CODEBLOGGS / LightHouse Analysis / **Home Page**
PERFORMANCE suggestions for improvement
Current note : **66/100**

Largest Contentful Paint (LCP):

- o Issue: The largest contentful element (img#headbar-logo-img) takes 2,620 ms to paint within the viewport.
- o Solution: Optimize the loading of this image by compressing it and serving appropriately sized images. Consider lazy loading or preloading to improve LCP.

Reduce Unused JavaScript:

- o Issue: Significant potential savings of 946 KiB by reducing unused JavaScript, with the bulk coming from various modules.
- o Solution: Review and eliminate unnecessary JavaScript modules. Defer loading scripts until they are required, improving overall performance and reducing network activity.

Minify JavaScript:

- o Issue: Potential savings of 361.9 KiB by minifying JavaScript, including files from extensions and plugins.
- o Solution: Minify JavaScript files to reduce payload sizes and script parse time, contributing to faster page loads.

Preconnect to Required Origins:

- o Issue: Potential savings of 100 ms by preconnecting to Google Fonts CDN.
- o Solution: Add preconnect or dns-prefetch hints for important third-party origins like Google Fonts to establish early connections and improve page load speed.

Preload Largest Contentful Paint Image:

- o Issue: Potential savings of 50 ms by preloading the LCP image.
- o Solution: If the LCP element (img#headbar-logo-img) is dynamically added, preload the image to improve LCP and overall page performance.

Properly Size Images:

- o <u>Issue</u>: Potential savings of 10 KiB by serving appropriately sized images.
- o <u>Solution</u>: Resize and serve images in the correct dimensions to save data and improve load times, especially for the identified image (img#headbar-logo-img).

Minify CSS:

- o <u>Issue</u>: Potential savings of 403 KiB by minifying CSS, including Bootstrap and custom styles.
- o <u>Solution</u>: Minify CSS files to reduce network payload sizes, contributing to faster page rendering.

Serve Static Assets with an Efficient Cache Policy:

- o <u>Issue</u>: Two resources found without a specified cache TTL (Time To Live).
- o <u>Solution</u>: Set an efficient cache policy for static assets (js/bundle.js and media/CodeBlogg...1627d07....png) to speed up repeat visits.

Reduce Unused CSS:

- o <u>Issue</u>: Potential savings of 483 KiB by reducing unused CSS, including Bootstrap and custom styles.
- o <u>Solution</u>: Remove or defer unused CSS rules to reduce network activity and improve overall page performance.

Avoid Serving Legacy JavaScript to Modern Browsers:

- o <u>Issue</u>: No potential savings, but consider adopting a modern script deployment strategy.
- o <u>Solution</u>: Use modern JavaScript deployment strategies with module/nomodule feature detection to ship less code to modern browsers while retaining support for legacy browsers.

Page Prevented Back/Forward Cache Restoration:

- o <u>Issue</u>: Failure to enter back/forward cache due to WebSocket usage.
- o <u>Solution</u>: Consider optimizing WebSocket usage or be aware that pages with WebSocket may not enter back/forward cache. Check for updates on browser support for WebSocket in back/forward cache.