| Area | API or Feature | May be removed in |
| --- | --- | --- |
| @angular/common | [ReflectiveInjector](https://angular.io/guide/deprecations#reflectiveinjector) | v10 |
| @angular/common | [CurrencyPipe - DEFAULT\_CURRENCY\_CODE](https://angular.io/api/common/CurrencyPipe#currency-code-deprecation) | v11 |
| @angular/core | [CollectionChangeRecord](https://angular.io/guide/deprecations#core) | v10 |
| @angular/core | [DefaultIterableDiffer](https://angular.io/guide/deprecations#core) | v10 |
| @angular/core | [ReflectiveKey](https://angular.io/guide/deprecations#core) | v10 |
| @angular/core | [RenderComponentType](https://angular.io/guide/deprecations#core) | v10 |
| @angular/core | [ViewEncapsulation.Native](https://angular.io/guide/deprecations#core) | v10 |
| @angular/core | [ModuleWithProviders without a generic](https://angular.io/guide/deprecations#moduleWithProviders) | v10 |
| @angular/core | [Undecorated base classes that use Angular features](https://angular.io/guide/deprecations#undecorated-base-classes) | v10 |
| @angular/forms | [ngModel with reactive forms](https://angular.io/guide/deprecations#ngmodel-reactive) | v10 |
| @angular/router | [preserveQueryParams](https://angular.io/guide/deprecations#router) | v10 |
| @angular/upgrade | [@angular/upgrade](https://angular.io/guide/deprecations#upgrade) | v10 |
| @angular/upgrade | [getAngularLib](https://angular.io/guide/deprecations#upgrade-static) | v10 |
| @angular/upgrade | [setAngularLib](https://angular.io/guide/deprecations#upgrade-static) | v10 |
| @angular/platform-webworker | [All entry points](https://angular.io/api/platform-webworker) | v10 |
| template syntax | [<template>](https://angular.io/guide/deprecations#template-tag) | v10 |
| polyfills | [reflect-metadata](https://angular.io/guide/deprecations#reflect-metadata) | v10 |
| npm package format | [esm5 and fesm5 entry-points in @angular/\* npm packages](https://angular.io/guide/deprecations#esm5-fesm5) | v10 |
| @angular/core | [defineInjectable](https://angular.io/guide/deprecations#core) | v11 |
| @angular/core | [entryComponents](https://angular.io/api/core/NgModule#entryComponents) | v11 |
| @angular/core | [ANALYZE\_FOR\_ENTRY\_COMPONENTS](https://angular.io/api/core/ANALYZE_FOR_ENTRY_COMPONENTS) | v11 |
| @angular/router | [loadChildren string syntax](https://angular.io/guide/deprecations#loadChildren) | v11 |
| @angular/core/testing | [TestBed.get](https://angular.io/guide/deprecations#testing) | v12 |
| @angular/router | [ActivatedRoute params and queryParams properties](https://angular.io/guide/deprecations#activatedroute-props) | unspecified |
| template syntax | [/deep/, >>>, and ::ng-deep](https://angular.io/guide/deprecations#deep-component-style-selector) | unspecified |

Deprecated APIs[link](https://angular.io/guide/deprecations#deprecated-apis)

This section contains a complete list all of the currently-deprecated APIs, with details to help you plan your migration to a replacement.

Tip: In the [API reference section](https://angular.io/api) of this doc site, deprecated APIs are indicated by You can filter the API list by [Status: deprecated](https://angular.io/api?status=deprecated).

@angular/common[link](https://angular.io/guide/deprecations#angularcommon)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [CurrencyPipe - DEFAULT\_CURRENCY\_CODE](https://angular.io/api/common/CurrencyPipe#currency-code-deprecation) | {provide: [DEFAULT\_CURRENCY\_CODE](https://angular.io/api/core/DEFAULT_CURRENCY_CODE), [useValue](https://angular.io/api/core/ValueSansProvider#useValue): 'USD'} | v9 | From v11 the default code will be extracted from the locale data given by LOCAL\_ID, rather than USD. |

@angular/core[link](https://angular.io/guide/deprecations#angularcore)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [CollectionChangeRecord](https://angular.io/api/core/CollectionChangeRecord) | [IterableChangeRecord](https://angular.io/api/core/IterableChangeRecord) | v4 | none |
| [DefaultIterableDiffer](https://angular.io/api/core/DefaultIterableDiffer) | n/a | v4 | Not part of public API. |
| [ReflectiveInjector](https://angular.io/api/core/ReflectiveInjector) | [Injector.create](https://angular.io/api/core/Injector#create) | v5 | See [ReflectiveInjector](https://angular.io/guide/deprecations#reflectiveinjector) |
| [ReflectiveKey](https://angular.io/api/core/ReflectiveKey) | none | v5 | none |
| [ViewEncapsulation.Native](https://angular.io/api/core/ViewEncapsulation#Native) | [ViewEncapsulation.ShadowDom](https://angular.io/api/core/ViewEncapsulation#ShadowDom) | v6 | Use the native encapsulation mechanism of the renderer. See [view.ts](https://github.com/angular/angular/blob/3e992e18ebf51d6036818f26c3d77b52d3ec48eb/packages/core/src/metadata/view.ts#L32). |
| [defineInjectable](https://angular.io/api/core/defineInjectable) | ɵɵ[defineInjectable](https://angular.io/api/core/defineInjectable) | v8 | Used only in generated code. No source code should depend on this API. |
| [entryComponents](https://angular.io/api/core/NgModule#entryComponents) | none | v9 | See [entryComponents](https://angular.io/guide/deprecations#entryComponents) |
| [ANALYZE\_FOR\_ENTRY\_COMPONENTS](https://angular.io/api/core/ANALYZE_FOR_ENTRY_COMPONENTS) | none | v9 | See [ANALYZE\_FOR\_ENTRY\_COMPONENTS](https://angular.io/guide/deprecations#entryComponents) |
| [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders) without a generic | [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders) with a generic | v9 | See [ModuleWithProviders section](https://angular.io/guide/deprecations#moduleWithProviders) |
| Undecorated base classes that use Angular features | Base classes with @[Directive](https://angular.io/api/core/Directive)() decorator that use Angular features | v9 | See [undecorated base classes section](https://angular.io/guide/deprecations#undecorated-base-classes) |

@angular/core/testing[link](https://angular.io/guide/deprecations#angularcoretesting)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [TestBed.get](https://angular.io/api/core/testing/TestBed#get) | [TestBed.inject](https://angular.io/api/core/testing/TestBed#inject) | v9 | Same behavior, but type safe. |

@angular/forms[link](https://angular.io/guide/deprecations#angularforms)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [ngModel with reactive forms](https://angular.io/guide/deprecations#ngmodel-reactive) | See [FormControlDirective usage notes](https://angular.io/api/forms/FormControlDirective#use-with-ngmodel) | v6 | none |

@angular/router[link](https://angular.io/guide/deprecations#angularrouter)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [preserveQueryParams](https://angular.io/api/router/NavigationExtras#preserveQueryParams) | [queryParamsHandling](https://angular.io/api/router/NavigationExtras#queryParamsHandling) | v4 | none |

@angular/platform-webworker[link](https://angular.io/guide/deprecations#angularplatform-webworker)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [All entry points](https://angular.io/api/platform-webworker) | none | v8 | See [platform-webworker](https://angular.io/guide/deprecations#webworker-apps). |

@angular/platform-webworker-dynamic[link](https://angular.io/guide/deprecations#angularplatform-webworker-dynamic)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [All entry points](https://angular.io/api/platform-webworker-dynamic) | none | v8 | See [platform-webworker](https://angular.io/guide/deprecations#webworker-apps). |

@angular/upgrade[link](https://angular.io/guide/deprecations#angularupgrade)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [All entry points](https://angular.io/api/upgrade) | [@angular/upgrade/static](https://angular.io/api/upgrade/static) | v5 | See [Upgrading from AngularJS](https://angular.io/guide/upgrade). |

@angular/upgrade/static[link](https://angular.io/guide/deprecations#angularupgradestatic)

| API | Replacement | Deprecation announced | Notes |
| --- | --- | --- | --- |
| [getAngularLib](https://angular.io/api/upgrade/static/getAngularLib) | [getAngularJSGlobal](https://angular.io/api/upgrade/static/getAngularJSGlobal) | v5 | See [Upgrading from AngularJS](https://angular.io/guide/upgrade). |
| [setAngularLib](https://angular.io/api/upgrade/static/setAngularLib) | [setAngularJSGlobal](https://angular.io/api/upgrade/static/setAngularJSGlobal) | v5 | See [Upgrading from AngularJS](https://angular.io/guide/upgrade). |

Deprecated features[link](https://angular.io/guide/deprecations#deprecated-features)

This section lists all of the currently-deprecated features, which includes template syntax, configuration options, and any other deprecations not listed in the [Deprecated APIs](https://angular.io/guide/deprecations#deprecated-apis) section above. It also includes deprecated API usage scenarios or API combinations, to augment the information above.

Web Tracing Framework integration[link](https://angular.io/guide/deprecations#web-tracing-framework-integration)

Angular previously has supported an integration with the [Web Tracing Framework (WTF)](https://google.github.io/tracing-framework/) for performance testing of Angular applications. This integration has not been maintained and defunct. As a result, the integration was deprecated in Angular version 8 and due to no evidence of any existing usage removed in version 9.

/deep/, >>> and :ng-deep component style selectors[link](https://angular.io/guide/deprecations#deep--and-ng-deep-component-style-selectors)

The shadow-dom-piercing descendant combinator is deprecated and support is being [removed from major browsers and tools](https://developers.google.com/web/updates/2017/10/remove-shadow-piercing). As such, in v4 we deprecated support in Angular for all 3 of /deep/, >>> and ::ng-deep. Until removal, ::ng-deep is preferred for broader compatibility with the tools.

For more information, see [/deep/, >>>, and ::ng-deep](https://angular.io/guide/component-styles#deprecated-deep--and-ng-deep) in the Component Styles guide.

<template> tag[link](https://angular.io/guide/deprecations#template-tag)

The <template> tag was deprecated in v4 to avoid colliding with the DOM's element of the same name (such as when using web components). Use <ng-template> instead. For more information, see the [Ahead-of-Time Compilation](https://angular.io/guide/angular-compiler-options#enablelegacytemplate) guide.

ngModel with reactive forms[link](https://angular.io/guide/deprecations#ngmodel-with-reactive-forms)

Support for using the [ngModel](https://angular.io/api/forms/NgModel) input property and ngModelChange event with reactive form directives was deprecated in version 6.

For more information, see the usage notes for [FormControlDirective](https://angular.io/api/forms/FormControlDirective#use-with-ngmodel) and [FormControlName](https://angular.io/api/forms/FormControlName#use-with-ngmodel).

ReflectiveInjector[link](https://angular.io/guide/deprecations#reflectiveinjector)

In v5, Angular replaced the [ReflectiveInjector](https://angular.io/api/core/ReflectiveInjector) with the StaticInjector. The injector no longer requires the Reflect polyfill, reducing application size for most developers.

Before:

ReflectiveInjector.resolveAndCreate(providers);

content\_copy

ReflectiveInjector.resolveAndCreate(providers);

After:

Injector.create({providers});

content\_copy

Injector.create({providers});

loadChildren string syntax[link](https://angular.io/guide/deprecations#loadchildren-string-syntax)

When Angular first introduced lazy routes, there wasn't browser support for dynamically loading additional JavaScript. Angular created our own scheme using the syntax [loadChildren](https://angular.io/api/router/Route#loadChildren): './lazy/lazy.module#LazyModule' and built tooling to support it. Now that ECMAScript dynamic import is supported in many browsers, Angular is moving toward this new syntax.

In version 8, the string syntax for the [loadChildren](https://angular.io/api/router/LoadChildren) route specification was deprecated, in favor of new syntax that uses import() syntax.

Before:

const routes: [Routes](https://angular.io/api/router/Routes) = [{ path: 'lazy', // The following string syntax for [loadChildren](https://angular.io/api/router/Route#loadChildren) is deprecated [loadChildren](https://angular.io/api/router/Route#loadChildren): './lazy/lazy.module#LazyModule' }];

content\_copy

const routes: [Routes](https://angular.io/api/router/Routes) = [{

path: 'lazy',

// The following string syntax for [loadChildren](https://angular.io/api/router/Route#loadChildren) is deprecated

[loadChildren](https://angular.io/api/router/Route#loadChildren): './lazy/lazy.module#LazyModule'

}];

After:

const routes: [Routes](https://angular.io/api/router/Routes) = [{ path: 'lazy', // The new import() syntax [loadChildren](https://angular.io/api/router/Route#loadChildren): () => import('./lazy/lazy.module').then(m => m.LazyModule) }];

content\_copy

const routes: [Routes](https://angular.io/api/router/Routes) = [{

path: 'lazy',

// The new import() syntax

[loadChildren](https://angular.io/api/router/Route#loadChildren): () => import('./lazy/lazy.module').then(m => m.LazyModule)

}];

Version 8 update: When you update to version 8, the [ng update](https://angular.io/cli/update) command performs the transformation automatically. Prior to version 7, the import() syntax only works in JIT mode (with view engine).

Declaration syntax: It's important to follow the route declaration syntax [loadChildren](https://angular.io/api/router/Route#loadChildren): () => import('...').then(m => m.ModuleName) to allow ngc to discover the lazy-loaded module and the associated [NgModule](https://angular.io/api/core/NgModule). You can find the complete list of allowed syntax constructs [here](https://github.com/angular/angular-cli/blob/a491b09800b493fe01301387fa9a025f7c7d4808/packages/ngtools/webpack/src/transformers/import_factory.ts#L104-L113). These restrictions will be relaxed with the release of Ivy since it'll no longer use NgFactories.

ActivatedRoute params and queryParams properties[link](https://angular.io/guide/deprecations#activatedroute-params-and-queryparams-properties)

[ActivatedRoute](https://angular.io/api/router/ActivatedRoute) contains two [properties](https://angular.io/api/router/ActivatedRoute#properties) that are less capable than their replacements and may be deprecated in a future Angular version.

| Property | Replacement |
| --- | --- |
| params | paramMap |
| queryParams | queryParamMap |

For more information see the [Router guide](https://angular.io/guide/router#activated-route).

Dependency on a reflect-metadata polyfill in JIT mode[link](https://angular.io/guide/deprecations#dependency-on-a-reflect-metadata-polyfill-in-jit-mode)

Angular applications, and specifically applications that relied on the JIT compiler, used to require a polyfill for the [reflect-metadata](https://github.com/rbuckton/reflect-metadata) APIs.

The need for this polyfill was removed in Angular version 8.0 ([see #14473](https://github.com/angular/angular-cli/pull/14473)), rendering the presence of the poylfill in most Angular applications unnecessary. Because the polyfill can be depended on by 3rd-party libraries, instead of removing it from all Angular projects, we are deprecating the requirement for this polyfill as of version 8.0. This should give library authors and application developers sufficient time to evaluate if they need the polyfill, and perform any refactoring necessary to remove the dependency on it.

In a typical Angular project, the polyfill is not used in production builds, so removing it should not impact production applications. The goal behind this removal is overall simplification of the build setup and decrease in the number of external dependencies.

@[ViewChild](https://angular.io/api/core/ViewChild)() / @[ContentChild](https://angular.io/api/core/ContentChild)() static resolution as the default[link](https://angular.io/guide/deprecations#viewchild--contentchild-static-resolution-as-the-default)

See the [dedicated migration guide for static queries](https://angular.io/guide/static-query-migration).

@[ContentChild](https://angular.io/api/core/ContentChild)() / @[Input](https://angular.io/api/core/Input)() used together[link](https://angular.io/guide/deprecations#contentchild--input-used-together)

The following pattern is deprecated:

@[Input](https://angular.io/api/core/Input)() @[ContentChild](https://angular.io/api/core/ContentChild)([TemplateRef](https://angular.io/api/core/TemplateRef)) tpl !: [TemplateRef](https://angular.io/api/core/TemplateRef)<any>;

content\_copy

@[Input](https://angular.io/api/core/Input)() @[ContentChild](https://angular.io/api/core/ContentChild)([TemplateRef](https://angular.io/api/core/TemplateRef)) tpl !: [TemplateRef](https://angular.io/api/core/TemplateRef)<any>;

Rather than using this pattern, separate the two decorators into their own properties and add fallback logic as in the following example:

@[Input](https://angular.io/api/core/Input)() tpl !: [TemplateRef](https://angular.io/api/core/TemplateRef)<any>; @[ContentChild](https://angular.io/api/core/ContentChild)([TemplateRef](https://angular.io/api/core/TemplateRef)) inlineTemplate !: [TemplateRef](https://angular.io/api/core/TemplateRef)<any>;

content\_copy

@[Input](https://angular.io/api/core/Input)() tpl !: [TemplateRef](https://angular.io/api/core/TemplateRef)<any>;

@[ContentChild](https://angular.io/api/core/ContentChild)([TemplateRef](https://angular.io/api/core/TemplateRef)) inlineTemplate !: [TemplateRef](https://angular.io/api/core/TemplateRef)<any>;

Cannot assign to template variables[link](https://angular.io/guide/deprecations#cannot-assign-to-template-variables)

In the following example, the two-way binding means that optionName should be written when the valueChange event fires.

<option \*[ngFor](https://angular.io/api/common/NgForOf)="let optionName of options" [(value)]="optionName"></option>

content\_copy

<option \*[ngFor](https://angular.io/api/common/NgForOf)="let optionName of options" [(value)]="optionName"></option>

However, in practice, Angular simply ignores two-way bindings to template variables. Starting in version 8, attempting to write to template variables is deprecated. In a future version, we will throw to indicate that the write is not supported.

<option \*[ngFor](https://angular.io/api/common/NgForOf)="let optionName of options" [value]="optionName"></option>

content\_copy

<option \*[ngFor](https://angular.io/api/common/NgForOf)="let optionName of options" [value]="optionName"></option>

Undecorated base classes using Angular features[link](https://angular.io/guide/deprecations#undecorated-base-classes-using-angular-features)

As of version 9, it's deprecated to have an undecorated base class that:

* uses Angular features
* is extended by a directive or component

Angular lifecycle hooks or any of the following Angular field decorators are considered Angular features:

* @[Input](https://angular.io/api/core/Input)()
* @[Output](https://angular.io/api/core/Output)()
* @[HostBinding](https://angular.io/api/core/HostBinding)()
* @[HostListener](https://angular.io/api/core/HostListener)()
* @[ViewChild](https://angular.io/api/core/ViewChild)() / @[ViewChildren](https://angular.io/api/core/ViewChildren)()
* @[ContentChild](https://angular.io/api/core/ContentChild)() / @[ContentChildren](https://angular.io/api/core/ContentChildren)()

For example, the following case is deprecated because the base class uses @[Input](https://angular.io/api/core/Input)() and does not have a class-level decorator:

class Base { @[Input](https://angular.io/api/core/Input)() foo: string; } @[Directive](https://angular.io/api/core/Directive)(...) class Dir extends Base { ngOnChanges(): void { // notified when bindings to [foo] are updated } }

content\_copy

class Base {

@[Input](https://angular.io/api/core/Input)()

foo: string;

}

@[Directive](https://angular.io/api/core/Directive)(...)

class Dir extends Base {

ngOnChanges(): void {

// notified when bindings to [foo] are updated

}

}

In a future version of Angular, this code will start to throw an error. To fix this example, add a selectorless @[Directive](https://angular.io/api/core/Directive)() decorator to the base class:

@[Directive](https://angular.io/api/core/Directive)() class Base { @[Input](https://angular.io/api/core/Input)() foo: string; } @[Directive](https://angular.io/api/core/Directive)(...) class Dir extends Base { ngOnChanges(): void { // notified when bindings to [foo] are updated } }

content\_copy

@[Directive](https://angular.io/api/core/Directive)()

class Base {

@[Input](https://angular.io/api/core/Input)()

foo: string;

}

@[Directive](https://angular.io/api/core/Directive)(...)

class Dir extends Base {

ngOnChanges(): void {

// notified when bindings to [foo] are updated

}

}

In version 9, the CLI has an automated migration that will update your code for you when ng [update](https://angular.io/api/forms/NgModel#update) is run. See [the dedicated migration guide](https://angular.io/guide/migration-undecorated-classes) for more information about the change and more examples.

Binding to innerText in platform-server[link](https://angular.io/guide/deprecations#binding-to-innertext-in-platform-server)

[Domino](https://github.com/fgnass/domino), which is used in server-side rendering, doesn't support innerText, so in platform-server's "domino adapter", there was special code to fall back to textContent if you tried to bind to innerText.

These two properties have subtle differences, so switching to textContent under the hood can be surprising to users. For this reason, we are deprecating this behavior. Going forward, users should explicitly bind to textContent when using Domino.

wtfStartTimeRange and all wtf\* APIs[link](https://angular.io/guide/deprecations#wtfstarttimerange-and-all-wtf-apis)

All of the wtf\* APIs are deprecated and will be removed in a future version.

Running Angular applications in platform-webworker[link](https://angular.io/guide/deprecations#running-angular-applications-in-platform-webworker)

The @angular/platform-\* packages enable Angular to be run in different contexts. For examples, @angular/platform-server enables Angular to be run on the server, and @angular/platform-browser enables Angular to be run in a web browser.

@angular/platform-webworker was introduced in Angular version 2 as an experiment in leveraging Angular's rendering architecture to run an entire web application in a [web worker](https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API). We've learned a lot from this experiment and have come to the conclusion that running the entire application in a web worker is not the best strategy for most applications.

Going forward, we will focus our efforts related to web workers around their primary use case of offloading CPU-intensive, non-critical work needed for initial rendering (such as in-memory search and image processing). Learn more in the [guide to Using Web Workers with the Angular CLI](https://angular.io/guide/web-worker).

As of Angular version 8, all platform-webworker APIs are deprecated. This includes both packages: @angular/platform-webworker and @angular/platform-webworker-dynamic.

entryComponents and [ANALYZE\_FOR\_ENTRY\_COMPONENTS](https://angular.io/api/core/ANALYZE_FOR_ENTRY_COMPONENTS) no longer required[link](https://angular.io/guide/deprecations#entrycomponents-and-analyze_for_entry_components-no-longer-required)

Previously, the entryComponents array in the [NgModule](https://angular.io/api/core/NgModule) definition was used to tell the compiler which components would be created and inserted dynamically. With Ivy, this isn't a requirement anymore and the entryComponents array can be removed from existing module declarations. The same applies to the [ANALYZE\_FOR\_ENTRY\_COMPONENTS](https://angular.io/api/core/ANALYZE_FOR_ENTRY_COMPONENTS) injection token.

[ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders) type without a generic[link](https://angular.io/guide/deprecations#modulewithproviders-type-without-a-generic)

Some Angular libraries, such as @angular/router and @ngrx/store, implement APIs that return a type called [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders) (typically via a method named [forRoot()](https://angular.io/api/router/RouterModule#forRoot)). This type represents an [NgModule](https://angular.io/api/core/NgModule) along with additional providers. Angular version 9 deprecates use of [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders) without an explicitly generic type, where the generic type refers to the type of the [NgModule](https://angular.io/api/core/NgModule). In a future version of Angular, the generic will no longer be optional.

If you're using the CLI, ng [update](https://angular.io/api/forms/NgModel#update) should [migrate your code automatically](https://angular.io/guide/migration-module-with-providers). If you're not using the CLI, you can add any missing generic types to your application manually. For example:

Before

@[NgModule](https://angular.io/api/core/NgModule)({...}) export class MyModule { [static](https://angular.io/api/upgrade/static) forRoot([config](https://angular.io/api/router/Router#config): SomeConfig): [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders) { return { ngModule: SomeModule, providers: [ {provide: SomeConfig, [useValue](https://angular.io/api/core/ValueSansProvider#useValue): [config](https://angular.io/api/router/Router#config)} ] }; } }

content\_copy

@[NgModule](https://angular.io/api/core/NgModule)({...})

export class MyModule {

[static](https://angular.io/api/upgrade/static) forRoot([config](https://angular.io/api/router/Router#config): SomeConfig): [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders) {

return {

ngModule: SomeModule,

providers: [

{provide: SomeConfig, [useValue](https://angular.io/api/core/ValueSansProvider#useValue): [config](https://angular.io/api/router/Router#config)}

]

};

}

}

After

@[NgModule](https://angular.io/api/core/NgModule)({...}) export class MyModule { [static](https://angular.io/api/upgrade/static) forRoot([config](https://angular.io/api/router/Router#config): SomeConfig): [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders)<SomeModule> { return { ngModule: SomeModule, providers: [ {provide: SomeConfig, [useValue](https://angular.io/api/core/ValueSansProvider#useValue): [config](https://angular.io/api/router/Router#config) } ] }; } }

content\_copy

@[NgModule](https://angular.io/api/core/NgModule)({...})

export class MyModule {

[static](https://angular.io/api/upgrade/static) forRoot([config](https://angular.io/api/router/Router#config): SomeConfig): [ModuleWithProviders](https://angular.io/api/core/ModuleWithProviders)<SomeModule> {

return {

ngModule: SomeModule,

providers: [

{provide: SomeConfig, [useValue](https://angular.io/api/core/ValueSansProvider#useValue): [config](https://angular.io/api/router/Router#config) }

]

};

}

}

esm5 and fesm5 code formats in @angular/\* npm packages[link](https://angular.io/guide/deprecations#esm5-and-fesm5-code-formats-in-angular-npm-packages)

As of Angular v8, the CLI primarily consumes the fesm2015 variant of the code distributed via @angular/\* npm packages. This renders the esm5 and fesm5 distributions obsolete and unnecessary, adding bloat to the package size and slowing down npm installations.

The future removal of this distribution will have no impact on CLI users, unless they modified their build configuration to explicitly consume these code distributions.

Any application still relying on the esm5 and fesm5 as the input to its build system will need to ensure that the build pipeline is capable of accepting JavaScript code conforming to ECMAScript 2015 (ES2015) language specification.

Note that this change doesn't make existing libraries distributed in this format incompatible with the Angular CLI. The CLI will fall back and consume libraries in less desirable formats if others are not available. However, we do recommend that libraries ship their code in ES2015 format in order to make builds faster and build output smaller.

In practical terms, the package.json of all @angular packages will change in the following way:

Before:

{ "name": "@angular/core", "version": "9.0.0", "main": "./bundles/core.umd.js", "module": "./fesm5/core.js", "es2015": "./fesm2015/core.js", "esm5": "./esm5/core.js", "esm2015": "./esm2015/core.js", "fesm5": "./fesm5/core.js", "fesm2015": "./fesm2015/core.js", ... }

content\_copy

{

"name": "@angular/core",

"version": "9.0.0",

"main": "./bundles/core.umd.js",

"module": "./fesm5/core.js",

"es2015": "./fesm2015/core.js",

"esm5": "./esm5/core.js",

"esm2015": "./esm2015/core.js",

"fesm5": "./fesm5/core.js",

"fesm2015": "./fesm2015/core.js",

...

}

After:

{ "name": "@angular/core", "version": "10.0.0", "main": "./bundles/core.umd.js", "module": "./fesm2015/core.js", "es2015": "./fesm2015/core.js", "esm2015": "./esm2015/core.js", "fesm2015": "./fesm2015/core.js", ... }

content\_copy

{

"name": "@angular/core",

"version": "10.0.0",

"main": "./bundles/core.umd.js",

"module": "./fesm2015/core.js",

"es2015": "./fesm2015/core.js",

"esm2015": "./esm2015/core.js",

"fesm2015": "./fesm2015/core.js",

...

}

For more information about the npm package format, see the [Angular Package Format spec](https://goo.gl/jB3GVv).

Removed APIs[link](https://angular.io/guide/deprecations#removed-apis)

The following APIs have been removed starting with version 9.0.0\*:

| Package | API | Replacement | Notes |
| --- | --- | --- | --- |
| @angular/core | [Renderer](https://v8.angular.io/api/core/Renderer) | [Renderer2](https://angular.io/api/core/Renderer2) | [Migration guide](https://angular.io/guide/migration-renderer) |
| @angular/core | [RootRenderer](https://v8.angular.io/api/core/RootRenderer) | [RendererFactory2](https://angular.io/api/core/RendererFactory2) | none |
| @angular/core | [RenderComponentType](https://v8.angular.io/api/core/RenderComponentType) | [RendererType2](https://angular.io/api/core/RendererType2) | none |
| @angular/core | [WtfScopeFn](https://v8.angular.io/api/core/WtfScopeFn) | none | v8 |
| @angular/core | [wtfCreateScope](https://v8.angular.io/api/core/wtfCreateScope) | none | v8 |
| @angular/core | [wtfStartTimeRange](https://v8.angular.io/api/core/wtfStartTimeRange) | none | v8 |
| @angular/core | [wtfEndTimeRange](https://v8.angular.io/api/core/wtfEndTimeRange) | none | v8 |
| @angular/core | [wtfLeave](https://v8.angular.io/api/core/wtfLeave) | none | v8 |
| @angular/common | DeprecatedI18NPipesModule | [CommonModule](https://angular.io/api/common/CommonModule#pipes) | none |
| @angular/common | DeprecatedCurrencyPipe | [CurrencyPipe](https://angular.io/api/common/CurrencyPipe) | none |
| @angular/common | DeprecatedDatePipe | [DatePipe](https://angular.io/api/common/DatePipe) | none |
| @angular/common | DeprecatedDecimalPipe | [DecimalPipe](https://angular.io/api/common/DecimalPipe) | none |
| @angular/common | DeprecatedPercentPipe | [PercentPipe](https://angular.io/api/common/PercentPipe) | none |
| @angular/forms | [NgFormSelectorWarning](https://v8.angular.io/api/forms/NgFormSelectorWarning) | none | none |
| @angular/forms | [ngForm](https://angular.io/api/forms/NgForm) element selector | [ng-form](https://angular.io/api/forms/NgForm) element selector | none |
| @angular/service-worker | versionedFiles | files | In the service worker configuration file ngsw-config.json, replace versionedFiles with files. See [Service Worker Configuration](https://angular.io/guide/service-worker-config#assetgroups). |

\*To see APIs removed in version 8, check out this guide on the [version 8 docs site](https://v8.angular.io/guide/deprecations#removed).

@angular/http[link](https://angular.io/guide/deprecations#angularhttp)

The entire [@angular/http](http://v7.angular.io/api/http) package has been removed. Use [@angular/common/http](https://angular.io/api/common/http) instead.

The new API is a smaller, easier, and more powerful way to make HTTP requests in Angular. The new API simplifies the default ergonomics: There is no need to map by invoking the .json() method. It also supports typed return values and interceptors.

To update your apps:

* Replace HttpModule with [HttpClientModule](https://angular.io/api/common/http/HttpClientModule) (from [@angular/common/http](https://angular.io/api/common/http)) in each of your modules.
* Replace the Http service with the [HttpClient](https://angular.io/api/common/http/HttpClient) service.
* Remove any map(res => res.json()) calls. They are no longer needed.

For more information about using @angular/common/[http](https://angular.io/api/common/http), see the [HttpClient guide](https://angular.io/guide/http).

| @angular/[http](https://angular.io/api/common/http) | Closest replacement in @angular/common/[http](https://angular.io/api/common/http) |
| --- | --- |
| BaseRequestOptions | [HttpRequest](https://angular.io/api/common/http/HttpRequest) |
| BaseResponseOptions | [HttpResponse](https://angular.io/api/common/http/HttpResponse) |
| BrowserXhr |  |
| Connection | [HttpBackend](https://angular.io/api/common/http/HttpBackend) |
| ConnectionBackend | [HttpBackend](https://angular.io/api/common/http/HttpBackend) |
| CookieXSRFStrategy | [HttpClientXsrfModule](https://angular.io/api/common/http/HttpClientXsrfModule) |
| Headers | [HttpHeaders](https://angular.io/api/common/http/HttpHeaders) |
| Http | [HttpClient](https://angular.io/api/common/http/HttpClient) |
| HttpModule | [HttpClientModule](https://angular.io/api/common/http/HttpClientModule) |
| Jsonp | [HttpClient](https://angular.io/api/common/http/HttpClient) |
| JSONPBackend | [JsonpClientBackend](https://angular.io/api/common/http/JsonpClientBackend) |
| JSONPConnection | [JsonpClientBackend](https://angular.io/api/common/http/JsonpClientBackend) |
| JsonpModule | [HttpClientJsonpModule](https://angular.io/api/common/http/HttpClientJsonpModule) |
| QueryEncoder | [HttpUrlEncodingCodec](https://angular.io/api/common/http/HttpUrlEncodingCodec) |
| ReadyState | [HttpBackend](https://angular.io/api/common/http/HttpBackend) |
| Request | [HttpRequest](https://angular.io/api/common/http/HttpRequest) |
| RequestMethod | [HttpClient](https://angular.io/api/common/http/HttpClient) |
| RequestOptions | [HttpRequest](https://angular.io/api/common/http/HttpRequest) |
| RequestOptionsArgs | [HttpRequest](https://angular.io/api/common/http/HttpRequest) |
| [Response](https://angular.io/api/common/http/HttpEventType#Response) | [HttpResponse](https://angular.io/api/common/http/HttpResponse) |
| ResponseContentType | [HttpClient](https://angular.io/api/common/http/HttpClient) |
| ResponseOptions | [HttpResponse](https://angular.io/api/common/http/HttpResponse) |
| ResponseOptionsArgs | [HttpResponse](https://angular.io/api/common/http/HttpResponse) |
| ResponseType | [HttpClient](https://angular.io/api/common/http/HttpClient) |
| URLSearchParams | [HttpParams](https://angular.io/api/common/http/HttpParams) |
| XHRBackend | [HttpXhrBackend](https://angular.io/api/common/http/HttpXhrBackend) |
| XHRConnection | [HttpXhrBackend](https://angular.io/api/common/http/HttpXhrBackend) |
| XSRFStrategy | [HttpClientXsrfModule](https://angular.io/api/common/http/HttpClientXsrfModule) |

| @angular/[http](https://angular.io/api/common/http)/testing | Closest replacement in @angular/common/[http](https://angular.io/api/common/http)/testing |
| --- | --- |
| MockBackend | [HttpTestingController](https://angular.io/api/common/http/testing/HttpTestingController) |
| MockConnection | [HttpTestingController](https://angular.io/api/common/http/testing/HttpTestingController) |