http://localhost:1234/

There were issues affecting this run of Lighthouse:

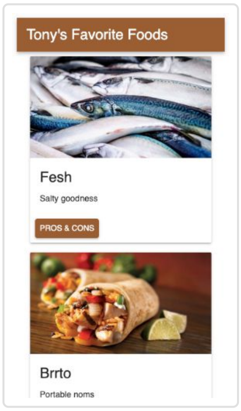
- **There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.**

39

Performance

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

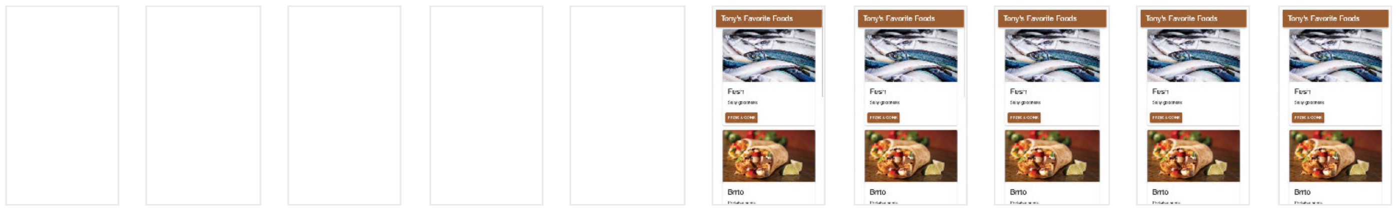
▲ 0–49 50–89 90–100



METRICS Expand view

▲ First Contentful Paint 7.2 s	▲ Time to Interactive 7.8 s
▲ Speed Index 7.2 s	Total Blocking Time 550 ms
▲ Largest Contentful Paint 7.9 s	Cumulative Layout Shift 0

View Treemap View Original Trace



Show audits relevant to: All FCP TBT LCP CLS

OPPORTUNITIES

Opportunity	Estimated Savings
Minify JavaScript	0.15 s

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.

URL	Transfer size	Potential savings
`chrome-extension://dagdlcijhfbmgkjokkjicnnfimlebcll/page_context.js`	10.6 KiB	5.0 KiB

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

DIAGNOSTICS

▲ Registers an `unload` listener

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

Source
(unknown)

▲ Minimise main-thread work — 6.6 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	6,150 ms
Other	196 ms
Script Parsing & Compilation	133 ms
Garbage Collection	102 ms
Style & Layout	10 ms
Parse HTML & CSS	5 ms

Category	Time Spent
Rendering	5 ms

▲ Reduce JavaScript execution time — 6.2 s

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

URL	Total CPU Time	Script Evaluation	Script Parse
/bundle.js (localhost)	6,078 ms	5,974 ms	12 ms
http://localhost:1234	176 ms	40 ms	42 ms
`chrome-extension://bnjjngeaknajbdcgpfkgnonkmififhfo/build/content-script.js`	127 ms	64 ms	59 ms
`Unattributable`	109 ms	5 ms	0 ms
`chrome-extension://kbfnbcaeplbcioakkpcpgfkobkghlhen/src/js/Grammarly-check.js`	57 ms	36 ms	16 ms

▲ Serve static assets with an efficient cache policy — 4 resources found

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

URL	Cache TTL	Transfer size
…small/pezza.jpg (storage.googleapis.com)	1 h	62 KiB
…small/fesh.jpg (storage.googleapis.com)	1 h	48 KiB
…small/soop.jpg (storage.googleapis.com)	1 h	40 KiB
…small/brrto.jpg (storage.googleapis.com)	1 h	32 KiB

○ 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 more. FCP LCP

Maximum critical path latency: **40 ms**

Initial Navigation

http://localhost:1234

/bundle.js (localhost) **- 20 ms, 67.59 KiB**

User Timing marks and measures — 2 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
@grammarly-extension:checkScriptInitStart	Mark	1,631.33 ms	
@grammarly-extension:checkScriptInitEnd	Mark	1,634.87 ms	

Keep request counts low and transfer sizes small — 8 requests • 263 KiB

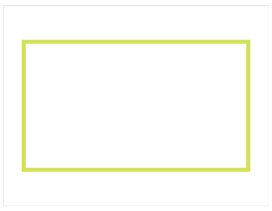
To set budgets for the quantity and size of page resources, add a budget.json file. Learn more.

Resource type	Requests	Transfer size
Total	8	263.3 KiB
Image	4	181.6 KiB
Script	2	78.1 KiB
Stylesheet	1	2.8 KiB
Document	1	0.7 KiB
Media	0	0.0 KiB
Font	0	0.0 KiB
Other	0	0.0 KiB

Resource type	Requests	Transfer size
Third-party	6	195.0 KiB

○ Largest contentful paint element — 1 element found

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

Element	
	div.jss69.jss67

○ 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

URL	Start Time	Duration
/bundle.js (localhost)	1,862 ms	6,078 ms
`chrome-extension://bnjjngeaknajbdcgpfkgnonkmififhfo/build/content-script.js`	730 ms	104 ms
http://localhost:1234	657 ms	71 ms
http://localhost:1234	606 ms	51 ms

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

PASSED AUDITS (30)

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 more. FCP LCP

Properly size images

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

Defer off-screen 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.

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

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

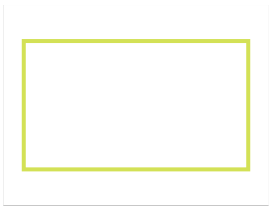
If you are not server-side rendering, split your JavaScript bundles with `React.lazy()`. Otherwise, code-split using a third-party library such as loadable-components.

Efficiently encode images

Optimised images load faster and consume less mobile data. Learn more.

Serve images in next-gen formats — Potential savings of 87 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.

		URL	Resource size	Potential savings
	div.js s69.j ss67	…small/pezza.jpg (storage.googleapis.com)	62.3 KiB	28.7 KiB
	div.js s69.j	…small/fesh.jpg (storage.googleapis.com)	47.4 KiB	22.8 KiB

	URL	Resource size	Potential savings
ss67			
div.js s69.j ss67	…small/soop.jpg (storage.googleapis.com)	39.8 KiB	19.7 KiB
div.js s69.j ss67	…small/brrto.jpg (storage.googleapis.com)	31.8 KiB	16.1 KiB

Enable text compression

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

Initial server response time was short — Root document took 10 ms

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.

URL	Time Spent
http://localhost:1234	10 ms

Avoid multiple page redirects

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

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

Pre-load key requests

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

Use HTTP/2

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

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

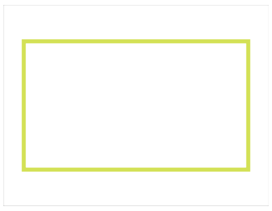
Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB

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 delivered to modern browsers, while retaining support for legacy browsers. Learn more TBT

URL		Potential savings
/bundle.js (localhost)		0.1 KiB
bundle.js:12	`@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

		URL	Potential savings
	div.jss 69.jss 67	…small/fesh.jpg (storage.googleapis.com)	0 ms

Avoids enormous network payloads — Total size was 263 KiB

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

✓ Show 3rd-party resources (4)

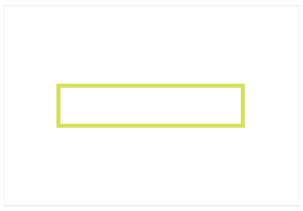
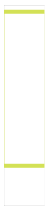
URL	Transfer size
/bundle.js (localhost)	67.6 KiB
…small/pezza.jpg (storage.googleapis.com)	62.4 KiB
…small/fesh.jpg (storage.googleapis.com)	47.5 KiB
…small/soop.jpg (storage.googleapis.com)	39.9 KiB
…small/brrto.jpg (storage.googleapis.com)	31.9 KiB
`chrome-extension://dagdlcijhfbmgkjokkjicnnfimlebcll/page_context.js`	10.6 KiB
`chrome-extension://dagdlcijhfbmgkjokkjicnnfimlebcll/style.css`	2.8 KiB
http://localhost:1234	0.7 KiB

Avoids an excessive DOM size — 49 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 minimise the number of DOM nodes created if you are rendering many repeated elements on the page. Learn more. Also, minimise 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		49

Statistic	Element	Value
Maximum DOM Depth	 span.jss75	9
Maximum Child Elements	 main.jss64	4

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

Minimise 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 more. TBT

Third-party	Transfer size	Main-thread blocking time
Other Google APIs/SDKs	182 KiB	0 ms
…small/pezza.jpg (storage.googleapis.com)	62 KiB	0 ms
…small/fesh.jpg (storage.googleapis.com)	48 KiB	0 ms
…small/soop.jpg (storage.googleapis.com)	40 KiB	0 ms
…small/brrto.jpg (storage.googleapis.com)	32 KiB	0 ms

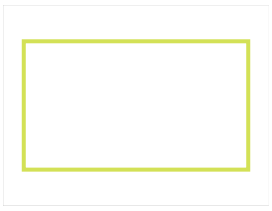
○ Lazy load third-party resources with facades

Some third-party embeds can be lazy loaded. Consider replacing them with a facade until they are required. Learn more. 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.

Element



div.jss69.jss67

○ 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.

○ Avoid non-composited animations

Animations which are not composited can be poor, slow and increase CLS. Learn more 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 more 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. TBT

■ Captured at 26 Mar 2023, 09:58 CEST
■ Initial page load
■ Emulated Moto G4 with Lighthouse 9.6.8
■ Slow 4G throttling
■ Single page load
■ Using Chromium 110.0.0.0 with devtools

Generated by **Lighthouse** 9.6.8 | File an issue