TO-DO APP documentation – DRAFT

**#Introduction**.

To-do-app is an application which helps to manage tasks. The user can add new task to to-do list, update and easy delete it, separately or whole list when the tasks are completed. The app has minimalistic design and simply functionality.

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.

**#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 todo-list),
* showActive (renders uncompleted tasks),
* showComplited(renders completed tasks),
* addItem (creates new todo task, saving it in the local storage by adding ID),
* editItem (starts editing mode of todo 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 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 todo 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 todo application 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.

**#Manual bug fixing.**

During the manual debugging were two bugs found:

1. Type mistake:

* Location: js/controller.js (line 95),
* Controller.prototype.adddItem,

Fixed:

* + Controller.prototype.addItem.

1. Associated label to the input with a class:

* Location: index.html, (line 16);
* <input class="toggle-all" type="checkbox">

<label for="toggle-all">Mark all as complete</label>

Fixed:

* + <input class="toggle-all" id=” toggle-all” type="checkbox">

**#Automatic Jasmine unit testing.**

In this Jasmine dubbing process was required to add some tests to already written ones. New tests have to check following cases:

1. ‘should show entries on start-up’

* the ‘todo’ array should be empty, when the application starts;

1. ‘should show all entries without "all" route’

* shows total count of the tasks, array can be empty or filled it with the tasks;

1. ‘should show active entries’

* the completed tasks which are set to false (completed = false);

1. 'should show completed entries'

* the completed tasks which are set to true (completed = true);

1. 'should show the content block when todos exists'

* create a list of the tasks, when they exist;

1. 'should highlight "All" filter by default'

* sets 'all' as default, takes total count, even if it's empty;

1. 'should toggle all todos to completed'

* updates all tasks as completed (model component);

1. 'should update the view'

* updates the status as completed (view component);

1. 'should add a new todo to the model'

* adds new task to the list;

1. 'should remove an entry from the model'

* removes todo task (model component), empty array.

Audit Performance

Comparison

* simply, readable code
* single page app by using session storage

**#Audit to-do-app**

The audit of to-do app was performed using Developer Tools in Google Chrome browser on iOS machine.

#Results:

* page loads fast, because it based only on html, css and vanilla.js technologies,
* page doesn’t need a large amount of memory, because it doesn’t require any media files,
* application is simple, without any heavy fonts, animations or complicated styles to be load.

#Audit a competitor site: <http://todolistme.net>

To highlight the performance of the to-do-app it was recommended to compare its results to the competitor webpage: todolistme application.