Technical Document

Project Introduction

The to-do-list-app helps people to manage their daily tasks. It's an easy to use app that allows people to add, update, delete and toggle through their daily tasks and to check which tasks have been completed and which still needs to be completed. The application has been created by using Model-View-Controller (MVC): an architectural design pattern that separates three main functionalities: Model–View–Controller. MVC has been used here to add dynamical functionality: user can add new task without reloading the webpage (like Single Page Application–SPA).

MODEL as a central component manages data logic and methods. Here are created prototypes to manage a local storage object and main app functionality like adding, updating and deleting task of the list.

VIEW as an output representor displays and manages the user interactions with the application, manipulates DOM structure and represents its functionality.

CONTROLLER as a third part of the MVC pattern connects model and view by converting inputs from the View for the Model component.

This is a simple management tool but unfortunately there are many other existing to-do-List-apps available whit the same functionality and ease of use.

Program Specifications

Detailed description of all functions:

Controller Object: *controls interactions between Model and View.*

PARAMETERS: model object and view object.

PROTOTYPES:

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

Model Object: *creates new Model instance and connects it with the storage.*

PARAMETERS: storage object.

PROTOTYPES:

- create (creates a new to-do-list-app model and saves it in the storage)
- read (finds and returns a model in storage, if the query isn't given, returns everything)
- update (updates a model, every action based on unique ID)
- remove (removes a model from storage)
- removeAll (removes all data from storage)
- getCount (counting active, completed and total tasks by finding the in the storage)

View Object: *manipulates DOM structures attached to user interaction. It has two simple entry points:*

- bind (takes a to-do-app event and registers the handler)
- render (renders the given command with the options)

Storage Object:

- manages data storage by using the local session storage.

Helpers:

- a Bunch of helper methods for querying the selectors and encapsulating the DOM

Template:

- Delivers template function to display list items, change button states, escape characters

Testing Results

Two stages of testing were performed: Manual testing and Automatic Jasmine based testing

---

Manual testing results:

- Misspelling function declaration:
  - Location: js/controller.js (line 95)
  - Controller.prototype.adddItem

- Fixed:
  - Controller.prototype.addItem

- Potential conflict between duplicate ID's
  - Location: js/store.js (line 83 - 91)
  - Code:
```
var newId = "";
var charset = "0123456789";
for (var i = 0; i < 6; i++) {
newId += charset.charAt(Math.floor(Math.random() *
charset.length));
}
```
- Fixed

```
var newId = new Date().getUTCMilliseconds();
```

Automatic Jasmine unit testing.

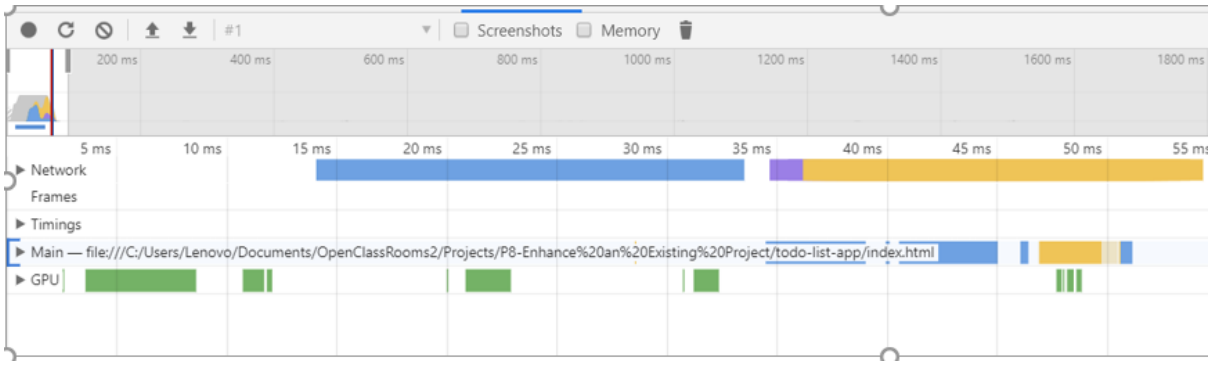
Process of automatic **Jasmine testing** required writing additional test to already written ones. Based on existing testing strategy new tests where added. Newly created tests were supposed to check following cases:

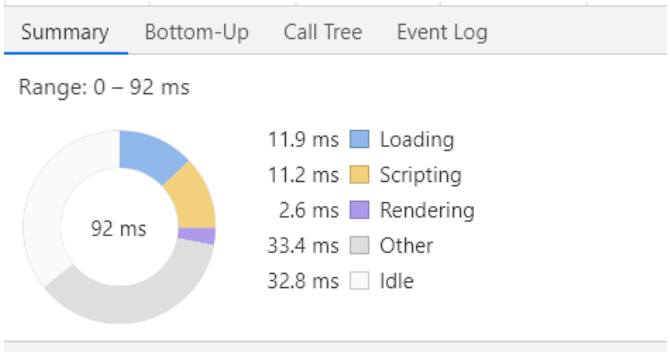
- Should show entries on start-up
  - The 'to-do-list-app' array should be empty, when the application starts
- Should show all entries without "all" route
  - Shows total count of the tasks, array can be empty or filled it with the tasks
- Should show active entries
  - The completed tasks which are set to false (completed = false)
- Should show completed entries
  - The completed tasks which are set to true (completed = true)
- Should show the content block when to-do-list-app exists
  - Create a list of the tasks, when they exist
- Should highlight "All" filter by default
  - Sets 'all' as default, takes total count, even if it's empty
- Should toggle all to-do-list-app to completed
  - Updates all tasks as completed (model component)
- Should update the view
  - Updates the status as completed (view component)
- Should add a new to-do-list-app to the model
  - Adds new task to the list
- Should remove an entry from the model
  - Removes to-do-list-app task (model component), empty array

Audit to-do-list-app

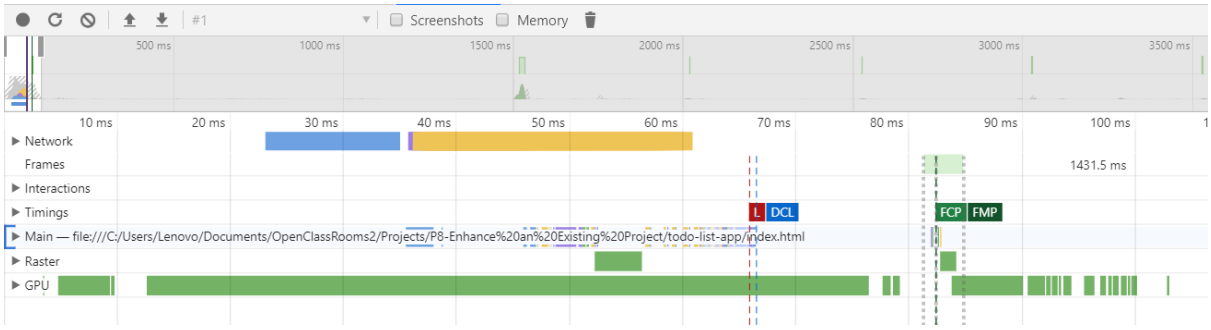
Audit was performed using Developer Tools in Chrome browser (v.64, 64-bit) on Windows 10 machine.

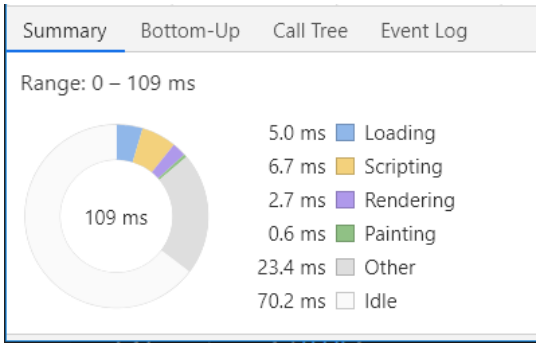
Loading the Page:



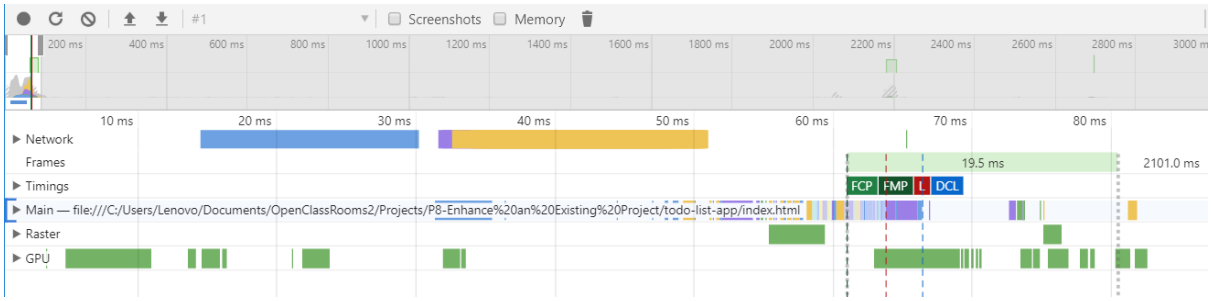


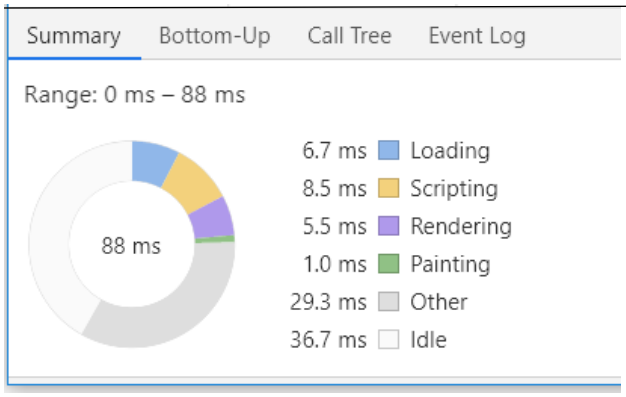
Typing and Adding One Task to The List:





Removing One Task from The List:





Network Connection by Loading the Page:

Name	Status	Type	Initiator	Size	Time
index.html	Finished	document	Other	1.5 KB	4.6 days
base.css	Finished	stylesheet	index.html	1.8 KB	7 ms
index.css	Finished	stylesheet	index.html	6.8 KB	8 ms
base.js	Finished	script	index.html	7.1 KB	10 ms
helpers.js	Finished	script	index.html	1.6 KB	10 ms
store.js	Finished	script	index.html	4.2 KB	11 ms
model.js	Finished	script	index.html	3.1 KB	11 ms
template.js	Finished	script	index.html	2.7 KB	12 ms
view.js	Finished	script	index.html	5.5 KB	12 ms
controller.js	Finished	script	index.html	6.9 KB	12 ms
app.js	Finished	script	index.html	604 B	12 ms
data:image/svg+xml;...	200	svg+xml	Other	(from memory ...	0 ms

Audit competitor's app: todolistme-app

The competitor's app is much more sophisticated

In addition to to-do-list-app it contains following functionalities:
Subpages with instructions, contact information and development options

- Registration
- Synchronizing
- Printing
- Commercials
- Multiple lists
- Sorting
- Drag and drop reordering

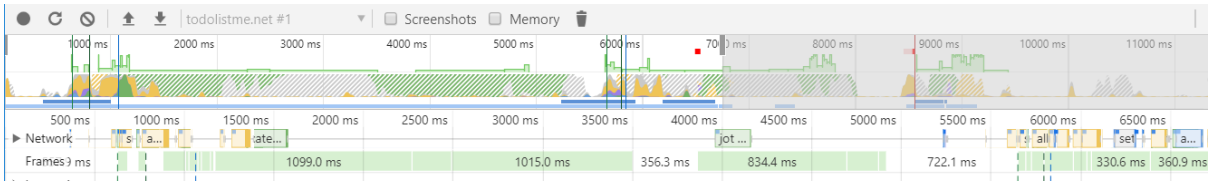
As these two apps differ a lot and todolistme is far more developed solution only basic parameters comparison was analysed.

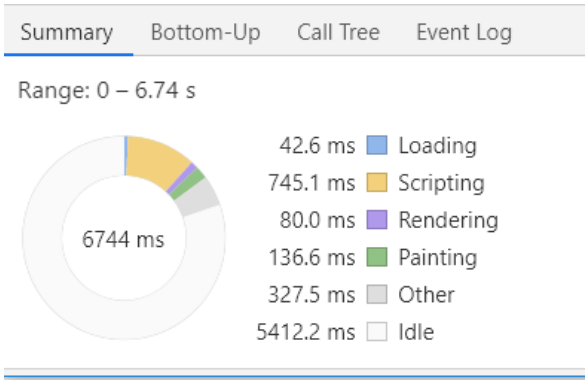
There is results of basic audit for todolistme page beneath.

RESULTS:

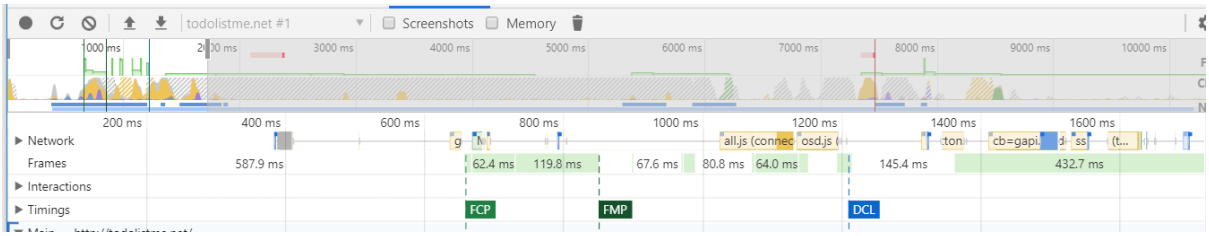
Loading time by loading the page, adding new task and removing task:

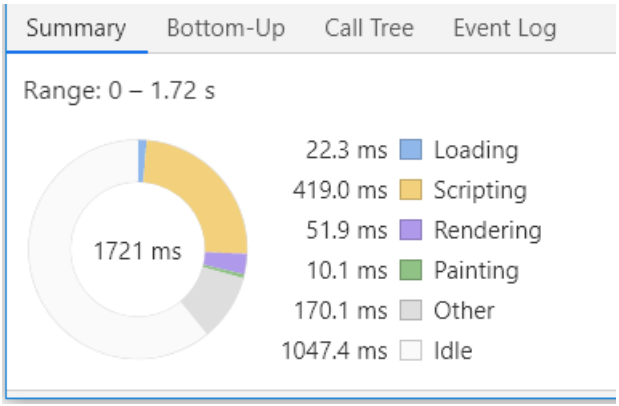
Loading the Page:



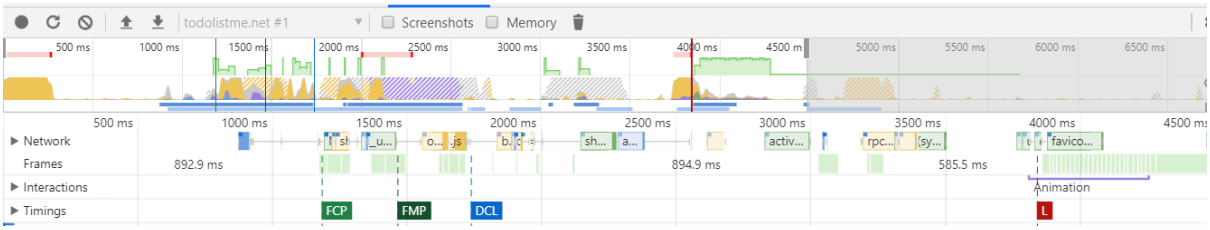


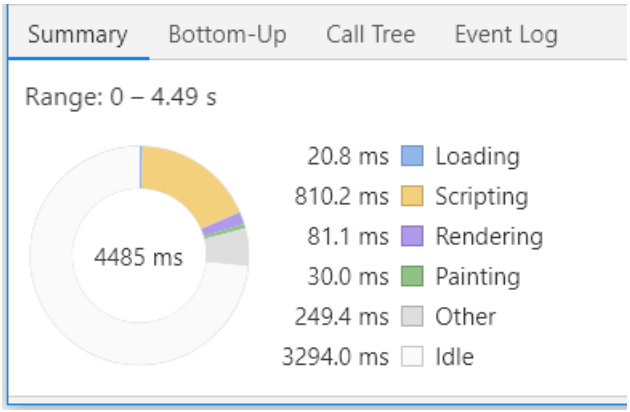
Typing and Adding One Task to The List:



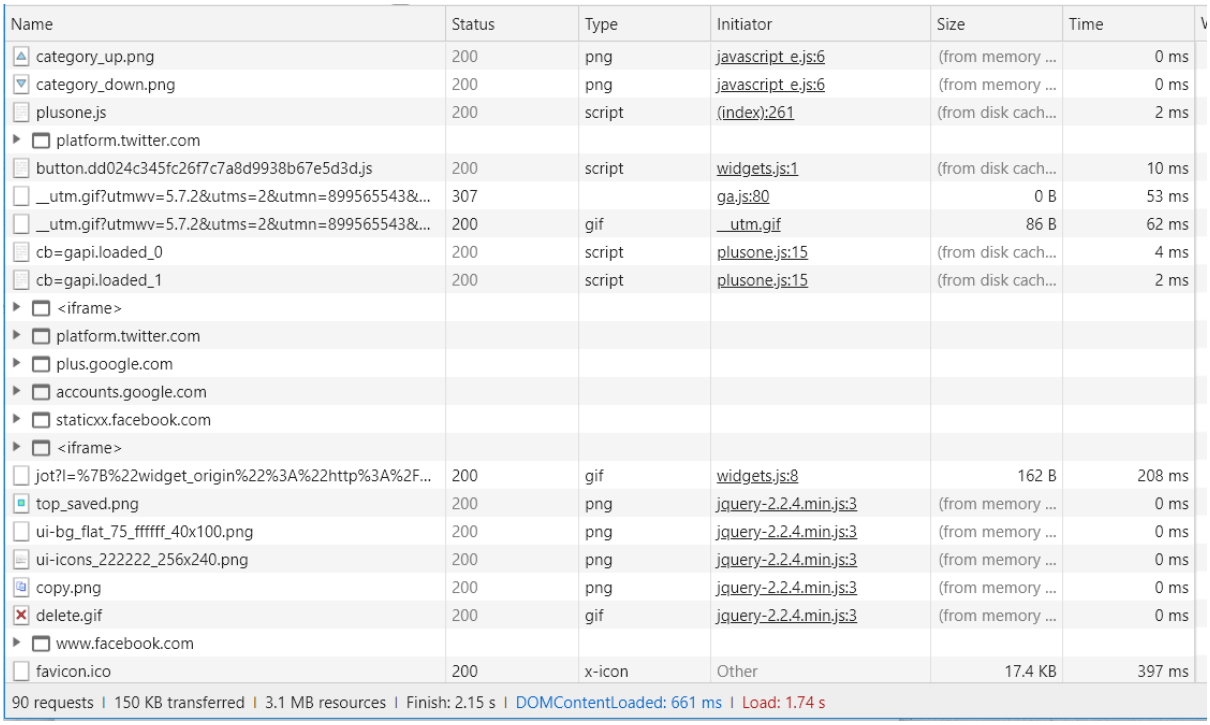


Removing One Task from The List:





Network Connection by Loading the Page:



Name	Status	Type	Initiator	Size	Time
category_up.png	200	png	javascript_e.js:6	(from memory ...	0 ms
category_down.png	200	png	javascript_e.js:6	(from memory ...	0 ms
plusone.js	200	script	(index):261	(from disk cach...	2 ms
platform.twitter.com					
button.dd024c345fc26f7c7a8d9938b67e5d3d.js	200	script	widgets.js:1	(from disk cach...	10 ms
__utm.gif?utmwv=5.7.2&utms=2&utmn=899565543&...	307		ga.js:80	0 B	53 ms
__utm.gif?utmwv=5.7.2&utms=2&utmn=899565543&...	200	gif	__utm.gif	86 B	62 ms
cb=gapi.loaded_0	200	script	plusone.js:15	(from disk cach...	4 ms
cb=gapi.loaded_1	200	script	plusone.js:15	(from disk cach...	2 ms
<iframe>					
platform.twitter.com					
plus.google.com					
accounts.google.com					
staticxx.facebook.com					
<iframe>					
jot?l=%7B%22widget_origin%22%3A%22http%3A%2F...	200	gif	widgets.js:8	162 B	208 ms
top_saved.png	200	png	jquery-2.2.4.min.js:3	(from memory ...	0 ms
ui-bg_flat_75_ffffff_40x100.png	200	png	jquery-2.2.4.min.js:3	(from memory ...	0 ms
ui-icons_222222_256x240.png	200	png	jquery-2.2.4.min.js:3	(from memory ...	0 ms
copy.png	200	png	jquery-2.2.4.min.js:3	(from memory ...	0 ms
delete.gif	200	gif	jquery-2.2.4.min.js:3	(from memory ...	0 ms
www.facebook.com					
favicon.ico	200	x-icon	Other	17.4 KB	397 ms

90 requests | 150 KB transferred | 3.1 MB resources | Finish: 2.15 s | DOMContentLoaded: 661 ms | Load: 1.74 s

Audit Comparison

The audit was performed using Developer Tools in Google Chrome browser

The table below shows the results of comparison between two web application described before.

Application comparison looks as follows:

	to-do-list-app	todolistme-app
Loading time	59 ms	1740 ms
Transferred	41.7 KB	150 KB
JS memory heap	9.2 MB – 17.2 MB	11.4 MB – 74.7 MB

Summary: to-do-list-app

Pros (+)

- Simple design
- Fast loading and operating
- Low memory consumption
- Low data transfer
- Basic functionality, easy to understand for a user
- Clean and readable code
- Page loads fast, because it based only on html, css and vanilla.js technologies
- Application is simple, without any heavy fonts, animations or complicated styles to be load

Cons (-)

- Limited functionality
- Only local storage in use, not possible to save the data for longer period of time

Summary: todolistme-app

Pros (+)

- Ready to use and sophisticated solution
- Plenty of useful functionalities
- Custom design
- Number of well optimized images for better user experience and well optimized css
- Well supported

Cons (-)

- Very slow loading time
- Memory consumption
- Google Ads delay loading and increases data transfer
- Code isn't well organized
- To many variables decleared globaly

Conclusion

App todo-list-app performs very efficient comparing to competitor. There is a space for further developments with regards of keeping app small and quick. It would be optimal to use dedicated CSS and continuing using vanilla JavaScript.

To-do-app can be developed as a sole application as well as a very efficient module to be combined in a larger project. One of the key challenges is to chose appropriate storage solution, that will allow to maintain its biggest advantages:

- Simplicity
- Speed
- Low recourses demand