CORE WEB VITALS

THE NEXT OFFICIAL GOOGLE RANKING FACTOR

zeno CLOUD

zeno CLOUD

Content

What are Core Web Vitals?

Core Web Vitals are a pool of three specific web vitals that Google believes are apt to evaluate the user experience of the web page.

They are the metrics that should be measured by the web site owner and should be improved if needed. Each of the three components of the Core Web Vitals represents a separate aspect of the user experience.

Google is always trying to serve the better result to the users and has continuously tested the various factors that could be used to measure the customer experience on a web page. It is an ongoing process.

For now, Google is focusing on three aspects of user experience -- loading, interactivity, and visual stability.

Core Web Vitals are:

- **Largest Contentful Paint:** LCP measures the loading performance of the webpage. An ideal LCP should be less than 2.5 seconds.
- **First Input Delay:** FID measures the interactiveness of the page. An ideal FID should be less than 100 ms.
- **Cumulative Layout Shift:** CLS measures the stability of the page. . An ideal CLS should be less than 0.1.

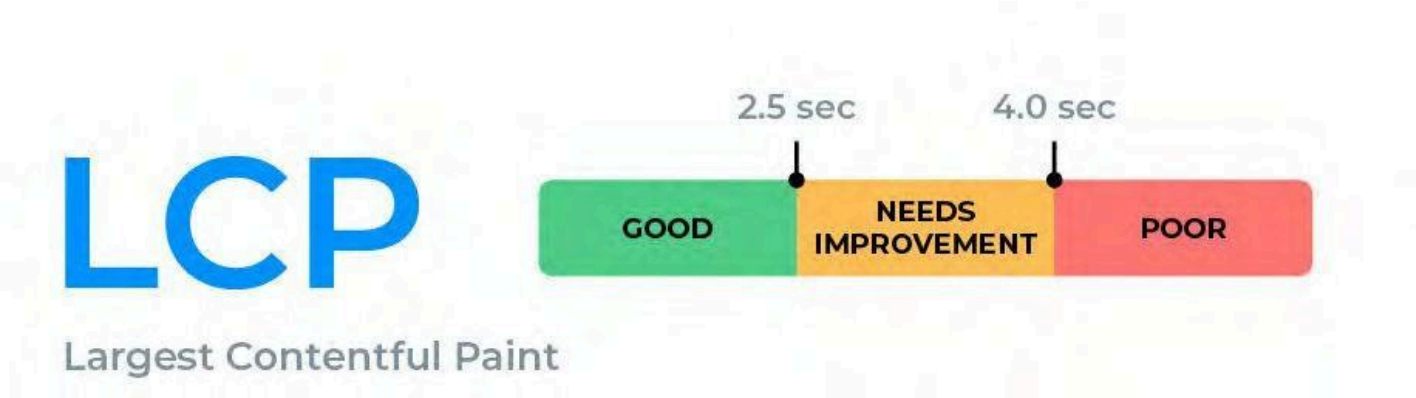
Largest Contentful Paint (LCP)

LCP measures the loading performance of the page.

We are not talking about the loading speed of the page, but the perceived page speed. It means at what moment the user feels that the page has been loaded.

The complete page can load for longer; LCP does not measure that. It measures the time it takes in the loading of the first largest impressionable element of the page.

A page loads elements by elements in stages. So, the main content at the top of your pages must load quickly. The sooner it will happen, the better be the LCP.



LCP should not be more than 2.5 seconds.

First Input Delay (FID)

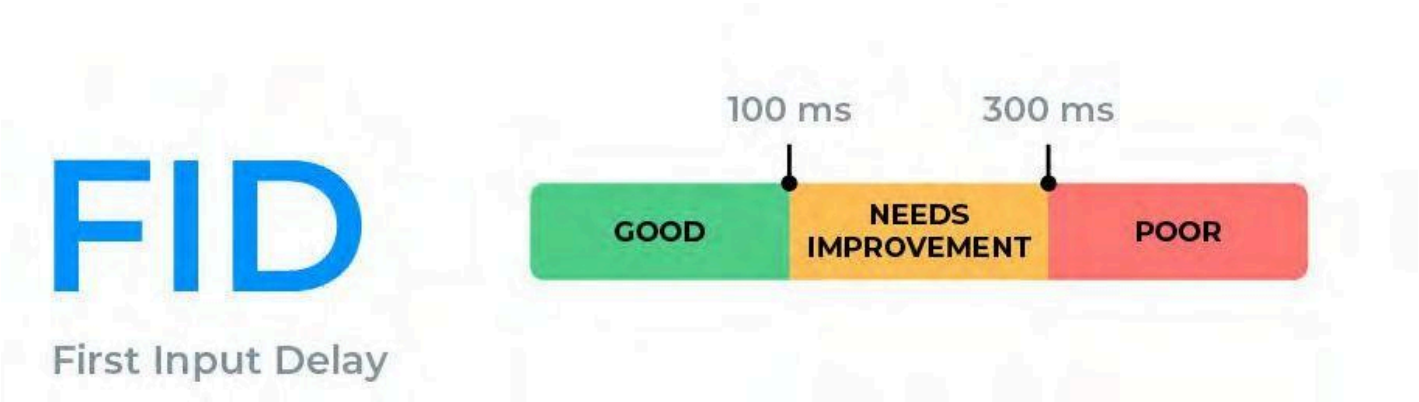
FID measures the user's first impression with the site.

Again, we are not talking about how fast the page loads, but how quickly the user can interact with it. This Core Web Vital metric captures the user's first impression with a page to the time when the browser can respond to that interaction.

The interaction could be when the user clicks a link, tap on a button, or use a custom form.

How long it takes for a link to become clickable. If the user fills a form and has to wait for the next page to load, because the browser is busy packing some other parts of the web page, then the FID is high.

This builds frustration as the user has filled the information, and he wants to move on to the next page.



First Input Delay of less than 100 milliseconds provides an excellent experience to the users.

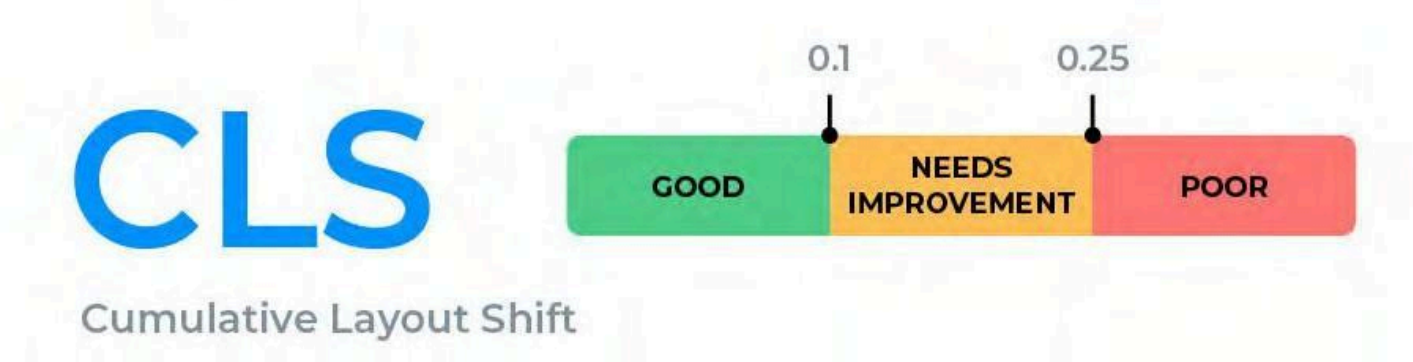
Cumulative Layout Shift

CLS measures the frequency of layout shift the user has to experience during the loading of the webpage.

Such an unexpected layout shift happens because various page resources are loaded asynchronously, or the DOM elements get added at the top of the existing content.

For example, ads, images, videos, fonts, headings -- they all change sizes dynamically.

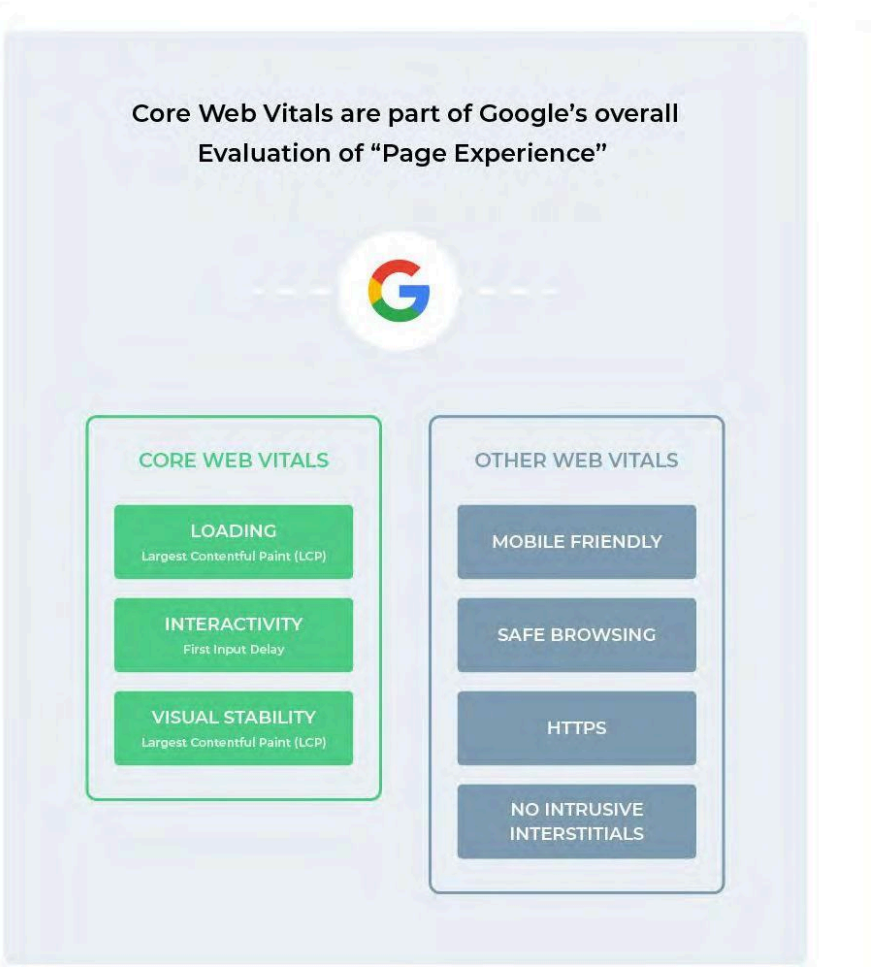
It is annoying and frustrating for the user as he might want to click something, and then the shift happens, and he accidentally clicks something else.



The CLS score of less than 0.1 provides a good user experience.

4 Other Web Vitals

Besides these three core web vitals, there are 4 more that Google uses to measure the page experience.



Mobile-Friendly: Google is screaming Mobile First for a long time. So you should check if your page/site is optimized for the mobile screen or not.

Safe-Browsing: Google regularly penalizes sites with malicious links and malware.

HTTPS: Having HTTPS is crucial for eCommerce store owners, and it is also a ranking factor.

Intrusive Interstitials: The page should not have elements that jam the main content. Many sites and blogs do have a full-page subscription box that hides the main content.

Along with these web vitals, Google has announced three core web vitals too. With time they will improve, and we may see more page experience measurement factors.

Importance of Core Web Vitals

Google has not implemented the Core Web Vitals into the search engine algorithm, but it plans it for the next year.

As per their notice:

> **A note on timing**: We recognize many site owners are rightfully placing their focus on responding to the effects of COVID-19. The ranking changes described in this post will not happen before next year, and we will provide at least six months notice before they're rolled out. We're providing the tools now to get you started (and because site owners have consistently requested to know about ranking changes as early as possible), **but there is no immediate need to take action**.

However, these new signals will work along with the other signals. This way, Google will be able to get the best data to serve users.

Still, the content quality is the top signal for the page experience. All these signals are for when all the blogs ranking for keywords have quality content. Then Google will look for the site that offers the best experience to the users.

However, if you want to be ahead of the competitors, you can optimize your pages/sites for the Core Web Vitals.

For that, you have to measure the vitals first.

Tools to Measure Core Web Vitals

Google is providing a range of tools to measure Core Web Vitals. All of Google's popular tools can help you in finding out the issues with the page.

However, there are two types of tests you can run.

Lab Tools helps in running the test in the lab conditions. It shows how potential users will see your page and enables you to optimize the page by setting up a test environment.

Field tools help you by offering you insights about how real users are experiencing your site.

As the Chrome browser collects the data of the users, and on behalf of that data, Google suggests changes to the pages.

Here are five easy to use tool for auditing the webpage:

- Lighthouse
- PageSpeed Insights
- Search Console
- Web Vitals Extension
- Web Vitals

Lighthouse

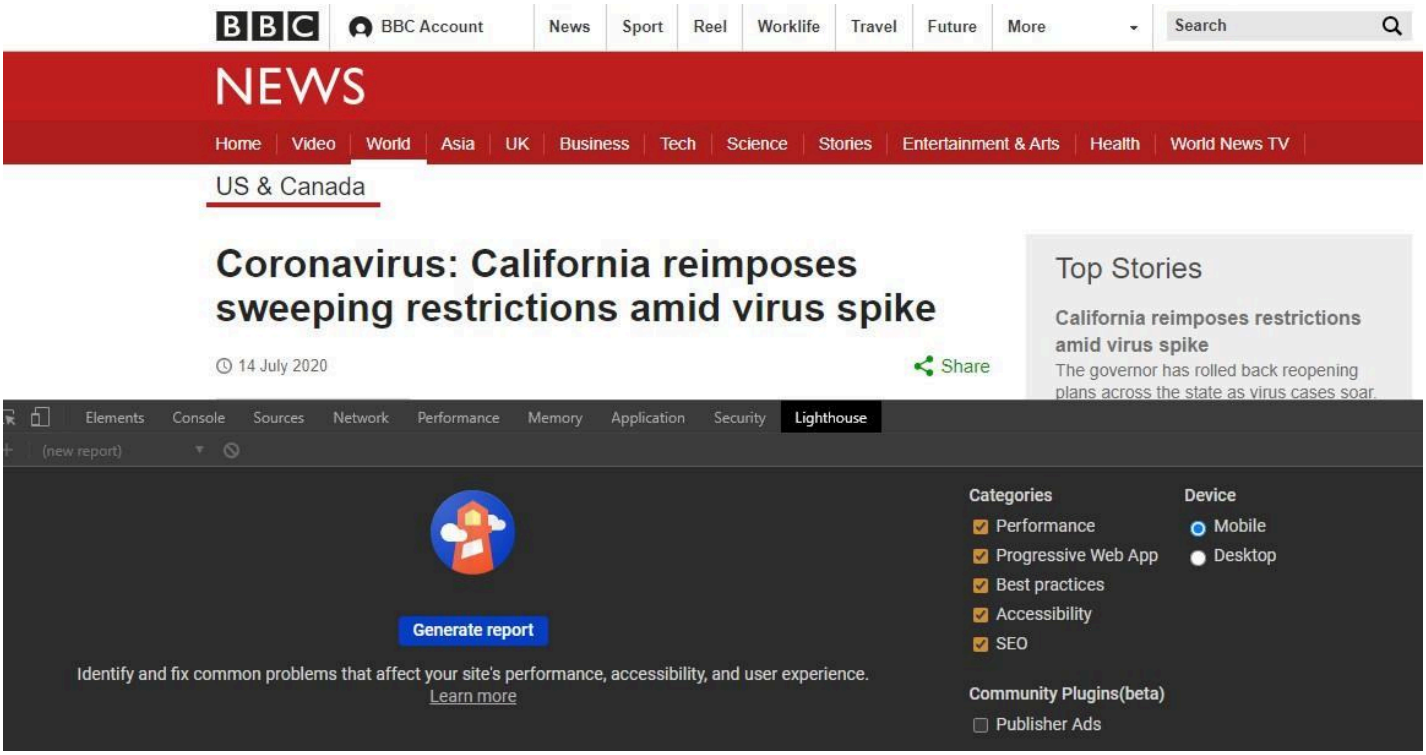
Lighthouse is a website auditing tool by Google that helps developers find the issues and opportunities to enhance the page's user experience.
It gives insight into various dimensions of user experience quality, such as performance and accessibility, but in a lab environment.

With the new upgrade, Lighthouse 6.0 brings new metrics, auditing factors, and performance scores for the web page. Among the new metrics added, two are Largest Contentful Paint (LCP) and Cumulative Layout Shift (CLS). The third metric is Total Blocking Time (TBT), similar to the First Input Delay (FID) but in lab conditions.

These three metrics are also a factor in calculating the Lighthouse performance score.

How to use Lighthouse:

Step 1: Open the Page on the Chrome
Step 2: Open Chrome Developer Tools
Step 3: Click on the "Lighthouse" Tab

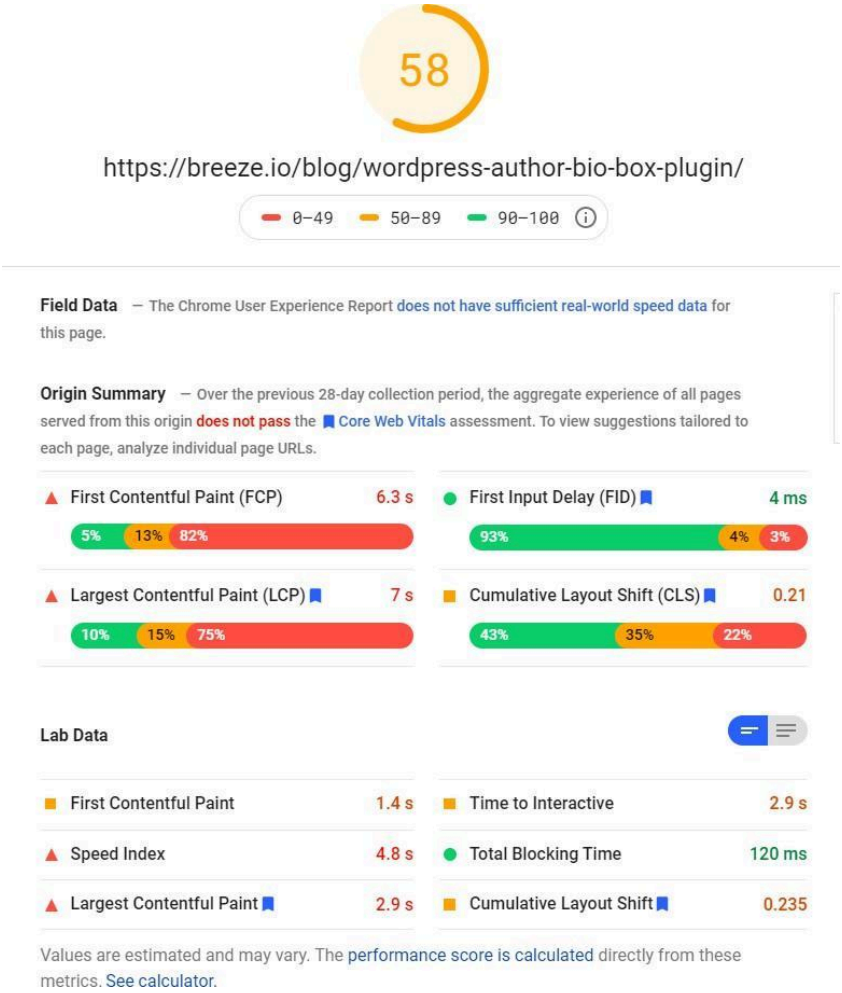


From there, you can generate a Lighthouse Report.

You can also install Lighthouse Chrome Extension, and it will speed up your task.

PageSpeed Insights

PageSpeed Insights provides both the lab and field performance reports of a web page for both mobile and desktop versions. The tool is integrated with the latest lighthouse, and also offers the Core Web Vitals report.



The tools also offer multiple recommendations to improve the page experience and rate them based on priority.

How to use PageSpeed:
• Step 1: Open PageSpeed Page
• Step 2: Run your URL

Search Console

You can find the specific pages of your site that demand your attention in terms of failing core web vitals with the help of Search Console. The data Search Console offers based on real-world users.

The pages of the site are lined up by status, metric type, and the URL group.

The report you will see is based on the Core Web Vitals metric: LCP, FID, and CLS.

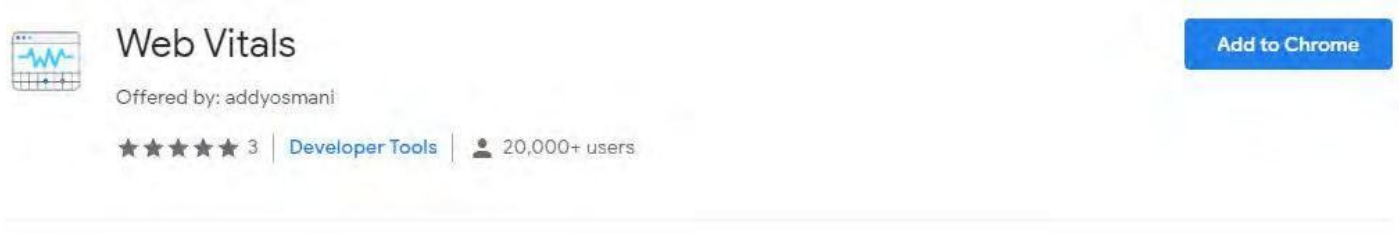
After checking out the failing Core Web Vital test, you can run those pages in the Lighthouse for the specific recommendation and optimization.

How to use the Search Console:

Step 1: Open Google Search Console
Step 2: Choose your Website
Step 3: Find the 'Core Web Vitals' option under "Enhancements" at the left panel

Web Vitals extension

This extension only measures the Core Web Vitals metrics of the live site for the desktop version.

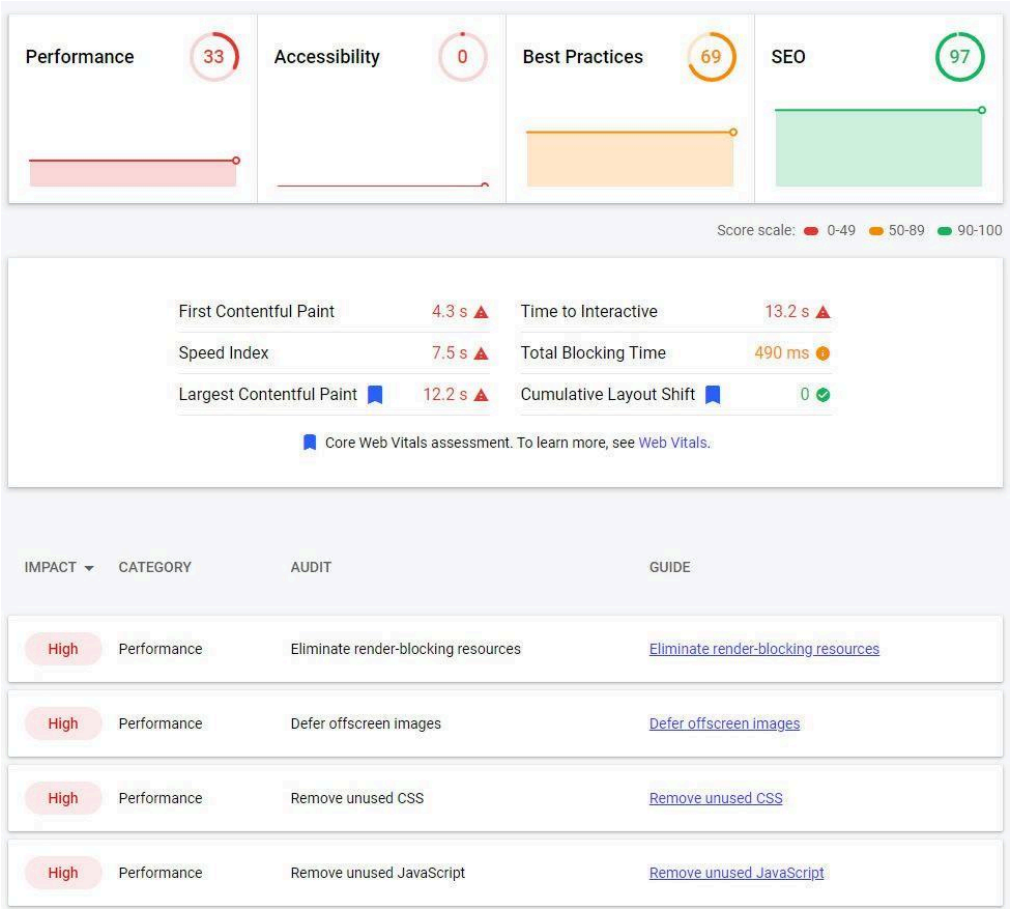


Web.dev

PageSpeed Insights power Web.dv measurements, but it does more than just suggesting the changes.

It gives you a prioritized list of improvements for a page experience and offers you a guide on how to do that.

Similar to the PageSpeed Insights, you have to run the Audit, and then it will serve you the result.



The links under the Guide option takes you to the respective page that helps you in solving that issue.

All the tools Summarize:

- Lighthouse: To check how the page is performing for Core Web Vitals and other Web Vitals
- PageSpeed Insights: To compare the Lab and Field performance of Core Web Vitals
- Search Console: To check the performance of the popular pages of your site
- Web Vitals Extension: To measure the CWV of a page in a single click
- Web Vitals: To get the prioritized list and guide to optimizing the page for vitals

Optimize the User Experience and the Core Web

Step 1: Identifying the pages with issues in Search Console

The first step is to check out the issues Google Search Console is showing.

Open the 'Core Web Vitals' report, and look for the pages that need urgent attention and improvement. Those are your key pages btw.

Step 2: Analyzing pages with PageSpeed Insights

You can open the PageSpeed Insight reports from the Search Console dashboard.

Click on the URL, and a panel will slide in from the right side.

Once you click on the 'PageSpeed Insight,' it will take you to the report in the new tab.

Read the report and look for the reasons for the issue.

Step 3: Get more specific data about pages with Lighthouse

PageSpeed and Lighthouse give the same suggestions and reports; however, Lighthouse has some additional features.

So it is wise to audit the page with Lighthouse to get additional information.

Step 4: Prioritize with web.dev

You will get lots of recommendations to improve the web page experience for the users; however, you have to prioritize the changes you are going to make.

It can be done with web.dev/measure, or you can do it manually.

Audit your site on web.dev to get a prioritized list of suggestions.

Step 5: Resolve the problems of your site

Now, resolve each problem one by one.

All the tests you have run have given you enough data and recommendations that now you know what has to be done.

Now is the time to take action.

Optimize Largest Contentful Paint

The excellent time for LCP is 2.5 seconds. After that, Google does not count it as a good user experience.

To achieve the 2.5 seconds mark, you should optimize the page for the LCP.

The most common causes of poor LCP are:

- Slow server response times
- Render-blocking JavaScript and CSS
- Slow resource load times
- Client-side rendering

Slow Server Response Times

The browser sends the request to the server, and the server responds with the requested packages. The longer it takes the browser to receive the response, the longer it will be to render it.

Faster server response time directly improves the overall speed of the page. This is why a powerful and quick web hosting is essential for the fast loading speed of the WordPress site.

At ServerGuy, we guarantee a loading speed of the site under 3 seconds.

Having a quality WordPress hosting is the first step because everything comes after this. If the server response time is large, no other improvements can improve the loading speed.

To measure the Server Response Time, you can use the Time to First Byte metric.

Optimize Your Server Response Time:

- Optimize your server
- Use CDN
- Cache assets
- Establish third-party connections early

Optimize your server

If your server is not optimized correctly, then it will delay the response.

The CMS platforms such as Magento, Joomla, WordPress, etc., require servers with the specific configuration for the ideal performance. You can install these platforms on any server, but for the optimal performance, you have to use specifically optimized servers.

ServerGuy offers WordPress Optimized Server, which means each hosting
has been tuned to work efficiently for the WordPress platform.

So this issue can be quickly solved by getting a better hosting.

Use CDN

A Content Delivery Network is a large network of servers distributed across the world. The CDN saves your content at the server near the user, and when the user opens the page again, the browser fetches the page

from the CDN server, instead of the webserver.

This way, the browser does not have to get the response from the server located far away, but it could be done from the nearest server. Hence reducing the latency time, and improving overall server response time.

There are many CDN in that world for Free. CloudFlare is one of them.

CDNs that you can use:

- CloudFlare
- Akamai
- Sucuri
- KeyCDN
- CDN77

Cache Assets

If the web page is static and there are no regular changes on the pages, storing a copy locally can increase the loading speed a lot.

The browser does not have to send requests to fetch the same resources every time, and loading them from the local storage will be faster.

There are many WordPress caching plugins that you can use at no cost.

Best WordPress Caching Plugin:

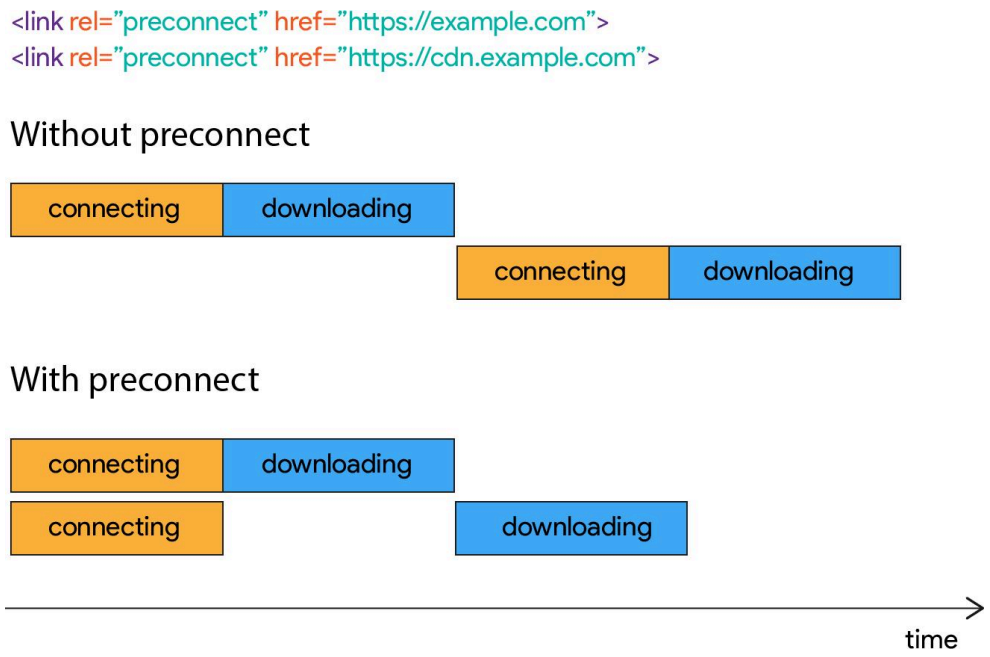
- W3 Total Cache
- WP Fastest Cache
- WP Super Cache
- WP Rocket
- Comet Cache

Establish third-party connections early

The web page does not only have the elements loading from the one domain but the multiple domains.

In that case, you should tell the browser that the page has elements that require the formation of connection with third-party apps. Modern browsers are much better in anticipating the third-party connection, but it would make a difference if you tell them early.

Merely adding the rel=preconnect to a link is enough to tell the browser that your page will establish a connection to another domain, and it would be better if the browser begins the process as soon as possible.



Render-blocking JavaScript and CSS

The browser has to parse the code before rendering the page. And the parsing will stop if it encounters the external CSS or JavaScript.

The parsing will not begin again until the JS is completely downloaded from the server; hence, it will block the other element from loading. This delays the FCP and then LCP.

The solution of render-blocking is to defer the JavaScript and CSS in WordPress.

Reduce CSS Blocking:

Minify CSS: Developers write the code in a way it is easier to read. There is so much spacing, comments, and indentation. Minifying means removing all the extra characters from the CSS so the code can compile quickly.

Defer non-critical CSS: There are many plugins and themes that, once un-installed, still leave behind the CSS. The browser does not need it, but it still gets downloaded with the other content. Deferring them increases the page loading.

This can be done with the Autoptimize plugin.
Go to the Autoptimize setting, and check the Minifying CSS and Inline CSS options.

Reduce JavaScript-blocking time

Minify JS: Similar to CSS, JS also has lots of extra space. Minifying it will reduce the JS blocking time.

Autoptimize Plugin will have the function of minifying Java.

Read the guide: How to Defer JavaScrip in WordPress site?
Also Read: How to Minify Java and CSS?

Slow Resource Load Times

CSS and JavaScript blocking time slow down the web page and directly impact the performance of the WordPress site. But there are other types of resources on the page that load slowly.

Elements such as :

- Image
- Videos
- Image behind the text
- Gifs

LCP is affected by the time taken by these elements in loading. You should ensure that these files load fast, and here are few ways to do this:

- Optimize and compress images
- Compress text files
- Adaptive Serving
- Cache assets using a service worker

Optimize and Compress Images

The majority of the WordPress sites have web pages filled with images. Large images, carousels, banner images, ads images, and much more.

Images are often made a large part of any webpage. And they are critical to building an interactive and attractive layout.

There are steps you can take to optimize and compress images.

• Don't use images (or high-quality images) if it is unnecessary and keep your content text-based.
• Compress Images with WordPress plugins (Smush or ShortPixel)
• Convert the images into Web Format (JPEG200, or WebP)
• Use responsive images
• Implement an Image CDN

Compress Text Files

Enabling Gzip compression on the WordPress site can significantly reduce the size of the file, which means faster transferring and downloading of the files from the server.

Compressing the resources will improve the loading time and LCP.

Check with your hosting providers if the Gzip compression is already enabled. Most of the hosting companies allow the compression by default or give an easy way to configure it.

As ServerGuy, our WordPress hosting is optimized for speed. So you don't have to worry about Gzip compression at all.

However, if you want to compress the text files, follow our guide on enabling Gzip Compression.

Adaptive Serving

Adaptive serving means understanding the user's device and network condition, and then adapting to it. For example, if the person is 4G, then the image loads first, and then the video. But if the person is on WiFi and the Internet speed is high, then the video loads first.

The coding for this would look something like this:

```
if (navigator.connection && navigator.connection.effectiveType) { if
  (navigator.connection.effectiveType === 'WiFi') {
    // Load video
  } else {
    // Load image
  }
}
```

Useful properties that you can use:

- navigator.connection.effectiveType: Effective connection type
- navigator.connection.saveData: Data-saver enabled/disabled
- navigator.hardwareConcurrency: CPU core count
- navigator.deviceMemory: Device Memory

Cache assets using a service worker

Service Worker is a script that the browser loads in the background.

Such service workers can be used to cache the assets so that they could be served from the browser instead of the server. Caching the critical resources using service workers can improve the loading speed of the site.

You don't have to build a custom service worker, and there are libraries like WorkBox that make the task easier.

Client-side Rendering

Many elements on the WordPress site use client-side rendering.

For example, when you sign-up to any site, it asks for the password twice. If the password is not the same in both boxes, it shows the warning sign, even before you click the submit. That is one example of client-side rendering, as the server doesn't have to do anything with that function.

Similarly, many functions and options work on the client-side rendering. Though useful, but it affects the LCP if there is JavaScript involved.

Some optimizations should be done to minimize the effects of client-side rendering on the WordPress site's LCP time.

- Minimize critical JavaScript
- Use pre-rendering

Minimize Critical JS

If the web page content depends heavily on the loading of javascript, you should cut down the size of the JS bundle as much as possible. You can do it by:

- Minifying JavaScript
- Deferring unused JavaScript
- Minimizing unused polyfills

Use pre-rendering

Pre-rendering is a more simple technique of improving the LCP of WordPress sites. In it, static HTML files are generated on a headless browser. These files can then be shipped along with the JS packages that are required for the rendering of the page.

Optimize First Input Delay

The excellent time for FID is 100 Milliseconds. More than that, the delay becomes annoying to the user as per the Google Chrome reports.

It is vital to have FID under 100 ms, and you should optimize WordPress sites for that. Heavy JavaScript is the root cause of the poor FID, and it can be reduced by optimizing the JS.

Tips to get the better FID:

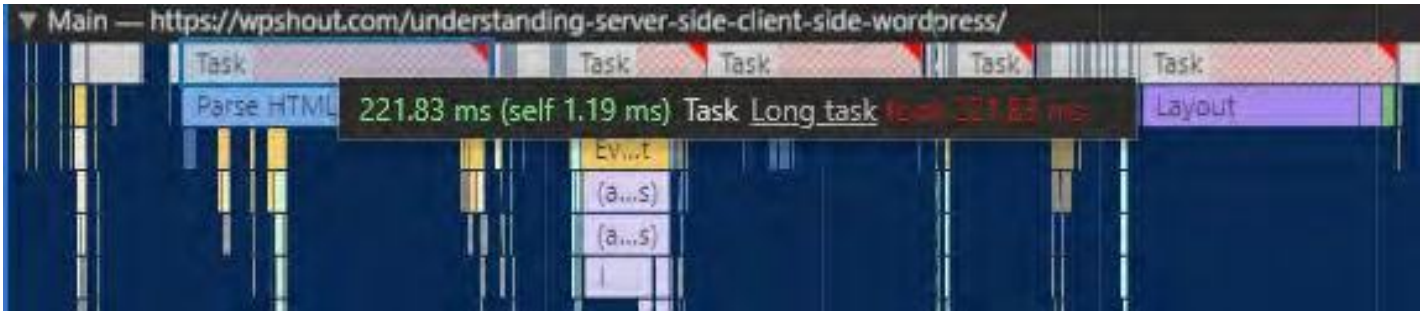
- Break up Long Tasks
- Optimize your page for interaction readiness
- Use a web worker
- Reduce JavaScript execution time

Break up Long Tasks

The piece of code that blocks the main thread for 50 ms or more can be considered a Long Task.

After reducing the JavaScript on your WordPress blog, you should look for these Long Tasks. You can break them in several tasks, and enable them to run asynchronously.

You can find the Long Tasks in the Chrome Developer tools. Go to the Performance panel.



Optimize your page for Interaction Readiness

First Party Script Execution:

Heavy JavaScript means big execution time, and inefficient chunking slows down the page interactiveness with the user, which affects the FID and TBT.
Solution: Progressive Loading of the code. Progressive Loading means to load the initial load as quickly, and then loading UI components only when required.

Third-party Script Execution:

The majority of the WordPress sites use Google Analytics, or other third-party scripts to perform various functions. These third-party scripts keep the network busy, and due to this, the main thread becomes unresponsive.
Solution: Don't load the script until they are closer to the viewport.

Use a Web Worker

The main reason for the input delay is the blocking of the main thread.
You can run scripts in the background with the Web workers. For example: Moving non-UI operations to a separate worker can reduce the workload of the main thread time, and eventually, it will improve FID.
There are many libraries from where you can find the web worker to use.

- Comlink
- Workway
- Workerize

Reduce JavaScript execution time

As I mentioned earlier, JavaScript execution time is critical to get better FID. The best way to reduce the JS execution time is to reduce the number of JS to execute.

This will speed up the overall process, and the browser can begin to respond to user interaction faster.

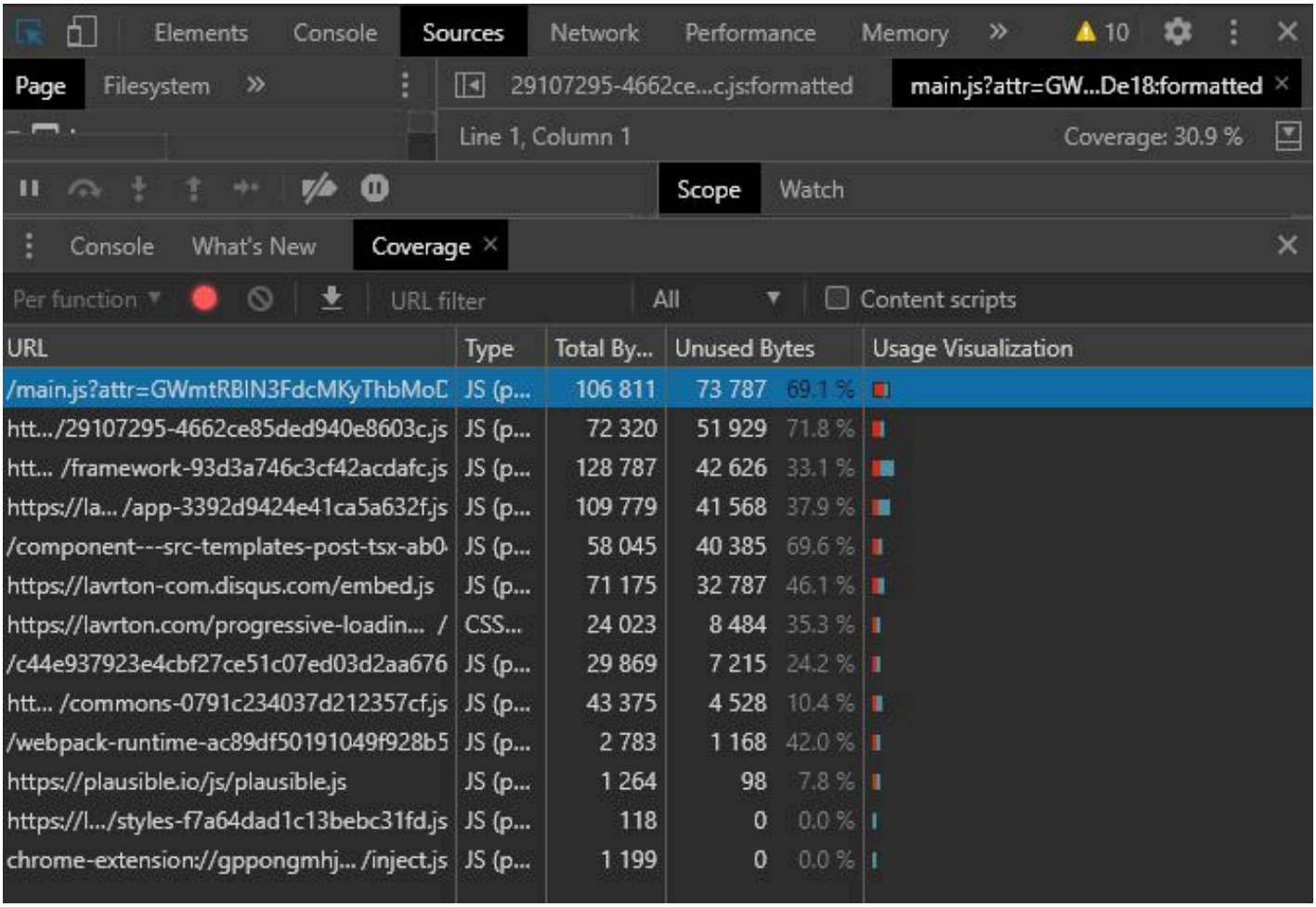
Ways to reduce the number of JS on page:

- Defer unused JavaScript
- Minimize unused polyfills

Defer unused JavaScript

When the browser encounters JavaScript, it holds all the code and downloads, parses, compiles, and executes that JavaScript at priority.
That's why you should only load the JS that is essential for the page.

You can find the unused JS on the Coverage tab of the Chrome Developers panel.



There are two ways to cut down unused JavaScript.

Method 1:
Code Splitting: Split the large code into smaller chunks and load them conditionally (lazy loading). All modern browsers support the dynamic import syntax.

Method 2:
Async or Defer: Use the Async and Defer tag for all the third-party scripts that are not necessary as the above the fold content.

```
<script defer src="..."></script>
<script async src="..."></script>
```

Minimize Unused Polyfills

With every upgrade, new HTML and JS features are introduced, and they make the development easier. But the problem is that the browsers are not that quick to implement those changes.

To solve this issue, the developer uses polyfills for the features not supported by the browsers. However, the Polyfills come with their disadvantages and increase the FID time.

The best solution is to minimize the unused Polyfills or to use polyfills only when it is ended. But it is not an easy task.

For more information about Polyfills optimization, read this article.

Cumulative Layout Shift

For good user experience, WordPress sites should have a CLS score of less than 1.

Layout shifts can be distracting and annoying to the point that users leave the page. Having a CLS score of less than one is essential, or the user experience will be terrible. You can optimize the page to get an optimal CLS score.

Common causes of a poor CLS are:

- Dimensionless Images
- Ads, embeds, and iframes without dimensions
- Dynamic Content Injection

Dimensionless Images

Having images without width and length attribute increases the CLS score. As the page loads from Top to Bottom, leave the blank space on the page if the size of the image is specified (length and width). In the beginning, developers used to add the Length and Width attributes to image tags, but those tags were pixel.

For example:

<img src="puppy.jpg" width="640" height="360" alt="Puppy with balloons">

The pixel dimension is specified, and the browser leaves that space for the image, despite the size of the image. The image has to stretch into 640x360 pixels of the area.

But then the responsive design happened, and the developers stopped writing the width and length attributes. Instead, they started using CSS to resize the images according to the device's display.

Example:

```
img {
width: 100%; /* or max-width: 100%; */
height: auto;
}
```

This approach works, but there is a downside to it. The browser can only know the dimension of the image once the image is downloaded. And as the image loads, the content of the page would re-assign itself. The sudden pop down of the screen and users losing its position on the page has become common.

To solve this, the aspect ratios were introduced. Aspect ratio means the ratio of Width to Height.

If the browser knows one dimension and aspect ratio, then it can find the other one.

If puppy.jpg has a 360px height, width is 360 x (16 / 9) = 640px
If puppy.jpg has a 640px width, height is 640 x (9 / 16) = 360px

However, the Modern Web practice included advanced use of the aspect ratio, CSS, and SRCSET.

WordPress allows you to change the size of the image at the front-end. Responsive images are a core feature of WordPress, so you don't have to worry about it much.

Ads, embeds, and iframes without Dimension

Advertisements

Ads are one of the primary reasons that contribute to the poor CLS score. Site owners often support dynamic ad sizes for the optimum performance/revenue, as the large ad sizes mean more clicks. But it sometimes hurts the user experience, as the large ads push down the content user is trying to consume.

Best practices to avoid such issue:

- Similar to images, reserve the spot for the ads.
- If the ads are not displayed at the reserved spot, don't collapse the space.
- Choose the best size of the advertisements, after checking out the historical data
- Avoid putting Ads at the top of the page, if do, ensure to reserve a spot

However, there are chances that the smaller-sized ads will fill the large space, or there will be no ad, but the risk you have to take for the better user experience.

You can use various Ad Inserter plugins on WordPress for this task. The Ad Inserter plugins will give you an option to set the spot's size for the ad.

Embeds and iframes

You can insert portable content on the WordPress page, such as YouTube videos, Google Maps, Tweets, etc.

Browsers are not able to know how large these embeds would be. For one, the embeds are dynamic and can change. WordPress can't see the size of the embeds, and it cannot reserve a spot for them.

Eventually, they will load, and it will affect the layout of the page.

The best solution to this issue is to precompute the size of the embeds and leave a placeholder on the page. However, for this, you will need a web developer.

- Inspect the embeds after it loads, and check the dimension
- Set a placeholder with the same dimension

Dynamic Content Injection

Besides ads, there is so much content that webmasters usually try to push by injecting them at the top of already loaded content.

A service banner, a form, updated privacy policy, such elements often disrupts the page experience.

More example:

- Sign-up for our newsletter!
- Related content
- Install our native app
- We're still taking orders
- GDPR notice.

The solution is the same. If you are going to load something, leave sufficient space on the page, so it does not affect the layout of the page.

Summary

Core Web Vital	Optimization	Solution
Largest Contentful Paint	Slow Server Respons Times	• Optimize your server • Use CDN • Cache Assets • Establish third-party Connections Early
	Slow Server Response Times	• Reduce JS • Reduce CSS
	Slow Resource Load Times	• Optimize and Compress Images • Compress text files • Adaptive Serving • Cache assets using a Service Worker
	Client-side Rendering	• Minimize critical JavaScript • Use pre-rendering
First Input Delay	Client-side Rendering	• Piece of code that blocks the thread for more than 50m ms. Split such tasks.
	Optimize your page for interaction readiness	• Progressive Loading of the Code • Avoid loading script till they are near the viewport

	Use a Web Worker	Run the threads in the background for various tasks, such as caching, or DNS lookup
	Reduce JavaScrip execution time	• Defer unused JavaScript • Minimize unused Polyfills
Cumulative Layout Shift	Dimensionless Images	Add width and length attributes to the image
	Advertisement	Avoid loading the Dynamic size ads at the top of the content
	Embeds and iFrames	Precompute the size of embeds and leave a reserved spot
	Dynamic Content Injection	Avoid inserting new content above the existing content

About Zeno Cloud

Zeno Coud offers AWS Consulting Service and Cloud Solutions to businesses WorldWide. Our expert team will assess your cloud needs and deliver the solutions that will help your business to achieve its goals. We are certified Amazon partners.

amazon web services | Partner Network

CONSULTING PARTNER

aws certified Cloud Practitioner

aws certified Solutions Architect Associate

Contact Sales

+91-9852704704

sales@zenocloud.io

Contact Support

+91-9852704704

support@zenocloud.io

Contact Billing

+91-9852704704

billing@zenocloud.io

zeno CLOUD

Website

www.zencloud.io

USA Office

2093 Philadelphia Pike, Claymont, DE 19703, USA.