Todo-list-app documentation

**#Introduction**

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.

**#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,
* showComplited 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.

**#Manual Bugs Fix**

1. Typo mistake:
   * Controller.prototype.adddItem at js/controller.js(line 95)

Fixed become :

* + Controller.prototype.addItem

1. 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">

**#Jasmine Unit Testing.**

Requirement of the test listed as follows:

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.