# How to audit Core Web Vitals

Back in May 2020, Google announced that Core Web Vitals would become a part of Google's algorithms in 2021, but told site owners that "there is no immediate need to take action." By November 2020, Google revealed that this update would take effect in May 2021, so for site owners and SEOs across the world, the time to take action on the aptly named Page Experience Update is now.

## What are Core Web Vitals?

Core Web Vitals are a set of metrics used to measure a website's loading, interactivity and visual stability. All three are related to site speed in one way or another, which is something we know has been important for both search engines and users for a very long time.

What's really interesting about Core Web Vitals, and the Page Experience Update in particular, is that Google is not often very forthcoming with the specifics of its algorithm updates. But in this case, we have been given the exact metrics we need to measure and improve, and the date this update will come into effect. This indicates that Page Experience certainly will be an important update, but also one that we can actually prepare for, as long as the auditing process is detailed and accurate. Here are the metrics that need to be analyzed in a Core Web Vitals audit:

Google’s Core Web Vitals. Source: Google.

**Largest Contentful Paint (LCP)** measures loading performance (i.e., how long it takes for the largest item in the viewport to load). To provide a good user experience, LCP should occur within 2.5 seconds of when the page first starts loading or a maximum of 4 seconds to avoid a “poor” score (although between 2.5 and 4 seconds still “needs improvement”).

**First Input Delay (FID)** measures interactivity (i.e., how long it takes for the website to respond when a user clicks on something). To provide a good user experience, pages should have an FID of less than 100 milliseconds or a maximum of 300 milliseconds to avoid a “poor” score (although between 100 and 300 milliseconds still “needs improvement”). Within the audit process detailed in this article a similar metric is used,

“Total Blocking Time (TBT),” because First Input Delay requires field data, but this audit uses lab data because field data may not always be available for the website you are auditing.

**Cumulative Layout Shift (CLS)** measures visual stability (i.e., whether or not the page jumps around as the user scrolls through the content). To provide a good user experience, pages should maintain a CLS of less than 0.1 or a minimum of 0.25 to avoid a “poor” score (although between 0.1 and 0.25 still “needs improvement”).

This audit focuses on metrics with a “poor” score as these will be the main priority areas, but you can adjust it to include “needs improvement” too. So, now we know what we’re auditing, let’s get into the audit process itself.

***RELATED:*** *Guide to Core Web Vitals for SEOs and Developers*

# How to audit Core Web Vitals using Screaming Frog

Knowing what Core Web Vitals are is one thing, but finding a way to audit and communicate Core Web Vitals issues to clients in a way that is both useful and actionable is a challenge that SEOs across the globe are facing. The audit process I have put together is designed to provide real details, examples and data to work with when tackling Core Web Vitals issues.

To start the audit, you will need three things:

**Step 1: Connect the PageSpeed Insights API key to Screaming Frog**

First, you’ll need to connect your PageSpeed Insights API key to Screaming Frog. This will enable you to access PageSpeed Insights data and recommendations on a page by page basis.

You'll only get a limited number of PageSpeed Insights queries (around 25,000 per day) which should be enough for smaller sites, but for larger sites, you will be able to apply learnings from the pages you do get queries for to the rest of the site.

- With your PageSpeed Insights API key in hand, open up Screaming Frog and navigate to Configuration > API Access > PageSpeed Insights.
- Paste your API key into the "Secret Key" box.
- Click "Connect."

Once connected, click on "Metrics." Here you will define the metrics that will be displayed within your crawl. For the

purposes of this audit, I am selecting “All Metric Groups,” but you can choose just the ones you want to report on and click “OK.”

The metrics groups available are as follows:

- Overview – Provides general overview information for the page, such as the size of the page and the potential load savings that could be made on the page.
- CrUX Metrics – Data from the Chrome User Experience Report. If field data is available from real-life, opted-in users, it will appear here.
- Lighthouse Metrics – Most of the lab data we use within the audit comes from here, including LCP, TBT and CLS scores.
- Opportunities – Provides suggestions for page speed improvements specific to each page.
- Diagnostics – Provides additional information about the overall performance of the website being crawled.

**Step 2: Crawl the website**

Next, you'll need to start your crawl. Copy the domain of the website you are crawling and paste it into the box at the top of the crawler that says "Enter URL to spider." As the site is crawled, you'll notice that there is both a "Crawl" and an "API" progress bar in the top right-hand corner. You'll need to wait for both of these to reach 100% before you start analyzing your data.

**Step 3: Report the size of the problem**

Before you get into the specifics of what needs fixing, the first step is to communicate the extent of the problem. To do this, you need to look at what percentage of pages fail each Core Web Vitals minimum thresholds.

In the top navigation bar, select “PageSpeed” and then “Export.”

Looking at your exported data, find the following columns and filter accordingly:

- Largest Contentful Paint Time (ms) – Filter to find all pages with LCP of 4000ms or more.
- Total Blocking Time (ms) – Filter to find all pages with TBT of 300ms or more.

- Cumulative Layout Shift – Filter to find all pages with CLS of 0.25 or more.

Add this data to a separate datasheet so that you or your client can easily view the pages that fail each Core Web Vital. You can then report on a percentage of pages on the site that fail each Core Web Vitals minimum threshold. Here's an example I sent to a client recently:

- 95% of pages have a Largest Contentful Paint of over 4 seconds (fail) – see "LCP >4s" tab in the attached datasheet.
- 58% of pages have a Total Blocking Time of over 300 milliseconds (fail) – see "TBT >300ms" tab in the attached datasheet.
- 93% of pages have a Cumulative Layout Shift score of over 0.25 (fail) – see "CLS >0.25" tab in the attached datasheet.

Now you are armed with a full list (or a sample list, if the site was too large) of pages that are failing to meet Core Web Vitals minimum thresholds, so developers know exactly where to look for failing pages. If you notice any patterns (e.g., it's only blog pages, etc.) then you can report this now, too.

**Step 4: Report the issues specific to each page and make appropriate recommendations**

This is the part of the audit where we turn problems into solutions. We know that X amount of pages are failing Core Web Vitals minimum thresholds, but what can we/the client do about it? This is where the PageSpeed Insights API really works its magic.

On the right-hand side, in the "Overview" tab, scroll down to "PageSpeed." Here you will find the list of issues/recommendations relating to page speed and for the most part, Core Web Vitals.

There are a wide variety of different issues reported here. If there are any you are unfamiliar with, search for it on the web.dev website to get more information. While the data available within Screaming Frog and PageSpeed Insights may not provide an entirely exhaustive list of every issue that can impact Core Web Vitals, it certainly helps when analyzing your/your client's site as a whole.

Click on an issue to see the pages affected, and export them to save into your datasheet. You are now reporting on the specifics of how many pages are impacted by a particular issue, and the URLs of affected pages. In the example below, I have exported a list of all the pages that have render-blocking resources that might be negatively impacting LCP. I can now recommend that the client looks at this list and decides whether inlining, deferring or removing the resources on these pages would be possible.

For each of the recommendations you are making, you will also be able to see the “Savings” that could be made by fixing that particular issue, either in bytes or milliseconds. Using your exported data for each issue, you can now add up the potential savings for each issue, and the average savings that could be made per page by resolving that issue, so you can make your recommendations for which issues to tackle first based on the amount of potential load savings that can be made. In the example below, the savings in both milliseconds and bytes are far greater for Deferring Offscreen Images than Removing Unused CSS, so Deferring Offscreen Images will be a higher priority.

**Step 5: Report examples of the issues specific to each page**

By reporting on examples of the issues specific to each page, we provide a more granular dataset that allows the client/developers to quickly understand what the issue is and whether it is something that can be resolved or not.

Following on from the render-blocking resources example, you now need to select one of the URLs affected by this issue, and select the “PageSpeed Details” tab in the bottom navigation bar. The bottom left panel will now show page speed information relevant to the selected page. Navigate to Opportunities > Eliminate Render Blocking Resources.

In the bottom-right panel, you will now see the URLs of render-blocking resources on that page, their size (in bytes)

and the potential page load savings that could be made (in milliseconds) if these render-blocking resources are eliminated.

Unfortunately, you can’t export these specific issues in bulk (as far as I am aware), but you can copy and paste a few examples into your datasheet and again look for any patterns. Often, the same resources will appear on multiple pages/every page on the site, so learnings can be applied sitewide.

Once you have pulled this data together for every issue on the site, you can provide a written report with recommendations for each issue in priority order and refer to the data in your datasheet.

**Step 6: Once changes have been made, crawl the site again and compare**

The sooner you complete this audit the better, as some of the issues will take time to resolve. Once the issues have been tackled, you can go back to step one and recrawl the site to see how things have changed. This is where your percentages of pages not meeting Core Web Vitals minimum thresholds will come in handy, as it shows a quick and easy way to understand whether your changes have had the desired impact or not.

When reporting on Core Web Vitals and the Page Experience Update to clients, the questions I am often asked are around how this update is going to impact rankings. Despite this clearly being an important update, I don't envisage websites that haven't met the minimum thresholds seeing a huge drop in rankings overnight. It will more likely be a case of sites that have excellent content that is able to meet or exceed Core Web Vitals minimum thresholds seeing a slight improvement in rankings, which will, of course, mean slight drops in rankings for the competitors that they overtake. This sentiment is supported by Google's own guidelines on the subject:

> *"While all of the components of page experience are important, we will prioritize pages with the best information overall, even if some aspects of page experience are subpar. A good page experience doesn't override having great, relevant content. However, in cases where there are multiple pages that have similar content, page experience becomes much more important for visibility in Search."*

Site owners that are able to meet the minimum thresholds are putting themselves at a distinct advantage in terms of search visibility, and while we can't predict exactly what will happen on the day the Page Experience Update goes live, this audit process will help you to get well prepared.

*Opinions expressed in this article are those of the guest author and not necessarily Search Engine Land. Staff authors are listed here.*

## About The Author

Tom is the Head of SEO at UK based digital marketing agency, Adido. With over six years of SEO experience, he is responsible for the planning and implementation of SEO strategies for a number of clients in a wide variety of industries.
http://feeds.searchengineland.com/~r/searchengineland/~3/xZQ1H5hy0no/how-to-audit-core-web-vitals-346076