

Building a Better Web with Lighthouse



By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

[Lighthouse](#) is an [open-source](#), automated tool for improving the quality of your web apps. You can install it as a [Chrome Extension](#) or run it as a Node command line tool. When you give Lighthouse a URL, it runs a barrage of tests against the page and then generates a report explaining how well the page did and indicating areas for improvement.



Since Google I/O, we've been hard at work making Lighthouse an awesome companion for building great Progressive Web Apps:

- Welcomed 50 new contributors to the project
- Shipped 15 releases
- Added ~20 additional audit tests (~50 total)

Today, we're happy to announce the 1.3 release of Lighthouse. Lighthouse 1.3 includes a bunch of big new features, audits, and the usual bug fixes. You can install it from npm (`npm i -g lighthouse`) or [download the extension](#) from the Chrome Web Store.

So what's new?

New look and feel

If you've used an earlier version of Lighthouse, you may have noticed that the logo is new! The HTML report and Chrome Extension have also undergone a complete refresh, with a cleaner presentation of scoring and more consistency across audit results. We've also

added helpful debug information when you fail a test and include pointers to recommended workarounds.

The screenshot shows the Lighthouse interface with a sidebar on the left containing 'Progressive Web App', 'Best Practices', 'Performance Metrics', and 'Fancier stuff'. The main content area displays the 'Progressive Web App' audit results for the URL <https://www.chromestatus.com/features>, generated on 12/10/2016 at 9:40:21 PM PST. The audit score is 100. The results are categorized into three sections: 'App can load on offline/flaky connections', 'Page load performance is fast', and 'Site is progressively enhanced'. The first section includes two passing checks: 'Has a registered Service Worker' and 'URL responds with a 200 when offline'. The second section details performance metrics: First meaningful paint (664.2ms), Perceptual Speed Index (753), First Visual Change (409ms), Last Visual Change (1227ms), Estimated Input Latency (58.3ms), and Time To Interactive (692.2ms). The third section notes that the site is progressively enhanced.

Lighthouse
Version: 1.2.2

BETA

Results for: <https://www.chromestatus.com/features>
Generated on: 12/10/2016, 9:40:21 PM PST

Progressive Web App

Best Practices

Performance Metrics

Fancier stuff

100 Progressive Web App

These audits validate the aspects of a Progressive Web App.

App can load on offline/flaky connections

Ensuring your web app can respond when the network connection is unavailable or flaky is critical to providing your users a good experience. This is achieved through use of a [Service Worker](#).

- ✓ Has a registered Service Worker ?
- ✓ URL responds with a 200 when offline ?

Page load performance is fast

Users notice if sites and apps don't perform well. These top-level metrics capture the most important perceived performance concerns.

- 100** First meaningful paint: **664.2ms** (target: 1,600ms) ?
- 100** Perceptual Speed Index: **753** (target: 1,250) ?
 - First Visual Change: **409ms**
 - Last Visual Change: **1227ms**
- 90** Estimated Input Latency: **58.3ms** (target: 50ms) ?
- 100** Time To Interactive (alpha): **692.2ms** (target: 5,000ms) ?
 - Content scrolls at 60fps (Coming soon)
 - Touch input gets a response in < 150ms (Coming soon)
 - App is interactive without jank after the first meaningful paint (Coming soon)

Site is progressively enhanced

Progressive enhancement means that everyone can access the basic content and functionality of a page in any browser, and those without certain browser features may receive a reduced but still functional experience.

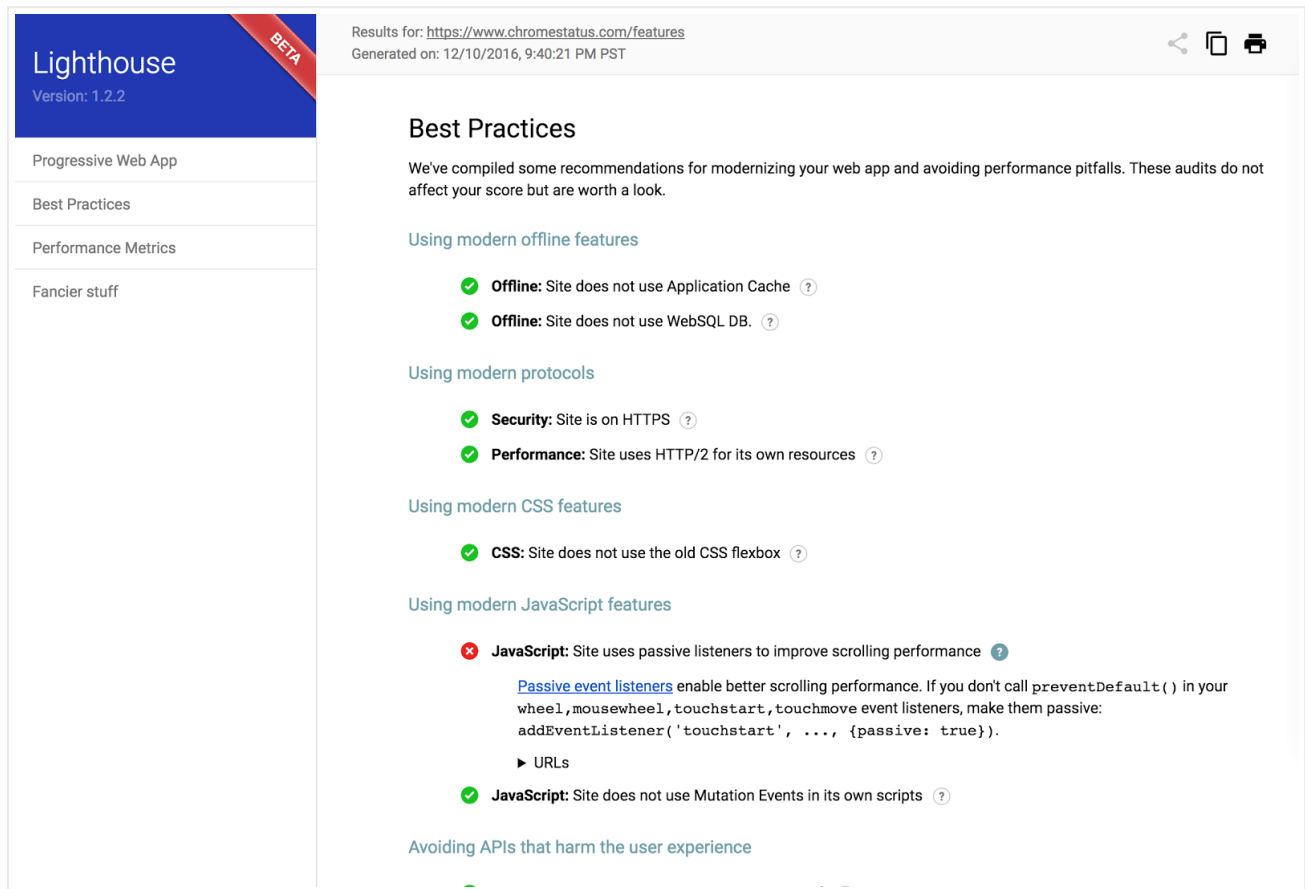
New Best Practices

To date, Lighthouse has focused on performance metrics and the quality of PWAs. However, an overarching goal of the project is to provide a guidebook for all of web development. This includes guidance on general best practices, performance and accessibility tips, and end-to-end help on making quality apps.

"Do Better Web" is an effort within the Lighthouse project to help developers do better on the web. In other words, help them modernize and optimize their web applications. Oftentimes, web devs use outdated practices, anti-patterns, or hit known performance pitfalls without realizing it. For example, it is widely known that JS-based animations should be done with `requestAnimationFrame()` instead of `setInterval()`. However, if the developer is unaware of the newer API, their web app needlessly suffers.

Lighthouse 1.3 includes 20+ new best practice suggestions ranging from tips for modernizing CSS & JavaScript features to performance recommendations like: "Reduce the

number of render-blocking assets", "Use passive event listeners to improve scrolling performance".



The screenshot shows the Lighthouse report viewer interface. On the left is a sidebar with the Lighthouse logo, version 1.2.2, and a 'BETA' badge. The sidebar has four menu items: 'Progressive Web App', 'Best Practices' (which is selected), 'Performance Metrics', and 'Fancier stuff'. The main content area displays the 'Best Practices' section. At the top, it says 'Results for: https://www.chromestatus.com/features' and 'Generated on: 12/10/2016, 9:40:21 PM PST'. Below this is a heading 'Best Practices' followed by an introductory paragraph. The section is divided into five sub-sections: 'Using modern offline features', 'Using modern protocols', 'Using modern CSS features', 'Using modern JavaScript features', and 'Avoiding APIs that harm the user experience'. The 'Using modern JavaScript features' sub-section contains two items: a failed audit for 'JavaScript: Site uses passive listeners to improve scrolling performance' with a red 'x' icon and a link to 'Passive event listeners' with an explanatory paragraph and code snippet, and a passed audit for 'JavaScript: Site does not use Mutation Events in its own scripts' with a green checkmark icon.

Lighthouse
Version: 1.2.2
BETA

Results for: <https://www.chromestatus.com/features>
Generated on: 12/10/2016, 9:40:21 PM PST

Progressive Web App
Best Practices
Performance Metrics
Fancier stuff

Best Practices

We've compiled some recommendations for modernizing your web app and avoiding performance pitfalls. These audits do not affect your score but are worth a look.

Using modern offline features

- ✓ **Offline:** Site does not use Application Cache ?
- ✓ **Offline:** Site does not use WebSQL DB. ?

Using modern protocols

- ✓ **Security:** Site is on HTTPS ?
- ✓ **Performance:** Site uses HTTP/2 for its own resources ?

Using modern CSS features

- ✓ **CSS:** Site does not use the old CSS flexbox ?

Using modern JavaScript features

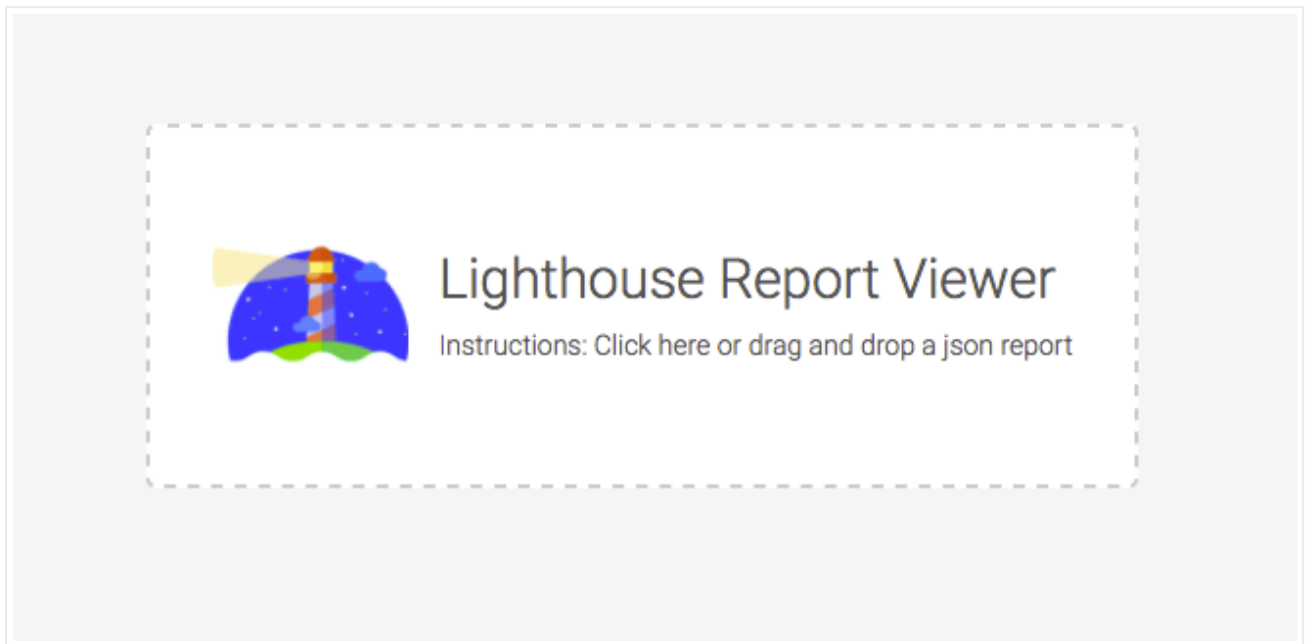
- ✗ **JavaScript:** Site uses passive listeners to improve scrolling performance ?
[Passive event listeners](#) enable better scrolling performance. If you don't call `preventDefault()` in your `wheel, mousewheel, touchstart, touchmove` event listeners, make them passive:
`addEventListener('touchstart', ..., {passive: true}).`
► URLs
- ✓ **JavaScript:** Site does not use Mutation Events in its own scripts ?

Avoiding APIs that harm the user experience

We'll continue to add more recommendations over time. If you have suggestions for best practices or want to help us write an audit, [file an issue](#) on Github.

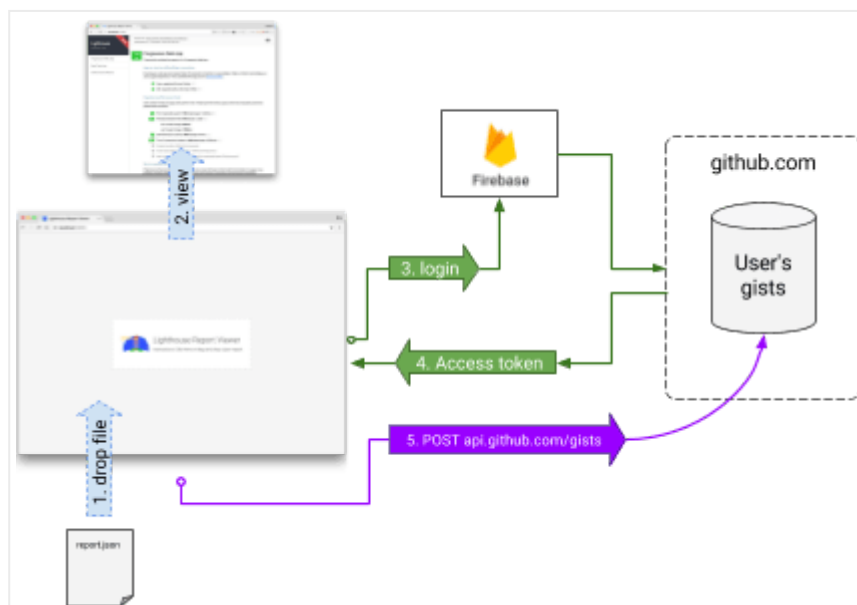
Report Viewer

Last but not least, we're excited to announce a new web viewer for Lighthouse results. Visit googlechrome.github.io/lighthouse/viewer, drag and drop the output of a Lighthouse run (or click to upload your file), and voila. "Insta" Lighthouse HTML report.



Report Viewer

Lighthouse Viewer also lets you share reports with others. Clicking the share icon will sign you in to Github. We stash reports as secret gist in your account so you can easily delete a shared report or update it later on. Using Github for data storage also means you get version control for free!



Viewer Architecture

Existing reports can be reloaded by Lighthouse Viewer by adding `?gist=GIST_ID` to the URL:



Viewer Architecture 2

For all the details on the latest in Lighthouse, see the [full release notes](#) over on Github. As always, [hit us up](#) to [report bugs](#), file feature requests, or brainstorm [ideas](#) on what you'd like to see next.

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated July 2, 2018.