
ANGULARJS
by Google

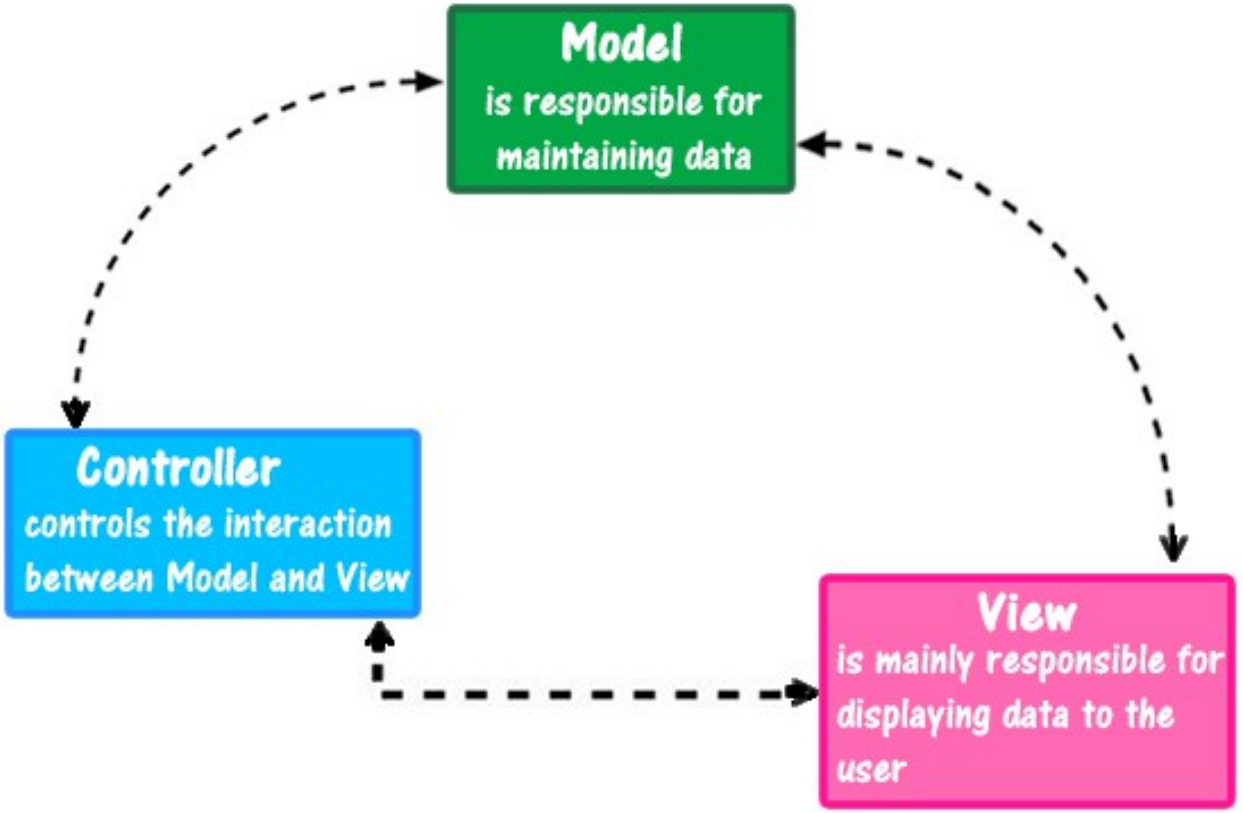
AngularJS 1.x

Généralités

- Framework Javascript libre et open-source
- Créé en 2009 puis repris et développé par Google
- But: créer des front-end d'applications web (client).
- Philosophie: étendre la syntaxe HTML pour l’adapter aux applications
- v1.0 sortie en 2012
- Version stable actuelle : v1.5.5

Spécificités : architecture MVC

- Model-View-Controller : séparation des éléments de l'application en blocs distincts (Contrôleurs, Services, Directives, Filtres, Vues/Routes…)



Spécificités : contrôleurs

- Gestion des vues et de la connexion avec les modèles de données et les services.
- Chaque contrôleur possède un scope : objet qui permet d'accéder aux modèles et aux fonctions dans la vue.

```
angular.module("stockApp", []).controller("lineController", function($scope) {
        var lines = [{ name: "Ms1YaH", archived: true },
                     { name: "ES alpha", archived: false }];
        $scope.lines = lines;
});
```

```
<div ng-controller="lineController">
        <table>
            <tr ng-repeat:"line in lines"><td>{{ line.name }}</td></tr>
        </table>
        </div>
```

Spécificités : routes

- Système de routing dans un module indépendant.
- Association d'une URL à un contrôleur et son template

```
angular.module("stockApp", ['ngRoute']).config(function($routeProvider) {
    $routeProvider
        .when('/lines-table', {
            templateUrl: 'lines/line-table.html',
            controller: 'LineController'
        })
        .when('/home', {
            templateUrl: 'index.html'
        })
        .otherwise({
            redirectTo: '/home'
        });
});
```

Spécificités : services

- Contiennent la logique de l'application. Appelés par le contrôleur.

```
angular.module("stockApp.services").factory("lineService", function() {
        return {
            getList : function(){
                    return [{ name: "Ms1YaH", archived : true },
                            { name: "ES alpha", archived : false }];
                }
        }
});

angular.module("stockApp", []).controller("lineController",   ['lineService',
    function($scope) {
            var lines = lineService.getList() ;
            $scope.lines = lines;
    }
]);
```

Spécificités : filtres

- Permet de trier les données dans un template et/ou dans un contrôleur

```
angular.module("stockApp").filter('archiveFilter', function () {
    return function (lines) {
        var filteredLines ={};
        angular.forEach(lines, function(line){
            if(line.archive == true){
                filteredLines.push(line);
            }
        });
        return filteredLines;
    }
});
```

```
<div ng-controller="lineController">
    <table>
        <tr ng-repeat:"line in lines | archiveFilter">
            <td>{{ line.name }}</td>
        </tr>
    </table>
</div>
```

Spécificités : directives

- Permettent la manipulation du DOM. Angular possède ses propres directives natives (ng-*)

```
angular.module("stockApp", []).controller("lineController",   ['lineService',
function($scope) {
        var lines = lineService.getLines() ;
        $scope.lines = lines;
  }]);

angular.module("stockApp.directives").directive("linesTable", function() {
          return {
              template : '<table><tr ng-repeat:"line in lines">
                                <td>{{ line.name }}</td></tr></table>'
          }
  });

   <div ng-controller="lineController">
       <lines-table></lines-table>
    </div>
```

Organisation d’une application

StockApp

- controllers
  - LineController.js
- directives
  - LineTableDirective.js
- services
  - LineService.js
- templates
  - line-table.html
- app.js
- index.html

VS

StockApp

- modules
  - line
    - line.js
    - line-table.html
    - LineController.js
    - LineService.js
    - LineTableDirective.js
  - app.js
- index.html

Spécificités : injection de dépendances

- Permet de lier et de réutiliser des composants externes ou internes à l'application
- But : fournir à n'importe quel endroit du code une référence à un objet nécessaire.

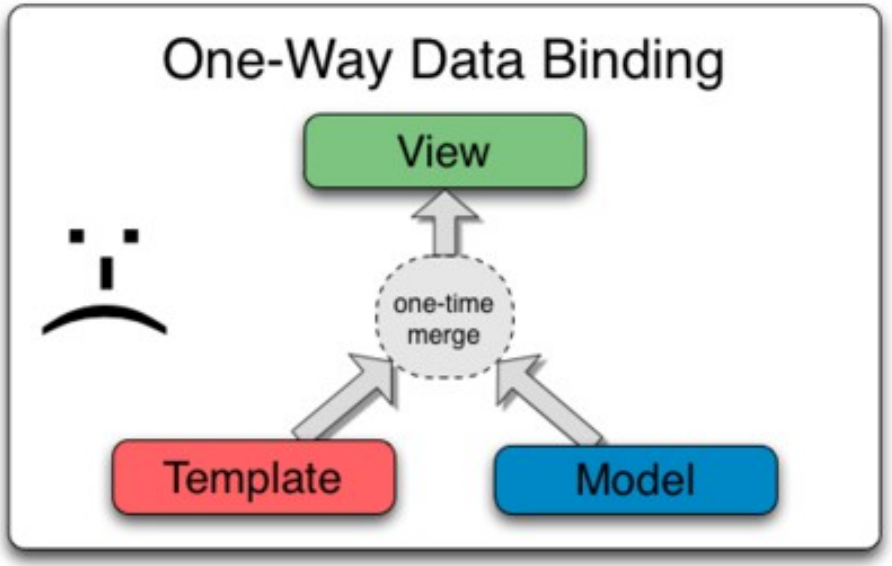
```
angular.module("stockApp", []).controller("lineController",   ['lineService',
    function($scope) {
        var lines = lineService.getList() ;
        $scope.lines = lines;
    }
]);
```

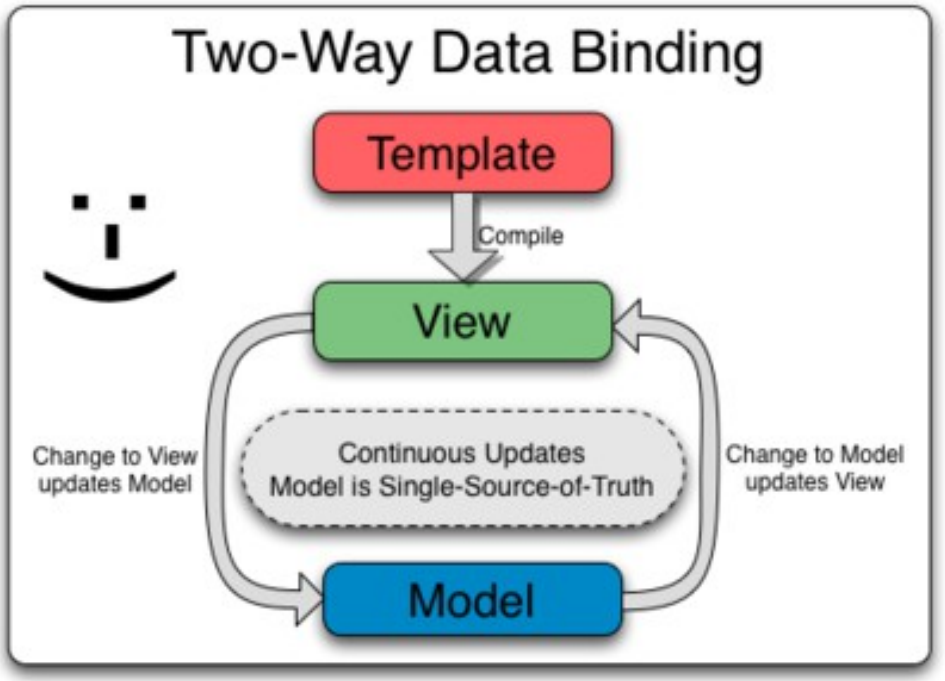
Spécificités : jqLite

- Manipulation du DOM via jqLite (basé sur jQuery)

Spécificités : « Two-way data binding »

- « Two-way data binding » : synchronisation automatique entre la Vue (ce que voit l'utilisateur) et le Modèle (les données manipulées par l'application).





Spécificités : « Two-way data binding »

- Utilisation du « dirty checking »: permet de surveiller et de détecter toutes les modifications sur un objet, une propriété ou une fonction.
- Pour savoir si un changement est survenu et si une mise à jour de la vue doit intervenir, Angular effectue un test sur toutes les valeurs observées à chaque fois qu’une donnée est susceptible d’avoir changée.

Avantages

- « Two-way data binding »
- Impose un code ordonné via le MVC
- Parfait pour les applications avec une grande interactivité côté client et/ou simple-page (SPA)
- Grande communauté
- Support de Google

Inconvénients

- Performance du dirty-checking : dégradation proportionnelle au nombre d’objets observés
- Documentation perfectible
- Pas de guide de bonnes pratiques lors de la sortie
- Certains concepts sont complexes à appréhender

A l'IGBMC

- Applications concernées
  - Zebrafish (angular v1.?)
  - Cells-stocks (angular v1.4.5)
  - Annuaire interne (angular v1.4.5)
  - Mouse 2 (angular v1.3.19)
  - UES (composant : angular v1.?)

Dans le reste du monde

- plnkr.co
- www.docker.com
- global.nba.com
- www.virginmobileusa.com
- www.emirates.com
- www.ryanair.com
- www.gotinder.com

Angular 2

Généralités

- Angular 2 annoncé en 2014 : pas de rétro-compatibilité !
- Actuellement en version beta : sortie prévue 1er semestre 2016
- Support de la v1.x jusqu'en 2018
- Mise en place de fonctions facilitant la migration à partir de la v1.5

Spécificités : TypeScript

- Angular 2 est écrit en TypeScript
- Langage open-source mis au point et supporté par Microsoft.
- Sur-ensemble de Javascript : possibilité d'inclure des fonctions JS dans un fichier TS, compilation en JS.
- Théoriquement, Javascript et Dart restent utilisables -> TS fortement recommandé.

Spécificités : TypeScript

- Avantages : typage des variables et des fonctions, création de classes, import de modules
- Exemple de code TypeScript vs Javascript :

```
class Greeter {
	greeting: string;
	constructor (message: string) {
		this.greeting = message;
	}
	greet() {
		return "Hello, " + this.greeting;
	}
}
```

```
var Greeter = (function () {
	function Greeter(message) {
		this.greeting = message;
	}
	Greeter.prototype.greet = function () {
		return "Hello, " + this.greeting;
	};
	return Greeter;
})();
```

Spécificités : suppression du contrôleur

- Plus de contrôleur à part entière -> architecture MV
- Mise en avant du concept de **composant,** similaire à la directive d’Angular 1
- Suppression du scope

Spécificités : directives

- **Component directive** : similaire aux directives actuelles et correspondant à un couple contrôleur / template (« composant »)
- **Decorator directive** : permet d'ajouter des fonctionnalités à un élément HTML existant (ng-show)
- **Template directive** : rend un élément HTML dynamique et réutilisable (ng-repeat)

Spécificités : fin du dirty-checking

- Le changement de détection se fait désormais de manière asynchrone et sous forme d’arbre.
- Chaque composant gère lui-même les changements et propage l’information aux composants parents : la détection n’a pas d’impact sur l’ensemble de l’application.
- Amélioration des performances jusqu'à 5 fois plus rapide (chiffre avancé par Google)

Autres spécificités

- Plus de jqLite: obligation d’avoir un navigateur récent supportant pleinement le HTML5 et ayant un « auto-update ».
- Amélioration du système de routing, avec ajout de systèmes d’authentification et de hiérarchie
- Orienté applications mobiles.
- Simplification des injections de dépendances

Comparaison de code

```
angular.module("stockApp.services").factory("lineService", function() {
        return {
            getList : function(){
                    return [{ name: "Ms1YaH", archived : true },
                            { name: "ES alpha", archived : false }];
                }
          }
        });

        angular.module("stockApp", []).controller("lineController",    ['lineService',
        function($scope) {
             var lines = lineService.getLines() ;
             $scope.lines = lines;
 }]);

angular.module("stockApp.directives").directive("lines-table", function() {
        return {
            template : '<table><tr ng-repeat:"line in lines">
                        <td>{{ line.name }}</td></tr></table>'
        }
 });
```

Comparaison de code

```
import { Injectable } from 'angular2/core';
@Injectable()
export class LineService {
  getList() {
    return [{ name: "Ms1YaH", archived : true },
            { name: "ES alpha", archived : false }];
  }
}

import { Component } from 'angular2/core';
@Component({
  selector: 'line-component',
  providers: [LineService],
  template: "<table> <tr *ngFor:#line of lines><td>{{ line.name }}</td></tr></table>"
})

export class LineComponent {
   constructor(
        private _router: Router,
        private _lineService: LineService
        ) {
            this._lineService.getList().then(lines => this.lines = lines);
        }
}
```

Comparaison de code

```
angular.module("stockApp", ['ngRoute']).config(function($routeProvider) {
    $routeProvider
        .when('/lines-table', {
            templateUrl: 'lines/line-table.html',
            controller: 'LineController'
        })
        .when('/home', {
            templateUrl: 'index.html'
        })
        .otherwise({
            redirectTo: '/home'
        });
});
```

Comparaison de code

```
import { RouteConfig } from 'angular2/router';
import { HomeComponent } from './home/home.component';
import { LineComponent } from './line/line.component';


@RouteConfig([
	{ path: '/home', name: 'Home', component: HomeComponent,
useAsDefault: true },
	{ path: '/line-table', name: 'LineTable', component: LineComponent },
])
```

Angular 1 → 2

Méthodes de migration

- « Big Bang » : ré-écriture complète en Angular 2 / TypeScript, convient aux petites applications
- Méthode incrémentale : migrer une application en v1.x composant par composant, pour les grosses applications.
- Mise en place de fonctionnalités facilitant la transition à partir de la version 1.5 (dont « component »)

Fonctions de migration : ngUpgrade

- Fourni avec Angular 2, boîte à outils permettant :
- L'utilisation de services/directives Angular 2 dans des composants Angular 1 et vice-versa.
- Inter-opérabilité du scope et du changement de détection
- Nécessite la cohabitation des versions 1.x et 2

Fonctions de migration : ngUpgrade

```
import {UpgradeAdapter} from 'angular2/upgrade';

let upgradeAdapter = new UpgradeAdapter() ;
upgradeAdapter.upgradeNg1Provider('Angular1LineService');


import {Angular1LineService} from '../services/a1-line-service';

@Injectable()
export class LineService {
    constructor(a1UpgradeService: Angular1LineService) {
        a1UpgradeService.getList() {
            return [{ name: "Ms1YaH", archived : true },
                    { name: "ES alpha", archived : false }];
        }
    }
})
```

Fonctions de migration : ngForward

- Fourni par la communauté, permettant l'utilisation de la syntaxe Angular 2 dans des applications Angular 1.
- Possibilité d'incorporer les modules de la v1.x
- Compatible à partir d'Angular 1.3

Fonctions de migration : ngForward

```
import { Injectable, Inject } from 'ng-forward';


@Injectable()
@Inject('$q', '$timeout')
class TestService {
	constructor($q, $timeout) {
		this.$q = $q;
		this.$timeout = $timeout;
	}
	getValue() {
		return this.$q(resolve => {
			this.$timeout(() => resolve('Value'), 3000);
		});
  }
}
```

Migration des modules

- Compatibilité des dépendances principales utilisées dans les applications
  - Restangular
  - AngularStrap
  - NgTable
- Possibilité d'utiliser ngUpgrade ?

Références

- Angular.io
- Rangle.io
- talkindotnet.com
- tutoriel-angularjs.fr
- maxlab.fr