

# Projet 8

## Reprenez et améliorez un projet existant

Étudiante : Marina Front

### Objectifs :

Reprendre un projet existant, corriger les bugs installer des tests et l'améliorer en optimisant ses performances.

### Analyse du code :

Le projet à une structure model-vue-contrôleur c'est à dire que le controlleur fait le lien entre le model et la vue.

Premièrement, dans le fichier App.js, l'application est initialisée, les objets Store, Model, Template, View et Controller sont instanciés.

Puis, une fois que la page du navigateur est chargée, la fonction setView du controlleur est appelée. Cette fonction est le point d'entrée du programme.

En premier lieu le programme lit l'url et en fonction, va mettre à jour les différents éléments de la vue.

### Étape 1 : Corriger les bugs :

#### Bug 1 :

Indice donné : une faute de frappe.

Je vais essayer d'ajouter une todo pour faire apparaître l'erreur dans la console.

Voici l'erreur :



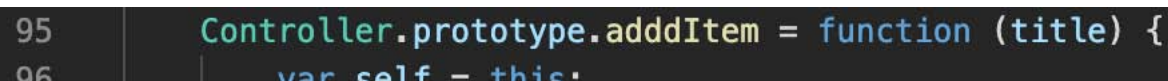
```
✖ ▶ Uncaught TypeError: self.addItem is not a function
    at controller.js:17
    at HTMLInputElement.<anonymous> (view.js:177)
```

Je vais donc dans controller.js à la ligne 17



```
15
16     self.view.bind('newTodo', function (title) {
17         self.addItem(title);
18     });
```

L'erreur dit que addItem() n'est pas une fonction, donc, soit self n'existe pas, or self existe, soit "addItem" n'existe pas. Je cherche addItem() dans le fichier et en effet je ne la trouve pas, je trouve :



```
95     Controller.prototype.addItem = function (title) {
96         var self = this;
```

Le nom de la fonction a été mal écrit. Après la correction, je peux ajouter des todos ;)

## Bug 2 :

Indice donné : conflit éventuel entre deux IDs identiques.

J'utilise l'application dans tous les sens pour faire apparaître le bug dans la console. Il n'apparaît pas, je vais, donc, chercher un peu dans le code là où sont créés les todos et leur id.

`controler.addItem() > Model.create() > store.save()`

```
83 // Generate an ID
84 var newId = "";
85 var charset = "0123456789";
86
87 for (var i = 0; i < 6; i++) {
88   newId += charset.charAt(Math.floor(Math.random() * charset.length));
89 }
```

Lors de la création d'une todo, pour générer l'id, le programme va piocher 6 fois un chiffre au hasard dans la chaîne de caractère `charset = "0123456789"`

Résultat : créer deux ids identiques est difficile (1 chance sur 1 million) mais pas impossible.

Il existe plusieurs possibilités pour créer des ids uniques :

- En reprenant le code existant et en déplaçant la génération de l'id dans une fonction récursive qui va boucler dans les todos et vérifier si l'id qui vient d'être généré n'existe pas déjà, si oui la fonction est appelée pour régénérer un id, puis la vérification est effectuée à nouveau et ceci jusqu'à ce que l'id n'existe pas déjà.

- Soit on utilise un uuid, c'est une chaîne de caractères générée aléatoirement de 128 bit. Bien que la probabilité qu'un UUID soit dupliqué n'est pas nulle, elle est suffisamment proche de zéro pour être négligeable.

- Soit en reprenant l'idée d'id très long où, le risque de duplication est quasi null, on ne change pas le système existant, mais on augmente le nombre de boucle, au lieu de 6 on peut piocher 36 chiffres qui définiront l'id. Ainsi, le risque d'avoir un id identique sera fortement diminué.

- Soit on utilise un nombre incrémenté à chaque nouvelle todo c'est-à-dire que même si on supprime la todo, la todo suivante aura un id incrémenté.

- Soit on utilise un timestamp. Celui-ci correspond au nombre de secondes qui s'est écoulé depuis le 1er janvier 1970, Ainsi nous sommes assurés de son unicité dans la mesure où ce projet de todolist ne peut pas demander deux id en même temps. (solution utilisée)

## Améliorations à faire (dans le code) :

J'ai trouvé dans le code plusieurs boucles inutiles. Quelques exemples :

1 -

```
158     Controller.prototype.removeItem = function (id) {
159         var self = this;
160         var items;
161         self.model.read(function(data) {
162             items = data;
163         });
164
165         items.forEach(function(item) {
166             if (item.id === id) {
167                 console.log("Element with ID: " + id + " has been removed.");
168             }
169         });
170
171         self.model.remove(id, function () {
172             self.view.render('removeItem', id);
173         });
174
175         self._filter();
176     };
```

Ici, lorsque qu'on veut supprimer un élément, le programme lit tous les elements `model.read()` puis il boucle dans les éléments pour trouver celui que l'on veut supprimer. Une fois trouvé, il fait un simple `console.log()` pour afficher l'id (alors qu'on avait déjà l'id à disposition fournis en paramètre )

Je supprime les lignes 160 à 169

```

119      * @param {number} id The ID of the item you want to remove
120      * @param {function} callback The callback to fire after saving
121      */
122      Store.prototype.remove = function (id, callback) {
123          var data = JSON.parse(localStorage[this._dbName]);
124          var todos = data.todos;
125          var todoId;
126
127          for (var i = 0; i < todos.length; i++) {
128              if (todos[i].id == id) {
129                  todoId = todos[i].id;
130              }
131          }
132
133          for (var i = 0; i < todos.length; i++) {
134              if (todos[i].id == todoId) {
135                  todos.splice(i, 1);
136              }
137          }
138
139          localStorage[this._dbName] = JSON.stringify(data);
140          callback.call(this, todos);
141      };

```

Ici, comme au dessus, la programme boucle pour trouver l'id à supprimer alors qu'on le reçoit en paramètre. Le seul avantage de cette boucle est de vérifier si la todo avec cet id existe bien. La deuxième boucle fait le splice() pour supprimer l'élément. Il n'est pas nécessaire de faire une boucle pour faire un splice, il nous faut juste la position de l'élément de plus faire un splice dans un tableau alors que l'on est en train de tourner dans ce tableau n'est pas conseillé.

Ma proposition de correction :

```

118      * @param {number} id The ID of the item you want to remove
119      * @param {function} callback The callback to fire after saving
120      */
121      Store.prototype.remove = function (id, callback) {
122          var data = JSON.parse(localStorage[this._dbName]);
123          var todos = data.todos;
124          var todoPosition;
125
126          for (var i = 0; i < todos.length; i++) {
127              if (todos[i].id == id) {
128                  todoPosition = i;
129              }
130          }
131          todos.splice(todoPosition, 1);
132
133          localStorage[this._dbName] = JSON.stringify(data);
134          callback.call(this, todos);
135      };

```

## Étape 2 : Où sont les tests :

Les tests se trouvent dans le dossier «test» dans le fichier «ControllerSpec.js». et pour lancer les tests il faut lancer le fichier «SpecRunner.html» ce fichier affiche les résultats des tests.

Le fichier de test contenait déjà de nombreux tests , j'en ai ajouté 9 ceux avec le commentaire : «// TODO: write test»

Un exemple de test :

```
165     it('should highlight "Active" filter when switching to active view', function () {
166         // TODO: write test
167         // lorsque le programme veut afficher les todos active, il faut encadre le filtre "active"
168         var todo = { id: 42, title: "my todo", completed: true };
169         setUpModel([todo]);
170
171         subject.setView("#/active");
172
173         expect(view.render).toHaveBeenCalledWith("setFilter", "active");
174     });
```

Les tests se trouvent dans des fonctions «describe» qui décrivent la fonctionnalité qui va être testée. Puis un test se trouve dans une fonction «it» qui décrit le test (ici : doit mettre en lumière le filtre «active» quand on veut afficher les todos actives)

Ligne 168 : on définit une todo

Ligne 169 : cette fonction du fichier ControllerSpec crée un faux environnement à partir du model

Ligne 171 : c'est la fonction qui est testée

Ligne 173 : c'est le test. ici on s'attend à ce que la fonction «view.render» soit appelée avec les paramètres «setFilter» et «active»

Si c'est le cas, le test passera sinon il affichera l'erreur

The screenshot shows the Jasmine test runner interface. At the top, it says "Jasmine 2.99.0" and "Options". Below that, a green bar indicates "32 specs, 0 failures" and "finished in 0.05s". The main area lists the test suites and their individual specs:

- controller
  - should show entries on start-up
- routing
  - should show all entries without a route
  - should show all entries without "all" route
  - should show active entries
  - should show completed entries
- should show the content block when todos exists
- should hide the content block when no todos exists
- should check the toggle all button, if all todos are completed
- should set the "clear completed" button
- should highlight "All" filter by default
- should highlight "Active" filter when switching to active view
- toggle all
  - should toggle all todos to completed with some todo completed and some active
  - should toggle all todos to completed with all todos active
  - should toggle all todos to completed with all todos completed
  - should update the view
- new todo
  - should add a new todo to the model
  - should add a new todo to the view
  - should clear the input field when a new todo is added
- element removal
  - should remove an entry from the model
  - should remove an entry from the view
  - should update the element count
- remove completed
  - should remove a completed entry from the model
  - should remove a completed entry from the view
- element complete toggle
  - should update the model
  - should update the view
- edit item
  - should switch to edit mode
  - should leave edit mode on done
  - should persist the changes on done
  - should remove the element from the model when persisting an empty title
  - should remove the element from the view when persisting an empty title
  - should leave edit mode on cancel
  - should not persist the changes on cancel



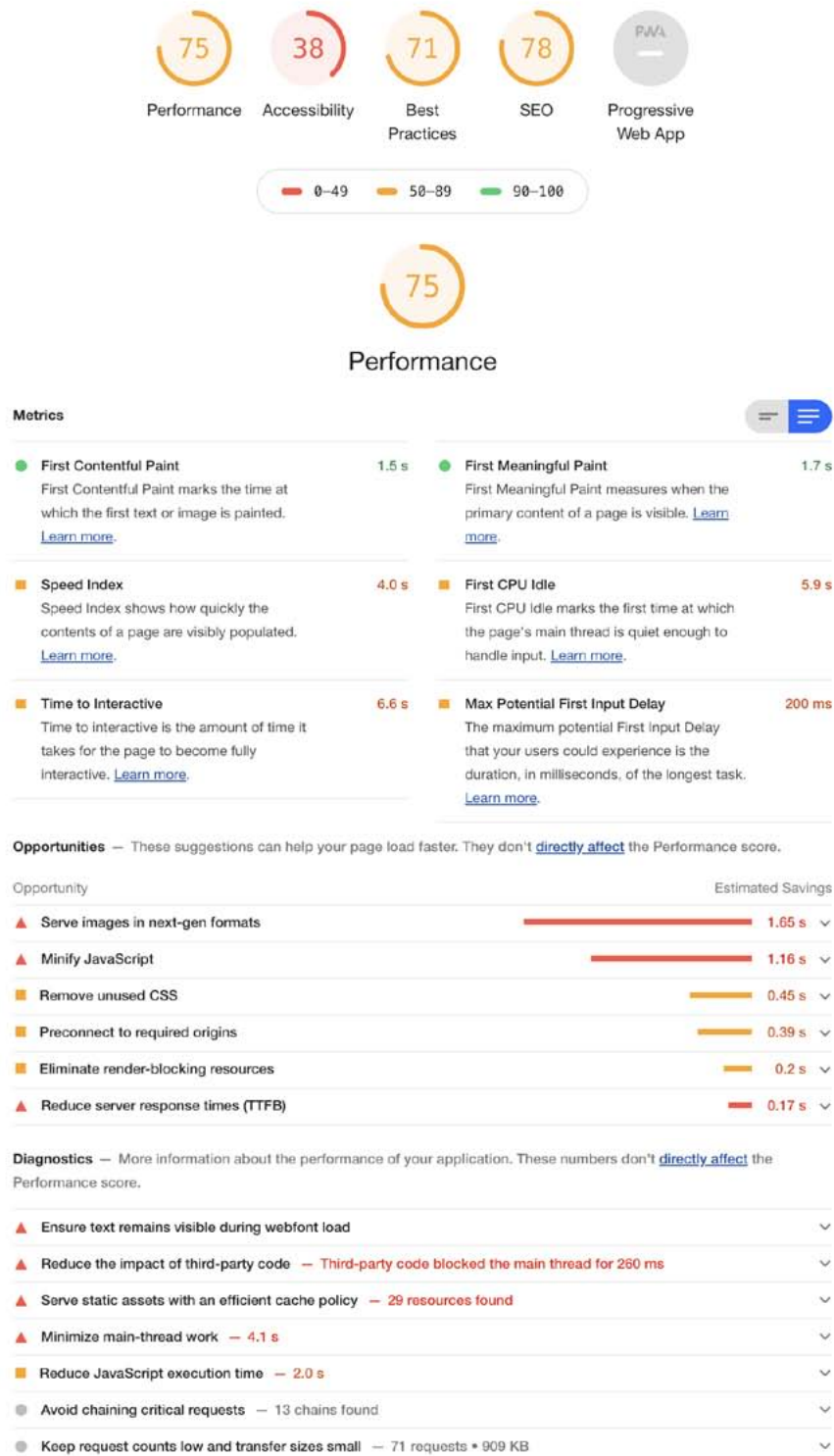
# Étape 3 : Optimiser la performance :

Le projet propose d'analyser la performance d'un site concurrent : <https://www.todolistme.net>

Je lance l'analyse avec lighthouse, un outil d'audit de google chrome. Cet outil analyse différents paramètres :

- Le performance du site, sa vitesse de chargement
- L'accessibilité aux personnes ayant des handicaps,
- PWA, sa capacité à s'adapter aux mobiles
- Les bonnes pratiques,
- Le référencement

Détail de l'analyse de todolistme.net :



#### Passed audits (11)

- Properly size images — Potential savings of 3 KB
- Defer offscreen images
- Minify CSS
- Efficiently encode images
- Enable text compression
- Avoid multiple page redirects
- Preload key requests
- Use video formats for animated content
- Avoids enormous network payloads — Total size was 909 KB
- Avoids an excessive DOM size — 363 elements
- User Timing marks and measures

38

## Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

**Contrast** — These are opportunities to improve the legibility of your content.

- ▲ Background and foreground colors do not have a sufficient contrast ratio.

**Best practices** — These items highlight common accessibility best practices.

- ▲ [id] attributes on the page are not unique

**Names and labels** — These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

- ▲ <frame> or <iframe> elements do not have a title
- ▲ Image elements do not have [alt] attributes
- ▲ Form elements do not have associated labels

**Internationalization and localization** — These are opportunities to improve the interpretation of your content by users in different locales.

- ▲ <html> element does not have a [lang] attribute

**Additional items to manually check (11)** — These items address areas which an automated testing tool cannot cover. Learn more in our guide on [conducting an accessibility review](#).

#### Passed audits (6)

- The page contains a heading, skip link, or landmark region
- Document has a <title> element
- Links have a discernible name
- Lists contain only <li> elements and script supporting elements (<script> and <template>).
- List items (<li>) are contained within <ul> or <ol> parent elements
- No element has a [tabindex] value greater than 0

Not applicable (23)

● [accesskey] values are unique	▼
● [aria-*] attributes match their roles	▼
● [role]s have all required [aria-*] attributes	▼
● Elements with an ARIA [role] that require children to contain a specific [role] have all required children.	▼
● [role]s are contained by their required parent element	▼
● [role] values are valid	▼
● [aria-*] attributes have valid values	▼
● [aria-*] attributes are valid and not misspelled	▼
● <audio> elements contain a <track> element with [kind="captions"]	▼
● Buttons have an accessible name	▼
● <dl>'s contain only properly-ordered <dt> and <dd> groups, <script> or <template> elements.	▼
● Definition list items are wrapped in <dl> elements	▼
● <html> element has a valid value for its [lang] attribute	▼
● <input type="image"> elements have [alt] text	▼
● Presentational <table> elements avoid using <th>, <caption> or the [summary] attribute.	▼
● The document does not use <meta http-equiv="refresh">	▼
● [user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5.	▼
● <object> elements have [alt] text	▼
● Cells in a <table> element that use the [headers] attribute refer to table cells within the same table.	▼
● <th> elements and elements with [role="columnheader"/"rowheader"] have data cells they describe.	▼
● [lang] attributes have a valid value	▼
● <video> elements contain a <track> element with [kind="captions"]	▼
● <video> elements contain a <track> element with [kind="description"]	▼



## Best Practices

▲ Does not use HTTPS — 41 insecure requests found	▼
▲ Does not use HTTP/2 for all of its resources — 27 requests not served via HTTP/2	▼
▲ Includes front-end JavaScript libraries with known security vulnerabilities — 2 vulnerabilities detected	▼
▲ Browser errors were logged to the console	▼

Passed audits (11)

● Avoids Application Cache	▼
● Uses passive listeners to improve scrolling performance	▼
● Avoids document.write()	▼
● Links to cross-origin destinations are safe	▼
● Avoids requesting the geolocation permission on page load	▼
● Page has the HTML doctype	▼
● Detected JavaScript libraries	▼
● Avoids requesting the notification permission on page load	▼
● Avoids deprecated APIs	▼
● Allows users to paste into password fields	▼
● Displays images with correct aspect ratio	▼





## SEO

These checks ensure that your page is optimized for search engine results ranking. There are additional factors Lighthouse does not check that may affect your search ranking. [Learn more.](#)

**Mobile Friendly** — Make sure your pages are mobile friendly so users don't have to pinch or zoom in order to read the content pages. [Learn more.](#)

▲ Does not have a <meta name="viewport"> tag with width or initial-scale  
No '<meta name="viewport">' tag found

**Content Best Practices** — Format your HTML in a way that enables crawlers to better understand your app's content.

▲ Image elements do not have [alt] attributes

**Additional items to manually check (1)** — Run these additional validators on your site to check additional SEO best practices.

**Passed audits (7)**

- Document has a <title> element
- Document has a meta description
- Page has successful HTTP status code
- Links have descriptive text
- Page isn't blocked from indexing
- Document has a valid hreflang
- Document avoids plugins

**Not applicable (4)**

- robots.txt is valid
- Document has a valid rel=canonical
- Document uses legible font sizes
- Tap targets are sized appropriately



## Progressive Web App

These checks validate the aspects of a Progressive Web App. [Learn more.](#)

**Fast and reliable**

- Page load is fast enough on mobile networks
- ▲ Current page does not respond with a 200 when offline
- ▲ start\_url does not respond with a 200 when offline No usable web app manifest found on page.

**Installable**

- ▲ Does not use HTTPS — 41 insecure requests found
- ▲ Does not register a service worker that controls page and start\_url
- ▲ Web app manifest does not meet the installability requirements Failures: No manifest was fetched.

★ PWA Optimized	
▲ Does not redirect HTTP traffic to HTTPS	▼
▲ Is not configured for a custom splash screen <b>Failures: No manifest was fetched.</b>	▼
▲ Does not set a theme color for the address bar. <b>Failures: No manifest was fetched, No `<meta &gt;`="" b="" found.<="" name="theme-color" tag=""/></b>	▼
● Content is sized correctly for the viewport	▼
▲ Does not have a <code>&lt;meta name="viewport"&gt;</code> tag with width or initial-scale <b>No `<code>&lt;meta name="viewport"&gt;</code>` tag found</b>	▼
● Contains some content when JavaScript is not available	▼
▲ Does not provide a valid apple-touch-icon	▼
<b>Additional items to manually check (3)</b> — These checks are required by the baseline <a href="#">PWA Checklist</a> but are not automatically checked by Lighthouse. They do not affect your score but it's important that you verify them manually.	

Runtime Settings	
URL	http://todolistme.net/
Fetch time	May 6, 2020, 3:07 PM GMT+2
Device	Emulated Desktop
Network throttling	150 ms TCP RTT, 1,638.4 Kbps throughput (Simulated)
CPU throttling	4x slowdown (Simulated)
User agent (host)	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36
User agent (network)	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3694.0 Safari/537.36 Chrome-Lighthouse
CPU/Memory Power	1622

## Résumé de l'audit du site concurrent

### Accessibilité

Le site todolistme.net a des notes plutôt correctes pour tous les critères sauf l'accessibilité. L'accessibilité permet de faire que le site soit facile à utiliser pour les personnes ayant un handicap. Ici il manque pas mal de chose pour rendre le site accessible :

- Il n'y a pas d'attribut « lang » pour la balise html
- Il n'y a pas d'attributs « alt » pour chaque image
- Il n'y a pas de label pour chaque input,
- La couleur du font ne contraste pas assez avec les couleurs du texte
- Il y a des ids identiques dans la page

Ces points sont les plus critiques, mais on peut ajouter d'autres éléments pour améliorer l'accessibilité (des attributs « aria- », permettre la navigation clavier... )

### Performance

Il est difficile d'avoir 100 en performance, ici les pertes se font sur le chargement de certaines parties :

- Chargement des librairies externe (il est donc préconisé d'en utiliser le moins possible et que des librairies vraiment utiles),
- Le chargement des fonts depuis internet (il est possible pour les fonts de les charger en local)

- L'exécution des scripts javascript (ici, il est préconisé d'écrire du code le plus performant possible et de minifier les scripts)

## Bonnes pratiques

todolistme.net ne respecte pas certaines bonnes pratiques :

- Il n'est pas en https et il utilise un http obsolète
- Il inclut des bibliothèques qui comportent des vulnérabilités
- Il affiche des erreurs dans la console

Pour les deux premiers points il est conseillé d'effectuer des mises à jour régulières.

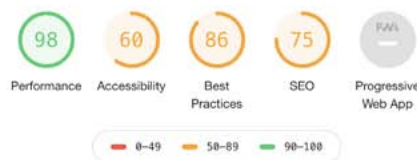
## Référencement

Pour le référencement, l'audit nous redit de penser à mettre des attributs « alt » à toutes les images et d'ajouter une balise <meta name="viewport"> pour optimiser le site sur les formats mobile.

## Progressive Web App

L'audit nous dit que le chargement du site est assez rapide pour les mobiles mais qu'il n'est pas prévu pour être une PWA. En effet, il n'a pas de manifeste.

## L'audit du projet



## Accessibilité

Input "new-todo" n'a pas de label

## Performance

Le chargement des CSS bloque le chargement du reste de la page

L'utilisation du cache pourrait accélérer le chargement

Minifier les CSS et JS pourrait également accélérer le chargement

## Bonnes pratiques

Pas de HTTPS

Une erreur dans la console

## Référencement

Pas de meta description

pas de balise viewport

## Progressive Web App

Cette application n'est pas une PWA

## Après quelques ajustements :



## Étape 4 : Améliorer le projet :

### Documentation technique et fonctionnelle

Pour améliorer le projet, il est proposé de créer une documentation.

Pour cela j'ai utilisé JsDoc. (<https://www.npmjs.com/package/jsdoc>)

JsDoc est un générateur de documentation, il la génère à partir des commentaires spéciaux qui se trouvent dans le code entre `/** */` :

Exemple :

```
139
140
141  /**
142   * Met à jour une todo avec un nouveau titre, si le nouveau titre est vide, la todo est supprimée
143   *
144   * @param {number} id L'id de la todo
145   * @param {string} title le nouveau titre de la todo
146   */
147  Controller.prototype.editItemSave = function (id, title) {
148    var self = this;
149    title = title.trim();
```

Ligne 183-186 : la description de la fonction

Ligne 188 : description du paramètre, son type, son nom et sa description

On peut ajouter plusieurs paramètres, décrire ce que retourne la fonction...

Résultat après avoir généré la doc :

`editItemSave(id, title)`

Met à jour une todo avec un nouveau titre, si le nouveau titre est vide, la todo est supprimée

#### Parameters:

Name	Type	Description
id	number	L'id de la todo
title	string	le nouveau titre de la todo

Source: [controller.js, line 146](#)

Documentation complète fonctionnelle et technique en lançant le fichier `index.html` dans le dossier `/out`

## Expérience utilisateur

Au niveau de l'expérience utilisateur, j'ai remarqué des fonctionnalités qui ne sont pas évidentes à comprendre, ni pratiques à utiliser.

1-

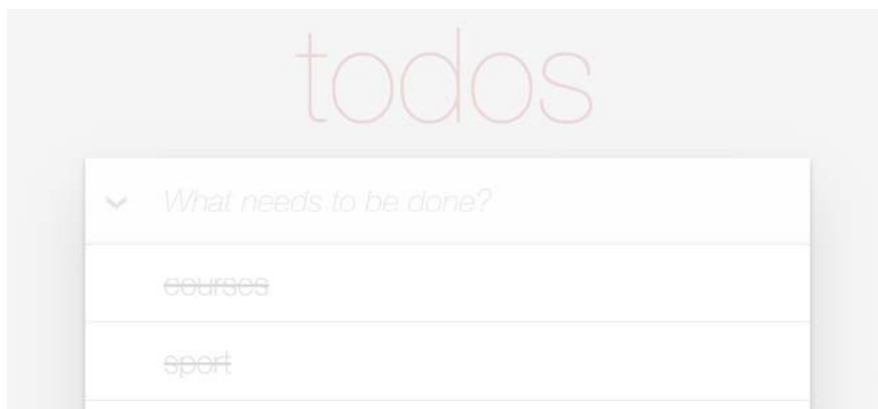
Pour le moment lorsqu'on commence à écrire une todo et que l'on clique ailleurs sur la page, la todo est ajoutée. Ce fonctionnement n'est pas pratique pour l'utilisateur.

Ma proposition d'amélioration est que l'ajout ne puisse se faire qu'avec la touche entrer :

```
226     View.prototype.bind = function (event, handler) {
227         var self = this;
228         if (event === "newTodo") {
229             $on(self.$newTodo, "keyup", function (event) {
230                 if (event.keyCode === self.ENTER_KEY) {
231                     handler(self.$newTodo.value);
232                 }
233             });
234         } else if (event === "removeCompleted") {
```

2-

La fonctionnalité toggle-all ne fonctionne pas. Normalement, en cliquant sur la petite flèche à gauche du champs d'ajout d'une todo, toutes les todos devraient se cocher ou se décocher.



En effet, il manque un id dans le html

Il faut ajouter id= « toggle-all » dans l'input

```
15     <section class="main">
16         <input class="toggle-all" type="checkbox">
17         <label for="toggle-all">Mark all as complete</label>
18         <ul class="todo-list"></ul>
19     </section>
```

Cette fonctionnalité n'est pas du tout explicite. Avec cet icon de flèche vers le bas, j'ai plutôt l'impression que si je clique je vais ajouter la todo que je viens d'écrire dans le champ à côté.



Ma proposition d'amélioration est d'afficher la checkbox avec son label :



Ainsi que de programmer l'ajout de la todo lorsque l'on clique sur la flèche vers le bas :

```
260     View.prototype.bind = function (event, handler) {
261         var self = this;
262         if (event === "newTodo") {
263             $on(self.$newTodo, "keyup", function (event) {
264                 if (event.keyCode === self.ENTER_KEY) {
265                     handler(self.$newTodo.value);
266                 }
267             });
268             $on(self.$additem, "click", function () {
269                 handler(self.$newTodo.value);
270             });
271         } else if (event === "removeCompleted") {
```

## Ajout d'une fonctionnalité

### Enregistrer des listes de tâches

- Afficher la liste des listes de tâches
- Ajouter / afficher / modifier / supprimer une liste de tâches
- Pouvoir mettre une date à la liste
- Une fois la date arrivée, les todos de la liste s'ajoutent aux todos existantes