https://peterdg.com.ar/

Performance	Accessibility	Best Practices	SEO
90	100	100	100

▲ 0–49 ■ 50–89 ● 90–100

90

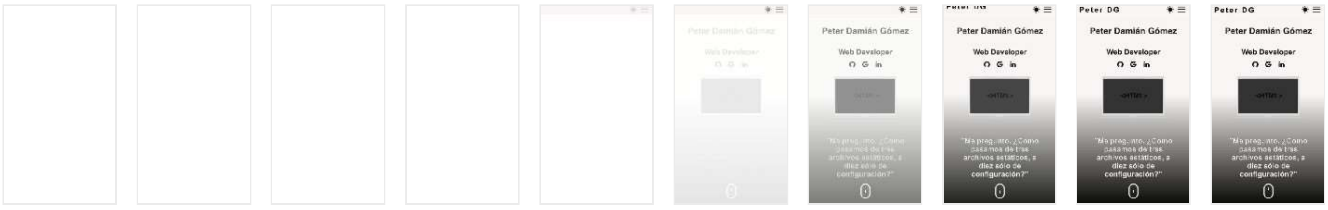
Performance

Metrics

● First Contentful Paint	0.9 s	● Time to Interactive	2.1 s
■ Speed Index	5.4 s	■ Total Blocking Time	250 ms
● Largest Contentful Paint	2.1 s	● Cumulative Layout Shift	0

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

View Treemap



Show audits relevant to: All FCP LCP TBT CLS

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

Opportunity	Estimated Savings
▲ Reduce initial server response time	1.47 s

Keep the server response time for the main document short because all other requests depend on it. Learn more. FCP LCP

If you are server-side rendering any React components, consider using `renderToNodeStream()` or `renderToStaticNodeStream()` to allow the client to receive and hydrate different parts of the markup instead of all at once. Learn more.

☐ ~~Show 3rd-party resources~~ (0)

URL	Time Spent
https://peterdg.com.ar	1,570 ms

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

■ Minimize main-thread work — 2.3 s

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

Category	Time Spent
Script Evaluation	730 ms
Style & Layout	626 ms
Other	518 ms
Rendering	364 ms
Script Parsing & Compilation	53 ms
Parse HTML & CSS	26 ms

● User Timing marks and measures — 4 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.

Use the React DevTools Profiler, which makes use of the Profiler API, to measure the rendering performance of your components. Learn more.

Name	Type	Start Time	Duration
Next.js-before-hydration	Measure	0 ms	2,007.24 ms
Next.js-hydration	Measure	2,007.24 ms	19.11 ms
beforeRender	Mark	2,007.28 ms	
afterHydrate	Mark	2,026.38 ms	

Keep request counts low and transfer sizes small — 13 requests • 154 KiB

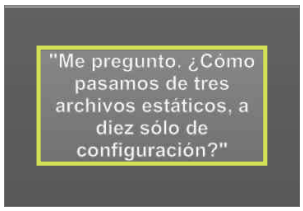
To set budgets for the quantity and size of page resources, add a budget.json file. Learn more.

Resource Type	Requests	Transfer Size
Total	13	153.9 KiB
Script	9	144.2 KiB
Stylesheet	3	6.7 KiB
Document	1	3.1 KiB
Image	0	0.0 KiB
Media	0	0.0 KiB
Font	0	0.0 KiB
Other	0	0.0 KiB
Third-party	0	0.0 KiB

Largest Contentful Paint element — 1 element found

This is the largest contentful element painted within the viewport. Learn More LCP

Element



"Me pregunto. ¿Cómo pasamos de tres archivos estáticos, a diez sólo de configur…
`<h4 style="opacity: 1;">`

Avoid long main-thread tasks — 4 long tasks found

Lists the longest tasks on the main thread, useful for identifying worst contributors to input delay. Learn more TBT

~~Show 3rd-party resources~~ (0)

URL	Start Time	Duration
…chunks/framework-c93ed74….js (peterdg.com.ar)	1,709 ms	252 ms
…chunks/375.bc40e54….js (peterdg.com.ar)	1,961 ms	86 ms
…chunks/537.8443575….js (peterdg.com.ar)	2,111 ms	77 ms
https://peterdg.com.ar	809 ms	52 ms

Avoid non-composited animations — 1 animated element found

Animations which are not composited can be janky and increase CLS. Learn more CLS

Element	Name
 Peter DG `<h1>`	
Unsupported CSS Property: top	appbar_enterTitle__1IWP-

Passed audits (29)

- 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 more. FCP LCP

- Properly size images

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

- Defer offscreen images

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

- Minify CSS

  Minifying CSS files can reduce network payload sizes. Learn more. FCP LCP

  If your build system minifies CSS files automatically, ensure that you are deploying the production build of your application. You can check this with the React Developer Tools extension. Learn more.

- Minify JavaScript

  Minifying JavaScript files can reduce payload sizes and script parse time. Learn more. FCP LCP

  If your build system minifies JS files automatically, ensure that you are deploying the production build of your application. You can check this with the React Developer Tools extension. Learn more.

- 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 more. FCP LCP

- Reduce unused JavaScript — Potential savings of 27 KiB

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

If you are not server-side rendering, split your JavaScript bundles with `React.lazy()`. Otherwise, code-split using a third-party library such as loadable-components.

Show 3rd-party resources (0)

URL	Transfer Size	Potential Savings
…pages/_app-c5a6d27….js (peterdg.com.ar)	58.3 KiB	27.1 KiB

Efficiently encode images

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

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.

Enable text compression

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

Preconnect to required origins

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

Avoid multiple page redirects

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

If you are using React Router, minimize usage of the `<Redirect>` component for route navigations.

Preload key requests

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

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 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 — Potential savings of 10 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 More TBT

~~Show 3rd-party resources~~ (0)

URL		Potential Savings
…pages/_app-c5a6d27….js (peterdg.com.ar)		9.9 KiB
…pages/_app-c5a6d27….js:1:156633 (peterdg.com.ar)	`Array.prototype.find`	
…pages/_app-c5a6d27….js:1:160233 (peterdg.com.ar)	`@babel/plugin-transform-classes`	
…chunks/main-9fa3b3c….js (peterdg.com.ar)		0.1 KiB

URL	Potential Savings
…chunks/main-9fa3b3c….js:1:52368 (peterdg.com.ar)	`@babel/plugin-transform-classes`

Preload Largest Contentful Paint image

Preload the image used by the LCP element in order to improve your LCP time. Learn more. LCP

Avoids enormous network payloads — Total size was 154 KiB

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

~~Show 3rd-party resources~~ (0)

URL	Transfer Size
…pages/_app-c5a6d27….js (peterdg.com.ar)	58.3 KiB
…chunks/framework-c93ed74….js (peterdg.com.ar)	43.9 KiB
…chunks/main-9fa3b3c….js (peterdg.com.ar)	21.5 KiB
…chunks/537.8443575….js (peterdg.com.ar)	6.8 KiB
…chunks/375.bc40e54….js (peterdg.com.ar)	5.9 KiB
…pages/index-bf00ada….js (peterdg.com.ar)	3.5 KiB
…css/7c0ffc4….css (peterdg.com.ar)	3.4 KiB
https://peterdg.com.ar	3.1 KiB
…css/6157044….css (peterdg.com.ar)	2.6 KiB
…chunks/webpack-d8ef315….js (peterdg.com.ar)	2.6 KiB

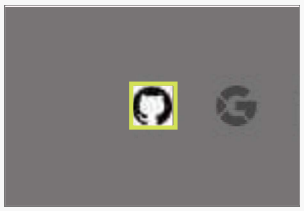
Uses efficient cache policy on static assets — 0 resources found

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

● Avoids an excessive DOM size — 60 elements

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

Consider using a "windowing" library like `react-window` to minimize the number of DOM nodes created if you are rendering many repeated elements on the page. Learn more. Also, minimize unnecessary re-renders using `shouldComponentUpdate`, `PureComponent`, or `React.memo` and skip effects only until certain dependencies have changed if you are using the `Effect` hook to improve runtime performance.

Statistic	Element	Value
Total DOM Elements		60
Maximum DOM Depth		article.socialmedia_socialmedia__PE2Zo > a > svg > path `<path d="M256 5a257 257 0 00-81 501c13 3 18-5 18-12l-1-44c-71 16-86-34-86-34-12-30-…">` 10
Maximum Child Elements		Notebook SVG `<svg aria-label="Notebook SVG" role="figure" width="90%" height="90%" viewBox="0 0 2200 1500">` 9

○ Avoid chaining critical requests

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

Maximum critical path latency: **1,570 ms**

Initial Navigation

└ https://peterdg.com.ar **- 1,570 ms, 3.06 KiB**

● JavaScript execution time — 0.8 s

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

Show 3rd-party resources (0)

URL	Total CPU Time	Script Evaluation	Script Parse
…pages/_app-c5a6d27….js (peterdg.com.ar)	857 ms	292 ms	12 ms
https://peterdg.com.ar	705 ms	18 ms	6 ms
…chunks/375.bc40e54….js (peterdg.com.ar)	180 ms	69 ms	5 ms
Unattributable	139 ms	16 ms	1 ms
…chunks/main-9fa3b3c….js (peterdg.com.ar)	116 ms	109 ms	5 ms
…chunks/framework-c93ed74….js (peterdg.com.ar)	114 ms	49 ms	11 ms
…pages/index-bf00ada….js (peterdg.com.ar)	104 ms	101 ms	3 ms
…chunks/537.8443575….js (peterdg.com.ar)	76 ms	69 ms	5 ms

All text remains visible during webfont loads

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

Minimize 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 more. 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 more. TBT

Avoid large layout shifts

These DOM elements contribute most to the CLS of the page. 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.

- Avoids `document.write()`

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

- Image elements have explicit `width` and `height`

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

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

review.

- The page has a logical tab order

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

- Interactive controls are keyboard focusable

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

- Interactive elements indicate their purpose and state

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

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

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

- User focus is not accidentally trapped in a region

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

- Custom controls have associated labels

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

- Custom controls have ARIA roles

  Custom interactive controls have appropriate ARIA roles. Learn more.

- Visual order on the page follows DOM order

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

- Offscreen content is hidden from assistive technology

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

- 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.

Passed audits (15)

- `[aria-*]` attributes match their roles

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

- `[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 more.

- `[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 more.

- `[role]`s have all required `[aria-*]` attributes

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

- `[role]` values are valid

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

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

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

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

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

- 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.

- Background and foreground colors have a sufficient contrast ratio

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

- 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.

- 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.

- <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.

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

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

- Links have a discernible name

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

- `[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.

Not applicable (29)

- `[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.

- `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 more.

- 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.

- ARIA `meter` 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 more.

- 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 more.

- 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.

- `[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.

- 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.

- ARIA `tooltip` 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 more.

- ARIA `treeitem` 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 more.

- Buttons have an accessible name

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

- `<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 more.

- Definition list items are wrapped in `<dl>` elements

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

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

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

- ARIA IDs are unique

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

- 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 more.

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

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

- Image elements have `[alt]` attributes

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

- `<input type="image">` elements have `[alt]` text

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

- Form elements have associated labels

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

- Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).

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

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

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

- 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.

- `<object>` elements have `[alt]` text

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

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

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

- Cells in a `<table>` element that use the `[headers]` attribute 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.

- `<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.

- `[lang]` attributes have a valid value

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

- `<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.

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 more

Description	Directive	Severity
No CSP found in enforcement mode		High

Passed audits (17)

- 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.

- Links to cross-origin destinations are safe

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

- Avoids requesting the geolocation permission on page load

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.

- Avoids requesting the notification permission on page load

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

- Avoids front-end JavaScript libraries with known security vulnerabilities

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

- Allows users to paste into password fields

  Preventing password pasting undermines good security policy. Learn more.

- Displays images with correct aspect ratio

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

- Serves images with appropriate resolution

  Image natural dimensions should be proportional to the display size and the pixel ratio to maximize image clarity. Learn more.

- Page has the HTML doctype

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

- 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.

- Avoids `unload` event listeners

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

- Avoids Application Cache

Application Cache is deprecated. Learn more.

- Detected JavaScript libraries

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

Name	Version
React	
Next.js	11.0.0
core-js	core-js-global@3.12.1

- Avoids deprecated APIs

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

- 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

- 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.

- 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.

Not applicable (1)

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

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

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.

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

- Structured data is valid

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

Passed audits (12)

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

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

- 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.

- Document has a meta description

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

- Page has successful HTTP status code

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

- Links have descriptive text

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

- 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 More

- 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.

- 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.

● 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.

● Document uses legible font sizes — 86.36% 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.

☐ ~~Show 3rd-party resources~~ (0)

Source	Selector	% of Page Text	Font Size
…css/7c0ffc4….css:1:568 (peterdg.com.ar)	`small`	13.64%	10.4px
`Legible text`		86.36%	≥ 12px

● Document avoids plugins

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

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

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

Not applicable (2)

● 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.

- Document has a valid `rel=canonical`

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

Runtime Settings

URL	https://peterdg.com.ar/
Fetch Time	Sep 6, 2021, 4:25 PM GMT-3
Device	Emulated Moto G4
Network throttling	Unknown
CPU throttling	Unknown
Channel	lr
User agent (host)	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/90.0.4430.97 Safari/537.36
User agent (network)	Mozilla/5.0 (Linux; Android 7.0; Moto G (4)) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4590.2 Mobile Safari/537.36 Chrome-Lighthouse
CPU/Memory Power	1309

Generated by **Lighthouse** 8.3.0 | File an issue