Introduction to Angular | What is Angular?

1 Comment / 6 minutes of reading / March 10, 2023

← Angular Tutorial

Angular Tutorial

Architecture overview and concepts →

Angular is the new front-end framework and is the successor to the most popular AngularJs. Angular is open-source and helps us build dynamic & single-page applications (SPAs). In this article, we will give you a brief introduction to Angular.

Table of Contents

What is Angular?

Angular is a development platform and framework for building single-page client applications using HTML and TypeScript. Angular is built using TypeScript.

Angular has many improvements and innovations over its predecessor, AngularJS. It is easy to learn and develop enterprise-scale applications. You can build extendable, maintainable, testable, and standardized applications using Angular.

Angular provides all the core functionality that you need to develop a client application. It comes bundled with a router module, an HTTP module, a forms API, etc. We can use it to build amazing client-side applications.

Angular is built with testing in mind. When you install Angular, it will also install the Jasmine testing framework. Hence, you can get started immediately.

Angular is an opinionated framework. It is a framework that follows a certain design pattern and forces you to follow it. This has its own benefits. Every Angular application will follow the same structure. As a result, if you switch to another project in another company as a developer, the Angular application will look similar.

Introduction to Angular

We have created a great tutorial on How to Install and get started with Angular. And find out how to Create your First Angular Application.

Features of Angular

Angular is loaded with Power-packed Features. Some of the features are listed below

Modular by design

Angular applications follow modular design concepts. The application comprises Angular modules, which are self-contained units of code. We can easily reuse them across the different parts of the application and better organize and manage our codebase.

Component-based structure

Angular components are the backbone of the Angular application. A component represents a piece of the UI. We compose complex UI elements by reusing the existing components.

Templates

Templates represent HTML. Angular enables us to use programming constructs like if conditions, loops, etc., to render and control how the HTML pages look. The component powers the templates.

Two-way data binding

Data binding is the most incredible feature of Angular. It is automatic and fast. Changes in the view or component reflect on the component class and vice versa instantly.

Dependency injection

Angular comes bundled with a dependency injection system, which promotes code reusability. It makes it easier to manage and inject dependencies. We can create Angular services and inject them into component classes.

Routing

Routing is another essential feature of Angular. It makes it easier to navigate between different views without reloading the page. The routing module helps us build single-page applications in Angular.

Angular Forms

The Angular Forms API provides rich tools for building complex forms. It includes built-in validation and support for reactive forms.

HTTP Module

The HTTP Module allows sending HTTP requests to the back-end server. It is very robust and can cover advanced client-server communication.

Testing

Angular has a built-in testing framework that makes writing and running tests for your application easy.

Active Community

Angular is supported by Google and has a very active community of supporters. It makes a lot of difference, as your queries are quickly resolved.

-

Key differences between AngularJs & Angular

Support for ES6

Angular is completely written in Typescript and meets the ECMAScript 6 specification. This means that it has support for ES6 Modules, Class frameworks, etc.

Components are new controllers

In AngularJS we had Controllers. AngularJs Controllers are replaced with Angular Components. The controllers and view in AngularJS are defined as follows

```

//View
<body ng-controller=’appController’>
    <h1>vm.message<h1>
</body>

//Controller angular.module(‘app’).controller(‘appController’,appcontroller) {
    message=’Hello Angular’;
}

```

In Angular, we are using Components. The simple component is shown below written using Typescript.

```

import { Component } from '@angular/core';

@Component({
    selector: 'app',
    template: '<h1>{{message}} </h1>'
})
export class AppComponent  {
      message: string=’Hello Angular’;
}

```

In Angular, a component represents a UI element. You can have many such components on a single web page. Each component is independent of the other and manages a section of the page. The components can have child components and parent components.

Directives

AngularJS had a lot of directives. Some of the most used directives are ng-repeat & ng-if

```

<ul>
    <li ng-repeat =customer in vm.customers>
        {{customer.name}}
    </li>
</ul>
<div ng-if=”vm.isVIP”>     <h3> VIP Customer </h3> </div>

```

Angular also has directives, but with a different syntax. It has a * before the directive name indicating it is a structural directive

```

<ul>
    <li *ngFor =#customer of customers>
        {{customer.name}}
    </li>
</ul>
<div *ngIf=”vm.isVIP”>
    <h3> VIP Customer </h3>
</div>

```

The style directives like ng-style, ng-src & ng-href are all gone. These are now replaced by property binding of HTML elements to the class properties

The creation of Custom Directives is vastly simplified in angular as shown in the example below.

```

@Directive({
  selector: '[MyDirective]'
})
class MyDirective { }

```

Data Bindings

The powerful angular data bindings stay the same, with minor syntax changes.

Interpolation

```

//AngularJS
<h3> {{vm.customer.Name}}</h3>

//Angular
<h3> {{customer.Name}}</h3>

```

Note that we used the controller alias VM to specify the controller instance in AngularJS. In Angular, the context is implied.

One way Binding

```

//AngularJS
<h3> ng-bind=vm.customer.name></h3>

//Angular
<h3 [innerText]="customer.name" ></h3>

```

Event Binding

```

//AngularJS
<button ng-click=”vm.save()”>Save<button>

//Angular
<button (click)=”save()”>Save<button>

```

The AngularJS uses the ngClick directive to bind to the event. In Angular ngClick Directive is removed. You can bind directly to the DOM events

Two-way binding

```

//AngularJS
<input ng-model=”vm.customer.name”>

//Angular
<input [(ng-model)]=”customer.name”>

```

$scopes are out

Angular is not using $scope anymore to glue the view and controller.

AngularJS used to run a dirty check on the scope objects to see if any changes occurred. Then it triggers the watchers. And then it used to re-running the dirty checking again.

Angular uses zone.js to detect changes. Zone.js apply patches on all the global asynchronous operations like click event, timer events, HTTP requests, etc. It then intimates the Angular, whenever the changes occur in the Angular Application. The Angular then runs the change detection for the entire application

Filters are renamed to Pipes

In AngularJS, we used Filters and as shown below

```

<td>{{vn.customer.name | uppercase}}</td>

```

Angular uses the same syntax but names them pipes

```

<td>{{customer.name | uppercase}}</td>

```

Platform-specific Bootstrap

In AngularJS we used the ng-app directive in our HTML, then the Angular would bootstrap and attach itself the ng-app

```

<body ng-app=’app’> </html>

```

Bootstrapping in Angular is done through code. The bootstrapping of Angular is not simple as that of AngularJS. The sample code below shows how the Angular application bootstraps the AppModule using platformBrowserDynamic Module

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import {AppModule } from './app.module'; platformBrowserDynamic().bootstrapModule(App

```

Bootstrap is also Platform-specific in Angular. You can have different bootstrappers for Mobile & Web applications.

You can read about Bootstrapping Angular Application

Services

AngularJS had Services, Factories, Providers, Constants, and values, which were used to create reusable code. These are then injected into Controllers so that it can use it

The Angular all the above is merged into a Service class.

Mobile Support

AngularJS was not built with mobile support in mind. Angular is designed with mobile development in mind.

Summary

This article gives a brief Introduction of Angular to you. In the next tutorial, we will learn core concepts of Angular. Subsequently we will show you how to Install Angular and Create your first Angular Application.

Read More