How to: Make your site's performance out of this world

You've spent months designing a stunning WordPress site and working with clients and collaborators to create something truly special. Now you've launched it to the public, and yay! You're all done. But, wait...it's kind of slow. You know that site speed is crucial to the success of your business (and that of your client's). Time and time again, you're hit over the head with scary statistics like "nearly half of web users expect a site to load in 2 seconds or less," and "users tend to abandon a site that isn't loaded within 3 seconds."

It's easy to feel frustrated at this point in the project, but don't fret. Before you knock down the door of your web host or add a ton of sketchy plugins, follow these easy steps to make your WordPress site smooth, solid, and super speedy.

Step 1: Test!

Before you get started, it's important to take some measurements and run a few tests to get some benchmarks, so you'll know if the changes you're making are actually improving your site.

To test the speed of your site, your best bet is a combination of WebPageTest and Google PageSpeed Insights. WebPageTest gives you a good idea of the actual time (in seconds) that a site takes to load, and these metrics will help you know whether it's the back-end or front-end causing a site to load slowly. Google PageSpeed Insights is best for looking at how your site is rendered by the browser and can help you identify what you can improve on the front-end.

Step 2: Make sure you've got the best host

The price tag of shared hosting always makes it seem like a bargain, but it comes at a different cost: slow site speed, irregular performance, and frequent downtime. The stress isn't worth it, especially when you have the ability to choose a managed WordPress host (like Flywheel!) that'll take care of a huge chunk of your WordPress performance woes. With Flywheel, support is top-notch, the dashboard is beautiful and easy-to-use, your site will stay secure, and you'll have a whole plethora of features at your disposal. When it comes to hosting, you don't want to mess with last-minute migrations, hacked sites, sub-par servers, and unexpected downtime at the most important times.

Step 3: Audit your plugins

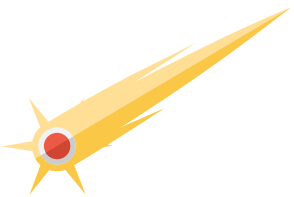
The addition of plugins to a site can bring a lot of functionality and fun stuff that makes your site far beyond basic. Unfortunately, many bring a performance overhead with them. We've seen so many slow sites that are immediately cured just by deactivating a few plugins. The best way to avoid a migraine from managing plugins is to keep performance in

mind as you are developing your site. You don't want to spend weeks relying on a plugin, only for it to become the bane of your existence when you find out it makes your site extremely slow. Read reviews, test them regularly, and be sure you're updating them to the latest version when it's available.

All this negative talk about plugins is only to make you aware of the potential risks involved when using bad ones. These are usually plugins that are developed by inexperienced programmers or those that make a ton of database queries and require intensive logic processing. However, there's good news! There are thousands of worthwhile plugins that don't make your users and readers frustrated when they crawl around your site. The magic trick is just to use those, and not the slow-performing ones.

You can track down which plugins might be causing performance losses on your site quite easily. We generally use a plugin called P3 Profiler that sifts through your whole site and reports back what percentage of total load time each plugin is responsible for using. If you're serious about speed, you'll run a P3 Profiler test each time you activate a new plugin, to ensure you're not installing something that will sacrifice speed. Plus, by benchmarking your plugins right off the bat, you won't waste time messing with a plugin if it's not going to perform well in the long run.

If there are plugins that you just can't live without but still make your site slow, there are ways to continue using them. Generally, if you determine that a plugin is loading slowly, the next step is to increase the memory on your server. With Flywheel, that generally means upping your plan to the next level to get more memory and, thus, more speed. Eliminating the need to do this is the best plan of action, but there probably are a handful of plugins that you might find yourself not willing to ditch (WooCommerce, for example).



Step 4: Cache everything

Caching, both on the server side and client side, is an important part of WordPress site performance. Once a user loads your site for the first time, you can take advantage of browser capabilities to cache the contents of that site locally, so on the next visit the user already has them loaded.

Similarly, on the server side, having a caching layer works wonders for serving your site up very quickly. The most common way to cache server-side with WordPress is by using the W3 Total Cache plugin. This plugin (or one similar) is needed on almost all hosts. The exception to this rule is Flywheel; we handle server-side caching for you behind the scenes with Varnish. Using Varnish instead of a plugin can increase speeds dramatically because Varnish never has to talk to PHP. It serves up cached static copies of your site immediately (without even touching the PHP application layer), whereas plugins are still run by PHP.

On managed hosts like Flywheel, the server-side caching tells the browser how long to cache things by setting 'expires headers'. The browser won't ask the server for more information if those headers tell it not to ask. You mainly have to start thinking about caching when you start installing plugins or third-party themes. Theme and plugin developers can set their own headers that can override your host's default settings, aversely affecting the caching mechanisms.

WooCommerce, for example, disables server-side caching by setting cookies. It's typically not very apparent when plugins or themes disable caching, so a good thing to do is reach out to your hosting provider and ask that they force cache your site. Be careful not to force caching on pages that truly need cookies to work, though, such as WooCommerce cart and checkout pages.

Step 4: Use a CDN

A content delivery network (or CDN) is a network of servers that serves up your website and its assets from different locations based on the user's location. For example, let's say you're not using a CDN and your site is hosted in San Francisco. When someone from, say, Barcelona visits your site, they have to retrieve all your assets from your server in San Francisco. The long distance between the two locations clearly takes longer than if someone from San Francisco loads your site that's hosted in San Francisco. A CDN serves your assets from a bunch of different servers based in New York, Seattle, Omaha, Paris, London, Beijing AND San Francisco (and many more). The idea is that users will hit the server closest to them and not sacrifice load time since there's a smaller distance between them and the server. With a CDN, that same person from Barcelona will now hit a datacenter in London or Paris, instead of San Francisco. Using a CDN will definitely increase the speed of your site for users throughout the world!

Some of the most popular CDNs include Amazon Web Services, CloudFlare, and MaxCDN. Most of them have free plans, but if you pull a lot of visitors and have lots of assets, you'll most likely have to pay for a CDN. They're typically easy to set up, but if you want a super easy setup solution, you should check out Flywheel's MaxCDN add-on.

Step 5: Downsize your static assets

There's some work to be done with static assets prior to uploading them to your server to serve to browsers. This work essentially comes down to compression and minification. You always want to serve the fewest amount of assets possible, compressed as much as you can, in order to make your site as fast as possible.

Images

Images are often the biggest files on a page, responsible for the largest delay in page load time. However, the good thing about images is that unlike CSS and JavaScript, most browsers load them asynchronously. That at least helps with the perceived performance of a page, but you still want to make sure that a.) you're serving as few images as possible and b.) those images are compressed as much as possible.

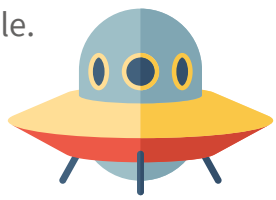
Compression tools are necessary for squeezing out as much excess as possible on images. ImageOptim is a great Mac app that does this well, along with OptiPNG and jpegtran for use with task runners like Grunt. TinyPNG is the go-to web-based compression tool. It can become cumbersome doing this with every image, though, and sometimes not even feasible if multiple people are contributing content. Thankfully, there are some WordPress specific tools available to help with this.

When you have multiple sizes of images that are cropped upon upload (usually defined in most themes), using an image compression WordPress plugin is particularly important. Even if the initial image that you uploaded is as trimmed and compressed as possible, when WordPress generates the new images from that, those won't be compressed.

We have used and recommend TinyPNG for WordPress and Ewww Cloud Optimizer to handle the compression and optimization of images uploaded to WordPress.

CSS

The most important thing to do with your CSS before it's delivered to the browser is simply to compress it and remove unused selectors. Getting a good development workflow down when developing themes makes this process trivial, especially if you're using a pre-processor like Sass. We'd recommend Grunt, which is a JavaScript task runner that executes commands for you while developing. There's a plugin for Grunt called grunt-contrib-sass that simply compiles all your Sass files down into one, minifies it, and compresses it. Throw in the grunt-contrib-watch plugin on top of that, and it will run the Sass task whenever you save a file. Easy!



Don’t lose sleep if you’re not using the latest and greatest CSS methodology, but try to follow some sort of standard while authoring CSSto avoid duplication and huge file sizes.

JavaScript

The golden rules of optimizing JavaScript are simple: Serve as few JavaScript files as possible, minify, and concatenate. Third-party WordPress plugins can be detrimental by bloating your document with unminified blocking JavaScript files, so it’s important to be mindful when choosing plugins. Ideally, you’d concatenate ALL JavaScript files into one and then minify the heck out of it. For times when it’s not possible to concatenate all your files into one, there are HTML attributes called “async” and “defer” that can be used to load JavaScript files asynchronously or once the rest of the page is loaded.

Step 6: Take care in writing your theme logic

There are many things you can do from a theme development perspective to make your back-end performance as high as possible. Nasty loops that do tons of comparisons and use lots of memory (if, if, if, else, if, else, so on) can definitely slow down the loading of a page, as all that logic takes time. There are lots of articles on performant PHP, but our best advice is to just take care in writing your theme logic!

One tool that has saved us from a logic intensive loop taking forever to load is the Transients API. WordPress transients store cached data in the database temporarily, which means your logic only has to run once (whenever the first visitor loads the page) and then the results of that are stored in the database. The Codex has good documentation on the usage of transients, as well as a host of other articles found with a Google search.

Step 7: Serve up assets on a silver platter

How you serve up static assets like images, CSS, and JavaScript is critical to the way your site loads.

CSS

Once your stylesheet is compressed and ready to go, the easiest (and by far, the standard) way to load it is to just reference it in the <head>. That way, the browser loads and parses it before the rest of the DOM is loaded.

```
<head>
    <link rel="stylesheet" href="path/to/style.css">
</head>
```

However, there's a fairly new technique where "critical" styles are inlined in the <head> and then the full stylesheet is loaded asynchronously using JavaScript. Critical styles are defined as those that are needed to paint everything that the user first sees ("above the fold") when they initially load your page. The best way to grab critical styles from a page is by using Grunt (or another task runner) paired with a task like grunt-critical or grunt-criticalcss. The Penthouse web generator is also a solid way of extracting critical styles.

Once you have the critical CSS ready, it's as simple as inserting it into a <style> tag in the head.

```
<head>
    <style>/* all of your critical styles go here
*/</style>
```

Now to get the full stylesheet loaded, a JavaScript loader like loadCSS is required. The Filament Group has a much more extensive article on using this technique, including examples with loadCSS.

JavaScript

The most common place nowadays to reference JavaScript is at the bottom of your document, right before the closing tag. However, there are more advanced techniques to load JavaScript. Again, the Filament Group has done tons of research on this and has several open source projects that you can use if you're aiming for a super fast page load time (we're in no way affiliated with Filament Group, by the way; we just think their stuff is awesome). The best approach seems to be to load scripts dynamically by inlining a small function in the <head>; that then appends script tags without blocking the rest of the page. For more information, check out the loadJS script.

After you've gone through these performance-enhancing practices, if your site is still really slow, we'd recommend hiring a developer who can look into it and find the source of the problem and/or choosing a new WordPress host.

What's Flywheel?

Flywheel offers WordPress hosting and site management tools for designers and agencies. Flywheel's simple collaboration tools, built-in staging sites, robust architecture and WordPress specific support team allows agencies to save time and launch more projects.

Stop wasting time on server management, security plugins, caching, and all those other boring repetitive tasks that take your focus away from growing your business and jeopardize your relationship with clients. Get Flywheel and get back to doing what you love.

Contact Sales

hello@getflywheel.com
(888) 928-8882

Or, sign up at getflywheel.com