PROJET 8

Préparé pour : OpenClassRooms
Préparé par : Audrey FRANÇOIS, Formation Développeuse d'Application Front-End
31 janvier 2020

I. Cadre du projet

I. 1 ENJEU

Il s'agit là, de pouvoir aider tout un chacun à s'organiser au quotidien.

I. 2 OBJECTIF

Mener à bien la création d'une Application de gestions de tâches aux fonctionnalités adaptées, maintenable et simple d'utilisation afin de toucher un large public sur tout type de support (ordinateur, tablette, smartphone, …)

I. 3 CONTEXTE

L'application existe déjà. Cependant, elle présente quelques dysfonctionnements. Elle gagne à être améliorée en conséquence. D'autant que de nombreuses applications de To-do-List sont déjà présentent sur le marché.

I. 4 CHARTE GRAPHIQUE

Brief créatif : Design simple aux lignes claires et épurées avec des contrastes suffisants pour entrer dans les normes d'accessibilités.

Police de caractère : Helvetica Neue

Palette de couleurs : Dégradé de gris et rose.

#f2f2f2	rgb(242, 242, 242)
#b4b3b3	rgb(180, 179, 179)
#e4bebc	rgb(228, 190, 188)
#cccccc	rgb(204, 204, 204)
#d4d4d4	rgb(212, 212, 212)

II. spécifications fonctionnelles

II. 1 FONCTIONNALITÉS

FRONT END
Créer une todo
Editer/modifier une todo existante
Déclarer une todo complète
Déclarer toutes les todos complètes
Supprimer une todo
Supprimer toutes les todos déclarées complètes
Filtre d'affichage : toutes les todos / actives / complètes
Afficher un décompte du nombre de todos actives

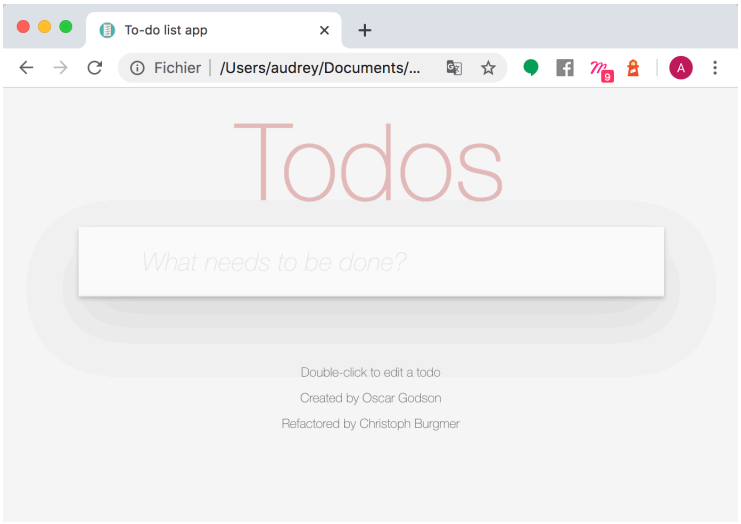
BACK-END
Enregistrer une nouvelle todo dans le stockage local du navigateur
Enregistrer de façon pérenne les modifications apportées à une todo dans le stockage local
Supprimer une Todo du stockage local
Supprimer toutes les todos complètes sur stockage local
Décompter le nombre de Todos actives

II. 2 DESCRIPTION FONCTIONNELLE

L'application est composée d'une unique page.

1. Accueil

Lors du premier lancement de l'application, voici la page d'accueil qui s'affiche, offrant à l'utilisateur la possibilité de renseigner sa toute première tâche à effectuer :



2. Composition de la page après ajout d'une ou plusieurs Todos :

A. HEADER

B. BLOCK LISTE TODOS

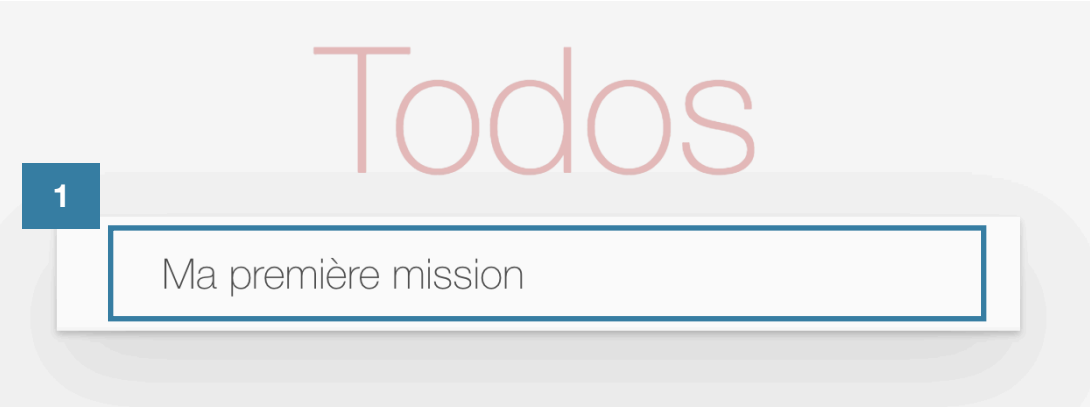
C. TODO

D. BARRE D'OUTILS

E. FOOTER

A. HEADER

Visuel :



Contexte :

Au premier lancement, un formulaire invite l'utilisateur à saisir sa première Todo.

Composition :

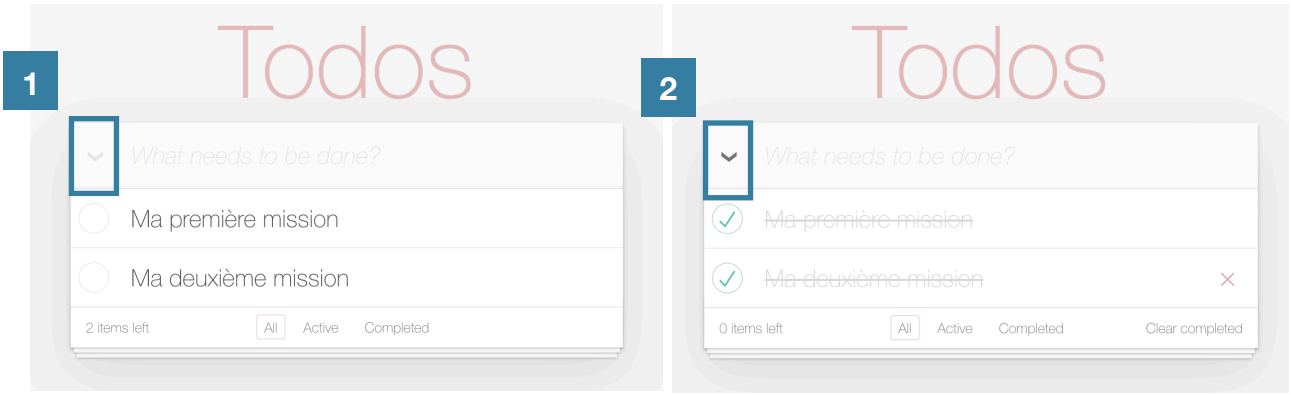
- Titre du site
- Champs de formulaire pour ajouter une todo.

Interactions utilisateur :

1. Cliquez dans le champ de saisie et saisissez votre 1ère tâche.
2. Appuyez sur la touche « Entrer « pour enregistrer votre Todo.

B. BLOCK LISTE TODOS

Visuels :



Contexte :

Un block de contenu apparaît une fois qu'une première Todo est enregistrée. Ce block contiendra l'ensemble des Todos saisies par l'utilisateur et donc celles qui seront enregistrées dans le Local Storage du navigateur lors d'ouvertures ultérieures de l'Application.

Composition :

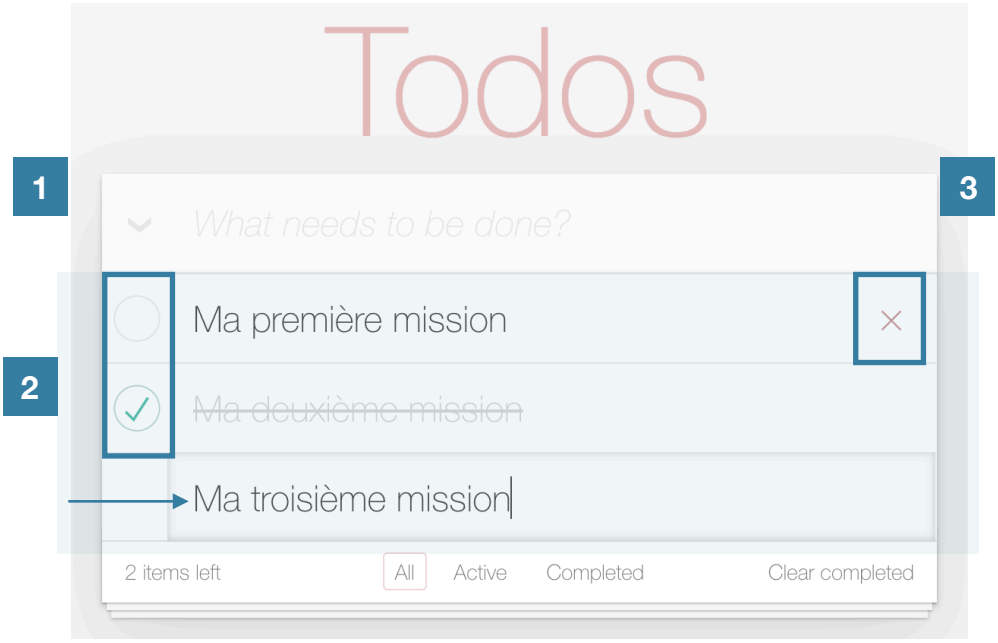
- Un icône Toggle-all à gauche de « *What needs to be done ?* »
- Une liste de Todos
- Une barre d'outils

Interactions utilisateur :

1. En cliquant sur l'icône Toggle-All, vous pouvez choisir de déclarer complètes l'ensemble des Todos.
2. Ou de les dé-compléter.

C. TODO

Visuel :



Contexte :

Une Todo est une tâche que l'utilisateur doit réaliser. Il peut en ajouter autant qu'il le souhaite.
Elle a deux statuts possibles : Active ou Complétée.
Une Todo peut être éditée afin d'être modifiée.
Elle peut être supprimée.

Composition :

- Un icône cercle « Toggle »
- Un contenu texte
- Un champ formulaire
- Un bouton « supprimé »

Interactions utilisateur :

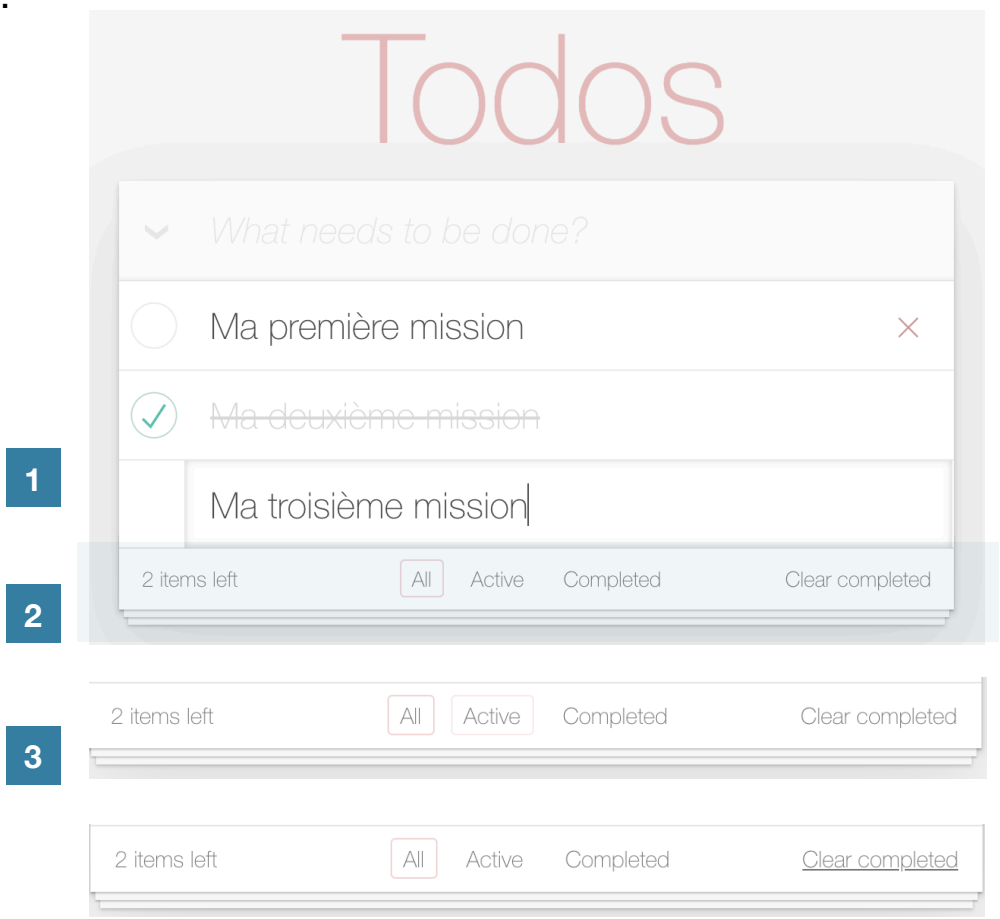
1. Changer le statut de la Todo :
   En cliquant sur le bouton « toggle », vous pouvez déclarer une tâche complète ou inversement. Lorsqu'elle est complète, l'affichage de la tâche s'en trouve modifié. Elle est barrée et apparaît en gris plus clair. Un petit icône vert « validé » apparaît dans le cercle de la « toggle ».

2. Editer une Todo :
   En double cliquant sur une tâche, cette dernière passe en mode édition. Il est ainsi possible d'en modifier le contenu. Et en cliquant sur « Entrer », la modification s'enregistre et le nouvel intitulé de la Todo s'affiche dans la liste. Si, au moment de la modification, l'utilisateur clique sur « escape », cette dernière est annulée. Si, après modification, le champs reste vide, la Todo est supprimée.

3. Suppression de la Todo :
   Au survol d'une Todo, une croix apparaît. En cliquant sur celle-ci, il est possible de supprimer définitivement une todo de la liste.

D. BARRE D'OUTILS

Visuels :



Contexte :

La barre d'outils permet à l'utilisateur de connaitre le nombre de tâches encore actives, de gérer l'affichage des Todos en fonction de leur statut.

Composition :

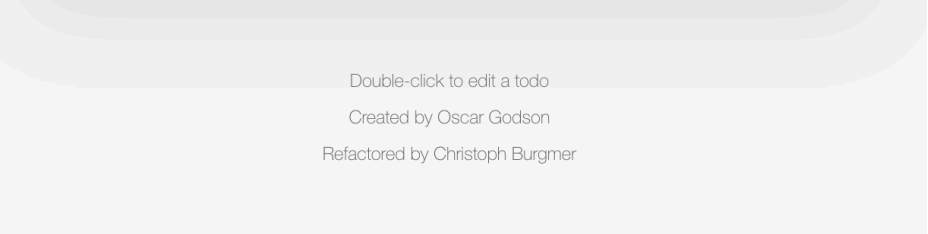
- Un span indiquant le nombre de Todos actives.
- 3 boutons Filtres permettant d'afficher au choix : toutes les todos / uniquement celles qui sont actives / Uniquement les todos complètes.
- Un bouton « Clear Completed ».

Interactions utilisateur :

1. Par défaut, le bouton « All » est actif. Il est encadré d'un trait rose foncé. Toutes les todos y apparaissent quelque soit leur statut.
   En cliquant sur le bouton « Active » seules les todos actives apparaitront.
   En cliquant sur le bouton « Completed », seules celles déclarées complètes seront affichées. S'il n'y a pas de Todos complètes la liste restera vide.
2. Au survol d'un élément du filtre, un encadré rose clair apparaît.
3. Lorsqu'une ou plusieurs Todos sont déclarées complètes, un bouton « Clear completed » apparaît. Il permet en un seul clique de supprimer l'ensemble des Todos complètes. Au survol de ce bouton, l'intitulé est surligné.

E. FOOTER

Visuel :



Contexte :

Le Footer donne des indications concernant : la méthode pour éditer une Todo (ligne 1) ainsi que l'identité de ceux qui ont créé l'application.

Composition :

- Texte.

Interactions utilisateur :

Aucune interaction.

III. spécifications techniques

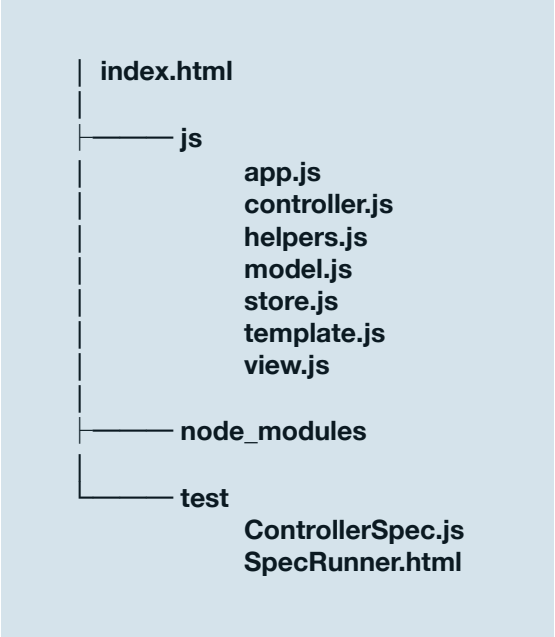
3.1 GÉNÉRALITÉS

A / Introduction

Nous avons pour ambition ici de réaliser une documentation technique destinée aux développeurs. Elle permettra de s'approprier le code en vue de modifications et améliorations ultérieures éventuelles.

Todos est une application simple et rapide réalisée avec **JavaScript**.

Voici son arborescence principale (non-exhaustive) :



Nous constatons que le coeur même de cette application est contenu dans le **dossier JS**. Ce sont ces fichiers qui gèrent toutes les fonctionnalités de la Todo-List.

Le dossier **node_modules** contient le framework de tests unitaires **Jasmine** ainsi que tous les fichiers nécessaires à l'affichage correct et cohérent de l'application (fichiers CSS).

Le dossier **test** contient les tests unitaires mis en place à l'aide du framework **Jasmine**.

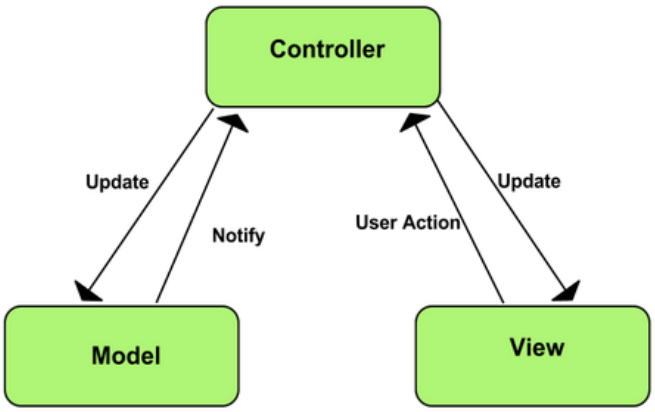
C'est le dossier JS qui retiendra, en particulier, notre attention pour l'aspect technique.

B/ Un modèle de conception MVC

Notre code javascript est organisé selon une architecture MVC (MODEL - VIEW - CONTROLLER) :

- **Modèle** - Gère les données d'une application.
- **Vue** - Donne une représentation visuelle du modèle.
- **Contrôleur** - Relie l'utilisateur et le système. Il assure le lien entre les interactions utilisateur et la vue.

C'est une architecture qui fonctionne en cycle :



Le **modèle** concerne les données. Dans cette application de tâches, ce seront les tâches réelles et les méthodes qui les ajouteront, les modifieront ou les supprimeront. Le modèle ici représente les attributs associés à chaque tâche à effectuer, tels que la description et l'état. Lorsqu'un nouvel élément de tâche est créé, il est stocké dans une instance du modèle.

La **vue** est la façon dont les données sont affichées. Dans cette application, ce sera le HTML rendu dans le DOM et le CSS. C'est ce qui est présenté aux utilisateurs et comment les utilisateurs interagissent avec l'application. La vue ne sait pas comment mettre à jour le modèle car c'est le travail du contrôleur.

Le **contrôleur** lie le modèle et la vue. Il prend la saisie de l'utilisateur, comme cliquer ou taper, et gère les rappels pour les interactions utilisateur. Le contrôleur met à jour la vue lorsque le modèle change. Il ajoute également des écouteurs d'événements à la vue et met à jour le modèle lorsque l'utilisateur manipule la vue.

Le modèle ne touche jamais la vue. La vue ne touche jamais le modèle. Le contrôleur les connecte.

Cette structuration claire du code permet de :

- construire un code maintenable et réutilisable.
- découpler l'application en une série de composants indépendants.

3.2 PRESENTATION GENERALE DES FICHERS JS

A/ Fichier helpers.js

Au lieu de charger une dépendance lourde comme jQuery, on a créé des « helpers », c'est-à-dire, des fonctions d'aide qui rendent les tâches compliquées ou répétitives un peu plus faciles.

FONCTIONS/ Elements de code	COMMENTAIRES
qs	querySelector, récupère le premier élément html d'après un sélecteur CSS à l'intérieur d'un autre élément.
qsa	query SelectorAll, récupères tous les éléments html d'après un sélecteur CSS à l'intérieur d'un autre élément.
$on	Attache un addEventListener à un élément existant.
$delegate	Attache un gestionnaire d'évènements à un élément existant ou qui sera créé par la suite.
$parent	Trouve l'élément parent avec le tagName désigné.
forEach	Simplifie la gestion d'un forEach sur des éléments HTML.

B/ Fichier app.js

C'est dans ce fichier que sera initialisé l'application. Le code visible ici montre l'architecture globale de l'application.

FONCTIONS/ Elements de code	COMMENTAIRES
Todo	Met en place l'application Todo-List qui sera stockée dans le Local Storage du navigateur. Créé une instance du model qui permettra d'interagir avec les données stockées dans le Local Storage. Définit le template html de base d'une Todo. La nouvelle View est rattachée à ce template. Le contrôleur fait le lien entre le model et la view.
setView	Gère l'URL de la page et le chargement de la View en fonction de celle-ci. Elle est appelée au chargement de la page, mais également lorsque le hashtag de l'url change.

C/ Fichier store.js

Il s'occupe de la sauvegarde des tâches dans le LocalStorage du navigateur.

La propriété localStorage permet d'accéder à un objet local Storage. Le localStorage est similaire au sessionStorage. La seule différence : les données stockées dans le localStorage n'ont pas de délai d'expiration, alors que les données stockées dans le sessionStorage sont nettoyées quand la session navigateur prend fin — donc quand on ferme le navigateur.[1]

FONCTIONS/ Elements de code	COMMENTAIRES
Store	créé une nouvelle base de donnée si elle n'existe pas déjà.
.find	Permet de trouver un item dans la base à partir d'une requête.
.findAll	Récupère toutes les données.
.save	Enregistre un nouvel item, ou met à jour un item déjà existant.
.remove	Supprime un item de la base.
.drop	Efface toute la base.

D/ Fichier model.js

C'est le fichier qui va structurer les données. Au moment de l'initialisation du Model, il va appeler le système de stockage, ici l'objet contenu dans store.js.

FONCTIONS/ Elements de code	COMMENTAIRES
Model	Créé une nouvelle instance de model et utilise le storage.
.create	Créé une nouvelle Todo.
.read	Recherche et retourne la Todo correspondant à la requête passée en paramètre.
.update	Met à jour des données de la Todo dont l'ID est passé en paramètre.
.remove	Supprime une Todo dont l'ID est passé en paramètre de la base de données.
.removeAll	Supprime tous les Todos du stockage.
.getCount	Compte le nombre de todos.

[1] https://developer.mozilla.org/fr/docs/Web/API/Window/localStorage

E/ Fichier template.js

Ce fichier contient les fonctions qui génèrent le template HTML pour l'application afin d'être ensuite utilisé par View.js. Habituellement, dans un schéma classique MVC, c'est le fichier View.js qui génère le HTML.

FONCTIONS Elements de code	COMMENTAIRES
Template	initialise le template par défaut.
.show	Récupère le template par défaut et y injecte les informations du todo.
.itemCounter	Gère l'affichage du nombre de todos à terminer.
.clearCompletedButton	Gère l'affichage du bouton "Clear completed » s'il y a plus de zéro tâches complétées.

F/ Fichier view.js

Ce fichier va se concentrer sur l'affichage et gérer l'interface utilisateur. C'est le code HTML du template par défaut qui sera modifié dans view.js selon les fonctions appelées dans ce fichier.

Il y a deux fonctions principales : .bind et .render.

FONCTIONS Elements de code	COMMENTAIRES
View	Initialise la vue en se basant sur le template
._removeItem	Supprime une todo.
._clearCompletedButton	Affiche ou cache le bouton "Clear completed".
._setFilter	Gère l'affichage de la page en fonction du choix de l'utilisateur (all, active, completed).
._elementComplete	Gère l'affichage des Todos marquées complètes
._editItem	Gère l'affichage lors de l'édition du titre d'une todo existante.
._editItemDone	Quitte le mode édition de la todo qui vient d'être modifié.
.render	Va, selon les paramètres utilisés, lancé une des fonctions énumérées ci-après.
showEntries	affiche les todos.
.removeItem	Supprime une todo
updateElementCount	Actualise le nombre de todos.

FONCTIONS Elements de code	COMMENTAIRES
clearCompletedButton	mets à jour le bouton Clear completed.
contentBlockVisibility	affiche ou masque le "footer" de la todo-list.
toggleAll	Check tous les éléments.
setFilter	Gère l'affichage des filtres
clearNewTodo	Vide le champ texte principal de la todo-list.
elementComplete	Gère l'affichage d'un todo complété.
editItem	Gère l'affichage d'un todo en cours de modification.
editItemDone	Gère l'affichage d'un todo dont la modification vient d'être terminée.
._itemId	Récupère l'ID d'un todo.
._bindItemEditDone	Gère l'affichage lors de la perte de focus du todo en cours d'édition.
._bindItemEditCancel	Gère l'affichage du todo dont la modification est annulée.
.bind	Attache un gestionnaire d'évènement, avec des évènements javascript associées qui permettront d'effectuer le rendu.

G/ Fichier controller.js

Il sera l'intermédiaire entre l'interface utilisateur (View.js) et la structure des données (Model.js). Quand une action sera effectuée, le controller interviendra à la fois au niveau des données et sur l'affichage.

FONCTIONS Elements de code	COMMENTAIRES
Controller	Crée le lien entre le model et la view.
.setView	Initialise la vue.
.showAll	Affiche tous les todos.
.showActive	Montre toutes les todos actives.
.showCompleted	Montre toutes les todos complétées.
.addItem	Ajoute une todo.
.editItem	Passe l'item en mode édition
.editItemSave	Enregistrer une todo dont le titre a été modifié et sort du mode édition.
.editItemCancel	Annule la modification du titre d'une todo.

FONCTIONS Elements de code	COMMENTAIRES
.removeItem	Supprime une todo correspondant à l'ID passé en paramètre
.removeCompletedItems	Supprime toutes les todos complétées du DOM et du Storage.
.toggleComplete	Coche ou décoche une todo et met à jour le statut de l'item dans le storage.
.toggleAll	Coche ou décoche tous les todos.
._updateCount	Met à jour le nombre de todos.
._filter	Filtre les todos selon leur statut.
._updateFilterState	Mets à jour les boutons de navigation en fonction de l'état des items (all, active, completed).

3. 3 REPRESENTATION SCHEMATIQUE

A/ Ajout d'une ToDo