Angular 2 Dart Transformer

Author: tjblasi@google.com

This doc is public

# Summary

The purpose of this doc is to explain the goals and design of the Angular 2 Dart’s code generation step. The mechanism used will be a [Pub Transformer](https://www.dartlang.org/tools/pub/transformers/), which is not related to the transformations which generate the Dart and Js flavors of Angular 2 from the Typescript codebase.

## Intended Audience: Angular 2 Developers

This is meant as documentation for those *developing* or *maintaining* the transformer. See [the wiki](https://github.com/angular/angular/wiki/Angular-2-Dart-Transformer) for directions on *using* the transformer in an Angular 2 app.

[Summary](#h.pouglkue930c)

[Intended Audience: Angular 2 Developers](#h.eky9z0uh1jvm)

[Links](#h.2u39tfq8s19r)

[Goal](#h.nuj6j1s9s9d0)

[Subgoals](#h.23x31qjw01uk)

[Design](#h.mppehk7k2rwb)

[Phase 1 - ReflectionRemover](#h.s7wjrm37m7v1)

[Note about the future](#h.skcg4y782qg0)

[Current (Jun 2015) Situation](#h.3dtlmali68d7)

[Implementation Overview](#h.eqesim6tm5ia)

[Phase 2 - DirectiveProcessor](#h.cqi9x37z9wg7)

[Implementation Overview](#h.g1lc63v8nj3l)

[Phase 3 - EmptyNgDepsRemover](#h.kivwghxsqab5)

[Implementation Overview](#h.kqje77s66ref)

[Phase 4a - Directive Linker](#h.rctduviwhjiz)

[Implementation Overview](#h.rp4w21avn39j)

[Phase 4b - DirectiveMetadataExtractor](#h.pm554baptl6l)

[Implementation Overview](#h.1zpo60gpschx)

[Phase 5 - BindGenerator](#h.i75cr5z0ti83)

[Example](#h.lm8u4pw6otrq)

[Implementation Overview](#h.pwrqlo9imj5p)

[Phase 6 - Template Compiler](#h.x6wt7rjthjsa)

[Implementation Overview](#h.tu2bczmewiad)

[Omissions](#h.ug1g838jiyn8)

[Tree Shake Transformed Angular 2 Apps](#h.da8qiq4upwe9)

[Proposed Solutions](#h.7q95r1wdz6az)

[Generate .ng\_deps.dart files per-class instead of per-file](#h.dli9oysrzxj3)

[Difficulties](#h.60s0xwtxoekl)

[Add a resolved step that walks from bootstrap calls](#h.n3cvqdq3pl1t)

[Difficulties](#h.i1ey04u5yjnv)

[Debug Transformation Failures](#h.2w0m23i2q27z)

[Current Situation](#h.iqkp2kk3js2y)

[Proposed Solution](#h.5g03e94ht508)

[Additional Considerations](#h.m9v3c3kbwqy5)

# Links

* [Current implementation](https://github.com/angular/angular/tree/master/modules/angular2/src/transform) (incomplete)
* [Angular 2 Dart Transform Example](https://docs.google.com/document/d/1qYkX7ekBbSdZe1BEp0ECsFtVRz_IQTTwlFfI10b_a-Y/edit?usp=sharing)

# Goal

Angular 2 uses reflection under the covers to provide dependency injection (DI) and change detection. While reflective access is very convenient, it can be slow and defeats Dart’s tree shaking ability, so we plan to create a [*Pub Transformer*](https://www.dartlang.org/tools/pub/transformers/)[[1]](#footnote-0) to generate code removing the need for reflection.

## Subgoals

* Make the generated code easy to understand and reason about.
* Allow effective code splitting (ensure this scheme works with [the routing-based code splitting proposal](https://docs.google.com/document/d/184KvGL9Kz5ycIRlvo490dDaa876VSvv9YRDdQvgGYNs/edit)).
* Good developer experience ⟶ quick transformations ⟶ Avoid resolving the Angular 2 app.
* Allow use of the DI system independent of the rest of Angular 2.

# Design

The Angular 2 Dart transformer will operate in several phases and in a few different modes to generate clean, understandable, tree-shakable code.

This is still under development and may change in the future. All “Implementation Details” are as of [efab032](https://github.com/angular/angular/commit/efab03274f07578347a1b337b58204ec7845edf4).

## Phase 1 - ReflectionRemover

### Note about the future

There are plans to change the details of calling bootstrap such that the instantiation & removal of ReflectionCapabilities is no longer user-visible. See [issue #2304](https://github.com/angular/angular/issues/2304) for details.

### Current (Jun 2015) Situation

This phase finds the instantiation of ReflectionCapabilities in the Angular 2 app and replaces it with a call to the necessary .ng\_deps.dart file to prime the reflection library such that dart:mirrors is not needed in the Angular 2 app.

This phase relies on syntactic parsing only.

### Implementation Overview

[remove\_reflection\_capabilities.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/reflection_remover/remove_reflection_capabilities.dart) defines removeReflectionCapabilities, which parses the file passed in as reflection\_entry\_point, searching for a call to reflector.reflectionCapabilities = new ReflectionCapabilities(); It then comments out this line as well as the import bringing in ReflectionCapabilities, removing the transitive import of dart:mirrors, a major source of bloat in the generated Javascript code.

Those calls are replaced with a call to the initReflector method created from the file passed in as entry\_point, ensuring all necessary reflection code is initialized.

## Phase 2 - DirectiveProcessor

This phase parses each .dart file in a package and generates a corresponding .ng\_deps.dart file. That file defines an idempotent function, initReflector, which registers any classes needed by the Injector with the reflection library via [registerType](https://github.com/angular/angular/blob/master/modules/angular2/src/reflection/reflector.ts#L27).

This phase relies on syntactic parsing only.

### Implementation Overview

[rewriter.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/directive_processor/rewriter.dart) defines createNgDeps which parses <file>.dart file and processes it with CreateNgDepsVisitor. This visitor copies over most directives, then creates a top-level, idempotent method initReflector. It then processes ClassDeclarations, generating registerType calls if the class is annotated @Injectable (or subtypes, as determined by [AnnotationMatcher](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/common/annotation_matcher.dart)). Additional visitors defined in [visitors.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/directive_processor/visitors.dart) consume the ClassDeclaration and generate the values that are passed to registerType.

In this step, templates referenced via templateUrl are inlined as template strings.

The resulting file is output as <file>.ng\_deps.dart.

## Phase 3 - EmptyNgDepsRemover

The DirectiveProcessor generates a .ng\_deps.dart file for every .dart input, but for many files this is unnecessary. This phase removes .ng\_deps.dart files that are not used.

See [issue #1929](https://github.com/angular/angular/issues/1929) for motivation and details.

This phase relies on syntactic parsing only.

### Implementation Overview

[linker.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/directive_linker/linker.dart) defines an [isNecessary](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/directive_linker/linker.dart) method which determines whether a .ng\_deps.dart file is required for the app. If it determines that the file is unnecessary, it is removed from the app.

## Phase 4a - Directive Linker

This phase processes the .ng\_deps.dart files produced by the DirectiveProcessor phase and modifies each such that it calls any of its dependencies’ registration methods.

This phase relies on syntactic parsing only.

### Implementation Overview

[linker.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/directive_linker/linker.dart) defines linkNgDeps which parses <file>.ng\_deps.dart (using [NgDeps.parse](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/common/ng_deps.dart)) and iterates over its imports. For each import i.dart, it then determines if i.ng\_deps.dart exists. If so, it generates import i.ng\_deps.dart as i# and a call to i#.initReflector() in <file>.ng\_deps.dart#initReflector, such that dependencies of classes defined in <file>.dart will also be registered.

Generating multiple calls to initReflector is safe as these are idempotent (see DirectiveProcessor).

## Phase 4b - DirectiveMetadataExtractor

This phase processes the .ng\_deps.dart files produced by the DirectiveProcessor phase and extracts metadata about any Directives defined in them.

This phase relies on syntactic parsing only.

### Implementation Overview

[extractor.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/directive_metadata_extractor/extractor.dart) defines extractDirectiveMetadata which parses <file>.ng\_deps.dart files (using [NgDeps.parse](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/common/ng_deps.dart)). It then processes the annotations of any [RegisteredType](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/common/registered_type.dart) and, if available, extracts [DirectiveMetadata](https://github.com/angular/angular/blob/master/modules/angular2/src/render/api.ts#L125) which is necessary for the TemplateCompiler step. The DirectiveMetadata is then converted to JSON (via [directiveMetadataToMap](https://github.com/angular/angular/blob/master/modules/angular2/src/render/dom/convert.ts) and the dart:convert package) and output as <file>.ng\_meta.json.

## Phase 5 - BindGenerator

This phase processes the .ng\_deps.dart files produced in the DirectiveProcessor phase and modifies each such that any ‘properties’ values in our Directive annotations are registered with the reflection library via [registerSetters](https://github.com/angular/angular/blob/master/modules/angular2/src/reflection/reflector.ts#L35).

This phase relies on syntactic parsing only.

### Example

@Directive(..., properties: [<class\_prop\_or\_getter>])

class MyDirective {}

generates…

reflector.registerSetters({‘<class\_prop\_or\_getter>’:

(o, v) => o.<class\_prop\_or\_getter> = v});

### Implementation Overview

[generator.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/bind_generator/generator.dart) defines createNgSetters which parses <file>.ng\_deps.dart (using [NgDeps.parse](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/common/ng_deps.dart)) and iterates over the annotations generated in the DirectiveProcessor phase. When a property named properties is found, its value is fed to ExtractSettersVisitor in [visitor.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/bind_generator/visitor.dart), which extracts the name of the necessary setter. Using those values, calls to registerSetters are then generated and added to initReflector(in <file>.ng\_deps.dart).

## Phase 6 - Template Compiler

Processes HTML templates to determine what information is needed to run statically.

* Generate Getters & Setters. See [Github Issue #501](https://github.com/angular/angular/issues/501) for details.
  + Process all inlined templates using the Angular 2 compiler pipeline mentioned in [#503](https://github.com/angular/angular/issues/503).
  + Record all calls that would be made through the reflection mechanism and generate calls to [registerGetters](https://github.com/angular/angular/blob/master/modules/angular2/src/reflection/reflector.ts#L31), [registerMethods](https://github.com/angular/angular/blob/master/modules/angular2/src/reflection/reflector.ts#L39), and [registerSetters](https://github.com/angular/angular/blob/master/modules/angular2/src/reflection/reflector.ts#L35) alongside the registerType calls in the .ng\_deps.dart file.
* Generate change detector classes. See [Github Issue #502](https://github.com/angular/angular/issues/502) for details.
  + Use the output from the compiler pipeline to create a dedicated [AbstractChangeDetector](https://github.com/angular/angular/blob/master/modules/angular2/src/change_detection/abstract_change_detector.ts) class for each component.

### Implementation Overview

[generator.dart](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/template_compiler/generator.dart) defines processTemplates which parses <file>.ng\_deps.dart (using [NgDeps.parse](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/common/ng_deps.dart)) and iterates over the annotations generated in the DirectiveProcessor phase. If a @View annotation is found, processes it with the Angular 2 [DomCompiler](https://github.com/angular/angular/blob/master/modules/angular2/src/render/dom/compiler/compiler.ts) and records where reflection is needed. Using that information, calls to register{Getters,Methods,Setters} are generated into initReflector.

The [DomCompiler](https://github.com/angular/angular/blob/master/modules/angular2/src/render/dom/compiler/compiler.ts) outputs a [ProtoViewDto](https://github.com/angular/angular/blob/master/modules/angular2/src/render/api.ts#L95) object. That object is converted to a [ChangeDetectorDefinition](https://github.com/angular/angular/blob/master/modules/angular2/src/change_detection/interfaces.ts), which is then passed to a [change\_detector\_codegen.dart#Codegen](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/template_compiler/change_detector_codegen.dart) object which creates the [AbstractChangeDetector](https://github.com/angular/angular/blob/master/modules/angular2/src/change_detection/abstract_change_detector.ts) subclass for its Component in the .ng\_deps.dart file. A call to register the ChangeDetector with the system is then added to the bottom of initReflector in the .ng\_deps.dart file.

# Omissions

## Tree Shake Transformed Angular 2 Apps

The above-described scheme [follows the import graph](#h.rp4w21avn39j) and therefore will not optimally remove unused code in some situations.

### Proposed Solutions

#### Generate .ng\_deps.dart files per-class instead of per-file

Rather than having one .ng\_deps.dart file per .dart file, generate one .ng\_deps.dart file per class in that file.

##### Difficulties

The current scheme does not require knowledge of which types are defined where, making it very simple -- each file just needs to know its imports. We would need additional type and dependency info to property link .ng\_deps.dart files together.

#### Add a resolved step that walks from bootstrap calls

This involves finding all calls to bootstrap and determining what types are visible from there, removing all unnecessary imports in our generated .ng\_deps.dart files.

##### Difficulties

We will potentially have to resolve the entire Angular 2 app, which may be very slow. This may be palatable as a deployment step but not as a development step.

## Debug Transformation Failures

There will be cases where the above transformations fail. Since most of these rely on syntactic transformations, they will be particularly vulnerable to naming-based confusion (for example, if an app defines its own class named “Injectable” or “Component”.

### Current Situation

The transformer currently uses the [AnnotationMatcher](https://github.com/angular/angular/blob/master/modules/angular2/src/transform/common/annotation_matcher.dart) class to determine whether a class needs to be registered with the system. This syntactically checks annotation names and imports, which means that any new annotations or re-exported annotations will not be recognized by the system. Users must register any new annotations with the transformer as described on the [wiki page](https://github.com/angular/angular/wiki/Angular-2-Dart-Transformer).

### Proposed Solution

Have a slow, resolved mode of the transformer that provides feedback on what the basic transformer is doing. We can give information like:

* Tuning for the basic transformer
  + How many unnecessary types are being retained (& how to improve)
  + Whether there are any app defined tokens that are not specifically listed in the transformer.
  + Any additional annotations that should be registered with the transformer.
* Better explained error messages
  + “Could not find type T” => “Type T defined in t.dart is not being registered by t.ng\_deps.dart. Are you missing an import or a reference in componentServices?”

# Additional Considerations

* Source maps go from source .dart to generated .js, and we sit right in the middle
  + ∴ We **cannot** change line numbers
  + ∴ Where possible we should avoid changing offsets.

1. Where “Pub” is the Dart tool allowing code generation, deployment processing, etc. See [pub.dartlang.org](http://pub.dartlang.org). [↑](#footnote-ref-0)