

Angular

Introducci n a Angular

CertiDevs

 ndice de contenidos

1. �Qu� es Angular?	1
2. Historia de Angular	1
3. �Para qu� sirve Angular?	1
4. �Por qu� aprender Angular?	1
5. Popularidad de Angular	2
6. Versiones de Angular	2
7. Arquitectura de Angular	3
7.1. Componentes	3
7.2. M�dulos	4
7.3. Relaci�n entre componentes y m�dulos	4
7.4. Creaci�n de componentes y m�dulos	4
7.5. Servicios	4
7.6. Uso de servicios en componentes y m�dulos	4
7.7. Inyecci�n de dependencias (DI)	5
7.8. Directivas	5
7.9. Enrutamiento y navegaci�n	5
7.10. Formularios	5
7.11. Pipes	5
7.12. HTTP y comunicaci�n con API	5
7.13. Detecci�n de cambios	6
7.14. Pruebas unitarias y de integraci�n	6
8. Recursos de aprendizaje	6
9. Instalaci�n de Angular	6
9.1. Instalaci�n global	6
9.2. Instalaci�n local	7
9.3. Creaci�n de un proyecto	7
10. Ejecuci�n de un proyecto	7
11. Creaci�n de un componente	7

1. ¿Qué es Angular?

Angular es un *framework* de JavaScript desarrollado por Google para construir aplicaciones web de una sola página (SPA) escalables y dinámicas.

Con Angular, los desarrolladores pueden crear aplicaciones web utilizando un enfoque basado en componentes y una arquitectura modular.

Angular es la evolución de AngularJS, la primera versión del framework lanzada en 2010.

El repositorio oficial de Angular se aloja en GitHub y se puede encontrar en:

<https://github.com/angular/angular>

2. Historia de Angular

AngularJS, también conocido como Angular 1.x, fue lanzado en octubre de 2010 por Google. AngularJS popularizó el concepto de desarrollo basado en directivas y la inyección de dependencias en aplicaciones web. Sin embargo, AngularJS tenía ciertas limitaciones y problemas de rendimiento en aplicaciones de gran escala.

Para abordar estos problemas, el equipo de AngularJS desarrolló una nueva versión del framework, conocida como Angular, que se lanzó en *septiembre de 2016*. Esta nueva versión de Angular es a menudo llamada Angular 2+, ya que abarca Angular 2 y todas las versiones posteriores (Angular 3 fue omitido para evitar confusiones).

Angular introdujo mejoras significativas en el rendimiento, la escalabilidad y la facilidad de uso en comparación con AngularJS.

3. ¿Para qué sirve Angular?

Angular es ideal para desarrollar aplicaciones web de una sola página y aplicaciones empresariales de gran escala. Algunos de los principales usos de Angular incluyen:

- ¥ Aplicaciones web de una sola página (SPA): Angular es especialmente adecuado para crear SPA, donde la navegación y la interacción ocurren en una única página sin recargarla completamente.
- ¥ Aplicaciones empresariales: Angular ofrece características y herramientas que facilitan la construcción de aplicaciones empresariales robustas y escalables.
- ¥ Aplicaciones móviles y de escritorio: Con frameworks como Ionic o NativeScript, Angular se puede utilizar para crear aplicaciones móviles híbridas o nativas, así como aplicaciones de escritorio.

4. ¿Por qué aprender Angular?

Aprender Angular tiene varias ventajas, incluyendo:

- ¥ Popularidad y adopci3n: Angular es uno de los frameworks m3s populares y ampliamente adoptados en la industria del desarrollo web. Aprender Angular aumenta las posibilidades de empleo y permite trabajar en proyectos de gran envergadura.
- ¥ Soporte de Google: Angular es respaldado por Google, lo que garantiza un soporte s3lido, una base de c3digo activa y una comunidad en constante crecimiento.
- ¥ Arquitectura basada en componentes: Angular utiliza una arquitectura basada en componentes que facilita la organizaci3n y el mantenimiento del c3digo. Esta arquitectura tambi3n promueve la reutilizaci3n de componentes y una mayor modularidad.
- ¥ Ecosistema rico: Angular tiene un ecosistema rico y en continua evoluci3n, con una gran cantidad de bibliotecas y herramientas de terceros disponibles para simplificar el desarrollo y mejorar la productividad.
- ¥ Programaci3n reactiva: Angular se integra bien con bibliotecas de programaci3n reactiva como RxJS, lo que permite a los desarrolladores gestionar *flujos de datos* y *eventos* de manera eficiente.
- ¥ Buenas pr3cticas integradas: Angular promueve buenas pr3cticas de desarrollo, como la inyecci3n de dependencias y el uso de m3dulos, lo que facilita la creaci3n de aplicaciones escalables y f3cilmente mantenibles.
- ¥ Herramientas de desarrollo: Angular proporciona una serie de herramientas de desarrollo, como Angular CLI y Angular DevTools, que facilitan la creaci3n, prueba y depuraci3n de aplicaciones Angular.
- ¥ Amplia documentaci3n y comunidad: Angular cuenta con una amplia documentaci3n y una comunidad activa y comprometida que puede ayudar a resolver problemas y compartir conocimientos.

5. Popularidad de Angular

Angular es uno de los frameworks de JavaScript m3s populares y ampliamente utilizados. Su adopci3n en la industria y su creciente base de usuarios han llevado a un ecosistema s3lido y una gran cantidad de recursos de aprendizaje.

Adem3s, la demanda de desarrolladores Angular en el mercado laboral sigue siendo alta, lo que hace que el aprendizaje de Angular sea una inversi3n valiosa en t3rminos de oportunidades de empleo y crecimiento profesional.

6. Versiones de Angular

- ¥ Angular 2.0 (septiembre de 2016): Lanzamiento inicial de Angular (renombrado de AngularJS). Introdujo una arquitectura basada en componentes, mejor3 el rendimiento con la detecci3n de cambios y la compilaci3n AOT, y a3adi3 soporte para programaci3n reactiva con RxJS.
- ¥ Angular 4.0 (marzo de 2017): Mejor3 el rendimiento y redujo el tama3o del paquete. Tambi3n introdujo la funci3n ngIf con else y una nueva API HttpClient.
- ¥ Angular 5.0 (noviembre de 2017): Mejor3 el soporte para aplicaciones web progresivas (PWA) y la optimizaci3n del tama3o del paquete. Introdujo el HttpClient como el m3todo est3ndar para

realizar solicitudes HTTP.

- ¥ Angular 6.0 (mayo de 2018): Introdujo el Angular CLI 6 y la configuración de múltiples proyectos en un único espacio de trabajo. También introdujo el paquete Angular Material CDK y mejoró el soporte para la creación de bibliotecas.
- ¥ Angular 7.0 (octubre de 2018): Mejoras en el rendimiento, actualizaciones automáticas de la aplicación y la CLI. Introdujo la función drag and drop y la función de redimensionamiento en Angular Material CDK y mejoró el soporte para el scrolling virtual.
- ¥ Angular 8.0 (mayo de 2019): Introdujo la carga diferida (lazy loading) con la sintaxis `import()`, el soporte para Web Workers y las mejoras en la compilación con Ivy, el nuevo motor de renderizado.
- ¥ Angular 9.0 (febrero de 2020): Ivy se convirtió en el motor de renderizado y compilación predeterminado. Esto trajo mejoras significativas en el rendimiento, la detección de errores y la reducción del tamaño del paquete.
- ¥ Angular 10.0 (junio de 2020): Mejoras en la calidad, el rendimiento y el ecosistema, incluida la adopción de la versión TypeScript 3.9 y la eliminación de soporte para versiones antiguas de TypeScript. También se actualizaron las advertencias de navegadores no compatibles.
- ¥ Angular 11.0 (noviembre de 2020): Mejoras en el rendimiento de la construcción y el despliegue, actualizaciones de herramientas como ESLint, soporte para TypeScript 4.0 y actualizaciones de Angular Language Service.
- ¥ Angular 12.0 (mayo de 2021): Introdujo soporte para TypeScript 4.2, eliminó el soporte para IE11, mejoró el soporte para Webpack 5 y comenzó a desalentar el uso de View Engine en favor de Ivy.
- ¥ Ver las nuevas versiones en <https://angular.io/guide/releases> y <https://github.com/angular/angular/releases>

7. Arquitectura de Angular

La arquitectura de Angular se compone de varios elementos principales que trabajan juntos para crear aplicaciones escalables y modulares.

Angular es un framework extenso y complejo, por lo que es importante tener una ruta de aprendizaje clara para dominar sus conceptos básicos y avanzados. A continuación se presentan algunos recursos que pueden ayudar a los desarrolladores a aprender Angular.

7.1. Componentes

Se empieza aprendiendo los componentes como bloques de construcción fundamentales en Angular.

Es necesario aprender cómo los componentes tienen su lógica (TypeScript), plantilla (HTML) y estilo (CSS).

Probar a crear un ejemplo básico de un componente, sin entrar en qué es un módulo en este punto, nos enfocamos en el concepto de componentes en sí mismo.

7.2. Módulos

Después de explicar los componentes, conviene explorar los módulos como contenedores para agrupar componentes y otros elementos relacionados.

Los módulos ayudan a organizar el código y promueven la *reutilización de código* y la *modularidad*.

Se comienza a mencionar el módulo raíz `AppModule` y cómo se relaciona con el componente principal `AppComponent`.

Será necesario estudiar las *declaraciones*, *importaciones*, *proveedores* y la *propiedad de arranque* en los módulos.

7.3. Relación entre componentes y módulos

Ahora que ya se comprenden los componentes y los módulos por separado, se debe conocer cómo están interconectados.

Crear un ejemplo de cómo un módulo declara los componentes que pertenecen a él y cómo importar módulos adicionales para acceder a sus componentes y servicios.

7.4. Creación de componentes y módulos

A continuación, se debe conocer el proceso de creación de componentes y módulos utilizando Angular CLI o Angular Schematics.

Cuando se utiliza una de las opciones mencionadas, se generan automáticamente los archivos necesarios, así como las modificaciones en el módulo raíz.

7.5. Servicios

Ahora que ya se tiene una base sólida sobre componentes y módulos, se presentan los servicios como una forma de compartir lógica y datos entre componentes.

Se debe conocer la inyección de dependencias y cómo Angular utiliza este enfoque para proporcionar instancias de servicios a componentes que los requieran.

7.6. Uso de servicios en componentes y módulos

Se debe saber cómo crear un servicio utilizando Angular CLI o Angular Schematics y cómo inyectar ese servicio en un componente.

Se debe aprender cómo los servicios se pueden proporcionar a nivel de módulo o componente y cómo se pueden utilizar en diferentes partes de la aplicación.

7.7. Inyección de dependencias (DI)

Aunque la inyección de dependencias ya se ha introducido en los servicios, se debe profundizar en más detalle.

Aprender cómo DI ayuda a *desacoplar* clases y a mejorar la escalabilidad y la capacidad de prueba (testing) de la aplicación.

7.8. Directivas

Se introducen las directivas como una forma de manipular el DOM y añadir comportamientos dinámicos a los elementos HTML.

Aprender a diferenciar las directivas estructurales (como `*ngIf` y `*ngFor`) y las directivas de atributos (como `[ngClass]` y `[ngStyle]`).

También se debe aprender cómo crear directivas personalizadas.

7.9. Enrutamiento y navegación

Se presenta el concepto de enrutamiento y navegación en aplicaciones Angular.

Es necesario aprender cómo configurar rutas, definir enlaces de navegación y cómo navegar programáticamente.

También se pueden estudiar temas como la protección de rutas y la carga diferida de módulos.

7.10. Formularios

Se deben conocer los dos enfoques para trabajar con formularios en Angular:

- ¥ Formularios basados en plantillas.

- ¥ Formularios reactivos.

Se debe aprender cómo validar y procesar formularios, así como el manejo de eventos de formularios.

7.11. Pipes

Se presentan los pipes como una forma de transformar y formatear datos en las plantillas de HTML.

Es necesario aprender a cómo utilizar los pipes integrados y cómo crear pipes personalizados.

7.12. HTTP y comunicación con API

Se debe aprender a cómo realizar solicitudes HTTP para interactuar con servicios web y API

utilizando `HttpClient`.

Cuando se interactúa con APIs es necesario aprender a manejar errores, interceptar solicitudes y respuestas y trabajar con observables.

7.13. Detección de cambios

Es necesario aprender el proceso de detección de cambios en Angular, el concepto de zonas y cómo utilizar estrategias de detección de cambios, como `ChangeDetectionStrategy.OnPush`, para optimizar el rendimiento de la aplicación.

7.14. Pruebas unitarias y de integración

Se introducen las pruebas unitarias y de integración en Angular utilizando herramientas como Jasmine, Karma y TestBed.

Se muestra cómo escribir y ejecutar pruebas para componentes, servicios y directivas.

8. Recursos de aprendizaje

¥ Angular: <https://angular.io/>

¥ Angular CLI: <https://cli.angular.io/>

¥ Angular Material: <https://material.angular.io/>

9. Instalación de Angular

Para instalar Angular, se debe tener instalado Node.js y npm (Node Package Manager).

Se puede instalar Angular de forma global o localmente.

9.1. Instalación global

Para instalar Angular de forma global, se debe ejecutar el siguiente comando:

```
npm install -g @angular/cli
```

Comprobar la versión de angular

```
ng version
```

En caso de querer desinstalarlo para volver a instalarlo

```
npm uninstall -g @angular/cli
```



```
npm install -g @angular/cli
```

9.2. Instalaci3n local

Para instalar Angular de forma local, se debe ejecutar el siguiente comando:

```
npm install --save-dev @angular/cli
```

9.3. Creaci3n de un proyecto

Para crear un proyecto en Angular, se debe ejecutar el siguiente comando:

```
ng new my-app
```

10. Ejecuci3n de un proyecto

Para ejecutar un proyecto en Angular, se debe ejecutar el siguiente comando:

```
ng serve
```

11. Creaci3n de un componente

Para crear un componente en Angular, se debe ejecutar el siguiente comando:

```
ng generate component my-component
```