Todo-list-app documentation

1 . Todo-list-app Brief

Todo-list-app is an application for tasks list management. This app allows us to add a new task to todo-list, update the task, delete the task, and change the state of the task to completed or active. This app build with minimalist design and basic todo-list-app functionality.

This app is created using MVC design pattern in Javascript which separates data logic (Model), display functionality (View), and connect them with Controller.

Model : contains local storage object and prototypes function to manage (create, update, remove, count) it for todolist-app tasks data in browser. It defines data management of the app.

View : contains DOM element and its structure manipulation functionality that represent displays and user interactions.

Controller : contains prototypes function for controlling interactions between Model and View. Controller functions do listens to user actions from view, converting input from View, then calling Model to perform requested operations, and calling View to display results from Model.

2. Program Specification

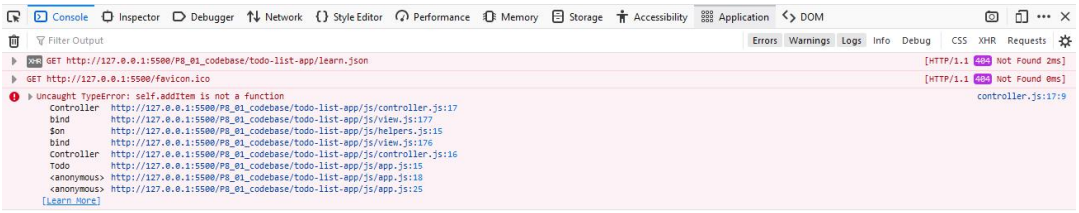
- index.html is the entry point of this todomvc application.
- Index.css is the css styles of this todomvc application.
- Base.css is the common style of this todomvc application.
- App.js creates the instance of todo application and initiate object needed like the storage, Model, View, and Controller Object.
- Store.js creates the data storage object using local session storage and manages it with functions.
- Helper.js contains helper functions for query selector, event listener wrapper, and handler attach or delegate.
- Template.js contains template function to escape characters, display list items, change button states.
- Controller.js creates controller objects that connect interaction between Model and View with its prototypes such as :

- setView to loads and initialize the view,
- showAll to displays all items in the todo-list,
- showActive to renders uncompleted tasks,
- showCompleted to renders completed tasks,
- addItem to creates new todo task, saving it in the local storage by adding ID,
- editItem to starts editing mode of todo task by matching with the correct ID,
- editItemSave to successfully edits item and save the changing by using matched ID,
- editItemCancel to cancels the item editing mode,
- removeItem to removes item from to-do-list and storage by using its ID as a parameter,
- removeCompletedItems to removes all completed tasks,
- toggleComplete to gives ID and updates the state of completeness of task in the storage,
- toggleAll to change the state of completeness of the tasks: on/off

- Model.js to create model objects and connects it with storage object and provide prototypes such as :
  - create to creates a new todo model and saves it in the storage,
  - read to finds and returns a model in storage, if the query isn't given, returns everything,
  - update to updates a model, every action based on unique ID,
  - remove to removes a model from storage,
  - removeAll to removes all data from storage,
  - getCount to counting active, completed and total tasks by finding the in the storage.

- View.js to create View Object to manipulate DOM attached to user interaction such as :
  - bind to takes a todo application event and registers the handler,
  - render to renders the given command with the options.

3. Manual Bugs Fix



1. Typo mistake:
   - Controller.prototype.adddItem at js/controller.js(line 95)
     Fixed become :
     - Controller.prototype.addItem
2. Missing learn.json file when base.js try to get file on line 248

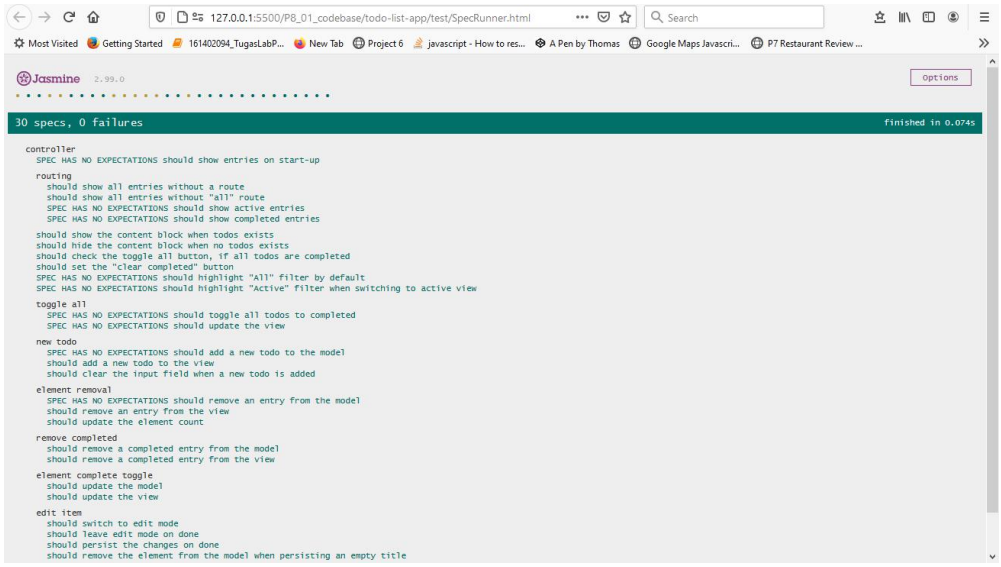
```
247     redirect();
248     getFile('learn.json', Learn);
249 })();
250
```

   Fixed become :

- Create learn.json inside todo-list-app

3. Associated label to the input with class not id since a label can be associated only with exactly one form control. The same class can be used by multiple elements therefore it will be problem later figuring out in which case, which element would the label be for?
   - <input class="toggle-all" type="checkbox">
     <label for="toggle-all">Mark all as complete</label>
     Fixed become:
     - <input class="toggle-all" id="toggle-all" type="checkbox">

4. Jasmine Unit Testing.

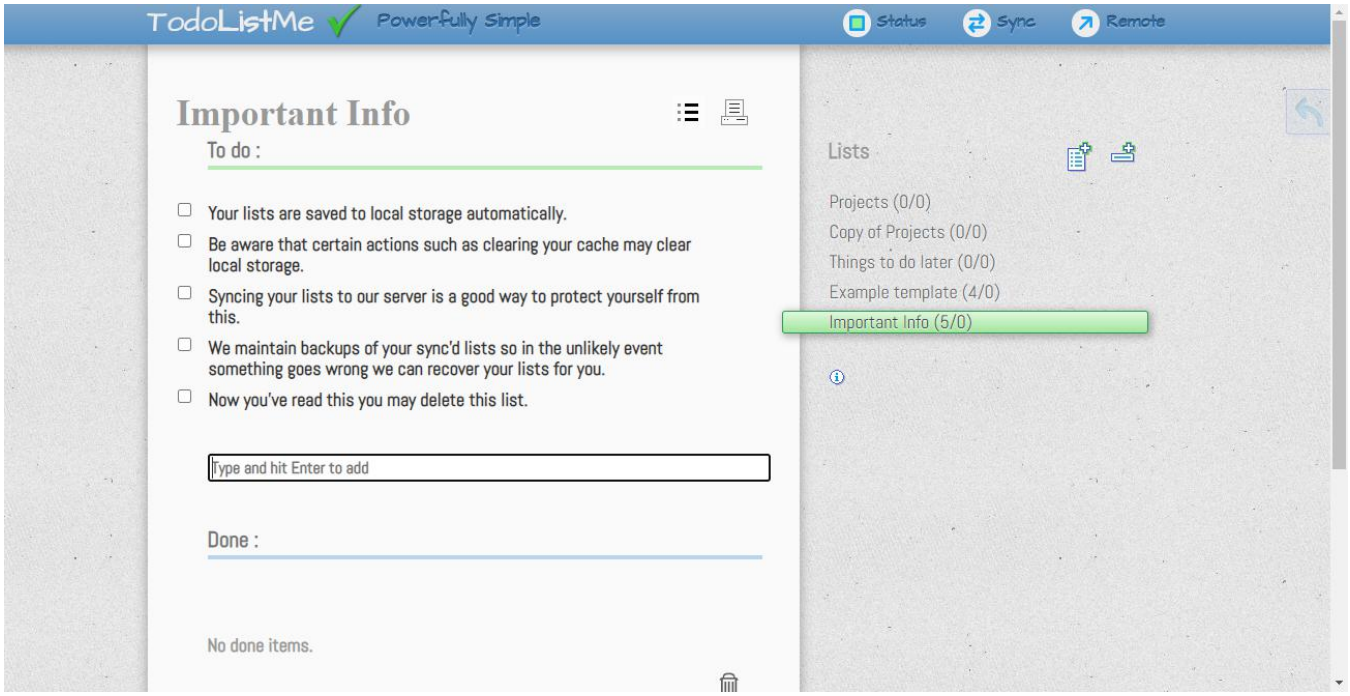


Additional test is required to already written one since some specs are still empty and have no expectation define as shown above. To do so new tests were added based on existing testing strategy. Newly created tests check following cases required:

1. should show entries on start-up
2. should show active entries
3. should show completed entries
4. should show the content block when todos exists
5. should highlight "All" filter by default
6. should toggle all todos to completed
7. should update the view
8. should add a new todo to the model
9. should remove an entry from the model

To run the jasmine unit testing open SpecRunner.html in test folder through any browser and you'll find newly created test asked on the requirement.

5. Audit Performance Competitor Todo-list app



#Brief

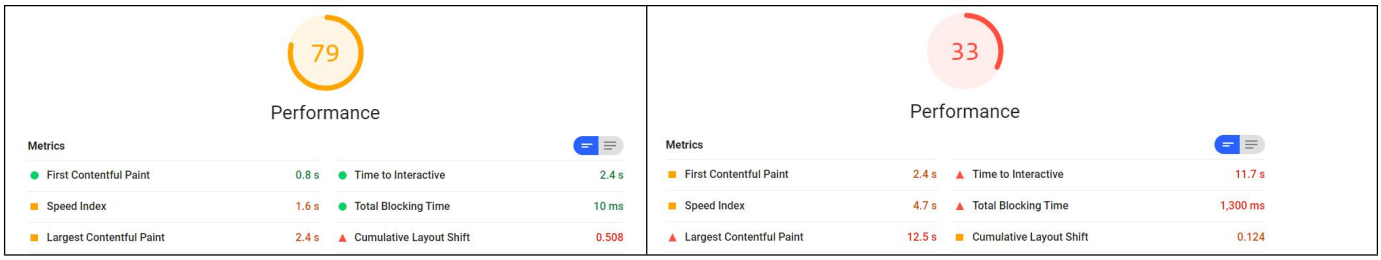
Competitors webpage todolistme.net was examined by using lighthouse 6.2.0 developer tools in google chrome as it gives automate audit, user-friendly metrics and suggestions for better site performance in the save-able report.

The audit test is conducted using Incognito (Private) Window to test website as a first-time user sees it. Cache is disabled and storages is cleared. Use simulated throttling to simulate how the page performances on a slower connection in both desktop and mobile device.

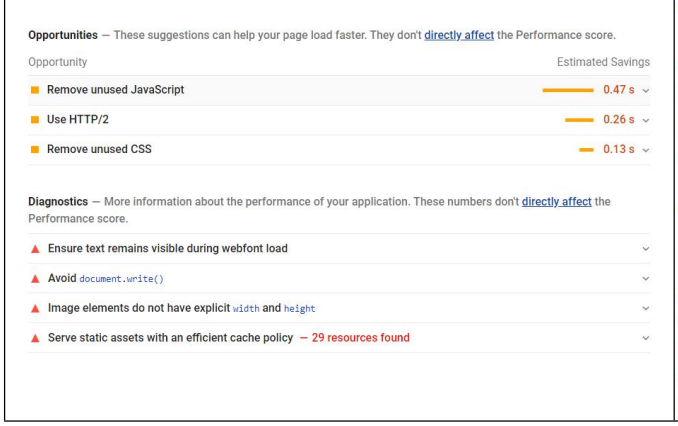
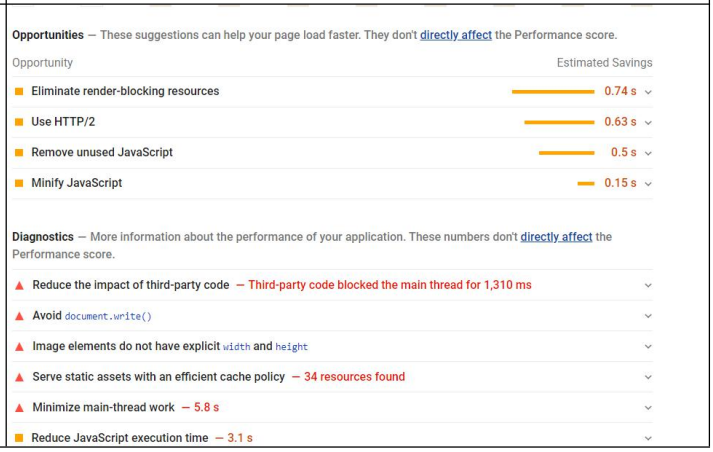
#Summary

The report shows the performance of todolistme.net is not bad in desktop but poor in mobile device with overall performance score 79 colored in orange in desktop and 33 colored in red in mobile device. The webpage shows good score for **Time To Interactive** (meaning it takes short time to render, load, register event handler, page ready for user interaction response within 50 milliseconds and display the visible content in the page as fast as it can) but shows bad **Cumulative Layout Shift** score **In Desktop**. While **In Mobile Device**, the **Time To Interactive** takes longer time but **First Contentful Paint** takes shorter time.

Desktop	Mobile



Some suggestions from lighthouse to improve the site performance are as follows:

Desktop	Mobile
	

#Comparison

By using the same audit tools setting like auditing the competitor todolistme.net before. The performance of todo-list-app are:

Desktop	Mobile
	

Compared to todolistme.net, todo-list-app has much better performance. The overall performance score 98 and 100 in todo-list-app shows that In both mobile and dekstop device, the site takes shorter time than todolistme.net and gives better user experience to render, load, register event handler, page is ready for user interaction response within 50 milliseconds and display the visible content in the page as fast as it can. Faster than competitor webpage since the app is more simple only basic functionality of todolist. The competitor todolistme.net has more complete todo-list-app feature where user can do registration, synchronizing, printing, commercials, create multiple lists,

multiple category, sorting, drag and drop reordering the tasks in the lists which also make this competitor webpage takes more time for loading and executing the feature code.

#Conclusions

The performance of todolistme.net is not as good as todo-list-app since it has more features that takes some time to load and execute than todo-list-app. Therefore external resources, scripts and css code in the todolistme.net could keep the content display delay, interaction not ready and delay user interaction. Therefore external resources, scripts and css needs more optimization. Lighthouse reports has suggestions that could be used to help analyze the solution faster and help understand defining the baseline aimed for achieving higher lighthouse performance score.

#Optimize Performance

Based on my analisis some optimization on todo-list-app are:

1) Minimizing scripts and stylesheets by deferring the download, marking it as async, or eliminating unused line codes, library or external scripts to keep transfer file small
2) reduce file sizes by compressing it with brotli or gzip to keep request counts low

Github Wiki: can be found in https://github.com/sintaanjelina/P8_sinta/wiki