**Documentation technique**

Ce document aborde la structure globale du programme ainsi que son fonctionnement

**Prototype Objet**

Ce programme utilise le mécanisme des prototypes qui permet aux objets JavaScript d'hériter des propriétés d'autres objets.

**Structure Générale**

Ce programme a une structure de type utilise le MVC (Model-Vue-Controller)

La structure principale de notre page HTML (DOM) existe ici à travers le fichier template. Toutes les interactions qui découlent à l’intégration de contenu html transitent par ce fichier en interdépendance avec les trois cités au dessus

* Le modèle gère les données, et fournira essentiellement des informations sur ce qu'il faut insérer dans nos pages, ou donnera des données que le contrôleur peut utiliser. Dans notre cas, un fichier store se charge du stockage des données
* Une vue contient la présentation de l'interface graphique.
* Le contrôleur contient les actions effectuées par l'utilisateur.
* En fonction des décisions du contrôleur, la vue rendra notre modèle.

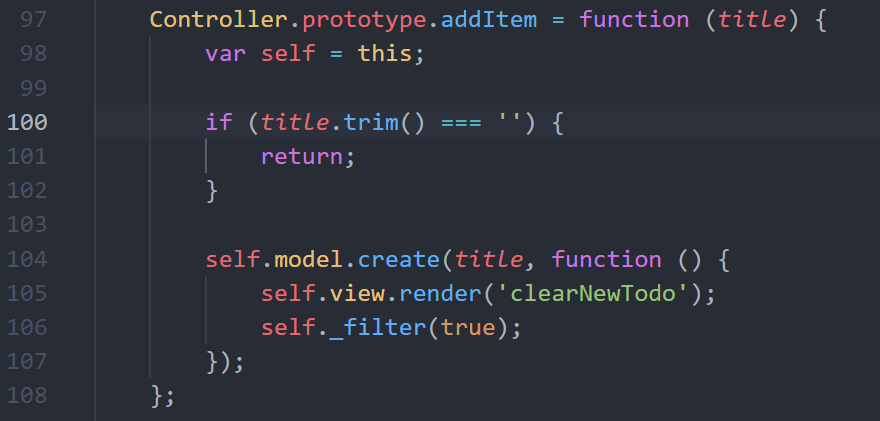
Ce système permet d'éviter que la base du programme soit surchargée, elle reste parfaitement compréhensible, maintenable et évolutive.

**Création d’une tâche et flux de données**

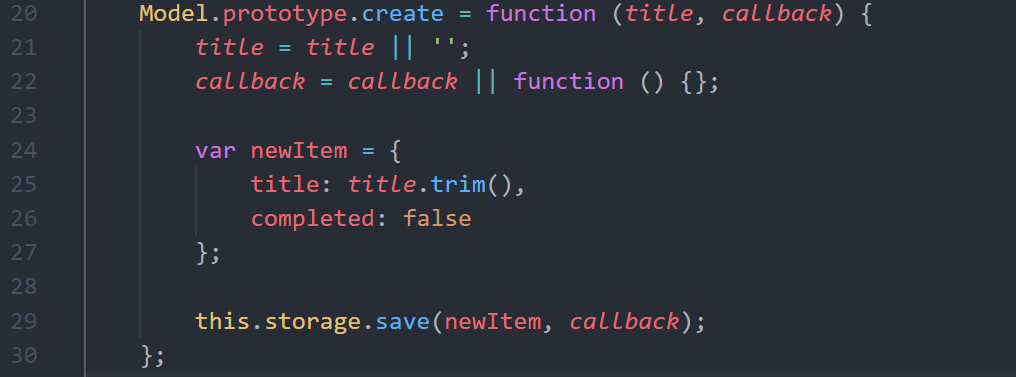
L’ajout d’une tâche à la liste des tâches nécessite une interaction avec notre vue et le modèle.

Lorsque la tâche est créée, notre contrôleur reçoit le nom de la tâche et ordonne au modèle de créer un nouvel élément, qui mettra à jour le store.

***La méthode “addItem” dans controller.js***



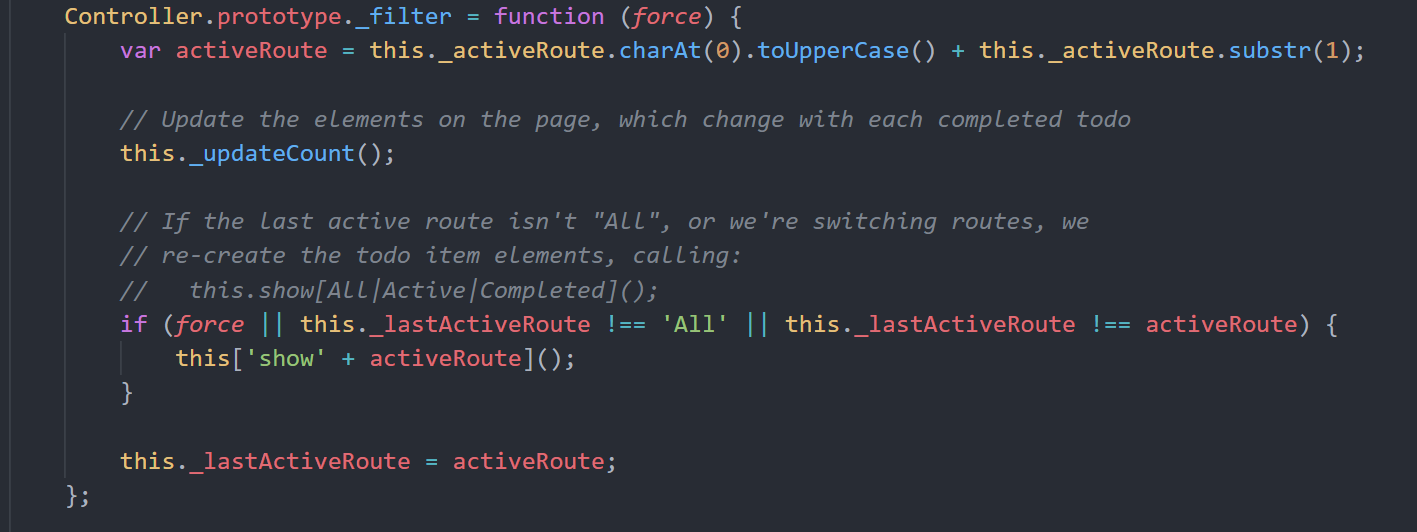
***La méthode “create” dans model.js***



Lorsque le store a été mis à jour, le contrôleur indique à la vue de vider l'entrée, parce qu'elle est maintenant stockée dans notre store.

le contrôleur appelle enfin sa propre méthode filter avec un argument égal à true, ce qui va forcer une recréation des todos en fonction de la route active :

**la méthode “filter” dans controller.js**



En savoir plus : https://developer.mozilla.org/fr/docs/Glossary/MVC