

Reference

Conceptual Reference

CLI Command Reference

API Reference

Error Reference

Example applications

Angular Glossary

Angular Style and Usage

Quick Reference

Coding Style Guide

Content Contributor's Guide

stable (v13.0.2)

API Reference

Error Reference

Example applications

Angular Glossary

Angular Style and Usage

Quick Reference

Coding Style Guide

Content Contributor's Guide

stable (v13.0.2)

API Reference

Error Reference

Example applications

Angular Glossary

Angular Style and Usage

Quick Reference

Coding Style Guide

Content Contributor's Guide

stable (v13.0.2)

API Reference

Error Reference

Example applications

Angular Glossary

Angular Style and Usage

Quick Reference

Coding Style Guide

Cheat Sheet

Bootstrapping

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

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

Bootstraps the application, using the root component from the specified `NgModule`.

NgModules

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

```
@NgModule({
  declarations: ...,
  imports: ...,
  exports: ...,
  providers: ...,
  bootstrap: ...
})
class MyModule {}
```

Defines a module that contains components, directives, pipes, and providers.

```
declarations: [MyRedComponent, MyBlueComponent, MyDatePipe]
```

List of components, directives, and pipes that belong to this module.

```
imports: [BrowserModule, SomeOtherModule]
```

List of modules to import into this module. Everything from the imported modules is available to `declarations` of this module.

```
exports: [MyRedComponent, MyDatePipe]
```

List of components, directives, and pipes visible to modules that import this module.

```
providers: [MyService, { provide: ... }]
```

List of dependency injection providers visible both to the contents of this module and to importers of this module.

```
entryComponents: [SomeComponent, OtherComponent]
```

List of components not referenced in any reachable template, for example dynamically created from code.

```
bootstrap: [MyAppComponent]
```

List of components to bootstrap when this module is bootstrapped.

Template syntax

```
<input [value]="firstName">
```

Binds property `value` to the result of expression `firstName`.

```
<div [attr.role]="myAriaRole">
```

Binds attribute `role` to the result of expression `myAriaRole`.

```
<div [class.extra-sparkle]="isDelightful">
```

Binds the presence of the CSS class `extra-sparkle` on the element to the truthiness of the expression `isDelightful`.

```
<div [style.width.px]="mySize">
```

Binds style property `width` to the result of expression `mySize` in pixels. Units are optional.

```
<button (click)="readRainbow($event)">
```

Calls method `readRainbow` when a click event is triggered on this button element (or its children) and passes in the event object.

Content Contributor's Guide stable (v13.0.2)	<pre><div title="Hello {{ponyName}}"></pre>	Binds a property to an interpolated string, for example, "Hello Seabiscuit". Equivalent to: <code><div [title]="'Hello' + ponyName"></code>
	<pre><p>Hello {{ponyName}}</p></pre>	Binds text content to an interpolated string, for example, "Hello Seabiscuit".
	<pre><my-cmp [(title)]="name"></pre>	Sets up two-way data binding. Equivalent to: <code><my-cmp [title]="name" (titleChange)="name=\$event"></code>
	<pre><video #movieplayer ...></video> <button (click)="movieplayer.play()">Play</button></pre>	Creates a local variable <code>movieplayer</code> that provides access to the <code>video</code> element instance in data-binding and event-binding expressions in the current template.
	<pre><p *myUnless="myExpression">...</p></pre>	The <code>*</code> symbol turns the current element into an embedded template. Equivalent to: <code><ng-template [myUnless]="myExpression"><p>...</p></ng-template></code>
	<pre><p>Card No.: {{cardNumber myCardNumberFormatter}}</p></pre>	Transforms the current value of expression <code>cardNumber</code> using the pipe called <code>myCardNumberFormatter</code> .
	<pre><p>Employer: {{employer?.companyName}}</p></pre>	The safe navigation operator (<code>?</code>) means that the <code>employer</code> field is optional and if <code>undefined</code> , the rest of the expression should be ignored.
	<pre><svg>rect x="0" y="0" width="100" height="100"/></pre>	An SVG snippet template needs an <code>svg:</code> prefix on its root element to disambiguate the SVG element from an HTML component.
	<pre><svg> <rect x="0" y="0" width="100" height="100"/> </svg></pre>	An <code><svg></code> root element is detected as an SVG element automatically, without the prefix.
	<h3>Built-in directives</h3> <pre><section *ngIf="showSection"></pre>	
<pre><li *ngFor="let item of list"></pre>		
<pre><div [ngSwitch]="conditionExpression"> <ng-template [ngSwitchCase]="case1Exp">...</ng-template> <ng-template [ngSwitchCase]="case2LiteralString">...</ng-template> </ng-template> <ng-template ngSwitchDefault>...</ng-template> </div></pre>	Conditionally swaps the contents of the div by selecting one of the embedded templates based on the current value of <code>conditionExpression</code> .	
<pre><div [ngClass]="{'active': isActive, 'disabled': isDisabled}"></pre>		
<pre><div [ngStyle]="{'property': 'value'}"> <div [ngStyle]="dynamicStyles()"></pre>		

<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> stable (v13.0.2) </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> API Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Error Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Example applications </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Glossary </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Style and Usage </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Quick Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Coding Style Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Content Contributor's Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> stable (v13.0.2) </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> API Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Error Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Example applications </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Glossary </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Style and Usage </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Quick Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Coding Style Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Content Contributor's Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> stable (v13.0.2) </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> API Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Error Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Example applications </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Glossary </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Style and Usage </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Quick Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Coding Style Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Content Contributor's Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> stable (v13.0.2) </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> API Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Error Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Example applications </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Glossary </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Angular Style and Usage </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Quick Reference </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Coding Style Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Content Contributor's Guide </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> stable (v13.0.2) </div>	<h3>Forms</h3> <pre>import { FormsModule } from '@angular/forms';</pre> <p><code><input [(ngModel)]="userName"></code></p> <p>Provides two-way data-binding, parsing, and validation for form controls.</p> <h3>Class decorators</h3> <pre>import { Directive, ... } from '@angular/core';</pre> <p>@Component({...}) class MyComponent() {}</p> <p>Declares that a class is a component and provides metadata about the component.</p> <p>@Directive({...}) class MyDirective() {}</p> <p>Declares that a class is a directive and provides metadata about the directive.</p> <p>@Pipe({...}) class MyPipe() {}</p> <p>Declares that a class is a pipe and provides metadata about the pipe.</p> <p>@Injectable() class MyService() {}</p> <p>Declares that a class can be provided and injected by other classes. Without this decorator, the compiler won't generate enough metadata to allow the class to be created properly when it's injected somewhere.</p> <h3>Directive configuration</h3> <pre>@Directive({ property1: value1, ... })</pre> <p>selector: '.cool-button:not(a)'</p> <p>Specifies a CSS selector that identifies this directive within a template. Supported selectors include <code>element</code>, <code>[attribute]</code>, <code>.class</code>, and <code>:not()</code>.</p> <p>Does not support parent-child relationship selectors.</p> <p>providers: [MyService, { provide: ... }]</p> <p>List of dependency injection providers for this directive and its children.</p> <h3>Component configuration</h3> <p>@Component extends <code>@Directive</code>, so the <code>@Directive</code> configuration applies to components as well</p> <p>moduleId: module.id</p> <p>If set, the <code>templateUrl</code> and <code>styleUrl</code> are resolved relative to the component.</p> <p>viewProviders: [MyService, { provide: ... }]</p> <p>List of dependency injection providers scoped to this component's view.</p> <p>template: 'Hello {{name}}' templateUrl: 'my-component.html'</p> <p>Inline template or external template URL of the component's view.</p> <p>styles: ['.primary {color: red}'] styleUrls: ['my-component.css']</p> <p>List of inline CSS styles or external stylesheet URLs for styling the component's view.</p>
--	---

API Reference	
Error Reference	>
Example applications	
Angular Glossary	
Angular Style and Usage	>
Quick Reference	
Coding Style Guide	
Content Contributor's Guide	>
stable (v13.0.2)	
API Reference	
Error Reference	>
Example applications	
Angular Glossary	
Angular Style and Usage	>
Quick Reference	
Coding Style Guide	
Content Contributor's Guide	>
stable (v13.0.2)	
API Reference	
Error Reference	>
Example applications	
Angular Glossary	
Angular Style and Usage	>
Quick Reference	
Coding Style Guide	
Content Contributor's Guide	>
stable (v13.0.2)	
API Reference	
Error Reference	>
Example applications	
Angular Glossary	
Angular Style and Usage	>
Quick Reference	
Coding Style Guide	
Content Contributor's Guide	>
stable (v13.0.2)	
API Reference	
Error Reference	>
Example applications	
Angular Glossary	
Angular Style and Usage	>
Quick Reference	
Coding Style Guide	
Content Contributor's Guide	>
stable (v13.0.2)	
API Reference	

Class field decorators for directives and components	
<code>@Input() myProperty;</code>	Declares an input property that you can update using property binding (example: <code><my-cmp [myProperty]="someExpression"></code>).
<code>@Output() myEvent = new EventEmitter();</code>	Declares an output property that fires events that you can subscribe to with an event binding (example: <code><my-cmp (myEvent)="doSomething()"></code>).
<code>@HostBinding('class.valid') isValid;</code>	Binds a host element property (here, the CSS class <code>valid</code>) to a directive/component property (<code>isValid</code>).
<code>@HostListener('click', ['\$event']) onClick(e) {...}</code>	Subscribes to a host element event (<code>click</code>) with a directive/component method (<code>onClick</code>), optionally passing an argument (<code>\$event</code>).
<code>@ContentChild(myPredicate) myChildComponent;</code>	Binds the first result of the component content query (<code>myPredicate</code>) to a property (<code>myChildComponent</code>) of the class.
<code>@ContentChildren(myPredicate) myChildComponents;</code>	Binds the results of the component content query (<code>myPredicate</code>) to a property (<code>myChildComponents</code>) of the class.
<code>@ViewChild(myPredicate) myChildComponent;</code>	Binds the first result of the component view query (<code>myPredicate</code>) to a property (<code>myChildComponent</code>) of the class. Not available for directives.
<code>@ViewChildren(myPredicate) myChildComponents;</code>	Binds the results of the component view query (<code>myPredicate</code>) to a property (<code>myChildComponents</code>) of the class. Not available for directives.
Directive and component change detection and lifecycle hooks	(implemented as class methods)
<code>constructor(myService: MyService, ...) { ... }</code>	Called before any other lifecycle hook. Use it to inject dependencies, but avoid any serious work here.
<code>ngOnChanges(changeRecord) { ... }</code>	Called after every change to input properties and before processing content or child views.
<code>ngOnInit() { ... }</code>	Called after the constructor, initializing input properties, and the first call to <code>ngOnChanges</code> .
<code>ngDoCheck() { ... }</code>	Called every time that the input properties of a component or a directive are checked. Use it to extend change detection by performing a custom check.
<code>ngAfterContentInit() { ... }</code>	Called after <code>ngOnInit</code> when the component's or directive's content has been initialized.
<code>ngAfterContentChecked() { ... }</code>	Called after every check of the component's or directive's

Error Reference Example applications Angular Glossary Angular Style and Usage Quick Reference Coding Style Guide Content Contributor's Guide	<p>content.</p> <hr/> <p>ngAfterViewInit() { ... }</p> <p>Called after <code>ngAfterContentInit</code> when the component's views and child views / the view that a directive is in has been initialized.</p> <hr/> <p>ngAfterViewChecked() { ... }</p> <p>Called after every check of the component's views and child views / the view that a directive is in.</p> <hr/> <p>ngOnDestroy() { ... }</p> <p>Called once, before the instance is destroyed.</p>
stable (v13.0.2) API Reference Error Reference Example applications Angular Glossary Angular Style and Usage Quick Reference Coding Style Guide Content Contributor's Guide	<h3>Dependency injection configuration</h3> <hr/> <p>{ provide: MyService, useClass: MyMockService }</p> <p>Sets or overrides the provider for <code>MyService</code> to the <code>MyMockService</code> class.</p> <hr/> <p>{ provide: MyService, useFactory: myFactory }</p> <p>Sets or overrides the provider for <code>MyService</code> to the <code>myFactory</code> factory function.</p> <hr/> <p>{ provide: MyValue, useValue: 41 }</p> <p>Sets or overrides the provider for <code>MyValue</code> to the value <code>41</code>.</p>
stable (v13.0.2) API Reference Error Reference Example applications Angular Glossary Angular Style and Usage Quick Reference Coding Style Guide Content Contributor's Guide	<h3>Routing and navigation</h3> <hr/> <pre>import { Routes, RouterModule, ... } from '@angular/router'; const routes: Routes = [{ path: '', component: HomeComponent }, { path: 'path/:routeParams', component: MyComponent }, { path: 'staticPath', component: ... }, { path: '**', component: ... }, { path: 'oldPath', redirectTo: '/staticPath' }, { path: ..., component: ..., data: { message: 'Custom' } }]; const routing = RouterModule.forRoot(routes);</pre> <p>Configures routes for the application. Supports static, parameterized, redirect, and wildcard routes. Also supports custom route data and resolve.</p> <hr/> <p><router-outlet></router-outlet></p> <p><router-outlet name="aux"></router-outlet></p> <p>Marks the location to load the component of the active route.</p>
stable (v13.0.2) API Reference Error Reference Example applications Angular Glossary Angular Style and Usage Quick Reference Coding Style Guide Content Contributor's Guide	<p></p> <p><a [routerLink]="['/path', routeParam]"></p> <p><a [routerLink]="['/path', { matrixParam: 'value' }]"></p> <p><a [routerLink]="['/path'] [queryParams]={ page: 1 }"></p> <p><a [routerLink]="['/path'] fragment="anchor"></p> <p>Creates a link to a different view based on a route instruction consisting of a route path, required and optional parameters, query parameters, and a fragment. To navigate to a root route, use the <code>/</code> prefix; for a child route, use the <code>./</code> prefix; for a sibling or parent, use the <code>../</code> prefix.</p> <hr/> <p><a [routerLink]="['/path']" routerLinkActive="active"></p> <p>The provided classes are added to the element when the <code>routerLink</code> becomes the current active route.</p> <hr/> <p>class CanActivateGuard implements CanActivate { canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean UrlTree> Promise<boolean UrlTree> bo</p> <p>An interface for defining a class that the router should call first to determine if it should activate this component. Should return a boolean UrlTree or an Observable/Promise that resolves to a boolean UrlTree.</p>

- [Example applications](#)
- [Angular Glossary](#)
- [Angular Style and Usage](#)
- [Quick Reference](#)
- [Coding Style Guide](#)
- [Content Contributor's Guide](#)

stable (v13.0.2)

- [API Reference](#)
- [Error Reference](#)
- [Example applications](#)
- [Angular Glossary](#)
- [Angular Style and Usage](#)
- [Quick Reference](#)
- [Coding Style Guide](#)
- [Content Contributor's Guide](#)

stable (v13.0.2)

- [API Reference](#)
- [Error Reference](#)
- [Example applications](#)
- [Angular Glossary](#)
- [Angular Style and Usage](#)
- [Quick Reference](#)
- [Coding Style Guide](#)
- [Content Contributor's Guide](#)

stable (v13.0.2)

- [API Reference](#)
- [Error Reference](#)
- [Example applications](#)
- [Angular Glossary](#)
- [Angular Style and Usage](#)
- [Quick Reference](#)
- [Coding Style Guide](#)
- [Content Contributor's Guide](#)

stable (v13.0.2)

- [API Reference](#)
- [Error Reference](#)
- [Example applications](#)
- [Angular Glossary](#)
- [Angular Style and Usage](#)
- [Quick Reference](#)
- [Coding Style Guide](#)
- [Content Contributor's Guide](#)

```
{ ... }
}

{ path: ..., canActivate: [CanActivateGuard] }
```

```
class CanDeactivateGuard implements CanDeactivate<T> {
  canDeactivate(
    component: T,
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): Observable<boolean|UrlTree>|Promise<boolean|UrlTree>|bo
  { ... }
}

{ path: ..., canDeactivate: [CanDeactivateGuard] }
```

```
class CanActivateChildGuard implements
CanActivateChild {
  canActivateChild(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): Observable<boolean|UrlTree>|Promise<boolean|UrlTree>|bo
  { ... }
}

{ path: ..., canActivateChild: [CanActivateGuard],
  children: ... }
```

```
class ResolveGuard implements Resolve<T> {
  resolve(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): Observable<any>|Promise<any>|any { ... }
}

{ path: ..., resolve: [ResolveGuard] }
```

```
class CanLoadGuard implements CanLoad {
  canLoad(
    route: Route
  ): Observable<boolean|UrlTree>|Promise<boolean|UrlTree>|bo
  { ... }
}

{ path: ..., canLoad: [CanLoadGuard], loadChildren:
  ... }
```

RESOURCES	HELP	COMMUNITY	LANGUAGES
About	Stack Overflow	Events	Español
Resource Listing	Join Discord	Meetups	简体中文版
Press Kit	Gitter	Twitter	正體中文版
Blog	Report Issues	GitHub	日本語版
Usage Analytics	Code of Conduct	Contribute	한국어
Complete language list			

Coding Style Guide

Content Contributor's
Guide >

stable (v13.0.2)

Super-powered by Google ©2010-2021.

Code licensed under an MIT-style License. Documentation licensed under CC BY 4.0.

Version 13.0.3-local+sha.16d026e499.