

The WordPress hero image **that fails LCP.**

Three causes, in the order worth checking them, for a WordPress hero image that still fails Largest Contentful Paint. Work down until one of them is yours, then stop and fix that one.

WordPress 6.3 and later already skips lazy loading on the first few images and adds `fetchpriority` to the hero. Everything here is about what is getting in the way of that.

BEFORE YOU CHANGE ANYTHING

- ☐ **WordPress is on 6.3 or later**
Below that, core is not prioritising the hero at all and the old advice still applies.
- ☐ **View source on the live page, not the DevTools element inspector**
The inspector shows the DOM after scripts have run, which hides the thing you are looking for.
- ☐ **The LCP element named by PageSpeed Insights is written down**
If it is a div rather than an img, you are on cause 2 and nothing in cause 1 will help.
- ☐ **The date of every change you make is written down as you go**
Field data moves 28 days later. Without dates, nobody can tell which change moved it.

CAUSE 1: A PLUGIN IS OVERRIDING CORE

- ☐ **The hero img has no `loading="lazy"` in the page source**
Lazy-loading the LCP image pushes the request behind the initial render, and published measurements put the cost at 500ms and up.
- ☐ **The `src` is the real image, not a placeholder with the URL in a `data-` attribute**
A plugin deferring the image behind JavaScript looks the same to a visitor and very different to the browser.
- ☐ **`fetchpriority="high"` is present on the hero img**
Core adds it on its own. If it has gone, something removed it.
- ☐ **The hero is excluded in the plugin's own lazy-load settings, by CSS class**
Better than switching the plugin off, because it is probably doing useful work elsewhere. Exclude by class, not by URL: URLs change when an image is re-uploaded.

CAUSE 2: THE HERO IS NOT AN IMG

- ☐ **The hero is a real img element in the initial HTML, not a CSS background**
The browser cannot discover a background image until it has downloaded and parsed the stylesheet, so the preload scanner never sees it.
- ☐ **If it is a carousel, the first slide is rendered server-side as static markup**
A first slide injected by JavaScript does not exist when the page starts painting.

-
- ☐ **Where the markup cannot change, a link `rel=preload as=image` is in the head for that exact file**

With `fetchpriority="high"`, and `imagesrcset` so you preload the variant the page will actually ask for.

- ☐ **The network panel shows the hero downloaded once, not twice**

A preload that does not match what the page requests downloads the image twice and makes the page heavier.

CAUSE 3: IT IS SIMPLY TOO HEAVY

- ☐ **The hero has a `srcset`, and the page builder has not overridden it**

Without one, a phone downloads a 2400px-wide image to display it at 390px.

- ☐ **The `sizes` attribute matches how big the image really is on screen**

`sizes` is what tells the browser which variant to take. A perfect `srcset` with a wrong `sizes` never serves the small one.

- ☐ **WebP or AVIF, with a fallback**

Often half the bytes of a JPEG at the same quality.

- ☐ **Width and height are on the `img`**

Without them you pay for layout shift as well as paint.

- ☐ **The transferred size on a throttled mobile profile is under about 150KB**

The transferred size, not the file size in the media library. Under that, the hero is rarely your LCP problem.

IF IT IS NONE OF THE THREE

- ☐ **Web fonts have `font-display: swap`, and the one used above the fold is preloaded**

The browser will hold text paint waiting for a font, and on a text-heavy hero the LCP element may be the heading.

- ☐ **Time to first byte checked before blaming the image**

If the first byte is 800ms, nothing you do to the hero recovers it. That is hosting or caching, not markup.

PROVE IT WORKED

- ☐ **Diagnosed in the lab, verified in the field**

The lab score responds to almost anything, including changes that do nothing for real users. Field data only responds to the real problem.

- ☐ **Search Console's Core Web Vitals report is the thing being watched, not the score**

It is drawn from real Chrome users over a rolling 28-day window, and it is what search actually considers.

- ☐ **Nothing else changed while you wait for the window to turn over**

Change a second thing and you will never know which one moved the number.

Take it, change it, use it on your own sites. No need to credit me. adamroe.co.uk/blog/wordpress-hero-image-lcp