AngularJS - Jour 2

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 components ou directement dans une vue.

Les vues

Les vues sont des pages HTML.
Elles peuvent être intégré 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 components.

Components in AngularJS 1.4 Router

Components dans AngularJS sont définies par un *nom*.
Components = 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 elements, 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

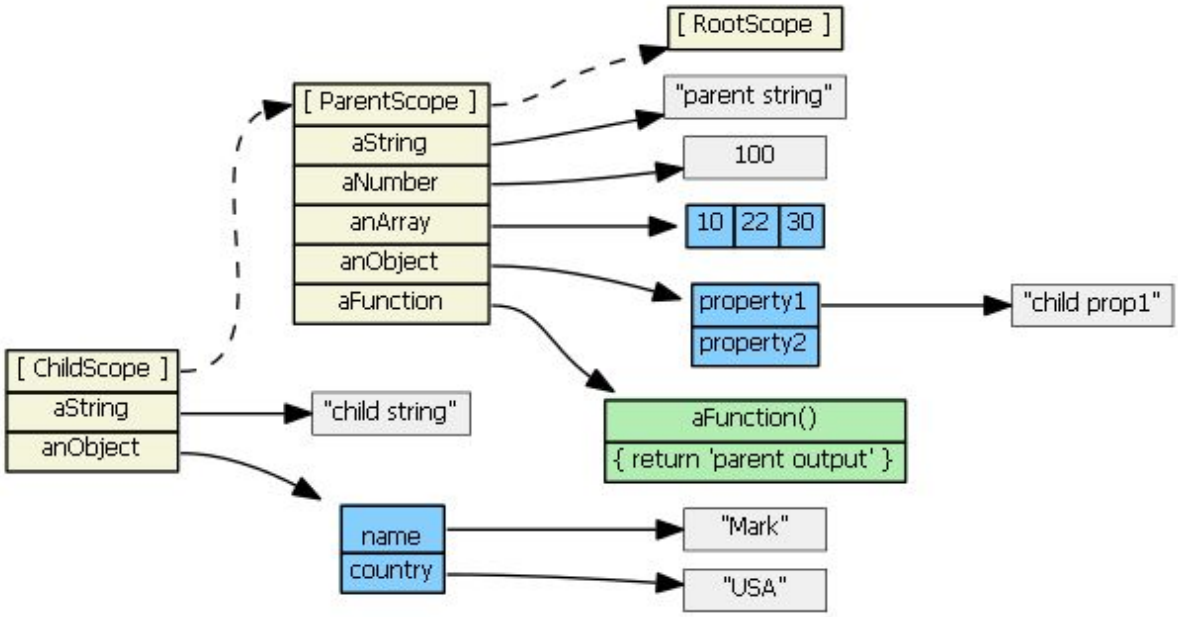
AngularJS fournit ngMock qui contient des utilitaires pour les tests.
Deux types de tests: TU (karma) & e2e (protractor)
TU : Isolé des petites parties de code pour les tester
e2e : Tester l'application end-to-end, en naviguant via HTTP sur le site et en simulant un utilisateur.

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

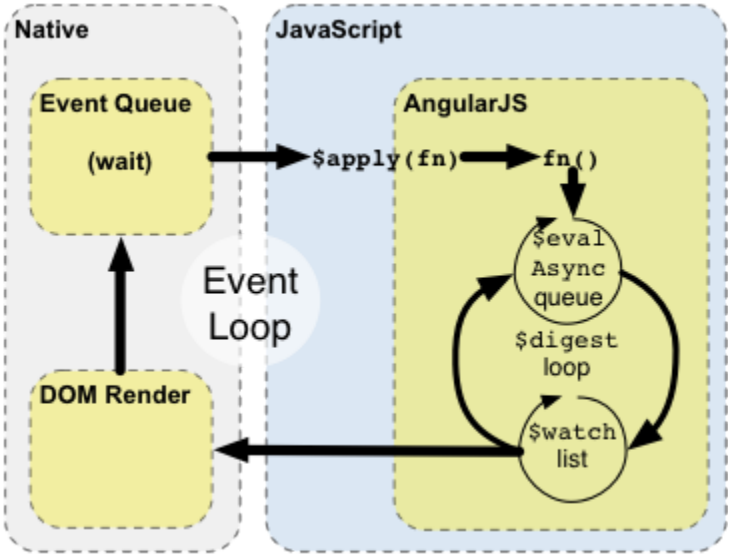
L'intéret de cette syntax est d'éviter l'héritage et de pouvoir accéder à tous les éléments.

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 considéeé comme "instantanné"
- 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]