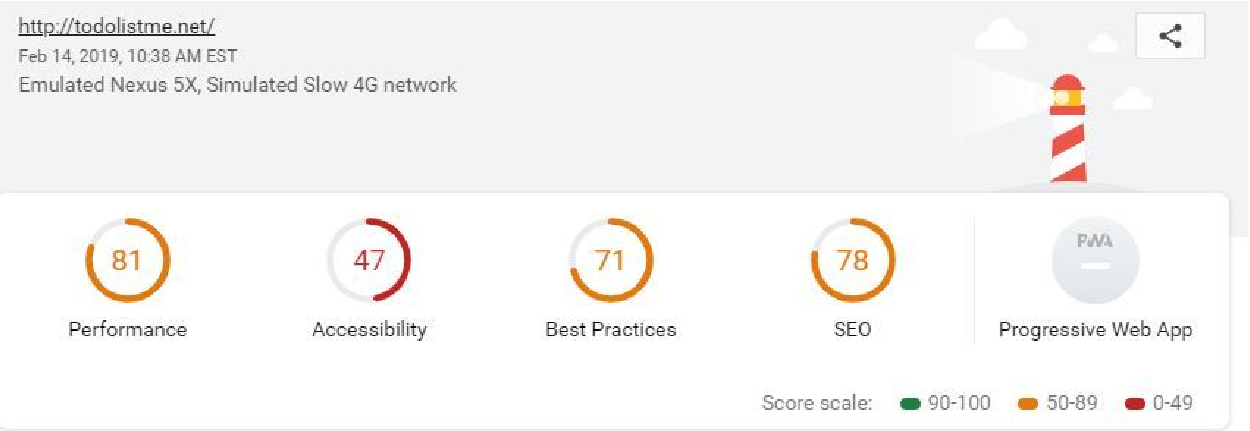
Performance Audit

Overview

This audit will focus on the performance of two “To-do” list apps using Google’s Lighthouse Audit Tool. Both apps function to allow a user to create, edit, and delete items on a list.

Todolistme



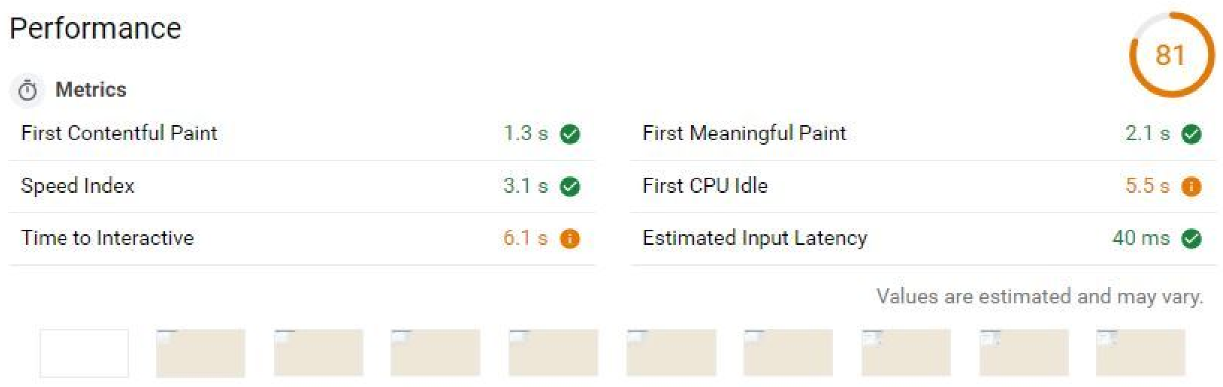
http://todolistme.net/
Feb 14, 2019, 10:38 AM EST
Emulated Nexus 5X, Simulated Slow 4G network

81 Performance
47 Accessibility
71 Best Practices
78 SEO
PWA Progressive Web App

Score scale: 🟩 90-100 🟧 50-89 🟥 0-49

Todolistme exhibits a score of 81 out of 100 for Performance. Performance Metrics are estimates that may vary, but they are nonetheless a good indication of how the site is running. The Audit tool shows the gravity of this score using color. Todolistme is within the orange range, which is not terrible but has room for improvement. Using the tool, we will break down the Performance score by looking at the provided Metrics, Opportunities, and Diagnostics.

Metrics



Performance

81

Metrics

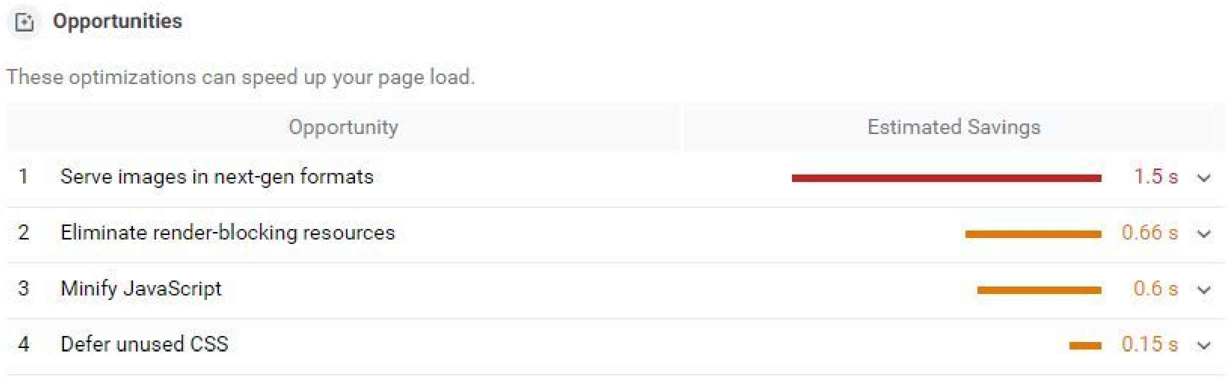
First Contentful Paint	1.3 s ✓	First Meaningful Paint	2.1 s ✓
Speed Index	3.1 s ✓	First CPU Idle	5.5 s !
Time to Interactive	6.1 s !	Estimated Input Latency	40 ms ✓

Values are estimated and may vary.

Most of these Metric scores are satisfactory, but the First CPU Idle score and Time to Interactive score are not great. The page's visuals load quickly, but there is a significant delay before the page is ready to handle initial input. Once that input is able to be handled and accepted, the response afterwards is quick.

Opportunities

Lighthouse provides guidance to help optimize scores that are less than ideal. These Opportunities also provide "Estimated Savings."



Opportunities

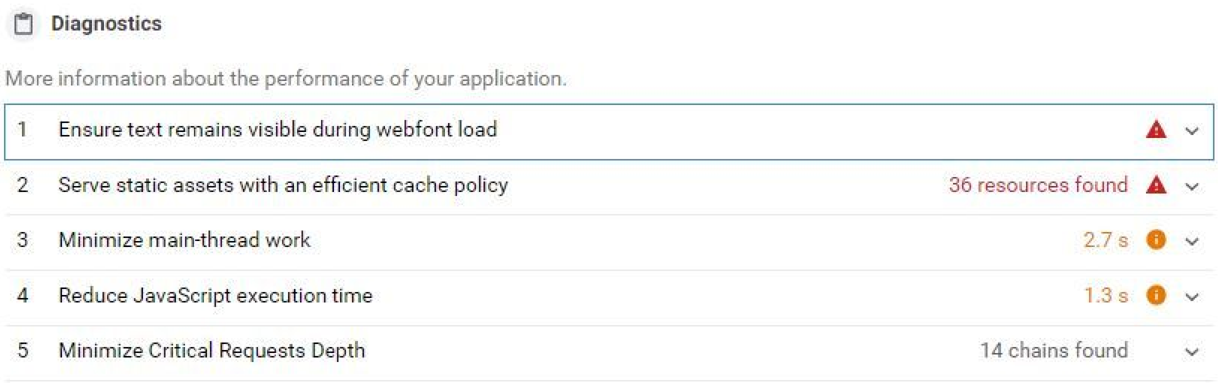
These optimizations can speed up your page load.

	Opportunity	Estimated Savings
1	Serve images in next-gen formats	1.5 s
2	Eliminate render-blocking resources	0.66 s
3	Minify JavaScript	0.6 s
4	Defer unused CSS	0.15 s

There is one very large image texture.png that is delaying response time as it loads. Changing the format of the image to JPEG 200 or WebP could greatly reduce this delay. However, not all browsers are compatible with this format choice. If browser compatibility is a priority, then putting the image through an optimizer like tinypng.com could help with this delay.

Resources that the site relies on are taking time to load. Specifically, Todolistme is using an un-minified version of jquery; simply switching to the optimized version would help the site load faster, along with other CSS resources (fonts and styles). Once all CSS and JS has been minified, the site also needs to 'Defer unused CSS' by inlining what is considered Critical CSS (the minimum CSS that is needed to load the page).

Diagnostics



Diagnostics

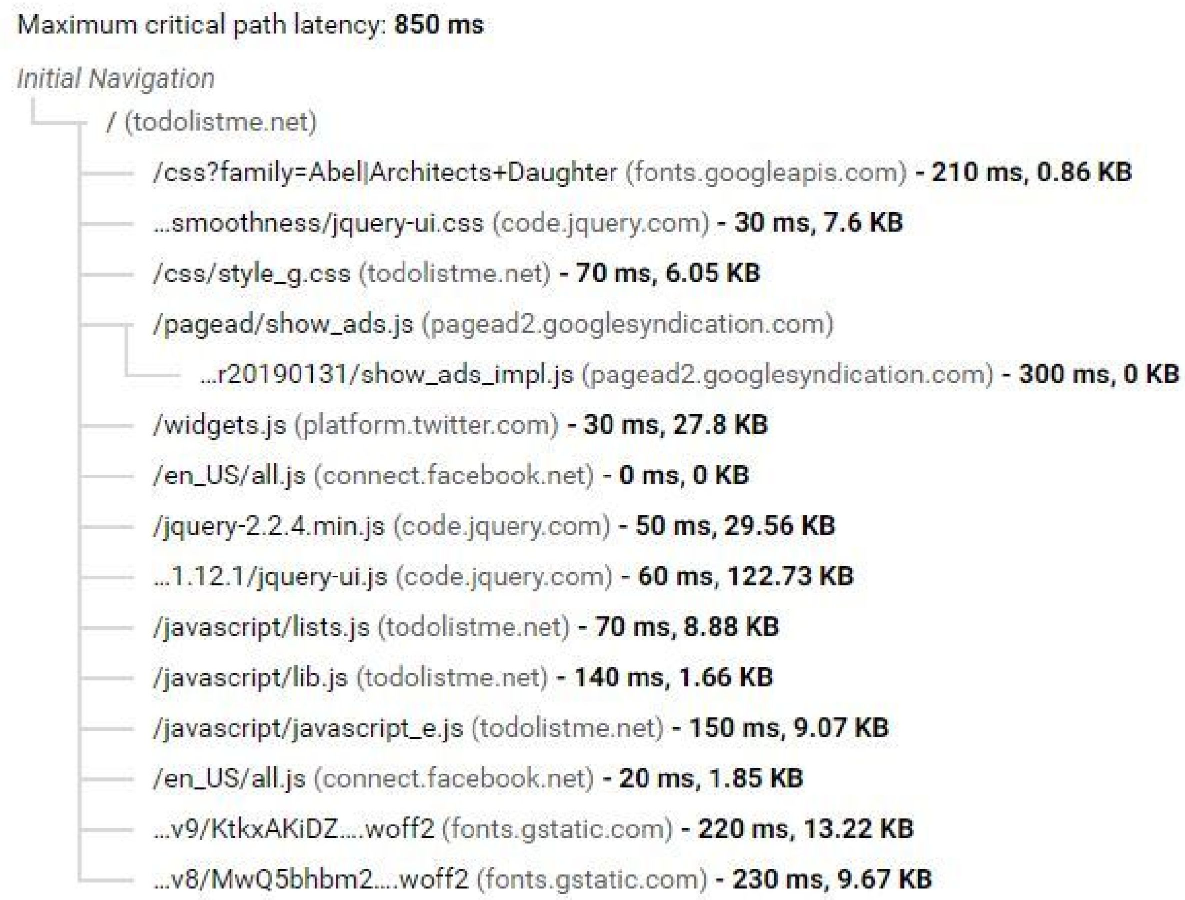
More information about the performance of your application.

1	Ensure text remains visible during webfont load		▲
2	Serve static assets with an efficient cache policy	36 resources found	▲
3	Minimize main-thread work	2.7 s	●
4	Reduce JavaScript execution time	1.3 s	●
5	Minimize Critical Requests Depth	14 chains found	

Todolistme uses a Google web font ("Architect's Daughter") that is being loaded via CSS. This font gives a fun handwritten look for the app, but is a heavy resource in the initial site load. To avoid this delay in load time, the site should implement the use of the font-display declaration. This will help control how a font displays on the site visually (and immediately) while the actual font is being downloaded. This way, a visitor can still start seeing text before the actual font is rendered.

When it comes to cache, this site is not very efficient. All fonts, images, stylesheets, and scripts should be cached and given a long shelf time. In detail, the report flags the aforementioned texture.png as None for Cache TTL (time to live) among 36 other resources that should be cached or cached for a longer amount of time.

Todolistme's JavaScript is having a rough impact on load times demonstrated by the 'Minimize main-thread work' and 'Reduce JavaScript execution time' scores. Long-running JavaScript needs to be moved off the main-thread and given to Web Workers. Using a web worker will help you control when the JavaScript is being used and loaded instead of clumped together with everything else on the site that is trying to load which will result in missed frames.

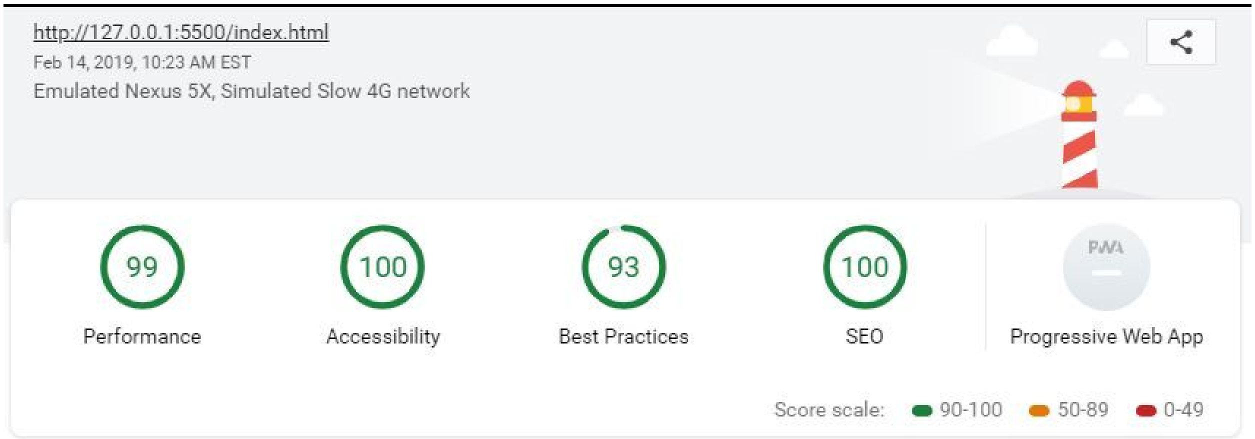


```
Maximum critical path latency: 850 ms

Initial Navigation
└── / (todolistme.net)
    ├── /css?family=Abel|Architects+Daughter (fonts.googleapis.com) - 210 ms, 0.86 KB
    ├── …smoothness/jquery-ui.css (code.jquery.com) - 30 ms, 7.6 KB
    ├── /css/style_g.css (todolistme.net) - 70 ms, 6.05 KB
    ├── /pagead/show_ads.js (pagead2.googlesyndication.com)
    │   └── …r20190131/show_ads_impl.js (pagead2.googlesyndication.com) - 300 ms, 0 KB
    ├── /widgets.js (platform.twitter.com) - 30 ms, 27.8 KB
    ├── /en_US/all.js (connect.facebook.net) - 0 ms, 0 KB
    ├── /jquery-2.2.4.min.js (code.jquery.com) - 50 ms, 29.56 KB
    ├── …1.12.1/jquery-ui.js (code.jquery.com) - 60 ms, 122.73 KB
    ├── /javascript/lists.js (todolistme.net) - 70 ms, 8.88 KB
    ├── /javascript/lib.js (todolistme.net) - 140 ms, 1.66 KB
    ├── /javascript/javascript_e.js (todolistme.net) - 150 ms, 9.07 KB
    ├── /en_US/all.js (connect.facebook.net) - 20 ms, 1.85 KB
    ├── …v9/KtkxAKiDZ….woff2 (fonts.gstatic.com) - 220 ms, 13.22 KB
    └── …v8/MwQ5bhbm2….woff2 (fonts.gstatic.com) - 230 ms, 9.67 KB
```

Todolistme’s largest critical path vampire is the use of ads. These ads are causing a 300ms latency just by themselves. Sources on this diagram should either be eliminated if they are not necessary or deferred. By minifying the jquery file and removing the Google Ad Service, there would be a dramatic improvement on Performance.

To-do list app



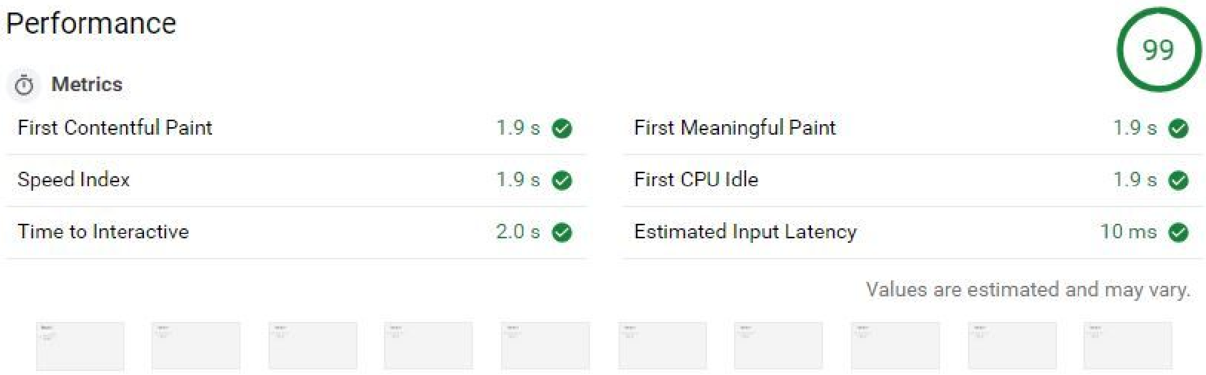
http://127.0.0.1:5500/index.html
Feb 14, 2019, 10:23 AM EST
Emulated Nexus 5X, Simulated Slow 4G network

Performance	Accessibility	Best Practices	SEO	Progressive Web App
99	100	93	100	PWA –

Score scale: ● 90-100 ● 50-89 ● 0-49

The To-do list app boast an impressive Performance score of 99 out of 100, shown in green. It should be taken into consideration this app is minimal compared to Todolistme. Not as many resources are needed, which shows less IS more, cause in the end it is about user friendly apps that function quickly and effectively.

Metrics



Performance 99

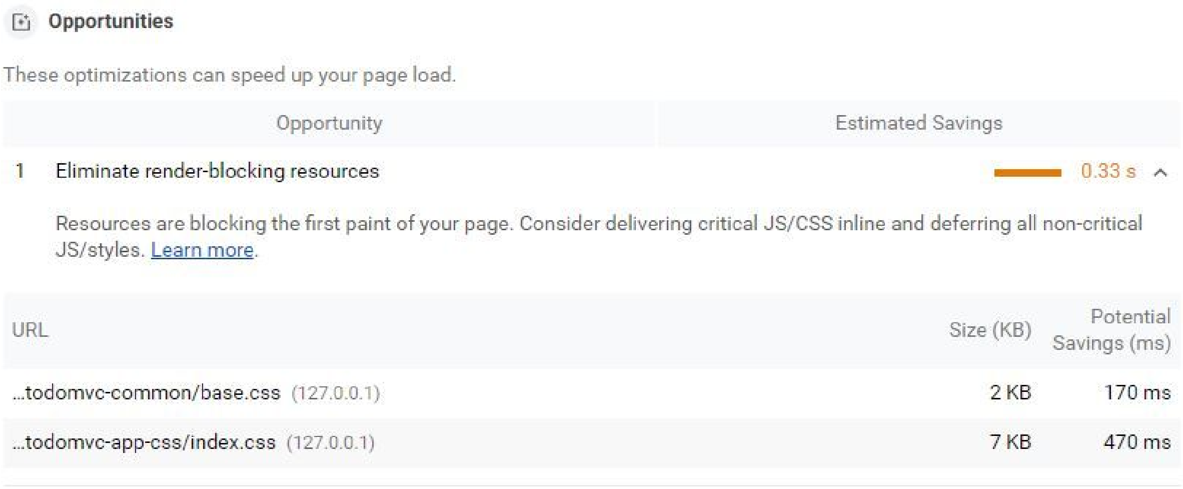
Metrics

First Contentful Paint	1.9 s ✓	First Meaningful Paint	1.9 s ✓
Speed Index	1.9 s ✓	First CPU Idle	1.9 s ✓
Time to Interactive	2.0 s ✓	Estimated Input Latency	10 ms ✓

Values are estimated and may vary.

From Contentful Paint to Estimated Input Latency performance scores excel. The site loads quickly and information can be inputted and handled immediately. The previous app took almost 5 seconds to finally handle user input. To-do list app only takes 1.9s!

Opportunities



Opportunities

These optimizations can speed up your page load.

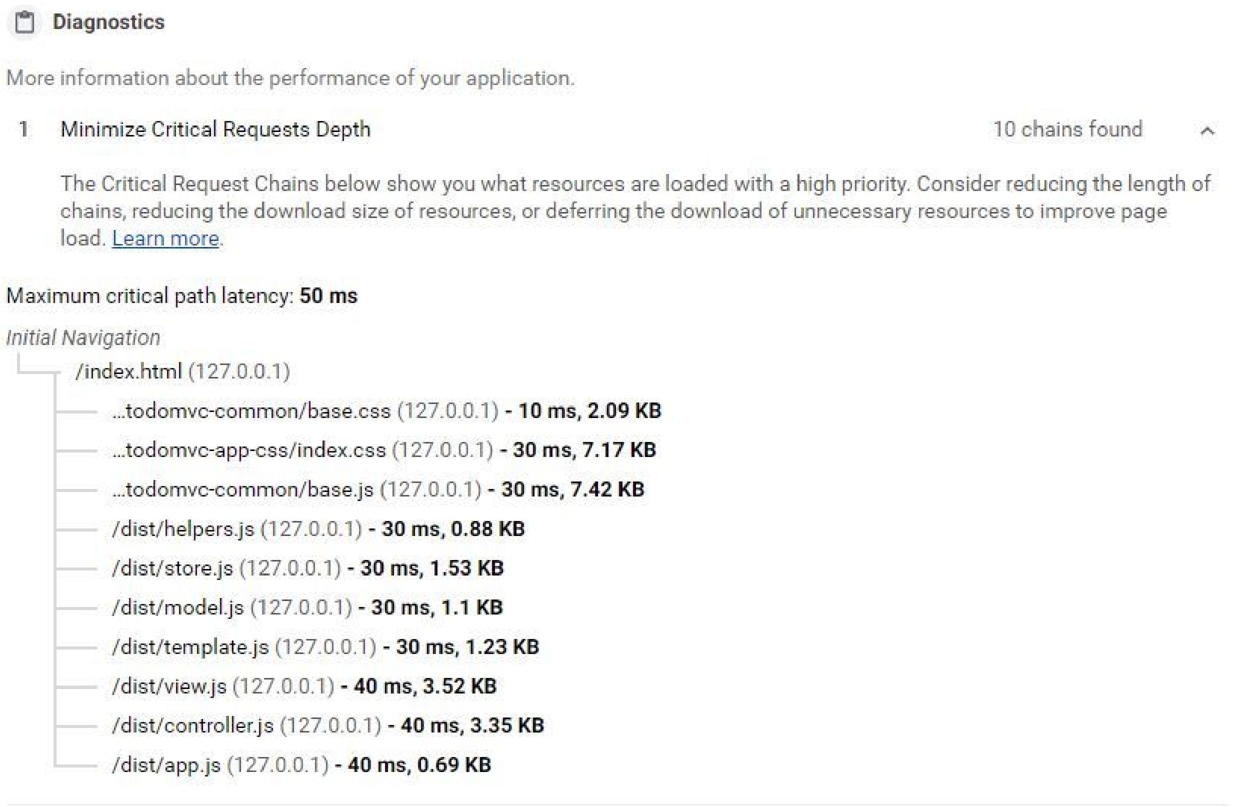
	Opportunity	Estimated Savings
1	Eliminate render-blocking resources	0.33 s

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn more.

URL	Size (KB)	Potential Savings (ms)
…todomvc-common/base.css (127.0.0.1)	2 KB	170 ms
…todomvc-app-css/index.css (127.0.0.1)	7 KB	470 ms

Since this app has minimal resources there is less room for error. However this Performance audit shows that two css files could be minified to eliminate render-block and load the site even quicker.

Diagnostics



Diagnostics

More information about the performance of your application.

1 Minimize Critical Requests Depth — 10 chains 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.

Maximum critical path latency: **50 ms**

Initial Navigation

/index.html (127.0.0.1)

- ...todomvc-common/base.css (127.0.0.1) **- 10 ms, 2.09 KB**
- ...todomvc-app-css/index.css (127.0.0.1) **- 30 ms, 7.17 KB**
- ...todomvc-common/base.js (127.0.0.1) **- 30 ms, 7.42 KB**
- /dist/helpers.js (127.0.0.1) **- 30 ms, 0.88 KB**
- /dist/store.js (127.0.0.1) **- 30 ms, 1.53 KB**
- /dist/model.js (127.0.0.1) **- 30 ms, 1.1 KB**
- /dist/template.js (127.0.0.1) **- 30 ms, 1.23 KB**
- /dist/view.js (127.0.0.1) **- 40 ms, 3.52 KB**
- /dist/controller.js (127.0.0.1) **- 40 ms, 3.35 KB**
- /dist/app.js (127.0.0.1) **- 40 ms, 0.69 KB**

The 'Maximmum critical path latency' is 50ms. Resources that are used are running efficiently and quickly. Minifying the CSS files mentioned before will accelerate those times and less bytes used to download.