**Two-way data binding**

Seguem afirmativas sobre two-way data binding:

1) Permite ler dados (model) sem propagar atualizações desses dados a partir da view.

2) É realizado através de Angular Expression (AE)

3) É realizado pela combinação de Angular Expression (AE) e ng-model

4) É realizado através da diretiva ng-bind.

Podemos dizer que:

* **Todas são falsas**
* Apenas 1 é falsa
* 1 e 3 são verdadeiras
* Apenas 2 e 4 são falsas

**Explicação**

**Diferente da Angular Expression (AE) que é somente leitura (one-way data binding), a diretiva ng-model permite ler e propagar alterações de dados a partir da view.**

**ng-model ou Angular Expression?**

**Qual a diferença entre a diretiva ng-model e Angular Expression (AE)?**

* Não há diferenças.
* Apenas o segundo é two-way data binding.
* **O primeiro lê e grava no model, enquanto o segundo apenas lê.**
* O primeiro é one-way data binding, enquanto o segundo é two-way data binding.
* O primeiro faz two-way data binding, assim como o segundo.

**Explicação**

**A diretiva ng-model permite ler e também alterar o model associado (*two-way data binding*). É muito comum em telas de cadastro, onde precisamos capturar os dados do usuário para depois enviá-los numa requisição Ajax:**

**<input type="text" ng-model="nome">**

**A Angular Expression (AE) apenas lê o model (*one-way data binding*). Podemos usar a diretivang-bind no lugar da AE, é apenas outra maneira de expor o model em somente leitura:**

**<p> {{nome}} </p>**

**É a mesma coisa que:**

**<p ng-bind="nome"></p>**

**Filtrando listas com Angular**

**Veja o esboço de código HTML abaixo:**

**<label>  
 <input type="search" ng-model="textoFiltro">  
</label>  
  
<ul>  
 <li ng-repeat="funcionario in funcionarios">  
 {{funcionario.nome}}  
 </li>  
</ul>**

**O que devemos colocar na diretiva ng-repeat para filtrar a lista de funcionários a partir do input, com base no valor de textoFiltro?**

**Explicação**

**No final da diretiva ng-repeat devemos adicionar um | (o char *pipe*) seguido do nome do model: textoFiltro**

**<label>  
 <input type="search" ng-model="textoFiltro">  
</label>  
  
<ul>  
 <li ng-repeat="funcionario in funcionarios | filter: textoFiltro">  
 {{funcionario.nome}}  
 </li>  
</ul>**

**Essa configuração fará com que a nossa lista seja filtrada dinamicamente com base no valor de textoFiltro. Poderoso e simples!**

**Além disso, podemos adicionar uma função no filter, para definir a regra de comparação. Vejo o exemplo:**

**<li ng-repeat="funcionario in funcionarios | filter: textoFiltro: function(atual, esperado) { return angular.equals(atual, esperado)}">  
 {{funcionario.nome}}  
 </li>**

**Essa função só devolve true se o nome do funcionário for exatamente igual ao input notextoFiltro. Podemos simplificar isso e colocar apenas:**

**<li ng-repeat="funcionario in funcionarios | filter: textoFiltro: true">  
 {{funcionario.nome}}  
 </li>**

**Quer ver mais um exemplo? Na documentação do angular mostra como fazer um filtro de vários campos:** [**https://docs.angularjs.org/api/ng/filter/filter**](https://docs.angularjs.org/api/ng/filter/filter)

**Simples e poderoso!**

**A diretiva ng-model-options**

**Temos as seguintes frases a respeito da diretiva ng-model-options:**

**1) Serve para postergar a atualização do modelo (dado).**

**2) Sua unidade de trabalho é em milissegundos, passada através da propriedade debounce.**

**3) Não pode ser usada sem a diretiva ng-model**

**4) Pode vir antes ou depois da diretiva ng-model sem problemas.**

* **Todas as afirmativas são verdadeiras**
* Todas as afirmativas são falsas
* Apenas 3 e 4 são falsas
* Apenas 3 é falsa

**Explicação**

**Não faz sentido usarmos ng-model-options sem ng-model, pelo fato da primeira ser a responsável em passar opções especiais para a segunda. A diretiva ng-model-options, para postergar a atualização do model, recebe um objeto com a propriedade debounce e lá indicamos em milissegundos quanto tempo queremos postergar a atualização. Vejamos um exemplo:**

**<input ng-model="filtro" ng-model-options="{ debounce: 500 }">**

**A ordem das diretivas também não importa, podemos ter:**

**<input ng-model-options="{ debounce: 500 }" ng-model="filtro">**

**No exemplo acima, estamos postergando meio segundo.**

**Bônus: como o Angular sabe qual diretiva aplicar primeiro? Internamente, todas as diretivas do angular possuem a propriedade priority. Por exemplo, a diretiva ng-model possui a prioridade 1 enquanto a ng-model-options 0. A diretiva ng-repeat possui prioridade 1000. Diretivas com menor número de prioridade são aplicadas primeiro. Isso faz todo sentido, pois ng-model-options precisa ser aplicada antes de ng-model entrar em ação. É por isso que a ordem da diretiva no elemento da tag HTML não dita quem será aplicado primeiro, mas sim a prioridade.**

**O módulo ngAnimate**

**No mundo CSS, não é incomum termos uma classe declarada que é ativada apenas quando algum elemento tiver essa classe. Geralmente, podemos adicionar e remover classes via JavaScript, o que permite a aplicação condicional do estilo. O Angular possui o módulo ngAnimate, que pode nos ajudar nessa tarefa. Sobre este módulo podemos afirmar que:**

**1) Quando carregado, faz com que algumas diretivas do core do Angular adicionem e removam classes dinamicamente. Porém, não temos controle das classes adicionadas e precisamos recorrer à documentação para ver aquelas que nos interessam.**

**2) Quando carregado, traz uma série de animações já prontas para uso.**

**3) Quando carregado, permite que apenas a diretiva ng-repeat adicione dinamicamente classes de acordo com o estado dos elementos.**

**4) É um módulo que não faz parte do core do Angular.**

* 1, 3 e 4 são verdadeiras
* **1 e 4 são verdadeiras**
* Apenas 3 é falsa

**Explicação**

**O módulo ngAnimate precisa ser carregado, uma vez que o módulo core angular.min.jsnão o carrega automaticamente. Este módulo, apesar do nome sugestivo, não traz qualquer animação pronta para uso, ele apenas habilita para uma série de diretivas a capacidade de adicionarem ou removerem classes de acordo com o estado de seus elementos. Fazendo uma analogia com CSS, é como se fossem pseudo classes.**

**Não é do mundo CSS? Não se preocupe, o mais importante é saber que se temos algo do tipo:**

**.carnaval.ng-leave-active {  
 background-color: red;  
 color: black;  
}**

**Ela será aplicada apenas se algum elemento de nossa página tiver as classes carnaval e ng-leave-active. Basta uma não existir para que o estilo não seja aplicado. Sendo assim, sabemos que um elemento repetido por ng-repeat, ao deixar a lista, ganhará a classe ng-leave-active. Resumindo: o estilo acima só será aplicado se o elemento tiver a classe carnaval e estiver saindo da lista.**