**[How do I think in angularjs if have a jquery background](http://stackoverflow.com/questions/14994391/how-do-i-think-in-angularjs-if-i-have-a-jquery-background)**

## 1. 不要先設計頁面，然後再使用DOM操作來改變它的展現

在jQuery中，你通常會設計一個頁面，然後再給它動態效果。這是因為jQuery的設計就是為了擴充DOM並在這個簡單的前提下瘋狂地增長；但是在AngularJS裡，必須一開始就在頭腦中思考架構。必須從你想要完成的功能開始，然後設計應用程式，最後裁示設計檢視(view)，而不是“我有這麼一個DOM片段，我想讓他可以實現XXX效果”。

## 2. 不要用AngularJS來加強jQuery

同樣的，不要以這樣的思維來混用AngularJS與jQuery，比如：用jQuery來做X，Y和Z，然後又把AngularJS的models和controllers加在這上面。這在一開始的時候或許很好用，但這也是為什麼我總是建議AngularJS的新手完全不使用jQuery，至少不要在習慣使用“Angular Way”開發之前這麼做。因為我在我的信箱裡看到很多開發者使用150或200行程式碼的jQuery外掛程式創造出複雜的解決方法，然後使用一堆callback函數以及$apply把它粘合到AngularJS裡，看起來複雜難懂；但是他們最終還是把它搞定了！但問題是在大多數情況下，這些jQuery外掛程式可以使用很少的AngularJS代碼重寫，而且所有的一切都很簡單直接容易理解。

所以，當你在開發application或網站前，請先思考要用何種架構開發 (AngularJS or jQuery?)。建議先“Think in AngularJS”，如果想不出一個適合的方案，不如就先去各社群求助，若還是沒有更好的方面，最後才考慮使用jQuery；不要讓jQuery成為你成長的絆腳石，導致你永遠無法真正掌握AngularJS。

## 3. 以架構的角度思考

首先要知道Single-Page-Application(SPA) 就是application，不是網頁。 所以在使用AngularJS開發時，除了要以前端的角度思考外，也要以後端的角度思考。我們必須考慮如何把application分割成獨立的，可擴展且可測試的組件。那麼如何做到呢？如何“Think in AngularJS”?? 這裡有一些基本原則—**“**跟jQuery對比**”**。

#### **View是“Official Record”**

在jQuery裡，我們會把一個下拉式功能表定義為一個ul ：

<ul class="main-menu">

<li class="active"> <a href="#/home">Home</a> </li>

<li> <a href="#/menu1">Menu 1</a>

<ul>

<li><a href="#/sm1">Submenu 1</a></li>

<li><a href="#/sm2">Submenu 2</a></li>

<li><a href="#/sm3">Submenu 3</a></li>

</ul>

</li>

<li> <a href="#/home">Menu 2</a> </li>

</ul>

在jQuery裡，這個下拉式功能表的程式邏輯：

$('.main-menu').dropdownMenu();

當我們只關注view，這裡不會立即且明顯地表現出任何程式邏輯，對於小型應用，這沒什麼不妥。

但是在規模較大的應用中，事情就會變得難以理解且難以維護。

而在AngularJS裡，view是基於view的功能。其ul的寫法就會像這樣：

<ul class="main-menu" dropdown-menu> ... </ul>

這兩種方式做了同樣的東西，但是在AngularJS的版本裡任何人看到這個模版(template)都可以知道將會發生什麼事。不論何時一個新成員加入開發團隊，他看到這個就會知道有一個叫做dropdownMenu的directive作用在這個ul標籤上；他不需去猜測此程式碼的功能或者去看任何程式碼，從view就可以直接簡單清楚此元素的功能。

首次接觸AngularJS的開發者通常會問這個問題：如何找到所有的某類元素然後給它們加上一個directive。但當我們告訴他：別這麼做時，他總會顯得非常的驚愕。而不這麼做的原因是這是一種半jQuery半AngularJS的方式，這麼做不好。這裡的問題在於某些開發者嘗試在 AngularJS的環境裡參入“ jQuery”，但這麼做總會有一些問題。view是official record （譯者注：作者可能想表達view是一等公民）。在一個directive外，絕不要改變DOM。所有的directive都應用在view上，意圖非常清晰。記住：不要設計，然後寫標籤。你需要架構，然後設計。

### Data binding

這是到現在為止最酷的AngularJS特性。這個特性使得前面提到的很多DOM操作都顯得不再需要。AngularJS會自動更新view，所以你自己不需再這麼做了!! 在jQuery裡，我們需有建立事件(event)，並在事件(event)發生後才會產生相對應的回應，就像這樣：

$.ajax({

url: '/myEndpoint.json',

success: function ( data, status ) {

$('ul#log').append('<li>Data Received!</li>');

}

});

對應的view:

<ul class="messages" id="log"> </ul>

除了要考慮多個方面，我們也需要手動更新一個DOM節點。如果我們想要刪除一個log條目，也需要針對DOM編碼。那麼如何脫離DOM來測試這個邏輯？如果想要改變展現形式怎麼辦？

在AngularJS裡，可以這樣來實現：

$http( '/myEndpoint.json' ).then( function ( response ) {

$scope.log.push( { msg: 'Data Received!' } );

});

view看起來是這個樣子的：

<ul class="messages">

<li ng-repeat="entry in log">{{ entry.msg }}</li>

</ul>

甚至可以用其他方式來建置view：

<div class="messages">

<div class="alert" ng-repeat="entry in log">

{{ entry.msg }}

</div>

</div>

又或者，如果我們現在改使用Bootstrap的alert boxes，根本不需要改變任何的controller程式碼！更重要的是，不論log在何處或如何被更新，view便會隨之更新。儘管我沒有在這裡展示，但Dara Binding其實是雙向的。所以這些log資訊在view裡也可以是可編輯的，只需要這麼做：

<input ng-model="entry.msg" />

### 清晰的模型（Model）層

在jQuery裡，DOM在一定程度上扮演了模型的角色。但在AngularJS中，我們有一個獨立的模型層(model)可以靈活的管理。完全與view獨立。這有助於上述的資料綁定，維護了關注點的分離（獨立的考慮view和model），並且引入了更好的可測性。後面還會提到這點。

### 關注點分離

上面所有的內容都與這個願景相關：保持你的關注點分離，讓view展現將要發生的事情；model來存取資料；service執行可複用的任務；在directive裡面進行DOM操作和擴展；使用controller來把上面的東西粘合起來。

### 相依性注入

相依性注入幫我們實現了關注點分離。如果你來自一個伺服器語言（java或php），可能對這個概念已經非常熟悉，但是如果你是一個來自jQuery的用戶端開發者，這個概念可能看起來有點傻而多餘，但其實不是!!

大體來講，DI意味著可以非常自由地宣告元件(A)，然後在另一個元件裡(B)，只需要請求一個該元件(A)的實例，就可以得到它(A)。不需關注載入順序、檔案的位置等類似事情。雖然這種強大可能不會立刻顯現，但是我只提供一個常見的例子—測試:

假設在你的應用程式裡，有一個服務(Service)通過REST API並根據不同的應用狀態來與伺服器端進行資料存儲，當在測試controller時，因為只是在測試controller邏輯，所以不一定需要測試伺服器端的資料儲存功能。在這種請況下，我們可以只注入一個與本來使用的service同名的mock service(假service)，相依性注入就會確保controller自動得到這個假service。

## 4. 測試驅動開發

這其實是關於架構的第3節。但是它太重要了，所以我把它單獨拿出來作為一個段落。在所有那些你見過，用過或寫過的jQuery外掛程式中，有多少是有測試過？不多，因為jQuery經不起測試的考驗。但是AngularJS可以。

在jQuery中，唯一的測試方式通常是在頁面上獨立開發一個元件，然後我們操作並測試這個頁面上的DOM元素來驗證此元件。所以我們必須獨立的開發一個元件，然後整合到我們的Application，想想看這有多不方便 ?! 在使用jQuery開發時，我們花太多的時間測試與反覆運算而非以測試驅動方式開發，但這又能怪誰呢？

但因為有了關注點分離，我們可以在AngularJS中反覆地做測試驅動開發！例如，想要一個超級簡單的directive來展現我們的當前路徑。可以在view裡聲明：

<a href="/hello" when-active>Hello</a>

OK，現在可以寫一個測試：

it('should add "active" when the route changes', inject(function () {

var elm = $compile('<a href="/hello" when-active>Hello</a>')($scope);

$location.path('/not-matching');

expect(elm.hasClass('active')).toBeFalsey();

$location.path('/hello');

expect(elm.hasClass('active')).toBeTruthy();

}));

執行這個測試來確認它是失敗的。然後我們可以開始寫這個directive了：

.directive('whenActive', function ($location) {

return {

scope: true,

link: function (scope, element, attrs) {

scope.$on('$routeChangeSuccess', function () {

if ($location.path() == element.attr('href')) {

element.addClass('active');

} else {

element.removeClass('active');

}

});

}

};

});

測試現在通過了，然後我們的menu按照請求的方式執行，開發過程既是反覆運算的也是測試驅動的。

## 5. 概念上，Directives並不是打包的jQuery

你經常會聽到“只在directive裡做DOM操作”。這是必需的！但讓我們再深入一點……

一些directive僅僅裝飾了view中已經存在的東西（如ngClass）並且因此有時候僅僅直接做完DOM操作然後就完事了。但是如果一個 directive像一個“widget” 並且有一個模版( template)，那麼它就要做到關注點分離，而且它的模版( template)本身一定程度上需要與其link和 controller保持分離。

AngularJS擁有一整套工具使這個過程非常簡單，比如：有了ngClass我們可以動態地更新class；ngBind使得我們可以做雙向資料綁定； ngShow和ngHide可程式設計地展示和隱藏一個元素以及更多(包括那些我們自己寫的)。換句話說，我們可以做到任何DOM操作能實現的特性。此外，通常DOM操作越少，其directive就越容易測試，也越容易給它們添加樣式，在未來也越容易擁抱變化，更可重複使用和發佈。

我見過很多AngularJS新手，把一堆jQuery扔到directive裡。 換句話說，他們認為“因為不能在controller裡做DOM操作，就把那些代碼弄到directive裡好了”。雖然這麼做確實好一些，但是依然是錯誤的。

回想一下我們在第3節裡寫的那個logger。即使要把它放在一個directive裡，我們依然希望用“Angular Way”來做，它依然沒有任何DOM操作！

這裡有一個示例，展示出了我見過最多的一種模式。我們想做一個可以toggle的按鈕。

.directive( 'myDirective', function () {

return {

template: '<a class="btn">Toggle me!</a>',

link: function ( scope, element, attrs ) {

var on = false;

$(element).click( function () {

on = !on;

$(element).toggleClass('active', on);

});

}

};

});

這裡有一些錯誤的地方:

* + - 1. jQeury根本沒必要出現。我們在這裡做的事情都根本用不著jQuery！
      2. 即使已經將jQuery用在了頁面上，也沒有理由用在這裡。
      3. 即使假設這個directive依賴jQuery來工作，jqLite(angular.element)在載入後總會使用jQuery！所以我們沒必要使用$，用angular.element就夠了。
      4. 和第三條緊密關聯，jqLite元素不需要被$封裝，傳到link裡的元素本來就會是一個jQuery元素！
      5. 我們在前面段落中說過，為什麼要把模版( template)的東西混到邏輯裡？

若改用directive可以寫得更簡單（即使是更複雜的情況下！）：

.directive( 'myDirective', function () {

return {

scope: true,

template: '<a class="btn" ng-class="{active: on}" ng-click="toggle()">Toggle me!</a>',

link: function ( scope, element, attrs ) {

scope.on = false;

scope.toggle = function () {

scope.on = !scope.on;

};

}

};

});

再次強調，模版( template)就在模版( template)裡，當有樣式需求時，你（或你的用戶）可以輕鬆的換掉它，不用去碰邏輯，這就是重用性( Reusability) !!

當然還有其他的好處，比如測試，不論模版( template)中有什麼，directive的內部API從來不會被碰到，所以重構也很容易。可以不碰directive就做到任意改變模版( template)。不論你怎麼改，測試總是通過的。

所以如果directive不僅僅是一組類似jQuery的函數，那他們是什麼？Directive實際是HTML的擴展。如果HTML沒有做你要的功能，你就可以自己寫一個directive，就像使用HTML一樣使用這個directive。

## 總結

不要用jQuery，甚至連include jQuery也不要。它會讓你停滯不前。如果遇到一個你認為已經知道如何使用jQuery來解決的問題，在使用$之前，請試想如何在AngularJS的限制下去解決這個問題。建議最好不要使用jQuery。如果嘗試使用jQuery會增加你的工作量。