Google: Why Lazy Loading Can Delay Largest Contentful Paint (LCP) via @sejournal, @MattGSouthern

In a recent episode of Google's Search Off the Record podcast, Martin Splitt and John Mueller discussed when lazy loading helps and when it can slow pages.

Splitt used a real-world example on developers.google.com to illustrate a common pattern: making every image lazy by default can delay Largest Contentful Paint (LCP) if it includes above-the-fold visuals.

Splitt said:

> *"The content management system that we are using for developers.google.com … defaults all images to lazy loading, which is not great."*

Splitt used the example to explain why lazy-loading hero images is risky: you tell the browser to wait on the most visible element, which can push back LCP and cause layout shifts if dimensions aren't set.

Splitt said:

> *"If you are using lazy loading on an image that is immediately visible, that is most likely going to have an impact on your largest contentful paint. It's like almost guaranteed."*

How Lazy Loading Delays LCP

LCP measures the moment the largest text or image in the initial viewport is painted.

Normally, the browser’s preload scanner finds that hero image early and fetches it with high priority so it can paint fast.

When you add loading="lazy" to that same hero, you change the browser’s scheduling:

- The image is treated as lower priority, so other resources start first.
- The browser waits until layout and other work progress before it requests the hero image.
- The hero then competes for bandwidth after scripts, styles, and other assets have already queued.

That delay shifts the paint time of the largest element later, which increases your LCP.

On slow networks or CPU-limited devices, the effect is more noticeable. If width and height are missing, the late image can also nudge layout and feel “jarring.”

SEO Risk With Some Libraries

Browsers now support a built-in loading attribute for images and iframes, which removes the need for heavy JavaScript in standard scenarios. WordPress adopted native lazy loading by default, helping it spread.

Splitt said:

> *“Browsers got a native attribute for images and iframes, the loading attribute … which makes the browser take care of the lazy loading for you.”*

Older or custom lazy-loading libraries can hide image URLs in nonstandard attributes. If the real URL never lands in src or srcset in the HTML Google renders, images may not get picked up for indexing.

Splitt said:

> *"We've seen multiple lazy loading libraries … that use some sort of data-source attribute rather than the source attribute… If it's not in the source attribute, we won't pick it up if it's in some custom attribute."*

How To Check Your Pages

Use Search Console's URL Inspection to review the **rendered HTML** and confirm that above-the-fold images and lazy-loaded modules resolve to standard attributes. Avoid relying on the screenshot.

Splitt advised:

> *"If the rendered HTML looks like it contains all the image URLs in the source attribute of an image tag … then you will be fine."*

Ranking Impact

Splitt framed ranking effects as modest. Core Web Vitals contribute to ranking, but he called it "a tiny minute factor in most cases."

What You Should Do Next

- Keep hero and other above-the-fold images eager with width and height set.
- Use native loading="lazy" for below-the-fold images and

iframes.
- If you rely on a library for previews, videos, or dynamic sections, make sure the final markup exposes real URLs in standard attributes, and confirm in rendered HTML.

Looking Ahead

Lazy loading is useful when applied selectively. Treat it as an opt-in for noncritical content.

Verify your implementation with rendered HTML, and watch how your LCP trends over time.

Featured Image: Screenshot from YouTube.com/GoogleSearchCentral, August 2025.

https://www.searchenginejournal.com/google-why-lazy-loading-can-delay-largest-contentful-paint-lcp/554418/