Todo List Perfomance Analysis

The goal of this performance analysis is to compare the todo app with a competitor in order to understand how to improve our todo list app should we decide to scale. The analysis of the two todo apps will be completed using chrome's developer tool's auditing feature called lighthouse.

Glossary of Terms Describing App Performance

Overall Performance: A measurement of the app's performance, with a specific focus on page load performance, based on a series of metrics intended to quantify the level of expected user satisfaction.

First Contentful Paint: The time, measured in seconds, at which the first piece of content appears visible to the user when the page is initially loading.

Speed Index: The time, measured in seconds, for the page's contents to be visible as a page loads.

Largest Contentful Paint: The time, measured in seconds, for the largest element, either text or an image to be visible to the user as the page loads.

Time To Interactive: The time, measured in seconds, for all of the various features and functions of the page to be useable when the page is loading.

Total Blocking Time: The sum of the time, measured in milliseconds, of all the features of an app to show functionality more then 50 milliseconds after an event (ie click, pressing enter, ect). Thus, if after a click the function takes 80 milliseconds to execute then the total blocking time for that one feature will be 80 - 50 which is 30 milliseconds.

Cumulative Layout Shift: A score which references the amount of movement various elements on a page will experience as a page loads. For example, other elements loading can cause already loaded elements to move causing a greater cumulative layout shift.

Competitor's Todo List App

audit link:
https://github.com/mbdev95/OpenClassrooms_Project_8_Debugging_todoListApp/blob/master/Lighthouse_Audits/Competitor%20Todo%20List%20Audit.pdf

Competitor Todo List App Metrics

Performance

Metrics			
First Contentful Paint	0.9 s	▲ Time to Interactive	7.8 s
▲ Speed Index	3.0 s	▲ Total Blocking Time	4,300 ms
Largest Contentful Paint	1.1 s	Cumulative Layout Shift	0.054

Performance Issues

- The use of http/1 increases the load time by approximately 0.34s.

- Vulnerable to security issues due to not using https.

- The use of unused 3rd party resources such as unused google api's, unused fonts or jQuery files are causing longer loading times.

- The use of document.write is not professional and reduces load times.

- Many images do not have a specified width and height which can increase cumulative layout shifts.

- Text does not appear while fonts are loading.

- The advertisments block the progress of the other content being loaded for 1095ms causing a delay in loading the app's content.

Improvements

- http/2 should be used instead of http/1 since http/2 takes advantage of binary headers and multiplexing to increase the load time by approximately 0.34s.

- Should use https to improve security.

- Only 3rd party resources which are being used should be kept in order to make a more succinct application which loads faster.

- Instead of using document.write to add content to the html use the JavaScript DOM to select and add content to the selected elements to ensure faster load times.

- Establish a specified width and height in order to reduce cumulative layout shifts.

- Allow text to appear in generic font until the actual fonts have loaded.

- The advertisments need to be loaded last and the web content first to shorten the time in which the web page becomes fully visible to the user. Also, the use of i-frames for ads should be avoided as they increase the load time of the ad. An alternative could be to use the quicker Ajax Api calls to retrieve the ads.

Additional Improvements Regarding Accessibility and Best Practices

- Increase the contrast of the completed list items' text so the completed list items will be more readable.

- Image elements should have alt attributes and frame elements should have labels so those using screen readers are able to better understand the images and frame elements purpose.

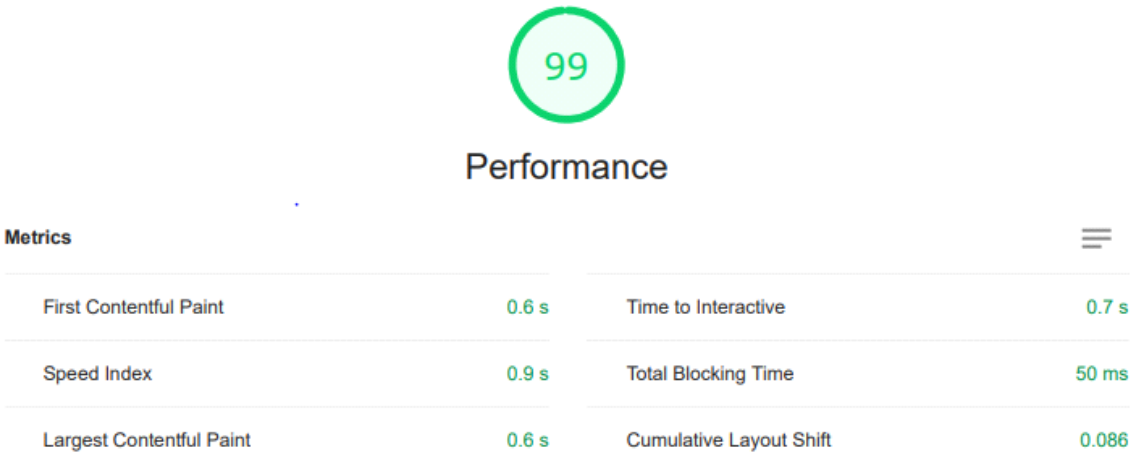
- Include a language attribute to alert the screen reader to the verbal language the application is using so the screen reader reads the application's text correctly.

***

Todo List App

audit link:
https://github.com/mbdev95/OpenClassrooms_Project_8_Debugging_todoListApp/blob/master/Lighthouse_Audits/Todo%20List%20Audit.pdf

Todo List App Metrics



Performance Issues

- The size of some JavaScript and CSS files could be reduced by minifying them in order to decrease page load time.

-Excess time spent parsing, compiling, and executing JavaScript files.

- The use of outdated code increases the amount of code the browser has to process increasing load times.

Performance Improvements

- Certain elements such as ::before and footer.info could be altered to reduce cumulative layout shift.

- Smaller JavaScript files to parse and execute will lead to quicker load times.

- Could create specific goals for the important metrics of the todo list application's performance, and take advantage of lighthouse for performance budgets to determine if the applications metrics meet those goals.

-The 12 total requests could be reduced to increase load time.

- The JavaScript files should be loaded with the JavaScript files having the most important functionality loaded first and less important files loaded last in order to reduce the time a user is unable to use the most important functionality of the website. For instance, controller.js should be loaded before helpers.js.

- Modernize the JavaScript code to make it more efficient for the browser to process.

Addition Improvements Regarding Accessibility and Best Practices

- Form input elements should have label elements to better explain the input elements function for those using screen readers.

Conclusion

The competitor's todo app could make numerous changes to decrease load times and make the application more useable. Our todo app by comparison has faster loading times, but there is still areas for improvement such as reducing the size of our JavaScript and CSS files. Moreover, our application's user interface is more readable and understandable but lacks some of the options of our competitor's app such as the ability to create multiple lists.