Bug n°1 : faute de frappe dans controller.js.

```
/**
 * An event to fire whenever you want to add an item. Simply pass in the event
 * object and it'll handle the DOM insertion and saving of the new item.
 */
Controller.prototype.adddItem = function (title) {
    var self = this;

    if (title.trim() === '') {
        return;
    }

    self.model.create(title, function () {
        self.view.render('clearNewTodo');
        self._filter(true);
    });
};
```

```
/**
 * Evénement à déclencher lorsque vous souhaitez ajouter un élément. Il suffit de passer
 * dans l'objet événement et il va gérer l'insertion DOM et la sauvegarde du nouvel élément.
 * @param {string} (title) Le contenu du todo.
 */
Controller.prototype.addItem = function (title) { // ETAPE 1 : correction erreur nom de fonction
    var self = this;

    if (title.trim() === '') {
        return;
    }

    self.model.create(title, function () {
        self.view.render('clearNewTodo');
        self._filter(true);
    });
};
```

Bug n°2 : création des ID dans store.js.

```
Store.prototype.save = function (updateData, callback, id) {
    var data = JSON.parse(localStorage[this._dbName]);
    var todos = data.todos;

    callback = callback || function () {};

    // Generate an ID
    var newId = "";
    var charset = "0123456789";

    for (var i = 0; i < 6; i++) {
        newId += charset.charAt(Math.floor(Math.random() * charset.length));
    }

    // If an ID was actually given, find the item and update each property
    if (id) {
        for (var i = 0; i < todos.length; i++) {
            if (todos[i].id === id) {
                for (var key in updateData) {
                    todos[i][key] = updateData[key];
                }
                break;
            }
        }

        localStorage[this._dbName] = JSON.stringify(data);
        callback.call(this, todos);
    } else {

        // Assign an ID
        updateData.id = parseInt(newId);


        todos.push(updateData);
        localStorage[this._dbName] = JSON.stringify(data);
        callback.call(this, [updateData]);
    }
};
```

```
Store.prototype.save = function (updateData, callback, id) {
    var data = JSON.parse(localStorage[this._dbName]);
    var todos = data.todos;

    callback = callback || function () {};

    // If an ID was actually given, find the item and update
    if (id) {
        for (var i = 0; i < todos.length; i++) {
            if (todos[i].id === id) {
                for (var key in updateData) {
                    todos[i][key] = updateData[key];
                }
                break;
            }
        }

        localStorage[this._dbName] = JSON.stringify(data);
        callback.call(this, todos);
    } else {

        // Assign an ID
        updateData.id = Date.now();

        todos.push(updateData);
        localStorage[this._dbName] = JSON.stringify(data);
        callback.call(this, [updateData]);
    }
};
```

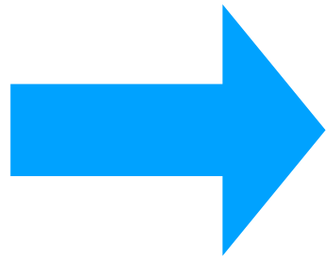
Amélioration n°1 : utilisation de la méthode trim().

```
 */
Controller.prototype.editItemSave = function (id, title) {
    var self = this;

    while (title[0] === " ") {
        title = title.slice(1);
    }

    while (title[title.length-1] === " ") {
        title = title.slice(0, -1);
    }

    if (title.length !== 0) {
        self.model.update(id, {title: title}, function () {
            self.view.render('editItemDone', {id: id, title: title});
        });
    } else {
        self.removeItem(id);
    }
};
```

```
/*
 * Termine le mode d'édition d'élément et élimine les espaces.
 * @param {number} (id) L' ID du model éditer à sauvegarder.
 * @param {string} (title) Le contenu du todo.
 */
Controller.prototype.editItemSave = function (id, title) {
    var self = this;

    title = title.trim();

    if (title.length !== 0) {
        self.model.update(id, {title: title}, function () {
            self.view.render('editItemDone', {id: id, title: title});
        });
    } else {
        self.removeItem(id);
    }
};
```

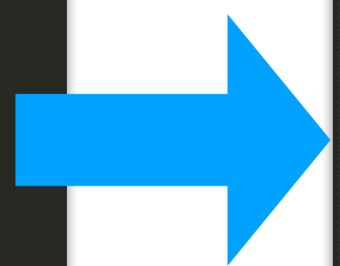
Amélioration n°2 : suppression des boucles inutiles.

```
/**
 * An event to fire whenever you want to add an it
 * object and it'll handle the DOM insertion and s
 */
Controller.prototype.adddItem = function (title) {
    var self = this;

    if (title.trim() === '') {
        return;
    }

    self.model.create(title, function () {
        self.view.render('clearNewTodo');
        self._filter(true);
    });
};
```



```
/**
 * Evénement à déclencher lorsque vous souhaitez a
 * dans l'objet événement et il va gérer l'inserti
 * @param {string} (title) Le contenu du todo.
 */
Controller.prototype.addItem = function (title) {
    var self = this;

    if (title.trim() !== '') {
        self.model.create(title, function() {
            self.view.render('clearNewTodo');
            self._filter(true);
        });
    }
};
```

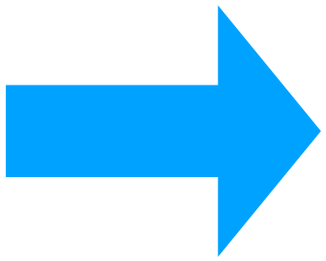
```
Store.prototype.find = function (query, callback) {
    if (!callback) {
        return;
    }

    var todos = JSON.parse(localStorage[this._dbName]).todos;

    callback.call(this, todos.filter(function (todo) {
        for (var q in query) {
            if (query[q] !== todo[q]) {
                return false;
            }
        }
        return true;
    }));
};
```



```
Store.prototype.find = function (query, callback) {
    if (callback) {
        var todos = JSON.parse(localStorage[this._dbName]).todos;
        callback.call(
            this,
            todos.filter(function(todo) {
                for (var q in query) {
                    if (query[q] !== todo[q]) {
                        return false;
                    }
                }
                return true;
            })
        );
    }
};
```

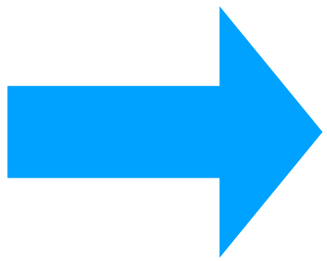
Amélioration n°2 (suite) : suppression des boucles inutiles.

```
View.prototype._elementComplete = function (id, completed) {
    var listItem = qs('[data-id="' + id + '"]');

    if (!listItem) {
        return;
    }

    listItem.className = completed ? 'completed' : '';

    // In case it was toggled from an event and not by clicking
    qs('input', listItem).checked = completed;
};
```



```
View.prototype._elementComplete = function (id, completed) {
    var listItem = qs('[data-id="' + id + '"]');

    if (listItem) {
        listItem.className = completed ? 'completed' : '';
        // On définit la tâche comme terminée par défaut
        qs('input', listItem).checked = completed;
    }
};
```

```
View.prototype._editItem = function (id, title) {
    var listItem = qs('[data-id="' + id + '"]');

    if (!listItem) {
        return;
    }

    listItem.className = listItem.className + ' editing';

    var input = document.createElement('input');
    input.className = 'edit';

    listItem.appendChild(input);
    input.focus();
    input.value = title;
};
```



```
View.prototype._editItem = function (id, title) {
    var listItem = qs('[data-id="' + id + '"]');

    if (listItem) {
        listItem.className = listItem.className + ' editing';
        var input = document.createElement('input');
        input.className = 'edit';
        listItem.appendChild(input);
        input.focus();
        input.value = title;
    }
};
```

Amélioration n°3 : remplacement des conditions if/else par un switch.

```
View.prototype.bind = function (event, handler) {
    var self = this;
    if (event === 'newTodo') {
        $on(self.$newTodo, 'change', function () {
            handler(self.$newTodo.value);
        });

    } else if (event === 'removeCompleted') {
        $on(self.$clearCompleted, 'click', function () {
            handler();
        });

    } else if (event === 'toggleAll') {
        $on(self.$toggleAll, 'click', function () {
            handler({completed: this.checked});
        });

    } else if (event === 'itemEdit') {
        $delegate(self.$todoList, 'li label', 'dblclick', function () {
            handler({id: self._itemId(this)});
        });

    } else if (event === 'itemRemove') {
        $delegate(self.$todoList, '.destroy', 'click', function () {
            handler({id: self._itemId(this)});
        });

    } else if (event === 'itemToggle') {
        $delegate(self.$todoList, '.toggle', 'click', function () {
            handler({
                id: self._itemId(this),
                completed: this.checked
            });
        });

    } else if (event === 'itemEditDone') {
        self._bindItemEditDone(handler);

    } else if (event === 'itemEditCancel') {
        self._bindItemEditCancel(handler);
    }
};
```

```
View.prototype.bind = function (event, handler) {
    var self = this;

    switch (event) {
        case 'newTodo':
            $on(self.$newTodo, 'change', function() {
                handler(self.$newTodo.value);
            });
            break;
        case 'removeCompleted':
            $on(self.$clearCompleted, 'click', function() {
                handler();
            });
            break;
        case 'toggleAll':
            $on(self.$toggleAll, 'click', function() {
                handler({ completed: this.checked });
            });
            break;
        case 'itemEdit':
            $delegate(self.$todoList, 'li label', 'dblclick', function() {
                handler({ id: self._itemId(this) });
            });
            break;
        case 'itemRemove':
            $delegate(self.$todoList, '.destroy', 'click', function() {
                handler({ id: self._itemId(this) });
            });
            break;
        case 'itemToggle':
            $delegate(self.$todoList, '.toggle', 'click', function() {
                handler({
                    id: self._itemId(this),
                    completed: this.checked,
                });
            });
            break;
        case 'itemEditDone':
            self._bindItemEditDone(handler);
            break;
        case 'itemEditCancel':
            self._bindItemEditCancel(handler);
            break;
    }
};
```

Les + :

- meilleure lisibilité,
- meilleure maintenabilité du code,
- amélioration des performances.