AngularJS - Jour 3

0 - Programme

Au programme:

- Les filters
- TP
- Le routing
- Les directives
- TP 2

1 - Les filtres

Les filtres natifs AngularJS

Les filtres sont des “pure functions” qui permette de muter une variable passée en paramètre.
Beaucoup de filtre fournis avec AngularJS : currency, date, filter, json, limitTo, lowercase, number, orderBy, uppercase

[DEMO/day_02/step_06]

Création de filtre personnalisé

On crée un filter comme on crée un controlleur.
app.filter(‘NameFilter’, function() {});
Un filtre est une fonction, qui prend x paramètre en entré et retourne une valeur.
[DEMO/day_02/step_07]

2 - TP

Création d’un projet depuis zéro

1. Vérifier les version de node & npm
2. Créer un nouveau dossier du nom de votre projet et vous rendre à l’intérieur
3. Initialiser le package.json (*npm init*)
4. Installer les dépendances (angular, angular-mocks?)
5. Dans votre dossier
    a. Créer un dossier **src/**
    b. Créer deux fichiers dans ce dossier **src/**: *index.html* et *app.js*
    c. Créer un dossier **test/**
    d. Créer un fichier de spécification dans **test/**: app.spec.js

Création d’une application AngularJS de base

1. Créer une application angular nommée “awesomeApp”
2. Créer un controller nommé “awesomeAppCtrl” avec:
    a. Une variable string avec le titre de la page
    b. Une variable avec la date du jour
3. Créer le code HTML pour afficher les valeurs

Gestion d’utilisateur

1. Créer une liste d’utilisateur (tableau d’objet User)
   User: firstname, lastname, email, birthday_day, birthday_month, birthday_year, avatar_url
2. Créer une méthode pour ajouter un User à la liste
3. Créer une méthode pour supprimer un User par indice dans le tableau
4. Afficher un formulaire pour ajouter un utilisateur (avec affichage des erreurs)
5. Afficher la liste des utilisateurs

Bonus

1. Utiliser des filtres natif
2. Créer des filtres personnalisés

3. Créer des watchers
4. Utiliser du code asynchrone (timeout, requête AJAX)

3 - Le routing

Routing AngularJS

Documentation officiel : https://docs.angularjs.org/api/ngRoute
Manque de fonctionnalité.
Module externe à angular (**angular-route**).
Inclusion via une dépendance module ['ngRoute']
Expose des composants AngularJS:

- provider ($routeProvider)
- directive (ngView)
- services ($route, $routeParams)

Définition d'une route (link):

- path (pattern url)
- route config (controller, template, resolve, etc.)

[DEMO/day_03/step_03]

Routing UI-router

Projet Github : https://github.com/angular-ui/ui-router
Pourquoi utiliser ui-router ?

- Les vues imbriquées
- Les vues multiples nommées
- Liens générique vers un state (et non une url en dur)
- Decorator pour avoir des urls dynamique
- States (Au lieu des routes)

Documentation :

API Reference: http://angular-ui.github.io/ui-router/site/#/api
Guide: https://github.com/angular-ui/ui-router/wiki
FAQs: https://github.com/angular-ui/ui-router/wiki/Frequently-Asked-Questions
Sample Application: http://angular-ui.github.io/ui-router/sample/#/

4 - Les componsants

Kezako

C'est la base du framework AngularJS.
Simplement : Un composant est une function JavaScript qui manipule et ajoute de nouveaux comportements au DOM HTML.
Elles peuvent être simple ou extrêmement compliqué.

Comment AngularJS détecte les composants

Le template HTML peut invoquer une directive AngularJS de quatre manières :

- En tant qu'attribut
  <span my-directive>
- En tant que class CSS

- En tant qu'element DOM
  <my-directive></my-directive>
- En tant que commentaire
  <!-- directive: my-directive expression →

La création
https://docs.angularjs.org/guide/component

Les options

restrict
Elle permet de définir à AngularJS le déclencheur dans le HTML.
'A' : Attribut
```
<span my-directive></span>
```
'E' : Element
```
<my-directive></my-directive>
```
'C' : Classe
```
<span class="my-directive"></span>
```
'M' : Commentaire
```
<!-- directive: my-directive -->
```

Template
Permet de défnir un template HTML directement dans la directive
```
template: '<div class="myclass"></div>'
```

TemplateUrl
Permet de définir une url de template HTML à utiliser
```
templateUrl: 'templates/ng-sparkline-template.html'
```

Bindings
Le scope permet de définir l'héritage.
Toujours un scope isolé.
Pour mettre en place une discussion avec l'extérieur on a la possibilité de lui spécifier un comportement par valeur.
Scope local
@ (or @attr)
Bi-directionel
= (or =attr)
Le context parent
& (or &attr)

Controller
Permet de définir un contrôleur

ControllerAs (optionnel depuis 1.5)
Permet de nommer le contrôleur sur le scope

Les étapes d'initialisation d'une directive

La compilation ($compile, compile)
Elle s'occupe de la manipulation du DOM avant d'effectuer le rendu.

Manipulation DOM = lent

Le controller

S'occupe de la logique présentation.
Permet d'exposer une API pour cette directive et d'autoriser la communication avec des directives soeur et parente.

[DEMO/day_03/step_04]
[DEMO/day_03/step_05]

5 - TP 2

Implementer un router

- Ajouter la dépendance (*angular-route, angular-ui-router)*
- Utiliser cette dépendance dans l'application
    - Dépendance module
    - Directive de vue
    - Directive de liens
- Configurer les routes suivantes:
    - / ou /home ⇒ Accueil
    - /users ⇒ Page listant les utilisateurs
    - /users/new ⇒ Page avec le formulaire
    - /users/[ID] ⇒ Page avec le détail d'un utilisateur

Ranger le code en composant

/src
-- /components
-- -- /user
/ controller, filter, directives directement reliés à un utilisateur */*
-- -- /home
/ controller, filter, directives directement reliés à l'accueil */*
-- -- /misc or /common
/ controller, filter, directives de type tooling/common */*
-- index.html
-- app.js

Aller plus loin: ui-router

Utiliser les fonctionnalités avancées de ui-router (https://github.com/angular-ui/ui-router).

- Vues imbriquées
- Vues multiples
- Etats parent/enfant