# **A sopa de caracteres: @, & e =**

Opa! Criar diretivas você já sabe, mas será que compreendeu bem a diferença entre @, & e =?

Faremos uma revisão (mas prepare-se, no final tem uma pergunta "mega blast ultra super":

* Usamos @ quando queremos realizar uma cópia do valor passado para a diretiva no HTML para dentro do escopo isolado na diretiva. Essa cópia é sempre um valor em string.
* Usamos & geralmente quando queremos executar dentro de uma diretiva uma **função** que pertence a um escopo pai, o de um controller.
* Usamos = quando queremos que tanto a diretiva quanto o controller acessem o mesmo dado, isto é, qualquer alteração no escopo privado da diretiva afetará a propriedade do controller e qualquer alteração nesta propriedade no controller afetará a diretiva. Temos aqui uma comunicação bidirecional.
* Agora que você já lembrou dessa "sopa de caracteres" vem a pergunta.

Partindo do pressuposto que já temos o módulo puzzle criado, temos a seguinte diretiva em uso:

<div ng-controller="CharadaController">  
 <minha-charada titulo="{{titulo}}" ></minha-charada>  
</div>

O controller:

angular.module('puzzle').controller('CharadaController', function($scope) {  
 $scope.titulo = "Alura";  
 });

Agora nossa diretiva:

angular.module('puzzle').directive('minhaCharada', function() {  
 var ddo = {};  
 ddo.restrict = 'E';  
 ddo.scope = {  
 titulo: '@'  
 };  
 ddo.template = '<p>{{titulo}}</p>';  
 return ddo;  
 });

Qual será o resultado do processamento da diretiva:

<div ng-controller="CharadaController">  
 <minha-charada titulo="{{titulo}}" ></minha-charada>  
</div>

* Nada será exibido, a diretiva não funcionará, dará um erro
* Será exibido para o usuário o texto {{titulo}}
* **Será exibido para o usuário o texto Alura**
* Nada será exibido, resultado branco apenas

**Explicação**

**Todo código é funcional e o que será exibido para o usuário será o texto Alura.**

**Vejamos a diretiva em uso:**

**<div ng-controller="CharadaController">  
 <minha-charada titulo="{{titulo}}" >   
 </minha-charada>  
</div>**

**Veja que usamos uma AE (Angular Expression) como valor do atributo titulo. Angular avaliará primeiro a AE e depois passará seu resultado para a diretiva, que será "Alura". Como nossa diretiva guarda em seu escopo privado a propriedade titulo com o modificador @, está tudo certo, porque ele indica que queremos capturar o que foi passado no HTML como string dentro de nossa diretiva.**

# **Manipulação de DOM**

# **Angular nos blinda de termos que lidar com o DOM, porém ele não tem resposta para tudo e acabamos tendo que manipulá-lo.**

# **Segue algumas afirmações sobre a manipulação de DOM no Angular:**

# **1) Deve ser feita nos controllers, pois não dificultam sua leitura, muito menos sua testabilidade.**

# **2) Deve ser realizada em diretivas, pois o código que manipula o DOM ficará separado do código de um controller ou de um serviço. Como o controller nem o serviço manipulam DOM, tornam-se mais fáceis de compreender e de testar em seu framework de teste favorito.**

# **3) Angular não permite a manipulação de DOM, já que ele visa nos blindar desta frágil API.**

# **4) Não é possível manipular DOM em controllers do Angular.**

# **Das afirmações acima, podemos afirmar que:**

* 1, 2 e 3 são falsas
* 2 e 3 são falsas
* **Somente 2 é verdadeira**
* 2 e 4 são verdadeiras

**Explicação**

**Manipulação de DOM deve ser realizada em diretivas, pois o código que manipula DOM ficará separado do código de um controller ou de um serviço. Como o controller nem o serviço manipulam DOM, tornam-se mais fáceis de compreender e de testar em seu framework de teste favorito. Porém, nada impede que o desenvolvedor manipule DOM dentro dos controllers, o Angular não bloqueia isso, apesar de contestável, pela filosofia do framework.**

# **Função link**

**Diretivas possuem a propriedade link, cuja função nos dá acesso ao elemento do DOM no qual ela foi adicionada na marcação HTML, inclusive ao escopo privado da diretiva. É nela que manipulamos DOM quando necessário.**

**Sobre o código seguinte:**

**// módulo omitido  
.directive('meuFocus', function() {  
 var ddo = {};  
 ddo.restrict = "A";  
 ddo.scope = {  
 focado : '='  
 };  
 ddo.link = function(element, scope) {  
  
 };  
  
 return ddo;  
 });**

**Podemos afirmar que:**

* **Há um erro, porque os parâmetros da função link são posicionais, isto é, a ordem dos parâmetros fazem diferença. O primeiro parâmetro deveria ser o escopo e o segundo o elemento**
* Há um erro, porque existem outros parâmetros que a função link recebe que são obrigatórios
* Não há nada de errado com sua declaração
* Há um erro, porque a propriedade link deve fazer parte do scope e não diretamente na diretiva.

**Explicação**

**Para deixar ainda mais claro, diferente dos parâmetros injetados em um controller, no qual a ordem não importa, a função link possui parâmetros posicionais, como os de costume no mundo JavaScript. Se invertemos a ordem dos parâmetros teremos sérios problemas em nossa diretiva.**

# **Watchman :)**

**Vejamos a seguinte diretiva:**

**// módulo omitido  
.directive('meuFocus', function() {  
 var ddo = {};  
 ddo.restrict = "A";  
 ddo.scope = {  
 focado : '='  
 };  
  
 ddo.link = function(scope, element) {  
 // quero observar qualquer mudança em `focado`!  
 };**

**Altere o código anterior para que exiba a mensagem "mudei" toda vez que a propriedade "focado" mudar, seja na diretiva ou pelo controller.**

**Explicação**

**// módulo omitido  
.directive('meuFocus', function() {  
 var ddo = {};  
 ddo.restrict = "A";  
 ddo.scope = {  
 focado : '='  
 };  
  
 ddo.link = function(scope, element) {  
 // quero observar qualquer mudança em `focado`!  
 scope.$watch('focado', function() {  
 alert('mudei');  
 });  
 };**

**O $watch é mais inteligente ainda, podemos receber o valor atual e o valor antes da mudança como parâmetros:**

**// módulo omitido  
.directive('meuFocus', function() {  
 var ddo = {};  
 ddo.restrict = "A";  
 ddo.scope = {  
 focado : '='  
 };  
  
 ddo.link = function(scope, element) {  
 // quero observar qualquer mudança em `focado`!  
 scope.$watch('focado', function(novoValor, valorAntigo) {  
 alert('mudei');  
 });  
 };**