AngularJS

Jour 1 - Introduction aux concepts AngularJS

Présentation

- Benjamin Longearet ou “Ben”
- Front-End Tech Lead @ Teads.tv
  - blongearet@gmail.com

@blongearet

https://github.com/firehist

GitHub

Teaser

- AngularJS
- Firebase
- NPM, Grunt, bower
- HTML5

1 - Introduction

Introduction

- AngularJS : Kezako

Introduction - AngularJS : Kezako

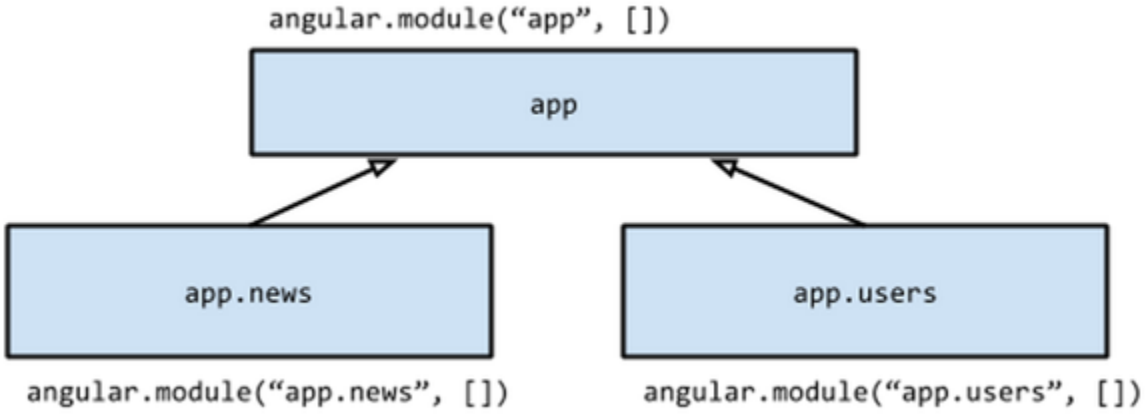
- Framework côté client
- Améliore les langages existants
- Indépendant
- Conception simple

Introduction

- AngularJS : Kezako
- Les modules

Introduction - Les modules

- Paquets de fonctionnalités
- Dépendances



Introduction

- AngularJS : Kezako
- Les modules
- Les templates

Introduction - Les templates

- HTML + AngularJS Expressions

Introduction

- AngularJS : Kezako
- Les modules
- Les templates
- Interactions

Introduction - Interactions

- Fluidité pour l’utilisateur
- Échange avec l’utilisateur
- Style & Animations

Introduction

- AngularJS : Kezako
- Les modules
- Les templates
- Interactions
- Formulaires

Introduction - Formulaires

- Suit les spécifications HTML5
- Gestion des erreurs

Introduction

- AngularJS : Kezako
- Les modules
- Les templates
- Interactions
- Formulaires
- Testing

Introduction - Testing

- Testable à 100%
- Très important
  - Refactoring
  - Automatisation
  - Sérénité

Introduction

- AngularJS : Kezako
- Les modules
- Les templates
- Interactions
- Formulaires
- Testing
- Les données

Introduction - Les données

- Dialogue avec le serveur
  - XHR / JSONP
- API
  - Custom, REST, JSON API v2, etc.

2 - Environnement de développement

Environnement de dév

NodeJS

- JavaScript côté serveur (nodejs.org)
- Pourquoi l’utiliser ?

Environnement de dév

NodeJS

- JavaScript côté serveur (nodejs.org)
- Pourquoi l’utiliser ?

Environnement de dév

NPM

- Node Package Manager (npmjs.org)
- Module JavaScript
- Version (semver.org)
- Arbre de dépendance

Environnement de dév

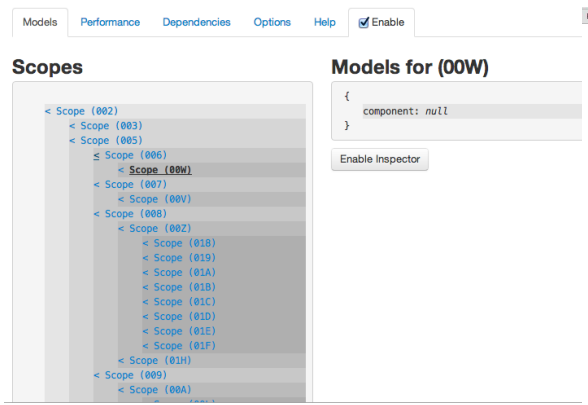
Bower

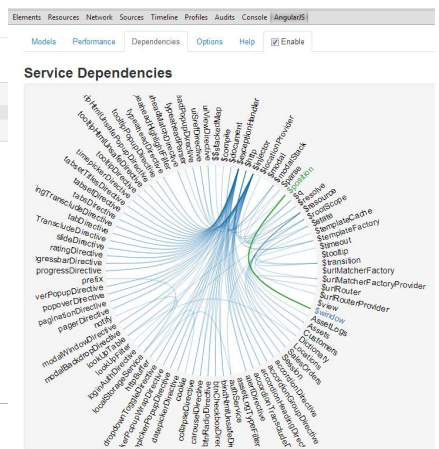
- Browser Package Manager (bower.io)
- NPM pour le navigateur
- Dépendance à plat

Environnement de dév

Batarag

- Chrome Extension (https://goo.gl/KWDnF6)

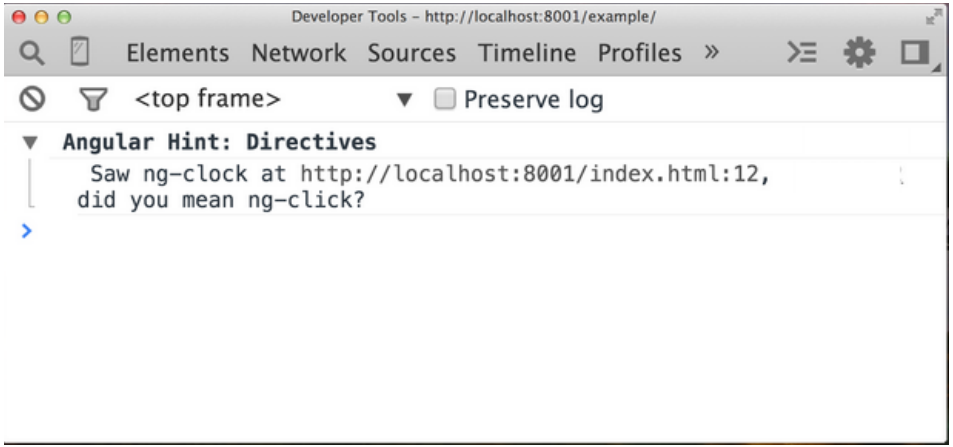




Environnement de dév

angular-hint

- JS Scripts (https://goo.gl/2urDTH)



Environnement de dév

IDE

- Sublim Text
- Webstorm
- Netbeans

Environnement de dév

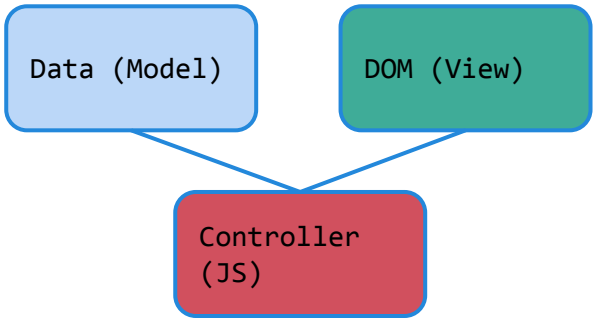
Lib JS

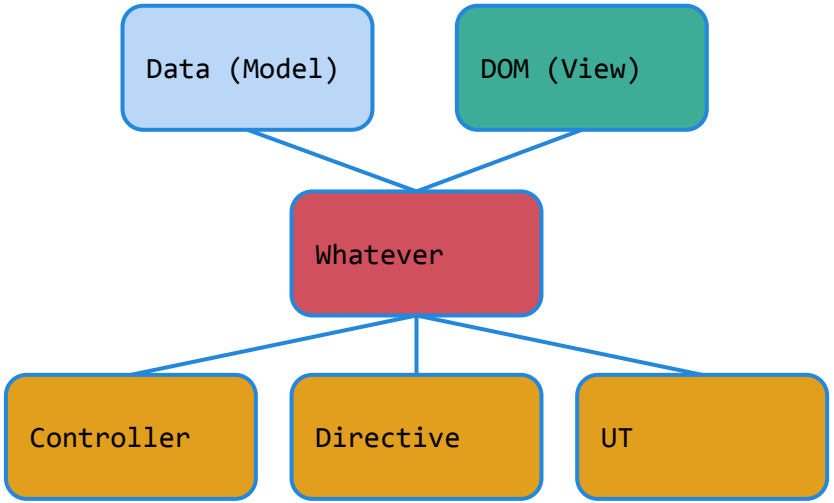
- Lodash
- ui-router
- angular-translate

3 - Structure et core concepts

Architecture MVW

- MVC vs MVW (https://plus.google.com/+AngularJS/posts/aZNVhj355G2)





Architecture MVW

Data (Model)

Structure du modèle métier

```
this.model = {
  "firstname": "Benjamin",
  "lastname": "Longearet"
};
```

DOM (View)

Représentation (HTML)

```
<div ng-app>
  {{model.firstname}}
</div>
```

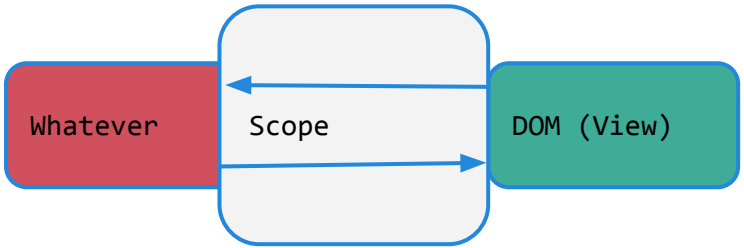
Contrôleur

Whatever - Code JS qui contrôle le flow de l'application

```
var myWhatever = function (myService) {
  this.user = myService.getUsers();
};
```

Le scope

- Responsable du dirty-checking
- Closure entre la vue et le contrôleur



Le contrôleur

- Logique de la vue

```
var myApp = angular.module('spicyApp1', []);
myApp.controller('SpicyController', ['$scope', function($scope) {
	this.spice = 'very';

	this.chiliSpicy = function() {
		this.spice = 'chili';
	};

	this.jalapenoSpicy = function() {
		this.spice = 'jalapeño';
	};
}]);
```

Les vues

- HTML amélioré
- Utilisées à différents niveaux :
  - Directive built-in
  - Directive personalisée
  - Routing

```
<div ng-app>
  {{model.firstname}}
  <input type="text" ng-model="model.firstname" />
</div>
```

Le routing

- Routing AngularJS
  - Trop simple, vieux (AngularJS 1.0)
- UI-Router
  - Multi-vues
  - Héritages
  - Built-in directive
- Nouveau router AngularJS (beta - AngularJS 2.0)

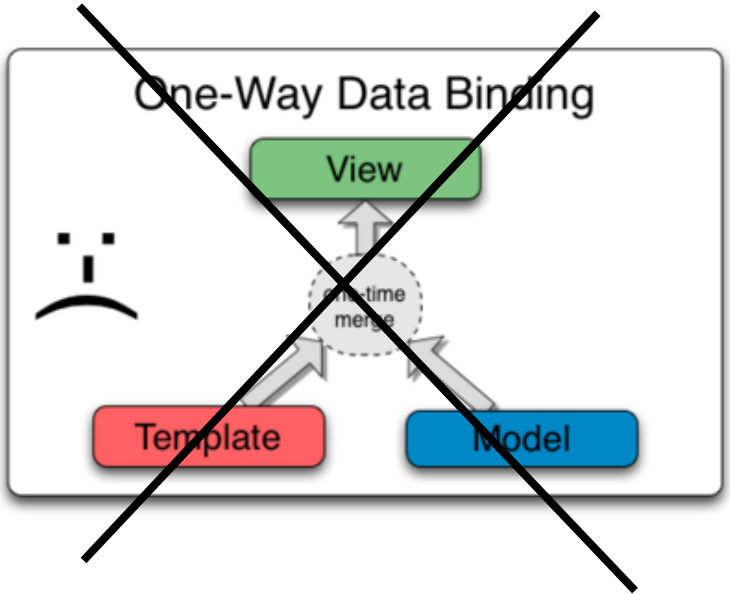
Les composants

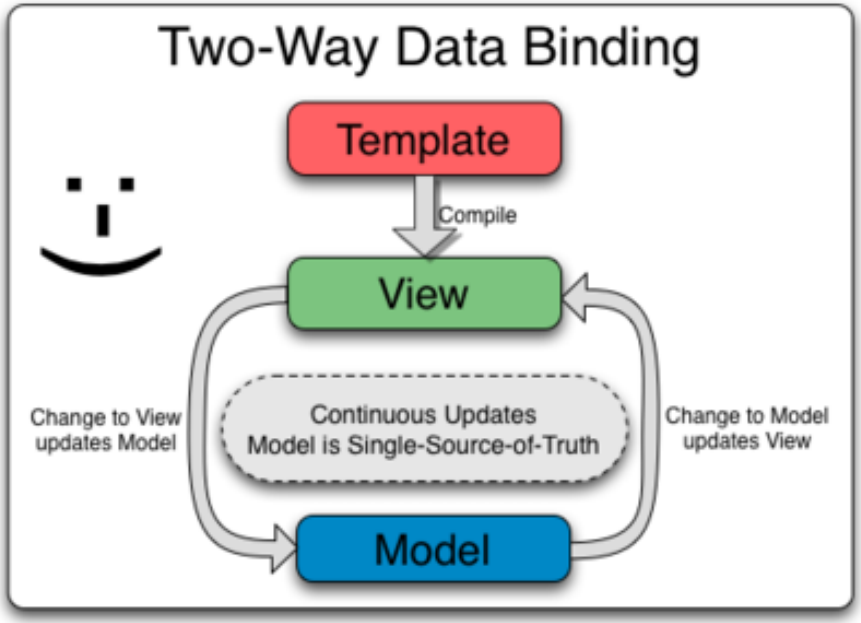
- Forcer la séparation des préoccupations
- newRouter composant = controller + view

/components
  /**user**
    user.html
    user.js
  /**office**
    office.html
    office.js
    office.directive.js

Data Binding

- Automatiser les changements





Directives

- Créer des composants **autonome** et **réutilisable**
- Marqueur HTML (element, class, attributes, etc.)
- Beaucoup de directive native

  ng-if, ng-show, ng-model, ng-repeat, etc.

Filters

- Des “pure functions”
- Mute une valeur
- Beaucoup de filtres natifs

  limitTo, json, currency

```
<div ng-app ng-init="amount = 1234.56">
  {{amount | currency:"USD$"}} // USD$1,234.56
  {{amount | currency:"USD$":0}} // USD$1,235
</div>
```

Services

- factory, service, provider, constant, value
- Singleton
- Logique métier
- Discussion serveur
- Beaucoup de services natifs
  $q, $http, $service, $timeout, etc.

L’injection dépendance

- *Dependency Injection*
- Importer les services par leur nom
- Facilite le testing et les dépendances

La communication avec le serveur

- Besoin de données (côté client)
- Pas trop ⇒ sécurité
- Service de requêtage
  - $http
  - $resource
  - Restangular

Testing

- Deux types de tests :
  - Tests unitaires (karma)
  - Tests e2e (protractor)
- Browser ViewLess
  - PhantomJS