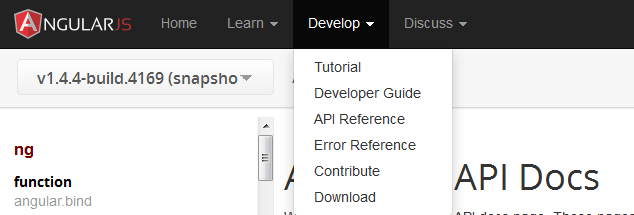
Introduction

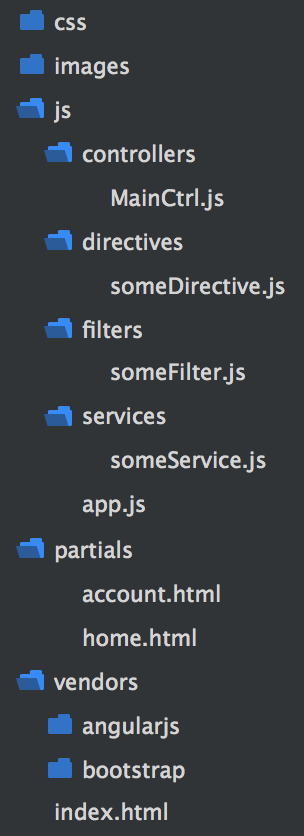
Avec Angular on code à la fois en javascript et en html, l’utilisation d’angular rend le html dynamique.



<https://docs.angularjs.org/api>

<http://www.tinci.fr/blog/apprendre-angular-en-un-jour-le-guide-ultime/>

<http://www.angular-js.fr/tous-les-articles/>



Les directives

Ce sont des attributs html

1. **NG-APP**

ng-app permet de dire à AngularJs qu’il doit être actif sur cette section de la page

<!DOCTYPE html>

<!--

ng-app indique que angular va etre utilisé dans cette balise

-->

<html ng-app="monApp">

</html>

1. **NG-MODEL**

C’est une des « killer features » d’AngularJs, c’est ce que l’on appelle du **two-way data binding**. Si les données sont mises à jour dans le contrôleur, les changements seront répercutés dans la vue, et si les données sont mises à jour dans la vue, les changements seront répercutés dans le contrôleur!

<input type="text" ng-model="variable\_champ"/>

<!--

ng-model indique que la valeur du champ va être stockée dans la variable "variable\_champ"

-->

Renvoi de valeur

{{ variable\_champ + "concatenation" }} <!-- renvoie la valeur de la variable -->

Tout ce qui est entre {{ }} sera **interprété**. Les variables et fonctions du contrôleur sont utilisables ici. Si les données changent dans le contrôleur, les changements seront répercutés sur la vue. C’est du **one-way data binding**

one-time binding.

When interpolating text or attributes:

<div name="attr: {{::color}}">text: {{::name | uppercase}}</div>

1. **NG-INIT**

En pratique peu utile car on s’attachera à déclarer les variable dans les contrôleurs angular

<div ng-init="users=['s','sd']"><!-- ng-init directive d'initialisation -->

</div>

1. **NG-REPEAT**

<li ng-repeat="produit in Panier">

<!-- directive equivalente à foreach, repete la balise contenante -->

<span>{{produit.nom}}</span>

<span>{{produit.quantite}}</span>

</li>

1. **NG-CLASS**

ng-class=”predicatTrue ? Class1 : ‘’ ”

<li ng-repeat="produit in Panier" ng-class={completed : predicatTrue}>

<!—affecte la class ‘completed’ si(:) predicat est true -->

</li>

1. **ng-submit**

execute the function passed in argument (that needs to be first defined in the controller )

<form ng-submit="addName()">

1. **ng-click**

<a href="" ng-click="removeName(name)">remove</a>

1. **ng-src**

only loads the src once the angular model has been loaded

<td><img ng-src="{{country.flagURL}}" width="100"></td>

1. **Ng-show/hide**
2. **ng-include/ ng-view**

anywhere including nesting in templates. ng-view managed by routing so typically only one exists

ngView works together with routing, and is mostly essential to an angular SPA. It's normally used to include the main content of your site, the bit in between the header and footer that changes every time a user clicks a link. Its integration with routing means changing the ngView content changes the controller too (depending on your configuration).

ngInclude is a general purpose include, you may not even need it. You'd sometimes use it inside a view to avoid repeating yourself, in the cases where several views need the same code included.

1. **Custom directive**

.directive('myDialog', function() {

return {

restrict: 'E',

transclude: true,

scope: {

'close': '&onClose'

},

templateUrl: 'my-dialog-close.html'

};

});

Les Filtres

Au niveau des directives

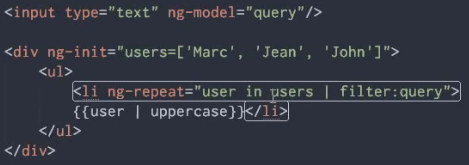
customdirective="truc **|** filter : qlqchse"

Au niveau des valeur de retour

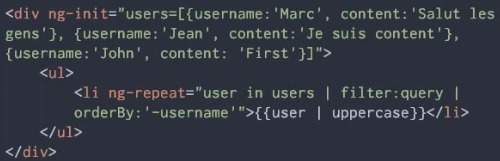
{{variable(ceci est l'input) **|** nomfiltre:4 :6}}

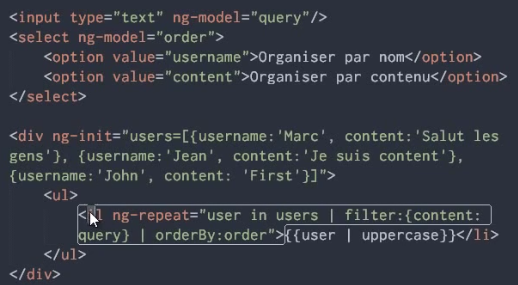
{{users **|** filter:’hn’}} //retourne les user qui contienent ‘hn’

Example



Cumule de filtres





[All Selector (“\*”)](http://api.jquery.com/all-selector/" \o "Permalink to All Selector ()

[Class Selector (“.class”)](http://api.jquery.com/class-selector/" \o "Permalink to Class Selector ()

[Element Selector (“element”)](http://api.jquery.com/element-selector/" \o "Permalink to Element Selector ()

[ID Selector (“#id”)](http://api.jquery.com/id-selector/" \o "Permalink to ID Selector ()

[Multiple Selector (“selector1, selector2, selectorN”)](http://api.jquery.com/multiple-selector/" \o "Permalink to Multiple Selector ()

Sélection par id

$(function(){

$(‘’#toto”).hide(); // select element with id ‘toto’ and hide it

}) ;

CUSTOM FILTERS

.filter('FILTER\_NAME', function() {

return function(input, uppercase) {

input = input || '';

var out = "";

for (var i = 0; i < input.length; i++) {

out = input.charAt(i) + out;

}

// conditional based on optional argument

if (uppercase) {

out = out.toUpperCase();

}

return out;

};

})

Le Controller et le scope

Un controller est lié à un element du dom et contient la logique business qui lui correspond.

A controller belongs to a module

Le scope permet de faire la liaison entre le controller et la vue, i.e. on peut modifier des variables html directement depuis le javascript

Le scope est une super variable qui contient différente sous-variables

Le scope est un objet javascript, et les propriétés qu’on lui ajoute (variables et fonctions) sont accessibles dans la vue, elles sont en quelque sorte publiques. Mais il est également possible de créer des variables et des fonctions privées (pas accessibles dans la vue).

//Public (accessible dans le contrôleur et dans la vue)

$scope.maVariable = "Hey";

$scope.maFonction = function () {

return 1;

};

//Privé (accessible seulement dans le contrôleur)

var maVariable2 = "Hey";

var maFonction2 = function () { //Une première manière de déclarer une fonction

return 2;

};

function maFonction3() { //Une deuxième manière de déclarer une fonction

return 3;

}

Au sein d’un ng-controller, à chaque Ng-repeat, ng-model, ng-if… un scope local (ou enfant ) est créé

Cette section de la page est gérée par le **contrôleur** CommentsCtrl (notez la majuscule). Les variables et fonctions déclarées dans le **scope** de ce contrôleur sont accessibles dans cette zone du html, et pas en dehors.

<input type="text" ng-model="query"/>

<div ng-controller ="commentsCtrl">

<!--ng-init="comments=[

{username:'Marc', content:'Salut les gens'},

{username:'Jean', content:'Je suis content'},

{username:'John', content:'First'}

]">-->

<div ng-repeat="comt in comments | filter:{content:query} | orderBy:order">

<p>

<strong>{{comt.username}}</strong><br />

{{comt.content}}

</p>

</div>

</div>

<script>

function commentsCtrl($scope){

// cette fonction controller permet d'initialiser a travers le scope des variables accessibles dans le html

$scope.comments = [...];

}

// reciproquement, la variable "query" declarée dans ng-model est accessible dans le script

</script>

$scope.$watch permet de controler les changement de valeur da la variable passée en paramtre et d’agir en consequence ($scope.$watch($scope.total, calculateDiscount\_ou fonction anonyme);)

<https://www.youtube.com/watch?v=E1NIJjTYq6U>

sharing data across controllers

myApp.factory/value/service('clientId', function clientIdFactory() {

return {var1 = 'a12345654321x'};

});

Then you inject it in the controller that uses it :

myApp.controller('DemoController', ['clientId', function DemoController(clientId) {

this.clientId = clientId;

}]);

Les modules

They are containers for various parts of an angular app (like controllers, services) they comprose a configa and a run block they can represent a functionality , behavior or sub

Ils permettent d’organiser le code angular

<body ng-app="MonApp\_module">

.

.

.

<script>

//on stocke les modules dans des variables

var app = angular.module('MonApp\_module',[/\*sert à injecter des dependances, eg des modules apres les avoir declaré/linké eg: nganimate\*/])

.config…

.run…

app.controller('commentsCtrl',['$scope',function($scope){ $scope.comments = [...];}] )

//on ajoute ['$scope', si on veut un jour minifier le code pour que $scope reste pris en compte

</script>

</body>

Le routing

Ils permettent d’organiser le code angular, il utilise la directive **ng-view,** il a besoin du module ng-route qui doit etre importé dans le header :

<script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.10/angular-route.min.js"></script>

<body ng-app="MonApp\_routeModule">

.

**<div ng-view></div>**

.

<script>

//le routing a besoin du module ngRoute

var app = angular.module('MonApp\_routeModule',[‘ngRoute’])

app.config( ['$routePovider', function($routePovider){

$routePovider

.when(‘/’,{templateUrl:’home.html’})

.when(‘/comments’,{templateUrl: ‘comments.html’,

**Controller: 'commentsCtrl'**})

// permet de ne pas declarer ng-controller ="commentsCtrl"

.otherwise({redirectTo : ‘/’});

}])

app.controller('commentsCtrl',

['$scope', function($scope){ $scope.comments = [...];}])

</script>

</body>

In html:

Data-ng-submit (framework directive)

Data-ng-model (framework directive)

Data-ng-if (framework directive)

Data-ng-repeat="truc **in** tableau" (framework directive)

Data-ng-click

Data-nouveauNG-directive (this is a CUSTOM directive)

In JS

MonApp.directive('afcdirective',[function(){

Return{

templateURL:'informationdeladirective.html',

restrict:'A', //restreint l'utilisation aux attributs

transclude:'false', //si true affect le comportement à l'élement

Scope : { ngdirectiv = '='},

Link: function(){

Element.on('dblclick',) }

}

}])

MonApp.filter('nomfiltre',[function(){

return function(input, arg1(va etre remplacé par 4), arg2(va etre remplacé par 6))

}.....])

appel dans html {{variabl(ceci est l'input) | nomfiltre:4 :6}}

service

contoller

routing

### **Job: RBS seek Visual Designer**

At The Royal Bank of Scotland, the UX Team in Edinburgh is looking for an experienced and creative Visual Designer to help us make our new account opening solution better for our customers and staff. We’re working on a new system that will let our customers open bank accounts and apply for loans and credit cards as simply as possible – and we want to make the experience satisfying and welcoming. The role involves: · Interpreting and applying the new NatWest and Royal Bank branding and styles to existing and new interactions · Looking for opportunities to improve the visual presentation in our designs to create greater customer engagement and build confidence · Develop new ways of applying and extending brand elements to digital interactions · Developing concepts for new interaction approaches to address business and customer needs · Working with research colleagues to help construct and execute suitable insight gathering exercises · Supporting designers in finding ways to improve the customer experience and reflecting brand values · Assist in the definition of a visual design language, design patterns, and principles that centre around re-use · Developing assets for use by designers and developers Ideally you’ll have 5-7 years experience in graphic design in a digital context. As well as having a creative approach to visual design you’ll be familiar with rich interaction design including the use of animation techniques (e.g. to provide focus and aid communication/understanding). You’ll take part in customer research and testing and we expect you to make a major contribution to the creation of an engaging and successful application together with the Bank’s emerging Design System. You’ll be an active contributor and collaborator in a team environment, working with a team of designers and writers along with developers who are building our applications using Agile methods. Please send your CV and links to relevant examples of your work to Brian.Cooper@rbs.co.uk Role will commence in January 2018.

Posted 2 days ago  
By Royal Bank of Scotland

Deadline 24 November 2017   
[Brian.Cooper@rbs.co.uk](http://Brian.Cooper@rbs.co.uk/)