Discovering injectables in Angular 2

Angular 2 needs a way to preserve type factories and annotation data for types that are:

* used by the application
* injectable via DI

# Proposal #1: tree shaking + @Injectable

* (Dart team) Provides a way for analyzer and/or dart2js produce the list of types retained after tree shaking. We use this to get a list of types used by the application.
* (Angular team) Builds a transformer that produces a list of all types that have @Injectable annotation on them. These determine types that are available for injection via DI.
* (Angular team) Extract type factories and annotation data on the set of types that’s an intersection of two type lists from above

Example:

|  |
| --- |
| main() {  var injector = new Injector([Foo]);  injector.get(Foo).printDate(new DateTime.now());  }  @Injectable  class Foo {  printDate(DateTime dt) { print(dt); }  }  @Injectable  class Bar {  } |

In the example above the list of classes used by the application is [Foo, DateTime], and the list of injectables is [Foo, Bar]. The intersection is [Foo] and so Foo is the only class that we build stubs for.

## Disadvantages

One disadvantage of this approach is that there’s a risk that we generate stubs for classes that are not injected by the DI framework.

Example:

|  |
| --- |
| main() {  new Foo().hi();  }  @Injectable  class Foo {  hi() {...}  } |

In this example we would generate stubs for Foo but we don’t need them.

# Proposal #2: crawl @Component annotation tree

Example:

|  |
| --- |
| bootstrap(MyApp);  @Component(  services: [  const Binding(Foo, toImplementation: FooBar),  const Binding(Baz, toImplemantation: BazImpl)  ]  )  @Template(  directives: [For]  )  class MyApp {  MyApp(Foo foo);  }  class FooBar implements Foo {  FooBar(Baz baz) {}  }  @AngularApp(MyApp)  myApp() {  bootstrap(MyApp);  } |

## Disadvantages

A disadvantage of this approach is that automatic type factory generation is available only to bindings created declaratively (or maybe this is an advantage). Also, the DI framework must be aware of the @Component and @Template annotations, which are Angular-specific.

Angular’s own internal classes, such as Compiler, are injectable, but not reachable via either @Component or @Template annotations. Those will have to be special-cased.

# Other approaches

* Retain all @Injectable classes
  + This will retain classes at runtime that are not actually used by the application. In the example above Bar would be retained needlessly.
* Extract type factories and annotations from classes that are used by the app.
  + This one’s silly, but I just wanted to mention it.