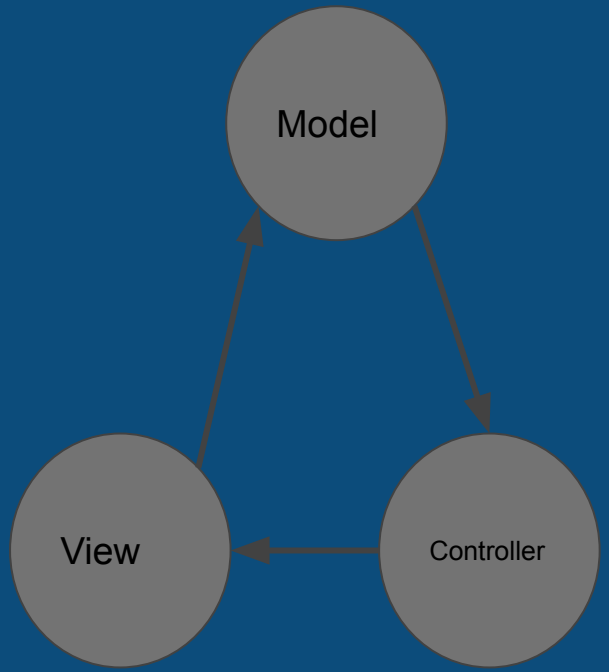
Documentation technique

Projet 8

Reprenez et améliorez un projet existant

Sonia Panisello
Parcours Développeur d’Applications Front End

MVC



Modèle

données du programme (algo, configuration, logique)

Vue

interface homme machine (IHM)

Contrôleur

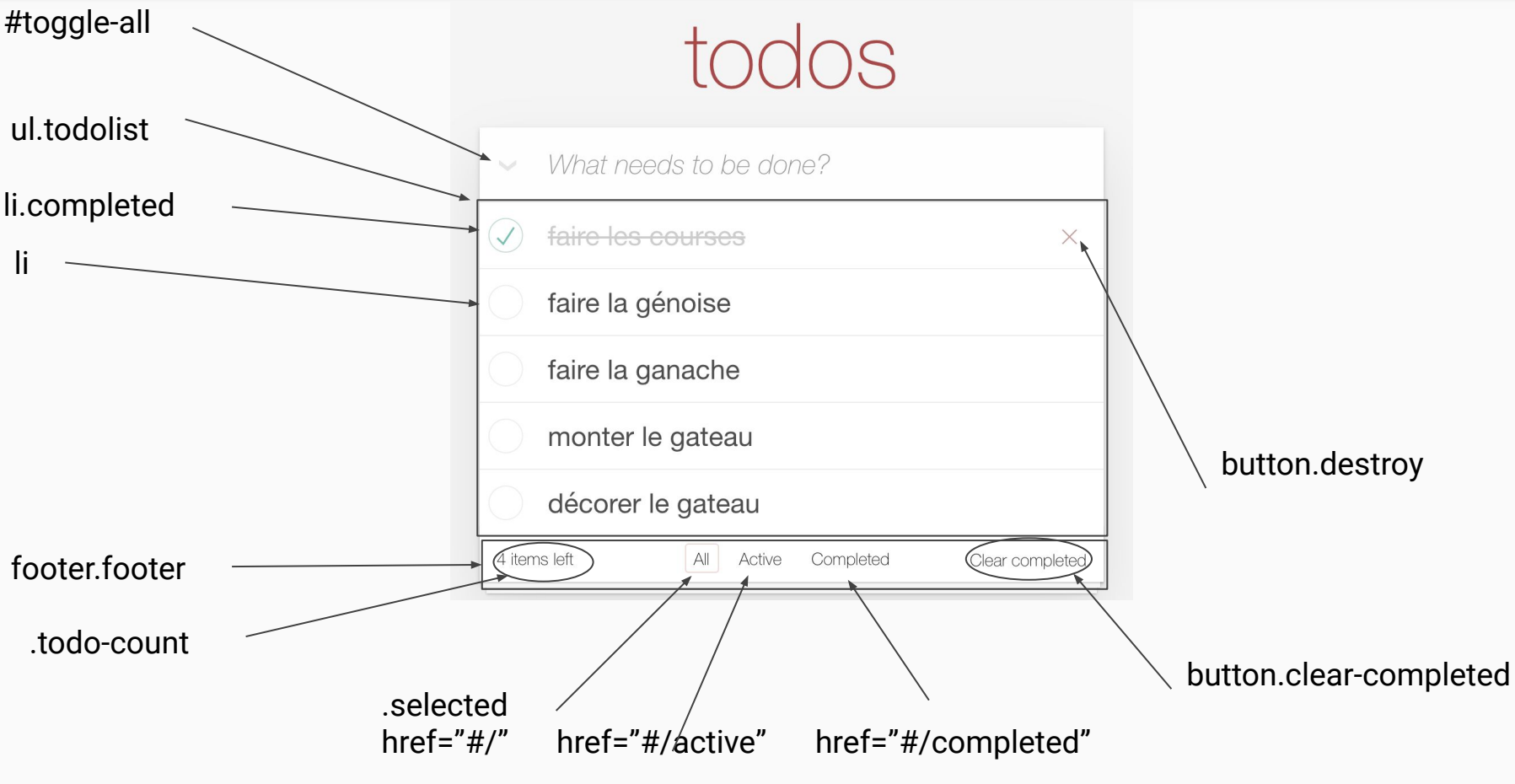
contrôle les données
transmet les infos entre la vue et le modèle
lorsque l'utilisateur effectue une action

Interface

h1

input.newtodo

footer.info


#toggle-all
ul.todolist
li.completed
li
todos
What needs to be done?
faire les courses
faire la génoise
faire la ganache
monter le gateau
décorer le gateau
button.destroy
footer.footer
.todo-count
4 items left
All
Active
Completed
Clear completed
button.clear-completed
.selected
href="#/"
href="#/active"
href="#/completed"

Organisation des fichiers

index.html

mise en page

scripts

node_modules

test

ControllerSpec.js
SpecRunner.html

js/app.js

Configure une nouvelle liste

Charge et initialise la view

js/controller.js

actions sur les todos
affichage, supression
...

js/helpers.js

eventlistener DOM

js/model.js
Crée une nouvelle instance de modèle et connecte le stockage

js/store.js
Crée un nouvel objet de stockage côté client et créera une collection vide si aucune collection n'existe déjà.

js/template.js
template HTML
modification d'élément html

js/view.js
Affichage
Vue qui résume complètement le DOM du navigateur.

POO class et méthodes associés : app.js

Class Todo

storage
model
template
view
controller

setView()

Todo : Configure une nouvelle liste Todo.

Prend en paramètre le nom de la nouvelle liste : name

Définit les paramètres :

storage : nouvel objet Store

model : nouvel objet Model

template : fonction Template() attribut le template par défaut

view : fonction View() prend en paramètre template

controller : nouvel objet Contrôler qui prend en paramètre model et view

déclaration de la nouvelle Todo 'todos-vanillajs'

setView : hash partie de l'Url qui suit le symbole # sur deux évènements (chargement de la page et changement d'url)

POO class et méthodes associés : controller.js

Class Controller

contrôle les données
transmet les infos entre la vue et le modèle
lorsque l'utilisateur effectue une action

model
view

view.bind.addItem(title)
view.bind.editItem(item.id)
view.bind.editItemSave(item.id, item.title)
view.bind.editItemCancel(item.id)
view.bind.removeITem(item.id)
view.bind.toggleComplet(item.id, item.completed)
view.bind.toggleAll(status.completed)

POO class et méthodes associés : controller.js

Class Controller

setView() => Charge et initialise la vue, prends en paramètre '' | 'active' | 'completed'
showAll() =>Obtiendra tous les éléments et les affichera dans la todo list
showActive() => affiche toutes les tâches actives, propriété du model { completed: false }
showCompleted() => affiche toutes les tâches terminées, propriété du model { completed: true }
addItem() => se déclencher chaque fois qu'on ajoute un élément. gére l'insertion DOM et l'enregistrement du nouvel élément.
editItem()=> Déclenche le mode d'édition des éléments.
editItemSave()=> Termine le mode d'édition d'élément avec succès.
editItemCancel() => Annule le mode d'édition des éléments.
removeItem() => prends en paramètre l'ID En lui donnant un ID, il trouvera l'élément DOM correspondant à cet ID, le supprime du DOM et le supprime également du stockage.
removeCompletedItems()=> Supprime tous les éléments terminés du DOM et du stockage.
toggleComplete()=>prends en paramètre l'ID d'un modèle et une case à cocher et il mettra à jour l'élément en stockage en fonction de l'état de la case à cocher.
toggleAll()=> coche ou décoche toutes les cases
updateCount()=> met à jour le nombre d'élément
filter()=>filtre les éléments
updatefilterState()=>

POO class et méthodes associés : helper.js

window

window.qs : retourne le premier élément dans le document correspondant au sélecteur CSS
window.qsa : retourne tous les éléments dans le document correspondant au sélecteur CSS
vindow.$on :
window.$delegate :
window.$parent : retourne le parent de l'élément avec le nom de balise donné

POO class et méthodes associés : model.js

class Model

Model Crée une nouvelle instance de model et connecte le stockage. prends en paramètre storage (Une référence à la classe de stockage côté client)

create()=> crée une nouvelle tâche
read()=> recherche et renvoie une tâche
update()=> met à jour la tâche
remove() => supprime une tâche
removeAll() => supprime toutes les tâches
getCount()=> Compte le nombre de tâches

POO class et méthodes associés : store.js

class Store

Store => crée un nouvel élément de stockage des données

find()=> Recherche des éléments en fonction d'une requête donnée en tant qu'objet JS
findAll()=> Récupère toutes les données de la base de données
save()=> Enregistre les données fournies dans la base de données.
drop() => Supprime un élément de la base de données en fonction de son ID

POO class et méthodes associés : template.js

class Template gestion du code html

Template => crée un template par défault

show()=> remplace les variables du template par les données de la nouvelle tâche
itemCounter() => Affiche un compteur du nombre de tâches restantes à terminer
clearCompletedButton()=> Met à jour le texte dans le bouton "Clear completed"

POO class et méthodes associés : view.js

class View gestion de l'interface

View => crée le template et cible ces éléments via des class CSS

_removeItem()=>supprime une tâche
_clearCompletedButton() => affiche ou cache le bouton "Clear Completed"
_setFilter()=> en fonction de la page affichée va entouré la catégorie des tâches affichées (All, Completed, Active)
_elementComplete()=> met à jour l'affiche de la tâche si elle est terminée ou pas
_editItem()=> change l'affichage d'une tâche lorsqu'elle est en cours de modification
_editItemDone()=> reprends un affichage normal

POO class et méthodes associés : view.js

class View gestion de l'interface

render()
- showEntries => affiche la tâche
- removeItem => supprime la tâche
- updateElementCount => met à jour le comptage des tâches
- clearCompletedButton => affiche ou pas le bouton "Clear completed"
- contentBlockVisibility => affiche ou pas le footer
- toggleAll =>
- setFilter =>
- clearNewTodo =>
- elementComplete =>
- editItem =>
- editItemDone =>

_itemID()=> retourne l'ID d'un tâche
_bindItemEditDone() =>
_bindItemEditCancel() =>
bind() =>