chrome-extension://iaapelhigghggmbghgcmnjeegdblejdp/_generated_background_page.html

- Print Summary
- Print Expanded
- Copy JSON
- Save as HTML
- Save as JSON
- Open in Viewer
- Toggle Dark Theme

?	?	?	?	?
Performance	Accessibility	Best Practices	SEO	Progressive Web App

0–49 50–89 90–100

There were issues affecting this run of Lighthouse:

- Something went wrong with recording the trace over your page load. Please run Lighthouse again. (NO_FCP)

?

Performance

Metrics

▲ First Contentful Paint Error!
Required traces gatherer did not run.

▲ First Meaningful Paint Error!
Required traces gatherer did not run.

▲ Speed Index Error!
Required traces gatherer did not run.

▲ First CPU Idle Error!
Required traces gatherer did not run.

▲ Time to Interactive Error!
Required traces gatherer did not run.

▲ Max Potential First Input Delay Error!
Required traces gatherer did not run.

Values are estimated and may vary. The performance score is based only on these metrics.

Opportunities — These suggestions can help your page load faster. They don't directly affect the Performance score.

Opportunity	Estimated Savings
▲ Eliminate render-blocking resources	Error!
Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn more.	
▲ Properly size images	Error!
Serve images that are appropriately-sized to save cellular data and improve load time. Learn more.	
▲ Defer offscreen images	Error!
Consider lazy-loading offscreen and hidden images after all critical resources have finished loading to lower time to interactive. Learn more.	
▲ Minify CSS	Error!
Minifying CSS files can reduce network payload sizes. Learn more.	
▲ Minify JavaScript	Error!
Minifying JavaScript files can reduce payload sizes and script parse time. Learn more.	
▲ Remove unused CSS	Error!
Remove dead rules from stylesheets and defer the loading of CSS not used for above-the-fold content to reduce unnecessary bytes consumed by network activity. Learn more.	
▲ Efficiently encode images	Error!
Optimized images load faster and consume less cellular data. Learn more.	
▲ Serve images in next-gen formats	Error!
Image formats like JPEG 2000, JPEG XR, and WebP often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more.	
▲ Enable text compression	Error!

Text-based resources should be served with compression (gzip, deflate or brotli) to minimize total network bytes. Learn more.

▲ Preconnect to required origins Error!

Consider adding `preconnect` or `dns-prefetch` resource hints to establish early connections to important third-party origins. Learn more.

▲ Server response times are low (TTFB) Error!

Time To First Byte identifies the time at which your server sends a response. Learn more.

▲ Avoid multiple page redirects Error!

Redirects introduce additional delays before the page can be loaded. Learn more.

▲ Preload key requests Error!

Consider using `<link rel=preload>` to prioritize fetching resources that are currently requested later in page load. Learn more.

▲ Use video formats for animated content Error!

Large GIFs are inefficient for delivering animated content. Consider using MPEG4/WebM videos for animations and PNG/WebP for static images instead of GIF to save network bytes. Learn more

Diagnostics — More information about the performance of your application. These numbers don't directly affect the Performance score.

▲ Avoids enormous network payloads — Error!

Large network payloads cost users real money and are highly correlated with long load times. Learn more.

▲ Uses efficient cache policy on static assets — Error!

A long cache lifetime can speed up repeat visits to your page. Learn more.

▲ Avoids an excessive DOM size — Error!

A large DOM will increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn more.

▲ Avoid chaining critical requests — Error!

The Critical Request Chains below show you what resources are loaded with a high priority. Consider reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources to improve page load. Learn more.

▲ User Timing marks and measures — Error!

Consider instrumenting your app with the User Timing API to measure your app's real-world performance during key user experiences. Learn more.

▲ JavaScript execution time — Error!

Consider reducing the time spent parsing, compiling, and executing JS. You may find delivering smaller JS payloads helps with this. Learn more.

▲ Minimizes main-thread work — Error!

Consider reducing the time spent parsing, compiling and executing JS. You may find delivering smaller JS payloads helps with this. Learn more

▲ All text remains visible during webfont loads — Error!

Leverage the font-display CSS feature to ensure text is user-visible while webfonts are loading. Learn more.

▲ Keep request counts low and transfer sizes small — Error!

To set budgets for the quantity and size of page resources, add a budget.json file. Learn more.

▲ Minimize third-party usage — Error!

Third-party code can significantly impact load performance. Limit the number of redundant third-party providers and try to load third-party code after your page has primarily finished loading. Learn more.

?

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

Navigation — These are opportunities to improve keyboard navigation in your application.

▲ `[accesskey]` values are unique — Error!

Access keys let users quickly focus a part of the page. For proper navigation, each access key must be unique. Learn more.

▲ The page contains a heading, skip link, or landmark region — Error!

Adding ways to bypass repetitive content lets keyboard users navigate the page more efficiently. Learn more.

▲ No element has a `[tabindex]` value greater than 0 — Error!

A value greater than 0 implies an explicit navigation ordering. Although technically valid, this often creates frustrating experiences for users who rely on assistive technologies. Learn more.

ARIA — These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

▲ `[aria-*]` attributes match their roles — Error!

Each ARIA `role` supports a specific subset of `aria-*` attributes. Mismatching these invalidates the `aria-*` attributes. Learn more.

▲ `[role]`s have all required `[aria-*]` attributes — Error!

Some ARIA roles have required attributes that describe the state of the element to screen readers. Learn more.

▲ Elements with an ARIA `[role]` that require children to contain a specific `[role]` have all required children. — Error!

Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. Learn more.

▲ `[role]`s are contained by their required parent element — Error!

Some ARIA child roles must be contained by specific parent roles to properly perform their intended accessibility functions. Learn more.

▲ `[role]` values are valid — Error!

ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more.

▲ `[aria-*]` attributes have valid values — Error!

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. Learn more.

▲ `[aria-*]` attributes are valid and not misspelled — Error!

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more.

Audio and video — These are opportunities to provide alternative content for audio and video. This may improve the experience for users with hearing or vision impairments.

▲ `<audio>` elements contain a `<track>` element with `[kind="captions"]` — Error!

Captions make audio elements usable for deaf or hearing-impaired users, providing critical information such as who is talking, what they're saying, and other non-speech information. Learn more.

▲ `<video>` elements contain a `<track>` element with `[kind="captions"]` — Error!

When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more.

▲ `<video>` elements contain a `<track>` element with `[kind="description"]` — Error!

Audio descriptions provide relevant information for videos that dialogue cannot, such as facial expressions and scenes. Learn more.

Names and labels — These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

▲ Buttons have an accessible name — Error!

When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn more.

▲ Document has a `<title>` element — Error!

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more.

▲ `<frame>` or `<iframe>` elements have a title — Error!

Screen reader users rely on frame titles to describe the contents of frames. Learn more.

▲ Image elements have `[alt]` attributes — Error!

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more.

▲ `<input type="image">` elements have `[alt]` text — Error!

When an image is being used as an `<input>` button, providing alternative text can help screen reader users understand the purpose of the button. Learn more.

▲ Form elements have associated labels — Error!

Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more.

▲ Links have a discernible name — Error!

Link text (and alternate text for images, when used as links) that is discernible, unique, and focusable improves the navigation experience for screen reader users. Learn more.

▲ `<object>` elements have `[alt]` text — Error!

Screen readers cannot translate non-text content. Adding alt text to `<object>` elements helps screen readers convey meaning to users. Learn more.

Contrast — These are opportunities to improve the legibility of your content.

▲ Background and foreground colors have a sufficient contrast ratio — Error!

Low-contrast text is difficult or impossible for many users to read. Learn more.

Tables and lists — These are opportunities to to improve the experience of reading tabular or list data using assistive technology, like a screen reader.

▲ `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>` or `<template>` elements. — Error!

When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn more.

▲ Definition list items are wrapped in `<dl>` elements — Error!

Definition list items (`<dt>` and `<dd>`) must be wrapped in a parent `<dl>` element to ensure that screen readers can properly announce them. Learn more.

▲ Presentational `<table>` elements avoid using `<th>`, `<caption>` or the `[summary]` attribute. — Error!

A table being used for layout purposes should not include data elements, such as the th or caption elements or the summary attribute, because this can create a confusing experience for screen reader users. Learn more.

▲ Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`). — Error!

Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more.

▲ List items (`<li>`) are contained within `<ul>` or `<ol>` parent elements — Error!

Screen readers require list items (`<li>`) to be contained within a parent `<ul>` or `<ol>` to be announced properly. Learn more.

▲ Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the same table. — Error!

Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more.

▲ `<th>` elements and elements with `[role="columnheader"/"rowheader"]` have data cells they describe. — Error!

Screen readers have features to make navigating tables easier. Ensuring table headers always refer to some set of cells may improve the experience for screen reader users. Learn more.

Best practices — These items highlight common accessibility best practices.

▲ `[id]` attributes on the page are unique — Error!

The value of an id attribute must be unique to prevent other instances from being overlooked by assistive technologies. Learn more.

▲ The document does not use `<meta http-equiv="refresh">` — Error!

Users do not expect a page to refresh automatically, and doing so will move focus back to the top of the page. This may create a frustrating or confusing experience. Learn more.

▲ `[user-scalable="no"]` is not used in the `<meta name="viewport">` element and the `[maximum-scale]` attribute is not less than 5. — Error!

Disabling zooming is problematic for users with low vision who rely on screen magnification to properly see the contents of a web page. Learn more.

Internationalization and localization — These are opportunities to improve the interpretation of your content by users in different locales.

▲ `<html>` element has a `[lang]` attribute — Error!

If a page doesn't specify a lang attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly. Learn more.

▲ `<html>` element has a valid value for its `[lang]` attribute — Error!

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn more.

▲ `[lang]` attributes have a valid value — Error!

Specifying a valid BCP 47 language on elements helps ensure that text is pronounced correctly by a screen reader. Learn more.

Additional items to manually check (11) — These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

The page has a logical tab order

Tabbing through the page follows the visual layout. Users cannot focus elements that are offscreen. Learn more.

Interactive controls are keyboard focusable

Custom interactive controls are keyboard focusable and display a focus indicator. Learn more.

Interactive elements indicate their purpose and state

Interactive elements, such as links and buttons, should indicate their state and be distinguishable from non-interactive elements. Learn more.

The user's focus is directed to new content added to the page

If new content, such as a dialog, is added to the page, the user's focus is directed to it. Learn more.

User focus is not accidentally trapped in a region

A user can tab into and out of any control or region without accidentally trapping their focus. Learn more.

Custom controls have associated labels

Custom interactive controls have associated labels, provided by aria-label or aria-labelledby. Learn more.

Custom controls have ARIA roles

Custom interactive controls have appropriate ARIA roles. Learn more.

Visual order on the page follows DOM order

DOM order matches the visual order, improving navigation for assistive technology. Learn more.

Offscreen content is hidden from assistive technology

Offscreen content is hidden with display: none or aria-hidden=true. Learn more.

Headings don't skip levels

Headings are used to create an outline for the page and heading levels are not skipped. Learn more.

HTML5 landmark elements are used to improve navigation

Landmark elements (<main>, <nav>, etc.) are used to improve the keyboard navigation of the page for assistive technology. Learn more.

?

Best Practices

▲ Avoids Application Cache — Error!

Application Cache is deprecated. Learn more.

▲ Uses HTTPS — Error!

All sites should be protected with HTTPS, even ones that don't handle sensitive data. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users, and is a prerequisite for HTTP/2 and many new web platform APIs. Learn more.

▲ Uses HTTP/2 for its own resources — Error!

HTTP/2 offers many benefits over HTTP/1.1, including binary headers, multiplexing, and server push. Learn more.

▲ Uses passive listeners to improve scrolling performance — Error!

Consider marking your touch and wheel event listeners as `passive` to improve your page's scroll performance. Learn more.

▲ Avoids `document.write()` — Error!

For users on slow connections, external scripts dynamically injected via `document.write()` can delay page load by tens of seconds. Learn more.

▲ Links to cross-origin destinations are safe — Error!

Add `rel="noopener"` or `rel="noreferrer"` to any external links to improve performance and prevent security vulnerabilities. Learn more.

▲ Avoids requesting the geolocation permission on page load — Error!

Users are mistrustful of or confused by sites that request their location without context. Consider tying the request to a user action instead. Learn more.

▲ Page has the HTML doctype — Error!

Specifying a doctype prevents the browser from switching to quirks-mode. Learn more.

▲ Avoids requesting the notification permission on page load — Error!

Users are mistrustful of or confused by sites that request to send notifications without context. Consider tying the request to user gestures instead. Learn more.

▲ Avoids deprecated APIs — Error!

Deprecated APIs will eventually be removed from the browser. Learn more.

▲ Allows users to paste into password fields — Error!

Preventing password pasting undermines good security policy. Learn more.

▲ No browser errors logged to the console — Error!

Errors logged to the console indicate unresolved problems. They can come from network request failures and other browser concerns. Learn more

▲ Displays images with correct aspect ratio — Error!

Image display dimensions should match natural aspect ratio. Learn more.

Passed audits (2)

Avoids front-end JavaScript libraries with known security vulnerabilities

Some third-party scripts may contain known security vulnerabilities that are easily identified and exploited by attackers. Learn more.

Detected JavaScript libraries

All front-end JavaScript libraries detected on the page. Learn more.

?

SEO

These checks ensure that your page is optimized for search engine results ranking. There are additional factors Lighthouse does not check that may affect your search ranking. Learn more.

Mobile Friendly — Make sure your pages are mobile friendly so users don’t have to pinch or zoom in order to read the content pages. Learn more.

▲ Has a `<meta name="viewport">` tag with `width` or `initial-scale` — Error!

Add a `<meta name="viewport">` tag to optimize your app for mobile screens. Learn more.

▲ Document uses legible font sizes — Error!

Font sizes less than 12px are too small to be legible and require mobile visitors to “pinch to zoom” in order to read. Strive to have >60% of page text ≥12px. Learn more.

▲ Tap targets are sized appropriately — Error!

Interactive elements like buttons and links should be large enough (48x48px), and have enough space around them, to be easy enough to tap without overlapping onto other elements. Learn more.

Content Best Practices — Format your HTML in a way that enables crawlers to better understand your app’s content.

▲ Document has a `<title>` element — Error!

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more.

▲ Document has a meta description — Error!

Meta descriptions may be included in search results to concisely summarize page content. Learn more.

▲ Links have descriptive text — Error!

Descriptive link text helps search engines understand your content. Learn more.

▲ Image elements have `[alt]` attributes — Error!

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more.

▲ Document has a valid `hreflang` — Error!

hreflang links tell search engines what version of a page they should list in search results for a given language or region. Learn more.

▲ Document has a valid `rel=canonical` — Error!

Canonical links suggest which URL to show in search results. Learn more.

▲ Document avoids plugins — Error!

Search engines can't index plugin content, and many devices restrict plugins or don't support them. Learn more.

Crawling and Indexing — To appear in search results, crawlers need access to your app.

▲ Page has successful HTTP status code — Error!

Pages with unsuccessful HTTP status codes may not be indexed properly. Learn more.

▲ Page isn’t blocked from indexing — Error!

Search engines are unable to include your pages in search results if they don't have permission to crawl them. Learn more.

▲ robots.txt is valid — Error!

If your robots.txt file is malformed, crawlers may not be able to understand how you want your website to be crawled or indexed. Learn more.

Additional items to manually check (1) — Run these additional validators on your site to check additional SEO best practices.

Structured data is valid

Run the Structured Data Testing Tool and the Structured Data Linter to validate structured data. Learn more.

?

Progressive Web App

These checks validate the aspects of a Progressive Web App. Learn more.

Fast and reliable

▲ Page load is fast enough on mobile networks — Error!

A fast page load over a cellular network ensures a good mobile user experience. Learn more.

▲ Current page responds with a 200 when offline — Error!

If you're building a Progressive Web App, consider using a service worker so that your app can work offline. Learn more.

▲ `start_url` responds with a 200 when offline — Error!

A service worker enables your web app to be reliable in unpredictable network conditions. Learn more.

Installable

▲ Uses HTTPS — Error!

All sites should be protected with HTTPS, even ones that don't handle sensitive data. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users, and is a prerequisite for HTTP/2 and many new web platform APIs. Learn more.

▲ Registers a service worker that controls page and `start_url` — Error!

The service worker is the technology that enables your app to use many Progressive Web App features, such as offline, add to homescreen, and push notifications. Learn more.

▲ Web app manifest does not meet the installability requirements Failures: No manifest was fetched.

Browsers can proactively prompt users to add your app to their homescreen, which can lead to higher engagement. Learn more.

PWA Optimized

▲ Redirects HTTP traffic to HTTPS — Error!

If you've already set up HTTPS, make sure that you redirect all HTTP traffic to HTTPS in order to enable secure web features for all your users. Learn more.

▲ Is not configured for a custom splash screen Failures: No manifest was fetched.

A themed splash screen ensures a high-quality experience when users launch your app from their homescreens. Learn more.

▲ Sets a theme color for the address bar. — Error!

The browser address bar can be themed to match your site. Learn more.

▲ Content is sized correctly for the viewport — Error!

If the width of your app's content doesn't match the width of the viewport, your app might not be optimized for mobile screens. Learn more.

▲ Has a `<meta name="viewport">` tag with `width` or `initial-scale` — Error!

Add a `<meta name="viewport">` tag to optimize your app for mobile screens. Learn more.

▲ Contains some content when JavaScript is not available — Error!

Your app should display some content when JavaScript is disabled, even if it's just a warning to the user that JavaScript is required to use the app. Learn more.

▲ Provides a valid `apple-touch-icon` — Error!

For ideal appearance on iOS when users add a progressive web app to the home screen, define an `apple-touch-icon`. It must point to a non-transparent 192px (or 180px) square PNG. Learn More.

Additional items to manually check (3) — These checks are required by the baseline PWA Checklist but are not automatically checked by Lighthouse. They do not affect your score but it's important that you verify them manually.

Site works cross-browser

To reach the most number of users, sites should work across every major browser. Learn more.

Page transitions don't feel like they block on the network

Transitions should feel snappy as you tap around, even on a slow network. This experience is key to a user's perception of performance. Learn more.

Each page has a URL

Ensure individual pages are deep linkable via URL and that URLs are unique for the purpose of shareability on social media. Learn more.

Runtime Settings

URL	chrome-extension://iaapelhigghggmbghgcmnjeegdblejdp/_generated_background_page.html
Fetch time	Apr 13, 2020, 3:09 PM MDT
Device	Emulated Desktop
Network throttling	150 ms TCP RTT, 1,638.4 Kbps throughput (Simulated)
CPU throttling	4x slowdown (Simulated)
User agent (host)	Mozilla/5.0 (X11; CrOS x86_64 12739.111.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.162 Safari/537.36
CPU/Memory Power	1278

Generated by **Lighthouse** 5.7.0 | File an issue