99	100	92	100
Performance	Accessibility	Best Practices	SEO

99

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	Largest Contentful Paint
1.6 s	1.6 s

Total Blocking Time	Cumulative Layout Shift
60 ms	0.045
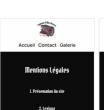
Speed Index	
1.6 s	

View Treemap   View trace

DIAGNOSTICS

◯ 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: **1,255.062 ms**

Initial Navigation

/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)

/style.css (tatoueur-a-mes-heures-new.vercel.app) **- 633.445 ms, 2.38 KiB**

◯ Keep request counts low and transfer sizes small — 6 requests • 62 KiB ⌃

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

Resource type	Requests	Transfer size
Total	6.0	62.2 KiB
Font	3.0	46.6 KiB
Document	1.0	11.8 KiB
Stylesheet	1.0	2.4 KiB
Script	1.0	1.4 KiB
Image	0.0	0.0 KiB
Media	0.0	0.0 KiB
Other	0.0	0.0 KiB
Third-party	0.0	0.0 KiB

Largest contentful paint element — 1,600 ms
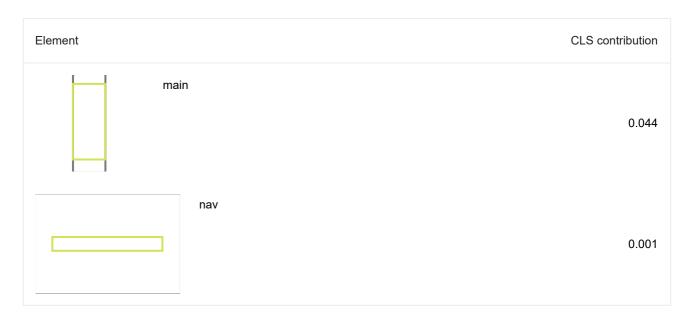
This is the largest contentful element painted within the viewport. Learn more about the Largest Contentful Paint element LCP

Element
dt.special_font

Phase	% of LCP	Timing
TTFB	21%	340 ms
Load delay	0%	0 ms
Load time	0%	0 ms
Render delay	79%	1,260 ms

Avoid large layout shifts — 2 elements found

These DOM elements contribute most to the CLS of the page. Learn how to improve CLS CLS

Element	CLS contribution
main	0.044
nav	0.001

Avoid long main-thread tasks — 5 long tasks 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
vercel.app [First Party]		**546 ms**
/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)	1,270 ms	312 ms
/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)	1,684 ms	98 ms
/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)	662 ms	69 ms
/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)	592 ms	67 ms
Unattributable		**61 ms**
Unattributable	1,623 ms	61 ms

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

PASSED AUDITS (35)   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]

Properly size images   ⌃

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

Defer off-screen 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 CSS   ⌃

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

Minify JavaScript — Potential savings of 8 KiB   ⌃

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

URL	Transfer size	Potential savings
chrome-extension://cjpalhdlnbpafiamejdnhcphjbkeiagm/js/contentscript.js	15.2 KiB	8.0 KiB

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. LCP

Efficiently encode images ⌃

Optimised images load faster and consume less mobile data. Learn how to efficiently encode images.

Serve images in next-gen formats ⌃

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.

Enable text compression ⌃

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

Pre-connect 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

Initial server response time was short — Root document took 330 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
vercel.app First Party	330 ms

URL	Time Spent
/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)	330 ms

Avoid multiple page redirects ∧

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

◯ Pre-load key requests ∧

Consider using `<link rel=preload>` to prioritise 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]

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. For your bundled JavaScript, adopt a modern script deployment strategy using module/nomodule feature detection to reduce the amount of code delivered to modern browsers, while retaining support for legacy browsers. Learn how to use modern JavaScript [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]

Avoids enormous network payloads — Total size was 62 KiB ∧

Large network payloads cost users real money and are highly correlated with long load times. Learn how to reduce payload sizes. [LCP]
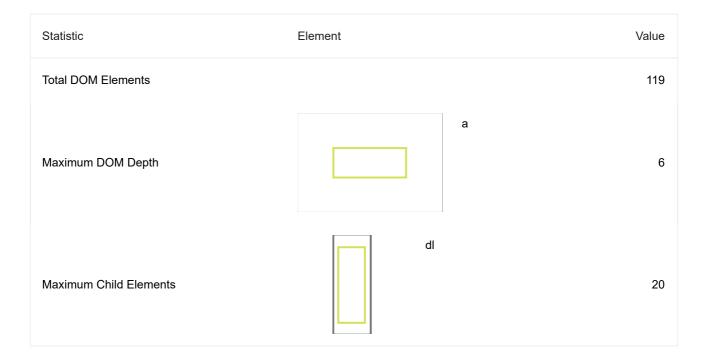
URL	Transfer size
vercel.app [First Party]	**62.2 KiB**
…fonts/manrope400.woff2 (tatoueur-a-mes-heures-new.vercel.app)	23.9 KiB
…fonts/manrope700.woff2 (tatoueur-a-mes-heures-new.vercel.app)	14.0 KiB
/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)	11.8 KiB
…fonts/pirataone400.woff2 (tatoueur-a-mes-heures-new.vercel.app)	8.7 KiB
/style.css (tatoueur-a-mes-heures-new.vercel.app)	2.4 KiB
/script-build.js (tatoueur-a-mes-heures-new.vercel.app)	1.4 KiB

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.

Avoids an excessive DOM size — 119 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		119
Maximum DOM Depth	a	6
Maximum Child Elements	dl	20

◯ 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 — 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 reduce Javascript execution time. [TBT]

URL	Total CPU Time	Script Evaluation	Script Parse
vercel.app [First Party]	**565 ms**	**30 ms**	**15 ms**
/mentions_legales.html (tatoueur-a-mes-heures-new.vercel.app)	565 ms	30 ms	15 ms
Unattributable	**197 ms**	**9 ms**	**0 ms**
Unattributable	197 ms	9 ms	0 ms

Minimises main-thread work — 0.8 s ⌃

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

Category	Time Spent
Style & Layout	391 ms
Other	268 ms
Script Evaluation	53 ms
Parse HTML & CSS	48 ms
Script Parsing & Compilation	16 ms
Rendering	15 ms

All text remains visible during webfont loads ⌃

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

◯ Minimise third-party usage ⌃

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 minimise third-party impact.](#) (TBT)

○ 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)

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 that are not composited can be poor, slow and increase CLS. [Learn how to avoid non-composited animations](#) (CLS)

Image elements 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)

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

A `<meta name="viewport">` not only optimises 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)

Page didn't prevent back-forward cache restoration ⌃

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](#)

100

100

Accessibility

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

ADDITIONAL ITEMS TO MANUALLY CHECK (10)   Hide

○ 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](#).

○ 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 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](#).

○ 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](#).

○ 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](#).

○ 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.

○ 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.

○ 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.

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

PASSED AUDITS (13)   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 alternative 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 colours have a sufficient contrast ratio ⌃

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

<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](#).

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 have a discernible name

Link text (and alternative 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](#).

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](#).

NOT APPLICABLE (35)   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](#).

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

Focusable descendants 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](#).

○ `[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](#).

○ 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 labelling `treeitem` elements](#).

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

Assistive technologies, such as 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, such as 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](#).

○ `[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 `id`s](#).

○ 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, such as screen readers. Learn more about form element labels.

○ 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 alternative text ⌃

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

○ 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 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.

○ 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 that`<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 that 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.

92

Best Practices

USER EXPERIENCE

⚠ Serves images with low resolution ⌃

Image natural dimensions should be proportional to the display size and the pixel ratio to maximise image clarity. [Learn how to provide responsive images](#).

	URL	Displayed size	Actual size	Expected size
img	data:image/webp;base64,UklGRm4PAAB XRUJQVlA4WAoAAAAQAAAAYwAATwA A...	100 x 80	100 x 80	150 x 120

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 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](#).

Allows users to paste into input fields ⌃

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

Displays images with correct aspect ratio ⌃

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

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 1,024 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 optimisations like the back-forward cache. Use `pagehide` or `visibilitychange` events instead. Learn more about unload event listeners

Avoids deprecated APIs ⌃

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

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 to 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.

○   Fonts with `font-display: optional` are preloaded   ⌃

Preload `optional` fonts so that 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](#).

100

SEO

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

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 (12)   Hide

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

A `<meta name="viewport">` not only optimises 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](#).

Document has a meta description   ⌃

Meta descriptions may be included in search results to concisely summarise page content. [Learn more about the meta description](#).

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 that 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 alternative 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 uses legible font sizes — 100% legible text ⌃

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](#).

Source	Selector	% of page text	Font size
Legible text		100.00%	≥ 12px

Document avoids plugins ⌃

Search engines can't index plug-in content and many devices restrict plug-ins or don't support them. [Learn more about avoiding plugins](#).

Tap targets are sized appropriately — 100% appropriately sized tap targets ⌃

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

NOT APPLICABLE (2)   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.