Table of Contents

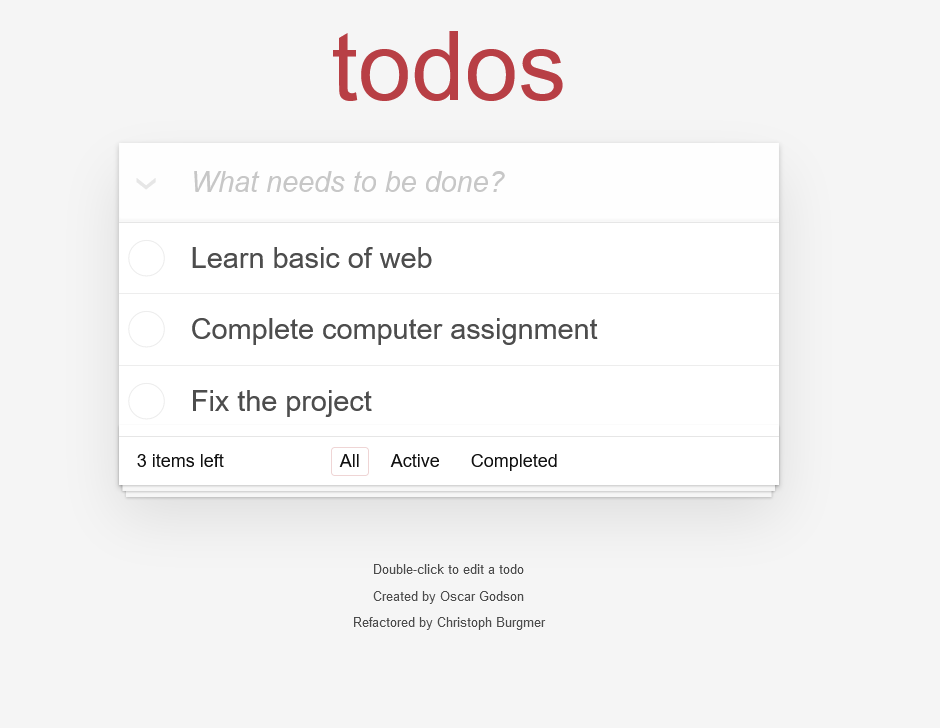
1. Introduction
2. Configuring project
3. How it works (Technically)
4. Bug fixing
5. Adding tests
6. Todos App performance audit
7. TodoListMe App performance audit
8. Audit Comparison

Introduction

Todos is a web app and simple to use application which allows the user to keep track of the task that he/she wish to complete in a certain duration of time.

The application allows to user the perform the following actions.

* Add new todo.
* Modify existing todo.
* Delete existing todo.
* Clear all completed todos.
* Mark todo item as active or completed.
* Mark all todo as active or completed.
* Filter by display all todos that are active or completed.



Configuring the projects

1. Getting required files.
2. Download the files from the GitHub repository if you do not have them.
3. Extract all the file into you are hard drive.
4. Install the project dependencies using the following command (npm install)

To start the app simply open file named index.html located at the root of the project folder.

1. How to use

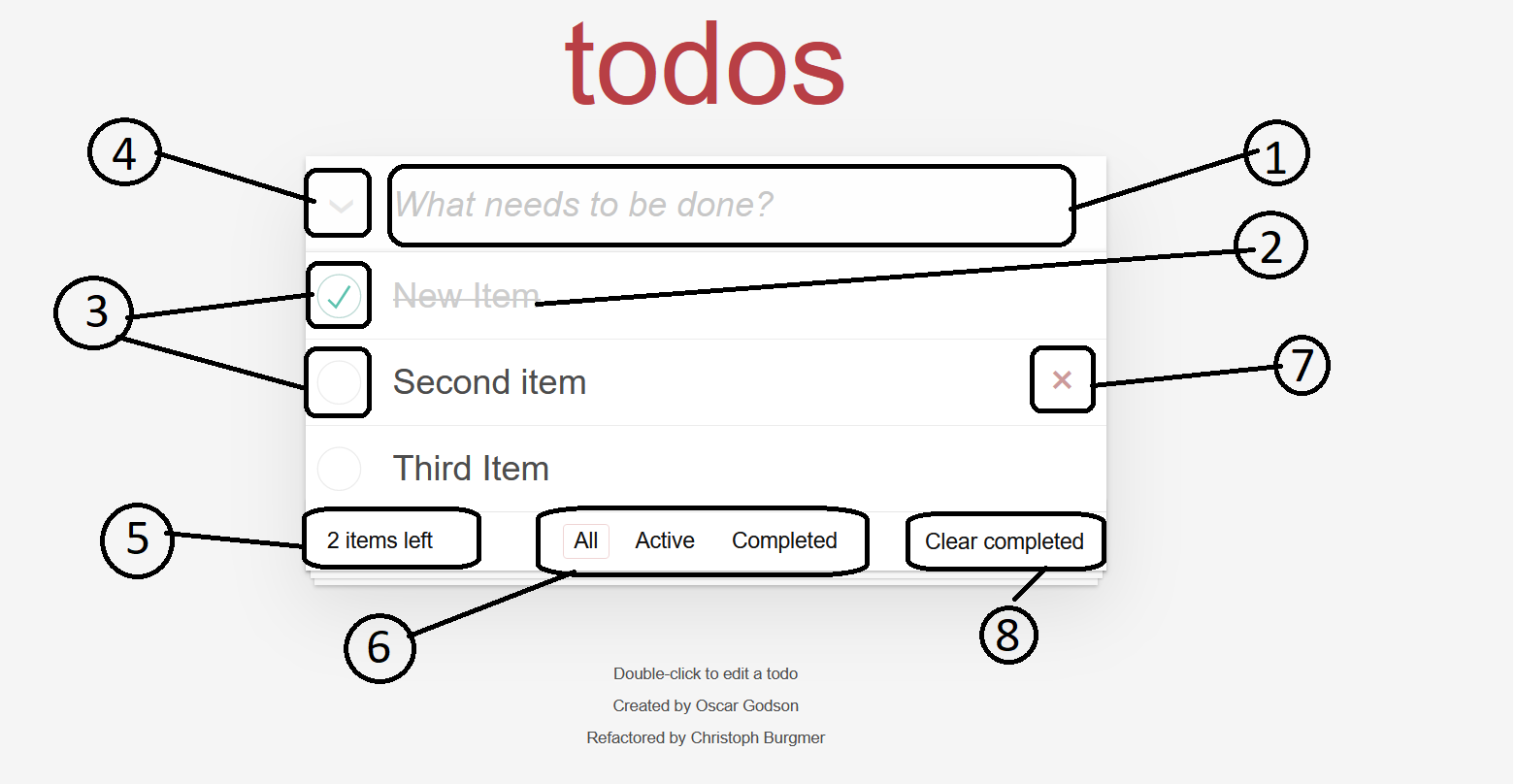


Diagram explanations

1. Used to add the new todo by clicking in the text “What needs to be done?”
2. Show the name of the task created.
3. Show the state of the task if it is checked means the task is completed otherwise it’s still active.
4. Button to mark all todo as either active or completed depending on the filter depending on the conditions below,

If the filter is all it will mark all todo as completed

If the filter is active it will mark all todo as completed

If the filter is completed it will mark todo as active

1. Shows all task that are active (not completed)
2. Filter section which let you change view to display all, active or completed todo.
3. To delete one active task hover to the given task name and the X mark will appear click it to perform the delete function.
4. To delete all completed todo click this element.

How it works (Technically)

The Todos app use the MVC architecture. MVC stands for MODEL – VIEW – CONTROLLER

Components

Model

* The central component of the pattern. It is the application’s dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

View

* Used for all the UI logic of the application, for example display todo, buttons, text boxes, etc, that the final user interacts with

Controller

* Act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model components and interact with the Views to render the final output. For example, Todo controller will handle all the interactions and inputs from the todo View and update the database using the todo model.

Files and Methods

.HTML FILES

***index.html*** - Our application starting file (in other words - It is the application entry point.)

.CSS FILES

***index.css*** - Defines our application CSS styles.

.JS FILES

***app.js*** - Sets up a brand new Todo list.

***model.js*** - Creates a new Model instance and hooks up the storage.

Methods:

* create - Creates a new todo model.
* read - Finds and returns a model in storage.
* update - Updates a model.
* remove - Removes a model from storage.
* removeAll - Removs all data from storage
* getCount - Returns a count of all todo

***controller.js*** - Takes a model and view and acts as the controller between them.

Methods:

* setView - Loads and initialises the view.
* showAll - Will get all items and display them in the todo-list.
* showActive - Renders all active tasks.
* showCompleted - Renders all completed tasks.
* addItem - Adding new item to our todo list.
* editItem - Triggers the item editing mode.
* editItemSave - Finishes the item editing mode.
* editItemCancel - Cancels the item editing mode.
* removeItem - Remove item from the DOM and remove it from storage.
* removeCompletedItems - Will remove all completed items from the DOM and storage.
* toggleComplete - Toggles item between completed and not completed (active).
* toggleAll - Will take all todo and make them complete or incomplete.
* \_updateCount - Update number of todo remaining as incomplete (active).
* \_filter - Re-filters the todo items, based on the active route.
* \_updateFilterState - Simply updates the filter nav's selected states.

***helpers.js*** - It functions are:

* Getting element by CSS selector and attaching event listener to it.
* Attaching a handler to event for all elements that match the selector.
* Finding the element's parent with the given tag name.
* Allowing for looping on nodes by chaining forEach method.

***store.js*** - Creates a new client-side storage object.

Methods:

* find - Finds items based on a query given as a JS object.
* findAll - Will retrieve all data from the collection.
* save - Will save the given data to the DB.
* remove - Will remove an item from the Store based on its ID.
* drop - Will drop all storage and start fresh.

***template.js*** - Sets up defaults for all Template methods such as a default template.

Methods:

* show - Creates an <li> HTML string and returns it for placement in our app.
* itemCounter - Displays a counter of how many to dos are left to complete.
* clearCompletedButton - Updates the text within the "Clear completed" button.

***view.js*** - View that abstracts away the browser's DOM completely. It has two simple entry points:

* bind(eventName, handler) - Takes a todo application event and registers the handler.
* render(command, parameterObject) - Renders the given command with the options.

Bugs Fixing

Four bugs have been found in application code.

1. Failing to add new todo o the list.

Location: controller.js line 95

Trace: Controller.prototype.adddItem = function (title) {

Solution: Controller.prototype.addItem = function(title) {

1. Bug which may leads to potential conflict between duplicate IDs (ID for new todo has been generated randomly with the set which is small and can result into duplicate ID’s)

Location: store.js line 84

Trace:

var newId = "";

var charset = "0123456789";

for (var i = 0; i < 6; i++) {

newId += charset.charAt(Math.floor(Math.random() \* charset.length));

}

Solution: var newID = Date.now();

1. Missing ID in input tag for toggle-all label which prevented toggle all todo to work correctly.

Location: index.html line 16

Trace: <input class="toggle-all" type="checkbox">

Solution: <input class="toggle-all" id=”toggle-all” type="checkbox">

Adding tests

Test has been created by using Jasmine.

Jasmine is an open-source testing framework for JavaScript.

To start test simply locate SpecRunner.html file and start it also to create or modify test use the ControllerSpec.js file in your directory.

Nine Tests has been added to the existing tests:

1. Should show entries on start-up.
2. Should show active entries.
3. Should show completed entries.
4. Should highlight “All” filter by default.
5. Should highlight “Active” filter when switching to active view.
6. Should toggle all to 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.

Todos App performance audit

Audit was performed by using Chrome browser developer tools.

TodoListMe App performance audit

Audit Comparison