

# Angular 2

[Inicio](#)

[Visión](#)

[Ambiente](#)

[Hola Mundo](#)

[Despliegue](#)

[Implementación en servidores NGNIX en Windows](#)

[Módulos](#)

[Arquitectura](#)

[Componentes](#)

[Parámetros](#)

[Classname - Este es el nombre que se debe dar a la clase.](#)

[Nombre de la propiedad - Este es el nombre que debe darse a la propiedad.](#)

[PropertyType - Dado que TypeScript está fuertemente mecanografiado, debe dar un tipo a la propiedad.](#)

[Valor - Este es el valor que debe darse a la propiedad.](#)

[Ejemplo](#)

[Plantillas](#)

[Directivas](#)

## Inicio

Angular 2 es un framework JavaScript de código abierto para crear aplicaciones web en HTML y JavaScript. Este tutorial examina los diversos aspectos del marco Angular 2 que incluye los fundamentos del marco, la configuración de Angular y cómo trabajar con los diversos aspectos del marco. Otros temas discutidos en el tutorial son capítulos avanzados como interfaces, componentes anidados y servicios dentro de Angular. Temas como enrutamiento, módulos y arreglos también se tratan en este tutorial.

Audiencia

Este tutorial es para aquellos que estén interesados en aprender la nueva versión del marco Angular. La primera versión del marco ha estado allí por bastante tiempo y es sólo fuera de la tarde que Angular 2 se ha convertido en popular entre la comunidad de desarrollo web.

#### Requisitos previos

El usuario debe estar familiarizado con los conceptos básicos de desarrollo web y JavaScript. Dado que el marco Angular se basa en el marco de JavaScript, se hace más fácil para el usuario entender Angular si saben JavaScript.

## Visión

Angular JS es un framework de código abierto construido sobre JavaScript. Fue construido por los desarrolladores de Google. Este marco se utilizó para superar los obstáculos encontrados al trabajar con aplicaciones de una sola página. Además, las pruebas se consideraron un aspecto clave a la hora de construir el marco. Se garantizó que el marco podría ser fácilmente probado. La liberación inicial del marco fue en octubre de 2010.

#### Características de Angular 2

A continuación se presentan las características clave de Angular 2 -

**Componentes** - La versión anterior de Angular tenía un enfoque de controladores, pero ahora ha cambiado el enfoque a tener componentes sobre los controladores. Los componentes ayudan a construir las aplicaciones en muchos módulos. Esto ayuda a mantener mejor la aplicación durante un período de tiempo.

**TypeScript** - La versión más reciente de Angular se basa en TypeScript. Este es un superconjunto de JavaScript y es mantenido por Microsoft.

**Servicios** - Los servicios son un conjunto de código que puede ser compartido por los diferentes componentes de una aplicación. Por ejemplo, si tuviera un componente de datos que recogiera datos de una base de datos, podría tenerlo como un servicio compartido que podría utilizarse en varias aplicaciones.

Además, Angular 2 tiene mejores capacidades de manejo de eventos, plantillas potentes y un mejor soporte para dispositivos móviles.

#### Componentes de Angular 2

Angular 2 tiene los siguientes componentes:

**Módulos** - Se utiliza para dividir la aplicación en partes lógicas del código. Cada pieza

de código o módulo está diseñada para realizar una sola tarea.

**Componente** - Se puede utilizar para reunir los módulos.

**Plantillas** - Se utiliza para definir las vistas de una aplicación Angular JS.

**Metadatos:** se puede utilizar para agregar más datos a una clase Angular JS.

**Servicio:** se utiliza para crear componentes que se pueden compartir en toda la aplicación.

Discutiremos todos estos componentes en detalle en los siguientes capítulos de este tutorial.

El sitio oficial de Angular es <https://angular.io/> El sitio tiene toda la información y documentación sobre Angular 2.

## Ambiente

Para comenzar a trabajar con Angular 2, debe instalar los siguientes componentes clave.

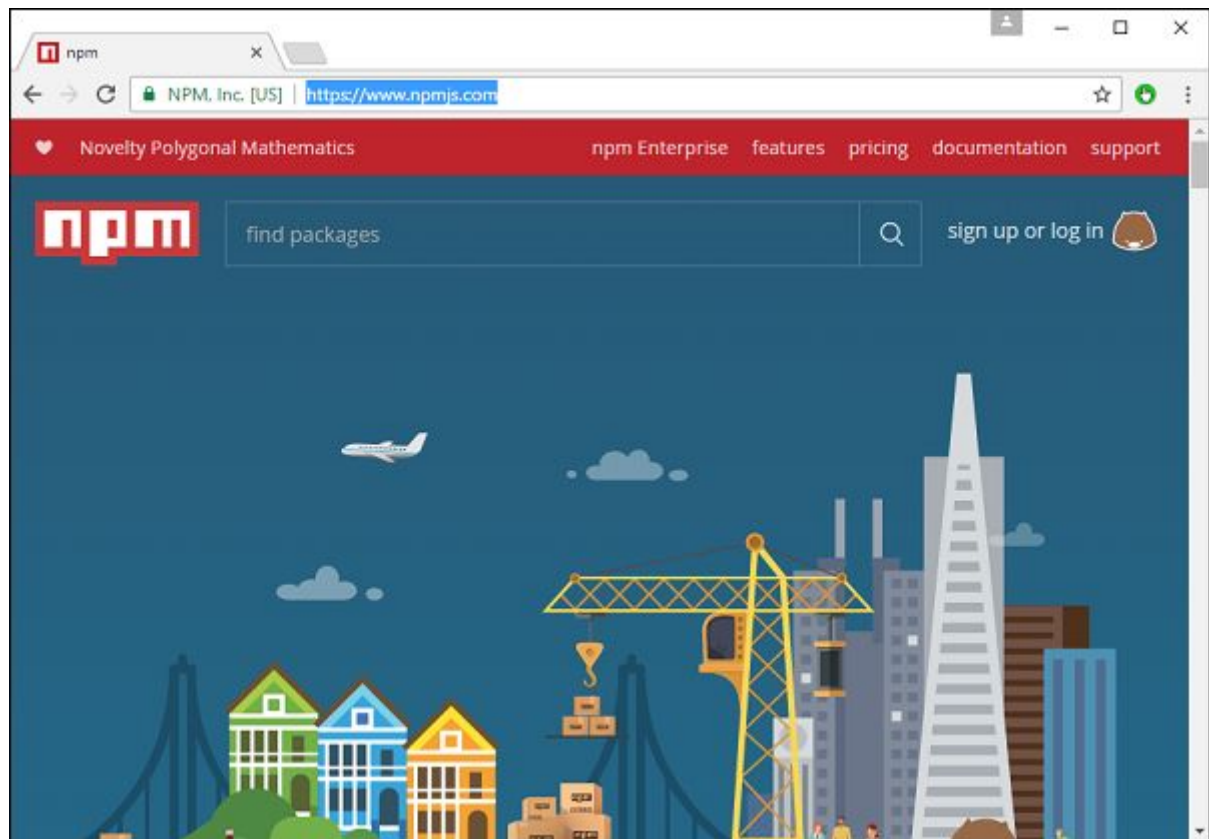
Npm - Esto se conoce como el gestor de paquetes de nodos que se utiliza para trabajar con los repositorios de código abierto. Angular JS como un marco tiene dependencias de otros componentes. Y npm se puede utilizar para descargar estas dependencias y adjuntarlas a su proyecto.

Git - Este es el software de código fuente que se puede utilizar para obtener la aplicación de muestra desde el sitio github angular.

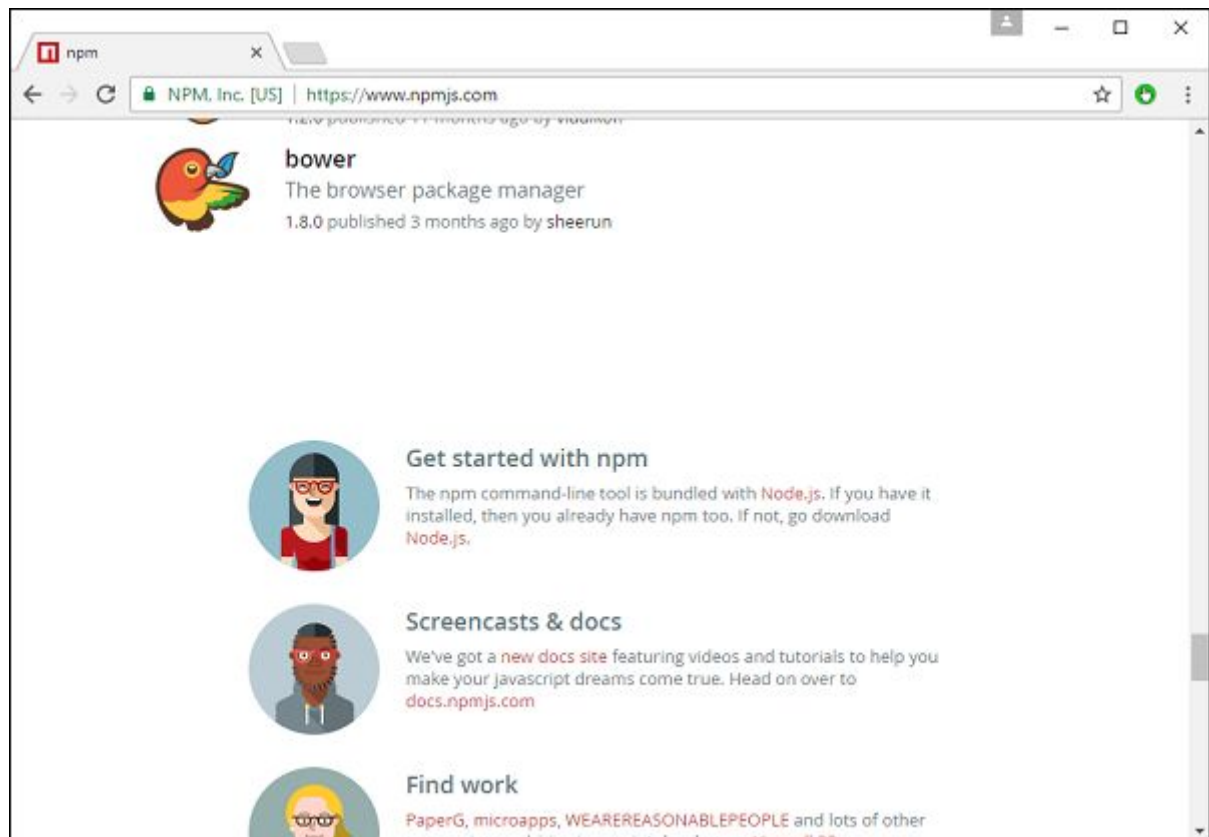
Editor - Hay muchos editores que se pueden utilizar para el desarrollo de JS Angular, como el código de Visual Studio y WebStorm. En nuestro tutorial, usaremos código de Visual Studio que viene gratis de Microsoft.

Instalación de npm

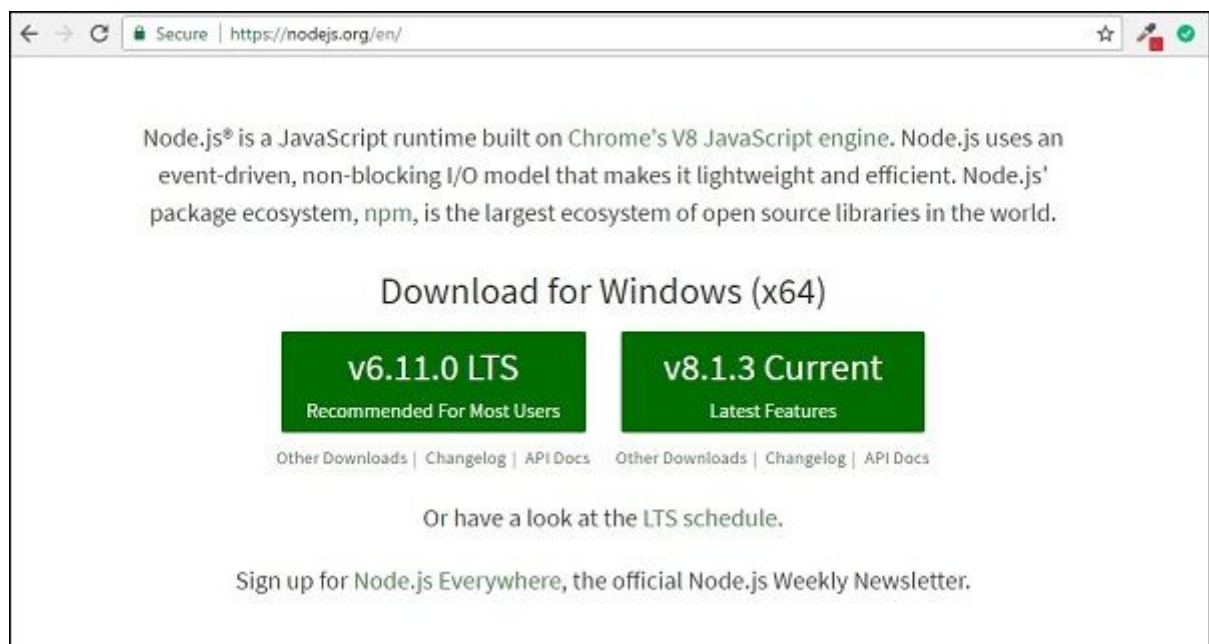
Veamos ahora los pasos para obtener npm instalado. El sitio oficial de npm es <https://www.npmjs.com/>



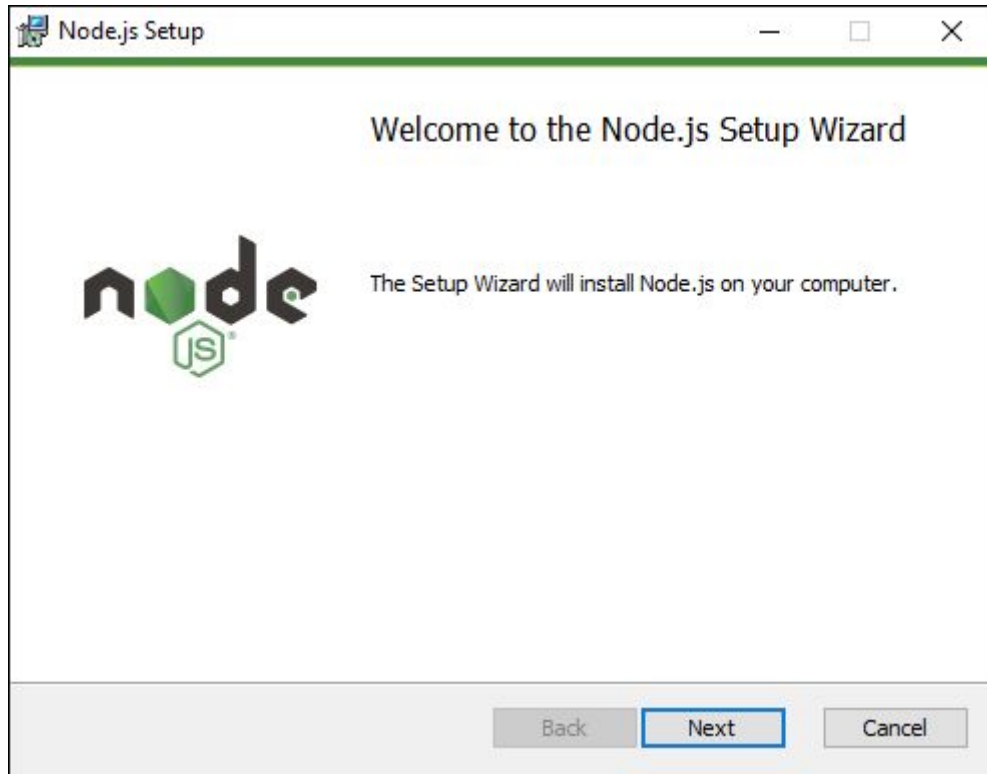
Paso 1 - Vaya a la sección "get stated with npm" en el sitio.



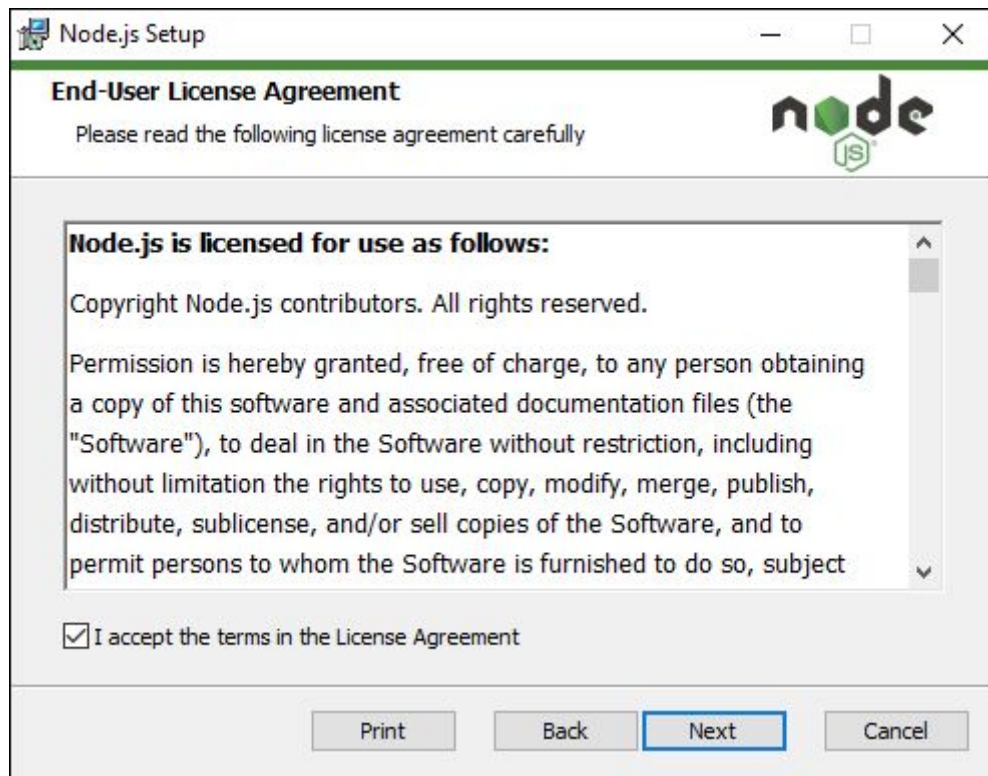
Paso 2 - En la siguiente pantalla, elija el instalador para descargar, dependiendo del sistema operativo. Para el propósito de este ejercicio, descargue la versión de 64 bits de Windows.



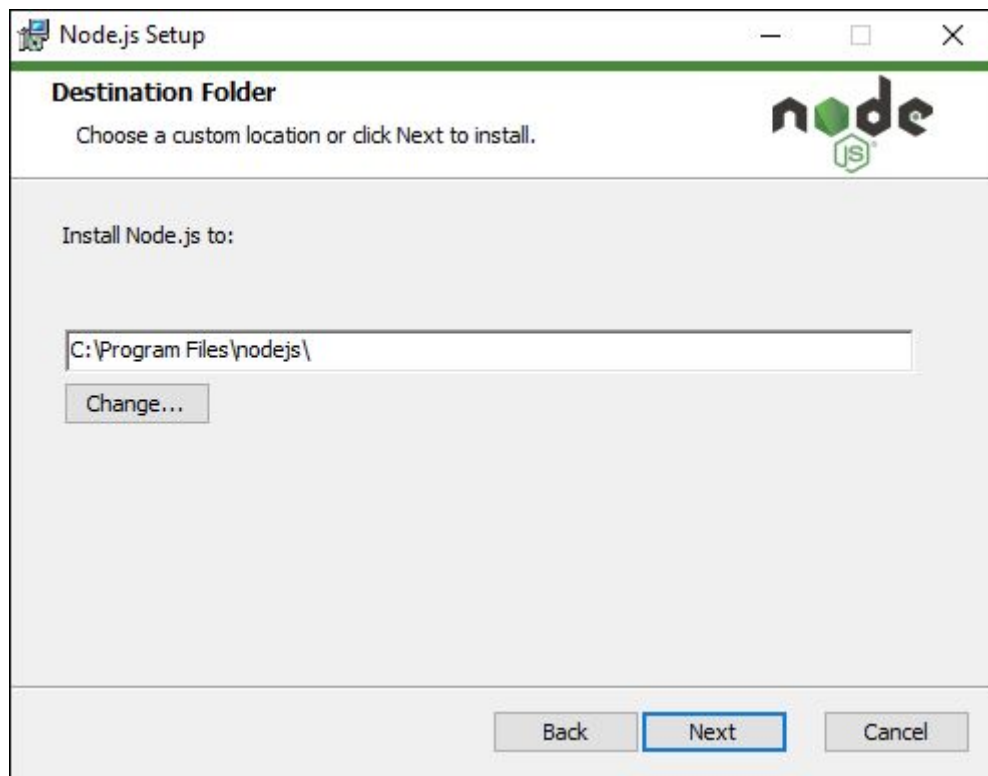
Paso 3 - Inicie el instalador. En la pantalla inicial, haga clic en el botón Siguiente.



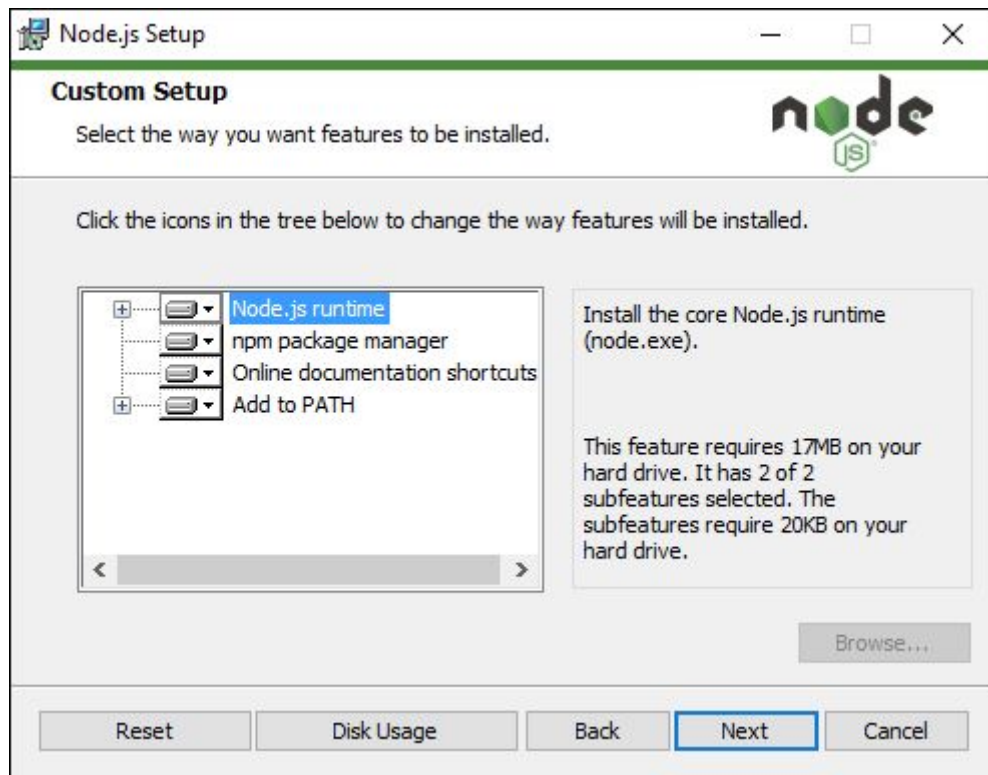
Paso 4 - En la siguiente pantalla, acepte el acuerdo de licencia y haga clic en el botón siguiente.



Paso 5 - En la siguiente pantalla, elija la carpeta de destino para la instalación y haga clic en el botón Siguiente.

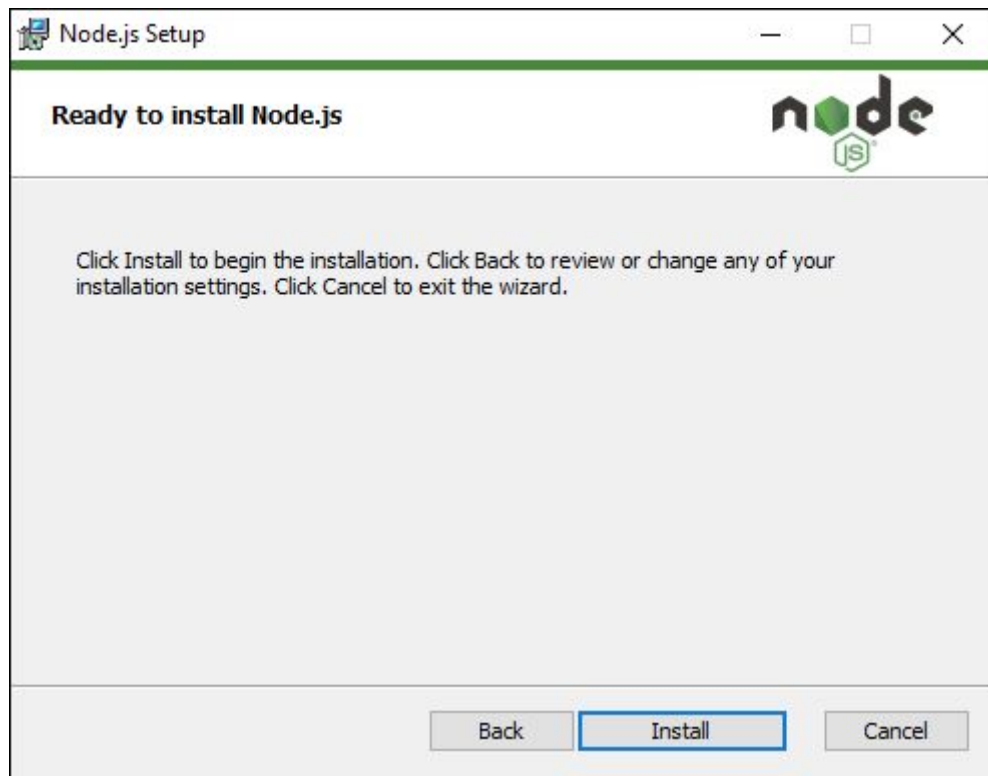


Paso 6 - Elija los componentes en la siguiente pantalla y haga clic en el botón Siguiente. Puede aceptar todos los componentes para la instalación predeterminada.

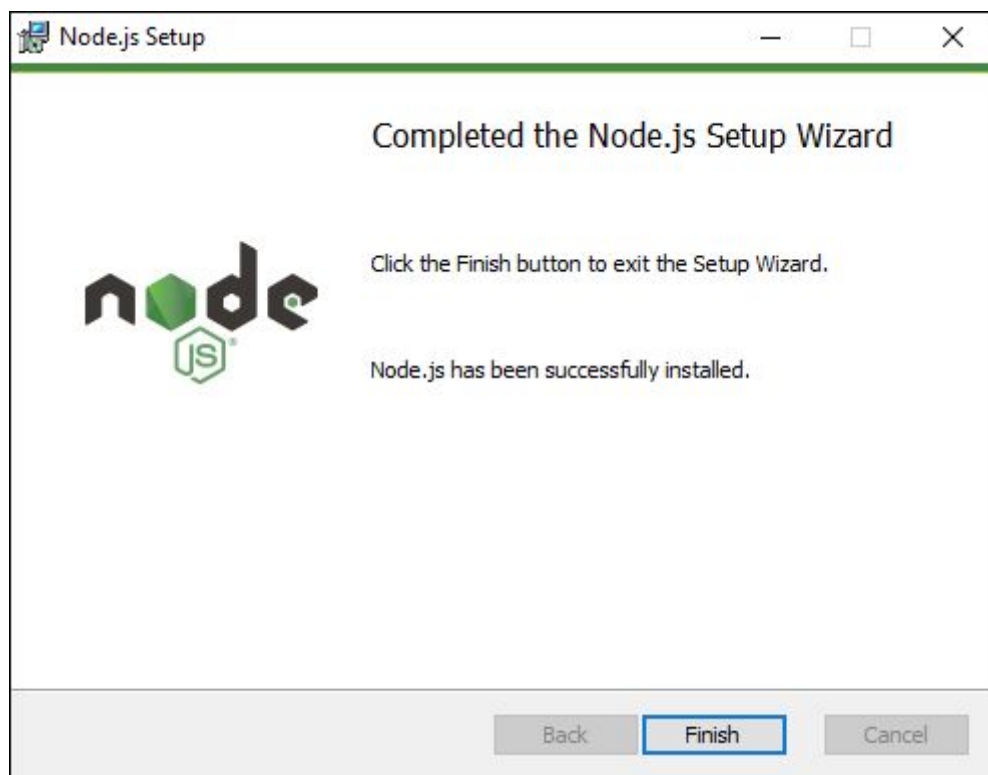


Paso 7 - En la siguiente pantalla, haga clic en el botón Instalar.

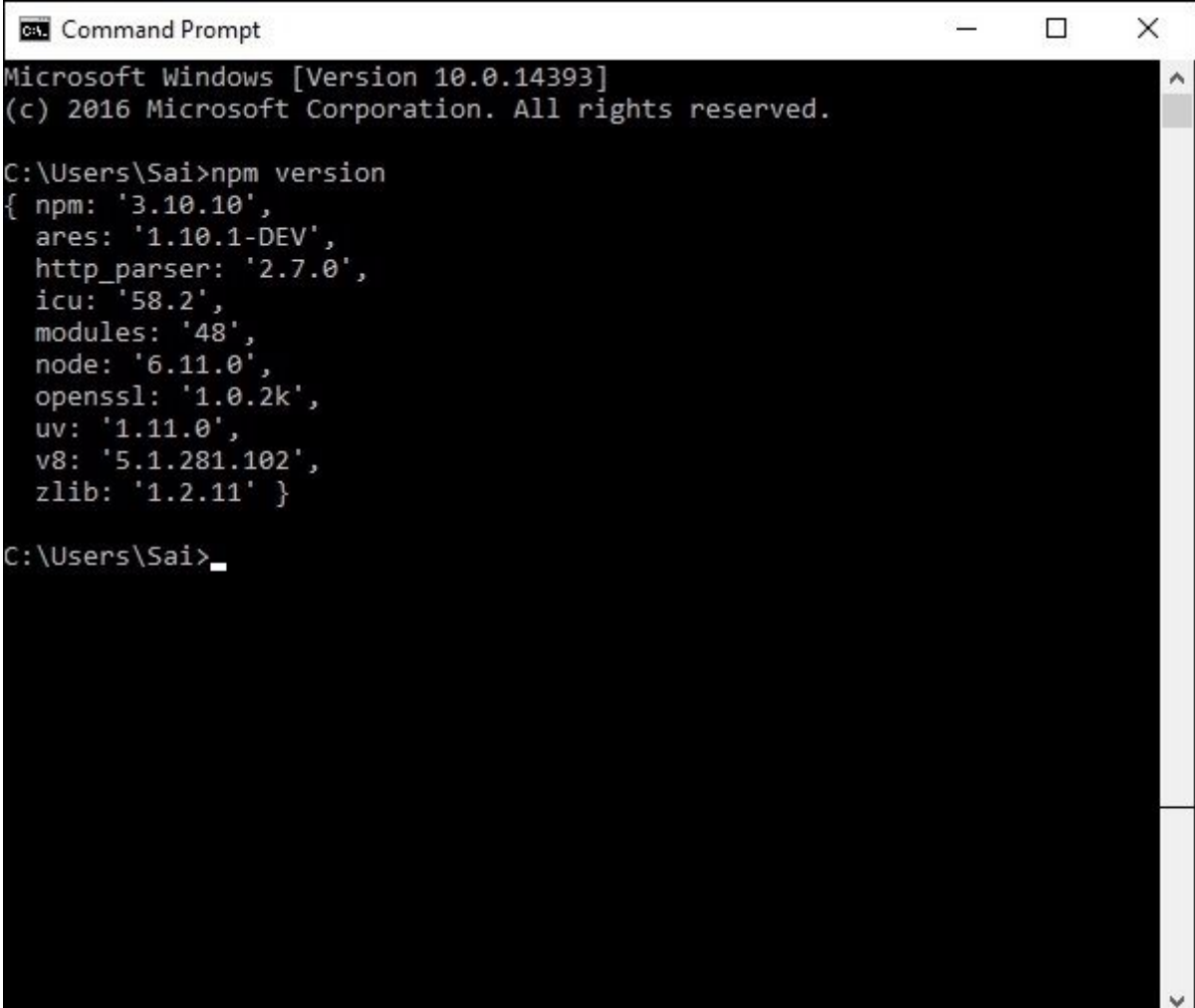




Paso 8 - Una vez finalizada la instalación, haga clic en el botón Finalizar.



Paso 9 - Para confirmar la instalación, en el símbolo del sistema puede emitir la versión de comando npm. Obtendrá el número de versión de npm como se muestra en la siguiente captura de pantalla.

A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". The window content shows the following text:

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Sai>npm version
{ npm: '3.10.10',
  ares: '1.10.1-DEV',
  http_parser: '2.7.0',
  icu: '58.2',
  modules: '48',
  node: '6.11.0',
  openssl: '1.0.2k',
  uv: '1.11.0',
  v8: '5.1.281.102',
  zlib: '1.2.11' }

C:\Users\Sai>
```

## Hola Mundo

Hay varias maneras de comenzar con su primera aplicación Angular JS.

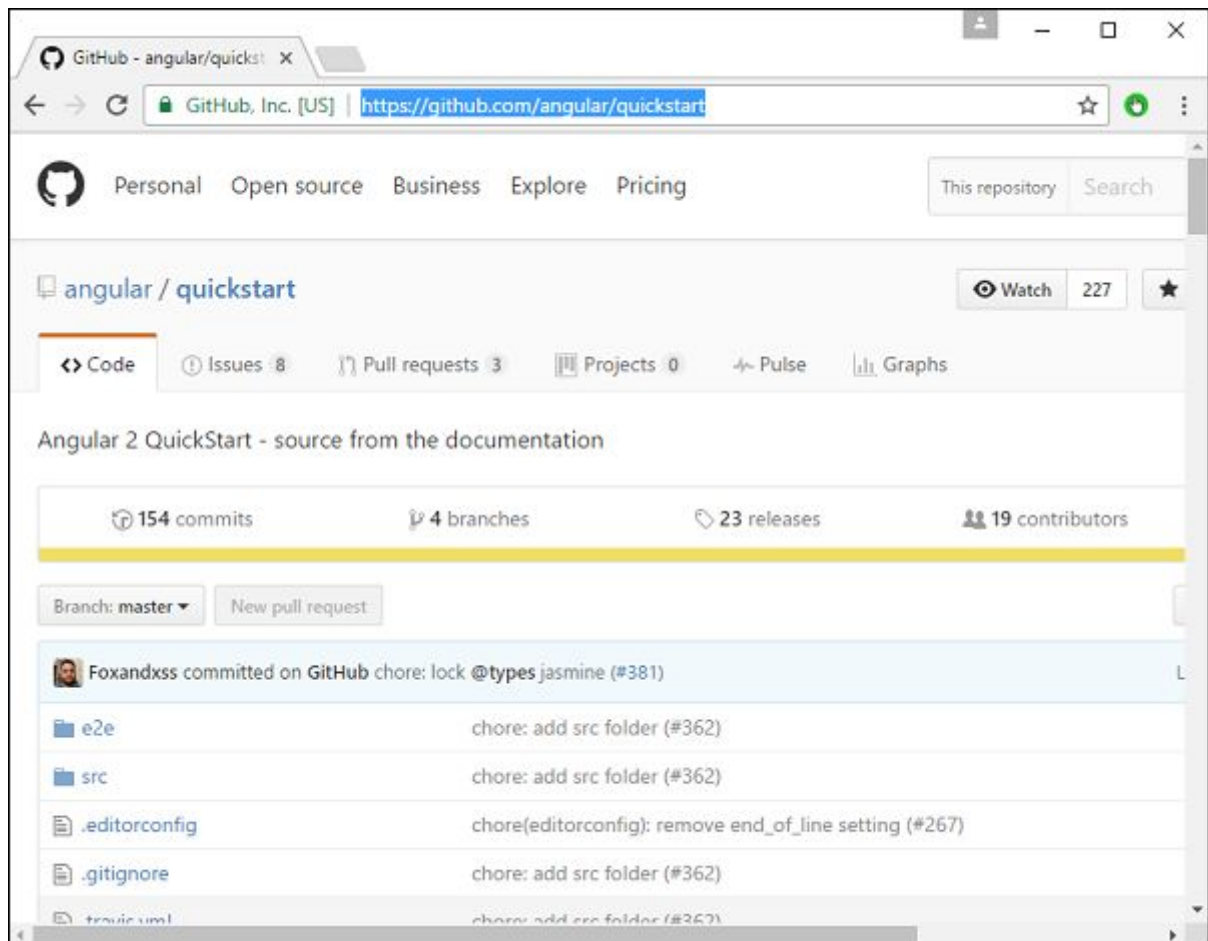
Una forma es hacer todo desde cero, que es la forma más difícil y no la preferida. Debido a las muchas dependencias, se hace difícil obtener esta configuración.

Otra forma es utilizar el inicio rápido en Angular Github. Contiene el código necesario para empezar. Esto es normalmente lo que es optado por todos los desarrolladores y esto es lo que vamos a mostrar para la aplicación Hello World.

La forma final es usar CLI Angular. Discutiremos esto en detalle en un capítulo aparte.

A continuación se indican los pasos para obtener una aplicación de ejemplo que se ejecuta a través de github.

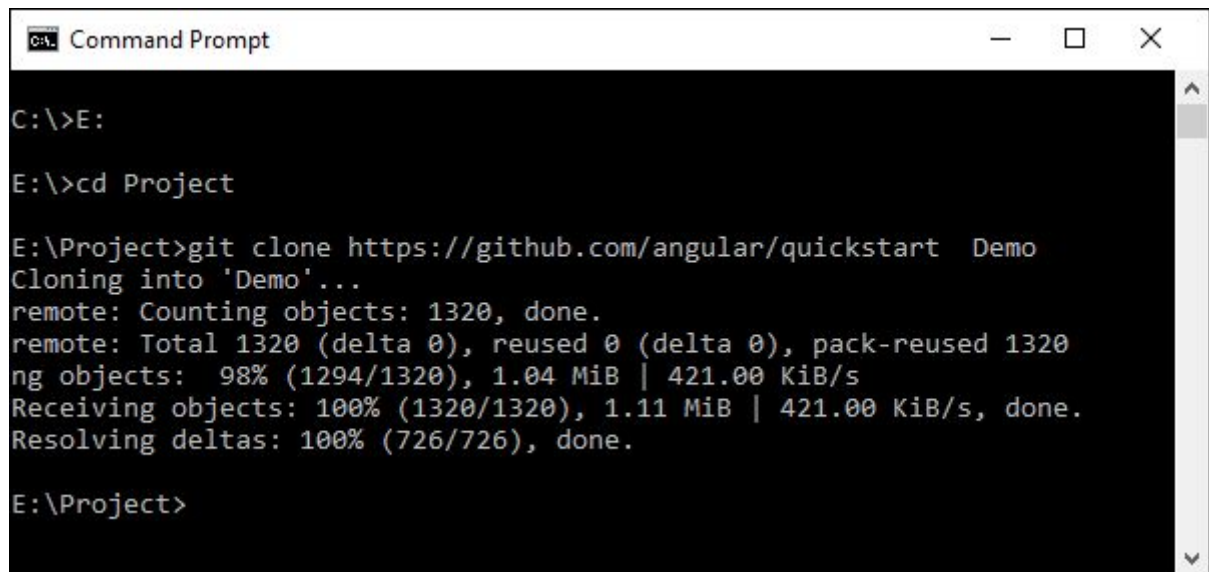
Paso 1 - Ir a la url de github - <https://github.com/angular/quickstart>



Paso 2 - Vaya a la línea de comandos, cree un directorio de proyecto. Esto puede ser un directorio vacío. En nuestro ejemplo, hemos creado un directorio llamado Project.

Paso 3 - A continuación, en el símbolo del sistema, vaya a este directorio y emita el siguiente comando para clonar el repositorio github en su sistema local. Puede hacerlo mediante la emisión del siguiente comando:

```
git clone https://github.com/angular/quickstart Demo
```



```
Command Prompt

C:\>E:

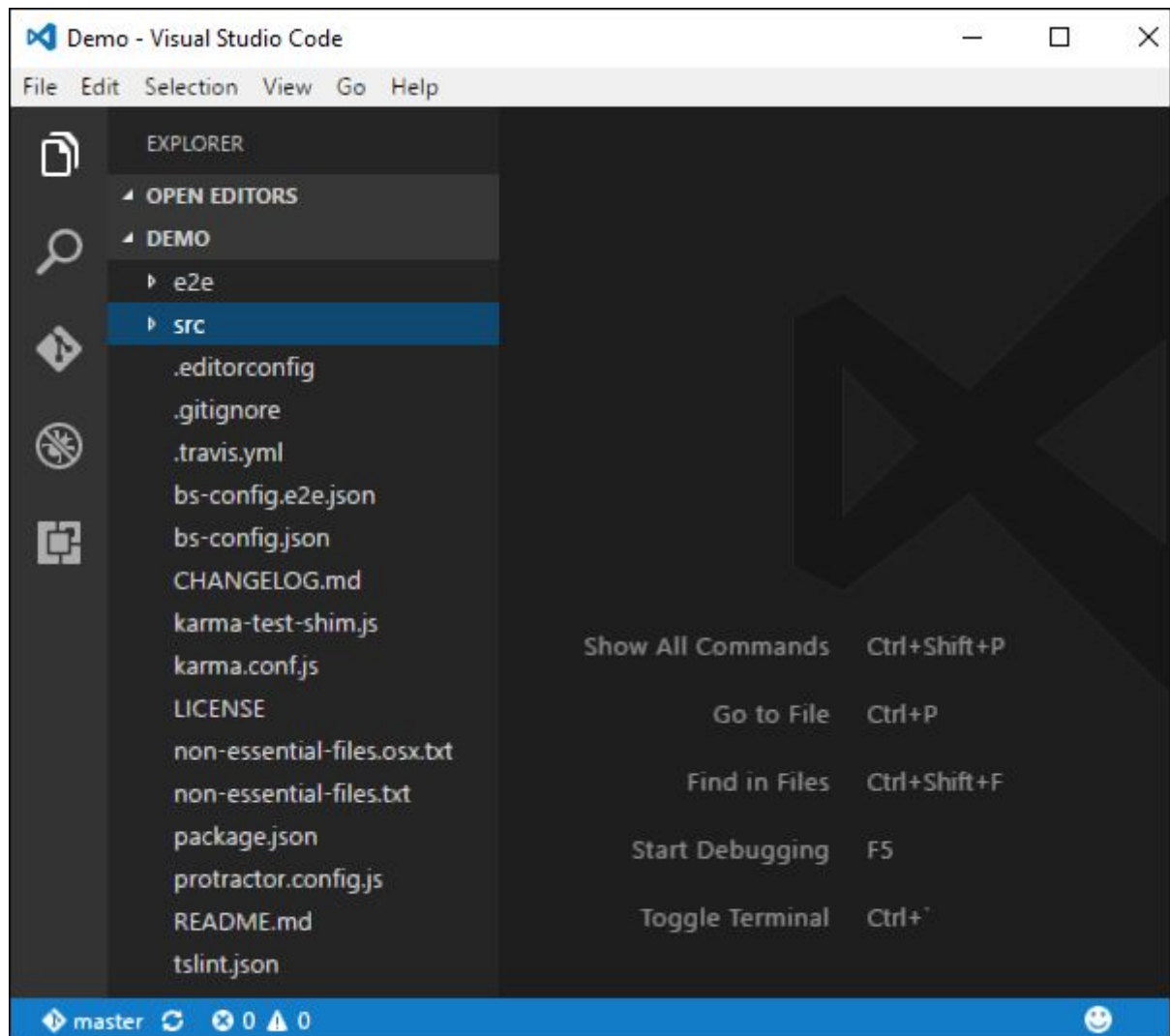
E:\>cd Project

E:\Project>git clone https://github.com/angular/quickstart Demo
Cloning into 'Demo'...
remote: Counting objects: 1320, done.
remote: Total 1320 (delta 0), reused 0 (delta 0), pack-reused 1320
Receiving objects: 98% (1294/1320), 1.04 MiB | 421.00 KiB/s
Receiving objects: 100% (1320/1320), 1.11 MiB | 421.00 KiB/s, done.
Resolving deltas: 100% (726/726), done.

E:\Project>
```

Esto creará una muestra de aplicación Angular JS en su máquina local.

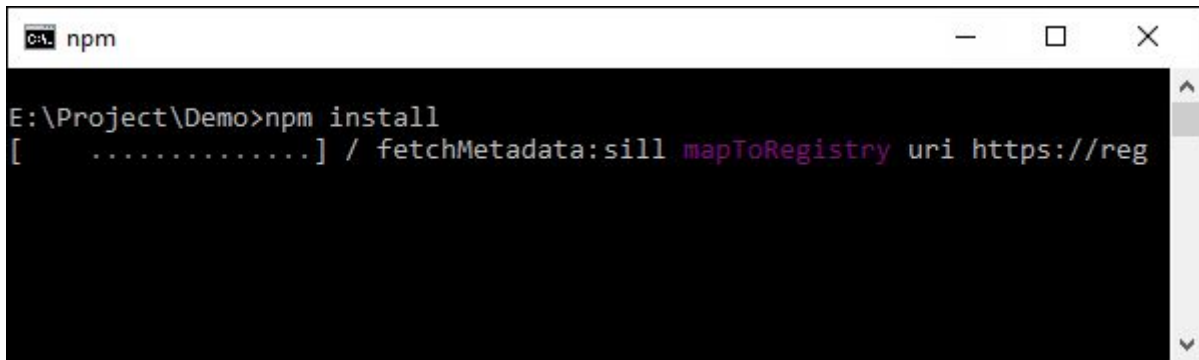
Paso 4: abra el código en código de Visual Studio.



Paso 5 - Vaya al símbolo del sistema y en su carpeta de proyecto de nuevo y emita el siguiente comando:

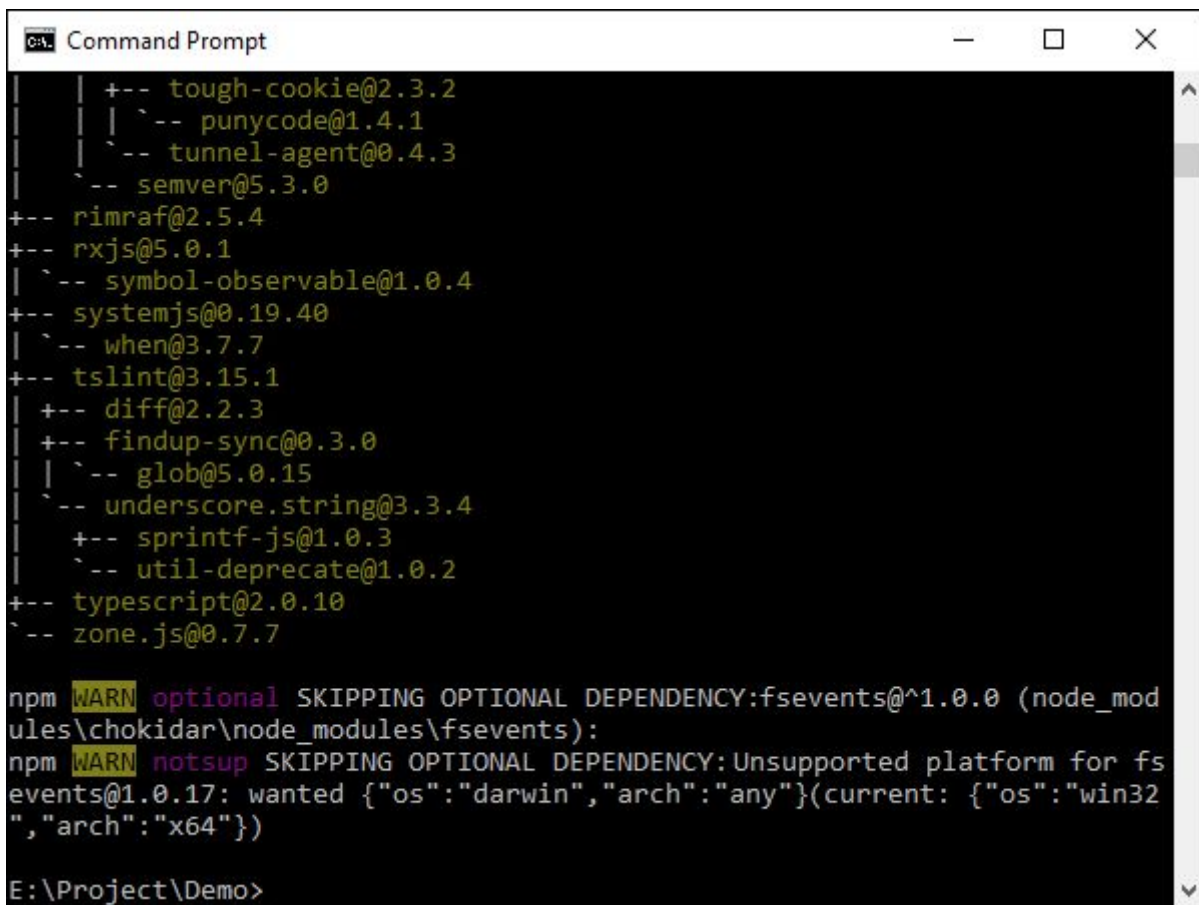
```
npm install
```

Esto instalará todos los paquetes necesarios que son necesarios para que la aplicación de JS Angular funcione.



```
npm
E:\Project\Demo>npm install
[ ..... ] / fetchMetadata:sill mapToRegistry uri https://reg
```

Una vez hecho esto, debería ver una estructura de árbol con todas las dependencias instaladas.



```
Command Prompt
+-- tough-cookie@2.3.2
|  |-- punycode@1.4.1
|  |-- tunnel-agent@0.4.3
|  |-- semver@5.3.0
+-- rimraf@2.5.4
+-- rxjs@5.0.1
|  |-- symbol-observable@1.0.4
+-- systemjs@0.19.40
|  |-- when@3.7.7
+-- tslint@3.15.1
|  |-- diff@2.2.3
|  |-- findup-sync@0.3.0
|  |  |-- glob@5.0.15
|  |  |-- underscore.string@3.3.4
|  |  |-- sprintf-js@1.0.3
|  |  |-- util-deprecate@1.0.2
+-- typescript@2.0.10
|-- zone.js@0.7.7

npm WARN optional SKIPPING OPTIONAL DEPENDENCY:fsevents@^1.0.0 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY:Unsupported platform for fsevents@1.0.17: wanted {"os":"darwin","arch":"any"}(current: {"os":"win32","arch":"x64"})

E:\Project\Demo>
```

Paso 6 - Vaya a la carpeta Demo → src → app → app.component.ts. Encuentra las siguientes líneas de código:

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

@Component({
```

```
    selector: 'my-app',  
    template: '<h1>Hello {{name}}</h1>',  
  }  
}  
export class AppComponent { name = 'Angular'; }
```

Y reemplazar la palabra clave Angular con **Hello** como se muestra a continuación

```
import { Component } from '@angular/core';  
  
@Component({  
  selector: 'my-app',  
  template: '<h1>Hello {{name}}</h1>',  
})  
export class AppComponent { name = 'World'; }
```

Hay otros archivos que se crean como parte de la creación del proyecto para la aplicación Angular 2. Por el momento, no es necesario que te preocupes por los otros archivos de código, ya que todos ellos están incluidos como parte de tu aplicación Angular 2 y no necesitas ser cambiados para la aplicación Hello World.

Estaremos discutiendo estos archivos en los siguientes capítulos en detalle.

Nota: el código de Visual Studio compilará automáticamente todos sus archivos y creará archivos JavaScript para todos sus archivos mecanografiados.

Paso 7 - Ahora vaya a la línea de comandos y emita el comando `npm start`. Esto hará que el gestor de paquetes del Nodo inicie un servidor web lite y lance su aplicación Angular.



```
CA: npm
E:\Project\Demo>npm start

> angular-quickstart@1.0.0 prestart E:\Project\Demo
> npm run build

> angular-quickstart@1.0.0 build E:\Project\Demo
> tsc -p src/
```

```
CA: lite-server
platform-browser-dynamic.umd.js
[1] 17.02.17 18:19:20 200 GET /app/app.module.js
[1] 17.02.17 18:19:20 304 GET /@angular/compiler/bundles/compiler.umd.js
[1] 17.02.17 18:19:20 304 GET /@angular/core/bundles/core.umd.js
[1] 17.02.17 18:19:20 304 GET /@angular/platform-browser/bundles/platform-browser.umd.js
[1] 17.02.17 18:19:20 200 GET /app/app.component.js
[1] 17.02.17 18:19:20 304 GET /rxjs/symbol/observable.js
[1] 17.02.17 18:19:20 304 GET /rxjs/Subject.js
[1] 17.02.17 18:19:20 304 GET /rxjs/Observable.js
[1] 17.02.17 18:19:20 304 GET /@angular/common/bundles/common.umd.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/root.js
[1] 17.02.17 18:19:20 304 GET /rxjs/Subscriber.js
[1] 17.02.17 18:19:20 304 GET /rxjs/Subscription.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/ObjectUnsubscribedError.js
[1] 17.02.17 18:19:20 304 GET /rxjs/SubjectSubscription.js
[1] 17.02.17 18:19:20 304 GET /rxjs/symbol/rxSubscriber.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/toSubscriber.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/isFunction.js
[1] 17.02.17 18:19:20 304 GET /rxjs/Observer.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/isArray.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/isObject.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/tryCatch.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/errorObject.js
[1] 17.02.17 18:19:20 304 GET /rxjs/util/UnsubscriptionError.js
```

La aplicación Angular JS se iniciará ahora en el navegador y verá "Hello World" en el navegador como se muestra en la siguiente captura de pantalla.





## Despliegue

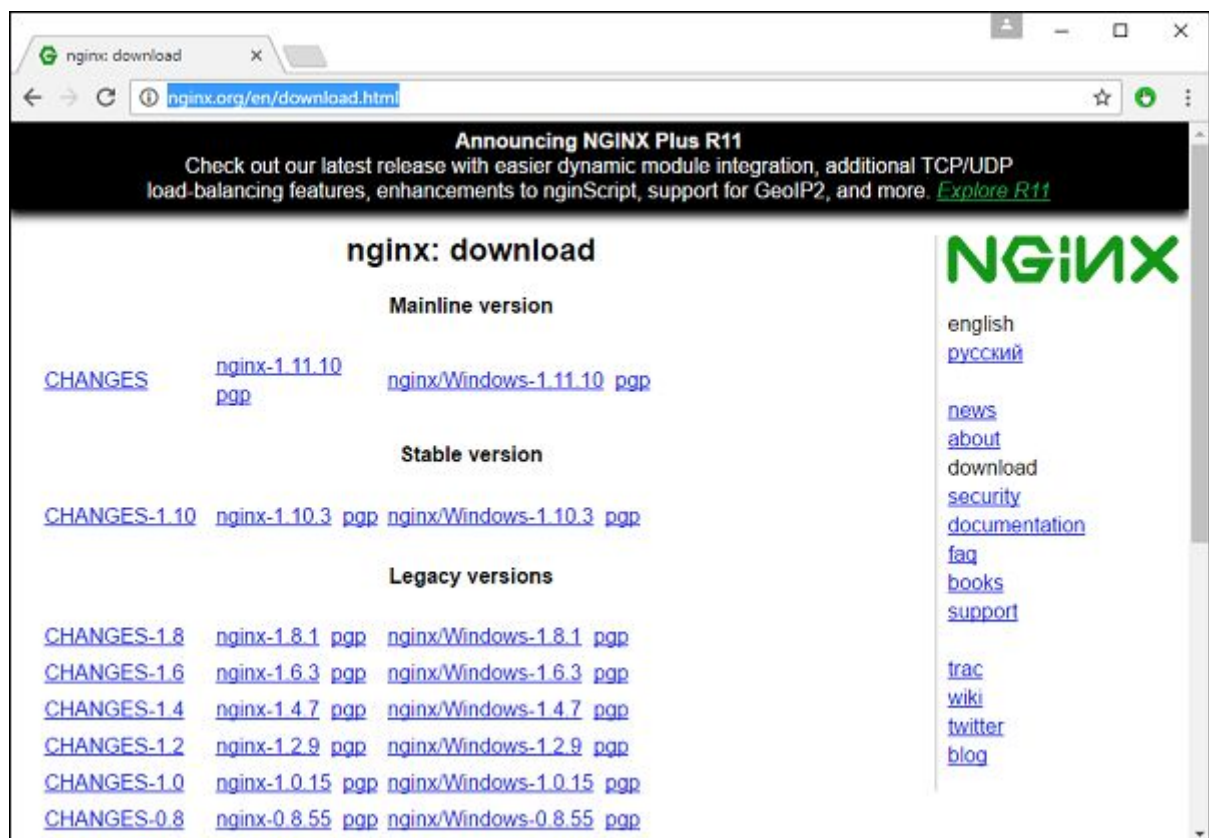
Este tema se centra en el despliegue de la aplicación Hello world anterior. Puesto que se trata de una aplicación Angular JS, se puede implementar en cualquier plataforma. Su desarrollo puede estar en cualquier plataforma.

En este caso, será en Windows con código de Visual Studio. Veamos ahora dos opciones de implementación.

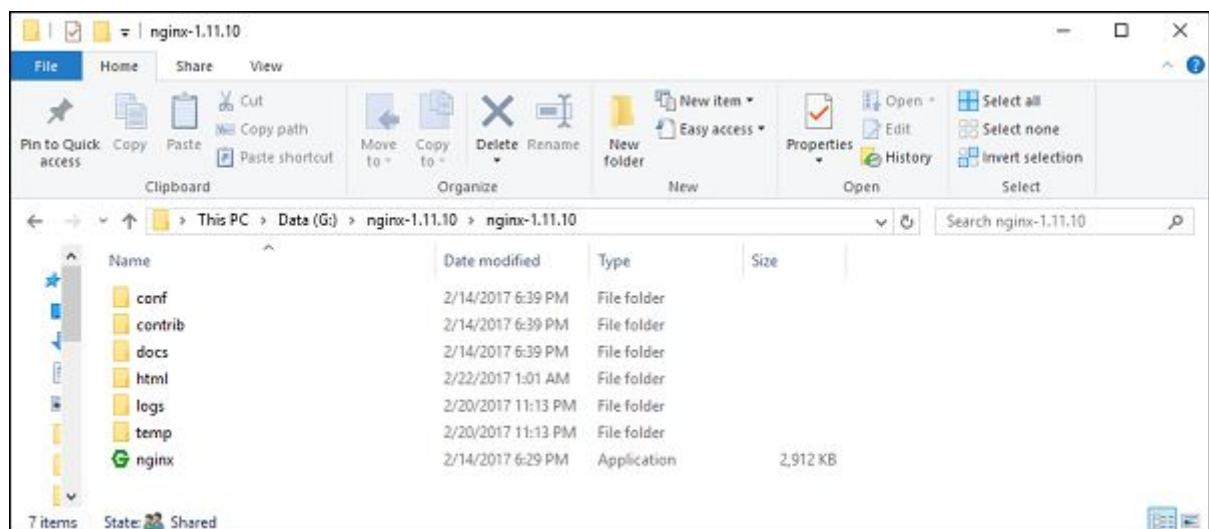
## Implementación en servidores NGNIX en Windows

Tenga en cuenta que puede utilizar cualquier servidor web en cualquier plataforma para alojar aplicaciones Angular JS. En este caso, tomaremos el ejemplo de NGNIX, que es un popular servidor web.

Paso 1 - Descargue el servidor web NGNIX desde la url siguiente  
<http://nginx.org/en/download.html>

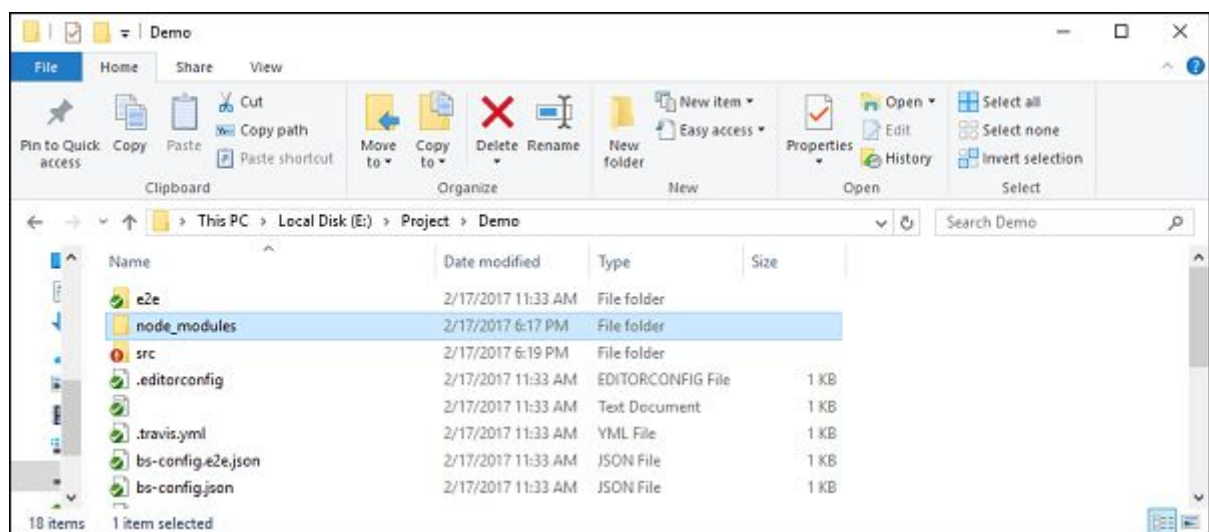


Paso 2 - Después de extraer el archivo zip descargado, ejecute el componente nginx.exe que hará que el servidor web se ejecute en segundo plano. A continuación, podrá ir a la página de inicio en la url - `http://localhost`

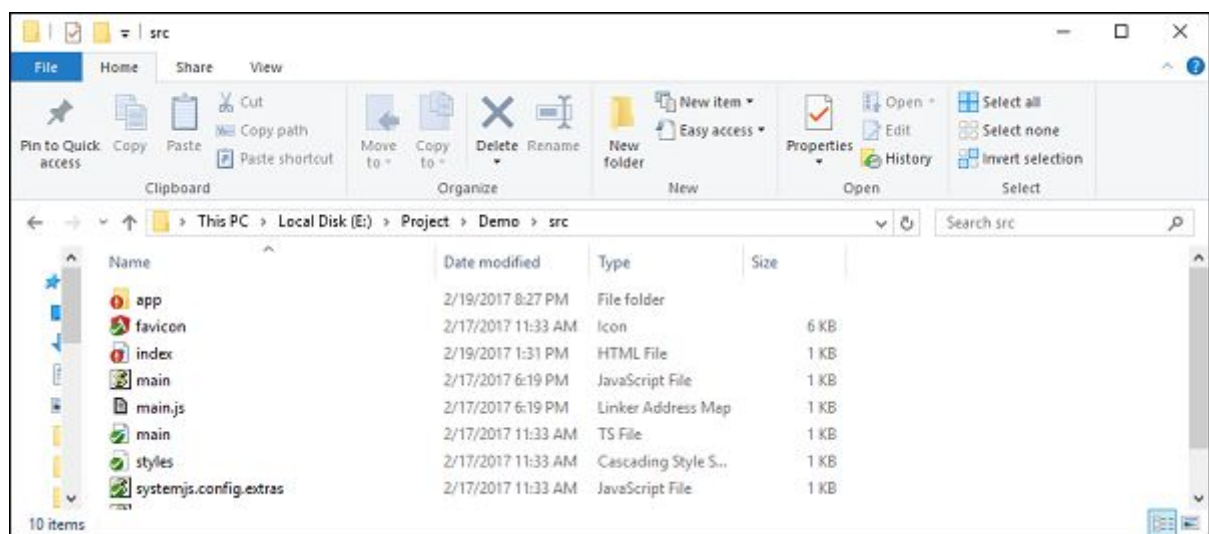


Paso 3 - Ir a la carpeta de proyecto Angular JS en el Explorador de Windows.

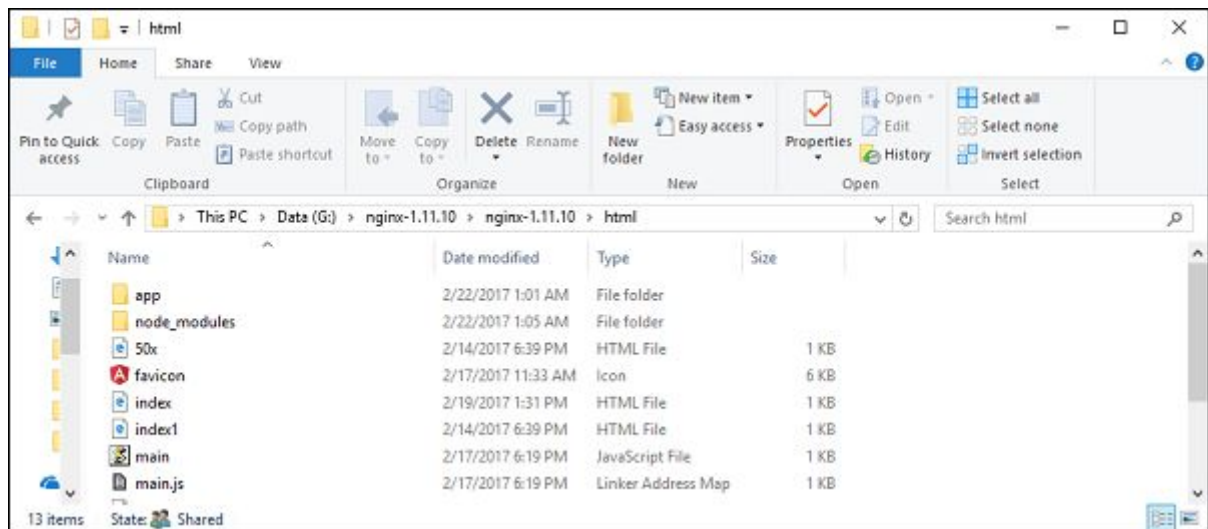
Paso 4 - Copie la carpeta Project → Demo → node-modules.



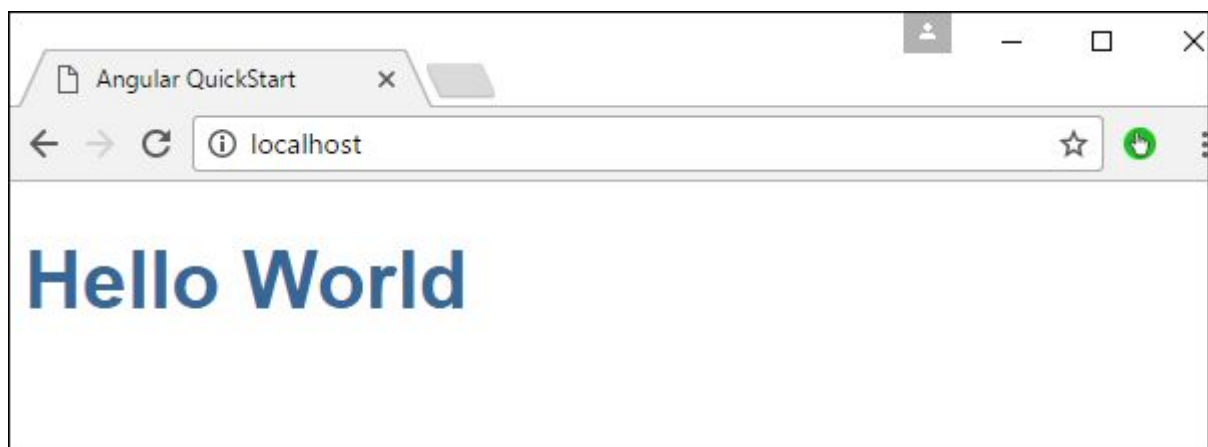
Paso 5 - Copie todo el contenido de la carpeta Proyecto → Demo → src



Paso 6 - Copie todo el contenido en la carpeta nginx / html.



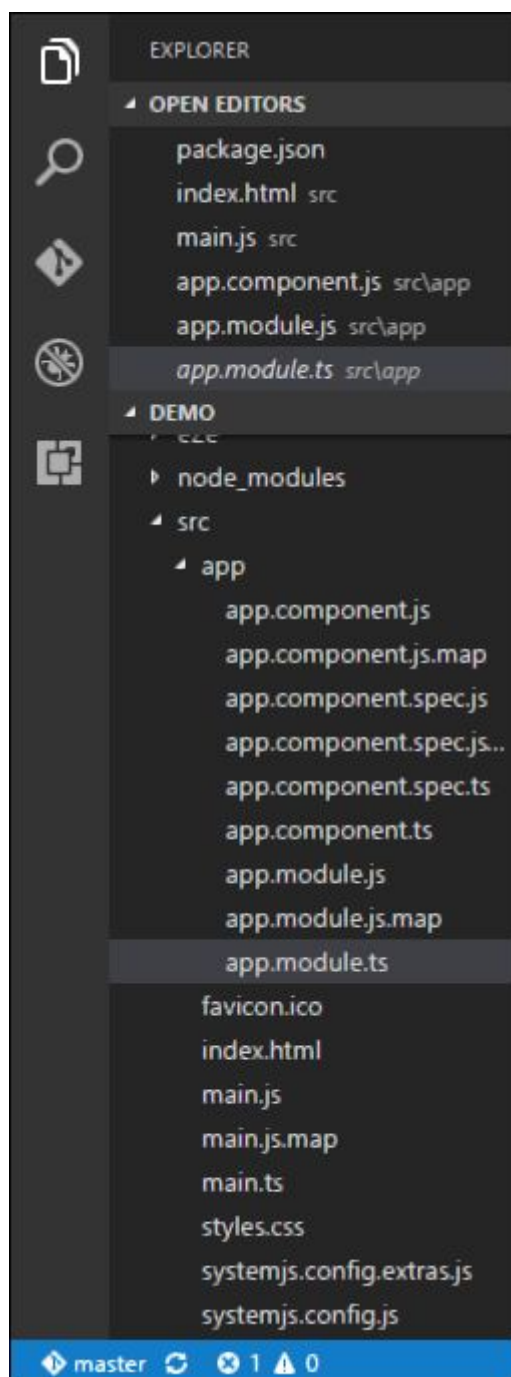
Now go to the URL – **http://localhost**, you will actually see the hello world application as shown in the following screenshot.



## Módulos

Los módulos se utilizan en Angular JS para poner límites lógicos en su aplicación. Por lo tanto, en lugar de codificar todo en una aplicación, en lugar de ello puede construir todo en módulos independientes para separar la funcionalidad de su aplicación. Vamos a inspeccionar el código que se agrega a la aplicación de demostración.

En el código de Visual Studio, vaya a la carpeta `app.module.ts` en la carpeta de su aplicación. Esto se conoce como la clase del módulo raíz.



El siguiente código estará presente en el archivo app.module.ts.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
@NgModule({
  imports: [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
```

```
}}  
export class AppModule {}
```

Vamos a pasar por cada línea del código en detalle.

La instrucción `import` se utiliza para importar la funcionalidad de los módulos existentes. Por lo tanto, las primeras 3 instrucciones se utilizan para importar los módulos **NgModule**, **BrowserModule** y `bootstrap` en este módulo.

El decorador **NgModule** se utiliza para posteriormente definir las importaciones, las declaraciones y las opciones de bootstrapping.

El **BrowserModule** es requerido por defecto para cualquier aplicación angular basada en web.

La opción `bootstrap` le indica a Angular qué componente debe arrancar en la aplicación.

Un módulo se compone de las siguientes partes:

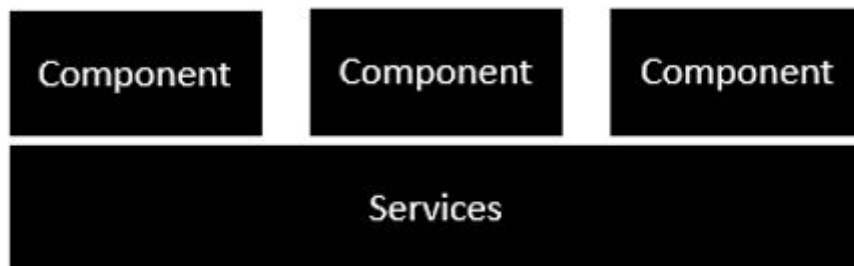
**Matriz de arranque** - Se utiliza para indicar a Angular JS qué componentes deben cargarse para poder acceder a su funcionalidad en la aplicación. Una vez que haya incluido el componente en la matriz de arranque, deberá declararlos para que puedan utilizarse en otros componentes de la aplicación Angular JS.

**Exportar matriz** - Se utiliza para exportar componentes, directivas y tuberías que luego se pueden utilizar en otros módulos.

**Matriz de importación** - Al igual que la matriz de exportación, la matriz de importación se puede utilizar para importar la funcionalidad de otros módulos Angular JS.

## Arquitectura

The following screenshot shows the anatomy of an Angular 2 application. Each application consists of Components. Each component is a logical boundary of functionality for the application. You need to have layered services, which are used to share the functionality across components.

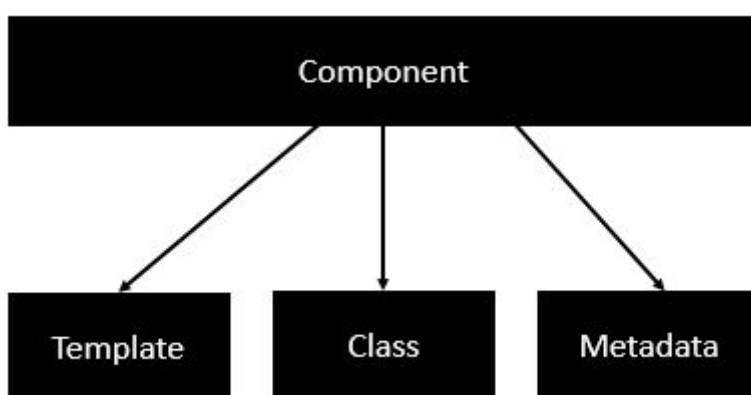


A continuación se muestra la anatomía de un Componente. Un componente consta de -

**Clase:** esto es como una clase C o Java que consiste en propiedades y métodos.

**Metadatos:** se usa para decorar la clase y extender la funcionalidad de la clase.

**Plantilla:** se utiliza para definir la vista HTML que se muestra en la aplicación.



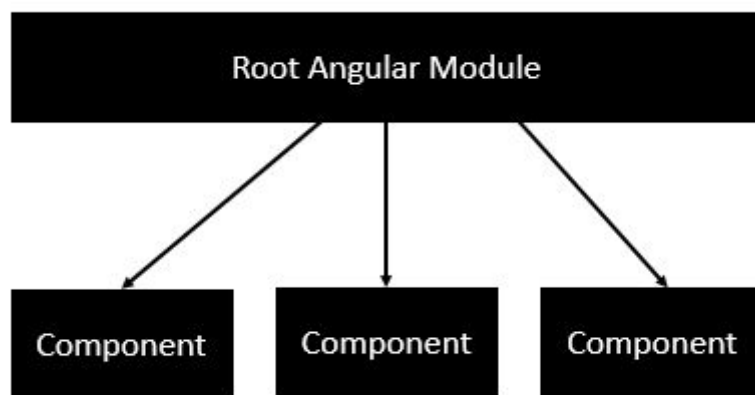
A continuación se muestra un ejemplo de un componente

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

@Component({
  selector: 'my-app',
  templateUrl: 'app/app.component.html'
})

export class AppComponent {
  appTitle: string = 'Welcome';
}
```

Cada aplicación está formada por módulos. Cada aplicación Angular 2 necesita tener un módulo de raíz angular. Cada módulo de raíz angular puede tener varios componentes para separar la funcionalidad.



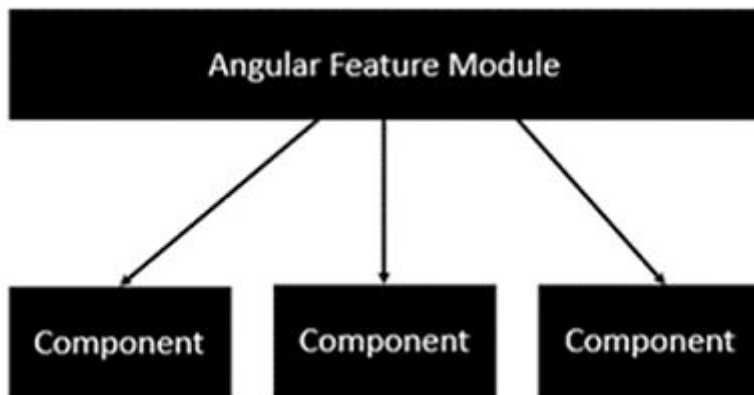
A continuación se muestra un ejemplo de un módulo raíz.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  imports: [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule {}
```

Cada aplicación está compuesta de módulos de características donde cada módulo tiene una característica independiente de la aplicación. Cada módulo de características angulares puede tener varios componentes para separar la funcionalidad.





## Componentes

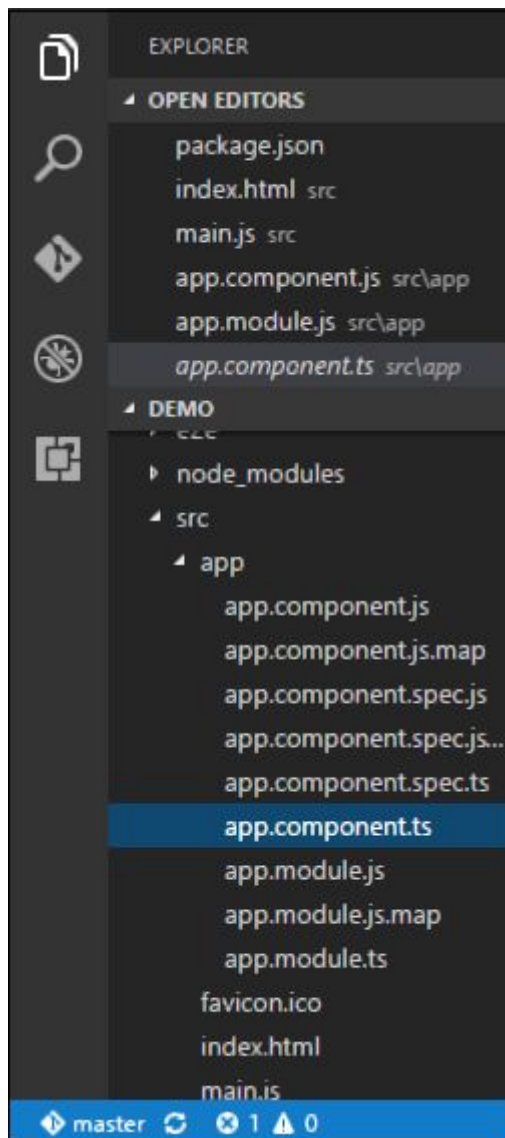
Los componentes son una pieza lógica de código para la aplicación de JS Angular. Un componente consta de lo siguiente:

**Plantilla:** se utiliza para representar la vista de la aplicación. Esto contiene el HTML que se debe procesar en la aplicación. Esta parte también incluye la vinculación y las directivas.

**Clase** - Esto es como una clase definida en cualquier lenguaje como C. Esto contiene propiedades y métodos. Esto tiene el código que se utiliza para apoyar la vista. Se define en TypeScript.

**Metadatos** - Esto tiene los datos adicionales definidos para la clase Angular. Se define con un decorador.

Ahora vayamos al archivo `app.component.ts` y creemos nuestro primer componente Angular.



Vamos a agregar el siguiente código al archivo y mirar cada aspecto en detalle.

### Clase

El decorador de clase. La clase se define en TypeScript. Normalmente, la clase tiene la siguiente sintaxis en TypeScript.

### Sintaxis

```
class classname {  
  Propertyname: PropertyType = Value  
}
```

Parámetros

**Classname** - Este es el nombre que se debe dar a la clase.

**Nombre de la propiedad** - Este es el nombre que debe darse a la propiedad.

**PropertyType** - Dado que TypeScript está fuertemente mecanografiado, debe dar un tipo a la propiedad.

**Valor** - Este es el valor que debe darse a la propiedad.

Ejemplo

```
export class AppComponent {  
  appTitle: string = 'Welcome';  
}
```

En el ejemplo, hay que señalar las siguientes cosas:

Estamos definiendo una clase llamada AppComponent.

La palabra clave de exportación se utiliza para que el componente se pueda utilizar en otros módulos en la aplicación Angular JS.

AppTitle es el nombre de la propiedad.

La propiedad se da el tipo de cadena.

La propiedad tiene un valor de 'Bienvenida'.

Modelo

Esta es la opinión que debe ser presentada en la solicitud.

Sintaxis

```
Template: '  
<HTML code>
```

```
class properties
```

## Parámetros

Código HTML: este es el código HTML que se debe procesar en la aplicación.

Propiedades de la clase: son las propiedades de la clase a las que se puede hacer referencia en la plantilla.

## Ejemplo

```
template: `
<div>
  <h1>{{appTitle}}</h1>
  <div>To Tutorials Point</div>
</div>
`
```

En el ejemplo, hay que señalar las siguientes cosas:

Estamos definiendo el código HTML que se traducirá en nuestra aplicación

También hacemos referencia a la propiedad `appTitle` de nuestra clase.

## Metadatos

Esto se utiliza para decorar la clase Angular JS con información adicional.

Echemos un vistazo al código completo con nuestra clase, plantilla y metadatos.

## Ejemplo

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

@Component({
  selector: 'my-app',
  template: ` <div>
    <h1>{{appTitle}}</h1>
    <div>To Tutorials Point</div>
  </div> `,
})

export class AppComponent {
  appTitle: string = 'Welcome';
}
```

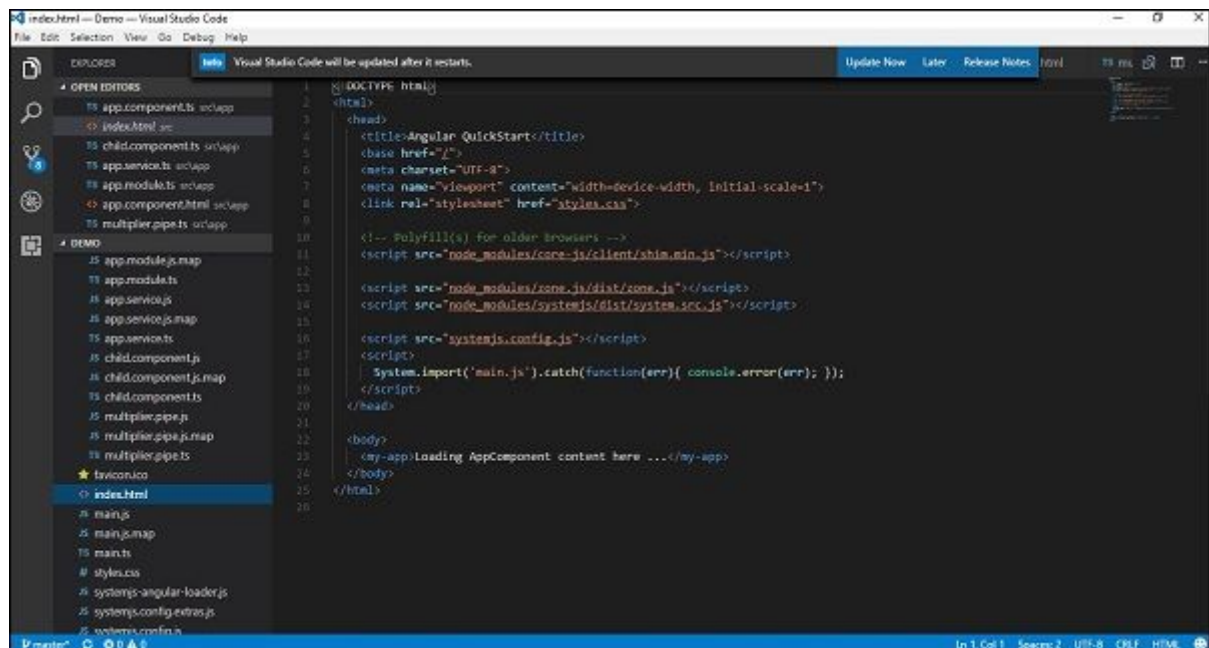
En el ejemplo anterior, se deben tener en cuenta las siguientes cosas:

Estamos usando la palabra clave `import` para importar el decorador `'Component'` del módulo `angular / core`.

A continuación, estamos utilizando el decorador para definir un componente.

El componente tiene un selector llamado `'my-app'`. Esto no es nada pero nuestra etiqueta de html de encargo que se puede utilizar en nuestra página principal del html.

Ahora, vayamos a nuestro archivo `index.html` en nuestro código.



Asegúrese de que la etiqueta de cuerpo contiene ahora una referencia a nuestra etiqueta personalizada en el componente. Por lo tanto, en el caso anterior, debemos asegurarnos de que la etiqueta `body` contenga el siguiente código

```
<body>
  <my-app></my-app>
</body>
```

Salida

Ahora, si vamos al navegador y vemos la salida, veremos que la salida se procesa tal como está en el componente.



## Plantillas

En el capítulo de Componentes, ya hemos visto un ejemplo de la siguiente plantilla.

```
template: `
  <div>
    <h1>{{appTitle}}</h1>
    <div>To Tutorial Point</div>
  </div>
`
```

Esto se conoce como una plantilla en línea. Hay otras maneras de definir una plantilla y que se puede hacer a través del comando `templateURL`. La forma más sencilla de utilizar esto en el componente es la siguiente.

### Sintaxis

```
templateURL:
  viewname.component.html
```

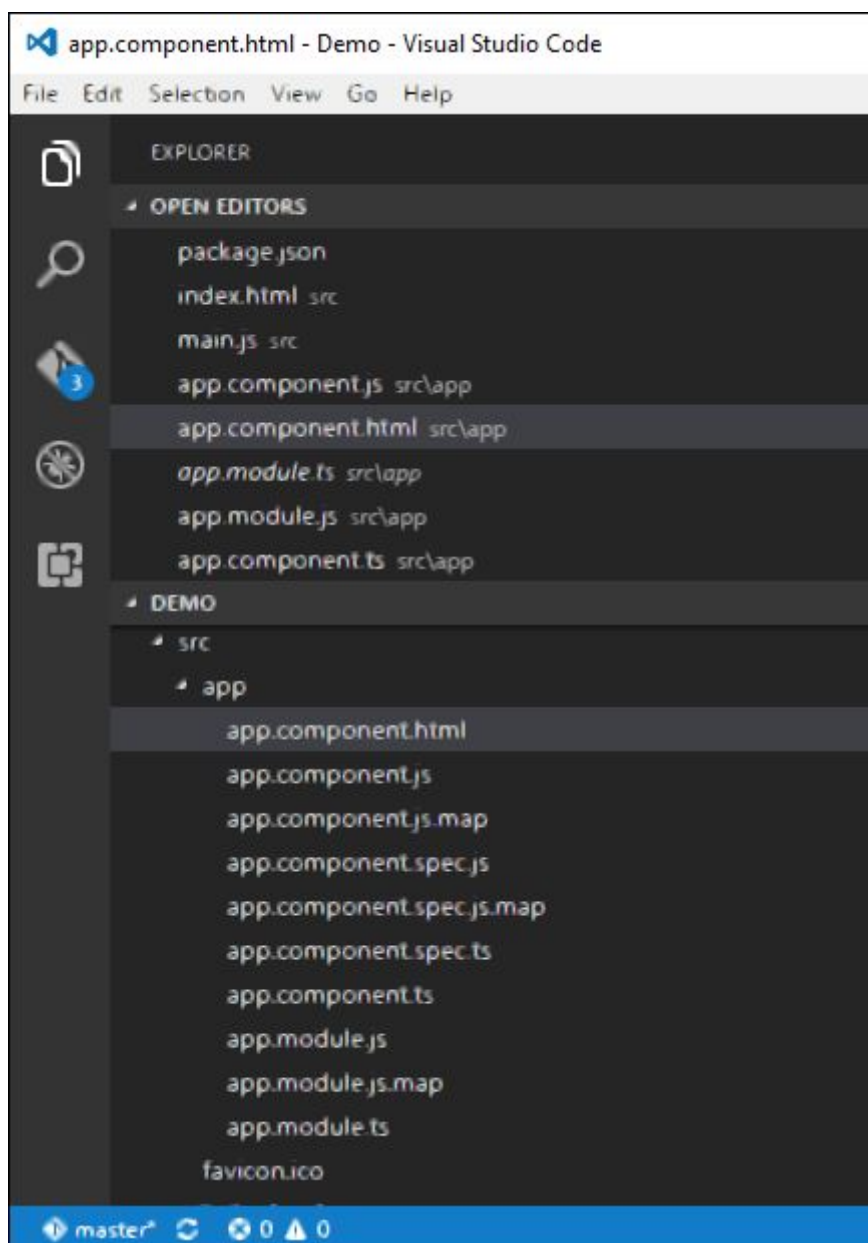
## Parámetros

Viewname - Este es el nombre del módulo componente de la aplicación.

Después del nombre de la vista, el componente debe agregarse al nombre del archivo.

A continuación se indican los pasos para definir una plantilla en línea.

Paso 1: cree un archivo llamado app.component.html. Esto contendrá el código html para la vista.



Paso 2: agregue el código siguiente en el archivo creado anteriormente.

```
<div>{{appTitle}} Tutorialspoint </div>
```

Esto define una etiqueta div sencilla y hace referencia a la propiedad appTitle de la clase app.component.

Paso 3 - En el archivo app.component.ts, agregue el código siguiente.

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

@Component ({
  selector: 'my-app',
  templateUrl: 'app/app.component.html'
})

export class AppComponent {
  appTitle: string = 'Welcome';
}
```

Desde el código anterior, el único cambio que puede observarse es el de la templateUrl, que da el enlace al archivo app.component.html que se encuentra en la carpeta de la aplicación.

Paso 4 - Ejecutar el código en el navegador, obtendrá la salida siguiente.



## Directivas



Una directiva es un elemento HTML personalizado que se utiliza para ampliar el poder del HTML. Angular 2 tiene las siguientes directivas que se llaman como parte del módulo BrowserModule.

## Ngif

## NgFor

Si ve el archivo app.module.ts, verá el código siguiente y el módulo **BrowserModule** definido. Al definir este módulo, tendrá acceso a las 2 directivas.

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent }  from './app.component';

@NgModule ({
  imports:      [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap:   [ AppComponent ]
})
export class AppModule { }
```

Ahora veamos cada directiva en detalle.

## NgIf

El elemento ngif se utiliza para agregar elementos al código HTML si se evalúa como verdadero, de lo contrario no agregará los elementos al código HTML.

## Sintaxis

```
*ngIf = 'expression'
```

Si la expresión se evalúa como verdadera, entonces se agrega el correspondiente, de lo contrario los elementos no se agregan.

Veamos ahora un ejemplo de cómo podemos usar la directiva \* ngif.

Paso 1: primero agregue una propiedad a la clase llamada appStatus. Este será de tipo booleano. Mantengamos este valor como verdadero.

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

@Component ({
```

```

    selector: 'my-app',
    templateUrl: 'app/app.component.html'
  })

export class AppComponent {
  appTitle: string = 'Welcome';
  appStatus: boolean = true;
}

```

Paso 2 - Ahora en el archivo app.component.html, agregue el código siguiente.

```
<div *ngIf = 'appStatus'>{{appTitle}} Tutorialspoint </div>
```

En el código anterior, ahora tenemos la directiva \*ngIf. En la directiva estamos evaluando el valor de la propiedad appStatus. Dado que el valor de la propiedad debe evaluarse como true, significa que la etiqueta div debe mostrarse en el navegador.

Una vez que agregamos el código anterior, obtendremos la siguiente salida en el navegador.

Salida



## NgFor

El elemento ngFor se utiliza para elementos basados en la condición del bucle For.

Sintaxis

```
*ngFor = 'let variable of variablelist'
```

La variable es una variable temporal para mostrar los valores en la variablelist.

Veamos ahora un ejemplo de cómo podemos usar la directiva \* ngFor.

Paso 1: primero agregue una propiedad a la clase llamada appList. Este será del tipo que se puede utilizar para definir cualquier tipo de matrices.

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

@Component ({
  selector: 'my-app',
  templateUrl: 'app/app.component.html'
})

export class AppComponent {
  appTitle: string = 'Welcome';
  appList: any[] = [ {
    "ID": "1",
    "Name" : "One"
  },
  {
    "ID": "2",
    "Name" : "Two"
  } ];
}
```

Por lo tanto, estamos definiendo la appList como una matriz que tiene 2 elementos. Cada elemento tiene 2 propiedades secundarias como ID y Nombre.

Paso 2 - En app.component.html, defina el siguiente código.

```
<div *ngFor = 'let lst of appList'>
  <ul>
    <li>{{lst.ID}}</li>
    <li>{{lst.Name}}</li>
  </ul>
</div>
```

En el código anterior, ahora estamos utilizando la directiva ngFor para iterar a través de la matriz appList. A continuación, definimos una lista en la que cada elemento de lista es el ID y el parámetro name de la matriz.

Una vez que agregamos el código anterior, obtendremos la siguiente salida en el navegador.

Salida

