Results for: https://sidharthrai.github.io/Calculator-PWA/
Sep 1, 2017, 5:01 PM UTC • Runtime settings

Print...
Save as JSON
Save as HTML
Save as JSON
Open in Viewer

64	98	91	92
Progressive Web App	Performance	Accessibility	Best Practices

Progressive Web App

64

These audits validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist.

4 failed audits

- **Does not provide fallback content when JavaScript is not available** ✗
  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.
  The page body should render some content if its scripts are not available.
- **User will not be prompted to Install the Web App** ✗
  Browsers can proactively prompt users to add your app to their homescreen, which can lead to higher engagement. Learn more.
  Failures: Manifest failed to parse as valid JSON, Manifest start_url is not cached by a Service Worker.
- **Is not configured for a custom splash screen** ✗
  A default splash screen will be constructed for your app, but satisfying these requirements guarantee a high-quality splash screen that transitions the user from tapping the home screen icon to your app's first paint
  Failures: Manifest failed to parse as valid JSON.
- **Address bar does not match brand colors** ✗
  The browser address bar can be themed to match your site. Learn more.
  Failures: Manifest failed to parse as valid JSON.

7 Passed Audits

- **Registers a Service Worker** ✓
  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.
- **Responds with a 200 when offline** ✓
  If you're building a Progressive Web App, consider using a service worker so that your app can work offline. Learn more.
- **Uses HTTPS** ✓
  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.

- Redirects HTTP traffic to HTTPS ✓
  If you've already set up HTTPS, make sure that you redirect all HTTP traffic to HTTPS. Learn more.
- Page load is fast enough on 3G ✓
  A fast page load over a 3G network ensures a good mobile user experience. Learn more.
- Has a `<meta name="viewport">` tag with `width` or `initial-scale` ✓
  Add a viewport meta tag to optimize your app for mobile screens. Learn more.
- Content is sized correctly for the viewport ✓
  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.

Manual checks to verify

These audits 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, a key to perceived performance. Learn more.
- Each page has a URL
  Ensure individual pages are deep linkable via the URLs and that URLs are unique for the purpose of shareability on social media. Learn more.

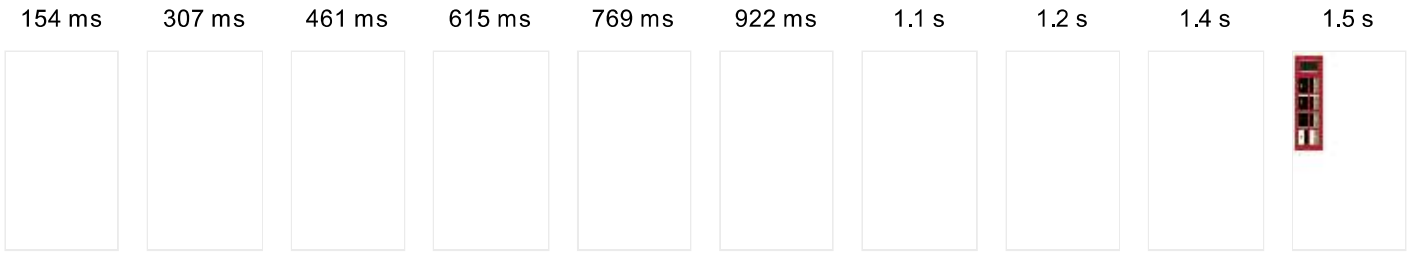
Performance

These encapsulate your app's performance.

98

Metrics

These metrics encapsulate your app's performance across a number of dimensions.



- First meaningful paint 1,530 ms
  First meaningful paint measures when the primary content of a page is visible. Learn more.
- First Interactive (beta) 1,530 ms
  The first point at which necessary scripts of the page have loaded and the CPU is idle enough to handle most user input.

Consistently Interactive (beta) 1,530 ms
The point at which most network resources have finished loading and the CPU is idle for a prolonged period.

Perceptual Speed Index: 1,537 (target: < 1,250) 97
Speed Index shows how quickly the contents of a page are visibly populated. Learn more.

Estimated Input Latency: 16 ms (target: < 50 ms) 100
The score above is an estimate of how long your app takes to respond to user input, in milliseconds. There is a 90% probability that a user encounters this amount of latency, or less. 10% of the time a user can expect additional latency. If your score is higher than Lighthouse's target score, users may perceive your app as laggy. Learn more.

Opportunities
These are opportunities to speed up your application by optimizing the following resources.

Reduce render-blocking stylesheets 690 ms

Link elements are blocking the first paint of your page. Consider inlining critical links and deferring non-critical ones. Learn more.

View Details

URL	Size (KB)	Delayed Paint By (ms)
…style/style.css	0.53 KB	693 ms

Diagnostics
More information about the performance of your application.

Critical Request Chains: 2
The Critical Request Chains below show you what resources are required for first render of this page. Improve page load by reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources. Learn more.
Longest chain: **1,350.6ms** over **2** requests, totalling **0.66 KB**

View critical network waterfall:

Initial Navigation
- /Calculator-PWA/ (sidharthrai.github.io)
  - …style/style.css (sidharthrai.github.io) **- 692.6ms, 0.53 KB**
  - …script/app.js (sidharthrai.github.io) **- 757.1ms, 0.66 KB**

9 Passed Audits

Reduce render-blocking scripts
Script elements are blocking the first paint of your page. Consider inlining critical scripts and deferring non-critical ones. Learn more.

Properly size images
Serve images that are appropriately-sized to save cellular data and improve load time. Learn more.

▼ Offscreen images
Consider lazy-loading offscreen images to improve page load speed and time to interactive. Learn more.

▼ Optimize images
Optimized images load faster and consume less cellular data. Learn more.

▼ Serve images as WebP
WebP provides better lossy and lossless compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more.

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

▼ Avoids enormous network payloads: Total size was 2 KB (target: < 1,600 KB) **100**
Network transfer size costs users real money and is highly correlated with long load times. Try to find ways to reduce the size of required files.

▼ View Details

URL	Total Size	Transfer Time
…script/app.js	1 KB	0 ms
/Calculator-PWA/	1 KB	0 ms
…style/style.css	1 KB	0 ms

▼ Avoids an excessive DOM size: 37 nodes (target: < 1,500 nodes) **100**
Browser engineers recommend pages contain fewer than ~1,500 DOM nodes. The sweet spot is a tree depth < 32 elements and fewer than 60 children/parent element. A large DOM can increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn more.

▼ View details

Total DOM Nodes	DOM Depth	Maximum Children
37 target: < 1,500 nodes	5 target: < 32	25 target: < 60 nodes

▼ User Timing marks and measures: 0
Consider instrumenting your app with the User Timing API to create custom, real-world measurements of key user experiences. Learn more.

Accessibility

These checks highlight opportunities to improve the accessibility of your app.

91

Elements Describe Contents Well

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- Form elements do not have associated labels. ✕
  Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more.
  - View failing elements

```
<input class="display" type="text" readonly="" size="18" id="d">
```

Color Contrast Is Satisfactory

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- Background and foreground colors do not have a sufficient contrast ratio. ✕
  Low-contrast text is difficult or impossible for many users to read. Learn more.
  - View failing elements

```
<input type="button" class="button_function" value="+"
onclick="get(&amp;quot;+&amp;quot;)">
<input type="button" class="button_function" value="-"
onclick="get(&amp;quot;-&amp;quot;)">
<input type="button" class="button_function" value="X"
onclick="get(&amp;quot;*&amp;quot;)">
<input type="button" class="button_function" value="/"
onclick="get(&amp;quot;/&amp;quot;)">
```

Page Specifies Valid Language

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- `<html>` element does not have a `[lang]` attribute. ✕
  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.
  - View failing elements

```
<html class="gr__sidharthrai_github_io">
```

7 Passed Audits

Elements Use Attributes Correctly

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- [accesskey] values are unique. ✓
  Access keys let users quickly focus a part of the page. For proper navigation, each access key must be unique. Learn more.

- <audio> elements contain a <track> element with [kind="captions"]. ✓
  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.

- Image elements have [alt] attributes. ✓
  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. ✓
  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.

- No element has a [tabindex] value greater than 0. ✓
  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.

- Cells in a <table> element that use the [headers] attribute only refer to other cells of that same table. ✓
  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. ✓
  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.

ARIA Attributes Follow Best Practices

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- [aria-*] attributes match their roles. ✓
  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. ✓
  Some ARIA roles have required attributes that describe the state of the element to screen readers. Learn more.

- Elements with [role] that require specific children [role]s, are present. ✓
  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. ✓
  Some ARIA child roles must be contained by specific parent roles to properly perform their intended accessibility functions. Learn more.

- `[role]` values are valid. ✓
  ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more.
- `[aria-*]` attributes have valid values. ✓
  Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. Learn more.
- `[aria-*]` attributes are valid and not misspelled. ✓
  Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more.

Elements Have Discernable Names

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- Buttons have an accessible name. ✓
  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.
- Links have a discernible name. ✓
  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.

Elements Describe Contents Well

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- The page contains a heading, skip link, or landmark region. ✓
  Adding ways to bypass repetitive content lets keyboard users navigate the page more efficiently. Learn more.
- Document has a `<title>` element. ✓
  Screen reader users use page titles to get an overview of the contents of the page. Learn more.
- `<frame>` or `<iframe>` elements have a title. ✓
  Screen reader users rely on frame titles to describe the contents of frames. Learn more.
- Presentational `<table>` elements avoid using `<th>`, `<caption>` or the `[summary]` attribute. ✓
  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.
- `<object>` elements have `[alt]` text. ✓
  Screen readers cannot translate non-text content. Adding alt text to `<object>` elements helps screen readers convey meaning to users. Learn more.
- `<video>` elements contain a `<track>` element with `[kind="captions"]`. ✓
  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"]`.
Audio descriptions provide relevant information for videos that dialogue cannot, such as facial expressions and scenes. Learn more.

Elements Are Well Structured

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>` or `<template>` elements.
  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.
  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.
- `[id]` attributes on the page are unique.
  The value of an id attribute must be unique to prevent other instances from being overlooked by assistive technologies. Learn more.
- Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).
  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.
  Screen readers require list items (`<li>`) to be contained within a parent `<ul>` or `<ol>` to be announced properly. Learn more.

Page Specifies Valid Language

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- `<html>` element has a valid value for its `[lang]` attribute.
  Specifying a valid BCP 47 language helps screen readers announce text properly. Learn more.
- `[lang]` attributes have a valid value.
  Specifying a valid BCP 47 language on elements helps ensure that text is pronounced correctly by a screen reader. Learn more.

Meta Tags Used Properly

Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.

- The document does not use `<meta http-equiv="refresh">`.
  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.

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.

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.

92

1 failed audits

- Manifest's `short_name` will be truncated when displayed on homescreen ✗
  Make your app's `short_name` fewer than 12 characters to ensure that it's not truncated on homescreens. Learn more.

12 Passed Audits

- Avoids Application Cache ✓
  Application Cache is deprecated. Learn more.
- Avoids WebSQL DB ✓
  Web SQL is deprecated. Consider using IndexedDB instead. Learn more.
- Uses HTTPS ✓
  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 ✓
  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 ✓
  Consider marking your touch and wheel event listeners as `passive` to improve your page's scroll performance. Learn more.
- Avoids Mutation Events in its own scripts ✓
  Mutation Events are deprecated and harm performance. Consider using Mutation Observers instead. Learn more.
- Avoids `document.write()` ✓
  For users on slow connections, external scripts dynamically injected via `document.write()` can delay page load by tens of seconds. Learn more.
- Opens external anchors using `rel="noopener"` ✓
  Open new tabs using `rel="noopener"` to improve performance and prevent security vulnerabilities. Learn more.
- Avoids requesting the geolocation permission on page load ✓
  Users are mistrustful of or confused by sites that request their location without context. Consider tying the request to user gestures instead. Learn more.

Avoids requesting the notification permission on page load ✓
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 ✓
Deprecated APIs will eventually be removed from the browser. Learn more.

Allows users to paste into password fields ✓

Generated by **Lighthouse** 2.3.0 on Sep 1, 2017, 5:01 PM UTC | File an issue