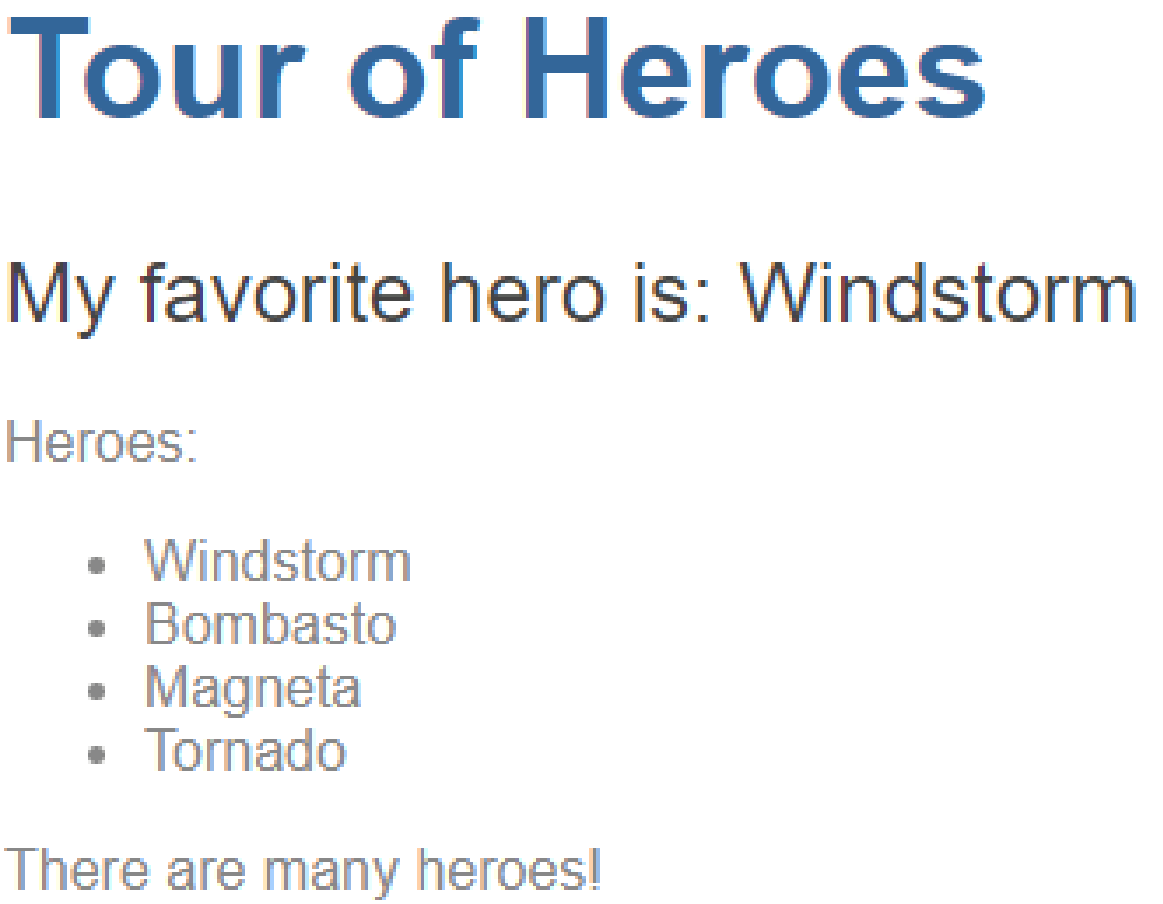
Mostrar datos

Angular.io

Puede mostrar datos vinculando controles en una plantilla HTML a las propiedades de un componente angular.

En esta página, creará un componente con una lista de héroes. Mostrará la lista de nombres de héroes y condicionalmente mostrará un mensaje debajo de la lista.

La interfaz de usuario final se ve así:



los ejemplo en vivo / ejemplo de descarga demuestra todos los fragmentos de código y sintaxis descritos en esta página.

Mostrar propiedades de componentes con interpolación

La forma más fácil de mostrar una propiedad de componente es vincular el nombre de la propiedad mediante interpolación. En la interpolación, se pone el nombre de la propiedad en la plantilla de vista, entre llaves dobles rizadas: {{myHero}}.

Use el comando CLI ng new displaying-datapara crear un espacio de trabajo y una aplicación llamada displaying-data.

Eliminar el app.component.html archivo. No es necesario para este ejemplo.

Luego modifique el app.component.ts archivo cambiando la plantilla y el cuerpo del componente.

Cuando haya terminado, debería verse así:

src / app / app.component.ts

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



@Component({

  selector: 'app-root',

  template: `

    <h1>{{title}}</h1>

    <h2>My favorite hero is:
{{myHero}}</h2>

    `

})
```

```
export class AppComponent {

  title = 'Tour of Heroes';

  myHero = 'Windstorm';

}
```

Agregó dos propiedades al componente anteriormente vacío: titley myHero.

La plantilla muestra las propiedades de dos componentes mediante la interpolación de llaves dobles:

src / app / app.component.ts (plantilla)

template: `

<h1>{{title}}</h1>

<h2>My favorite hero is: {{myHero}}</h2>

`

La plantilla es una cadena de varias líneas dentro de los backticks de ECMAScript 2015 (`). El backtick ( `), que *no* es el mismo carácter que una comilla simple ('), le permite componer una cadena en varias líneas, lo que hace que el HTML sea más legible.

Angular extrae automáticamente el valor de las propiedades titley myHerodel componente e inserta esos valores en el navegador.

Angular actualiza la pantalla cuando cambian estas propiedades.

> **Más precisamente, la nueva visualización se produce después de algún tipo de evento asincrónico relacionado con la vista, como una pulsación de tecla, una finalización del temporizador o una respuesta a una solicitud HTTP.**

Tenga en cuenta que no llama a new para crear una instancia de la AppComponentclase. Angular está creando una instancia para ti. ¿Cómo?

El CSS selectoren el decorador @Componentespecifica un elemento llamado <app-root> . Ese elemento es un marcador de posición en el cuerpo de su archivo: index.html
src / index.html (body)

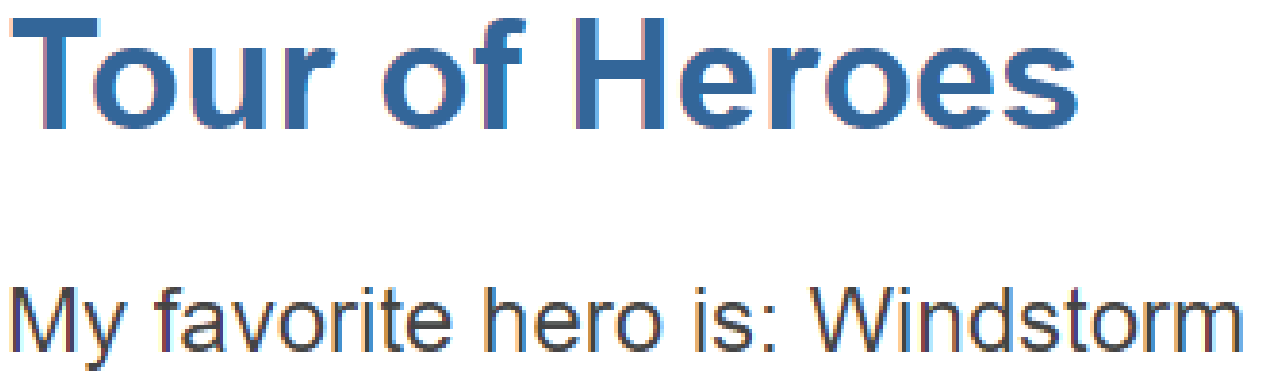
```
<body>

  <app-root></app-root>

</body>
```

Cuando arranca con la AppComponentclase (in main.ts), Angular busca un <app-root> en el index.html, lo encuentra, crea una instancia de AppComponenty lo muestra dentro de la <app-root>etiqueta.

Ahora ejecuta la aplicación. Debe mostrar el título y el nombre del héroe:



Las siguientes secciones revisan algunas de las opciones de codificación en la aplicación.

¿Template en línea o template de plantilla?

Puede almacenar la plantilla de su componente en uno de dos lugares. Puede definirlo en *línea* usando la templatepropiedad, o puede definir la plantilla en un archivo HTML separado y vincularlo en los component metadata using the `@Component` decorator's `templateUrl` property

La elección entre HTML en línea y por separado es una cuestión de gusto, circunstancias y política de organización. Aquí la aplicación usa HTML en línea porque la plantilla es pequeña y la demostración es más simple sin el archivo HTML adicional.

En cualquier estilo, los enlaces de datos de plantilla tienen el mismo acceso a las propiedades del componente.

De manera predeterminada, el comando Angular CLI ng generate componentgenera componentes con un archivo de plantilla. Puede anular eso con:

```
ng generate component hero -it
```

¿Constructor o inicialización variable?

Aunque este ejemplo utiliza la asignación de variables para inicializar los componentes, en su lugar podría declarar e inicializar las propiedades utilizando un constructor:

```
export class AppComponent {
    title: string;
    myHero: string;

    constructor() {
      this.title = 'Tour of Heroes';
      this.myHero = 'Windstorm';
    }
  }
```

Esta aplicación utiliza un estilo de "asignación variable" más conciso simplemente por brevedad.

Mostrar una propiedad de matriz con el * ngFor

Para mostrar una lista de héroes, comience agregando una matriz de nombres de héroes al componente y redefina myHeropara ser el primer nombre en la matriz.
src / app / app.component.ts (clase)

```
export class AppComponent {

    title = 'Tour of Heroes';

    heroes = ['Windstorm',
  'Bombasto', 'Magneta', 'Tornado'];

    myHero = this.heroes[0];

  }
```

Ahora use la ngFor directiva Angular en la plantilla para mostrar cada elemento en la heroes lista.

src / app / app.component.ts (plantilla)

```
template: `

    <h1>{{title}}</h1>

    <h2>My favorite hero is:
  {{myHero}}</h2>

    <p>Heroes:</p>

    <ul>

      <li *ngFor="let hero of heroes">

        {{ hero }}

      </li>

    </ul>

  `
```

**Esta interfaz de usuario usa la lista desordenada de HTML con y etiquetas. The `*ngFor` in the `<li>` element is the Angular "repeater" directive. It marks that `<li>` element (and its children) as the "repeater template"(El en el elemento es la directiva angular "repetidor". Marca ese elemento (y sus elementos secundarios) como la "plantilla repetidora":*ngFor
**

src / app / app.component.ts (li))

```
<li *ngFor="let hero of heroes">

  {{ hero }}

</li>
```

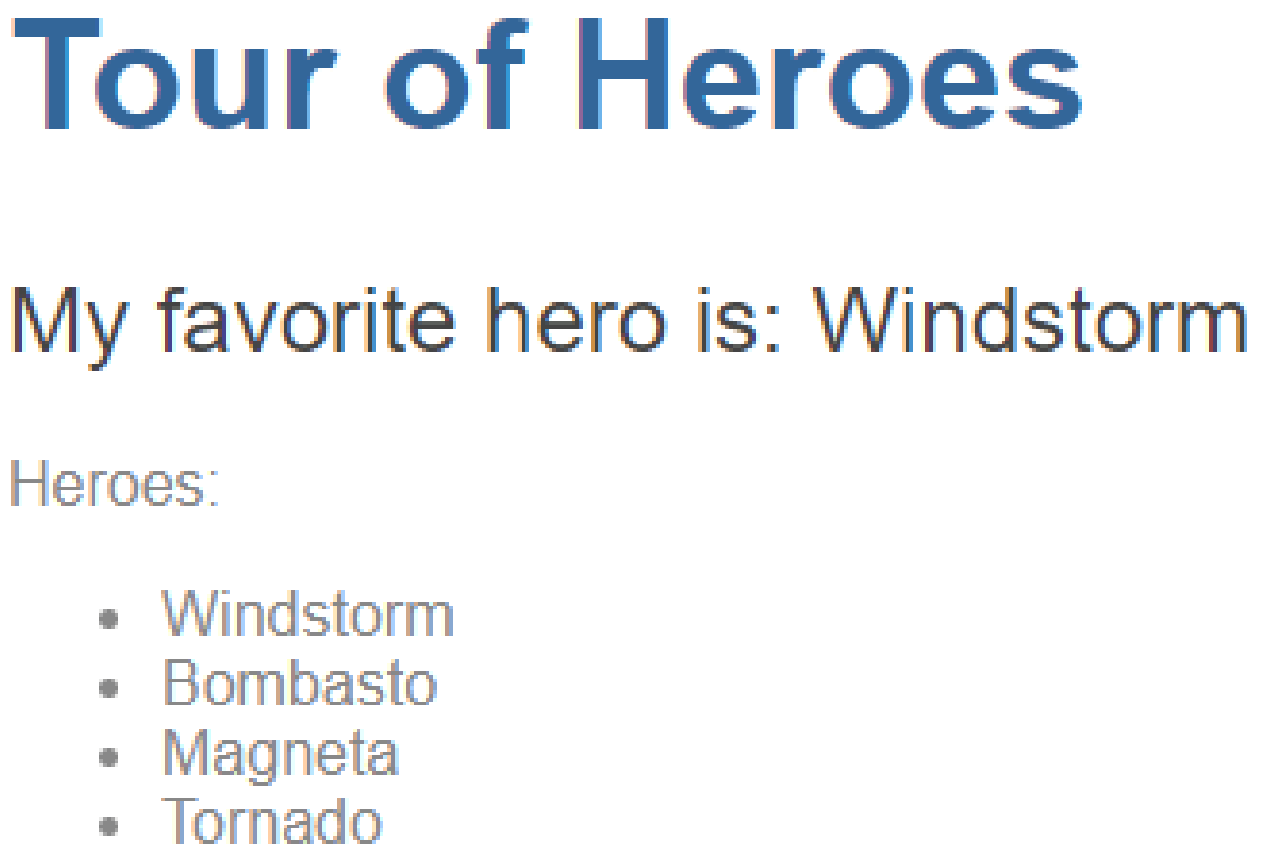
No olvide el asterisco principal (*) en . Es una parte esencial de la sintaxis. Para obtener más información, consulte la página Sintaxis de plantilla .*ngFor

Observe el heroen la ngForinstrucción entre comillas dobles; Es un ejemplo de una variable de entrada de plantilla. Lea más sobre las variables de entrada de plantilla en la sección de microsintaxis de la página Sintaxis de plantilla .

Angular duplica el para cada elemento de la lista, configurando la herovariable para el elemento (el héroe) en la iteración actual. Angular usa esa variable como contexto para la interpolación en las llaves dobles.

En este caso, ngFormuestra una matriz, pero ngForpuede repetir elementos para cualquier objeto iterable .

Ahora los héroes aparecen en una lista desordenada.



Crear una class para el datos

El código de la aplicación define los datos directamente dentro del componente, lo cual no es la mejor práctica. En una demostración simple, sin embargo, está bien.

Por el momento, la unión es a una serie de cadenas. En aplicaciones reales, la mayoría de los enlaces son a objetos más especializados.

Para convertir este enlace para usar objetos especializados, convierta la matriz de nombres de héroes en una matriz de Heroobjetos. Para eso necesitarás una Heroclase:

```
ng generate class hero
```

Con el siguiente código:
src / app / hero.ts

```
export class Hero {

  constructor(

    public id: number,

    public name: string) { }

}
```

Ha definido una clase con un constructor y dos propiedades: idy name.

Puede que no parezca que la clase tiene propiedades, pero las tiene. La declaración de los parámetros del constructor aprovecha un acceso directo de TypeScript.

Considere el primer parámetro:

src / app / hero.ts (id)

```
    public id: number,
```

Esa breve sintaxis hace mucho:

- **Declara un parámetro constructor y su tipo.**
- **Declara una propiedad pública del mismo nombre.**
- **Inicializa esa propiedad con el argumento correspondiente al crear una instancia de la clase.**

Usando el class Hero

Después de importar la Heroclase, la AppComponent.heroes propiedad puede devolver un *escrito* conjunto de Heroobjetos:

src / app / app.component.ts (héroes)

```
heroes = [
      new Hero(1, 'Windstorm'),
      new Hero(13, 'Bombasto'),
      new Hero(15, 'Magneta'),
      new Hero(20, 'Tornado')
    ];
    myHero = this.heroes[0];
```

A continuación, actualice la plantilla. Por el momento muestra el
héroe idy name. Arregla eso para mostrar solo la namepropiedad del héroe .

src / app / app.component.ts (template)

```
content_copytemplate: `
  <h1>{{title}}</h1>
  <h2>My favorite hero is:
{{myHero.name}}</h2>
  <p>Heroes:</p>
  <ul>
    <li *ngFor="let hero of heroes">
      {{ hero.name }}
    </li>
  </ul>
`
```

La pantalla se ve igual, pero el código es más claro.

Visualización condicional con NgIf

A veces, una aplicación necesita mostrar una vista o una parte de una vista solo en circunstancias específicas.

Cambiemos el ejemplo para mostrar un mensaje si hay más de tres héroes.

La ngIfdirectiva angular inserta o elimina un elemento basado en una *condición de verdad / falsedad* . Para verlo en acción, agregue el siguiente párrafo en la parte inferior de la plantilla:
src / app / app.component.ts (mensaje)

```
<p *ngIf="heroes.length > 3">There are many heroes!</p>
```

No olvide el asterisco principal (*) en *ngIf. Es una parte esencial de la sintaxis. Lea más sobre y en la sección ngIf de la página Sintaxis de plantilla .

La expresión de plantilla dentro de las comillas dobles , .*ngIf="heroes.length > 3", se ve y se comporta de manera muy similar a TypeScript. Cuando la lista de héroes del componente tiene más de tres elementos, Angular agrega el párrafo al DOM y aparece el mensaje. Si hay tres elementos o menos, Angular omite el párrafo, por lo que no aparece ningún mensaje. Para obtener más información, consulte la sección de expresiones de plantilla de la página Sintaxis de plantilla

Angular no muestra ni oculta el mensaje. Está agregando y eliminando el elemento de párrafo del DOM. Eso mejora el rendimiento, especialmente en proyectos más grandes cuando se incluyen o excluyen condicionalmente grandes fragmentos de HTML con muchos enlaces de datos.

Pruébalo. Como la matriz tiene cuatro elementos, el mensaje debería aparecer. Vuelve app.component.ts y elimina o comenta uno de los elementos de la matriz de héroes. El navegador debería actualizarse automáticamente y el mensaje debería desaparecer.

resumen

Ahora ya sabes cómo usar:

- **Interpolación con llaves dobles para mostrar una propiedad de componente.**
- **ngFor para mostrar una variedad de elementos.**
- **Una clase TypeScript para dar forma a los datos del modelo para su componente y mostrar las propiedades de ese modelo.**
- **ngIf para mostrar condicionalmente un fragmento de HTML basado en una expresión booleana.**

Aquí está el código final:

src / app / app.component.ts

src / app / hero.ts

src / app / app.module.ts

main.ts

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


import { Hero } from './hero';
```

```
@Component({
  selector: 'app-root',
  template: `
  <h1>{{title}}</h1>
  <h2>My favorite hero is:
{{myHero.name}}</h2>
  <p>Heroes:</p>
  <ul>
    <li *ngFor="let hero of heroes">
      {{ hero.name }}
      </li>
  </ul>
```

```
  <p *ngIf="heroes.length >
3">There are many heroes!</p>

`

})

export class AppComponent {

  title = 'Tour of Heroes';

  heroes = [

    new Hero(1, 'Windstorm'),

    new Hero(13, 'Bombasto'),

    new Hero(15, 'Magneta'),

    new Hero(20, 'Tornado')

  ];

  myHero = this.heroes[0];
```

```
}
```