Supported Public API Surface of Angular

Our semver, timed-release cycle and deprecation policy currently applies to these npm packages:

- `@angular/animations`
- `@angular/core`
- `@angular/common`
- `@angular/elements`
- `@angular/forms`
- `@angular/platform-browser`
- `@angular/platform-browser-dynamic`
- `@angular/platform-server`
- `@angular/upgrade`
- `@angular/router`
- `@angular/service-worker`

One intentional omission from this list is `@angular/compiler`, which is currently considered a low level api and is subject to internal changes. These changes will not affect any applications or libraries using the higher-level apis (the command line interface or JIT compilation via `@angular/platform-browser-dynamic`). Only very specific use-cases require direct access to the compiler API (mostly tooling integration for IDEs, linters, etc). If you are working on this kind of integration, please reach out to us first.

Package `@angular/bazel` is currently an Angular Labs project and not covered by the public API guarantees.

Additionally only the command line usage (not direct use of APIs) of `@angular/compiler-cli` is covered.

Other projects developed by the Angular team like angular-cli, Angular Material, will be covered by these or similar guarantees in the future as they mature.

Within the supported packages, we provide guarantees for:

- symbols exported via the main entry point (e.g. `@angular/core`) and testing entry point (e.g. `@angular/core/testing`). This applies to both runtime/JavaScript values and TypeScript types.
- symbols exported via global namespace `ng` (e.g. `ng.core`)
- bundles located in the `bundles/` directory of our npm packages (e.g. `@angular/core/bundles/core.umd.js`)

We explicitly don't consider the following to be our public API surface:

- any file/import paths within our package except for the `/`, `/testing` and `/bundles/*` and other documented package entry-points.
- constructors of injectable classes (services and directives) - please use DI to obtain instances of these classes
- any class members or symbols marked as `private`, or prefixed with underscore (`_`), barred latin o (`ɵ`), and double barred latin o (`ɵɵ`).
- extending any of our classes unless the support for this is specifically documented in the API docs
- the contents and API surface of the code generated by Angular's compiler (with one notable exception: the existence and name of `NgModuleFactory` instances exported from generated code is guaranteed)

Our peer dependencies (such as TypeScript, Zone.js, or RxJS) are not considered part of our API surface, but they are included in our SemVer policies. We might update the required version of any of these dependencies in minor releases if the update doesn't cause breaking changes for Angular applications. Peer dependency updates that result in non-trivial breaking changes must be deferred to major Angular releases.

Extending Angular classes

All classes in Angular's public API are `final` (they should not be extended) unless explicitly stated in the API documentation.

Extending such `final` classes is not supported, since protected members and internal implementation may change outside of major releases.

Golden files

Angular tracks the status of the public API in a *golden file*, maintained with a tool called the *public API guard*. If you modify any part of a public API in one of the supported public packages, the PR can fail a test in CI with an error message that instructs you to accept the golden file.

The public API guard provides a Bazel target that updates the current status of a given package. If you add to or modify the public API in any way, you must use yarn to execute the Bazel target in your terminal shell of choice (a recent version of `bash` is recommended).

```
yarn bazel run //packages/<modified_package>:<modified_package>_api.accept
```

Using yarn ensures that you are running the correct version of Bazel. (Read more about building Angular with Bazel here.)

Here is an example of a Circle CI test failure that resulted from adding a new allowed type to a public property in `core.d.ts`. Error messages from the API guard use `git-diff` formatting.

```
FAIL: //packages/core:core_api (see
/home/circleci/.cache/bazel/_bazel_circleci/9ce5c2144ecf75d11717c0aa41e45a8d/execroot/angular/bazel-
out/k8-fastbuild/testlogs/packages/core/core_api/test_attempts/attempt_1.log)
INFO: From Action packages/compiler-cli/ngcc/test/fesm5_angular_core.js:
[BABEL] Note: The code generator has deoptimised the styling of /b/f/w/bazel-out/k8-
fastbuild/bin/packages/core/npm_package/fesm2015/core.js as it exceeds the max of 500KB.
FAIL: //packages/core:core_api (see
/home/circleci/.cache/bazel/_bazel_circleci/9ce5c2144ecf75d11717c0aa41e45a8d/execroot/angular/bazel-
out/k8-fastbuild/testlogs/packages/core/core_api/test.log)

FAILED: //packages/core:core_api (Summary)

/home/circleci/.cache/bazel/_bazel_circleci/9ce5c2144ecf75d11717c0aa41e45a8d/execroot/angular/bazel-
out/k8-fastbuild/testlogs/packages/core/core_api/test.log

/home/circleci/.cache/bazel/_bazel_circleci/9ce5c2144ecf75d11717c0aa41e45a8d/execroot/angular/bazel-
out/k8-fastbuild/testlogs/packages/core/core_api/test_attempts/attempt_1.log
INFO: From Testing //packages/core:core_api:
==================== Test output for //packages/core:core_api:
/b/f/w/bazel-out/k8-
fastbuild/bin/packages/core/core_api.sh.runfiles/angular/packages/core/npm_package/core.d.ts(7,1):
error: No export declaration found for symbol "ComponentFactory"
--- goldens/public-api/core/core.d.ts   Golden file
+++ goldens/public-api/core/core.d.ts   Generated API
@@ -563,9 +563,9 @@
     ngModule: Type<T>;
     providers?: Provider[];
 }

-export declare type NgIterable<T> = Array<T> | Iterable<T>;
+export declare type NgIterable<T> = Iterable<T>;

 export declare interface NgModule {
     bootstrap?: Array<Type<any> | any[]>;
     declarations?: Array<Type<any> | any[]>;


If you modify a public API, you must accept the new golden file.


To do so, execute the following Bazel target:
  yarn bazel run //packages/core:core_api.accept
```