AUDIT PERFORMANCE DOCUMENT

App: Todos App

Competitor App: todolistme.net

Date: December 17, 2019

Document Version: 1.0

SUMMARY

A Performance Audit of the Todos App and a Competitor App (todolistme.net) was carried out using Chrome Dev. Tools and compared.

TESTING ENVIRONMENT

Operating System: Windows 10 (Version 1809)

Browser: Version 79.0.3945.79 (Official Build) (64-bit)

PERFORMANCE METRICS

First Contentful Paint(FCP)

First Contentful Paint marks the time at which the first text or image is painted.

FCP time (in seconds)

0–2 (fast)

2–4 (average)

Over 4 (slow)

First Meaningful Paint(FMP)

First Meaningful Paint measures when the primary content of a page is visible.

FMP time (in seconds)

0–2 (fast)

2–4 (average)

Over 4 (slow)

Speed Index

Speed Index shows how quickly the contents of a page are visibly populated.

Speed Index metric(in seconds)

0–4.3 (fast)

4.4–5.8 (average)

Over 5.8 (slow)

First CPU Idle

First CPU Idle marks the first time at which the page's main thread is quiet enough to handle input.

First CPU Idle metric (in seconds)

0–4.7 (fast)

4.8–6.5 (average)

Over 6.5 (slow)

Time to Interactive(TTI)

Time to interactive is the amount of time it takes for the page to become fully interactive.

TTI metric (in seconds)

0–5.2 (fast)

5.3–7.3 (average)

Over 7.3 (slow)

Max Potential First Input Delay

The maximum potential First Input Delay that your users could experience is the duration, in milliseconds, of the longest task.

RESULTS

todolistme.net

No Throttling			todolistme.net		
First Contentful Paint	**First Meaningful Paint**	**Speed Index**	**First CPU Idle**	**Time to interactive**	**Max Potential First Input Delay**
2.6s	3.3s	3.3s	6.4s	6.4s	300ms

Throttling 4x CPU Slowdown			todolistme.net		
First Contentful Paint	**First Meaningful Paint**	**Speed Index**	**First CPU Idle**	**Time to interactive**	**Max Potential First Input Delay**
2.8s	2.9s	9.8s	10.9s	12.2s	860ms

Runtime Performance (No Throttling)		todolistme.net	
	Total Time	**Scripting**	**Rendering**
Adding Todos	166ms	125ms	21ms
Deleting Todos	82ms	60ms	21ms

Runtime Performance (Throttling 4x CPU Slowdown)		todolistme.net	
	Total Time	**Scripting**	**Rendering**
Adding Todos	288ms	268ms	11ms
Deleting Todos	227ms	152ms	74ms

Todos-App

Throttling 4x CPU Slowdown			Todos-App		
First Contentful Paint	**First Meaningful Paint**	**Speed Index**	**First CPU Idle**	**Time to interactive**	**Max Potential First Input Delay**
1.9s	1.9s	1.9s	1.9s	1.9s	23.33ms

No Throttling			Todos-App		
First Contentful Paint	**First Meaningful Paint**	**Speed Index**	**First CPU Idle**	**Time to interactive**	**Max Potential First Input Delay**
0.56ms	0.56ms	0.46ms	0.56ms	0.56ms	20ms

Runtime Performance (No Throttling)		Todos-App	
	Total Time	**Scripting**	**Rendering**
Adding Todos	21ms	5ms	1ms
Deleting Todos	32ms	7ms	0ms

Runtime Performance (Throttling 4x CPU Slowdown)		Todos-App	
	Total Time	**Scripting**	**Rendering**
Adding Todos	129ms	31ms	3ms
Deleting Todos	121ms	34ms	2ms

RECOMMENDATIONS

1. 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.

2. Preconnect to required origins. Consider adding `preconnect` or `dns-prefetch` resource hints to establish early connections to important third-party origins

3. Eliminate render-blocking resources. Todolistme.net has resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles

4. Reduce the impact of third-party code. Todolistme.net has third-party code that blocked the main thread for **4,150ms.**

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. In this case **Google/Doubleclick Ads** had the most impact with a **size of 223KB** and a **Main-Thread Blocking Time of 1,460ms**.

Other recommendations to remove performance Include:

1. Properly size images. Continue with using SVG for complex icons where possible
2. Defer off-screen images. Consider lazy-loading these images after all critical resources have finished loading to lower Time to Interactive.
3. Minify CSS
4. Minify JavaScript
5. Remove unused CSS