Performance

96

🕑 **Metrics**

First Contentful Paint	1.330 ms ✅	First Meaningful Paint	1.540 ms ✅
Speed Index	1.330 ms ✅	First CPU Idle	3.430 ms ✅
Time to Interactive	3.430 ms ✅	Estimated Input Latency	21 ms ✅

Values are estimated and may vary.

📑 **Opportunities**

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

Resource to optimize	Estimated Savings
1 Avoid multiple, costly round trips to any origin	— 0,15 s ∧

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

Origin	Potential Savings
http://localhost:1337	150 ms

🔍 **Diagnostics**

More information about the performance of your application.

1 Uses efficient cache policy on static assets	1 asset found ✅ ∧

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

URL	Cache TTL	Size (KB)
…api/js?key=AIzaSyACH…&libraries=places&callback=init…	30 m	27 KB

| 2 | Minimizes main thread work | 1.770 ms ✓ ⌃ |

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

Category	Time Spent
Other	864 ms
Script Evaluation	634 ms
Style & Layout	100 ms
Garbage Collection	72 ms
Script Parsing & Compilation	51 ms
Rendering	27 ms
Parse HTML & CSS	24 ms

| 3 | Critical Request Chains | 1 chains found ⌃ |

The Critical Request Chains below show you what resources are issued 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.

Longest chain: **119,9ms** over **2** requests, totalling **0,2 KB**
Initial Navigation
  / (localhost)
    /js/main.js (localhost) **- 43,9ms, 0,24 KB**

| ✓ | **Passed audits** | 18 audits ⌃ |

| 1 | 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.

| 2 | Properly size images | ✓ ⌃ |

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

| 3 | Defer offscreen images | ✓ ⌃ |

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

| 4 | Minify CSS | ✓ ⌃ |

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

5 Minify JavaScript ✅ ⌄

Minifying JavaScript files can reduce payload sizes and script parse time. [Learn more](#).

6 Defer unused CSS   Potential savings of 2 KB ✅ ⌄

Remove unused rules from stylesheets to reduce unnecessary bytes consumed by network activity.
[Learn more](#).

URL	Original	Potential Savings
/css/styles.css (localhost)	3 KB	2 KB

7 Efficiently encode images ✅ ⌄

Optimized images load faster and consume less cellular data. [Learn more](#).

8 Serve images in next-gen formats ✅ ⌄

Image formats like JPEG 2000, JPEG XR, and WebP often provide better compression than PNG or
JPEG, which means faster downloads and less data consumption. [Learn more](#).

9 Enable text compression   Potential savings of 5 KB ✅ ⌄

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

Uncompressed resource URL	Original	GZIP Savings
/restaurants (localhost)	7 KB	5 KB

10 Keep server response times low (TTFB) ✅ ⌄

Time To First Byte identifies the time at which your server sends a response. [Learn more](#).

11 Avoid multiple page redirects   0 ms ✅ ⌄

Redirects introduce additional delays before the page can be loaded. [Learn more](#).

12 Preload key requests   Potential savings of 0 ms ✅ ⌄

Consider using <link rel=preload> to prioritize fetching late-discovered resources sooner. [Learn more](#).

13 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](#)

14 Avoids enormous network payloads   Total size was 49 KB ✅ ⌄

Large network payloads cost users real money and are highly correlated with long load times. [Learn
more](#).

URL	Total Size	Transfer Time
…api/js?key=AIzaSyACH…&libraries=places&callback=initMap ..	27 KB	40 ms
/restaurants (localhost)	7 KB	10 ms
/css/styles.css (localhost)	3 KB	0 ms
/js/main.js (localhost)	2 KB	0 ms
/js/dbhelper.js (localhost)	2 KB	0 ms
/js/idbhelper.js (localhost)	2 KB	0 ms
http://localhost:8080	1 KB	0 ms
/js/utils.js (localhost)	1 KB	0 ms
/js/maphelper.js (localhost)	1 KB	0 ms
/js/imghelper.js (localhost)	1 KB	0 ms

15   Avoids an excessive DOM size   55 nodes ✓ ^

Browser engineers recommend pages contain fewer than ~1.500 DOM nodes. The sweet spot is a tree depth < 32 elements and fewer than 60 children/parent element. A large DOM can increase memory usage, cause longer style calculations, and produce costly layout reflows. Learn more.

Total DOM Nodes	Maximum DOM Depth	Maximum Children
55	7	17
	<option value="all">	<head>

16   User Timing marks and measures   ✓ ^

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

17   JavaScript boot-up time   400 ms ✓ ^

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

URL	Total	Script Evaluation	Script Parsing & Compilation
…api/js?key=AIzaSyACH…&libraries…	264 ms	247 ms	17 ms
…js/inject.js (melpgahbngpgnbhhccnop…	85 ms	85 ms	0 ms
/closure/closure_preinit.js (mipjggdmda..	53 ms	52 ms	0 ms

18   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.

http://localhost:8080/
Jun 29, 2018, 11:11 AM GMT+2
Emulated Nexus 5X, Simulated Fast 3G network

Lighthouse 3.0.0   http://localhost:8080/   ✎

96   **92**   **100**   **93**   **89**
Performance   Progressive Web App   Accessibility   Best Practices   SEO

To reach the most number of users, sites should work across every major browser. Learn more.

2   Page transitions don't feel like they block on the network   ⌃

Transitions should feel snappy as you tap around, even on a slow network, a key to perceived performance. Learn more.

3   Each page has a URL   ⌃

Ensure individual pages are deep linkable via the URLs and that URLs are unique for the purpose of shareability on social media. Learn more.

✓ **Passed audits**   11 audits ⌃

1   Page load is fast enough on 3G   ✓ ⌃

A fast page load over a 3G network ensures a good mobile user experience. Learn more.

2   Responds with a 200 when offline   ✓ ⌃

If you're building a Progressive Web App, consider using a service worker so that your app can work offline. Learn more.

3   User can be prompted to Install the Web App   ✓ ⌃

Browsers can proactively prompt users to add your app to their homescreen, which can lead to higher engagement. Learn more.

4   Uses HTTPS   ✓ ⌃

All sites should be protected with HTTPS, even ones that don't handle sensitive data. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users, and is a prerequisite for HTTP/2 and many new web platform APIs. Learn more.

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

Add a viewport meta tag to optimize your app for mobile screens. Learn more.

6   Registers a service worker   ✓ ⌃

The service worker is the technology that enables your app to use many Progressive Web App features, such as offline, add to homescreen, and push notifications. Learn more.

7   Contains some content when JavaScript is not available   ✓ ⌃

Your app should display some content when JavaScript is disabled, even if it's just a warning to the user that JavaScript is required to use the app. Learn more.

8   Configured for a custom splash screen   ✓ ⌃

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

| 9 | Address bar matches brand colors | ✓ ∧ |

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

| 10 | Content is sized correctly for the viewport | ✓ ∧ |

If the width of your app's content doesn't match the width of the viewport, your app might not be optimized for mobile screens. Learn more.

| 11 | The short_name won't be truncated on the homescreen | ✓ ∧ |

Make your app's `short_name` fewer than 12 characters to ensure that it's not truncated on homescreens. Learn more.

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.

(100)

🔍 **Additional items to manually check**   10 audits ∧

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

| 1 | The page has a logical tab order | ∧ |

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

| 2 | Interactive controls are keyboard focusable | ∧ |

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

| 3 | 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.

| 4 | 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.

| 5 | Custom controls have associated labels | ∧ |

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

| 6 | Custom controls have ARIA roles | ∧ |

Custom interactive controls have appropriate ARIA roles. Learn more.

| 7 | Visual order on the page follows DOM order | ∧ |

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

| 8 | Offscreen content is hidden from assistive technology | ∧ |

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

| 9 | Headings don't skip levels | ∧ |

Headings are used to create an outline for the page and heading levels are not skipped. Learn more.

| 10 | 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**   17 audits ∧

Elements Use Attributes Correctly   ∧
These are opportunities to improve the configuration of your HTML elements.

1   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.

ARIA Attributes Follow Best Practices   ∧
These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

1   `[aria-*]` attributes match their roles   ✓ ∧

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

2   `[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.

3   Elements with `[role]` that require specific children `[role]`s, are present   ✓ ∧

Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. Learn more.

4   `[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.

5   `[role]` values are valid   ✓ ∧

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

6   `[aria-*]` attributes have valid values   ✓ ∧

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

7   `[aria-*]` attributes are valid and not misspelled   ✓ ∧

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

Elements Have Discernible Names   ∧
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.

1   Links have a discernible name   ✓ ∧

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

Elements Describe Contents Well

These are opportunities to make your content easier to understand for a user of assistive technology, like a screen reader.

1   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.

2   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.

3   Form elements have associated labels   ✓ ⌄

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

Color Contrast Is Satisfactory

These are opportunities to improve the legibility of your content.

1   Background and foreground colors have a sufficient contrast ratio   ✓ ⌄

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

Elements Are Well Structured

These are opportunities to make sure your HTML is appropriately structured.

1   `[id]` attributes on the page are unique   ✓ ⌄

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

Page Specifies Valid Language

These are opportunities to improve the interpretation of your content by users in different locales.

1   `<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.

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

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

Meta Tags Used Properly

These are opportunities to improve the user experience of your site.

1   `[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**　　　　　　　　　　　　　　　　　18 audits ⌃

Elements Use Attributes Correctly　　　　　　　　　　　　　⌃
These are opportunities to improve the configuration of your HTML elements.

1　`[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.

2　`<audio>` elements contain a `<track>` element with `[kind="captions"]`　　✔ ⌃

Captions make audio elements usable for deaf or hearing-impaired users, providing critical information such as who is talking, what they're saying, and other non-speech information. Learn more.

3　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.

4　`<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.

5　Cells in a `<table>` element that use the `[headers]` attribute only refer to other cells of that　✔ ⌃
same table.

Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more.

6　`<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.

Elements Have Discernible Names　　　　　　　　　　　　　⌃
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.

1　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.

Elements Describe Contents Well　　　　　　　　　　　　　⌃
These are opportunities to make your content easier to understand for a user of assistive technology, like a screen reader.

1　`<frame>` or `<iframe>` elements have a title　　　　　　　　　✔ ⌃

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

2　Presentational `<table>` elements avoid using `<th>`, `<caption>` or the `[summary]` attribute.　✔ ⌃

A table being used for layout purposes should not include data elements, such as the th or caption elements or the summary attribute, because this can create a confusing experience for screen reader users. Learn more.

　　`<object>` elements have `[alt]` text　　　　　　　　　　　✔ ⌃

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

4   `<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.

5   `<video>` elements contain a `<track>` element with [kind="description"]   ✅ ⌄

       Audio descriptions provide relevant information for videos that dialogue cannot, such as facial
       expressions and scenes. Learn more.

Elements Are Well Structured   ⌄
These are opportunities to make sure your HTML is appropriately structured.

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

       When definition lists are not properly marked up, screen readers may produce confusing or inaccurate
       output. Learn more.

2   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.

3   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.

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

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

Page Specifies Valid Language   ⌄
These are opportunities to improve the interpretation of your content by users in different locales.

1   [lang] attributes have a valid value   ✅ ⌄

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

Meta Tags Used Properly   ⌄
These are opportunities to improve the user experience of your site.

1   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.

Best Practices   93

1   Does not use HTTP/2 for all of its resources   9 requests not served via HTTP/2 ⚠ ⌄

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

URL	Protocol
http://localhost:8080	http/1.1
/css/styles.css (localhost)	http/1.1
/js/register_sw.js (localhost)	http/1.1
/js/utils.js (localhost)	http/1.1
/js/imghelper.js (localhost)	http/1.1
/js/idbhelper.js (localhost)	http/1.1
/js/dbhelper.js (localhost)	http/1.1
/js/maphelper.js (localhost)	http/1.1
/js/main.js (localhost)	http/1.1

✓ **Passed audits**   13 audits ⌃

1   Avoids Application Cache   ✓ ⌃

Application Cache is deprecated. [Learn more](#).

2   Avoids WebSQL DB   ✓ ⌃

Web SQL is deprecated. Consider using IndexedDB instead. [Learn more](#).

3   Uses HTTPS   ✓ ⌃

All sites should be protected with HTTPS, even ones that don't handle sensitive data. HTTPS prevents intruders from tampering with or passively listening in on the communications between your app and your users, and is a prerequisite for HTTP/2 and many new web platform APIs. [Learn more](#).

4   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](#).

5   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](#).

6   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](#).

7   Avoids requesting the geolocation permission on page load   ✓ ⌃

Users are mistrustful of or confused by sites that request their location without context. Consider tying the request to user gestures instead. [Learn more](#).

8   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](#).

9   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. <u>Learn more</u>.

10   Avoids deprecated APIs   ✅ ⌄

Deprecated APIs will eventually be removed from the browser. <u>Learn more</u>.

11   Allows users to paste into password fields   ✅ ⌄

Preventing password pasting undermines good security policy. <u>Learn more</u>.

12   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.

13   Displays images with correct aspect ratio   ✅ ⌄

Image display dimensions should match natural aspect ratio. <u>Learn more</u>.

SEO

These checks ensure that your page is optimized for search engine results ranking. There are additional factors Lighthouse does not check that may affect your search ranking. <u>Learn more</u>.

89

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

1   Document does not have a meta description   ⚠️ ⌄

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

🔍 **Additional items to manually check**   2 audits ⌄

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

1   Page is mobile friendly   ⌄

Take the <u>Mobile-Friendly Test</u> to check for audits not covered by Lighthouse, like sizing tap targets appropriately. <u>Learn more</u>.

2   Structured data is valid   ⌄

Run the <u>Structured Data Testing Tool</u> and the <u>Structured Data Linter</u> to validate structured data. <u>Learn more</u>.

✓ **Passed audits**   8 audits ⌄

Mobile Friendly   ⌄
Make sure your pages are mobile friendly so users don't have to pinch or zoom in order to read the content pages. <u>Learn more</u>.

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

Add a viewport meta tag to optimize your app for mobile screens. <u>Learn more</u>.

✅ ⌄

2   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.

Source	Selector	% of Page Text	Font Size
Legible text		100.00%	≥ 12px

Content Best Practices ^

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

1   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.

2   Links have descriptive text   ✓ ^

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

3   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.

4   Document avoids plugins   ✓ ^

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

Crawling and Indexing ^

To appear in search results, crawlers need access to your app.

1   Page has successful HTTP status code   ✓ ^

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

2   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.

⊖ **Not applicable**   2 audits ^

Content Best Practices ^

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

1   Document has a valid `rel=canonical`   ✓ ^

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

Crawling and Indexing ^

To appear in search results, crawlers need access to your app.

1   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.

Runtime settings

- **URL:** http://localhost:8080/
- **Fetch time:** Jun 29, 2018, 11:11 AM GMT+2
- **Device:** Emulated Nexus 5X
- **Network throttling:** 150 ms TCP RTT, 1.638,4 Kbps throughput (Simulated)
- **CPU throttling:** 4x slowdown (Simulated)
- **User agent:** Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36

Generated by **Lighthouse** 3.0.0 | File an issue