Angular JS Tutorial

Published On: July 30, 2024

Angular JS Tutorial

For building strong and capable MVC-based web applications, developers all around the world rely on the well-liked JavaScript framework, AngularJS. Here is the comprehensive Angular JS tutorial that helps beginners get started with their learning journey.

Download Angular JS Tutorial PDF

Introduction to Angular JS

The JavaScript Framework AngularJS is quite strong. Projects using SPAs (Single Page Applications) use it. It increases the DOM's responsiveness to user input and adds new properties to it. In this Angular JS tutorial, we cover the following:

- Overview of Angular JS
- Angular JS Fundamental Elements
- HTML-DOM Elements
- MVC Architecture for Angular JS
- Dependency Injection
- First Application

Angular JS Tutorial Interview Questions

Overview of Angular JS

An open-source framework for web applications is called AngularJS. Adam Abrons and Misko Hevery produced it for the first time in 2009. Google is now in charge of its

upkeep. It is currently running on 1.2.21.

Features of Angular JS

- Potent framework to create rich internet applications.
- Open-source with Apcahe Lincense.
- Cross-browser compliant to make the JavaScript code suitable for web browsers automatically.
- Specialized features with data binding, scope, cotroller, services, filters, directives, templates, routing, and so on.
- Modern MVC-supported framework to write client side apps with JavaScript.
- MVVW (Model-View-Whatever) for modernized MVC design pattern.
- Deep linking to encode the application's state.
- Dependency injection to simplify the SDLC.

Advantages of Angular JS

The following are AngularJS's advantages:

- Separation of concerns and dependency injection are two features of AngularJS.
- Unit tests can be run on AngularJS code.
- It makes it possible to construct SPAs in a very organized and manageable manner.
- Reusable components are offered using AngularJS.
- It gives HTML the ability to bind data. As a result, the user has a responsive and rich experience.
- JavaScript controllers conduct business processing in AngularJS, whereas views are just HTML pages.
- Developers may accomplish greater functionality with shorter code when they use AngularJS.

Angular JS Developer Salary

Angular JS Fundamental Elements

All popular browsers and smartphones, including those with Android and iOS operating systems, can execute AngularJS applications. A few of the more well-liked components are described below.

Directives

The AngularJS framework is comprised of the following

ASP DOTNET Tutorial
Published On: July 31, 2024
ASP DOTNET Tutorial Microsoft created the web framework known as ASP.NET. It is employed in...

Artificial Intelligence Tutorial
Published On: July 30, 2024
Artificial Intelligence Tutorial Artificial intelligence (AI) is significant since it enhances many facets of society...

Appium Testing Tutorial
Published On: July 30, 2024
Appium Testing Tutorial Designed to make the UI automation of many app platforms easier, Appium...

primary components:

- **ng-app**: An AngularJS application is defined and linked to HTML via this directive.
- **ng-init**: Application data is initialized using this directive.
- **ng-model:** This directive ties HTML input controls to the values of the application data for AngularJS.
- **ng-repeat:** This directive makes every element in a collection repeatable in HTML.
- **ng-bind:** The AngularJS application data is bound to HTML tags by means of this directive.

ng-app:

An AngularJS application is started with the ng-app directive. As soon as the page containing the AngularJS application loads, it automatically bootstraps or initializes the application.

Example

* *

…

* *

ng-init:

An AngularJS application’s data is initialized using the ng-init directive. It’s employed to provide the variables values.

An array of countries is initialized in the example that follows. The array of nations is defined using JSON syntax.

Example

<div ng-app = “” ng-init = “countries = [{locale:’en-US’,name:’United States’},

{locale:’en-GB’,name:’United Kingdom’}, {locale:’en-FR’,name:’France’}]”>

…

* *

ng-model

The ng-model directive defines the model or variable that will be used in an AngularJS application.

Example

```
<div ng-app = “”>
  ...
  <p>Enter your Name: <input type = “text” ng-model = “name”></p>
</div>
```

ng-repeat

Every item in a collection has its HTML elements repeated using the ng-repeat directive.

Example

```
<div ng-app = “”>
  ...
  <p>List of Countries with locale:</p>
  <ol>
    <li ng-repeat = “country in countries”>
      {{ ‘Country: ‘ + country.name + ‘, Locale: ‘ + country.locale }}
    </li>
  </ol>
</div>
```

ng-bind

The text content is to be replaced with a template, as per the AngularJS ng-bind-template directive. It substitutes the value of the specified expressions for the content of an HTML element.

Example

```
<div ng-app=”myApp” ng-bind-template=”{{firstName}} {{lastName}}” ng-controller=”myCtrl”>
</div>
```


Controllers

AngularJS controllers regulate the flow of data in an AngularJS application.

- The ng-controller directive is used to define a controller.
- A JavaScript object with functions and attributes is called a controller.

Each controller takes a $scope parameter, which indicates which application or module it is to control.

> **Example**
>
> *<!DOCTYPE html>*
>
> *<html>*
>
> *<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>*
>
> *<body>*
>
> * *
>
> *First Name: <input type="text" ng-model="firstName">
*
>
> *Last Name: <input type="text" ng-model="lastName">
*
>
> *
*
>
> *Full Name: {{firstName + " " + lastName}}*
>
> * *
>
> *<script>*
>
> *var app = angular.module('myApp', []);*
>
> *app.controller('myCtrl', function($scope) {*
>
> *$scope.firstName = "Aryan";*
>
> *$scope.lastName = "Khanna";*
>
> *});*
>
> *</script>*
>
> *</body>*
>
> *</html>*

Expressions

HTML is bound to application data using expressions. Double curly braces are used to write expressions, like in {{expression}}.

Expressions function similarly to directives from ng-bind. Expressions in AngularJS are made entirely of JavaScript and output the data in the locations where they are utilized.

Data Type	**Expression**
Numbers	Expense on Books : {{cost * quantity}} Rs
Strings	Hello {{student.firstname + " " + student.lastname}}!
Object	Roll No: {{student.rollno}}
Array	Marks(Math): {{marks[3]}}

Example

```
<html>
  <head>
    <title>AngularJS Expressions</title>
  </head>
  <body>
    <h1>Sample Application</h1>
    <div ng-app = "" ng-init = "quantity = 1;cost = 30;
      student = {firstname:'Mahesh',lastname:'Parashar',rollno:101};
      marks = [80,90,75,73,60]">
      <p>Hello {{student.firstname + " " + student.lastname}}!</p>
      <p>Expense on Books : {{cost * quantity}} Rs</p>
      <p>Roll No: {{student.rollno}}</p>
      <p>Marks(Math): {{marks[3]}}</p>
    </div>
        <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>
  </body>
```

> *</html>*

Filters

Modifying the data is done with filters. The pipe (|) character can be used to group them in expressions or directives. The frequently used filters are displayed in the list below.

Uppercase Filter

Use the pipe character to add an uppercase filter to an expression. In order to publish the student's name in all capital letters, we have introduced an uppercase filter.

Example

Enter first name:<input type = "text" ng-model = "student.firstName">

Enter last name: <input type = "text" ng-model = "student.lastName">

Name in Upper Case: {{student.fullName() | uppercase}}

Lowercase Filter

Use the pipe character to add a lowercase filter to an expression. To print the student's name in all lowercase letters, we have introduced a lowercase filter.

Enter first name:<input type = "text" ng-model = "student.firstName">

Enter last name: <input type = "text" ng-model = "student.lastName">

Name in Lower Case: {{student.fullName() | lowercase}}

Currency Filter

Use a pipe character to add a currency filter to an expression that returns a number. To print fees in currency format, we have included a currency filter here.

> *Enter fees: <input type = "text" ng-model = "student.fees">*
>
> *fees: {{student.fees | currency}}*
>
> Filter
>
> We utilize subjectName as a filter to show just subjects that are required.

```
Enter subject: <input type = “text” ng-model = “subjectName”>

Subject:

<ul>

  <li ng-repeat = “subject in student.subjects | filter: subjectName”>

    {{ subject.name + ‘, marks:’ + subject.marks }}

  </li>

</ul>
```

OrderBy Filter

We utilize orderBy marks to arrange subjects according to marks.

Example

```
Subject:

<ul>

  <li ng-repeat = “subject in student.subjects | orderBy:’marks'”>

    {{ subject.name + ‘, marks:’ + subject.marks }}

  </li>

</ul>
```

Tables

In general, table data can be repeated. Drawing tables is simple when you use the ng-repeat directive.

Example

```
<table>

  <tr>

    <th>Name</th>

    <th>Marks</th>

  </tr>

  <tr ng-repeat = “subject in student.subjects”>

    <td>{{ subject.name }}</td>

    <td>{{ subject.marks }}</td>

  </tr>

</table>
```

HTML – DOM

To link application data to the properties of HTML DOM elements, use the following directives:

ng-disabled

Give an HTML button a model and the ng-disabled attribute. To observe the variation, bind the model to a checkbox.

Example

<input type = "checkbox" ng-model = "enableDisableButton">Disable Button

<button ng-disabled = "enableDisableButton">Click Me! </button>

ng-show

Give a button in HTML the ng-show property and pass it a model. To observe the variation, bind the model to a checkbox.

Example

<input type = "checkbox" ng-model = "showHide1">Show Button

<button ng-show = "showHide1">Click Me!</button>

ng-hide

Give a button in HTML the ng-hide property and pass it a model. To observe the variation, bind the model to a checkbox.

Example

<input type = "checkbox" ng-model = "showHide2">Hide Button

<button ng-hide = "showHide2">Click Me!</button>

ng-click

Update a model and give an HTML button the ng-click attribute. Bind the model to HTML to observe the difference.

Example

* Total click: {{ clickCounter }} *

<button ng-click = "clickCounter = clickCounter + 1">Click Me!</button>

MVC Architecture for Angular JS

The software design pattern known as Model View Controller, or MVC, as it is more commonly known, is used to create web applications.

Because it allows for the separation of concerns and isolates application functionality from the user interface layer, MVC is widely used.

After receiving all application requests, the controller collaborates with the model to prepare any data required by the view.

The view then creates a final, presentable response using the controller's prepared data.

The following three components make up a Model View Controller pattern:

- **Model:** The pattern's lowest level is in charge of data maintenance.
- **View:** It's in charge of showing the user all or some of the information.
- **Controller:** A piece of software code that governs how the model and view interact with each other.

Dependency Injection

In a dependency injection software design, dependencies are assigned to components rather than being hardcoded into the component itself.

It releases a component from the burden of locating dependencies and allows for dependency configuration. Additionally, it aids in the reusability, maintainability, and testing of components.

Angular offers an excellent dependency injection method with the following essential elements, which can be inserted as dependencies into one another.

Value

During the config phase, when AngularJS bootstraps itself, Value, a basic JavaScript object, is needed to pass values to the controller.

> **Example**
>
> ```
> var mainApp = angular.module("mainApp", []);
>
> mainApp.value("defaultInput", 5);
>
> mainApp.controller('CalcController', function($scope, CalcService,
> defaultInput) {
>
>   $scope.number = defaultInput;
>
>   $scope.result = CalcService.square($scope.number);
>
>   $scope.square = function() {
>
>     $scope.result = CalcService.square($scope.number);
>
>   }
>
> });
> ```

Factory

A function called factory is used to return values. On demand, it generates value whenever a controller or a service calls for it. Usually, a factory function is used to compute and return the value.

Example

```
var mainApp = angular.module("mainApp", []);

mainApp.factory('MathService', function() {

  var factory = {};

  factory.multiply = function(a, b) {

    return a * b

  }

  return factory;

});

mainApp.service('CalcService', function(MathService) {
```

```
  this.square = function(a) {
    return MathService.multiply(a,a);
  }
});
```

Service

A JavaScript object that is unique and has a set of functions to carry out specific duties is called a service. The service() function is used to define the service, which is subsequently injected into the controllers.

Example

```
var mainApp = angular.module("mainApp", []);
mainApp.service('CalcService', function(MathService) {
  this.square = function(a) {
    return MathService.multiply(a,a);
  }
});
mainApp.controller('CalcController', function($scope, CalcService,
defaultInput) {
  $scope.number = defaultInput;
  $scope.result = CalcService.square($scope.number);
  $scope.square = function() {
    $scope.result = CalcService.square($scope.number);
  }
});
```

Provider

AngularJS internally uses providers to establish factories, services, and other things at the configuration stage.

To construct the MathService that we previously established, run the script below. To return the value, service, or factory, the provider is a unique factory method that uses the get() method.

Example

```
var mainApp = angular.module("mainApp", []);
```

```
mainApp.config(function($provide) {
  $provide.provider('MathService', function() {
    this.$get = function() {
      var factory = {};
      factory.multiply = function(a, b) {
        return a * b;
      }
      return factory;
    };
  });
});
```

Constant

Given that values cannot be utilized during the configuration step, constants are used to pass values at that stage.

Syntax: *mainApp.constant("configParam", "constant value");*

First Application

We are offering a test app sample. Our app was developed using AngularJS, HTML, and CSS.

Sample Code

```
<html>
  <head>
    <title>AngularJS First Application</title>
  </head>
  <body>
    <h1>Sample Application</h1>
    <div ng-app = "">
      <p>Enter your Name: <input type = "text" ng-model = "name"></p>
      <p>Hello <span ng-bind = "name"></span>!</p>
    </div>
    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
```

```
    </script>
  </body>
</html>
```

Output

Conclusion

This Angular.JS tutorial will be helpful to you in understanding the fundamentals of using elements of Angular.JS to build single page apps. Explore deeper with hands-on exposure in our **Angular.JS training in Chennai.**

Share on your Social Media

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Disclaimer

Privacy Policy

Terms and Conditions

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber