http://localhost/dev/wordpressbase/

55

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 50–89 90–100

METRICS Expand view

▲ First Contentful Paint
11.7 s

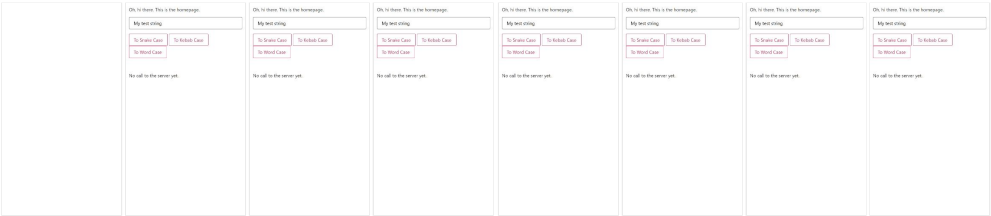
▲ Largest Contentful Paint
11.7 s

Total Blocking Time
0 ms

Cumulative Layout Shift
0

▲ Speed Index
11.7 s

View Treemap



Show audits relevant to: All FCP LCP TBT

DIAGNOSTICS

▲ Enable text compression — Potential savings of 1,662 KiB

Text-based resources should be served with compression (gzip, deflate or brotli) to minimize total network bytes. Learn more about text compression. FCP LCP

You can enable text compression in your web server configuration.

URL	Transfer Size	Potential Savings
localhost 1st Party	**2,028.7 KiB**	**1,661.8 KiB**
…css/frontend.css (localhost)	492.0 KiB	445.5 KiB
…jquery/jquery.js (localhost)	278.6 KiB	196.6 KiB
…css/frontend.css (localhost)	191.9 KiB	170.2 KiB
…js/svete-test.js (localhost)	189.1 KiB	150.9 KiB

URL	Transfer Size	Potential Savings
…js/frontend-modules.js (localhost)	180.9 KiB	149.2 KiB
…vendor/wp-polyfill.js (localhost)	125.8 KiB	99.1 KiB
…js/elements-handlers.js (localhost)	100.4 KiB	85.4 KiB
…js/frontend.js (localhost)	100.8 KiB	78.9 KiB
…js/frontend.js (localhost)	56.0 KiB	45.5 KiB
…css/global.css (localhost)	40.8 KiB	37.8 KiB
…dist/i18n.js (localhost)	48.7 KiB	37.2 KiB
…ui/core.js (localhost)	48.7 KiB	35.6 KiB
…vendor/wp-polyfill-inert.js (localhost)	29.5 KiB	22.5 KiB
…vendor/regenerator-runtime.js (localhost)	24.6 KiB	17.6 KiB
…dist/hooks.js (localhost)	19.3 KiB	15.0 KiB
…css/swiper.css (localhost)	18.8 KiB	14.0 KiB
…waypoints/waypoints.js (localhost)	17.5 KiB	13.9 KiB
…js/webpack-pro.runtime.js (localhost)	15.5 KiB	11.4 KiB
…js/webpack.runtime.js (localhost)	15.1 KiB	11.2 KiB
/dev/wordpressbase/ (localhost)	12.1 KiB	8.7 KiB
…hello-elementor/style.css (localhost)	11.6 KiB	8.0 KiB
…hello-elementor/theme.css (localhost)	6.4 KiB	4.8 KiB
…js/wpa.js (localhost)	4.4 KiB	3.0 KiB

▲ Largest Contentful Paint element — 11,710 ms

This is the largest contentful element painted within the viewport. Learn more about the Largest Contentful Paint element LCP

Element
p

Phase	% of LCP	Timing
TTFB	5%	550 ms
Load Delay	0%	0 ms

Phase	% of LCP	Timing
Load Time	0%	0 ms
Render Delay	95%	11,160 ms

▲ Eliminate render-blocking resources — Potential savings of 4,950 ms

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn how to eliminate render-blocking resources. FCP LCP

There are a number of WordPress plugins that can help you inline critical assets or defer less important resources. Beware that optimizations provided by these plugins may break features of your theme or plugins, so you will likely need to make code changes.

☑ Show 3rd-party resources (1)

URL	Transfer Size	Potential Savings
localhost 1st Party	**1,045.4 KiB**	**17,260 ms**
…css/wpa.css (localhost)	1.0 KiB	300 ms
…hello-elementor/style.css (localhost)	11.9 KiB	450 ms
…hello-elementor/theme.css (localhost)	6.7 KiB	150 ms
…css/frontend.css (localhost)	192.3 KiB	3,150 ms
…css/swiper.css (localhost)	19.2 KiB	450 ms
…css/post-7.css (localhost)	1.4 KiB	150 ms
…css/frontend.css (localhost)	492.3 KiB	7,500 ms
…css/global.css (localhost)	41.1 KiB	750 ms
…css/post-8.css (localhost)	0.6 KiB	150 ms
…jquery/jquery.js (localhost)	279.0 KiB	4,200 ms
Google Fonts Cdn	**1.9 KiB**	**800 ms**
/css?family=… (fonts.googleapis.com)	1.9 KiB	800 ms

▲ Reduce unused CSS — Potential savings of 736 KiB

Reduce unused rules from stylesheets and defer CSS not used for above-the-fold content to decrease bytes consumed by network activity. Learn how to reduce unused CSS. FCP LCP

Consider reducing, or switching, the number of WordPress plugins loading unused CSS in your page. To identify plugins that are adding extraneous CSS, try running code coverage in Chrome DevTools. You can identify the theme/plugin responsible from the URL of the stylesheet. Look out for plugins that have many stylesheets in the list which have a lot of red in code coverage. A plugin should only enqueue a stylesheet if it is actually used on the page.

URL	Transfer Size	Potential Savings
localhost 1st Party	**743.6 KiB**	**736.0 KiB**

URL	Transfer Size	Potential Savings
…css/frontend.css (localhost)	492.0 KiB	492.0 KiB
…css/frontend.css (localhost)	191.9 KiB	185.0 KiB
…css/global.css (localhost)	40.8 KiB	40.8 KiB
…css/swiper.css (localhost)	18.8 KiB	18.2 KiB

▲ Reduce unused JavaScript — Potential savings of 652 KiB

Reduce unused JavaScript and defer loading scripts until they are required to decrease bytes consumed by network activity. Learn how to reduce unused JavaScript. LCP

Consider reducing, or switching, the number of WordPress plugins loading unused JavaScript in your page. To identify plugins that are adding extraneous JS, try running code coverage in Chrome DevTools. You can identify the theme/plugin responsible from the URL of the script. Look out for plugins that have many scripts in the list which have a lot of red in code coverage. A plugin should only enqueue a script if it is actually used on the page.

URL	Transfer Size	Potential Savings
localhost 1st Party	**1,129.0 KiB**	**551.8 KiB**
…jquery/jquery.js (localhost)	278.6 KiB	152.4 KiB
…js/svete-test.js (localhost)	189.1 KiB	102.5 KiB
…./node_modules/svelte/src/runtime/internal/dom.js	29.8 KiB	21.3 KiB
…./node_modules/svelte/src/runtime/internal/Component.js	14.7 KiB	9.0 KiB
…./node_modules/svelte/src/runtime/internal/transitions.js	10.6 KiB	8.7 KiB
…./node_modules/svelte/src/runtime/internal/dev.js	9.0 KiB	5.8 KiB
…./node_modules/svelte/src/runtime/internal/utils.js	6.8 KiB	5.1 KiB
…js/frontend-modules.js (localhost)	180.9 KiB	76.0 KiB
…js/elements-handlers.js (localhost)	100.4 KiB	48.2 KiB
…vendor/wp-polyfill.js (localhost)	125.8 KiB	40.8 KiB
…js/frontend.js (localhost)	100.8 KiB	40.8 KiB
…ui/core.js (localhost)	48.7 KiB	39.0 KiB
…js/frontend.js (localhost)	56.0 KiB	30.5 KiB
…dist/i18n.js (localhost)	48.7 KiB	21.6 KiB
Unattributable	**173.8 KiB**	**100.3 KiB**
chrome-extension://bgnkhhnnamicmpeenaelnjfhikgbkllg/pages/content-script-start.js	173.8 KiB	100.3 KiB

▲ Minify JavaScript — Potential savings of 580 KiB

Minifying JavaScript files can reduce payload sizes and script parse time. Learn how to minify JavaScript. FCP LCP

A number of WordPress plugins can speed up your site by concatenating, minifying, and compressing your scripts. You may also want to use a build process to do this minification up front if possible.

URL	Transfer Size	Potential Savings
localhost 1st Party	**1,250.6 KiB**	**466.4 KiB**
…jquery/jquery.js (localhost)	278.6 KiB	142.4 KiB
…js/svete-test.js (localhost)	189.1 KiB	73.6 KiB
…vendor/wp-polyfill.js (localhost)	125.8 KiB	39.8 KiB
…js/frontend-modules.js (localhost)	180.9 KiB	36.9 KiB
…dist/i18n.js (localhost)	48.7 KiB	34.0 KiB
…js/frontend.js (localhost)	100.8 KiB	29.2 KiB
…vendor/wp-polyfill-inert.js (localhost)	29.5 KiB	18.3 KiB
…ui/core.js (localhost)	48.7 KiB	17.7 KiB
…js/elements-handlers.js (localhost)	100.4 KiB	17.2 KiB
…vendor/regenerator-runtime.js (localhost)	24.6 KiB	13.7 KiB
…js/frontend.js (localhost)	56.0 KiB	12.1 KiB
…dist/hooks.js (localhost)	19.3 KiB	11.7 KiB
…js/webpack.runtime.js (localhost)	15.1 KiB	7.4 KiB
…js/webpack-pro.runtime.js (localhost)	15.5 KiB	6.8 KiB
…waypoints/waypoints.js (localhost)	17.5 KiB	5.5 KiB
Unattributable	**231.3 KiB**	**114.0 KiB**
`chrome-extension://bgnkhhnnamicmpeenaelnjfhikgbkllg/pages/content-script-start.js`	173.8 KiB	88.4 KiB
`chrome-extension://bgnkhhnnamicmpeenaelnjfhikgbkllg/pages/subscribe.js`	37.5 KiB	17.2 KiB
`chrome-extension://bgnkhhnnamicmpeenaelnjfhikgbkllg/pages/content-script-end.js`	20.0 KiB	8.4 KiB

▲ Minify CSS — Potential savings of 79 KiB

Minifying CSS files can reduce network payload sizes. Learn how to minify CSS. FCP LCP

A number of WordPress plugins can speed up your site by concatenating, minifying, and compressing your styles. You may also want to use a build process to do this minification up-front if possible.

URL	Transfer Size	Potential Savings
localhost 1st Party	**715.7 KiB**	**78.6 KiB**
…css/frontend.css (localhost)	492.3 KiB	47.2 KiB
…css/frontend.css (localhost)	192.3 KiB	22.5 KiB
…hello-elementor/style.css (localhost)	11.9 KiB	5.9 KiB

URL	Transfer Size	Potential Savings
…css/swiper.css (localhost)	19.2 KiB	3.0 KiB

Serve static assets with an efficient cache policy — 25 resources found

A long cache lifetime can speed up repeat visits to your page. Learn more about efficient cache policies.

Read about Browser Caching in WordPress.

URL	Cache TTL	Transfer Size
localhost 1st Party		**2,026 KiB**
…css/frontend.css (localhost)	None	492 KiB
…jquery/jquery.js (localhost)	None	279 KiB
…css/frontend.css (localhost)	None	192 KiB
…js/svete-test.js (localhost)	None	189 KiB
…js/frontend-modules.js (localhost)	None	181 KiB
…vendor/wp-polyfill.js (localhost)	None	126 KiB
…js/frontend.js (localhost)	None	101 KiB
…js/elements-handlers.js (localhost)	None	101 KiB
…js/frontend.js (localhost)	None	56 KiB
…dist/i18n.js (localhost)	None	49 KiB
…ui/core.js (localhost)	None	49 KiB
…css/global.css (localhost)	None	41 KiB
…vendor/wp-polyfill-inert.js (localhost)	None	30 KiB
…vendor/regenerator-runtime.js (localhost)	None	25 KiB
…dist/hooks.js (localhost)	None	20 KiB
…css/swiper.css (localhost)	None	19 KiB
…waypoints/waypoints.js (localhost)	None	18 KiB
…js/webpack-pro.runtime.js (localhost)	None	16 KiB
…js/webpack.runtime.js (localhost)	None	15 KiB
…hello-elementor/style.css (localhost)	None	12 KiB
…hello-elementor/theme.css (localhost)	None	7 KiB
…js/wpa.js (localhost)	None	5 KiB

URL	Cache TTL	Transfer Size
…css/post-7.css (localhost)	None	1 KiB
…css/wpa.css (localhost)	None	1 KiB
…css/post-8.css (localhost)	None	1 KiB

Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB

Polyfills and transforms enable legacy browsers to use new JavaScript features. However, many aren't necessary for modern browsers. For your bundled JavaScript, adopt a modern script deployment strategy using module/nomodule feature detection to reduce the amount of code shipped to modern browsers, while retaining support for legacy browsers. Learn how to use modern JavaScript TBT

URL		Potential Savings
localhost 1st Party		**0.2 KiB**
…vendor/wp-polyfill-inert.js (localhost)		0.2 KiB
wp-polyfill-inert.js:9	`@babel/plugin-transform-classes`	
…vendor/regenerator-runtime.js (localhost)		0.1 KiB
regenerator-runtime.js:151	`@babel/plugin-transform-regenerator`	

Initial server response time was short — Root document took 550 ms

Keep the server response time for the main document short because all other requests depend on it. Learn more about the Time to First Byte metric. FCP LCP

Themes, plugins, and server specifications all contribute to server response time. Consider finding a more optimized theme, carefully selecting an optimization plugin, and/or upgrading your server.

URL	Time Spent
localhost 1st Party	**550 ms**
/dev/wordpressbase/ (localhost)	550 ms

Avoids enormous network payloads — Total size was 2,072 KiB

Large network payloads cost users real money and are highly correlated with long load times. Learn how to reduce payload sizes. LCP

Consider showing excerpts in your post lists (e.g. via the more tag), reducing the number of posts shown on a given page, breaking your long posts into multiple pages, or using a plugin to lazy-load comments.

URL	Transfer Size
localhost 1st Party	**1,767.5 KiB**
…css/frontend.css (localhost)	492.3 KiB
…jquery/jquery.js (localhost)	279.0 KiB
…css/frontend.css (localhost)	192.3 KiB

URL	Transfer Size
…js/svete-test.js (localhost)	189.4 KiB
…js/frontend-modules.js (localhost)	181.2 KiB
…vendor/wp-polyfill.js (localhost)	126.1 KiB
…js/frontend.js (localhost)	101.1 KiB
…js/elements-handlers.js (localhost)	100.7 KiB
…js/frontend.js (localhost)	56.3 KiB
…dist/i18n.js (localhost)	49.1 KiB

Avoids an excessive DOM size — 43 elements

A large DOM will increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn how to avoid an excessive DOM size. TBT

Statistic	Element
Total DOM Elements	
Maximum DOM Depth	input.svelte-1gvrw7w
Maximum Child Elements	body.home.page-template-default.page.page-id-8.elementor-default.elementor-kit-7.elementor-page.elementor-page-8.e--ua-blink.e--ua-chror

Avoid chaining critical requests — 26 chains found

The Critical Request Chains below show you what resources are loaded with a high priority. Consider reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources to improve page load. Learn how to avoid chaining critical requests. FCP LCP

Maximum critical path latency: **617.501 ms**

Initial Navigation

/dev/wordpressbase/ (localhost)

- …css/wpa.css (localhost) **- 8.891 ms, 1.02 KiB**
- …hello-elementor/style.css (localhost) **- 11.71 ms, 11.92 KiB**
- …hello-elementor/theme.css (localhost) **- 12.359 ms, 6.66 KiB**
- …css/frontend.css (localhost) **- 13.534 ms, 192.25 KiB**
- …css/swiper.css (localhost) **- 12.574 ms, 19.15 KiB**
- …css/post-7.css (localhost) **- 12.023 ms, 1.44 KiB**
- …css/frontend.css (localhost) **- 14.185 ms, 492.33 KiB**
- …css/global.css (localhost) **- 16.448 ms, 41.07 KiB**

…css/post-8.css (localhost) **- 16.959 ms, 0.55 KiB**

/css?family=… (fonts.googleapis.com) **- 57.754 ms, 1.90 KiB**

…jquery/jquery.js (localhost) **- 18.477 ms, 278.96 KiB**

…js/wpa.js (localhost) **- 17.077 ms, 4.75 KiB**

…js/svete-test.js (localhost) **- 17.805 ms, 189.40 KiB**

…js/webpack-pro.runtime.js (localhost) **- 17.021 ms, 15.83 KiB**

…js/webpack.runtime.js (localhost) **- 20.628 ms, 15.46 KiB**

…js/frontend-modules.js (localhost) **- 22.405 ms, 181.21 KiB**

…vendor/wp-polyfill-inert.js (localhost) **- 21.698 ms, 29.79 KiB**

…vendor/regenerator-runtime.js (localhost) **- 21.63 ms, 24.93 KiB**

…vendor/wp-polyfill.js (localhost) **- 21.871 ms, 126.15 KiB**

…dist/hooks.js (localhost) **- 21.492 ms, 19.58 KiB**

…dist/i18n.js (localhost) **- 24.231 ms, 49.05 KiB**

…js/frontend.js (localhost) **- 24.452 ms, 56.29 KiB**

…waypoints/waypoints.js (localhost) **- 2.948 ms, 17.86 KiB**

…ui/core.js (localhost) **- 2.617 ms, 49.00 KiB**

…js/frontend.js (localhost) **- 2.808 ms, 101.13 KiB**

…js/elements-handlers.js (localhost) **- 3.585 ms, 100.68 KiB**

JavaScript execution time — 0.2 s

Consider reducing the time spent parsing, compiling, and executing JS. You may find delivering smaller JS payloads helps with this. Learn how to reduce Javascript execution time. TBT

URL	Total CPU Time	Script Evaluation	Script Parse
localhost 1st Party	**259 ms**	**100 ms**	**57 ms**
/dev/wordpressbase/ (localhost)	180 ms	37 ms	48 ms
…jquery/jquery.js (localhost)	79 ms	64 ms	9 ms
Unattributable	**83 ms**	**3 ms**	**0 ms**
`Unattributable`	83 ms	3 ms	0 ms

Minimizes main-thread work — 0.5 s

Consider reducing the time spent parsing, compiling and executing JS. You may find delivering smaller JS payloads helps with this. Learn how to minimize main-thread work TBT

Category	Time Spent
Script Evaluation	197 ms
Other	118 ms
Script Parsing & Compilation	103 ms
Style & Layout	63 ms
Parse HTML & CSS	59 ms
Rendering	2 ms

Minimize third-party usage — Third-party code blocked the main thread for 0 ms

Third-party code can significantly impact load performance. Limit the number of redundant third-party providers and try to load third-party code after your page has primarily finished loading. Learn how to minimize third-party impact. TBT

Third-Party	Transfer Size	Main-Thread Blocking Time
Google Fonts Cdn	**2 KiB**	**0 ms**
/css?family=… (fonts.googleapis.com)	2 KiB	0 ms

Avoid long main-thread tasks — 2 long tasks found

Lists the longest tasks on the main thread, useful for identifying worst contributors to input delay. Learn how to avoid long main-thread tasks TBT

URL	Start Time	Duration
localhost 1st Party		**161 ms**
/dev/wordpressbase/ (localhost)	601 ms	102 ms
…vendor/wp-polyfill-inert.js (localhost)	11,705 ms	59 ms

More information about the performance of your application. These numbers don't directly affect the Performance score.

PASSED AUDITS (22) Hide

Properly size images

Serve images that are appropriately-sized to save cellular data and improve load time. Learn how to size images.

Upload images directly through the media library to ensure that the required image sizes are available, and then insert them from the media library or use the image widget to ensure the optimal image sizes are used (including those for the responsive breakpoints). Avoid using `Full Size` images unless the dimensions are adequate for their usage. Learn More.

Defer offscreen images

Consider lazy-loading offscreen and hidden images after all critical resources have finished loading to lower time to interactive. Learn how to defer offscreen images.

Install a lazy-load WordPress plugin that provides the ability to defer any offscreen images, or switch to a theme that provides that functionality. Also consider using the AMP plugin.

Efficiently encode images

Optimized images load faster and consume less cellular data. Learn how to efficiently encode images.

Consider using an image optimization WordPress plugin that compresses your images while retaining quality.

Serve images in next-gen formats

Image formats like WebP and AVIF often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more about modern image formats.

Consider using the Performance Lab plugin to automatically convert your uploaded JPEG images into WebP, wherever supported.

Preconnect to required origins

Warnings: A `<link rel=preconnect>` was found for "https://fonts.gstatic.com" but was not used by the browser. Only use `preconnect` for important origins that the page will certainly request.

Consider adding `preconnect` or `dns-prefetch` resource hints to establish early connections to important third-party origins. Learn how to preconnect to required origins. FCP LCP

Avoid multiple page redirects

Redirects introduce additional delays before the page can be loaded. Learn how to avoid page redirects. FCP LCP

Preload key requests

Consider using `<link rel=preload>` to prioritize fetching resources that are currently requested later in page load. Learn how to preload key requests. FCP LCP

Use HTTP/2

HTTP/2 offers many benefits over HTTP/1.1, including binary headers and multiplexing. Learn more about HTTP/2.

Use video formats for animated content

Large GIFs are inefficient for delivering animated content. Consider using MPEG4/WebM videos for animations and PNG/WebP for static images instead of GIF to save network bytes. Learn more about efficient video formats LCP

Consider uploading your GIF to a service which will make it available to embed as an HTML5 video.

Remove duplicate modules in JavaScript bundles

Remove large, duplicate JavaScript modules from bundles to reduce unnecessary bytes consumed by network activity. TBT

Preload Largest Contentful Paint image

If the LCP element is dynamically added to the page, you should preload the image in order to improve LCP. Learn more about preloading LCP elements. LCP

User Timing marks and measures

Consider instrumenting your app with the User Timing API to measure your app's real-world performance during key user experiences. Learn more about User Timing marks.

All text remains visible during webfont loads

Leverage the `font-display` CSS feature to ensure text is user-visible while webfonts are loading. Learn more about `font-display`. FCP LCP

Lazy load third-party resources with facades

Some third-party embeds can be lazy loaded. Consider replacing them with a facade until they are required. Learn how to defer third-parties with a facade. TBT

Largest Contentful Paint image was not lazily loaded

Above-the-fold images that are lazily loaded render later in the page lifecycle, which can delay the largest contentful paint. Learn more about optimal lazy loading. LCP

Avoid large layout shifts

These are the largest layout shifts observed on the page. Each table item represents a single layout shift, and shows the element that shifted the most. Below each item are possible root causes that led to the layout shift. Some of these layout shifts may not be included in the CLS metric value due to windowing. Learn how to improve CLS CLS

Uses passive listeners to improve scrolling performance

Consider marking your touch and wheel event listeners as `passive` to improve your page's scroll performance. Learn more about adopting passive event listeners.

Avoids `document.write()`

For users on slow connections, external scripts dynamically injected via `document.write()` can delay page load by tens of seconds. Learn how to avoid document.write().

Avoid non-composited animations

Animations which are not composited can be janky and increase CLS. Learn how to avoid non-composited animations CLS

Image elements have explicit `width` and `height`

Set an explicit width and height on image elements to reduce layout shifts and improve CLS. Learn how to set image dimensions CLS

Has a `<meta name="viewport">` tag with `width` or `initial-scale`

A `<meta name="viewport">` not only optimizes your app for mobile screen sizes, but also prevents a 300 millisecond delay to user input. Learn more about using the viewport meta tag. TBT

Page didn't prevent back/forward cache restoration

Many navigations are performed by going back to a previous page, or forwards again. The back/forward cache (bfcache) can speed up these return navigations. Learn more about the bfcache

- Captured at May 9, 2024, 3:00 AM EDT
- Initial page load
- Emulated Moto G Power with Lighthouse 11.6.0
- Slow 4G throttling
- Single page session
- Using Chromium 124.0.0.0 with devtools

Generated by **Lighthouse** 11.6.0 | File an issue