UNINOVE

Introdução ao Framework Angular: Conceitos, Instalação e Configuração

REALIZAR UMA BREVE INTRODUÇÃO SOBRE A TECNOLOGIA ANGULAR JS PARA O DESENVOLVIMENTO DE INTERFACES DINÂMICAS.

AUTOR(A): PROF. EDSON MELO DE SOUZA

1. Introdução

A evolução da internet deu um enorme salto na última década no que diz respeito a forma de como as aplicações estão sendo desenvolvidas e, devido a isto, diversas tecnologias surgiram tanto para o *Front-End* como para o *Back-End*.

O avanço se deu na medida em que as aplicações necessitavam atender aos usuários, apresentando mais desempenho e praticidade no acesso aos dados remotos. Desta necessidade, o AJAX foi fortemente incorporado nas aplicações web por não necessitar do recarregamento da página para atualizar informações, mas, por outro lado, esta tecnologia não possui padronização para implementação, além de ser incompatível com alguns *frameworks*.

Para superar todos, ou grande parte dos problemas, eis que surge o Angular JS, trazendo como proposta a redução de linhas de código, aumento do desempenho, facilidade de uso, entre outros.

Neste tópico veremos uma breve introdução a este *framework* que está revolucionando o método de desenvolver aplicações para a internet.

2. O Angular JS

O Angular JS ou simplesmente Angular é um *Framework* MVC (Model – View – Controller) baseado em JavaScript, mas pode ser entendido também como um MV*, ou seja, aceita mais de uma camada para implementação (ANGULAR JS, 2017) .

Sua principal característica é permitir a criação de aplicações *Front-End* com base na web e Mobile. O "pulo do gato" do Angular é estender o HTML, permitindo uma interação altamente dinâmica. Além dessa característica, o Angular é baseado em programação declarativa, ou seja, é a programação que "descreve o que são suas estruturas" e não "como elas são utilizadas".

Existem diversos *frameworks* focados em dar dinâmica às páginas como o jQuery, Prototype; padrões de construção como o Bootstrap e Materialize, mas nenhum deles está tendo tanta atenção atualmente como o Angular, pois seus desenvolvedores são profissionais de alto gabarito técnico e aficionados por aquilo que fazem, levando os programadores adotarem esta tecnologia na construção de suas aplicações.

Aprender Angular é difícil? Definitivamente, não! A curva de aprendizado ou melhor, o tempo que se leva para aprender, uma nova tecnologia é pequena para o Angular em relação a outras tecnologias, pois sua sintaxe é clara e direta. Mas nem tudo são "flores", pois, conforme vamos nos aprofundando, as coisas ficam mais complicadas, mas não teria graça se não fosse assim.

3. Sintaxe e Utilização

Para utilizar os recursos principais do Angular basta incluir a biblioteca referenciando-a no "*head*" do arquivo HTML da seguinte forma:

```
1. <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angul
```

O Angular trabalha com o conceito de diretivas, que são instruções ou orientações que o interpretador JavaScript deve executar quando as localiza dentro de um arquivo HTML, através da especificação de tags personalizadas que podem ser reutilizadas. As diretivas do Angular são iniciadas com as letras "ng", seguidas da sua funcionalidade. A seguir estão listadas as principais diretivas do Angular (W3SCHOOLS, 2017):

- ng-app - define que se trata de uma aplicação Angular, configurando um elemento HTML como *root* ou raiz da aplicação, ou seja, onde a aplicação será iniciada dentro do arquivo, alterando o comportamento da *tag* que foi utilizada;
- ng-bind - vincula os dados de uma aplicação como uma visualização do HTML, alterando automaticamente o texto de um HTML com os dados originados de um resultado obtido.
- ng-model - é a diretiva que vincula um valor de um controle do HTML como (campos de entrada, caixas de seleção, entre outros) ao dados da aplicação, sendo similar ao ng-bind, entretanto, permite que as mensagens sejam bidirecional (*two-way*);

- ng-class – essa diretiva permite que seja definida dinamicamente classes CSS em um elemento HTML por ligação de dados a uma expressão que representa todas as classes a serem adicionadas.
- ng-click – permite instanciar o evento de click (onclick);
- ng-controller – atribui uma classe de controlador (*controller*) para uma visualização (*view*). Este é um aspecto fundamental de como angular suporta os princípios por trás do padrão de projeto *Model-View-Controller*.
- ng-repeat – instancia um item de uma coleção onde cada instância, onde a variável de loop (repetição) informada é definida para o item de coleção atual e $ index é definido como o índice do item ou da chave, similar ao laço de repetição (*for*);
- ng-show e ng-hide – mostra ou exibe uma *tag* HTML com base no resultado de uma expressão booleana (*true* ou *false*);
- ng-switch – usada para trocar condicionalmente a estrutura DOM em seu modelo com base em uma expressão de escopo.;
- ng-view – diretiva que complementa o serviço *$route* incluindo o modelo renderizado da rota atual no arquivo de layout principal (index.html). Toda vez que a rota atual mudar, a visualização incluída muda com ela de acordo com a configuração do *$route service.template*;
- ng-if – permite criar um bloco lógico (*if*), removendo ou recriando uma parte da árvore DOM, com base em uma expressão lógica.

Sabendo agora sobre algumas diretivas, podemos avançar e iniciar a construção de exemplos. No código a seguir vamos ver um simples exemplo para podermos compreender a sintaxe e como as coisas funcionam com o Angular. Neste exemplo veremos como utilizar expressões, ou seja, realizar algumas operações matemáticas básicas.

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
8.     </head>
9.     <body>
10.         <div ng-app="">
11.             <h1>Expressões com Angular JS</h1>
12.             <h3>Adição</h3>
13.             <p>10 + 35 = {{10 + 35}}</p>
14.
15.             <h3>Subtração</h3>
16.             <p>38 - 12 = {{38 - 12}}</p>
17.
18.             <h3>Multiplicação</h3>
19.             <p>7 * 9 = {{7 * 9}}</p>
20.
21.             <h3>Divisão</h3>
22.             <p>34 / 2 = {{34 / 2}}</p>
23.             <p>121 / 43 = {{121 / 43}}</p>
24.         </div>
25.     </body>
26. </html>
```

Na linha 7 é feita a inclusão da biblioteca do Angular para disponibilizar suas funções dentro da página

```
1. <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angul
```

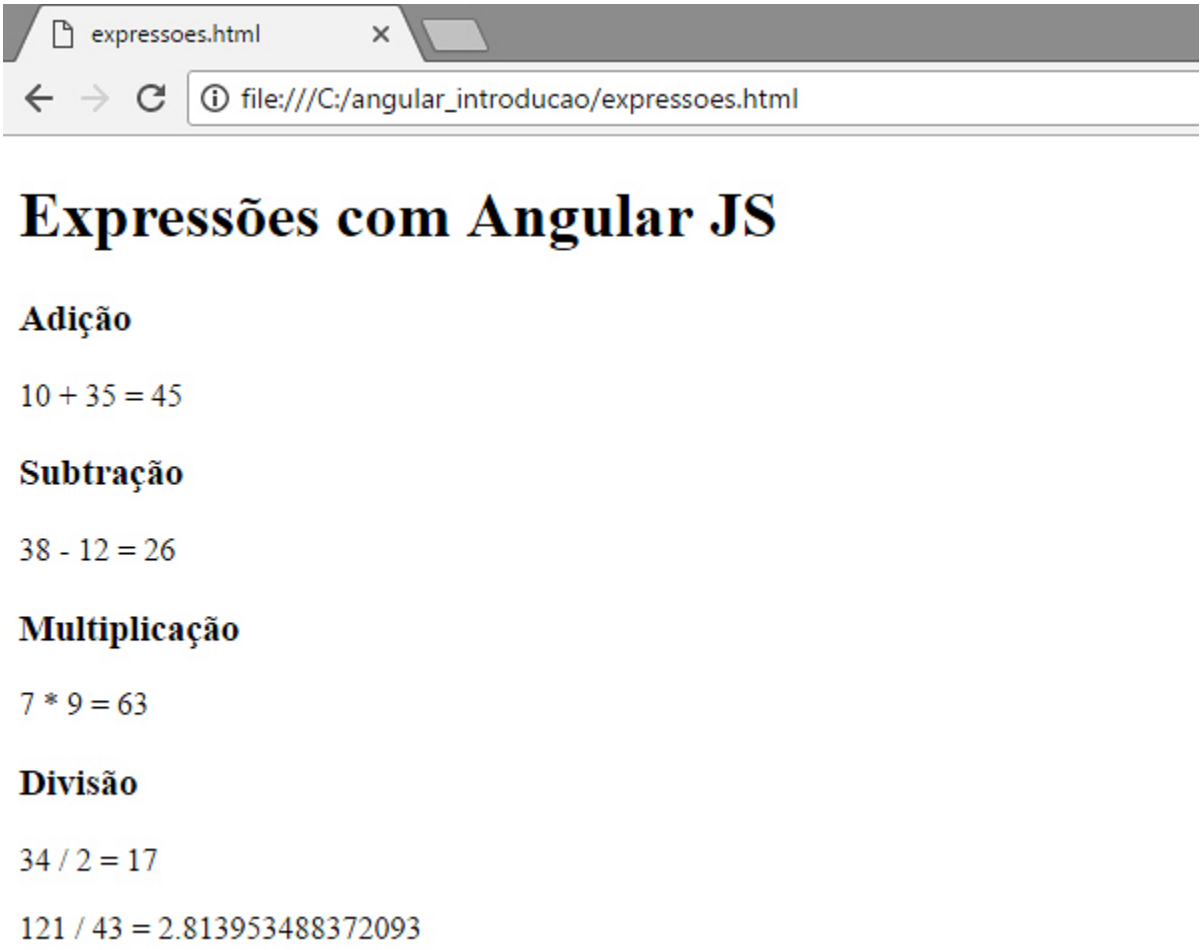
Na linha 10, dentro da tag *div* é incluída a diretiva ng-app="", indicando que a partir dali será iniciada a aplicação Angular. Esse bloco da aplicação é encerrado na linha 24 com o fechamento da *div*, conforme o bloco de código a seguir:

```
1. <div ng-app="">
2.     <h1>Operações Matemáticas</h1>
3.     <h3>Adição</h3>
4.     <p>10 + 35 = {{10 + 35}}</p>
5.
6.     <h3>Subtração</h3>
7.     <p>38 - 12 = {{38 - 12}}</p>
8.
9.     <h3>Multiplicação</h3>
10.    <p>7 * 9 = {{7 * 9}}</p>
11.
12.    <h3>Divisão</h3>
13.    <p>34 / 2 = {{34 / 2}}</p>
14.    <p>121 / 43 = {{121 / 43}}</p>
15. </div>
```

Na linha 13, do código completo, aparece a primeira expressão que é demarcada pela sequência de duas chaves “{{“ de abertura e mais duas chaves de fechamento “}}”, ou seja, toda expressão de deverá estar em chaves duplas.

Nas linhas 16, 19, 22 e 23 estão as demais expressões que realizam outros cálculos.

Na imagem a seguir é mostrado o resultado da execução do código.



Perceba que os valores foram calculados e os resultados apresentados. Entretanto, o último cálculo, que é uma divisão, apresentou um resultado com diversas casas decimais. Existe uma forma de limitar o número de casas a serem apresentadas, bastando utilizar um argumento dentro da expressão da seguinte forma:

```
1. <p>121 / 43 = {{121 / 43 | number:2}}</p>
```

Perceba que foi incluído na expressão um argumento "| number:2", ou seja, o símbolo barra "|" indica que haverá um argumento para esta expressão, seguido do nome do argumento "number:2", significando que o número deverá conter apenas duas casas decimais. No código a seguir são mostradas quatro variações de casas decimais para o mesmo cálculo.

```
1. <div ng-app="">
2.     <h1>Expressões com Argumentos</h1>
3.     <p>121 / 43 = {{121 / 43| number:2}}</p>
4.     <p>121 / 43 = {{121 / 43| number:3}}</p>
5.     <p>121 / 43 = {{121 / 43| number:4}}</p>
6.     <p>121 / 43 = {{121 / 43| number:5}}</p>
7. </div>
```

A imagem a seguir mostra o resultado após a renderização no navegador:

Na imagem anterior é possível visualizar os diferentes resultados, apresentando o número de casas decimais para cada operação realizada.

Você viu como é simples criar expressões em Angular. Agora vamos comparar uma operação de adição realizada com o Angular e com jQuery. Os códigos estão completos para que a comparação possa ser feita de forma completa.

Angular JS

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
    </head>
    <body>
        <div ng-app="">
            <h1>Expressões com Argumentos</h1>
            <p>30 + 20 = {{30 + 20}}</p>
        </div>
    </body>
</html>
```

jQuery

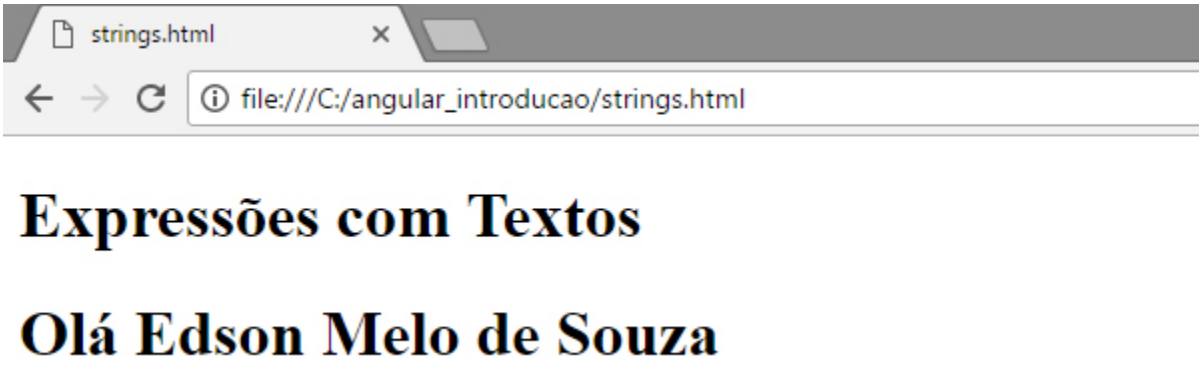
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/

        <script type="text/javascript">
            $(document).ready(function () {
                var total = (30 + 20);
                $("#total").text(total.toFixed(2));
            });
        </script>
    </head>
    <body>
        <div>
            <h1>Expressões com Argumentos</h1>
            <p id="total"></p>
        </div>
    </body>
</html>
```

Comparando os códigos, é possível perceber como o Angular reduz muito a codificação, deixando o código mais limpo e inteligível, tornando o desenvolvimento muito mais produtivo.

As expressões em Angular também podem ser utilizadas para concatenar textos (*strings*), utilizando a mesma simplicidade. Vejamos um exemplo:

```
<h1>Olá {{'Edson' + ' Melo ' + ' de ' + 'Souza'}}</h1>
```

Perceba que foi utilizado o sinal de adição para concatenar as palavras. Na imagem a seguir é mostrado o resultado após a renderização no navegador.



4. Valores *não Literais*

Nos exemplos anteriores trabalhos com valores literais, ou seja, valores fixos. Agora vamos avançar um pouco e utilizar declaração de valores e recuperação, ou seja, vamos trabalhar com variáveis inicializadas. Para isso devemos utilizar a diretiva ng-init, que tem a função de declarar e inicializar as declarações. Vejamos um exemplo:

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4

8.     </head>
9.     <body>
10.        <div ng-app="" ng-init="valor1 = 30; valor2 = 20">
11.            <h1>Variáveis (ng-init)</h1>
12.            <h3>O resultado de (30 + 20) é: {{valor1 + valor2}}</h3>
13.        </div>
14.    </body>
15. </html>
```

Na linha 10 é inicializada a aplicação “ng-app” e, em seguida, é utilizada a diretiva “ng-init”. A diretiva declara os parâmetros valor1 e valor2, atribuindo os respectivos valores 30 e 20. Na linha 12 é mostrado o resultado, utilizando-se apenas o recurso das expressões que é a inclusão das variáveis dentro das chaves duplas.

A imagem a seguir mostra o resultado após a renderização no navegador:

Uma outra forma de realizar a mesma operação, mas com a vantagem de colocar o resultado em um elemento HTML, podendo realizar formatações e utilizando o ng-bind. Vamos ver seu funcionamento no código a seguir.

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
8.         <style type="text/css">
9.             .resultado{
10.                font-size: 20px;
11.                color: blue;
12.                margin-left: 50px;
13.            }
14.        </style>
15.    </head>
16.    <body>
17.        <div ng-app="" ng-init="valor1 = 30; valor2 = 20">
18.            <h1>Variáveis (ng-init com ng-bind)</h1>
19.            <h3>O resultado de (30 + 20) é: <span class="resultado" ng-b
20.        </div>
21.    </body>
22. </html>
```

Na linha 19 é utilizada a diretiva ng-bind para apresentar os valores que estão armazenados nas variáveis criadas com o ng-init (linha 17). Neste exemplo as variáveis estão sendo colocadas dentro de uma tag *span* e, para isso, foi criada um CSS para fazer uma formatação. Perceba que tag *span* há uma classe CSS, conforme mostra o código a seguir:

```
1. <style type="text/css">
2.     .resultado{
3.         font-size: 20px;
4.         color: blue;
5.         margin-left: 50px;
6.     }
7. </style>
```

Você pode perceber as possibilidades de personalizar os resultados fazendo a formatação dos valores com CSS.

A seguir vamos ver como trabalhar com objetos no Angular.

5. Objetos em Angular

Objetos são estruturas complexas que suportam dados estruturados, de forma que podemos referenciar diversos valores por meio da utilização de apenas uma palavra "objeto".

Vamos supor que temos um conjunto de informações sobre uma pessoa como nome, email, telefone e endereço. Se fossemos armazenar individualmente cada valor em uma variável ficaria muito confuso. Então, utilizando um objeto, vamos agrupar todos os dados em uma só "variável" (*objeto*) que chamaremos de pessoa. Vamos ver como fazer isso utilizando o Angular no código a seguir:

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
8.         <style type="text/css">
9.             body {
10.                font-size: 20px;
11.                margin-left: 50px;
12.            }
13.            .pessoa{
14.                font-weight: bold;
15.                line-height: 2em;
16.            }
17.        </style>
18.    </head>
19.    <body>
20.        <div ng-app="" ng-init="pessoa={nome: 'Edson Melo de Souza', ema
21.            <h1>Objetos em Angular</h1>
22.            <h3>Dados Pessoais</h3>
23.            Nome: <span class="pessoa" ng-bind="pessoa.nome"></span><br>
24.            Email: <span class="pessoa" ng-bind="pessoa.email"></span><b
25.            Telefone: <span class="pessoa" ng-bind="pessoa.telefone"></s
26.            Endereço: <span class="pessoa" ng-bind="pessoa.endereco"></s
27.        </div>
28.    </body>
29. </html>
```

Na linha 20 é inicializa a aplicação e, na mesma linha, é declarado um objeto com o nome de pessoa. Perceba que está sendo criado um *array* com quatro elementos, onde o primeiro é nome da referência (atributo) e o segundo o valor, ou seja, a declaração de qualquer elemento de um objeto será da forma "chave" : "valor".

Nas linhas de 23 a 26 são mostrados os valores utilizando o ng-bind. Para mostrar o valor armazenado, foi utilizada a expressão "pessoa.atributo". Portanto, utilizar objetos também é uma solução muito interessante, pois fica muito mais fácil realizar a distribuição dentro do código HTML.

Neste exemplo foi criado um CSS apenas para destacar os valores apresentados, conforme segue:

```
1. <style type="text/css">
2.     body {
3.         font-size: 20px;
4.         margin-left: 50px;
5.     }
6.     .pessoa{
7.         font-weight: bold;
8.         line-height: 2em;
9.     }
10. </style>
```

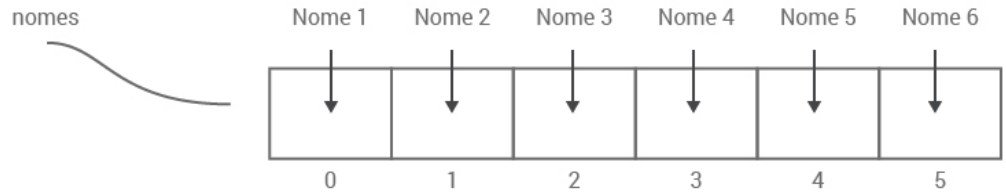
Na imagem a seguir é apresentado o resultado da renderização no navegador.

6. Arrays em Angular

Para finalizar nossa breve introdução ao Angular, vamos ver como trabalhar com *arrays* (vetores ou matrizes), pois esse recurso é amplamente utilizado em programação.

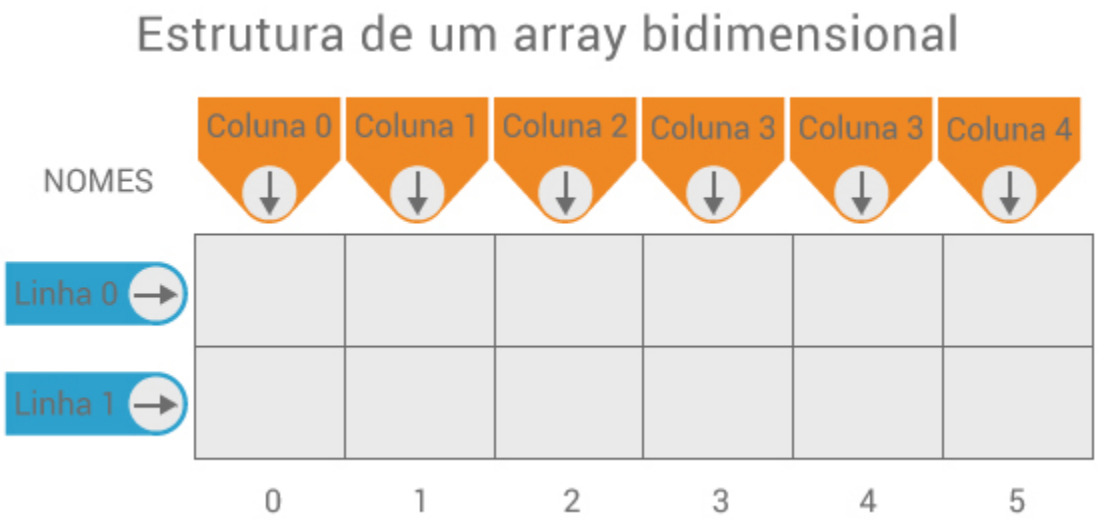
Apenas para relembrar o conceito, os *arrays* podem armazenar diversos tipos de dados como textos, caracteres, inteiros, duplos, entre outros. Os arrays podem ser de uma dimensão (vetor) ou de mais dimensões (matrizes), conforme mostra as figuras a seguir:

Array de uma dimensão (vetor)



Legenda: ESTRUTURA DE UM ARRAY UNIDIMENSIONAL

Array de duas dimensões (matriz)



Legenda: ESTRUTURA DE UM ARRAY BIDIMENSIONAL

Para declarar um array em Angular usamos a seguinte declaração:

```
1. ng-init=?nome_do_array=[valor1, valor2, valor3, ...]?
```

Vamos ver um exemplo onde será declarado um array com o nome “exemplo_array” com dados variados (texto, números inteiros, reais e booleanos (*true* ou *false*).

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
8.     </head>
9.     <body>
10.         <div ng-app="" ng-init="exemplo_array = ['Edson', 46, 54.3, fals
11.             <h1>Arrays em Angular</h1>
12.             <h3>Lendo dados do Array</h3>
13.             exemplo_array[0]: <span ng-bind="exemplo_array[0]"></span><b
14.             exemplo_array[1]: <span ng-bind="exemplo_array[1]"></span><b
15.             exemplo_array[2]: <span ng-bind="exemplo_array[2]"></span><b
16.             exemplo_array[3]: <span ng-bind="exemplo_array[3]"></span>
17.         </div>
18.     </body>
19. </html>
```

Na linha 10 o *array* exemplo_array é declarado com valores e tipos de dados variados. Nas linhas de 13 a 16 os valores são lidos do *array* utilizando a sintaxe “[...]”, ou seja, colchetes. Para ter acesso ao valor armazenado no *array* basta utilizar o nome do *array* seguido da posição desejada, lembrando que a posição sempre começará em zero (0). O acesso aos dados é realizado da mesma forma que no JavaScript.

A imagem a seguir mostra os valores processados no navegador.

Chegamos ao final deste tópico e, após essa breve introdução, você já pode ter uma noção de como o Angular é poderoso. Agora é hora de treinar, criando outros programas para fixar o conteúdo abordado até aqui. Nos links em destaque a seguir você encontrará muita informação para se aprofundar no assunto.

SAIBA MAIS!

Angular JS - https://angularjs.org/ (https://angularjs.org/)

Criando uma aplicação simples com AngularJS - https://tableless.com.br/criando-uma-aplicacao-simples-com-angularjs/ (https://tableless.com.br/criando-uma-aplicacao-simples-com-angularjs/)

Tutorials Point - https://www.tutorialspoint.com/angularjs/ (https://www.tutorialspoint.com/angularjs/)

Resumo

Neste tópico foi apresentada uma breve introdução ao *framework* Angular JS, mostrando suas características, além de exemplos e descrição sobre a utilização das instruções para criação de aplicações.

Conclusão

O Angular JS veio para mudar a forma de como as aplicações para a internet serão desenvolvidas daqui para frente. Você pode perceber como ele é poderoso e, como o próprio site oficial diz ele é “AngularJS — Superheroic JavaScript MVW Framework”.

Portanto, agora é hora acessar os links acima, em destaque, e aprofundar seus estudos para se preparar para o mercado com essa ferramenta moderna e tão poderosa.

ATIVIDADE FINAL

Qual é a função da diretiva ng-app?

A. Define que se trata de uma aplicação Angular, configurando um elemento HTML como *root* ou raiz da aplicação, ou seja, onde a aplicação será iniciada dentro do arquivo, alterando o comportamento da *tag* que foi utilizada.
B. É a diretiva que vincula um valor de um controle do HTML como (campos de entrada, caixas de seleção, entre outros) ao dados da aplicação, sendo similar ao ng-bind, entretanto, permite que as mensagens sejam bidirecional (*two-way*)
C. Essa diretiva permite que seja definida dinamicamente classes CSS em um elemento HTML por ligação de dados a uma expressão que representa todas as classes a serem adicionadas.
D. Instancia um item de uma coleção onde cada instância, onde a variável de loop (repetição) informada é definida para o item de coleção atual e $ index é definido como o índice do item ou da chave, similar ao laço de repetição (*for*);

Qual o significado da instrução {{121 / 43}}?

A. Exibe o resultado da divisão entre os dois números.
B. Atribui dois valores não literais para realizar uma operação.
C. Define um argumento para o Angular.
D. Configura para duas casas decimais os valores.

Para realizar a concatenação d textos (Strings) em Angular, é necessário