WordPress Performance Team: Roadmap 2024 Draft

Last updated: Jan 16, 2024
Author: flixos90
Contributors: See Props section (Please add your wp.org user name!)

This public draft was for the 2024 roadmap, which is now published. This doc is archived and should no longer be updated.

Everything up to before the "The Big Picture" section will match the contents of https://make.wordpress.org/performance/roadmap-2023/, except that relevant links and the year will be updated.

The drafts below are for the remaining sections, starting with "The Big Picture".

The Big Picture

The WordPress Core Performance Team is focused primarily on improving the following pillars:

- **WordPress interactivity performance:** Many WordPress sites, especially among the more popular ones, struggle with slow reactivity or delays when users interact with the site. This performance aspect can be measured via the "Interaction to Next Paint" metric (INP). INP will become a new Core Web Vital in March, which is another reason to focus on improving interactivity performance especially now.
- **WordPress load time performance:** Continuing the 2023 effort, WordPress sites at scale still struggle the most in this performance aspect, which is measured via the "Largest Contentful Paint" metric (LCP). Enhancing load time performance encompasses both server-side performance, which can be measured via the "Time to First Byte" metric (TTFB), and client-side performance, which can be effectively measured as the result of "LCP minus TTFB". Particularly client-side performance remains a priority, as those efforts have shown greater success than the server-side efforts in impacting Core Web Vitals performance in 2023.

- **Ecosystem activation:** Providing better guard rails, documentation and implementing tooling for plugin and theme developers to follow performance best practices is vital to support the performance efforts focused on WordPress core. This was already a priority in 2023, however it should take a larger role this year, as the past year has shown that certain efforts, such as addressing TTFB bottlenecks, would benefit from a more ecosystem driven approach rather than a WordPress core focused approach.
- **Performance measurement:** As another continuation of 2023 efforts, there is still work to do on facilitating WordPress core and ecosystem developers to assess performance of individual changes or even of the software as a whole. Exploring and establishing methodologies to measure, benchmark, and profile performance reliably will enable the community to inform their priorities and strategy based on data-driven decisions.

The primary area for these four pillars is on *website frontend* performance, i.e. the part of the site which visitors consume. The performance of administrative areas such as the block editor, the site editor, and other areas of WP Admin, while also in scope of the performance team, is not as much in focus as they affect a much smaller subset of users.

Priorities for 2024

The priorities on this roadmap are grouped in four categories, reflecting the areas outlined in the previous section:

WordPress interactivity performance

- **Identifying common key problems and opportunities to improve interactivity bottlenecks in the WordPress**
    - Research effort covering popular WordPress sites, plugins and themes to identify potential high-impact opportunities
    - The focus should be on development patterns that could be fixed via the plugin ecosystem or additional guard rails in WordPress core, preferably without major refactoring such as by using the Interactivity API

    - Findings may additionally influence design decisions for the Interactivity API (see below)
    - Outcome should be a list of more specific INP priority projects
- **Collaborating on Interactivity API efforts with focus on performance (of itself and consumers)**
    - See Interactivity API 1.0 tracking issue to better understand this new API.
    - Performance of the Interactivity API itself, for example reducing its JS footprint
    - Facilitating best practices for interactivity via events, for example, avoid main thread stampedes due to conflicts between multiple plugins or themes
    - As part of this, new JS modules API should also be considered (see modules API tracking issue)
- **Enhancing block editor interactivity performance**
    - Relevant tracking issue
    - More reactive typing performance
    - Improving scalability of themes with many templates and patterns to not negatively affect editor interactivity
    - Reducing page transitions delay
- **Implementing an API for facilitating usage of web workers for certain interactions**
    - In early exploration phase
    - This could take the shape of a WP Scripts API enhancement or an Interactivity API enhancement (e.g. worker-based actions or callbacks)
    - Scope needs to be defined (framework agnostic worker for expensive logic, or worker support with DOM access using a framework
    - Review of existing web worker usage in WordPress core to optimize emoji loader (see ticket)
    - For more info, please see the early discussion on supporting worker modules

WordPress load time performance

- **Enhancing WordPress core translation engine to enhance performance**
    - Expected to improve localization performance by >10% or even more when using PHP-based translation files
    - See WordPress core announcement post, ticket and pull request
    - PHP l10n file support to be added on wordpress.org / GlotPress (see Meta ticket)

- **Implementing client-side detection framework to optimize images and other server-side output for frontend performance considerations**
    - Currently in development as a new Performance Lab module/plugin (see overview issue)
    - Client-side mechanism to detect LCP image reliably, including support for different LCP images between desktop and mobile viewports
    - Implementing the feature in WordPress core should be discussed/considered
    - Making the approach reusable for other client-side heuristics should be explored (e.g. more accurate image "sizes" attribute, avoiding lazy-loading in-viewport images)
- **Improving WordPress core's logic to provide a more accurate "sizes" attribute for images**
    - Originally introduced in WordPress 4.4, at the time WordPress had almost no way of knowing the intended layout of an image and therefore only came with a reasonable default value, intended to be modified by theme developers
    - With additional layout information present especially in block themes, automatically enhancing accuracy of the attribute is now a realistic opportunity
    - Also encompasses support for new `sizes="auto"` web API enhancement (see Performance Lab module PR)
- **Enabling client-side image compression and thumbnail generation on upload**
    - This enhancement improves UX for users on less capable servers, helps reduce file size, and facilitates usage of more modern image formats
    - See Gutenberg issue and Media Experiments repository, plus this overview issue
- **Adding support for the AVIF image format**
    - Loading AVIF images can be much faster than JPEG and WebP, and the format is supported by all major browsers as of January 2024
    - Processing AVIF on the server typically requires PHP 8.1+, though support for this could be greatly expanded in combination with the previous client-side image thumbnail generation effort
- **Supporting Gutenberg phase 3 (block editor collaboration) with performance focused guidance, enhancements, and review**
    - Performance work needed here is pending further scoping of the implementation

- **Adding support for speculative prerendering and prefetching for near-instance page loads**
    - Support of new Speculation Rules API (also see slide deck)
    - See overview issue for new Performance Lab module/plugin
    - Additional exploration for dynamically using the classic approach of `<link>` tags
    - Could also be used to improve wp-admin navigations
- **Enhancing block theme and classic theme template loading performance**
    - Caching of block theme templates, template parts, and patterns (see ticket)
    - Enhancing performance of parsing `theme.json` (see tracking issue, ticket)
    - Caching of certain aspects around classic theme templates and template parts
    - Lazy-loading template related logic only if it is used (e.g. via #59532, #59969)
- **Exploring improvements to page caching strategies**
    - Continued research into the impact of popular page caching plugins on TTFB outcomes
    - Exploration of opportunities in WordPress core, e.g. better guard rails (keeping in mind that bundling a file caching mechanism into core was removed in WordPress 2.5 as a design decision)
- **Optimizing oEmbeds with lazy iframes and delayed JavaScript execution**
    - Launch of a new plugin to test lazy iframes and JavaScript execution delay for oEmbed blocks
    - Measuring its impact on CWVs in the field
    - Implementation of related core enhancements based on plugin learnings
- **Providing additional API enhancements to facilitate more effective loading of autoloaded options**
    - Encouraging explicit usage of autoload flag via API enhancements
    - Disabling autoload for excessively large options (see ticket)
    - Enabling site owners to manually address problems with large autoloaded options via Site Health (see issue)

Ecosystem activation

- **Increasing Interactivity API adoption in the plugin and theme ecosystems by contributing documentation, examples, and collaboration**
    - Outreach effort and boilerplate code for common use-cases, covering both blocks and classic scenarios like themes and plugins manually rendering HTML

    - Proof of concept pull request for using Interactivity API in Twenty Twenty-One
- **Reducing jQuery usage in WordPress themes and plugins by relying on vanilla JavaScript or Interactivity API**
    - Early Trac tickets and pull requests exist e.g. for Twenty Twelve, Twenty Fifteen, and Twenty Sixteen to use vanilla JS instead of jQuery
    - Related is the above proof of concept PR for using Interactivity API in a default theme
- **Increasing adoption of script loading strategies in the plugin and theme ecosystems through documentation and training best practices**
    - Continued effort related to the API introduced in WordPress 6.3
    - See also relevant 6.4 enhancements
- **Documenting and scaling best practices for effectively autoloading options in the plugin ecosystem**
    - Enhancing the very limited documentation about considering the "autoload" value when storing options in WordPress (e.g. posts on WordPress Developer Blog)
    - Preferably this should wait until the previously mentioned additional enhancements around autoloading options have launched in WordPress core
- **Facilitating adoption of the WordPress plugin checker once stable**
    - Implementation and documentation of an easy-to-use GitHub Action as part of the WordPress project
    - Outreach efforts with plugin developers

Performance measurement

- **Launching WordPress plugin checker 1.0 based on enhanced architecture and additional checks**
    - Currently in development in the "trunk" branch of this repository
    - See 1.0 overview issue
- **Scaling automated performance testing in plugins and themes via reusable environment**
    - Enhancements to existing tooling for automated performance testing, making existing scripts more reusable and shareable
    - Implementation of an easy-to-use GitHub Action for performance testing
    - Exploration of more scalable approaches such as Blueprints support for running tests against a larger number of projects

- **Enhancing stability, coverage, and visualization for automated performance tests**
    - Continued exploration for ways to enhance stability of results / usage of a more consistent environment (see overview issue)
    - Revising data collection and visualization tools (e.g. CodeVitals Dashboard)
    - More coverage through representative test scenarios (e.g. more realistic data, testing with i18n, multisite, object cache, ...)

Previous roadmap

The roadmap for 2023 can be found at this link.

Props

Contributing to this roadmap can take various forms, whether it is proactively focusing on a new priority, adding a new section, or reviewing the overall roadmap and providing feedback.

The following people have contributed to this roadmap (in alphabetical order): @adamsilverstein @annezazu @clarkeemily @flixos90 @joemcgill @jorbin @mukesh27 @peterwilsoncc @swissspidy @thekt12 @tweetythierry @westonruter ... you?