http://127.0.0.1:5501/WebTech/4thsemProject/htmlbasics/index.html

Performance	Accessibility	Best Practices	SEO	PWA	Publisher Ads
97	91	78	90	—	-

There were issues affecting this run of Lighthouse:

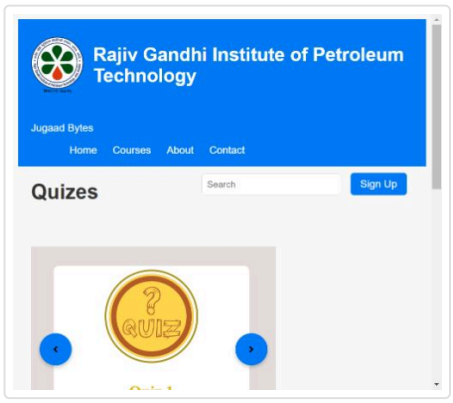
- **Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.**
- **No ads were requested when fetching this page.**
- **No ads were rendered when rendering this page.**
- **The GPT tag was not requested.**

97

Performance

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

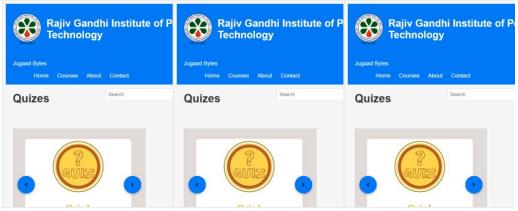
▲ 0–49 50–89 90–100



METRICS Expand view

First Contentful Paint 0.6 s	Largest Contentful Paint 0.7 s
Total Blocking Time 0 ms	Cumulative Layout Shift 0
Speed Index 1.6 s	

View Treemap



Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

▲ Properly size images — Potential savings of 135 KiB

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

URL	Resource Size	Potential Savings
127.0.0.1 1st Party	**139.7 KiB**	**135.4 KiB**
img …htmlbasics/rgipt_logopng.png (127.0.0.1)	139.7 KiB	135.4 KiB

▲ Serve images in next-gen formats — Potential savings of 89 KiB

Image formats like WebP and AVIF often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more about modern image formats.

URL	Resource Size	Potential Savings
127.0.0.1 1st Party	**139.7 KiB**	**88.6 KiB**
img …htmlbasics/rgipt_logopng.png (127.0.0.1)	139.7 KiB	88.6 KiB

▲ Page prevented back/forward cache restoration — 3 failure reasons

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. …htmlbasics/index.html (127.0.0.1) …htmlbasics/sample_005.html (127.0.0.1) …htmlbasics/sample_006.html (127.0.0.1)	Pending browser support

Image elements do not have explicit `width` and `height`

Set an explicit width and height on image elements to reduce layout shifts and improve CLS. Learn how to set image dimensions CLS

URL	
127.0.0.1 1st Party	
	img …htmlbasics/rgipt_logopng.png (127.0.0.1)

Minify CSS — Potential savings of 2 KiB

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

URL	Transfer Size	Potential Savings
`:root { font-size: 100%; } …`	24.1 KiB	2.4 KiB

Enable text compression — Potential savings of 14 KiB

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

URL	Transfer Size	Potential Savings
127.0.0.1 1st Party	**22.1 KiB**	**14.1 KiB**
…htmlbasics/sample_006.html (127.0.0.1)	3.5 KiB	2.2 KiB

URL	Transfer Size	Potential Savings
…htmlbasics/sample_005.html (127.0.0.1)	3.5 KiB	2.2 KiB
…htmlbasics/index.html (127.0.0.1)	3.5 KiB	2.0 KiB
…htmlbasics/styles.css (127.0.0.1)	2.5 KiB	1.7 KiB
…htmlbasics/sample_005.js (127.0.0.1)	2.3 KiB	1.5 KiB
…htmlbasics/sample_006.js (127.0.0.1)	2.3 KiB	1.5 KiB
…htmlbasics/sample_005.css (127.0.0.1)	2.3 KiB	1.4 KiB
…htmlbasics/sample_006.css (127.0.0.1)	2.3 KiB	1.4 KiB

Ensure text remains visible during webfont load

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

URL	Potential Savings
Loom – Screen Recorder & Screen Capture Chrome Extension	**40 ms**
`chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/fonts/CircularXXWeb-Book.woff2`	40 ms

Avoid serving legacy JavaScript to modern browsers — Potential savings of 8 KiB

Polyfills and transforms enable legacy browsers to use new JavaScript features. However, many aren't necessary for modern browsers. For your bundled JavaScript, adopt a modern script deployment strategy using module/nomodule feature detection to reduce the amount of code shipped to modern browsers, while retaining support for legacy browsers. Learn how to use modern JavaScript TBT

URL	Potential Savings
Loom – Screen Recorder & Screen Capture Chrome Extension	**8.4 KiB**
`chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/js/companionBubble.js`	8.4 KiB
companionBubble.js:4	`@babel/plugin-transform-classes`

URL	Potential Savings
companionBubble.js:4 Object.keys	

Reduce unused CSS — Potential savings of 42 KiB

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

URL	Transfer Size	Potential Savings
:root { font-size: 100%; } …	24.1 KiB	23.6 KiB
#inner-shadow-companion { font-size: 100%; } …	21.3 KiB	18.5 KiB

Reduce unused JavaScript — Potential savings of 776 KiB

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

Show 3rd-party resources (1)

URL	Transfer Size	Potential Savings
Loom – Screen Recorder & Screen Capture Chrome Extension	**988.1 KiB**	**573.1 KiB**
chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/js/companionBubble.js	988.1 KiB	573.1 KiB
Unattributable	**331.9 KiB**	**202.8 KiB**
chrome-extension://aapbdbdomjkkjkaonfhkkikfgjllcleb/bubble_compiled.js	46.9 KiB	37.1 KiB
chrome-extension://aapbdbdomjkkjkaonfhkkikfgjllcleb/bubble_compiled.js	46.9 KiB	37.1 KiB
chrome-extension://aapbdbdomjkkjkaonfhkkikfgjllcleb/bubble_compiled.js	46.9 KiB	37.1 KiB
chrome-extension://kbfnbcaeplbcioakkpcpgfkobkghlhen/src/js/Grammarly-check.js	63.7 KiB	30.7 KiB
chrome-extension://kbfnbcaeplbcioakkpcpgfkobkghlhen/src/js/Grammarly-check.js	63.7 KiB	30.7 KiB
chrome-extension://kbfnbcaeplbcioakkpcpgfkobkghlhen/src/js/Grammarly-check.js	63.7 KiB	30.2 KiB

User Timing marks and measures — 6 user timings

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.

Name	Type	Start Time	Duration
@grammarly-extension:checkScriptInitStart	Mark	340.27 ms	
@grammarly-extension:checkScriptInitEnd	Mark	368.31 ms	
@grammarly-extension:checkScriptInitStart	Mark	624.92 ms	
@grammarly-extension:checkScriptInitEnd	Mark	632.61 ms	
@grammarly-extension:checkScriptInitStart	Mark	655.78 ms	
@grammarly-extension:checkScriptInitEnd	Mark	662.84 ms	

Initial server response time was short — Root document took 0 ms

Keep the server response time for the main document short because all other requests depend on it. Learn more about the Time to First Byte metric. FCP LCP

URL	Time Spent
127.0.0.1 1st Party	**0 ms**
…htmlbasics/index.html (127.0.0.1)	0 ms

Avoids enormous network payloads — Total size was 271 KiB

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

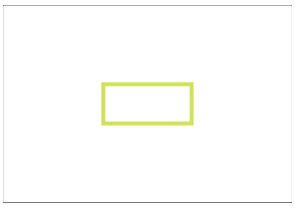
☑ Show 3rd-party resources (2)

URL	Transfer Size
127.0.0.1 1st Party	**173.8 KiB**
…htmlbasics/rgipt_logopng.png (127.0.0.1)	140.0 KiB
…htmlbasics/quiz.png (127.0.0.1)	11.1 KiB

URL	Transfer Size
…htmlbasics/gamepng.png (127.0.0.1)	5.7 KiB
…htmlbasics/sample_006.html (127.0.0.1)	3.8 KiB
…htmlbasics/sample_005.html (127.0.0.1)	3.8 KiB
…htmlbasics/index.html (127.0.0.1)	3.8 KiB
…htmlbasics/styles.css (127.0.0.1)	2.9 KiB
…htmlbasics/sample_005.js (127.0.0.1)	2.7 KiB
Cloudflare CDN Cdn	**87.4 KiB**
…webfonts/fa-solid-900.woff2 (cdnjs.cloudflare.com)	76.8 KiB
…css/all.min.css (cdnjs.cloudflare.com)	10.6 KiB

Avoids an excessive DOM size — 41 elements

A large DOM will increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn how to avoid an excessive DOM size. TBT

Statistic	Element	Value
Total DOM Elements		41
Maximum DOM Depth	 a	7
Maximum Child Elements	 body	6

Avoid chaining critical requests — 1 chain found

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 how to avoid chaining critical requests. FCP LCP

Maximum critical path latency: **23.882 ms**

Initial Navigation

…htmlbasics/index.html (127.0.0.1)

…htmlbasics/styles.css (127.0.0.1) **- 2.919 ms, 2.86 KiB**

JavaScript execution time — 0.4 s

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

Show 3rd-party resources (1)

URL	Total CPU Time	Script Evaluation	Script Parse
Loom – Screen Recorder & Screen Capture Chrome Extension	**264 ms**	**174 ms**	**78 ms**
`chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/js/companionBubble.js`	264 ms	174 ms	78 ms
Unattributable	**236 ms**	**83 ms**	**24 ms**
`chrome-extension://kbfnbcaeplbcioakkpcpgfkobkghlhen/src/js/Grammarly-check.js`	131 ms	81 ms	24 ms
`Unattributable`	105 ms	2 ms	0 ms
127.0.0.1 1st Party	**73 ms**	**5 ms**	**1 ms**
…htmlbasics/index.html (127.0.0.1)	73 ms	5 ms	1 ms

Minimizes main-thread work — 0.7 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
Script Evaluation	284 ms
Other	157 ms
Script Parsing & Compilation	128 ms

Category	Time Spent
Style & Layout	68 ms
Garbage Collection	15 ms
Parse HTML & CSS	13 ms
Rendering	6 ms

Minimize third-party usage — Third-party code blocked the main thread for 0 ms

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 how to minimize third-party impact. TBT

Third-Party	Transfer Size	Main-Thread Blocking Time
Cloudflare CDN Cdn	**87 KiB**	**0 ms**
…webfonts/fa-solid-900.woff2 (cdnjs.cloudflare.com)	77 KiB	0 ms
…css/all.min.css (cdnjs.cloudflare.com)	11 KiB	0 ms
Loom – Screen Recorder & Screen Capture Chrome Extension	**67 KiB**	**0 ms**
`chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/fonts/CircularXXWeb-Book.woff2`	67 KiB	0 ms

Largest Contentful Paint element — 740 ms

This is the largest contentful element painted within the viewport. Learn more about the Largest Contentful Paint element LCP

Element
 h1

Phase	% of LCP	Timing
TTFB	17%	120 ms

Phase	% of LCP	Timing
Load Delay	0%	0 ms
Load Time	0%	0 ms
Render Delay	83%	610 ms

Avoid long main-thread tasks — 1 long task found

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

URL	Start Time	Duration
Loom – Screen Recorder & Screen Capture Chrome Extension		**215 ms**
chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/js/companionBubble.js	275 ms	215 ms

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

PASSED AUDITS (19) Hide

Eliminate render-blocking resources

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn how to eliminate render-blocking resources. FCP LCP

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.

Minify JavaScript

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

Efficiently encode images

Optimized images load faster and consume less cellular data. Learn how to efficiently encode images.

Preconnect to required origins

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

Avoid multiple page redirects

Redirects introduce additional delays before the page can be loaded. Learn how to avoid page redirects. FCP LCP

Preload key requests

Consider using `<link rel=preload>` to prioritize fetching resources that are currently requested later in page load. Learn how to preload key requests. FCP LCP

Use HTTP/2

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

Use video formats for animated content

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 about efficient video formats LCP

Remove duplicate modules in JavaScript bundles

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

Preload Largest Contentful Paint image

If the LCP element is dynamically added to the page, you should preload the image in order to improve LCP. Learn more about preloading LCP elements. LCP

Uses efficient cache policy on static assets — 0 resources found

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

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

Largest Contentful Paint image was not lazily loaded

Above-the-fold images that are lazily loaded render later in the page lifecycle, which can delay the largest contentful paint. Learn more about optimal lazy loading. LCP

Avoid large layout shifts

These are the largest layout shifts observed on the page. Each table item represents a single layout shift, and shows the element that shifted the most. Below each item are possible root causes that led to the layout shift. Some of these layout shifts may not be included in the CLS metric value due to windowing. Learn how to improve CLS CLS

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 non-composited animations

Animations which are not composited can be janky and increase CLS. Learn how to avoid non-composited animations CLS

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. TBT

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.

CONTRAST

Background and foreground colors do not have a sufficient contrast ratio.

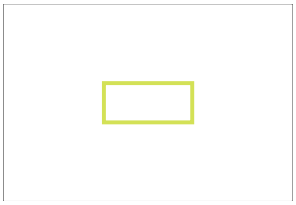
Low-contrast text is difficult or impossible for many users to read. Learn how to provide sufficient color contrast.

Failing Elements

p

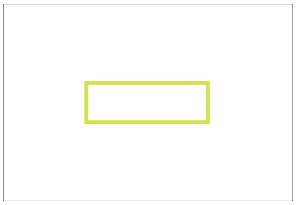
Failing Elements

header#header2



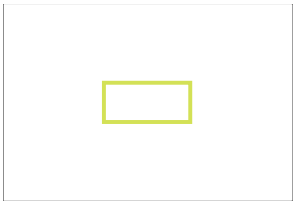
a

header#header2

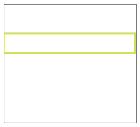


a

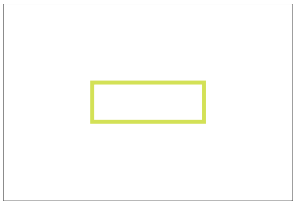
header#header2



a



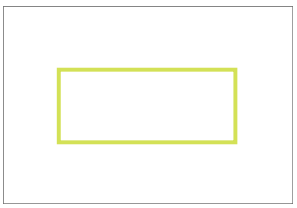
header#header2



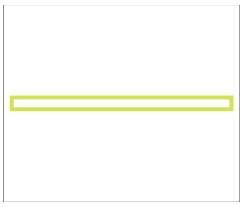
a

header#header2

Failing Elements

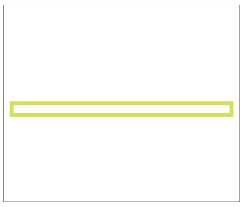


button.signup-btn



p

footer#contact



p

footer#contact

a

footer#contact

These are opportunities to improve the legibility of your content.

NAMES AND LABELS

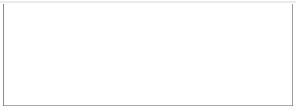
▲ `<frame>` or `<iframe>` elements do not have a title

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

Failing Elements

iframe

Failing Elements



iframe

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.

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 (20) Hide

- [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.

- [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.

- [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.

- [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.

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.

`[role]` values are valid

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

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.

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.

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.

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.

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.

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.

Values assigned to `role=""` are valid ARIA roles.

ARIA `roles` enable assistive technologies to know the role of each element on the web page. If the `role` values are misspelled, not existing ARIA `role` values, or abstract roles, then the purpose of the element will not be communicated to users of assistive technologies. Learn more about ARIA roles.

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 (38)

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.

`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.

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 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.

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.

- 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.

- `[id]` attributes on active, focusable elements are unique

  All focusable elements must have a unique `id` to ensure that they're visible to assistive technologies. Learn how to fix duplicate `ids`.

- 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.

- `<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.

- 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.

- 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.

- 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.

- 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.

- All heading elements contain content.

  A heading with no content or inaccessible text prevent screen reader users from accessing information on the page's structure. Learn more about headings.

- Identical links have the same purpose.

  Links with the same destination should have the same description, to help users understand the link's purpose and decide whether to follow it. Learn more about identical links.

- 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.

- Elements with visible text labels have matching accessible names.

  Visible text labels that do not match the accessible name can result in a confusing experience for screen reader users. Learn more about accessible names.

- Tables use `<caption>` instead of cells with the `[colspan]` attribute to indicate a caption.

  Screen readers have features to make navigating tables easier. Ensuring that tables use the actual caption element instead of cells with the `[colspan]` attribute may improve the experience for screen reader users. Learn more about captions.

- `<td>` elements in a large `<table>` have one or more table headers.

  Screen readers have features to make navigating tables easier. Ensuring that `<td>` elements in a large table (3 or more cells in width and height) have an associated table header may improve the experience for screen reader users. Learn more about table headers.

Best Practices

GENERAL

▲ Uses third-party cookies — 2 cookies found

Support for third-party cookies will be removed in a future version of Chrome. Learn more about phasing out third-party cookies.

Name	URL
Cloudflare CDN Cdn	
_ga	…css/all.min.css (cdnjs.cloudflare.com)
_ga_SQCRB0TXZW	…css/all.min.css (cdnjs.cloudflare.com)

▲ Issues were logged 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.

Issue type
Cookie
…css/all.min.css (cdnjs.cloudflare.com)

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

PASSED AUDITS (12) 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.

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.

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.

Avoids `unload` event listeners

The `unload` event does not fire reliably and listening for it can prevent browser optimizations like the Back-Forward Cache. Use `pagehide` or `visibilitychange` events instead. Learn more about unload event listeners

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

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.

URL	Map URL
Loom – Screen Recorder & Screen Capture Chrome Extension	
`chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/js/companionBubble.js`	`chrome-extension://liecbddmkiiihnedobmlmillhodjkdmb/js/companionBubble.js.map`
`Error: Failed fetching source map (null)`	

NOT APPLICABLE (2) Hide

Fonts with `font-display: optional` are preloaded

Preload `optional` fonts so first-time visitors may use them. Learn more about preloading fonts

Detected JavaScript libraries

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

SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

CONTENT BEST PRACTICES

- ▲ Document does not have a meta description

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

Format your HTML in a way that enables crawlers to better understand your app's content.

ADDITIONAL ITEMS TO MANUALLY CHECK (1) Hide

- ○ Structured data is valid

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

Run these additional validators on your site to check additional SEO best practices.

PASSED AUDITS (9) Hide

- 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. TBT

- 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.

- Page has successful HTTP status code

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

- Links have descriptive text

Descriptive link text helps search engines understand your content. Learn how to make links more accessible.

Links are crawlable

Search engines may use `href` attributes on links to crawl websites. Ensure that the `href` attribute of anchor elements links to an appropriate destination, so more pages of the site can be discovered. Learn how to make links crawlable

Page isn't blocked from indexing

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

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.

Document has a valid `hreflang`

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

Document avoids plugins

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

NOT APPLICABLE (4) Hide

robots.txt is valid

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 about robots.txt.

Document has a valid `rel=canonical`

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

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.

Tap targets are sized appropriately

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

As per Chrome's updated Installability Criteria, Lighthouse will be deprecating the PWA category in a future release. Please refer to the updated PWA documentation for future PWA testing.

PWA

These checks validate the aspects of a Progressive Web App. Learn what makes a good Progressive Web App.

INSTALLABLE

▲ Web app manifest or service worker do not meet the installability requirements — 1 reason

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. With proper service worker and manifest implementations, browsers can proactively prompt users to add your app to their homescreen, which can lead to higher engagement. Learn more about manifest installability requirements.

Failure reason
Page has no manifest <link> URL

PWA OPTIMIZED

▲ Is not configured for a custom splash screen

Failures: Manifest does not have a PNG icon of at least 512px, Manifest does not have `background_color`, Manifest does not have `theme_color`, Manifest does not have `name`.

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

▲ Does not set a theme color for the address bar.

Failures: Manifest does not have `theme_color`, No `<meta name="theme-color">` tag found.

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

○ 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 how to size content for the viewport.

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. TBT

▲ Manifest doesn't have a maskable icon

A maskable icon ensures that the image fills the entire shape without being letterboxed when installing the app on a device. Learn about maskable manifest icons.

ADDITIONAL ITEMS TO MANUALLY CHECK (3) Hide

○ Site works cross-browser

To reach the most number of users, sites should work across every major browser. Learn about cross-browser compatibility.

○ 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 about page transitions.

○ 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 about providing deep links.

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.

-

Publisher Ads

A Lighthouse plugin to improve ad speed and overall quality that is targeted at sites using GPT or AdSense tag. Learn more

NOT APPLICABLE (23) Hide

○ Tag load time — No tag requested

This metric measures the time for the ad tag's implementation script (pubads_impl.js for GPT; adsbygoogle.js for AdSense) to load after the page loads. Learn more.

First bid request time — No bids detected

This metric measures the elapsed time from the start of page load until the first bid request is made. Delayed bid requests will decrease impressions and viewability, and have a negative impact on ad revenue. Learn More.

First ad request time — No ads requested

This metric measures the elapsed time from the start of page load until the first ad request is made. Delayed ad requests will decrease impressions and viewability, and have a negative impact on ad revenue. Learn more.

Latency of first ad render — No ads rendered

This metric measures the time for the first ad iframe to render from page navigation. Learn more.

Cumulative ad shift — No layout shift events found

Measures layout shifts that were caused by ads or happened near ads. Reducing cumulative ad-related layout shift will improve user experience. Learn more.

Total ad JS blocking time — No ad-related requests

Ad-related scripts are blocking the main thread. Learn more.

GPT and bids loaded in parallel — GPT not requested

To optimize ad loading, bid requests should not wait on GPT to load. This issue can often be fixed by making sure that bid requests do not wait on `googletag.pubadsReady` or `googletag.cmd.push`. Learn More.

Header bidding is parallelized — No bids detected

Send header bidding requests simultaneously, rather than serially, to retrieve bids more quickly. Learn more.

No bottleneck requests found — No ad-related requests

Speed up, load earlier, parallelize, or eliminate the following requests and their dependencies in order to speed up ad loading. Learn More.

Ad scripts are loaded statically — No tag requested

Load the following scripts directly with `<script async src=...>` instead of injecting scripts with JavaScript. Doing so allows the browser to preload scripts sooner. Learn more.

Ads not blocked by load events — No ad-related requests

Waiting on load events increases ad latency. To speed up ads, eliminate the following load event handlers. Learn More.

Minimal render-blocking resources found — No tag requested

Render-blocking resources slow down tag load times. Consider loading critical JS/CSS inline or loading scripts asynchronously or loading the tag earlier in the head. Learn more.

No long tasks blocking ad-related network requests — No ad-related requests

Tasks blocking the main thread can delay ad requests and cause a poor user experience. Consider removing long blocking tasks or moving them off of the main thread. These tasks can be especially detrimental to performance on less powerful devices. Learn more.

Ad request waterfall — No ads requested

Consider reducing the number of resources, loading multiple resources simultaneously, or loading resources earlier to improve ad speed. Requests that block ad loading can be found below. Learn more.

Few or no ads loaded outside viewport — No visible slots

Too many ads loaded outside the viewport lowers viewability rates and impacts user experience. Consider loading ads below the fold lazily as the user scrolls down. Consider using GPT's Lazy Loading API. Learn more.

Ad tag is loaded asynchronously — No tag requested

Loading the ad tag synchronously blocks content rendering until the tag is fetched and loaded. Consider using the `async` attribute to load gpt.js and/or adsbygoogle.js asynchronously. Learn more.

Ad tag is loaded over HTTPS — No tag requested

For privacy and security, always load GPT/AdSense over HTTPS. Insecure pages should explicitly request the ad script securely. GPT Example: `<script async src="https://securepubads.g.doubleclick.net/tag/js/gpt.js">` AdSense Example: `<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js">`. Learn more.

GPT tag is loaded from an official source — GPT not requested

Load GPT from 'securepubads.g.doubleclick.net' for standard integrations or from 'pagead2.googlesyndication.com' for limited ads. Learn more.

Ads to page-height ratio is within recommended range — No visible slots

The ads to page-height ratio can impact user experience and ultimately user retention. Learn more.

- No ad found at the very top of the viewport — No visible slots

  Over 10% of ads are never viewed because users scroll past them before they become viewable. By moving ad slots away from the very top of the viewport, users are more likely to see ads before scrolling away. Learn more.

- No duplicate tags found — No tags requested

  Loading a tag more than once in the same page is redundant and adds overhead without benefit. Learn more.

- Deprecated GPT API Usage — GPT not requested

  Deprecated GPT API methods should be avoided to ensure your page is tagged correctly. Learn more.

- GPT Errors — GPT not requested

  Fix GPT errors to ensure your page is tagged as intended. Learn more.

Captured at Apr 28, 2024, 2:06 PM GMT+5:30	Emulated Desktop with Lighthouse 11.6.0	Single page session
Initial page load	Custom throttling	Using Chromium 124.0.0.0 with devtools

Generated by **Lighthouse** 11.6.0 | File an issue