ANGULARJS

AngularJS é um framework JavaScript. É uma biblioteca escrita em JavaScript. AngularJS é distribuída como um arquivo JavaScript, é adicionado a uma página web com uma tag script: **<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>**

AngularJS estende HTML com ng-directives. A diretiva ng-app define um aplicativo AngularJS. A diretriz ng-model vincula o valor dos controles HTML (entrada, seleção, textarea) aos dados do aplicativo. A diretriz ng-bind liga os dados do aplicativo à visualização HTML.

<!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>Name: <input type="text" ng-model="name"></p>  
  <p ng-bind="name"></p>  
</div>  
  
</body>  
</html>

**Exemplo explicado:** O AngularJS é iniciado automaticamente quando a página da Web for carregada. A diretiva ng-app diz ao AngularJS que o elemento <div> é o "proprietário" de um aplicativo AngularJS. A diretiva ng-model vincula o valor do campo de entrada ao nome da variável da aplicação. A diretriz ng-bind liga o innerHTML do elemento <p> ao nome da variável da aplicação.

Diretivas AngularJS

As diretivas AngularJS são atributos HTML com um prefixo ng. A diretiva ng-init inicializa as variáveis da aplicação AngularJS. <div ng-app="" ng-init="firstName='John'">  
<p>The name is <span ng-bind="firstName"></span></p>  
</div

Alternativamente, com HTML válido: <div data-ng-app="" data-ng-init="firstName='John'">  
<p>The name is <span data-ng-bind="firstName"></span></p>  
</div>

Você pode usar data-ng-, em vez de ng-, se você deseja tornar a sua página HTML válida. Usar ng-init não é muito comum. O mais comum é o uso de controladores.

Expressões AngularJS

As expressões AngularJS são escritas dentro de chaves duplas: {{expressão}}. AngularJS irá "produzir" dados exatamente onde a expressão está escrita: <!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>My first expression: {{ 5 + 5 }}</p>   
</div> // retorna my first expression: 10  
  
</body>  
</html>

As expressões AngularJS ligam os dados AngularJS ao HTML da mesma maneira que a diretriz ng-bind. Se você remover a diretriz do ng-app, HTML exibirá a expressão como está, sem resolvê-la.

Aplicações AngularJS

Os módulos AngularJS definem as aplicações AngularJS. Os controladores AngularJS controlam as aplicações AngularJS. A diretiva ng-app define o aplicativo, a diretiva ng-controller define o controlador.

<div ng-app="**myApp**" ng-controller="**myCtrl**">  
  
First Name: <input type="text" ng-model="firstName"><br>  
Last Name: <input type="text" ng-model="lastName"><br>  
<br>  
Full Name: {{firstName + " " + lastName}}  
  
</div>  
  
<script>  
var app = angular.module('**myApp**', []);  
app.controller('**myCtrl**', function($scope) {  
    $scope.firstName= "John";  
    $scope.lastName= "Doe";  
});  
</script>

Os módulos AngularJS definem aplicações: **var app = angular.module('myApp', []);** Controladores AngularJS controlam aplicativos: **app.controller('myCtrl', function($scope) {  
    $scope.firstName= "John";  
    $scope.lastName= "Doe";  
});**

EXPRESSÕES

As expressões AngularJS podem ser escritas dentro de chaves duplas: {{expression}}. As expressões AngularJS também podem ser escritas dentro de uma diretiva: ng-bind = "expression". AngularJS resolverá a expressão e retornará o resultado exatamente onde a expressão está escrita. As expressões AngularJS são muito parecidas com as expressões de JavaScript: elas podem conter literais, operadores e variáveis. **Exemplo {{5 + 5}} ou {{firstName + "" + lastName}}**

Você pode escrever expressões onde quiser, o AngularJS simplesmente resolverá a expressão e retornará o resultado. **Exemplo**: Deixe AngularJS alterar o valor das propriedades CSS. Altere a cor da caixa de entrada abaixo, alterando seu valor: <div ng-app="" ng-init="myCol='blue'"> // inicia o input com azul  
<input style="background-color:{{myCol}}" ng-model="myCol"  value="{{myCol}}">  
</div>

Números em AngularJS são como números de JavaScript: <div ng-app="" ng-init="quantity=1;cost=5">  
<p>Total in dollar: {{ quantity \* cost }}</p>  
</div> // retorna Total in dollar: 5

O mesmo exemplo usando ng-bind: <div ng-app="" ng-init="quantity=1;cost=5">  
<p>Total in dollar: <span ng-bind="quantity \* cost"></span></p>  
</div> // retorna Total in dollar: 5

Strings em AngularJS são como strings de JavaScript: <div ng-app="" ng-init="firstName='John';lastName='Doe'">  
<p>The name is {{ firstName + " " + lastName }}</p>  
</div> // retorna John Doe

Objetos em AngularJS são como objetos JavaScript: <div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">  
<p>The name is {{ person.lastName }}</p>  
</div> // retorna The name is Doe

Vetores em AngularJS são como vetores de JavaScript: <div ng-app="" ng-init="points=[1,15,19,2,40]">  
<p>The third result is {{ points[2] }}</p>  
</div> // retorna The third result is 19

Como expressões de JavaScript, as expressões AngularJS podem conter literais, operadores e variáveis. Ao contrário das expressões de JavaScript, as expressões AngularJS podem ser escritas dentro do HTML. As expressões AngularJS não suportam condicionais, loops e exceções, enquanto as expressões de JavaScript suportam. As expressões AngularJS suportam filtros, enquanto as expressões de JavaScript não.

MÓDULOS

Um módulo AngularJS define uma aplicação. O módulo é um recipiente para as diferentes partes de um aplicativo. O módulo é um recipiente para os controladores de aplicativos. Os controladores sempre pertencem a um módulo.

Criando um módulo

Um módulo é criado usando a função angularJS angular.module: <div ng-app="myApp">...</div>

<script>  
var app = angular.module("myApp", []);   
</script>

O parâmetro "myApp" refere-se a um elemento HTML no qual o aplicativo será executado. Agora, você pode adicionar controladores, diretivas, filtros e muito mais ao seu aplicativo.

Adicionando um controlador

Adicione um controlador à sua aplicação e consulte o controlador com a diretiva ng-controller: <div ng-app="**myApp**" ng-controller=**"myCtrl"**>  
{{ firstName + " " + lastName }}  
</div>  
  
<script>  
var app = angular.module(**"myApp"**, []);  
app.controller(**"myCtrl"**, function($scope) {  
    $scope.firstName = "John";  
    $scope.lastName = "Doe";  
});  
</script>

Adicionando uma Diretiva

O AngularJS possui um conjunto de diretivas integradas que você pode usar para adicionar funcionalidades à sua aplicação. Além disso, você pode usar o módulo para adicionar suas próprias diretivas às suas aplicações: <div ng-app="myApp" w3-test-directive></div>  
  
<script>   
var app = angular.module("myApp", []);  
app.directive("w3TestDirective", function() {  
    return {  
        template : "I was made in a directive constructor!"  
    };  
});  
</script>

Módulos e controladores em arquivos

É comum em aplicações AngularJS colocar o módulo e os controladores em arquivos JavaScript. Neste exemplo, "myApp.js" contém uma definição de módulo de aplicativo, enquanto "myCtrl.js" contém o controlador: <!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
<body>  
  
<div ng-app="**myApp**" ng-controller="**myCtrl**">  
{{ firstName + " " + lastName }}  
</div>  
  
<script src="**myApp.js**"></script>  
<script src="**myCtrl.js**"></script>  
  
</body>  
</html>

myApp.js

var app = angular.module(**"myApp"**, []);

O parâmetro [] na definição do módulo pode ser usado para definir módulos dependentes. Sem o parâmetro [], você não está criando um novo módulo, mas recuperando um existente.

myCtrl.js

app.controller(**"myCtrl"**, function($scope) {  
    $scope.firstName = "John";  
    $scope.lastName= "Doe";  
});

**Funções podem poluir o espaço de nomes globais**

As funções globais devem ser evitadas em JavaScript. Elas podem ser facilmente substituídos ou destruídos por outros scripts. Os módulos AngularJS reduzem esse problema, mantendo todas as funções locais no módulo.

**Quando carregar a biblioteca**

Embora seja comum em aplicativos HTML colocar scripts no final do elemento <body>, é recomendável que você carregue a biblioteca AngularJS na <head> ou no início do <body>. Isso ocorre porque as chamadas para angular.module só podem ser compiladas após o carregamento da biblioteca.

DIRETIVAS

O AngularJS permite expandir HTML com novos atributos chamados Diretivas. O AngularJS possui um conjunto de diretivas integradas que oferecem funcionalidades para suas aplicações. AngularJS também permite definir suas próprias diretrizes.

Algumas diretivas AngularJS

As diretivas AngularJS são atributos HTML estendidos com o prefixo ng-. A diretiva ng-app inicializa um aplicativo AngularJS. A diretiva ng-init inicializa os dados do aplicativo. A diretiva ng-model vincula o valor dos controles HTML (entrada, seleção, textarea) aos dados. <div ng-app="" ng-init="firstName='John'">  
<p>Name: <input type="text" ng-model="firstName"></p>  
<p>You wrote: {{ firstName }}</p>  
</div>

A diretiva ng-app também diz ao AngularJS que o elemento <div> é o "proprietário" do aplicativo AngularJS.

Data Binding

A expressão {{firstName}}, no exemplo acima, é uma expressão de ligação de dados AngularJS. A ligação de dados(data binding) no AngularJS liga expressões AngularJS com dados AngularJS. {{FirstName}} é vinculado com ng-model = "firstName".

No próximo exemplo, dois campos de texto são vinculados com duas diretivas ng-model: <div ng-app="" ng-init="quantity=1;price=5">  
Quantity: <input type="number" ng-model="quantity">  
Costs:    <input type="number" ng-model="price">  
Total in dollar: {{ quantity \* price }}  
</div> // retorna Total in Dollar: 5

Repetindo elementos HTML

A diretiva ng-repeat repete um elemento HTML: <div ng-app="" ng-init="names=['Jani','Hege','Kai']">  
  <ul>  
    <li ng-repeat="x in names">  
      {{ x }}  
    </li>  
  </ul>  
</div> // retorna Jani Hege Kai

A diretriz ng-repeat realmente clona elementos HTML uma vez para cada item em uma coleção. Agora a diretiva ng-repeat sendo usada em uma série de objetos: <div ng-app="" ng-init="names=[  
{name:'Jani',country:'Norway'},  
{name:'Hege',country:'Sweden'},  
{name:'Kai',country:'Denmark'}]">  
<ul>  
  <li ng-repeat="x in names">  
    {{ x.name + ', ' + x.country }}  
  </li>  
</ul>  
</div> // retorna Jani,Norway Hege,Sweden Kai,Denmark

AngularJS é perfeito para banco de dados CRUD (Criar Read Update Delete). Basta imaginar que esses objetos foram registrados em um banco de dados.

A diretriz ng-app

A diretiva ng-app define o elemento raiz de uma aplicação AngularJS. A diretiva ng-app irá inicializar automaticamente o aplicativo quando uma página da Web é carregada.

A diretriz ng-init

A diretiva ng-init define valores iniciais para uma aplicação AngularJS. Normalmente, você não usará ng-init. Você usará um controlador ou módulo em vez disso.

A diretriz ng-model

A diretiva ng-model vincula o valor dos controles HTML (input, select, textarea) aos dados do aplicativo. A diretriz ng-model também pode:

* Fornecer validação de tipo para dados de aplicação (número, email, requisito).
* Fornecer o status dos dados do aplicativo (invalid, dirty, touched, error).
* Fornecer classes CSS para elementos HTML.
* Vincular elementos HTML aos formulários HTML.

**Criando novas diretivas**

Além de todas as diretrizes incorporadas do AngularJS, você pode criar suas próprias diretivas. Novas diretivas são criadas usando a função **.directive**. Para invocar a nova diretiva, faça um elemento HTML com o mesmo nome da tag que a nova diretiva. Ao nomear uma diretiva, você deve usar um nome camel case, **w3TestDirective**, mas ao invocá-lo, você deve usar - nome separado, **w3-test-directive**: <body ng-app="myApp">  
<w3-test-directive></w3-test-directive>  
<script>  
var app = angular.module("myApp", []);  
app.directive("w3TestDirective", function() {  
    return {  
        template : "<h1>Made by a directive!</h1>"  
    };  
});  
</script>  
</body>

Você pode invocar uma diretiva usando:

* Nome do elemento <w3-test-directive></w3-test-directive>
* Atributo <div w3-test-directive></div>
* Classe <div class="w3-test-directive"></div>
* Comentário <!-- directive: w3-test-directive -->

Nota: Você deve adicionar o valor "C" à propriedade de restrição para poder invocar a diretiva de um nome de classe: return { restrict: "C",  
      template : "<h1>Made by a directive!</h1>"  
    };

Nota: Você deve adicionar o valor "M" à propriedade de restrição para poder invocar a diretiva de um comentário. Adicionamos a propriedade de substituição neste exemplo, caso contrário o comentário seria invisível: return { restrict: "M", replace: true,  
      template : "<h1>Made by a directive!</h1>"  
    };

**Restrições**

Você pode restringir suas diretrizes somente para serem invocadas por alguns dos métodos. **Exemplo:** Ao adicionar uma propriedade restrita com o valor "A", a diretiva só pode ser invocada por atributos: var app = angular.module("myApp", []);  
app.directive("w3TestDirective", function() {  
    return {  
        restrict : "A",  
        template : "<h1>Made by a directive!</h1>"  
    };  
}); // Para funcionar deve se fazer a chamada pelo atributo <div w3-test-directive></div>

Os valores restritos legais são:

* E para o nome do elemento
* A para Atributo
* C para a classe
* M para comentário

Por padrão, o valor é EA, o que significa que ambos os nomes de elementos e nomes de atributos podem invocar a diretiva.

Com a diretriz ng-model, você pode vincular o valor de um campo de entrada a uma variável criada no AngularJS. <div ng-app="myApp" ng-controller="myCtrl">  
Name: <input ng-model="name">  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.name = "John Doe";  
});  
</script>

**Ligação de duas vias**

A ligação vai em ambos os sentidos. Se o usuário alterar o valor dentro do campo de entrada, a propriedade AngularJS também alterará seu valor: <div ng-app="myApp" ng-controller="myCtrl">  
    Name: <input ng-model="name">  
    <h1>You entered: {{name}}</h1>  
</div>

**Validar a entrada do usuário**

A diretiva ng-model pode fornecer validação de tipo para dados do aplicativo (número, e-mail, requisitado): <form ng-app="" name="myForm">  
    Email:  
     <input type="email" name="myAddress" ng-model="text">  
     <span ng-show="myForm.myAddress.$error.email">Not a valid e-mail address</span>  
</form> // Se o e-mail não conter um @ e um . é mostrada a mensagem

No exemplo acima, o span será exibido somente se a expressão no atributo ng-show retornar true. Se a propriedade no atributo ng-model não existir, o AngularJS criará um para você.

**Status do aplicativo**

O ng-model pode fornecer status para dados do aplicativo (invalid, dirty, touched, error). <form ng-app="" name="myForm" ng-init="myText = 'post@myweb.com'">  
    Email:  
    <input type="email" name="myAddress" ng-model="myText" required>  
    <h1>Status</h1>  
    {{myForm.myAddress.$valid}}  
    {{myForm.myAddress.$dirty}}  
    {{myForm.myAddress.$touched}}  
</form>

**CSS Classes**

A diretiva ng-model fornece classes CSS para elementos HTML, dependendo do status deles: <style> input.ng-invalid {  
    background-color: lightblue;  
}

</style>  
<body>  
  
<form ng-app="" name="myForm">  
    Enter your name:  
    <input name="myName" ng-model="myText" required>  
</form> // Se o input não tiver valor retorna falso e a cor fica azul claro

A diretiva ng-model adiciona / remove as seguintes classes, de acordo com o status do campo de formulário:

* ng-empty
* ng-not-empty
* ng-touched
* ng-untouched
* ng-valid
* ng-invalid
* ng-dirty
* ng-pending
* ng-pristine

DATA BINDING

A ligação de dados (data binding) no AngularJS é a sincronização entre o model e a view.

Data Model

As aplicações AngularJS geralmente possuem um modelo de dados. O modelo de dados é uma coleção de dados disponíveis para a aplicação.

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.firstname = "John";  
    $scope.lastname = "Doe";  
});

Vew HTML

O contêiner HTML onde o aplicativo AngularJS é exibido, é chamado de view. A view tem acesso ao model, e há várias maneiras de exibir os dados do model na view. Você pode usar a diretiva ng-bind, que irá vincular o innerHTML do elemento à propriedade do modelo especificado: <p ng-bind="firstname"></p>

Você também pode usar chaves duplas {{ }} para exibir o conteúdo do model: <p>First name: {{firstname}}</p>

Ou você pode usar a diretiva ng-model em controles HTML para vincular o modelo à view. Use a diretiva ng-model para vincular dados do model à view em controles HTML (input, select, textarea): <input ng-model="firstname">

A diretiva ng-model fornece uma ligação bidirecional entre o model e a view.

**Ligação de dois sentidos (Two-Way binding)**

A ligação de dados no AngularJS é a sincronização entre o model e a view. Quando os dados no model mudam, a view reflete a mudança, e quando os dados na view mudam, o model também é atualizado. Isso acontece de forma imediata e automática, o que garante que o model e a view sejam atualizados em todos os momentos.

CONTROLADORES

As aplicações no AngularJS são controladas por controladores. Devido à sincronização imediata do model e da view, o controlador pode ser completamente separado da view e simplesmente se concentrar nos dados do model. Graças à ligação de dados no AngularJS, a view refletirá quaisquer alterações feitas no controlador.

A diretiva ng-controller define o controlador do aplicativo. Um controlador é um objeto JavaScript, criado por um construtor de objetos JavaScript padrão.

<div ng-app="myApp" ng-controller="myCtrl">  
  
First Name: <input type="text" ng-model="firstName"><br>  
Last Name: <input type="text" ng-model="lastName"><br>  
<br>  
Full Name: {{firstName + " " + lastName}}  
  
</div>  
  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.firstName = "John";  
    $scope.lastName = "Doe";  
});  
</script>

**Exemplo explicado**: O aplicativo AngularJS é definido por **ng-app = "myApp".** O aplicativo é executado dentro do <div>. O atributo **ng-controller = "myCtrl"** é uma diretiva AngularJS. Ele define um controlador. A função myCtrl é uma função JavaScript. AngularJS invocará o controlador com um objeto de escopo $.

Em AngularJS, $ scope é o objeto da aplicação (o proprietário das variáveis e funções do aplicativo). O controlador cria duas propriedades (variáveis) no escopo (firstName e lastName). As diretivas ng-model vinculam os campos de entrada às propriedades do controlador (firstName e lastName).

Métodos do controlador

O exemplo acima demonstrou um objeto controlador com duas propriedades: lastName e firstName. Um controlador também pode ter métodos (variáveis como funções): <div ng-app="myApp" ng-controller="personCtrl">  
First Name: <input type="text" ng-model="firstName"><br>  
Last Name: <input type="text" ng-model="lastName"><br>  
<br>  
Full Name: {{fullName()}}  
</div>  
  
<script>  
var app = angular.module('myApp', []);  
app.controller('personCtrl', function($scope) {  
    $scope.firstName = "John";  
    $scope.lastName = "Doe";  
    $scope.fullName = function() {  
        return $scope.firstName + " " + $scope.lastName;  
    };  
});  
</script>

Controladores em arquivos externos

Em aplicações maiores, é comum armazenar controladores em arquivos externos. Basta copiar o código entre as tags <script> em um arquivo externo chamado personController.js: <div ng-app="myApp" ng-controller="personCtrl">  
First Name: <input type="text" ng-model="firstName"><br>  
Last Name: <input type="text" ng-model="lastName"><br>  
<br>  
Full Name: {{fullName()}}  
</div>  
  
<script src="personController.js"></script>

ESCOPOS

O escopo é a parte vinculante entre o HTML (view) e o JavaScript (controlador). O escopo é um objeto com as propriedades e métodos disponíveis. O escopo está disponível tanto para a view quanto para o controlador.

Como usar o escopo?

Quando você faz um controlador no AngularJS, você passa o objeto **$scope** como um argumento. As propriedades feitas no controlador, podem ser referidas na visualização: <div ng-app="myApp" ng-controller="myCtrl">  
<h1>{{carname}}</h1>  
</div>  
  
<script> var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.carname = "Volvo";  
});  
</script>

Ao adicionar propriedades ao objeto $scope no controlador, a view (HTML) obtém acesso a essas propriedades. Na view, você não usa o prefixo $ do escopo, basta se referir a um nome de propriedade, como {{carname}}.

Compreendendo o escopo

Se considerarmos um aplicativo AngularJS ele é consistido de:

* View, que é o HTML.
* Model, são os dados disponíveis para a view atual.
* Controlador, função JavaScript que faz / muda / remove / controla os dados.

Então o escopo faz parte do Model.

O escopo é um objeto JavaScript com propriedades e métodos, que estão disponíveis tanto para a view como para o controlador. Exemplo: Se você fizer alterações na view, o model e o controlador serão atualizados: <div ng-app="myApp" ng-controller="myCtrl">  
<input ng-model="name">  
<h1>My name is {{name}}</h1>  
</div>  
  
<script> var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.name = "John Doe";  
});  
</script>

Conheça o seu escopo

É importante saber em que escopo você está lidando, a qualquer momento. Nos dois exemplos acima, existe apenas um escopo, portanto, conhecer o escopo não foi um problema, mas para aplicações maiores, pode haver seções no DOM HTML que só podem acessar certos escopos. Exemplo: Ao lidar com a diretiva ng-repeat, cada repetição tem acesso ao objeto de repetição atual: <div ng-app="myApp" ng-controller="myCtrl">  
<ul>  
    <li ng-repeat="x in names">{{x}}</li>  
</ul>  
</div>  
  
<script> var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.names = ["Emil", "Tobias", "Linus"];  
});  
</script>

Cada elemento <li> tem acesso ao objeto de repetição atual, neste caso, uma seqüência de caracteres, a que se refere usando x.

**Escopo Raiz**

Todos os aplicativos têm um $rootScope que é o escopo criado no elemento HTML que contém a diretiva ng-app. O rootScope está disponível em todo o aplicativo. Se uma variável tiver o mesmo nome no escopo atual e no raiz, o aplicativo usa o do escopo atual. Exemplo: Uma variável chamada "cor" existe tanto no escopo do controlador como no raiz do scanner: <body ng-app="myApp">  
<p>The rootScope's favorite color:</p>  
<h1>{{color}}</h1>  
<div ng-controller="myCtrl">  
    <p>The scope of the controller's favorite color:</p>  
    <h1>{{color}}</h1>  
</div>  
<p>The rootScope's favorite color is still:</p>  
<h1>{{color}}</h1>  
  
<script> var app = angular.module('myApp', []);  
app.run(function($rootScope) {  
    $rootScope.color = 'blue';  
});  
app.controller('myCtrl', function($scope) {  
    $scope.color = "red";  
});  
</script>  
</body>

FILTROS

Os filtros podem ser adicionados em AngularJS para formatar dados. Filtros em AngularJS:

* currency - Formata um número para um formato de moeda.
* date - Formata uma data para um formato especificado.
* filter  - Seleciona um subconjunto de itens de uma matriz.
* Json - Formata um objeto para uma string JSON.
* limitTo - Limita uma matriz / string, em um número especificado de elementos / caracteres.
* Lowercase - Formata uma string para minúsculas.
* Number - Formata um número para uma string.
* orderBy - Ordena uma matriz por uma expressão.
* uppercase - Formata uma string para maiúscula.

**Adicionando filtros a expressões**

Os filtros podem ser adicionados às expressões usando o caractere pipe (|), seguido por um filtro: <div ng-app="myApp" ng-controller="personCtrl">  
<p>The name is {{ lastName | uppercase }}</p>  
</div>

**Adicionando Filtros a Diretivas**

Os filtros são adicionados às diretivas, como ng-repeat, usando o caractere do pipe (|), seguido de um filtro. O filtro orderBy classifica uma matriz: <div ng-app="myApp" ng-controller="namesCtrl">  
<ul>  
  <li ng-repeat="x in names | orderBy:'country'">  
    {{ x.name + ', ' + x.country }}  
  </li>  
</ul>  
</div>

<script> angular.module('myApp', []).controller('namesCtrl', function($scope) {$scope.names = [

{name:'Jani',country:'Norway'},

{name:'Carl',country:'Sweden'},

{name:'Margareth',country:'England'},

{name:'Hege',country:'Norway'},

{name:'Joe',country:'Denmark'},

{name:'Gustav',country:'Sweden'},

{name:'Birgit',country:'Denmark'},

{name:'Mary',country:'England'},

{name:'Kai',country:'Norway'}

];

}); // Ordena a matriz por ordem alfabética pelo nome dos países </script>

O filtro de moeda

O filtro de moeda formata um número como moeda: <div ng-app="myApp" ng-controller="costCtrl">  
<h1>Price: {{ price | currency }}</h1>  
</div>

<script> var app = angular.module('myApp', []); app.controller('costCtrl', function($scope) { $scope.price = 58; }); </script> // retorna Price: $58,00

O filtro Filter

O filtro filter seleciona um subconjunto de uma matriz. Só pode ser usado em arrays, e ele retorna uma matriz contendo apenas os itens correspondentes. Exemplo: Retornar os nomes que contêm a letra "i": <div ng-app="myApp" ng-controller="namesCtrl">  
<ul>  
  <li ng-repeat="x in names | filter : 'i'">  
    {{ x }}  
  </li>  
</ul>  
</div>

<script> angular.module('myApp', []).controller('namesCtrl', function($scope) { $scope.names = [ 'Jani', 'Carl', 'Margareth', 'Hege', 'Joe', 'Gustav', 'Birgit', 'Mary', 'Kai' ]; }); </script> // retorna Jani, Birgit, Kai

Filtrar uma matriz baseada na entrada do usuário

Ao definir a diretiva ng-model em um campo de entrada, podemos usar o valor do campo de entrada como uma expressão em um filtro. Digite uma letra no campo de entrada e a lista encolherá / crescerá dependendo da partida: <div ng-app="myApp" ng-controller="namesCtrl">  
<p><input type="text" ng-model="test"></p>  
<ul>  
  <li ng-repeat="x in names | filter : test">  
    {{ x }}  
  </li>  
</ul>  
</div>

<script> angular.module('myApp', []).controller('namesCtrl', function($scope) { $scope.names = [ 'Jani', 'Carl', 'Margareth', 'Hege', 'Joe', 'Gustav', 'Birgit', 'Mary', 'Kai' ]; }); </script>

Classificar uma matriz baseada na entrada do usuário

Ao adicionar a diretiva ng-click nos cabeçalhos da tabela, podemos executar uma função que altera a ordem de classificação da matriz: <div ng-app="myApp" ng-controller="namesCtrl">  
<table border="1" width="100%">  
  <tr>  
    <th ng-click="orderByMe('name')">Name</th>  
    <th ng-click="orderByMe('country')">Country</th>  
  </tr>  
  <tr ng-repeat="x in names | orderBy:myOrderBy">  
    <td>{{x.name}}</td>  
    <td>{{x.country}}</td>  
  </tr>  
</table>  
</div>

<script>  
angular.module('myApp', []).controller('namesCtrl', function($scope) {  
  $scope.names = [  
    {name:'Jani',country:'Norway'},  
    {name:'Carl',country:'Sweden'},  
    {name:'Margareth',country:'England'},  
    {name:'Hege',country:'Norway'},  
    {name:'Joe',country:'Denmark'},  
    {name:'Gustav',country:'Sweden'},  
    {name:'Birgit',country:'Denmark'},  
    {name:'Mary',country:'England'},  
    {name:'Kai',country:'Norway'}  
  ];  
  $scope.orderByMe = function(x) {  
    $scope.myOrderBy = x;  
  }  
});  
</script>

Também pode-se criar filtros próprios registrando uma nova função de fábrica de filtro com seu módulo.

# SERVIÇOS

Em AngularJS você pode fazer seu próprio serviço, ou usar um dos muitos serviços internos. No AngularJS, um serviço é uma função, ou objeto, que está disponível para e limitado a sua aplicação AngularJS. O AngularJS possui cerca de 30 serviços integrados. Um deles é o serviço $location. O serviço $location possui métodos que retornam informações sobre a localização da página da Web atual, use o serviço $location em um controlador: var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $location) {  
    $scope.myUrl = $location.absUrl();  
});

Observe que o serviço $ location é passado para o controlador como um argumento. Para usar o serviço no controlador, ele deve ser definido como uma dependência.

**Por que usar os Serviços?**

Para muitos serviços, como o serviço $ location, parece que você pode usar objetos que já estão no DOM, como o objeto window.location, e você poderia, mas teria algumas limitações, pelo menos para o seu aplicativo AngularJS.

O AngularJS supervisiona constantemente sua aplicação e, para que ela manipule mudanças e eventos adequadamente, o AngularJS prefere que você use o serviço $ location em vez do objeto window.location.

O serviço $http

O serviço $http é um dos serviços usados mais comuns em aplicações AngularJS. O serviço faz uma solicitação ao servidor e permite que seu aplicativo lida com a resposta.

Use o serviço $http para solicitar dados do servidor: <div ng-app="myApp" ng-controller="myCtrl">   
<p>Today's welcome message is:</p>  
<h1>{{myWelcome}}</h1>  
</div>

<script> var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("welcome.htm").then(function (response) {  
        $scope.myWelcome = response.data;  
    });  
}); </script>

O serviço $http solicita uma página no servidor e a resposta é definida como o valor da variável "myWelcome".

**Métodos**

O exemplo acima usa o método .get do serviço $http. O método .get é um método de atalho do serviço $http. Existem vários métodos de atalho para se chamar um serviço $http:

* .delete()
* .get()
* .head()
* .jsonp()
* .patch()
* .post()
* .put()

**Propriedades**

A resposta do servidor é um objeto com essas propriedades:

* .config - o objeto usado para gerar a solicitação.
* .data - uma string ou um objeto, carregando a resposta do servidor.
* .headers - uma função para usar para obter informações de cabeçalho.
* .status - um número que define o status HTTP.
* .statusText - uma string que define o status HTTP.

<div ng-app="myApp" ng-controller="myCtrl">  <p>Data : {{content}}<p> <p>Status : {{statuscode}}</p> <p>StatusText : {{statustext}}</p> </div> <p>O objeto response possui muitas propriedades. Este exemplo demonstra o valor das propriedades de dados, status e statusText.p>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("welcome.htm")  
    .then(function(response) {  
        $scope.content = response.data;  
        $scope.statuscode = response.status;  
        $scope.statustext = response.statustext;   
    });  
}); //retorna Data : Hello AngularJS Students Status : 200 StatusText:

Para lidar com erros, adicione mais uma função ao método .then:

<div ng-app="myApp" ng-controller="myCtrl">  <h1>{{content}}</h1> </div> <p>O serviço $ http executa funções diferentes em caso de sucesso e falha.</p>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("wrongfilename.htm")  
    .then(function(response) {  
        //First function handles success  
        $scope.content = response.data;  
    }, function(response) {  
        //Second function handles error  
        $scope.content = "Something went wrong";  
    });  
}); // retorna "Something went wrong"

**JSON**

Os dados que você obtém da resposta esperam estar no formato JSON. JSON é uma ótima maneira de transportar dados, e é fácil de usar dentro do AngularJS, ou qualquer outro JavaScript. Exemplo: no servidor, temos um arquivo que retorna um objeto JSON contendo 15 clientes, todos envolvidos em conjuntos chamados de registros. A diretiva ng-repeat é perfeita para fazer o loop através de uma matriz:

<div ng-app="myApp" ng-controller="customersCtrl">   
<ul>  
  <li ng-repeat="x in myData">  
    {{ x.Name + ', ' + x.Country }}  
  </li>  
</ul>  
</div>  
  
<script>

var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $http) {  
    $http.get("customers.php").then(function(response) {  
        $scope.myData = response.data.records;  
    });  
});  
</script> // retorna uma lista

**Aplicação explicada**: O aplicativo define o controlador customersCtrl, com um $ scope e $ http. $ Http é um objeto XMLHttpRequest para solicitar dados externos. $ Http.get () lê dados JSON de *https://www.w3schools.com/angular/customers.php*. No sucesso, o controlador cria uma propriedade, myData, no escopo, com dados JSON do servidor.

O serviço $timeout

O serviço $ timeout é a versão AngularJS da função window.setTimeout. Exemplo: Exibir uma nova mensagem após dois segundos: <div ng-app="myApp" ng-controller="myCtrl">  <p>This header will change after two seconds:</p> <h1>{{ myHeader }}</h1> </div>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $timeout) {  
    $scope.myHeader = "Hello World!";  
    $timeout(function () {  
        $scope.myHeader = "How are you today?";  
    }, 2000);  
});

O serviço $interval

O serviço $ interval é a versão AngularJS da função window.setInterval. Exemplo: Exibir o tempo a cada segundo: <div ng-app="myApp" ng-controller="myCtrl">  <p>The time is:</p> <h1>{{ theTime }}</h1> </div>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $interval) {  
    $scope.theTime = new Date().toLocaleTimeString();  
    $interval(function () {  
        $scope.theTime = new Date().toLocaleTimeString();  
    }, 1000);  
});

**Criando um serviço próprio**

Para criar seu próprio serviço, conecte seu serviço ao módulo:

app.service('hexafy', function() {  
    this.myFunc = function (x) {  
        return x.toString(16);  
    }  
});

Para usar seu serviço personalizado, adicione-o como uma dependência ao definir o controlador. Exemplo: Use o serviço personalizado chamado hexafy para converter um número em um número hexadecimal: <div ng-app="myApp" ng-controller="myCtrl">  <p>The hexadecimal value of 255 is:</p> <h1>{{ hex }}</h1> </div>

app.controller('myCtrl', function($scope, **hexafy**) {  
    $scope.hex = **hexafy**.myFunc(255);  
}); // retorna ff

**Usar um serviço personalizado dentro de um filtro**

Depois de criar um serviço e conectá-lo ao seu aplicativo, você pode usar o serviço em qualquer controlador, diretiva, filtro ou mesmo dentro de outros serviços. Para usar o serviço dentro de um filtro, adicione-o como uma dependência ao definir o filtro. Exemplo: O serviço hexafy usado no filtro myFormat: app.filter('myFormat',[**'hexafy'**, function(**hexafy**) {  
    return function(x) {  
        return **hexafy**.myFunc(x);  
    };  
}]);

Você pode usar o filtro ao exibir valores de um objeto ou uma matriz: <div ng-app="myApp" ng-controller="myCtrl">  <p>Use a filter when displaying the array [255, 251, 200]:</p> <ul> <li>ng-repeat="x in counts"{{x | myFormat}}</li> <ul> <p>This filter uses a service that converts numbers into hexadecimal values. </div> // retorna ff fb c8

</script> var app = angular.module('myApp', []); app.service('hexafy', function() { this.myFunc = function (x) { return x.toString(16); } });

app.filter('myFormat',[ 'hexafy', function(hexafy) {

return function(x) { return hexafy.myFunc(x); }; }]);

app.controller('myCtrl', function($scope) { $scope.counts = [255, 251, 200]; }); </script>

TABELAS

Mostrar tabelas com angular é muito simples. A diretiva ng-repeat é perfeita para exibi- lás. <div ng-app="myApp" ng-controller="customersCtrl">   
<table>  
  <tr ng-repeat="x in names">  
    <td>{{ x.Name }}</td>  
    <td>{{ x.Country }}</td>  
  </tr>  
</table>  
</div>  
  
<script>  
var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $http) {  
    $http.get("customers.php")  
    .then(function (response) {$scope.names = response.data.records;});  
});  
</script> // retorna uma lista dentro de uma tabela de um arquivo externo chamado customers.php

Para ordenar a tabela, adicione um filtro orderBy: <table>  
  <tr ng-repeat="x in names | orderBy : 'Country'">  
    <td>{{ x.Name }}</td>  
    <td>{{ x.Country }}</td>  
  </tr>  
</table>

Para exibir o índice da tabela, adicione um <td> com $ index: <table>  
  <tr ng-repeat="x in names">  
    <td>{{ $index + 1 }}</td>  
    <td>{{ x.Name }}</td>  
    <td>{{ x.Country }}</td>  
  </tr>  
</table>

Usando $ even e $ odd: <table>  
<tr ng-repeat="x in names">  
<td ng-if="$odd" style="background-color:#f1f1f1">{{ x.Name }}</td>  
<td ng-if="$even">{{ x.Name }}</td>  
<td ng-if="$odd" style="background-color:#f1f1f1">{{ x.Country }}</td>  
<td ng-if="$even">{{ x.Country }}</td>  
</tr>  
</table>

BOXES SELECT

AngularJS permite que você crie listas suspensas com base em itens em uma matriz ou um objeto.Para isto, você deve usar a diretiva ng-options: <div ng-app="myApp" ng-controller="myCtrl">  
<select ng-model="selectedName" ng-options="x for x in names">  
</select>  
</div>  
  
<script> var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.names = ["Emil", "Tobias", "Linus"];  
});  
</script>

Você também pode usar a diretiva ng-repeat para fazer a mesma lista suspensa: <select>  
<option ng-repeat="x in names">{{x}}</option>  
</select>

Como a diretiva ng-repeat repete um bloco de código HTML para cada item em uma matriz, ele pode ser usado para criar opções em uma lista suspensa, mas a diretriz ng-options foi feita especialmente para preencher uma lista suspensa com opções e tem pelo menos uma vantagem importante:

**Dropdowns feitos com ng-options permite que o valor selecionado seja um objeto, enquanto os menus suspensos feitos a partir de ng-repeat devem ser uma string.**

O que usar?

Suponha que você tenha uma série de objetos: $scope.cars = [  
    {model : "Ford Mustang", color : "red"},  
    {model : "Fiat 500", color : "white"},  
    {model : "Volvo XC90", color : "black"}  
];

A diretiva ng-repeat tem suas limitações, o valor selecionado deve ser uma string: <select ng-model="selectedCar">  
<option ng-repeat="x in cars" value="{{x.model}}">{{x.model}}</option>  
</select>  
<h1>You selected: {{selectedCar}}</h1>

Neste exemplo você terá que escolher entre a cor ou o modelo para ter o valor selecionado.

Ao usar a diretiva ng-options, o valor selecionado pode ser um objeto: <select ng-model="selectedCar" ng-options="x.model for x in cars">  
</select>  
<h1>You selected: {{selectedCar.model}}</h1>  
<p>Its color is: {{selectedCar.color}}</p>

Quando o valor selecionado pode ser um objeto, ele pode conter mais informações, e sua aplicação pode ser mais flexível.

**A fonte de dados como um objeto**

Nos exemplos anteriores, a fonte de dados era uma matriz, mas também podemos usar um objeto. Suponha que você tenha um objeto com pares chave-valor: $scope.cars = {  
    car01 : "Ford",  
    car02 : "Fiat",  
    car03 : "Volvo"  
};

A expressão no atributo ng-options é um pouco diferente para objetos. Exemplo: Usando um objeto como fonte de dados, x representa a chave e y representa o valor: <select ng-model="selectedCar" ng-options="**x for (x, y) in cars**">  
</select>  
<h1>You selected: {{selectedCar}}</h1> // na caixa de seleção será mostrado car01 ou car02 ou car03

O valor selecionado será sempre o valor em um par chave-valor. O valor em um par chave-valor também pode ser um objeto. Exemplo: O valor selecionado ainda será o valor em um par de chave-valor, só que desta vez é um objeto: <div ng-app="myApp" ng-controller="myCtrl"> <p>Select a car: </p> <select ng-model="selectedCar" ng-options="**x for (x, y) in cars**">  
</select>  
<h1>You selected: {{selectedCar.brand}}</h1> <h1>Model: {{selectedCar.model}} and color: {{selectedCar.color}}</h1>

$scope.cars = {  
car01 : {brand : "Ford", model : "Mustang", color : "red"},  
car02 : {brand : "Fiat", model : "500", color : "white"},  
car03 : {brand : "Volvo", model : "XC90", color : "black"}  
};

As opções na lista suspensa não precisam ser a chave em um par de chave-valor, também pode ser o valor ou uma propriedade do objeto de valor: <select ng-model="selectedCar" ng-options="**y.brand** for (x, y) in cars">  
</select>

DOM

O AngularJS possui diretivas para vincular os dados do aplicativo aos atributos dos elementos HTML DOM.

A diretiva ng-disabled

A diretiva ng-disabled vincula os dados do aplicativo AngularJS ao atributo desativado de elementos HTML. <div ng-app="" ng-init="mySwitch=true">  
<p><button ng-disabled="mySwitch">Click Me!</button></p>  
<p><input type="checkbox" ng-model="mySwitch">Button</p>  
<p>  
{{ mySwitch }}  
</p>  
</div>

**Aplicação explicada**: A diretiva ng-disabled vincula os dados do aplicativo mySwitch ao atributo desativado do botão HTML. A diretiva ng-model vincula o valor do elemento da caixa de seleção HTML ao valor de mySwitch. Se o valor do mySwitch for verdadeiro, o botão será desativado.

A diretiva ng-show

A diretiva ng-show mostra ou esconde um elemento HTML. <div ng-app="">  
<p ng-show="true">I am visible.</p>  
<p ng-show="false">I am not visible.</p>  
</div>

A diretiva ng-show mostra (ou esconde) um elemento HTML com base no seu valor. Você pode usar qualquer expressão que seja avaliada como verdadeira ou falsa: <div ng-app="" ng-init="hour=13">  
<p ng-show="hour > 12">I am visible.</p>  
</div>

A diretiva ng-hide

A diretiva ng-hide esconde ou mostra um elemento HTML. <div ng-app="">  
<p ng-hide="true">I am not visible.</p>  
<p ng-hide="false">I am visible.</p>  
</div>

FORMULÁRIOS

Os formulários no AngularJS fornecem ligação e validação de dados de controles de entrada. Os controles de entrada são os elementos de entrada HTML:

* elementos de input
* elementos de select
* elementos de buttons
* elementos de textarea

**Ligação de dados(Data binding)**

Os controles de entrada fornecem ligação de dados usando a diretiva ng-model: <input type="text" ng-model="firstname">

O aplicativo agora possui uma propriedade chamada firstname. A diretiva ng-model vincula o controlador de entrada ao restante da sua aplicação. O nome da propriedade, pode ser referido em um controlador: <script>  
var app = angular.module('myApp', []);  
app.controller('formCtrl', function($scope) {  
    $scope.firstname = "John";  
});  
</script>

Também pode ser referido em outro lugar no aplicativo: <form>  
First Name: <input type="text" ng-model="firstname">  
</form>  
<h1>You entered: {{firstname}}</h1>

Checkbox

Uma caixa de seleção tem o valor verdadeiro ou falso. Aplique a diretiva ng-model em uma caixa de seleção e use seu valor em seu aplicativo. Exemplo - Mostre o cabeçalho se a caixa de seleção estiver marcada: <form>  
    Check to show a header:  
    <input type="checkbox" ng-model="myVar">  
</form>  
<h1 ng-show="myVar">My Header</h1>

Radio buttons

Vincule botões de rádio para sua aplicação com a diretriz ng-model. Os botões de rádio com o mesmo ng-model podem ter valores diferentes, mas somente o selecionado será usado. Exemplo - Visualize algum texto, com base no valor do botão de opção selecionado:   <form>  
Pick a topic:  
<input type="radio" ng-model="myVar" value="dogs">Dogs  
<input type="radio" ng-model="myVar" value="tuts">Tutorials  
<input type="radio" ng-model="myVar" value="cars">Cars  
</form> // O valor do myVar será dogs, tuts ou cars.

<div ng-switch="myVar">  
<div ng-switch-when="dogs">  
<h1>Dogs</h1> <p> Welcome to a world of dogs.</p>  
</div>

<div ng-switch-when="tuts">  
<h1>Tutorials</h1> <p> Learn from examples.</p>  
</div>

<div ng-switch-when="cars">  
<h1>Cars</h1> <p> Read about cars.</p>  
</div> </div>

Selectbox

Vincule as caixas de seleção à sua aplicação com a diretriz ng-model. A propriedade definida no atributo ng-model terá o valor da opção selecionada na caixa de seleção. Exemplo - Visualize algum texto, com base no valor da opção selecionada: <form>  
Select a topic:  
<select ng-model="myVar">  
    <option value="">  
    <option value="dogs">Dogs  
    <option value="tuts">Tutorials  
    <option value="cars">Cars  
</select>  
</form> // O valor do myVar será dogs, tuts ou cars.

<div ng-switch="myVar">  
<div ng-switch-when="dogs">  
<h1>Dogs</h1> <p> Welcome to a world of dogs.</p>  
</div>

<div ng-switch-when="tuts">  
<h1>Tutorials</h1> <p> Learn from examples.</p>  
</div>

<div ng-switch-when="cars">  
<h1>Cars</h1> <p> Read about cars.</p>  
</div> </div>

Código de Aplicação

<div ng-app="myApp" ng-controller="formCtrl">  
  <form novalidate>  
    First Name:<br>  
    <input type="text" ng-model="user.firstName"><br>  
    Last Name:<br>  
    <input type="text" ng-model="user.lastName">  
    <br><br>  
    <button ng-click="reset()">RESET</button>  
  </form>  
  <p>form = {{user}}</p>  
  <p>master = {{master}}</p>  
</div>  
  
<script>  
var app = angular.module('myApp', []);  
app.controller('formCtrl', function($scope) {  
    $scope.master = {firstName: "John", lastName: "Doe"};  
    $scope.reset = function() {  
        $scope.user = angular.copy($scope.master);  
    };  
    $scope.reset();  
});  
</script>

**Exemplo Explicado**: A diretiva ng-app define o aplicativo AngularJS. A diretiva ng-controller define o controlador do aplicativo. A diretiva ng-model vincula dois elementos de entrada ao objeto do usuário no modelo. O controlador formCtrl define valores iniciais para o objeto mestre e define o método reset (). O método reset () define o objeto do usuário igual ao objeto mestre. A diretiva ng-click invoca o método reset (), somente se o botão for clicado.

O atributo novalidate não é necessário para este aplicativo, mas normalmente você o usará em formulários AngularJS, para substituir a validação HTML5 padrão.O atributo novalidate é novo no HTML5. Desabilita qualquer validação padrão do navegador.

**Validação de formulário**

AngularJS oferece validação de formulário do lado do cliente, podendo assim validar dados de entrada. AngularJS monitora o estado dos campos de formulário e entrada (input, textarea, select) e permite que você notifique o usuário sobre o estado atual.

O AngularJS também contém informações sobre se eles foram tocados e modificados ou não. Você pode usar atributos HTML5 padrão para validar a entrada, ou você pode criar suas próprias funções de validação. A validação do lado do cliente não pode isolar a entrada do usuário segura. A validação do lado do servidor também é necessária.

Required

Use o atributo HTML5 necessário para especificar que o campo de entrada deve ser preenchido. Exemplo - O campo de entrada é necessário: <form name="myForm">  
<input name="myInput" ng-model="myInput" required>  
</form>  
<p>The input's valid state is:</p>  
<h1>{{myForm.myInput.$valid}}</h1>

Email

Use o email do tipo HTML5 para especificar que o valor deve ser um e-mail. Exemplo - O campo de entrada deve ser um e-mail: <form name="myForm">  
<input name="myInput" ng-model="myInput" type="email">  
</form>  
<p>The input's valid state is:</p>  
<h1>{{myForm.myInput.$valid}}</h1>

**Estado do formulário e estado de entrada**

AngularJS está atualizando constantemente o estado dos campos de formulário e de entrada. Campos de entrada têm os seguintes estados:

* $ untouched - O campo ainda não foi tocado
* $ touched - O campo foi tocado
* $ pristine - O campo ainda não foi modificado
* $ dirty - O campo foi modificado
* $ invalid - O conteúdo do campo não é válido
* $ valid - O conteúdo do campo é válido

Os formulários têm os seguintes estados:

* $ pristine - Nenhum campo foi modificado ainda
* $ dirty - Um ou mais foram modificados
* $ invalid - O conteúdo do formulário não é válido
* $ valid - O conteúdo do formulário é válido
* $ submitted - O formulário é enviado

Você pode usar esses estados para mostrar mensagens significativas para o usuário. Exemplo, se for necessário um campo e o usuário o deixa em branco, você deve avisar o usuário -

Mostrar uma mensagem de erro se o campo tiver sido tocado E estiver vazio: <input name="myName" ng-model="myName" required>  
<span ng-show="myForm.myName.$touched && myForm.myName.$invalid">The name is required.</span>

Classes CSS

AngularJS adiciona classes CSS a formulários e campos de entrada dependendo de seus estados. As seguintes classes são adicionadas ou removidas aos campos de entrada:

* ng-untouched - O campo ainda não foi tocado
* ng-touched - O campo foi tocado
* ng-pristine - O campo ainda não foi modificado
* ng-dirty - O campo foi modificado
* ng-valid - O conteúdo do campo é válido
* ng-invalid - O conteúdo do campo não é válido
* ng-valid-key - Uma tecla para cada validação. Exemplo: ng-valid-required, útil quando há mais de uma coisa que deve ser validada
* ng-invalid-key - Exemplo: ng-invalid-required

As seguintes classes são adicionadas ou removidas de formulários:

* ng-pristine - Nenhum campo ainda não foi modificado
* ng-dirty - Um ou mais campos foram modificados
* ng-valid - O conteúdo do formulário é válido
* ng-invalid - O conteúdo do formulário não é válido
* ng-válido-key - Uma tecla para cada validação. Exemplo: ng-valid-required, útil quando há mais de uma coisa que deve ser validada
* ng-invalid-key - Exemplo: ng-invalid-required

As classes são removidas se o valor que representam é falso. Adicione estilos para essas classes para dar ao seu aplicativo uma interface de usuário melhor e mais intuitiva. Exemplo -Aplicar estilos, usando CSS padrão: <style> input.ng-invalid {  
    background-color: pink;  
}  
input.ng-valid {  
    background-color: lightgreen;  
}  
</style>

Os formulários também podem ser estilizados: Exemplo - Aplicar estilos para formas não modificadas (pristine) e para formas modificadas: <style> form.ng-pristine {  
    background-color: lightblue;  
}  
form.ng-dirty {  
    background-color: pink;  
}  
</style>

**Validação personalizada**

Para criar sua própria função de validação é um pouco mais complicado. Você deve adicionar uma nova diretiva ao seu aplicativo e lidar com a validação dentro de uma função com certos argumentos especificados.

Exemplo - Crie sua própria diretiva, contendo uma função de validação personalizada, e consulte-o usando a minha diretiva. O campo só será válido se o valor contiver o caractere "e": <form name="myForm">  
<input name="myInput" ng-model="myInput" required my-directive>  
</form>  
  
<script> var app = angular.module('myApp', []);  
app.directive('myDirective', function() {  
  return {  
    require: 'ngModel',  
    link: function(scope, element, attr, mCtrl) {  
      function myValidation(value) {  
        if (value.indexOf("e") > -1) {  
          mCtrl.$setValidity('charE', true);  
        } else {  
          mCtrl.$setValidity('charE', false);  
        }  
        return value;  
      }  
      mCtrl.$parsers.push(myValidation);  
    }  
  };  
});  
</script>

**Exemplo Explicado:** Em HTML, a nova diretiva será referida usando o atributo my-directive.No JavaScript, começamos por adicionar uma nova diretiva chamada myDirective.Lembre-se, ao nomear uma diretiva, você deve usar um nome **camel case**, myDirective, mas ao invocá-lo, você deve usar - separando o nome, my-directive.

Em seguida, retorne um objeto onde você especifica que exige o ngModel, que é o ngModelController. Faça uma função de ligação que leve alguns argumentos, onde o quarto argumento, mCtrl, é o ngModelController, em seguida, especifique uma função, neste caso chamada myValidation, que leva um argumento, esse argumento é o valor do elemento de entrada.

Teste se o valor contiver a letra "e" e defina a validade do controlador do modelo como verdadeiro ou falso. Por fim, mCtrl. $ Parsers.push (myValidation); Irá adicionar a função myValidation a uma matriz de outras funções, que serão executadas sempre que o valor de entrada for alterado.

<h2>Validation Example</h2>  
<form  ng-app="myApp"  ng-controller="validateCtrl"  
name="myForm" novalidate>  
<p>Username:<br>  
  <input type="text" name="user" ng-model="user" required>  
  <span style="color:red" ng-show="myForm.user.$dirty && myForm.user.$invalid">  
  <span ng-show="myForm.user.$error.required">Username is required.</span>  
  </span>  
</p>  
<p>Email:<br>  
  <input type="email" name="email" ng-model="email" required>  
  <span style="color:red" ng-show="myForm.email.$dirty && myForm.email.$invalid">  
  <span ng-show="myForm.email.$error.required">Email is required.</span>  
  <span ng-show="myForm.email.$error.email">Invalid email address.</span>  
  </span>  
</p>  
<p>  
  <input type="submit"  
  ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||  
  myForm.email.$dirty && myForm.email.$invalid">  
</p>  
</form>

<script>  
var app = angular.module('myApp', []);  
app.controller('validateCtrl', function($scope) {  
    $scope.user = 'John Doe';  
    $scope.email = 'john.doe@gmail.com';  
});  
</script>  
**Exemplo Explicado**: A directiva AngularJS ng-model liga os elementos de entrada ao modelo. O objeto modelo tem duas propriedades: usuário e email. Por causa do ng-show, os intervalos com cor: vermelho são exibidos somente quando o usuário ou o email são $ dirty e $ invalid.

INCLUDES

Com o AngularJS, você pode incluir HTML de um arquivo externo usando a diretiva ng-include. <body ng-app="">  
<div ng-include="'myFile.htm'"></div>  
</body>

Incluir o código AngularJS

Os arquivos HTML que são incluidos com a diretiva ng-include também podem conter códigos: <table>  
<tr ng-repeat="x in names">  
<td>{{ x.Name }}</td>  
<td>{{ x.Country }}</td>  
</tr>  
</table>

Inclua o arquivo "myTable.htm" em sua página da Web e todo o código AngularJS será executado, mesmo o código dentro do arquivo incluído: <body>  
<div ng-app="myApp" ng-controller="customersCtrl">   
  <div ng-include="'myTable.htm'"></div>  
</div>  
<script> var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $http) {  
    $http.get("customers.php").then(function (response) {  
        $scope.names = response.data.records;  
    });  
});  
</script>

**Incluir Cross Domains**

Por padrão, a diretiva ng-include não permite que você inclua arquivos de outros domínios. Para incluir arquivos de outro domínio, você pode adicionar uma lista branca de arquivos legais e / ou domínios na função de configuração de seu aplicativo. Exemplo - <body ng-app="myApp">  
<div ng-include="'https://tryit.w3schools.com/angular\_include.php'"></div>  
<script> var app = angular.module('myApp', [])  
app.config(function($sceDelegateProvider) {  
    $sceDelegateProvider.resourceUrlWhitelist([  
        'https://tryit.w3schools.com/\*\*'  
    ]); });  
</script>  
</body>

ROUTING

O módulo ngRoute ajuda seu aplicativo a tornar-se um aplicativo de página única.

**O que é roteamento em AngularJS?**

Se você quiser navegar para páginas diferentes em seu aplicativo, mas você também quer que o aplicativo seja um SPA (Single Page Aplication - Aplicação de página única), sem recarregar a página, você pode usar o módulo ngRoute.

O módulo ngRoute encaminha seu aplicativo para páginas diferentes sem recarregar todo o aplicativo. Exemplo - Navegue até "red.htm", "green.htm" e "blue.htm": <body ng-app="myApp">  
  
<p><a href="#/!">Main</a></p>  
  
<a href="#!red">Red</a>  
<a href="#!green">Green</a>  
<a href="#!blue">Blue</a>  
  
<div ng-view></div>  
  
<script>  
var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
    $routeProvider  
    .when("/", {  
        templateUrl : "main.htm"  
    })  
    .when("/red", {  
        templateUrl : "red.htm"  
    })  
    .when("/green", {  
        templateUrl : "green.htm"  
    })  
    .when("/blue", {  
        templateUrl : "blue.htm"  
    });  
});  
</script>  
</body>

**O que é preciso?**

Para preparar seus aplicativos para o roteamento, você deve incluir o módulo AngularJS Route: <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular-route.js"></script>

Em seguida, você deve adicionar o ngRoute como uma dependência no módulo de aplicação: var app = angular.module("myApp", ["ngRoute"]);

Agora, seu aplicativo tem acesso ao módulo de rotas, que fornece o $routeProvider. Use o $routeProvider para configurar rotas diferentes em seu aplicativo: app.config(function($routeProvider) {  
  $routeProvider  
  .when("/", {  
    templateUrl : "main.htm"  
  })  
  .when("/red", {  
    templateUrl : "red.htm"  
  })  
  .when("/green", {  
    templateUrl : "green.htm"  
  })  
  .when("/blue", {  
    templateUrl : "blue.htm"  
  });  
});

**Onde isso vai?**

Seu aplicativo precisa de um contêiner para colocar o conteúdo fornecido pelo roteamento. Este contêiner é a diretiva ng-view. Existem três maneiras diferentes de incluir a diretriz ng-view em seu aplicativo:

* <div ng-view></div>
* <ng-view></ng-view>
* <div class="ng-view"></div>

As aplicações só podem ter uma diretiva ng-view, e este será o espaço reservado para todas as visualizações fornecidas pela rota.

$RouteProvider

Com o $routeProvider, você pode definir a página a ser exibida quando um usuário clicar em um link. Exemplo: Defina um $routeProvider: var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
    $routeProvider  
    .when("/", {  
        templateUrl : "main.htm"  
    })  
    .when("/london", {  
        templateUrl : "london.htm"  
    })  
    .when("/paris", {  
        templateUrl : "paris.htm"  
    });  
});

Defina o $routeProvider usando o método de configuração de sua aplicação. O trabalho registrado no método de configuração será executado quando o aplicativo estiver sendo carregado.

Controladores

Com o $routeProvider, você também pode definir um controlador para cada "visualização". Exemplo- Adicionar controladores: var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
    $routeProvider  
    .when("/", {  
        templateUrl : "main.htm"  
    })  
    .when("/london", {  
        templateUrl : "london.htm",  
        controller : "londonCtrl"  
    })  
    .when("/paris", {  
        templateUrl : "paris.htm",  
        controller : "parisCtrl"  
    });  
});  
app.controller("londonCtrl", function ($scope) {  
    $scope.msg = "I love London";  
});  
app.controller("parisCtrl", function ($scope) {  
    $scope.msg = "I love Paris";  
});

Os "londres.htm" e "paris.htm" são arquivos HTML normais, que você pode adicionar expressões AngularJS como faria com outras seções HTML do seu aplicativo AngularJS. Os arquivos se parecem com isto: <h1>London</h1>  
<h3>London is the capital city of England.</h3>  
<p>It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>  
<p>{{msg}}</p>

<h1>Paris</h1>  
<h3>Paris is the capital city of France.</h3>  
<p>The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants.</p>  
<p>{{msg}}</p>

Template

Nos exemplos anteriores, usamos a propriedade templateUrl no método $routeProvider.when.

Você também pode usar a propriedade do modelo, que permite escrever HTML diretamente no valor da propriedade e não se referir a uma página. Exemplo - Escreva templates: var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
    $routeProvider  
    .when("/", {  
        template : "<h1>Main</h1><p>Click on the links to change this content</p>"  
    })  
    .when("/banana", {  
        template : "<h1>Banana</h1><p>Bananas contain around 75% water.</p>"  
    })  
    .when("/tomato", {  
        template : "<h1>Tomato</h1><p>Tomatoes contain around 95% water.</p>"  
    });  
});:

O método otherwise

Nos exemplos anteriores, usamos o método when do $routeProvider. Você também pode usar o método otherwise, que é a rota padrão quando nenhum dos outros recebe uma correspondência. Exemplo - Se nem o "Banana" nem o link "Tomate" foram clicados, informe: var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
   $routeProvider  
    .when("/banana", {  
        template : "<h1>Banana</h1><p>Bananas contain around 75% water.</p>"  
    })  
    .when("/tomato", {  
        template : "<h1>Tomato</h1><p>Tomatoes contain around 95% water.</p>"  
    })  
    .otherwise({  
        template : "<h1>None</h1><p>Nothing has been selected</p>"  
    });  
});