Performance audit

This audit compares the performance and features' differences between the todolistme.net and our website, as well as optimisations' suggestions that could be applied when scaling up our app.

Features

Both solutions have same purpose of adding, completing and deleting things to be done in the form of list. Nonetheless, todolistme.net offers wider range of functionalities, including:

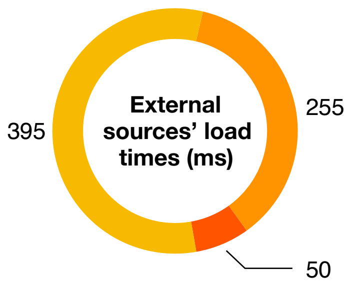
- ability to create new lists,
- categorising lists,
- opening a list in separate window (acts as distraction free mode),
- support for drag-and-drop to arrange, complete, and un-complete todo items,
- postponing todo items to be shown on the list the next or on a specific day.

Performance

Opening competitor's website takes 3.3 seconds to be fully loaded, whereas our solution takes only 0.5s to be fully loaded (6.6x slower).

External sources

- **Advertisements -** Depending on the audience size and behaviour (whether they use ad-blocking software or not), Google Ads service ensures the competitor's website is partly or fully covered financially or even profitable, by displaying advertisements on the screen. On the other hand, it directly affects the load times, as it takes almost 255ms to load and process external scripts, another 50ms to download the advertisement itself and another 90ms to render it on the screen. It does not only affect the performance, but also user experience.



- **Analytical tools -** Google Analytics is another tool that directly affects the performance, as it takes almost 255ms to load and process the script. Nevertheless, it might be treated as exception to the rule, as it may provide data that is relevant to determine traffic, audience's behaviour, and be used to scale up the app.

- **Custom font faces -** The competitor uses two external type-faces, that extend the website load by not much more than 50ms, which is a very good result. It's a sensible feature that makes the site unique, but at the same time it is optimised, as only selected type-faces are downloaded, not entire font-faces (collections).

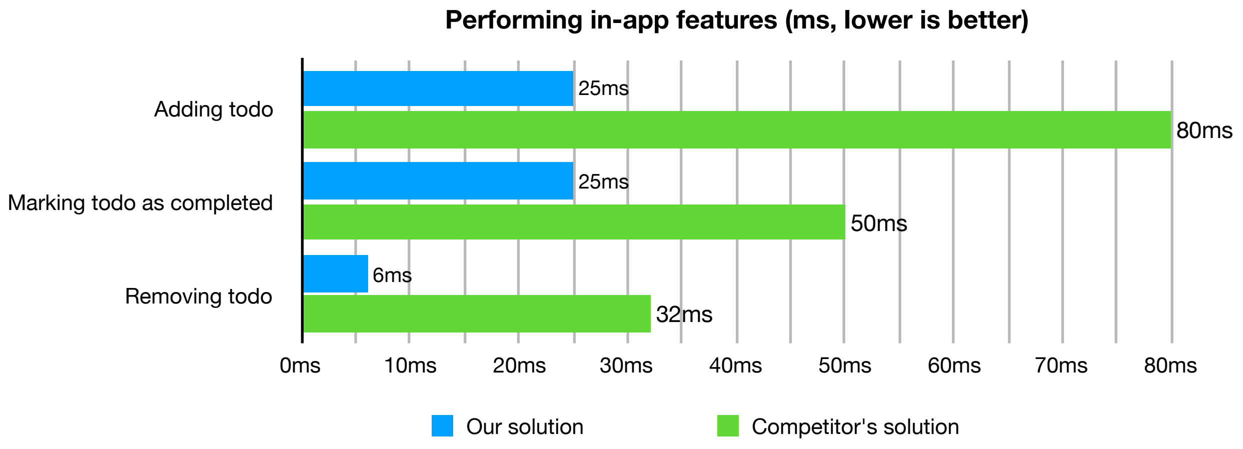
In-app functions

- **Adding todo**
  Adding a new item to the list on the competitor's website takes approximately 80ms, 40ms to run *add_todo* function and another 40ms to render the changes on the screen. This excludes the animation duration of 190ms.

  On the other hand, adding a new item to the list in our solution takes only 25ms, 20ms to run Controller.*addItem, Model.create* functions and 5ms to render the differences on the screen. Even if added 20ms duration of potential animation, our solution is 1.7x faster, without animation, it is 3.2x faster.

- **Marking todo as completed**
  Marking a todo item as completed on the competitor's site, takes approximately 50ms, 10ms to execute *finish_todo* function, and another 40ms (including animation duration) to render the changes on the screen.
  Our solution perform this function in 25ms, 21ms to run *Controller.toggleComplete, Model.update* functions. Another 4ms are used to update the DOM. It's 2x faster than the competitor's website.

- **Removing todo**
  Removing a todo takes 32ms on the competitor's website, 11ms to execute *purge_done_items* function and reaming 21ms to update the DOM.
  Our solution, perform this function in just 6ms, 2ms to execute *Controller.removeItem* and *Model.remove* functions, and reaming 4ms to update the DOM. This makes our approach 5.3x faster than the competitor.



Performance optimisations when scaling up

- **Variables** - it is strongly recommended to use `let` especially in functions that run a `for` loop within a `setTimeout`. This will prevent the `i` value not incrementing or decrementing. On the other hand, in some cases, `const` may well provide an opportunity for optimisation that `var` wouldn't, especially for global variables. With `const`, we're explicitly telling the engine that the value cannot change. So it's free to do any optimisation it wants, including emitting a literal instead of a variable reference to code using it, knowing that the values cannot be changed.

- **Existing codebase** - the current state of app's codebase is resistant for future upgrades, and so it is necessary to use it, when scaling up the app, wherever possible. If you were to allow the users to create multiple todo lists for organisation purposes, it is recommended to use already defined methods. For instance, you should define `Store` instance for each list, which will provide you access to the cache manipulation, thus the ability to add, modify and remove items from the list. The use of existing codebase will prevent repetitive code and functions that have the same purpose.

- **MVC** - it is recommended to follow the MVC pattern that is currently used in the app codebase, to make the source readable and optimised to other developers. One change that could benefit the app with better performance is to use ES6 *template literals*, as modern browser's perform them between 2-40% faster than standard string concatenation. At the top of that, template literals are more readable and will shorten the code, especially when using `String.replace` to replace the variables names from the string.