http://127.0.0.1:5500/

98 100 100

98
Performance

100
Accessibility

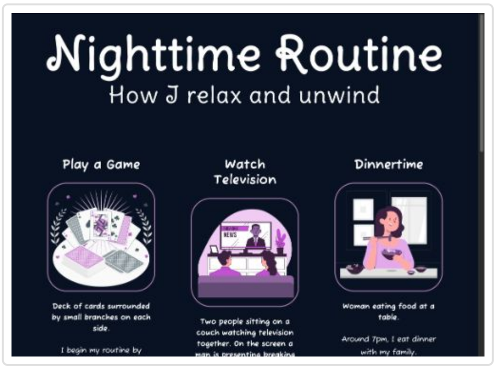
100
Best Practices

98

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100


Nighttime Routine

How J relax and unwind

Play a Game

Deck of cards surrounded by small branches on each side.

I begin my routine by

Watch Television

Two people sitting on a couch watching television together. On the screen a man is presenting breaking

Dinnertime

Woman eating food at a table.

Around 7pm, I eat dinner with my family.

METRICS Expand view

First Contentful Paint	Largest Contentful Paint

0.4 s

1.1 s

Total Blocking Time

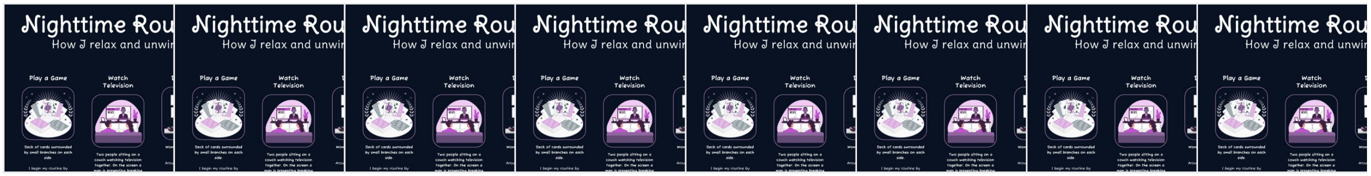
0 ms

Cumulative Layout Shift

0

Speed Index

0.4 s



Later this year, insights will replace performance audits. Learn more and provide feedback here.

Go back to audits

Show audits relevant to: All FCP LCP

INSIGHTS

▲ Network dependency tree

Avoid chaining critical requests by reducing the length of chains, reducing the download size of resources, or deferring the download of unnecessary resources to improve page load. LCP

Maximum critical path latency: **29 ms**

Initial Navigation

http://127.0.0.1:5500 **- 19 ms,** 233.97 KiB

/style.css (127.0.0.1) **- 29 ms,** 11.54 KiB

Preconnected origins

preconnect hints help the browser establish a connection earlier in the page load, saving time when the first request for that origin is made. The following are the origins that the page preconnected to.

no origins were preconnected

Preconnect candidates

Add preconnect hints to your most important origins, but try to use no more than 4.

No additional origins are good candidates for preconnecting

Document request latency — Est savings of 157 KiB

Your first network request is the most important. Reduce its latency by avoiding redirects, ensuring a fast server response, and enabling text compression. FCP LCP

Avoids redirects

Server responds quickly (observed 5 ms)

No compression applied

Render blocking requests

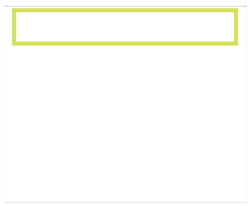
Requests are blocking the page's initial render, which may delay LCP. Deferring or inlining can move these network requests out of the critical path. FCP LCP

URL	Transfer Size	Duration
127.0.0.1 1st Party	**11.5 KiB**	**0 ms**
/style.css (127.0.0.1)	11.5 KiB	

LCP breakdown

Each subpart has specific improvement strategies. Ideally, most of the LCP time should be spent on loading the resources, not within delays. LCP

Subpart	Duration
Time to first byte	10 ms
Element render delay	80 ms



h1

These insights are also available in the Chrome DevTools Performance Panel - record a trace to view more detailed information.

DIAGNOSTICS

▲ Page prevented back/forward cache restoration — 1 failure reason

Many navigations are performed by going back to a previous page, or forwards again. The back/forward cache (bfcache) can speed up these return navigations. Learn more about the bfcache

Failure reason	Failure type
Pages with WebSocket cannot enter back/forward cache.	Pending browser support
http://127.0.0.1:5500	

Minify CSS — Est savings of 6 KiB

Minifying CSS files can reduce network payload sizes. Learn how to minify CSS. FCP LCP

URL	Transfer Size	Est Savings
127.0.0.1 1st Party	**11.5 KiB**	**6.2 KiB**
/style.css (127.0.0.1)	11.5 KiB	6.2 KiB

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

PASSED AUDITS (27)

Hide

Use efficient cache lifetimes

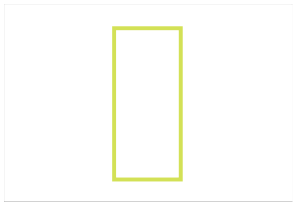
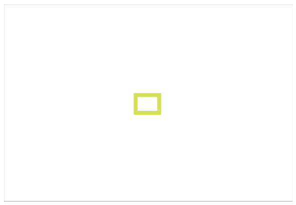
A long cache lifetime can speed up repeat visits to your page. Learn more. FCP LCP

Layout shift culprits

Layout shifts occur when elements move absent any user interaction. Investigate the causes of layout shifts, such as elements being added, removed, or their fonts changing as the page loads. CLS

Optimize DOM size

A large DOM can increase the duration of style calculations and layout reflows, impacting page responsiveness. A large DOM will also increase memory usage. Learn how to avoid an excessive DOM size.

Statistic	Element		Value
Total elements			918
Most children		g#freepik--Hanger--inject-35	114
DOM depth		path	14

Duplicated JavaScript

Remove large, duplicate JavaScript modules from bundles to reduce unnecessary bytes consumed by network activity. FCP LCP

Font display

Consider setting `font-display` to `swap` or `optional` to ensure text is consistently visible. `swap` can be further optimized to mitigate layout shifts with font metric overrides.

Forced reflow

A forced reflow occurs when JavaScript queries geometric properties (such as `offsetWidth`) after styles have been invalidated by a change to the DOM state. This can result in poor performance. Learn more about forced reflows and possible mitigations.

Improve image delivery

Reducing the download time of images can improve the perceived load time of the page and LCP. Learn more about optimizing image size FCP LCP

○ INP breakdown

Start investigating with the longest subpart. Delays can be minimized. To reduce processing duration, optimize the main-thread costs, often JS.

○ LCP request discovery

Optimize LCP by making the LCP image discoverable from the HTML immediately, and avoiding lazy-loading

Legacy JavaScript

Polyfills and transforms enable older browsers to use new JavaScript features. However, many aren't necessary for modern browsers. Consider modifying your JavaScript build process to not transpile Baseline features, unless you know you must support older browsers. Learn why most sites can deploy ES6+ code without transpiling FCP LCP

Modern HTTP

HTTP/2 and HTTP/3 offer many benefits over HTTP/1.1, such as multiplexing. Learn more about using modern HTTP. FCP LCP

3rd parties

3rd party code can significantly impact load performance. Reduce and defer loading of 3rd party code to prioritize your page's content.

Optimize viewport for mobile

Tap interactions may be delayed by up to 300 ms if the viewport is not optimized for mobile.

meta

Defer offscreen images

Consider lazy-loading offscreen and hidden images after all critical resources have finished loading to lower time to interactive. Learn how to defer offscreen images. FCP LCP

Minify JavaScript

Minifying JavaScript files can reduce payload sizes and script parse time. Learn how to minify JavaScript. FCP LCP

Reduce unused CSS

Reduce unused rules from stylesheets and defer CSS not used for above-the-fold content to decrease bytes consumed by network activity. Learn how to reduce unused CSS. FCP LCP

Reduce unused JavaScript

Reduce unused JavaScript and defer loading scripts until they are required to decrease bytes consumed by network activity. Learn how to reduce unused JavaScript. FCP LCP

Use HTTP/2

HTTP/2 offers many benefits over HTTP/1.1, including binary headers and multiplexing. Learn more about HTTP/2. LCP FCP

Avoid serving legacy JavaScript to modern browsers

Polyfills and transforms enable legacy browsers to use new JavaScript features. However, many aren't necessary for modern browsers. Consider modifying your JavaScript build process to not transpile Baseline features, unless you know you must support legacy browsers. Learn why most sites can deploy ES6+ code without transpiling FCP LCP

Avoids enormous network payloads — Total size was 932 KiB

Large network payloads cost users real money and are highly correlated with long load times. Learn how to reduce payload sizes.

URL	Transfer Size
127.0.0.1 1st Party	**932.0 KiB**
/Card%20game-amico-hsl.svg (127.0.0.1)	564.1 KiB
http://127.0.0.1:5500	234.0 KiB
/fonts/FuzzyBubb….woff2 (127.0.0.1)	50.7 KiB
/fonts/FuzzyBubbles-Bold.woff2 (127.0.0.1)	49.1 KiB
/fonts/DeliusSwa….woff2 (127.0.0.1)	22.6 KiB
/style.css (127.0.0.1)	11.5 KiB

○ User Timing marks and measures

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

○ JavaScript execution time

Consider reducing the time spent parsing, compiling, and executing JS. You may find delivering smaller JS payloads helps with this. Learn how to reduce Javascript execution time. TBT

Minimizes main-thread work — 0.1 s

Consider reducing the time spent parsing, compiling and executing JS. You may find delivering smaller JS payloads helps with this. Learn how to minimize main-thread work TBT

Category	Time Spent
Other	30 ms
Style & Layout	14 ms
Script Evaluation	6 ms
Parse HTML & CSS	3 ms
Rendering	2 ms
Script Parsing & Compilation	0 ms

○ Lazy load third-party resources with facades

Some third-party embeds can be lazy loaded. Consider replacing them with a facade until they are required. Learn how to defer third-parties with a facade. TBT

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 about adopting passive event listeners.

Avoids `document.write()`

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

○ Avoid long main-thread tasks

Lists the longest tasks on the main thread, useful for identifying worst contributors to input delay. Learn how to avoid long main-thread tasks TBT

100

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

○ Interactive controls are keyboard focusable

Custom interactive controls are keyboard focusable and display a focus indicator. Learn how to make custom controls focusable.

○ 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 how to decorate interactive elements with affordance hints.

○ The page has a logical tab order

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

○ Visual order on the page follows DOM order

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

○ 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 how to avoid focus traps.

○ 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 how to direct focus to new content.

○ 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 about landmark elements.

○ Offscreen content is hidden from assistive technology

Offscreen content is hidden with display: none or aria-hidden=true. Learn how to properly hide offscreen content.

○ Custom controls have associated labels

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

○ Custom controls have ARIA roles

Custom interactive controls have appropriate ARIA roles. Learn how to add roles to custom controls.

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

PASSED AUDITS (11) Hide

`[aria-hidden="true"]` is not present on the document `<body>`

Assistive technologies, like screen readers, work inconsistently when `aria-hidden="true"` is set on the document <body>. Learn how `aria-hidden` affects the document body.

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 about the `alt` attribute.

`[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 about the viewport meta tag.

Background and foreground colors have a sufficient contrast ratio

Low-contrast text is difficult or impossible for many users to read. Learn how to provide sufficient color contrast.

Document has a `<title>` element

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 about document titles.

`<html>` element has 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 about the `lang` attribute.

`<html>` element has a valid value for its `[lang]` attribute

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn how to use the `lang` attribute.

Links are distinguishable without relying on color.

Low-contrast text is difficult or impossible for many users to read. Link text that is discernible improves the experience for users with low vision. Learn how to make links distinguishable.

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 how to make links accessible.

Heading elements appear in a sequentially-descending order

Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies. Learn more about heading order.

Image elements do not have `[alt]` attributes that are redundant text.

Informative elements should aim for short, descriptive alternative text. Alternative text that is exactly the same as the text adjacent to the link or image is potentially confusing for screen reader users, because the text will be read twice. Learn more about the `alt` attribute.

NOT APPLICABLE (46) Hide

`[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 about access keys.

`[aria-*]` attributes match their roles

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

attributes. Learn how to match ARIA attributes to their roles.

○ Uses ARIA roles only on compatible elements

Many HTML elements can only be assigned certain ARIA roles. Using ARIA roles where they are not allowed can interfere with the accessibility of the web page. Learn more about ARIA roles.

○ `button`, `link`, and `menuitem` elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to make command elements more accessible.

○ ARIA attributes are used as specified for the element's role

Some ARIA attributes are only allowed on an element under certain conditions. Learn more about conditional ARIA attributes.

○ Deprecated ARIA roles were not used

Deprecated ARIA roles may not be processed correctly by assistive technology. Learn more about deprecated ARIA roles.

○ Elements with `role="dialog"` or `role="alertdialog"` have accessible names.

ARIA dialog elements without accessible names may prevent screen readers users from discerning the purpose of these elements. Learn how to make ARIA dialog elements more accessible.

○ `[aria-hidden="true"]` elements do not contain focusable descendents

Focusable descendents within an `[aria-hidden="true"]` element prevent those interactive elements from being available to users of assistive technologies like screen readers. Learn how `aria-hidden` affects focusable elements.

○ ARIA input fields have accessible names

When an input field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about input field labels.

○ ARIA `meter` elements have accessible names

When a meter element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to name `meter` elements.

○ ARIA `progressbar` elements have accessible names

When a `progressbar` element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to label `progressbar` elements.

○ Elements use only permitted ARIA attributes

Using ARIA attributes in roles where they are prohibited can mean that important information is not communicated to users of assistive technologies. Learn more about prohibited ARIA roles.

○ `[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 about roles and required attributes.

○ Elements with an ARIA `[role]` that require children to contain a specific `[role]` have all required children.

Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. Learn more about roles and required children elements.

○ `[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 about ARIA roles and required parent element.

○ `[role]` values are valid

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

○ Elements with the `role=text` attribute do not have focusable descendents.

Adding `role=text` around a text node split by markup enables VoiceOver to treat it as one phrase, but the element's focusable descendents will not be announced. Learn more about the `role=text` attribute.

○ ARIA toggle fields have accessible names

When a toggle field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about toggle fields.

○ ARIA `tooltip` elements have accessible names

When a tooltip element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to name `tooltip` elements.

○ ARIA `treeitem` elements have accessible names

When a `treeitem` element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about labeling `treeitem` elements.

○ `[aria-*]` attributes have valid values

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

○ `[aria-*]` attributes are valid and not misspelled

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

○ 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 how to make buttons more accessible.

○ 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 about bypass blocks.

○ `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>`, `<template>` or `<div>` elements.

When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn how to structure definition lists correctly.

○ 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 how to structure definition lists correctly.

○ ARIA IDs are unique

The value of an ARIA ID must be unique to prevent other instances from being overlooked by assistive technologies. Learn how to fix duplicate ARIA IDs.

○ No form fields have multiple labels

Form fields with multiple labels can be confusingly announced by assistive technologies like screen readers which use either the first, the last, or all of the labels. Learn how to use form labels.

○ `<frame>` or `<iframe>` elements have a title

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

○ `<html>` element has an `[xml:lang]` attribute with the same base language as the `[lang]` attribute.

If the webpage does not specify a consistent language, then the screen reader might not announce the page's text correctly. Learn more about the `lang` attribute.

○ Input buttons have discernible text.

Adding discernable and accessible text to input buttons may help screen reader users understand the purpose of the input button. Learn more about input buttons.

○ `<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 about input image alt text.

○ Form elements have associated labels

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

○ 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 about proper list structure.

○ List items (`<li>`) are contained within `<ul>`, `<ol>` or `<menu>` parent elements

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

○ 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 about the refresh meta tag.

○ `<object>` elements have alternate text

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

○ Select elements have associated label elements.

Form elements without effective labels can create frustrating experiences for screen reader users. Learn more about the `select` element.

○ Skip links are focusable.

Including a skip link can help users skip to the main content to save time. Learn more about skip links.

○ 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 about the `tabindex` attribute.

○ Tables have different content in the summary attribute and `<caption>`.

The summary attribute should describe the table structure, while `<caption>` should have the onscreen title. Accurate table mark-up helps users of screen readers. Learn more about summary and caption.

○ Touch targets have sufficient size and spacing.

Touch targets with sufficient size and spacing help users who may have difficulty targeting small controls to activate the targets. Learn more about touch targets.

○ Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the 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 about the `headers` attribute.

○ `<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 about table headers.

○ `[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 how to use the `lang` attribute.

○ `<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 about video captions.

100

Best Practices

TRUST AND SAFETY

○ Ensure CSP is effective against XSS attacks

A strong Content Security Policy (CSP) significantly reduces the risk of cross-site scripting (XSS) attacks. Learn how to use a CSP to prevent XSS

Description	Directive	Severity
No CSP found in enforcement mode		High

○ Use a strong HSTS policy

Deployment of the HSTS header significantly reduces the risk of downgrading HTTP connections and eavesdropping attacks. A rollout in stages, starting with a low max-age is recommended. Learn more about using a strong HSTS policy.

Description	Directive	Severity
No HSTS header found		High

○ Ensure proper origin isolation with COOP

The Cross-Origin-Opener-Policy (COOP) can be used to isolate the top-level window from other documents such as pop-ups. Learn more about deploying the COOP header.

Description	Directive	Severity
No COOP header found		High

○ Mitigate clickjacking with XFO or CSP

The `X-Frame-Options` (XFO) header or the `frame-ancestors` directive in the `Content-Security-Policy` (CSP) header control where a page can be embedded. These can mitigate clickjacking attacks by blocking some or all sites from embedding the page. Learn more about mitigating clickjacking.

Description	Severity
No frame control policy found	High

○ Mitigate DOM-based XSS with Trusted Types

The `require-trusted-types-for` directive in the `Content-Security-Policy` (CSP) header instructs user agents to control the data passed to DOM XSS sink functions. Learn more about mitigating DOM-based XSS with Trusted Types.

Description	Severity
No `Content-Security-Policy` header with Trusted Types directive found	High

PASSED AUDITS (14) Hide

Uses HTTPS

All sites should be protected with HTTPS, even ones that don't handle sensitive data. This includes avoiding mixed content, where some resources are loaded over HTTP despite the initial request being served over HTTPS. 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 about HTTPS.

Avoids deprecated APIs

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

Avoids third-party cookies

Third-party cookies may be blocked in some contexts. Learn more about preparing for third-party cookie restrictions.

Allows users to paste into input fields

Preventing input pasting is a bad practice for the UX, and weakens security by blocking password managers.Learn more about user-friendly input fields.

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 a user action instead. Learn more about the geolocation permission.

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 about responsibly getting permission for notifications.

Displays images with correct aspect ratio

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

Serves images with appropriate resolution

Image natural dimensions should be proportional to the display size and the pixel ratio to maximize image clarity. Learn how to provide responsive images.

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

A `<meta name="viewport">` not only optimizes your app for mobile screen sizes, but also prevents a 300 millisecond delay to user input. Learn more about using the viewport meta tag.

Page has the HTML doctype

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

Properly defines charset

A character encoding declaration is required. It can be done with a `<meta>` tag in the first 1024 bytes of the HTML or in the Content-Type HTTP response header. Learn more about declaring the character encoding.

No browser errors logged to the console

Errors logged to the console indicate unresolved problems. They can come from network request failures and other browser concerns. Learn more about this errors in console diagnostic audit

No issues in the `Issues` panel in Chrome Devtools

Issues logged to the `Issues` panel in Chrome Devtools indicate unresolved problems. They can come from network request failures, insufficient security controls, and other browser concerns. Open up the Issues panel in Chrome DevTools for more details on each issue.

Page has valid source maps

Source maps translate minified code to the original source code. This helps developers debug in production. In addition, Lighthouse is able to provide further insights. Consider deploying source maps to take advantage of these benefits. Learn more about source maps.

NOT APPLICABLE (3)

Hide

○ Redirects HTTP traffic to HTTPS

Make sure that you redirect all HTTP traffic to HTTPS in order to enable secure web features for all your users. Learn more.

○ Document uses legible font sizes

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 about legible font sizes.

○ Detected JavaScript libraries

All front-end JavaScript libraries detected on the page. Learn more about this JavaScript library detection diagnostic audit.

Captured at Dec 5, 2025, 10:45 AM EST	Emulated Desktop with Lighthouse 12.8.2	Single page session
Initial page load	Custom throttling	Using Chromium 142.0.0.0 with devtools

Generated by **Lighthouse** 12.8.2 | File an issue