

AngularJS - Jour 2

0 - Programme

1 - Structure et core concepts

Architecture MV*

Scope

Contrôleur

Les vues

Routing

Components in AngularJS 1.4 Router

Data binding

Directives

Filters

Services

L'injection dépendance

Les modules

La communication avec le server

Le testing

AngularJS fournit ngMock qui contient des utilitaires pour les tests.

e2e : Tester l'application end-to-end, en naviguant via HTTP sur le site et en simulant un utilisateur.

2 - Le Scope

L'héritage par prototypage dans JavaScript

L'héritage par prototypage dans AngularJS

ControllerAs syntax

Héritage des types primitifs

[DEMO/day_02/step_01]

3 - Le cycle de vie

\$apply et \$digest

[DEMO/day_02/step_02]

Quand appeler \$apply manuellement

Performance

L'avenir du dirty-checking

Lien utile

4 - Les templates

Les directives

Les formulaires

5 - Les filtres

Les filtres natifs AngularJS

Création de filtre personnalisé

6 - TP

Création de l'architecture

Création d'une application AngularJS de base

Gestion d'utilisateur

Bonus

0 - Programme

Au programme:

- Structure et core concepts
- Le scope
- Le cycle de vie AngularJS
- Les templates (directive, formulaire)
- Les filtres
- TP

1 - Structure et core concepts

Architecture MV*

Explication évolution modèle MVC / MVVM / MVW ou MV* (schémas).

<https://plus.google.com/+AngularJS/posts/aZNVhj355G2>

Scope

Le scope représente la closure JavaScript qui hydrate la vue.

C'est la sortie du contrôleur et l'entrée de la vue.

Contrôleur

Le contrôleur : logique de la vue (Fonction JavaScript).

Peut-être utilisé dans les directives, les composants ou directement dans une vue.

Les vues

Les vues sont des pages HTML.

Elles peuvent être intégrées dans l'application par différents moyens:

- Directive built-in
- Custom directive
- Routing

Elles permettent d'utiliser les directives built-in ou custom ainsi que les expressions angularJS.

Routing

Il n'y a pas de SPA sans routing.

La version AngularJS 1.4 apporte un nouveau router qui utilise des composants.

Components in AngularJS 1.4 Router

Composants dans AngularJS sont définies par un *nom*.

Composants = template + controller

Data binding

AngularJS permet de reporter automatiquement les changements d'un objet du model à la vue et vice-versa.

Directives

Ce sont des marqueurs HTML (comme les éléments, attributs, classes CSS).

Ils fonctionnent comme des Web Components.

Beaucoup de directive native dans AngularJS (ng-repeat, ng-if, etc.)

Filters

Les filters sont des "pure functions" et permettent de modifier une valeur.

Services

factory, service, provider, constant, value

Les services sont des singletons pour l'application.

Ils portent la partie logique métier de l'application.

Beaucoup de service natifs dans AngularJS (\$q, \$http, \$timeout, etc)

L'injection dépendance

Dependency Injection

Le moyen pour angularJS d'injecter un service par rapport à son nom.

Les modules

angular.module

AngularJS permet de créer des "package" avec la notion de module.

[EX:archi module]

La communication avec le server

Une application côté client permet de fluidifier l'utilisation du programme mais les données doivent être récupérées sur le server.

AngularJS met à disposition 2 services \$http (requête XHR classique) & \$resource (requête XHR pour une API REST).

Le testing

AngularJS fournit ngMock qui contient des utilitaires pour les tests.

Deux types de tests: TU (karma) & e2e (protractor)

TU : Isolé des petites parties de code pour les tester

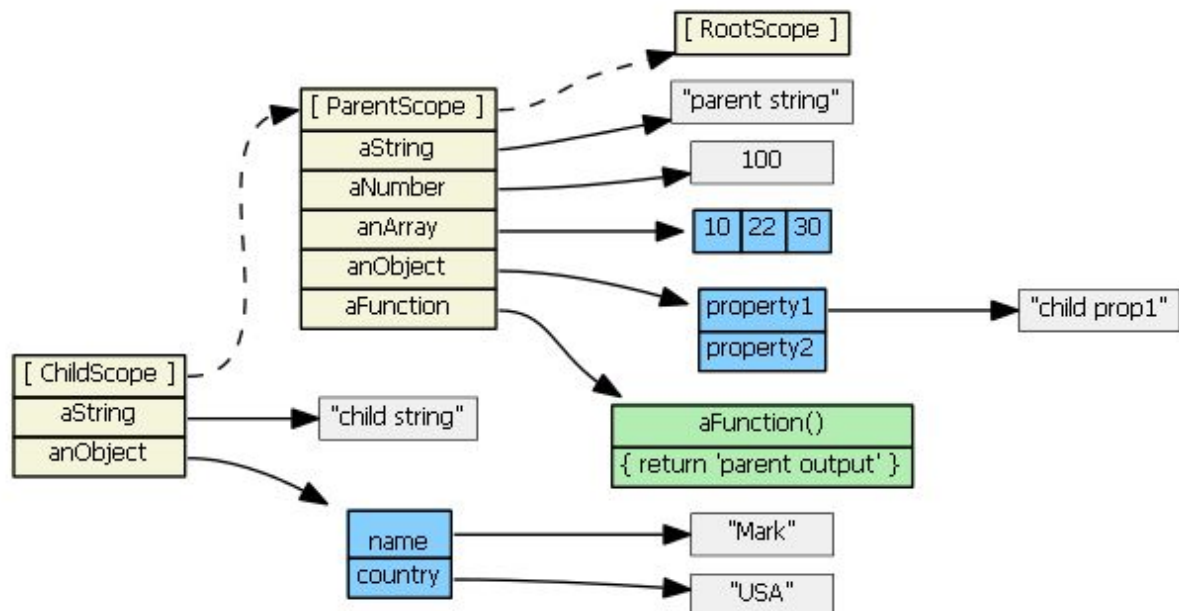
e2e : Tester l'application end-to-end, en naviguant via HTTP sur le site et en simulant un utilisateur.

Le testing: Démo

- Installation de karma-cli au global
- Installation de karma, karma-chrome-launcher, karma-jasmine, angular, angular-mocks
- Initialise karma en CLI: karma init
- Renseigner les fichiers à écouter: 1.Angular, 2.Mocks, 3.App, 4.Testing
- Ecrire des tests
- Lancer les tests: karma start myConfigFile.js

2 - Le Scope

L'héritage par prototypage dans JavaScript



Ex:

```
childScope.aString === 'parent string'
childScope.anArray[1] === 20
childScope.anObject.property1 === 'parent prop1'
childScope.aFunction() === 'parent output'
```

L'héritage par prototypage dans AngularJS

Un Scope peut être créé de plusieurs manière:

- Normal (héritage: ON)
- Isolé (aucun héritage)

Toujours un rootScope (accessible via le service \$rootScope).

ControllerAs syntax

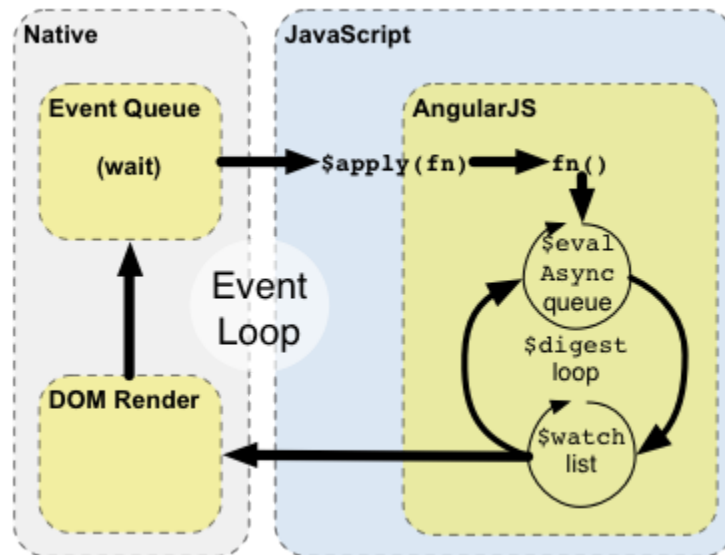
L'intérêt de cette syntax est d'éviter l'héritage et de pouvoir accéder à tous les éléments.

Héritage des types primitifs

/!\ ng-include, ng-switch & ng-repeat et l'utilisation des cas primitifs (String, Number)

[DEMO/day_02/step_01]

3 - Le cycle de vie



Dirty-checking => \$digest

Update AngularJS Application => \$apply

\$apply et \$digest

AngularJS permet le two-way data binding.

Cela veut dire que l'on peut écouter tout changement sur un élément du Scope avec la méthode `$scope.$watch`.

[DEMO/day_02/step_02]

Quand appeler \$apply manuellement

Cette méthode est à appeler quand on veut utiliser du code Non AngularJS et impacter AngularJS.

Ex: tout code asynchrone `setTimeout`, request AJAX

[DEMO/day_02/step_03]

Performance

Les humains sont:

- Lent : <50ms est imperceptible et peut être considérée comme "instantané"
- Limité : Ne pas afficher plus de 2000 informations sur une même page (mauvaise UI)

Benchmarks (jsperf):

<http://jsperf.com/angularjs-digest/6> => 10k watchers

L'avenir du dirty-checking

Object.observe prévu pour ES7

Lien utile

<https://www.youtube.com/watch?v=Mk2WwSxK218>

4 - Les templates

Les directives

ng-if, ng-show, ng-class, ng-src, ng-click, ng-copy, ng-paste, ng-href, ng-switch, etc...

[DEMO/day_02/step_04]

Les formulaires

HTML5 Validation API

HTML5 Attribute	ng Attribute	Registered Error
required="bool"	ng-required="..."	ngModel.\$error.required
minlength="number"	ng-minlength="number"	ngModel.\$error.minlength
maxlength="number"	ng-maxlength="number"	ngModel.\$error.maxlength
min="number"	ng-min="number"	ngModel.\$error.min
max="number"	ng-max="number"	ngModel.\$error.max
pattern="patternValue"	ng-pattern="patternValue"	ngModel.\$error.pattern

<input type="...">	Registered Error
type="email"	ngModel.\$error.email
type="url"	ngModel.\$error.url
type="number"	ngModel.\$error.number
type="date"	ngModel.\$error.date
type="time"	ngModel.\$error.time
type="datetime-local"	ngModel.\$error.datetimelocal
type="week"	ngModel.\$error.week
type="month"	ngModel.\$error.month

[DEMO/day_02/step_05]

5 - 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]

6 - TP

Création de l'architecture

Nécessite nodejs, npm, bower

1. Créer un nouveau dossier
2. Initialiser Bower (bower.json)
3. Installer angular
4. Créer un dossier **src/**
5. Créer deux fichiers dans ce dossier **src/**: *index.html* et *app.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)