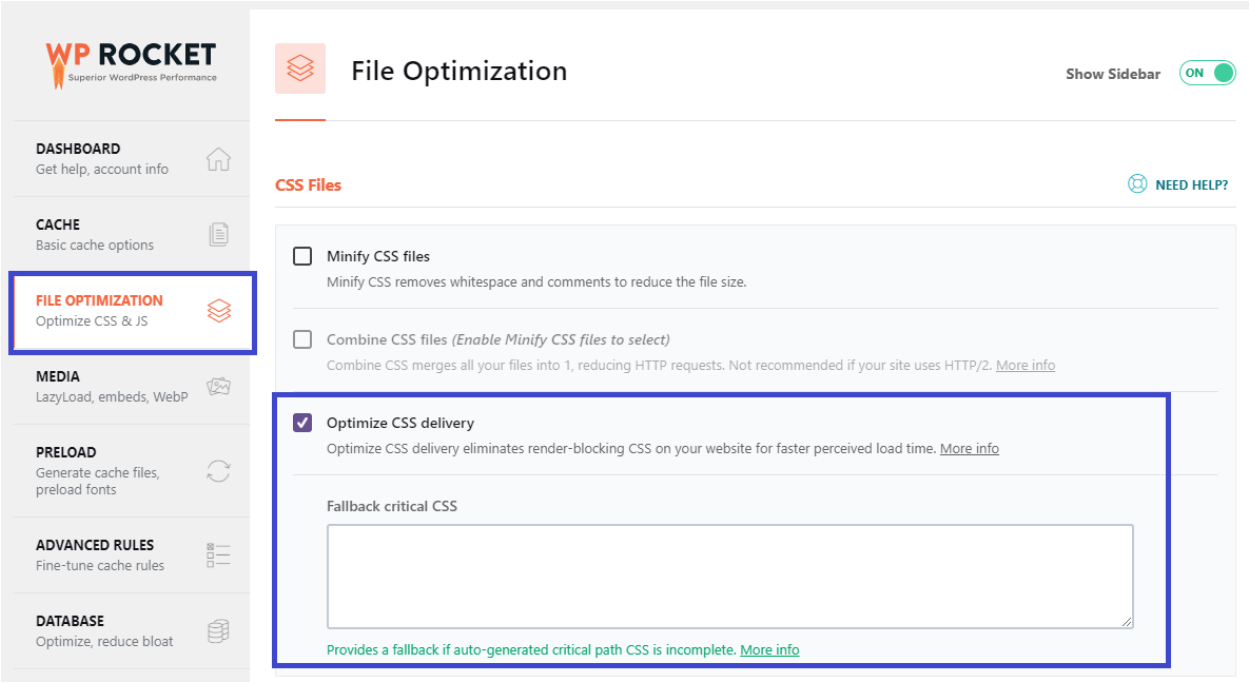
2) Load the rest of the classes asynchronously by applying the following pattern.

If you're looking for more detailed information, we recommend you read the dedicated Google resource.

An extra tip to keep in mind is to avoid placing large non-critical CSS code in the <head> of the code.

If you're looking for a faster and easier way to quickly take care of both critical and non-critical CSS, WP Rocket can help you. Our cache plugin offers the **Optimize CSS delivery feature** that defers non-critical CSS and inline critical CSS.

You'll remove all the render-blocking CSS resources by enabling the option in the File Optimization tab:



File optimization tab - Optimize CSS delivery

By implementing these actions, you'll take care once again of the "Eliminate render-blocking resources" PageSpeed Insights recommendations. You'll also address the "Avoid chaining critical requests" recommendation.

6. Compress text files

Performance impact: high

As you can guess at this point, compression is something you need to take care of. It goes without saying that “Enable text compression” is one of the PSI recommendations that apply to FID times.

By compression and reducing files’ size, the browser and the server will send over files faster. The browser will load these resources quicker.

The most common compression formats are Gzip and Brotli. Brotli is the most recommended format right now. You can read more about the two formats in our dedicated article.

The easiest way to enable Gzip compression on WordPress is using a plugin. You can choose between different options, from the Enable Gzip Compression plugin to WP Rocket, which includes GZIP compression by default. Keep in mind that some hosts enable GZIP compression automatically.

7. Break up Long Tasks

Performance impact: high

As we explained at the beginning of the article, when the main thread is busy and blocked, the FID grade is negatively affected, and the page can't respond to user inputs nor interactions.

The main thread is blocked because of the long tasks that the browser can’t interrupt – that is, all the tasks running longer than 50 ms. That’s why when the main thread is blocked, a page can't respond to user inputs, and responsiveness gets affected.

To solve this issue, you should split long-running scripts into smaller chunks that can be run in less than 50ms.

`Content-visibility` is a new powerful CSS property that can help boost the rendering performance by enabling the user agent to skip an element's rendering work until it is needed.
You can improve your load performance by applying `Content-visibility: auto; contain-intrinsic-size: 1px 5000px;` to elements where you want to delay the paint. Don't forget the second part: it's important to fix some usability issues.

Currently, this CSS property works only on Chrome and the majority of browsers based on it.

Lastly, we recommend reading two resources on this topic, about the long tasks and intensive JavaScript.

THE FID CHEAT SHEET

ACTION	IMPACT ON PERFORMANCE	PSI RECOMMENDATION ADDRESSED	CAN WP ROCKET HELP?
Delay the JS Execution Time Until User Interaction		• Reduce javascript execution time	✓
Defer JavaScript		• Eliminate render-blocking resources • Reduce the impact of third party code	✓ ✓
Compress Text Files		• Enable text compression	✓
Remove Unused JavaScript		• Remove unused Javascript • Eliminate render-blocking resources • Reduce javascript execution time	✓
Defer Non-Critical CSS and Inline Critical CSS		• Eliminate render-blocking resources • Avoid chaining critical requests	✓ ✓
Minify JS Files		• Minify JS • Avoid enormous network payloads	✓ ✓

CHAPTER 4

What Affects Cumulative Layout Shift (CLS) and How to Improve it

The most common factors of a poor Cumulative Layout Shift grade on WordPress are:

- Images and videos without dimensions
- Ads, embeds, and iframes without dimensions
- Web Fonts causing Flash of Unstyled Text (FOUT) or Flash of invisible text (FOIT)
- Actions waiting for a network response before updating DOM (especially for ads)
- Dynamically injected content (e.g., animations).

Keep in mind that CLS has the most significant impact on mobile — the most critical and challenging device for optimizing performance. There are several reasons, from a smaller viewport to a challenging mobile network and a weaker Central Processing unit (CPU).

Images and Videos Without Dimensions

Images and videos without dimensions are a common cause for a layout shift.

If you don't specify the width and height size attributes, the browser doesn't know how much space has to allocate while loading these elements. Likely, the space reserved won't be enough. As a result, once these elements are fully loaded, they will take more space than expected — the content already displayed will shift.

You can solve this issue by including image dimensions on images and video elements in different ways. We've got you covered in the dedicated section!

Ads, Embeds, and Iframes Without Dimensions

The same “dimension” issue goes for ads, embeds, and iframes. Once again, not reserving enough space makes these dynamic elements push down the content already displayed. Therefore, new layout shifts will occur on the page.

You'll manage this problem by assigning fixed dimensions to the ads and managing the size reserved for such elements through specific tactics.

Web Fonts Causing Flash of Unstyled Text (FOUT) or Flash of Invisible Text (FOIT)

Web fonts can cause layout shifts, plus a pretty unpleasant user experience while rendering the page. It’s about how slow the fonts load. You might face two different issues: Flash of Unstyled Text (FOUT) or Flash of invisible text (FOIT).

On the one hand, you could see the text on the page with a “not-so-good” style (FOUT). It’s because the custom font takes a bit to load. In the meantime, you’ll see the fallback font. Once the custom font is ready, it will replace the fallback one. You’ll then see the font changing on the page — and the content will inevitably shift.

On the other hand, you could wait a bit before seeing any text displayed. It’s because the custom font is still being loaded (FOIT). You’ll see the content on the page only after the custom fonts have been rendered. Once fully loaded, that content might cause a layout shift.

The main way to solve this issue is to preload fonts, as you’ll read in a minute.

Actions Waiting for a Network Response Before Updating DOM & Content Injected on the Page

Animations and dynamic content injected on the page — such as banners, ads, or Instagram feeds — can also cause a layout shift. Once again, it's because there's not enough space reserved for such elements.

At this point, you know how essential it is to allocate space for the elements that engage users and shouldn't ruin the user experience.

Let's see how to fix these problems.

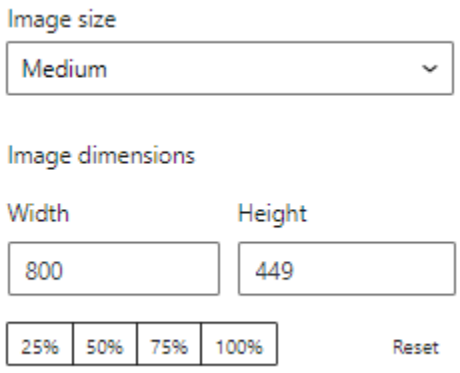
How to Fix a Cumulative Layout Shift More Than 0.1 S on WordPress

You'll go over different ways to improve a bad CLS score. For each of them, you'll find a piece of information about its performance impact – from low to high. The highest the impact is, the highest chance that the Cumulative Layout Shift grade will improve after following that specific recommendation.
As you'll see, some best practices don't include a specific solution – they're more about managing well space for ads and other crucial elements.

1. Include Width and Height Size Attributes on Images and Video Elements

Performance impact: high

One of the simplest ways to fix CLS is to include the width and height attributes on your images and video elements in your WordPress CMS:

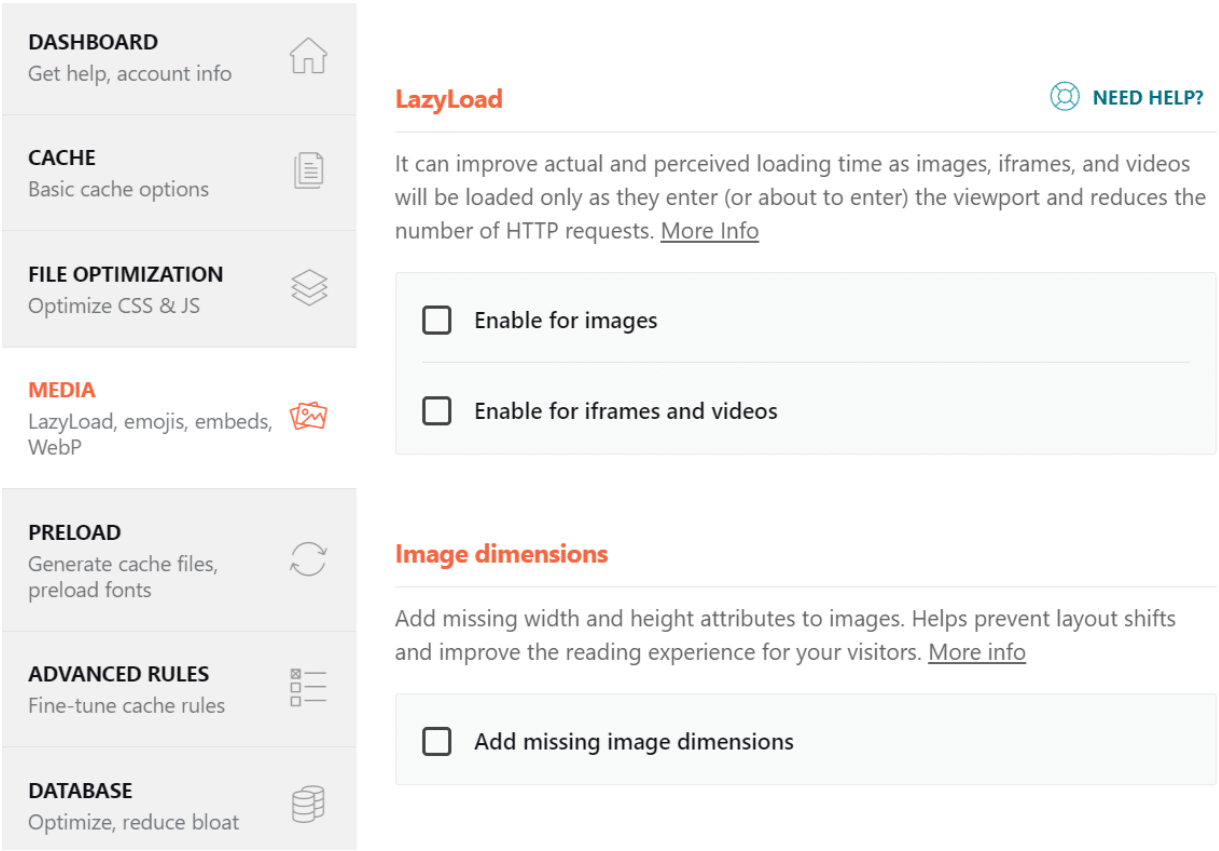


Setting image dimensions fixes CLS.

WordPress adds image dimensions by default. So, this action should be automatically solved.

In case you're facing any issue, keep in mind that WP Rocket automatically includes any missing "width" and "height" values to images.

You only have to select the "Add missing image dimensions" option in the Media tab. Fast and straightforward as that!



You can easily add missing image dimensions.

Another way to solve this issue is to take advantage of the CSS aspect ratio boxes and let the browsers set the default ratio of images.

Simply put, you should include the width or the height attribute and set the aspect ratio using CSS. The browser will figure out the missing attribute and get the image dimensions before rendering the page. By doing so, it will allocate the space needed while the image is loading. As a result, the content won't move around, and layout shifts will be avoided.
It's helpful information to keep in mind because many plugins, such as YouTube video embed ones, use aspect-ratio on their output.

Don't forget about responsive images! You can use the srcset attribute:

```
<img
width="1000"
height="1000"
src="puppy-1000.jpg"
srcset="puppy-1000.jpg 1000w, puppy-2000.jpg 2000w,
puppy-3000.jpg 3000w"
alt="Puppy with balloons"
/>
```

Source: Google

Thanks to the srcset attribute, the browser can choose between a set of images and related sizes. Keep in mind that images should use the same aspect ratio to set image size.

By including size images, you'll serve images with correct dimensions and address the PageSpeed Insights opportunity.

2. Preload Fonts (And Optimize Them)

Performance impact: low (high only if the site had large text)

As we explained, if web fonts don't load fast, they cause a terrible user experience and affect the CLS grade.

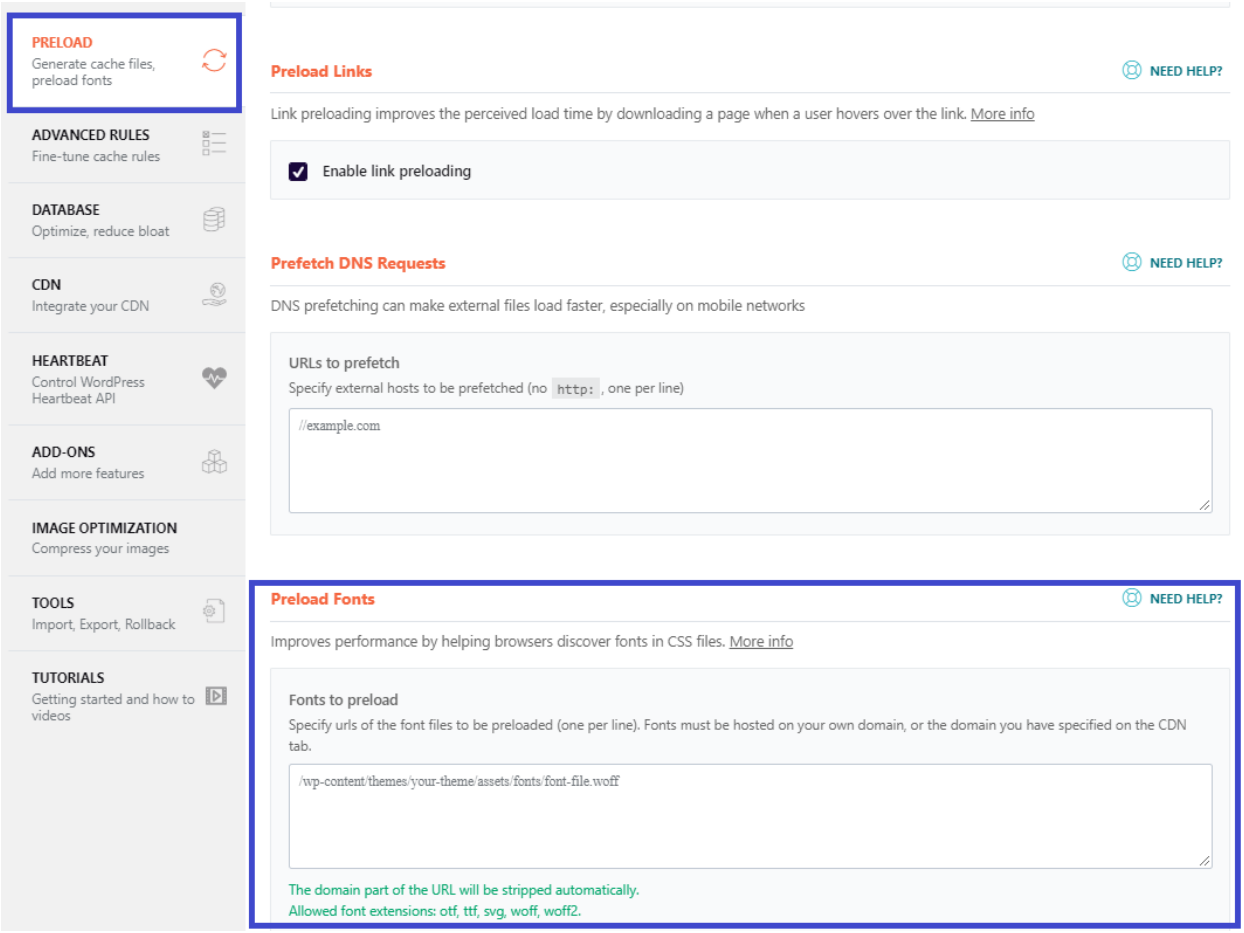
As a best practice for avoiding layout shifts, you should preload fonts.

By preloading fonts, you'll tell the browser to load the fonts as one of the top-priority resources. When rendering the page, the browser will load the fonts as fast as possible. As a result, the browser will likely include the fonts in the first meaningful paint — that's when the page's primary content is fully loaded and displayed. In that case, no layout shift will occur.

You can add the `rel=preload` to the key web fonts:

```
<link rel="preload" href="font.woff2" as="font"
type="font/woff2" crossorigin>
```

Did you know that you can easily preload fonts with WP Rocket? In the dedicated tab, you only have to include the URLs of the font files to be preloaded:



Preload tab - Preload fonts feature

By preloading fonts, you'll address the "Ensure text remains visible during webfont load" PageSpeed Insight recommendation.

There's more to this point. To prevent any FOIT and FOUT issues, you should combine the `rel=preload` (or the WP Rocket feature enabled) with the CSS line `font-display: optional`.

The CSS font-display descriptor determines how font files are downloaded and displayed by the browser.
With `font-display: optional`, the browser will download and cache the font files

to make them immediately available for rendering. So, even though `font-display` has several values, `optional` is the one you should use.

Another useful tip to reduce the FOUT issue is to add the `display:swap;` missing on font-display properties. WP Rocket can help you do it if you minify/combine CSS files.

There are other ways to load fonts faster:

- **Convert the icon fonts to SVG**. Font icons take a while to load and don't help accessibility. There's no reason to use them. Using SVGs, the font will render faster, and you will load the exact fonts you need.

- **Make multiple font formats available**. By doing so, the browsers will pick the compatible format and only load its font. Here is some information about font formats that you may find helpful:

    1) Avoid TTF. It's usually 10 - 20% more heavy than WOFF.
    2) Use SVG for Safari. It's usually a bit smaller than WOFF.
    3) Use WOFF2 for modern browsers. It's the smallest size - around 30% smaller than WOFF and SVG.
    4) Implement WOFF as a fallback when SVG or WOFF2 can't be used.

- **Host your fonts locally or use a CDN to cache them**. You'll avoid any delay and deliver fonts faster.

- **Optimize your fonts to make them as small and fast as possible**. As for Google Fonts, did you know that WP Rocket automatically takes care of them?

By applying these recommendations, you'll optimize your fonts and avoid several layout shifts. You'll address the PSI recommendation: "Ensure text remains visible during webfont load".

3. Manage Space and Size for Ad Slots

Performance impact: high

There are several best practices to avoid any layout shift for ads:

- Assign fixed dimensions to the ads so that you'll reserve enough space for the ads to be loaded.

- Reserve the biggest possible space for ads. Historical data come in handy to assess what's the best dimension for each ad slot.

- Keep every space reserved for ads that have not been displayed. In other words, you shouldn't collapse any area on the viewport. You could rather include a placeholder or a fallback element.

- Place non-sticky ads in the middle of the page — anyway, far from the top of the viewport.

The Delay JavaScript Execution feature provided by WP Rocket can help you control dynamic content above the fold like Google Ads. The feature can be used to stop dynamic interaction, content injection (ads), and dynamic class changes until there is an interaction on the page.

Once again, you'll address the "Serve images with correct dimensions" PSI recommendation. The same goes for the next section.

4. Manage Space for Embeds and Iframes

Performance impact: high

The recommendations for managing embeds and iframes are similar to the ones for ads.

In particular, you should precompute enough space for such elements. Once again, historical data can be useful to understand how much space you should reserve. Placeholder or fallback is an excellent solution to manage the unknown embed size.

5. Manage Dynamic Content

Performance impact: high

Dynamic content such as banners can also affect Cumulative Layout Shift. That's why you should avoid displaying new content unless it's triggered by user interaction. As you know, CLS counts only the layout shifts that occurred when users are not interacting with the page.

As explained in the "Manage Space and Size for Ad Slots" section, you can take advantage of the Delay JavaScript Execution option provided by WP Rocket to control dynamic content above the fold.

By managing dynamic content, you'll take care of the following PageSpeed recommendations:

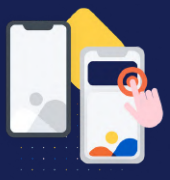
- Avoid large layout shifts
- Avoid enormous network payloads.

6. Prefer the CSS Transform Property for Animations

Performance impact: low

The last best practice to ensure visual stability is to take care of animations. You can use the CSS property: transform, which doesn't trigger any layout changes.

You'll address the "Avoid non-composited animations" PageSpeed recommendation.



THE CLS CHEAT SHEET

ACTION	IMPACT ON PERFORMANCE	PSI RECOMMENDATION ADDRESSED	CAN WP ROCKET HELP?
Include Dimension Attributes on Images and Videos	🎯🎯🎯	• Serve images with correct dimensions	✓ *For images*
Manage Space and Size for Ad Slots, Embeds and Iframes	🎯🎯🎯	• Serve images with correct dimensions	✓ *It can control dynamic content above the fold*
Manage Dynamic Content	🎯🎯🎯	• Avoid large layout shifts • Avoid enormous network payloads	✓ *It can control dynamic content above the fold*
Preload and Optimize Fonts	🎯	• Ensure text remains visible during webfont load	✓
Prefer the CSS Transform Property for Animations	🎯	• Avoid non-composited animations	✗

Start Optimizing Your Core Web Vitals on WordPress Today

You made it! Thank you so much for reading our ebook.

You now understand how Core Web Vitals can affect user experience and SEO.

It's time to put into practice everything you have learned so far and optimize your Core Web Vitals performance.

Save yourself time and let WP Rocket do the job for you!

WP Rocket is the easiest way to improve the Core Web Vitals scores.

You don't even have to touch any settings! WP Rocket will automatically apply 80% of web performance best practices. You'll see an instant improvement to the naked eye in your Core Web Vitals scores right away.

What's more, you'll stop managing multiple web performance plugins. You will only need WP Rocket to boost your performance score – no technical knowledge required, we promise!

GET WP ROCKET NOW, AND TEST THE IMPROVEMENTS RIGHT AWAY!

It’s time to optimize
your Core Web Vitals performance.

Save yourself time
and let WP Rocket do the job for you!

Get WP Rocket Now →

WP ROCKET
Superior WordPress Performance