

**TRABAJO PRÁCTICO N°6:**

**Requerimientos ágiles - Implementación de user stories**

**Cátedra:** Ingeniería de Software.

**Ciclo lectivo:** 2022.

**Grupo:** 4.

**Fecha de entrega:** 26/04/2022.

**Profesores:** Covaro, Laura Ines (Titular)

Boiero Rovera, Gerardo Javier (JTP)

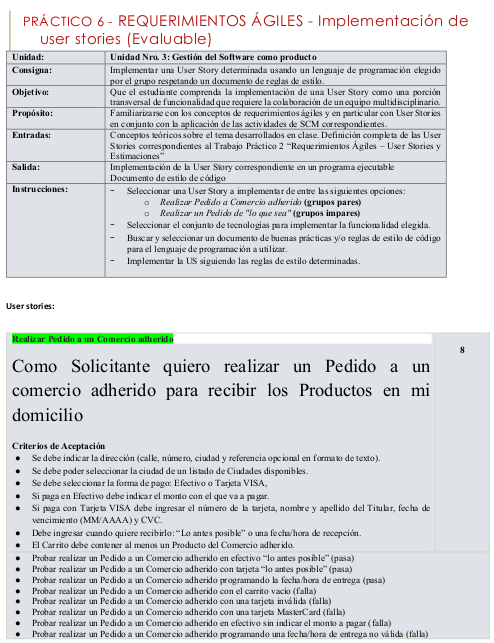
Crespo, María Mickaela (Ayudante 1ra)

**Integrantes:**

* 48701 - Basso Martín Federico
* 61539 - Cruz Karen Yanina
* 54910 - Gómez Iván Alejandro

**PRÁCTICO 6 - REQUERIMIENTOS ÁGILES - Implementación de**

**user stories (Evaluable)**



La implementación se llevó a cabo bajo el framework Angular, utilizando la librería de Ionic.

Link del repositorio:

<https://github.com/ivangomez854/isw-tp6-app.git>

Guía de estilo de la documentación de Angular

Esta guía de estilo cubre los estándares para escribir [documentación angular en angular.io](https://runebook.dev/es/docs/angular/guide/docs) . Estos estándares garantizan la coherencia en el estilo de escritura, las convenciones de Markdown y los fragmentos de código.

## **Prerrequisitos**

Antes de contribuir a la documentación de Angular, es útil que estés familiarizado con lo siguiente:

* git : para una introducción, consulte el [Manual de Git](https://guides.github.com/introduction/git-handbook/) de GitHub
* GitHub : para una introducción, consulte [Hello World de](https://guides.github.com/activities/hello-world/) GitHub
* Markdown: consulta el [Mastering Markdown de](https://guides.github.com/features/mastering-markdown/) GitHub
* Estilo de codificación angular: consulte la [Guía de estilo angular](https://runebook.dev/es/docs/angular/guide/styleguide)
* Estilo de escritura de Google: [la Guía de estilo de documentación para desarrolladores de Google](https://developers.google.com/style/) es un recurso completo sobre el que se basa esta guía de estilo de documentación angular.

## **Tipos de documentación de Angular**

Las categorías de la documentación de Angular incluyen:

* [Guías](https://runebook.dev/es/docs/angular/guide/docs) : gran parte de lo que hay en la [sección de documentación de angular.io](https://runebook.dev/es/docs/angular/guide/docs) . Las guías guían al lector paso a paso a través de las tareas para demostrar conceptos y, a menudo, van acompañadas de un ejemplo práctico. Estos incluyen [Getting Started](https://runebook.dev/es/docs/angular/guide/start) , [Tour of Heroes](https://runebook.dev/es/docs/angular/guide/tutorial) y páginas sobre [Formularios](https://runebook.dev/es/docs/angular/guide/forms-overview) , [Inyección de dependencias](https://runebook.dev/es/docs/angular/guide/dependency-injection) y [HttpClient](https://runebook.dev/es/docs/angular/guide/http) . Los miembros contribuyentes de la comunidad y los miembros del equipo de Angular mantienen esta documentación en [Markdown](https://daringfireball.net/projects/markdown/syntax) .
* [Documentación de la API](https://runebook.dev/es/docs/angular/guide/api) : documentos de referencia para la [interfaz de programación de aplicaciones angular o API](https://runebook.dev/es/docs/angular/guide/api) . Estos son más concisos que las guías y sirven como referencia para las características angulares. Son especialmente útiles para las personas que ya están familiarizadas con los conceptos de Angular. La infraestructura de [angular.io](https://angular.io/) genera estos documentos a partir del código fuente y los comentarios que editan los colaboradores.
* [Documentación CLI](https://runebook.dev/es/docs/angular/guide/cli) : la infraestructura [angular.io](https://angular.io/) genera estos documentos a partir del código fuente CLI.

## **Maquetado de HTML**

Si bien las guías angulares son archivos [Markdown](https://daringfireball.net/projects/markdown/syntax) , hay algunas secciones dentro de las guías que usan HTML.

Para habilitar HTML en una guía angular, **siempre** siga cada etiqueta HTML de apertura y cierre con una línea en blanco.

Observe la línea en blanco requerida después de la apertura <div> en el siguiente ejemplo:

<div class="alert is-helpful">

\*\*Always\*\* follow every opening and closing HTML tag with \_a blank line\_.

</div>

Es habitual,pero no obligatorio,preceder la etiqueta HTML de cierre con una línea en blanco también.

## **Titulo**

Todo documento de la guía debe tener un título, que debe aparecer en la parte superior de la página.

Comience el título con el carácter Markdown # , que se muestra como <h1> en el navegador.

# Guía de estilo de documentación angular

**Un documento solo puede tener un <h1> .**

El texto del título debe estar en *mayúsculas* y *minúsculas* , lo que significa que la primera palabra está en mayúscula y el resto de palabras en minúsculas. Los términos técnicos que siempre están en mayúscula, como "Angular", son la excepción.

# Política de obsolescencia en Angular

**Siempre siga el título con al menos una línea en blanco.**

El texto correspondiente en el panel de navegación izquierdo está en el *caso del título*, lo que significa que usa letras mayúsculas para comenzar las primeras palabras y todas las palabras principales. Utilice letras minúsculas para palabras secundarias como "en", "de" y "el". También puede acortar el título de navegación para que quepa en la columna.

## **Secciones**

Un documento típico tiene secciones.

Todos los títulos de las secciones están en *mayúsculas* y *minúsculas*, lo que significa que la primera palabra está en mayúscula y el resto de palabras en minúsculas.

**Siga siempre el encabezado de una sección con al menos una línea en blanco.**

### **Título de la sección principal**

Suele haber una o más secciones principales que pueden dividirse a su vez en secciones secundarias.

Comience un encabezado de sección principal con los caracteres Markdown ## , que se muestra como <h2> en el navegador.

Siga los títulos de las secciones principales con una línea en blanco y luego el contenido de ese título, como en el siguiente ejemplo:

## Main section heading

Content after a blank line.

### **Título de la sección secundaria**

Un epígrafe secundario está relacionado con un epígrafe principal y se encuentra textualmente dentro de los límites de ese epígrafe principal.

Comience un encabezado secundario con los caracteres Markdown ###, que se muestra como <h3> en el navegador.

Después de un epígrafe secundario hay una línea en blanco y luego el contenido de ese epígrafe, como en el siguiente ejemplo:

### Secondary section heading

Content after a blank line.

### **Títulos de secciones adicionales**

Si bien puede usar encabezados de sección adicionales, el generador de [tabla de contenido (TOC)](https://runebook.dev/es/docs/angular/guide/docs-style-guide#table-of-contents) solo muestra los encabezados <h2> y <h3> en el TOC a la derecha de la página.

#### El TOC no mostrará esto

Content after a blank line.

## **Tabla de contenido**

La mayoría de las páginas muestran una tabla de contenidos o TOC. El índice de contenidos aparece en el panel derecho cuando la ventana gráfica es amplia. Cuando es estrecha, la TOC aparece en una región plegable cerca de la parte superior de la página.

No es necesario que cree su propio TOC a mano porque el generador de TOC crea uno automáticamente a partir de los encabezados <h2> y <h3> de la página.

Para excluir un encabezado del TOC, cree el encabezado como un elemento <h2> o <h3> con una clase llamada 'no-toc'.

<h3 class="no-toc">

This heading is not displayed in the TOC

</h3>

Puede desactivar la generación de TOC para toda la página escribiendo el título con una etiqueta <h1> y la clase no-toc .

<h1 class="no-toc">

A guide without a TOC

</h1>

## **Navegación**

Para generar los enlaces de navegación en la parte superior, izquierda e inferior de la pantalla, use el archivo de configuración JSON, content/navigation.json .

Si tiene una idea que resultaría en cambios de navegación, [presente un problema](https://github.com/angular/angular/issues/new/choose) primero para que el equipo y la comunidad de Angular puedan discutir el cambio.

Para obtener una nueva página de guía, edite el nodo SideNav en navigation.json . El nodo SideNav es una matriz de nodos de navegación. Cada nodo es un nodo de artículo para un solo documento o un nodo de encabezado con nodos secundarios.

Encuentra el encabezado de tu página. Por ejemplo, una página de guía que describe una característica angular probablemente sea un elemento secundario del encabezado Fundamentals .

{

"title": "Fundamentals",

"tooltip": "The fundamentals of Angular",

"children": [ ... ]

}

Un nodo de cabecera hijo puede ser un nodo de artículo u otro nodo de cabecera.Si la página de la guía pertenece a una subcabecera,busque esa subcabecera en el JSON.

Añada un nodo de artículo para su página de guía como hijo del nodo de cabecera correspondiente,como en el siguiente ejemplo:

{

"url": "guide/docs-style-guide",

"title": "Doc authors style guide",

"tooltip": "Style guide for documentation authors.",

},

Un nodo de navegación tiene las siguientes propiedades:

* url : la URL de la página de la guía, que es solo un nodo de elemento.
* title : el texto que se muestra en el panel de navegación lateral.
* tooltip : texto que aparece cuando el lector se desplaza sobre el enlace de navegación.
* children : una matriz de nodos secundarios, que es solo un nodo de encabezado.
* hidden : definido y establecido como true si se trata de una página de guía que no debe mostrarse en el panel de navegación.

No cree un nodo que sea a la vez un encabezado y un nodo de elemento especificando la propiedad url de un nodo de encabezado.

## **Fragmentos de código**

[Angular.io](https://runebook.dev/es/docs/angular/guide/docs) tiene un marco personalizado que permite a los autores incluir fragmentos de código directamente desde aplicaciones de ejemplo de trabajo que se prueban automáticamente como parte de las compilaciones de documentación.

Además de fragmentos de código de trabajo,el código de ejemplo puede incluir comandos de terminal,un fragmento de TypeScript o HTML,o un archivo de código completo.

Cualquiera que sea la fuente, el visor de documentos los representa como fragmentos de código, ya sea individualmente con el componente de [ejemplo de código](https://runebook.dev/es/docs/angular/guide/docs-style-guide#code-example) o como una colección con pestañas con el componente de [pestañas de código](https://runebook.dev/es/docs/angular/guide/docs-style-guide#code-tabs) .

### **Cuándo utilizar la fuente de código**

Puede mostrar un mínimo fragmento de código en línea con la sintaxis de la marca Markdown. Utilice un solo signo de retroceso a cada lado de un término cuando se refiera al código o al nombre de un archivo en una frase.Los siguientes son algunos ejemplos:

* En app.component.ts , agregue un método logger() .
* El name propiedad es Sally .
* Agregue el nombre de la clase de componente a la matriz de declarations .

El Markdown es el siguiente:

\* In the `app.component.ts`, add a `logger()` method.

\* The <code class="no-auto-link">item</code> property is `true`.

\* Add the component class name to the `declarations` array.

### **Enlaces automáticos en fragmentos de código**

En ciertos casos,cuando se aplican retrocesos en torno a un término,puede que se establezca un autoenlace con la documentación del API.Si no pretende que el término sea un enlace,utilice la siguiente sintaxis:

The <code class="no-auto-link">item</code> property is `true`.

### **Hard-coded snippets**

Idealmente, debería obtener fragmentos de código de muestra de trabajo , aunque hay ocasiones en las que es necesario un fragmento en línea.

Para la entrada y salida del terminal, coloque el contenido entre las etiquetas <code-example> y establezca el atributo de idioma en sh como en este ejemplo:

npm start

<code-example language="sh">

npm start

</code-example>

Los fragmentos en línea codificados como éste no son comprobables y,por lo tanto,intrínsecamente poco fiables.Este ejemplo pertenece al pequeño conjunto de fragmentos en línea preaprobados que incluyen la entrada del usuario en un shell de comandos o la salida de algún proceso.

En todos los demás casos,los fragmentos de código deben generarse automáticamente a partir de muestras de código probadas.

Para obtener ejemplos hipotéticos, como ilustraciones de opciones de configuración en un archivo JSON, use la etiqueta <code-example> con el atributo de header para identificar el contexto.

### **Aplicaciones de ejemplo compilables**

Uno de los objetivos de diseño de la documentación de Angular es que los fragmentos de código de la página guía sean ejemplos de código en funcionamiento.

Los autores cumplen este objetivo mostrando fragmentos de código directamente de aplicaciones de ejemplo en funcionamiento,escritas específicamente para estas páginas de la guía.

Encuentre aplicaciones de muestra en subcarpetas del directorio de content/examples del repositorio angular/angular . El nombre de una carpeta de ejemplo suele ser el mismo que el de la página de guía que admite.

Una página guía puede no tener su propio código de muestra.Puede que se refiera en cambio a una muestra perteneciente a otra página.

El proceso de CI de Angular ejecuta todas las pruebas de extremo a extremo para cada PR de Angular.Angular vuelve a probar las muestras después de cada nueva versión de una muestra y cada nueva versión de Angular.

Cuando sea posible, cada fragmento de código en una página de guía debe derivarse de un archivo de muestra de código. Le indica al motor de documentación de Angular qué archivo de código, o fragmento de un archivo de código, mostrar configurando los atributos <code-example> .

### **Visualización de un archivo de código completo**

Esta guía de estilo de documentación de Angular que está leyendo actualmente tiene su propia aplicación de ejemplo, ubicada en la carpeta content/examples/docs-style-guide .

El siguiente <code-example> muestra el app.module.ts la muestra :

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

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({

imports: [ BrowserModule, FormsModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

El siguiente marcado produce ese fragmento:

<code-example path="docs-style-guide/src/app/app.module.ts" header="src/app/app.module.ts">

</code-example>

El atributo de path identifica el archivo de origen del fragmento en la ubicación de la carpeta de la aplicación de ejemplo dentro de content/examples . En este ejemplo, esa ruta es docs-style-guide/src/app/app.module.ts .

El encabezado le dice al lector dónde encontrar el archivo. Siguiendo la convención, establezca el atributo de header en la ubicación del archivo dentro de la carpeta raíz de la aplicación de ejemplo.

A menos que se indique lo contrario, todos los fragmentos de código en esta página son de código fuente de muestra ubicado en el directorio content/examples/docs-style-guide .

Las herramientas de documentación informan de un error si el archivo identificado en la ruta no existe o está en el [archivo](https://github.com/angular/angular/blob/master/aio/content/examples/.gitignore).git-ignore . La mayoría de los archivos .js están en .git-ignore .

Para incluir un archivo de código ignorado en su proyecto y mostrarlo en una guía, elimínelo de .git-ignore . Actualice el content/examples/.gitignore siguiente manera:

# my-guide

!my-guide/src/something.js

!my-guide/more-javascript\*.js

### **Visualización de parte de un archivo de código**

Para incluir un fragmento de código dentro de un archivo de código de muestra, en lugar de todo el archivo, use el atributo de region <code-example> . El siguiente ejemplo se centra en la clase AppModule y sus @[NgModule](https://runebook.dev/es/docs/angular/api/core/ngmodule)() @ NgModule () :

@NgModule({

imports: [ BrowserModule, FormsModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

Para representar el ejemplo anterior,el HTML del archivo Markdown es el siguiente:

<code-example

path="docs-style-guide/src/app/app.module.ts"

region="class">

</code-example>

La path apunta al archivo, al igual que en los ejemplos que representan el [archivo completo](https://runebook.dev/es/docs/angular/guide/docs-style-guide#display-whole-file) . El atributo de region especifica una parte del archivo fuente delimitada por una #docregion apertura y una #enddocregion de cierre .

Puede ver la class #docregion en el archivo fuente a continuación. Observe las líneas comentadas #docregion y #enddocregion en content/examples/docs-style-guide/src/app/app.module.ts con el nombre class .

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

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

// #docregion class

@NgModule({

imports: [ BrowserModule, FormsModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

// #enddocregion class

Las líneas #docregion inicial y final designan cualquier línea de código entre ellas como incluida en el fragmento de código. Es por eso que las declaraciones de importación fuera de la class #docregion no están en el fragmento de código.

Para obtener más información sobre cómo preparar archivos de aplicación de ejemplo para su uso en guías, consulte [Marcado de código fuente](https://runebook.dev/es/docs/angular/guide/docs-style-guide#source-code-markup) .

### **Opciones de fragmentos de código**

Especifique la salida <code-example> con los siguientes atributos:

* path : la ruta al archivo en la carpeta de content/examples .
* header : el encabezado de la lista de códigos. Este es el título del fragmento de código y puede incluir la ruta e información adicional, como si el fragmento es un fragmento.
* region : muestra el fragmento del archivo de origen con ese nombre de región; las regiones se identifican mediante el marcado #docregion en el archivo de origen. Consulte [Visualización de un fragmento de código](https://runebook.dev/es/docs/angular/guide/docs-style-guide#region) .
* linenums : el valor puede ser true , false o un number . El valor predeterminado es false , lo que significa que el navegador no muestra números de línea. La opción number comienza la numeración de líneas en el valor dado. Por ejemplo, linenums=4 establece el número de la línea inicial en 4.
* class : los fragmentos de código se pueden diseñar con las clases CSS no-box y avoid .
* hideCopy : oculta el botón de copiar.
* language : el idioma del código fuente, como javascript , html , css , typescript , json o sh . Este atributo solo se aplica a los ejemplos codificados.

### **Visualización de código incorrecto**

Ocasionalmente, desea mostrar un ejemplo de código o diseño menos que ideal, como con ejemplos de **evitar** en la [Guía de estilo angular](https://runebook.dev/es/docs/angular/guide/styleguide) . Dado que es posible que los lectores copien y peguen ejemplos de código inferior en sus propias aplicaciones, intente minimizar el uso de dicho código.

En los casos en los que necesite ejemplos inaceptables, puede configurar la class para avoid o tener la palabra avoid en el nombre del archivo de origen. Al poner la palabra avoid en el nombre del archivo o en la ruta, el generador de documentación agrega automáticamente la clase avoid al <code-example> . Cualquiera de estas opciones enmarca el fragmento de código en rojo brillante para captar la atención del lector.

Aquí está el marcado para un ejemplo de "evitar" en la [Guía de estilo angular](https://runebook.dev/es/docs/angular/guide/styleguide#style-05-03) que usa la palabra avoid en el nombre de la ruta:

<code-example

path="styleguide/src/05-03/app/heroes/shared/hero-button/hero-button.component.avoid.ts"

region="example"

header="app/heroes/hero-button/hero-button.component.ts">

</code-example>

Tener la palabra "evitar" en el nombre del archivo hace que el navegador muestre el fragmento de código con una cabecera y un borde rojos:

/\* evitar \*/

@Component({

selector: '[tohHeroButton]',

templateUrl: './hero-button.component.html'

})

export class HeroButtonComponent {}

Alternativamente, el HTML podría incluir la clase de avoid como se muestra a continuación:

<code-example

path="docs-style-guide/src/app/not-great.component.ts"

header="docs-style-guide/src/app/not-great.component.ts"

region="not-great"

class="avoid">

</code-example>

La aplicación explícita de la clase avoid provoca el mismo resultado de un encabezado rojo y un borde rojo:

export class NotGreatComponent {

buggyFunction() {

// las cosas podrían ir mejor aquí ...

}

}

### **Pestañas de código**

Las pestañas de código muestran código de manera muy similar a code-examples con la ventaja adicional de mostrar múltiples ejemplos de código dentro de una interfaz con pestañas. Cada pestaña muestra el código mediante un code-pane .

#### **code-tabs attributes**

* linenums : el valor puede ser true , false o un número que indique el número de la línea de partida. El valor predeterminado es false .

#### **code-pane attributes**

* path : un archivo en la carpeta de content/examples
* header : lo que se muestra en el encabezado de una pestaña
* linenums : anula la propiedad linenums en el nivel de las code-tabs para este panel en particular. El valor puede ser true , false o un número que indique el número de la línea inicial. El valor predeterminado es false .

El siguiente ejemplo muestra varias pestañas de código, cada una con su propio encabezado. Demuestra cómo se muestran los números de línea en <code-tabs> y <code-pane> .

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

<h2>My Heroes</h2>

<ul class="heroes">

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

[class.selected]="hero === selectedHero"

(click)="onSelect(hero)">

<span class="badge">{{hero.id}}</span> {{hero.name}}

</li>

</ul>

<div \*ngIf="selectedHero">

<h2>{{selectedHero.name}} details!</h2>

<div><label>id: </label>{{selectedHero.id}}</div>

<div>

<label>name: </label>

<input [(ngModel)]="selectedHero.name" placeholder="name"/>

</div>

</div>

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

import { Hero, HEROES } from './hero';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Authors Style Guide Sample';

heroes = HEROES;

selectedHero!: Hero;

onSelect(hero: Hero): void {

this.selectedHero = hero;

}

}

.heroes {

margin: 0 0 2em 0;

list-style-type: none;

padding: 0;

width: 15em;

}

{

"scripts": {

"start": "concurrently \"npm run build:watch\" \"npm run serve\"",

"test": "concurrently \"npm run build:watch\" \"karma start karma.conf.js\"",

"lint": "tslint ./src/\*\*/\*.ts -t verbose"

}

}

El atributo linenums establecido en true en <code-tabs> habilita explícitamente la numeración de todos los paneles. Sin embargo, el atributo linenums establecido en false en el segundo <code-pane> deshabilita la numeración de líneas solo para sí mismo.

<code-tabs linenums="true">

<code-pane

header="app.component.html"

path="docs-style-guide/src/app/app.component.html">

</code-pane>

<code-pane

header="app.component.ts"

path="docs-style-guide/src/app/app.component.ts"

linenums="false">

</code-pane>

<code-pane

header="app.component.css (heroes)"

path="docs-style-guide/src/app/app.component.css"

region="heroes">

</code-pane>

<code-pane

header="package.json (scripts)"

path="docs-style-guide/package.1.json">

</code-pane>

</code-tabs>

## **Preparación del código fuente para los fragmentos de código**

Para mostrar fragmentos de <code-example> y <code-tabs> , agregue el marcado de fragmentos de código a los archivos de código fuente de muestra.

El código fuente de muestra para esta página, ubicado en content/examples/docs-style-guide , contiene ejemplos de cada marcado de fragmento de código descrito en esta sección.

El marcado de fragmentos de código siempre tiene la forma de un comentario. El marcado #docregion predeterminado para un archivo TypeScript o JavaScript es el siguiente:

// #docregion

... some TypeScript or JavaScript code ...

// #enddocregion

<!-- #docregion -->

... some HTML ...

<!-- #enddocregion -->

/\* #docregion \*/

... some CSS ...

/\* #enddocregion \*/

El proceso de generación de documentación borra estos comentarios antes de mostrarlos en el visor de documentación,en StackBlitz y en las descargas de código de ejemplo.

Debido a que JSON no permite comentarios, el marcado de fragmentos de código no funciona en archivos JSON. Consulte la sección sobre [archivos JSON](https://runebook.dev/es/docs/angular/guide/docs-style-guide#json-files) para obtener más información.

### **#docregion**

Use #docregion en los archivos fuente para marcar el código para su uso en componentes <code-example> o <code-tabs> .

El comentario #docregion comienza una región de fragmento de código. Cada línea de código después de ese comentario pertenece a la región hasta que el procesador de fragmentos de código encuentra el final del archivo o un #enddocregion de cierre .

El siguiente src/main.ts es un ejemplo de un archivo con un solo #docregion en la parte superior del archivo.

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

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

if (environment.production) {

enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule);

Como resultado, todo el archivo está en <code-example> .

### **Nombrar una #docregion**

Para mostrar varios fragmentos de diferentes fragmentos dentro del mismo archivo, #docregion cada fragmento su propio nombre #docregion de la siguiente manera, donde your-region-name es una cadena en minúscula con guión:

// #docregion tu-region-name

... some code ...

// #enddocregion tu-región-nombre

Haga referencia a esta región por su nombre en el atributo de region de <code-example> o <code-pane> de la siguiente manera:

<code-example

path="your-example-app/src/app/your-file.ts"

region="your-region-name"></code-example>

Debido a que #docregion sin nombre es la región predeterminada, no es necesario establecer el atributo de region cuando se hace referencia a la #docregion predeterminada .

### **Anidando una #docregion**

Coloque un #docregion dentro de otro #docregion como en el siguiente ejemplo con una inner-region anidada :

// #docregion

... some code ...

// #docregion inner-region

... more code ...

// #enddocregion inner-region

... yet more code ...

/// #enddocregion

### **Combinación de fragmentos de código**

Combine varios fragmentos del mismo archivo en un solo fragmento de código definiendo varias secciones #docregion con el mismo nombre de región. El siguiente ejemplo define dos secciones #docregion anidadas .

La región interna, class-skeleton , aparece dos veces: una para capturar el código que abre la definición de clase y una segunda vez para capturar el código que cierra la definición de clase.

// #docplaster

...

// #docregion class, class-skeleton

export class AppComponent {

// #enddocregion class-skeleton

title = 'Authors Style Guide Sample';

heroes = HEROES;

selectedHero: Hero;

onSelect(hero: Hero): void {

this.selectedHero = hero;

}

// #docregion class-skeleton

}

// #enddocregion clase, clase-esqueleto

El marcador #docplaster le dice al procesador qué cadena de texto usar, es decir, el "yeso", para unir cada uno de los fragmentos en un solo fragmento. Coloque el texto "yeso" en la misma línea. Por ejemplo, #docplaster --- usaría --- como el texto "yeso". En el caso del archivo anterior, el texto "yeso" está vacío, por lo que no habrá nada entre cada fragmento.

Sin #docplaster , el procesador inserta el yeso predeterminado (un comentario de puntos suspensivos) entre los fragmentos.

Aquí están los dos fragmentos de código correspondientes para compararlos.

export class AppComponent {

title = 'Authors Style Guide Sample';

heroes = HEROES;

selectedHero!: Hero;

onSelect(hero: Hero): void {

this.selectedHero = hero;

}

}

export class AppComponent {

}

El ejemplo anterior también demuestra que un comentario #docregion o #enddocregion puede especificar dos nombres de región, lo cual es una forma conveniente de iniciar o detener varias regiones en la misma línea de código. Alternativamente, puede poner estos comentarios en líneas separadas como en el siguiente ejemplo:

// #docplaster

...

// #docregion class

// #docregion class-skeleton

export class AppComponent {

// #enddocregion class-skeleton

title = 'Authors Style Guide Sample';

heroes = HEROES;

selectedHero: Hero;

onSelect(hero: Hero): void {

this.selectedHero = hero;

}

// #docregion class-skeleton

}

// #enddocregion class

// #enddocregion class-skeleton

### **Los archivos JSON**

El componente <code-example> no puede mostrar partes de un archivo JSON porque JSON prohíbe los comentarios. Sin embargo, puede mostrar un archivo JSON completo haciendo referencia a él en el atributo <code-example> src .

En el caso de archivos JSON de gran tamaño,podría copiar los nodos de interés en las marcas de Markdown,pero como es fácil crear por error un JSON no válido de esa manera,considere la posibilidad de crear un archivo parcial JSON con el fragmento que desea mostrar.

No puedes probar un archivo parcial ni utilizarlo en la aplicación,pero al menos tu editor puede confirmar que es sintácticamente correcto.También puedes guardar el archivo parcial junto al original,así es más probable que el autor se acuerde de mantener los dos sincronizados.

Aquí hay un ejemplo que extrae ciertos scripts de package.json en un archivo parcial llamado package.1.json .

{

"scripts": {

"start": "concurrently \"npm run build:watch\" \"npm run serve\"",

"test": "concurrently \"npm run build:watch\" \"karma start karma.conf.js\"",

"lint": "tslint ./src/\*\*/\*.ts -t verbose"

}

}

<code-example path="docs-style-guide/package.1.json" header="package.json (selected scripts)"></code-example>

En algunos casos, es preferible utilizar el nombre del archivo completo en lugar del parcial. En este caso, el archivo completo es package.json y el archivo parcial es package.1.json . Dado que el enfoque generalmente está en el archivo completo en lugar de en el parcial, el uso del nombre del archivo que edita el lector, en este ejemplo , package.json , aclara en qué archivo trabajar.

### **Nombramiento parcial de archivos**

La naturaleza paso a paso de las guías hace necesaria la refactorización,lo que significa que hay fragmentos de código que evolucionan a través de una guía.

Utilice archivos parciales para mostrar versiones intermedias del código fuente final con fragmentos de código que no aparecen en la aplicación final.La convención de nomenclatura de ejemplo añade un número antes de la extensión del archivo,como sigue:

package.1.json

app.component.1.ts

app.component.2.ts

Recuerde excluir estos archivos de StackBlitz enumerándolos en stackblitz.json como se ilustra aquí:

{

"description": "Authors style guide",

"files": [

"!\*\*/\*.d.ts",

"!\*\*/\*.js",

"!\*\*/\*.[1,2,3].\*"

],

"tags": ["author", "style guide"]

}

### **Estilo del código fuente**

El código fuente debe seguir [la guía de estilo de Angular](https://runebook.dev/es/docs/angular/guide/styleguide) siempre que sea posible.

#### **Hexadecimals**

El hexadecimal debe utilizar la taquigrafía siempre que sea posible,y utilizar sólo letras minúsculas.

## **Ejemplos vivos**

Agregar <live-example></live-example> a una página genera dos enlaces predeterminados: ejemplo en vivo.

El primero es un enlace al ejemplo de StackBlitz, que define el archivo stackblitz.json predeterminado . Puede encontrar el archivo stackblitz.json en el directorio content/examples/example-app , donde example-app es la carpeta de la aplicación de muestra que está usando para la guía.

De forma predeterminada, el generador de documentación usa el nombre de la guía como el nombre del ejemplo. Entonces, si está trabajando en router.md y usa <live-example></live-example> en el documento, el generador de documentación busca content/examples/router . Al hacer clic en este enlace, se abre el ejemplo de código en StackBlitz en una nueva pestaña del navegador.

El segundo enlace descarga la aplicación de ejemplo.

Defina ejemplos en vivo mediante uno o más archivos stackblitz.json en la raíz de una carpeta de muestra de código. Cada carpeta de muestra suele tener un único archivo de definición sin nombre, el stackblitz.json predeterminado .

### **Ejemplo en vivo para el llamado StackBlitz**

Puede crear archivos de definición con nombre adicionales con el formato name.stackblitz.json . La guía de [prueba](https://runebook.dev/es/docs/angular/guide/testing) ( aio/content/guide/testing.md ) hace referencia a un archivo StackBlitz con nombre de la siguiente manera:

<live-example stackblitz="specs">Tests</live-example>

El valor del atributo stackblitz de specs se refiere al archivo examples/testing/specs.stackblitz.json . Si stackblitz atributo stackblitz , el valor predeterminado sería examples/testing/stackblitz.json .

### **Etiqueta personalizada y punta de herramienta**

Cambie la apariencia y el comportamiento del ejemplo en vivo con atributos y clases. El siguiente ejemplo le da al ancla de ejemplo en vivo una etiqueta personalizada e información sobre herramientas estableciendo el atributo de title :

<live-example title="Live Example with title"></live-example>

El navegador muestra lo siguiente:

Puede lograr el mismo efecto colocando la etiqueta entre las etiquetas <live-example> :

<live-example>Live example with content label</live-example>

El navegador muestra lo siguiente:

Ejemplo en vivo con etiqueta de contenido

### **Ejemplo en vivo de otra guía**

Para vincular a un ejemplo en una carpeta donde el nombre no es el mismo que el de la página de guía actual, establezca el atributo de name en el nombre de esa carpeta.

Por ejemplo, para incluir el ejemplo de la guía del [enrutador](https://runebook.dev/es/docs/angular/guide/router) en esta guía de estilo, establezca el atributo de name en router , es decir, el nombre de la carpeta donde reside ese ejemplo.

<live-example name="router">Live example from the Router guide</live-example>

Ejemplo en vivo de la guía del enrutador

### **Ejemplo en vivo sin descarga**

Para omitir el enlace de descarga, agregue el atributo noDownload .

<live-example noDownload>Just the StackBlitz</live-example>

El navegador muestra lo siguiente:

Sólo el StackBlitz

### **Ejemplo en vivo con sólo una descarga**

Para omitir el enlace StackBlitz en vivo y solo el enlace a la descarga, agregue el atributo downloadOnly .

<live-example downloadOnly>Download only</live-example>

El navegador muestra lo siguiente:

Sólo descarga.

### **Ejemplo en vivo incrustado**

De forma predeterminada, un enlace de ejemplo en vivo abre un ejemplo de StackBlitz en una pestaña separada del navegador. Puede incrustar el ejemplo de StackBlitz dentro de la página de la guía agregando el atributo embedded .

Por motivos de rendimiento, StackBlitz no se inicia de inmediato. En cambio, el componente <live-example> representa una imagen. Al hacer clic en la imagen, se inicia el proceso de iniciar el StackBlitz incrustado dentro de un <iframe> .

El siguiente es un <live-example> incrustado para esta guía:

**<live-example embedded></live-example>**

El navegador muestra el siguiente <iframe> y un <p> con un enlace para descargar el ejemplo:

## **Anchors**

Cada cabecera de sección es también un punto de anclaje.Otra página de la guía podría añadir un enlace a esta sección de "Anclas" con lo siguiente:

<div class="alert is-helpful">

See the ["Anchors"](guide/docs-style-guide#anchors "Style Guide&mdash;Anchors") section for details.

</div>

El navegador muestra lo siguiente:

Vea la sección ["Anclas"](https://runebook.dev/es/docs/angular/guide/docs-style-guide#anchors) para más detalles.

Observe que el ejemplo anterior incluye un título de "Guía de estilo: anclajes". Utilice títulos en los anclajes para crear información sobre herramientas y mejorar la experiencia de usuario.

Al navegar dentro de una página, puede omitir la URL de la página al especificar el enlace que se [desplaza hasta](https://runebook.dev/es/docs/angular/guide/docs-style-guide#anchors) el principio de esta sección, como se muestra a continuación:

... the link that [scrolls up](#anchors "Anchors") to ...

### **Anclajes de cabecera de sección**

Aunque el generador de documentación crea automáticamente anclas para los encabezados basándose en la redacción de los mismos,los títulos pueden cambiar,lo que puede romper potencialmente cualquier enlace a esa sección.

Para mitigar la rotura de enlaces,añada un ancla personalizada explícitamente,justo encima del título o texto al que se aplica,utilizando la sintaxis especial siguiente:

{@a section-anchors}

#### Section header anchors

Entonces haz referencia a ese ancla así:

This is a [link to that custom anchor name](#section-anchors).

El navegador muestra lo siguiente:

Este es un [enlace a ese nombre de ancla personalizado](https://runebook.dev/es/docs/angular/guide/docs-style-guide#section-anchors) .

Cuando edites un archivo,no elimines ninguna ancla.Si cambias la estructura del documento,puedes mover un ancla existente dentro del mismo documento sin romper un enlace.También puede añadir más anclas con un texto más apropiado.

Como alternativa, puede utilizar la etiqueta HTML <a> . Cuando utilice la etiqueta <a> , establezca el atributo id , en lugar del atributo name , porque el generador de documentación no convierte el name a la URL de enlace adecuada. Por ejemplo:

**<a id="anchors"></a>**

## Anchors

## **Alertas y avisos**

Las alertas y los avisos presentan advertencias,detalles adicionales o referencias a temas relacionados.

Una alerta o llamada de atención no debe contener nada esencial para la comprensión del contenido principal.Las instrucciones o los pasos del tutorial deben estar en el cuerpo principal de una guía y no en una subsección.

### **Alerts**

Las alertas llaman la atención sobre puntos breves e importantes. Para contenido de varias líneas, consulte las [leyendas](https://runebook.dev/es/docs/angular/guide/docs-style-guide#callouts) .

Consulte la sección de [ejemplos en vivo](https://runebook.dev/es/docs/angular/guide/docs-style-guide#live-examples) para obtener más información.

Tenga en cuenta que al menos una línea en blanco debe seguir las etiquetas <div> de apertura y cierre . Una línea en blanco antes del cierre </div> es convencional pero no obligatoria.

<div class="alert is-helpful">

See the [live examples](guide/docs-style-guide#live-examples "Live examples") section for more information.

</div>

Hay tres niveles diferentes para estilizar las alertas según la importancia del contenido.

Utilice la siguiente clase de alert a junto con la clase CSS adecuada is-helpful , is-important o is-critical , de la siguiente manera:

<div class="alert is-helpful">

A helpful, informational alert.

</div>

<div class="alert is-important">

An important alert.

</div>

<div class="alert is-critical">

A critical alert.

</div>

El navegador muestra lo siguiente:

Una alerta útil e informativa.

Una alerta importante.

Una alerta crítica.

### **Callouts**

Las llamadas,al igual que las alertas,resaltan los puntos importantes.Utiliza una llamada cuando necesites una cabecera y un contenido de varias líneas.

Si tiene más de dos párrafos,considere la posibilidad de crear una nueva página o hacerla parte del contenido principal.

Las llamadas utilizan los mismos niveles de estilo que las alertas.

Utilice la callout clase CSS junto con la clase apropiada is-helpful , is-important o is-critical . El siguiente ejemplo usa la clase is-helpful :

<div class="callout is-helpful">

<header>A helpful or informational point</header>

\*\*A helpful note\*\*. Use a helpful callout for information requiring explanation.

Callouts are typically multi-line notes.

They can also contain code snippets.

</div>

El navegador representa los tres estilos de la siguiente manera:

Un punto útil o informativo

**Una nota útil** . Utilice una llamada útil para obtener información que requiera una explicación. Las llamadas suelen ser notas de varias líneas. También pueden contener fragmentos de código.

Un punto importante

**Una nota importante** . Utilice una llamada importante para obtener información importante que requiera explicación. Las llamadas suelen ser notas de varias líneas. También pueden contener fragmentos de código.

Un punto crítico

**Una nota crítica** . Utilice una llamada crítica para obtener información convincente que requiera una explicación. Las llamadas suelen ser notas de varias líneas. También pueden contener fragmentos de código.

Al utilizar las llamadas,tenga en cuenta los siguientes puntos:

* El estilo del texto de la cabecera de la llamada es mayúscula.
* La cabecera no aparece en el índice.
* Puedes escribir el cuerpo de la llamada en Markdown.
* Una línea en blanco separa la etiqueta <header> del contenido de Markdown.
* Evite el uso de <h2> , <h3> , <h4> , <h5> o <h6> , ya que el CSS para las llamadas aplica estilo al elemento <header> .

Utilice las llamadas de atención con moderación para captar la atención del usuario.

## **Trees**

Utilizar árboles para representar datos jerárquicos como la estructura de directorios.

sample-dir

src

app

app.component.ts

app.module.ts

styles.css

tsconfig.json

node\_modules ...

package.json

Aquí está la marca de este árbol de archivos.

<div class='filetree'>

<div class='file'>

sample-dir

</div>

<div class='children'>

<div class='file'>

src

</div>

<div class='children'>

<div class='file'>

app

</div>

<div class='children'>

<div class='file'>

app.component.ts

</div>

<div class='file'>

app.module.ts

</div>

</div>

<div class='file'>

styles.css

</div>

<div class='file'>

tsconfig.json

</div>

</div>

<div class='file'>

node\_modules ...

</div>

<div class='file'>

package.json

</div>

</div>

</div>

## **Images**

Almacene imágenes en el directorio de content/images/guide en una carpeta con el **mismo nombre** que la página de guía. Debido a que la generación de documentación de Angular copia estas imágenes en el generated/images/guide/your-guide-directory , configure el atributo image src en la ubicación del tiempo de ejecución del generated/images/guide/your-guide-directory .

Por ejemplo, las imágenes de esta "Guía de estilo de documentación angular" se encuentran en la carpeta content/images/guide/docs-style-guide , pero el atributo src especifica la ubicación generated .

El siguiente es el atributo src para la imagen del "héroe volador" que pertenece a esta guía:

src="generated/images/guide/docs-style-guide/flying-hero.png"

Utilice el código HTML <img> etiqueta

Especifique imágenes usando la etiqueta <img> . **No utilice la sintaxis de imagen de Markdown,! [...] (...).**

Para accesibilidad, establezca siempre el atributo alt con una descripción significativa de la imagen.

Anida la etiqueta <img> dentro de una etiqueta <div class="lightbox"> , que diseña la imagen de acuerdo con el estándar de documentación.

<div class="lightbox">

<img src="generated/images/guide/docs-style-guide/flying-hero.png" alt="flying hero">

</div>

Tenga en cuenta que el elemento HTML <img> no tiene una etiqueta de cierre.

El navegador muestra lo siguiente:



### **Leyendas de las imágenes y de las figuras**

Aparece una leyenda debajo de la imagen como un resumen conciso y completo de la imagen. Los subtítulos son opcionales a menos que esté utilizando figuras numeradas, como Figura 1, Figura 2, etc. Si está utilizando figuras numeradas en una página, siga las pautas en los [títulos](https://developers.google.com/style/images#figure-captions) de las figuras en la Guía de estilo de documentación para desarrolladores de Google.

### **Dimensiones de la imagen**

El generador de documentos lee las dimensiones de la imagen de un archivo de imagen y agrega atributos de width y height a la etiqueta <img> automáticamente.

Para controlar el tamaño de la imagen, proporcione sus propios atributos de width y height .

Aquí está el marcado "héroe volador" con un ancho de 200px:

<div class="lightbox">

<img src="generated/images/guide/docs-style-guide/flying-hero.png"

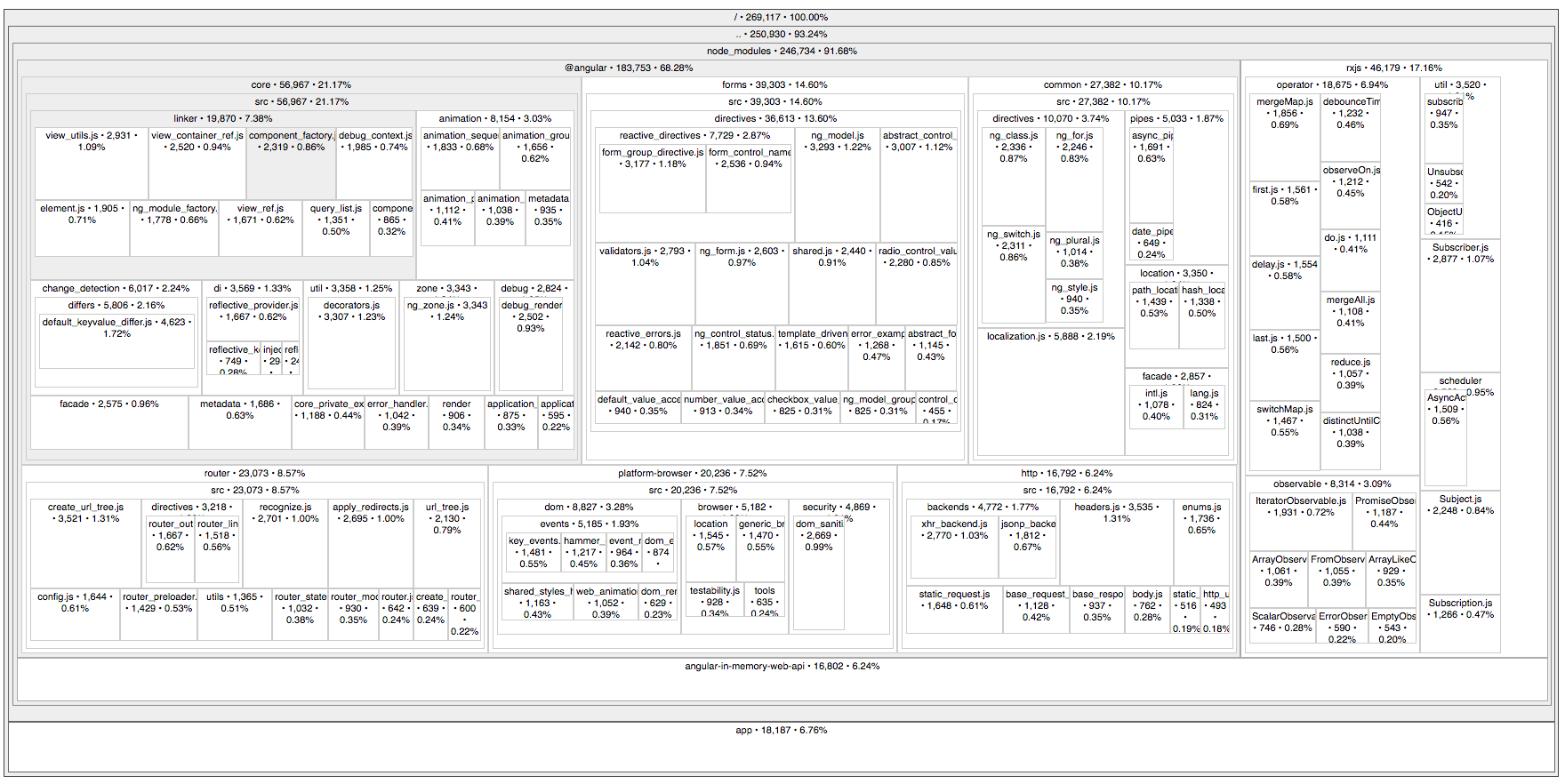
alt="flying Angular hero"

width="200">

</div>

### **Wide images**

Para evitar que las imágenes desborden sus ventanas gráficas, **utilice anchos de imagen inferiores a 700 px** . Para mostrar una imagen más grande, proporcione un vínculo a la imagen real en la que el usuario puede hacer clic para ver la imagen a tamaño completo por separado, como en este ejemplo de salida de source-map-explorer de la guía "Compilación anticipada":



Link imagen: https://angular.io/generated/images/guide/docs-style-guide/toh-pt6-bundle.png

El siguiente es el HTML para crear un enlace a la imagen:

<a href="generated/images/guide/docs-style-guide/toh-pt6-bundle.png" title="Click to view larger image">

<div class="lightbox">

<img src="generated/images/guide/docs-style-guide/toh-pt6-bundle-700w.png" alt="toh-pt6-bundle" width="300px">

</div>

</a>

### **Compresión de la imagen**

Para tiempos de carga más rápidos, comprima siempre las imágenes. Considere utilizar un sitio web de compresión de imágenes como [tinypng](https://tinypng.com/) .

### **Floated images**

Puede hacer flotar la imagen a la izquierda o a la derecha del texto aplicando los atributos class="left" o class="right" respectivamente.Por ejemplo:

<img src="generated/images/guide/docs-style-guide/flying-hero.png"

alt="flying Angular hero"

width="200"

class="left">

This text wraps around to the right of the floating "flying hero" image.

Headings and code-examples automatically clear a floated image. If you need to force a piece of text to clear a floating image, add `<br class="clear">` where the text should break.

<br class="clear">

Este texto se envuelve a la derecha de la imagen flotante del "héroe volador".

Los encabezados y los componentes <code-example> borran automáticamente una imagen flotante. Para borrar explícitamente una imagen flotante, agregue <br class="clear"> donde debe dividirse el texto.

Generalmente, no envuelve una imagen flotante en un elemento <figure> .

### **Flotantes dentro de una subsección**

Si tiene una imagen flotante dentro de una alerta, llamada o subsección, aplique la clase clear-fix al <div> para asegurarse de que la imagen no se desborde de su contenedor. Por ejemplo:

<div class="alert is-helpful clear-fix">

<img src="generated/images/guide/docs-style-guide/flying-hero.png"

alt="flying Angular hero"

width="100"

class="right">

A subsection with \*\*Markdown\*\* formatted text.

</div>

El navegador muestra lo siguiente:

