AngularJS 2.0 - Documentation Generation

*Status: (Draft)*

*Authors:* [*pete@bacondarwin.com*](mailto:pete@bacondarwin.com)

# Objective

The objective of this design document is to capture what needs to be implemented to generate documentation for the various modules and projects that will comprise AngularJS 2.0.

In addition it will consider how users of AngularJS can be supported in generating documentation their own Angular applications and libraries.

# Background

Documentation is a key aspect of the success of any framework or library, whether it will be published publicly or only used in a private environment. Even applications that are not published as libraries of code need to be maintained by developers.

Access to relevant and up to date documentation significantly improves the speed at which developers can comprehend and navigate the code of a project. It also makes it much easier to bring new developers into a project as they can quickly and independently orientate themselves and find information about the code.

# Prior Art

We have a pretty good web application at <http://docs.angularjs.org>, which provides access to the documentation for AngularJS 1.x.

The original documentation for AngularJS was generated with a utility (ngdoc.js), which lived inside the angularjs repository. While it did the job it was becoming more and more difficult to maintain and improve.

At the start of 2014, we migrated to a new home-grown tool Dgeni (<http://github.com/angular/dgeni>). Dgeni is more modular and configurable than the old ngdoc.js utility. For the purposes of the AngularJS 1.x documentation, Dgeni has been configured to produce basically the same web application as was being generated before.

The tool is very versatile but at the moment we have a monolithic approach where the code is assumed to be stored in a single project. The modularity of AngularJS 2.0 means that this single project view is no longer valid. We must evolve the documentation generation and web app to support multiple sources of documentation.

Another goal of AngularJS 2.0 is to provide a comprehensive development suite, which provides support for things like testing and benchmarking, in addition to the code itself. We should also provide support for developers to generation documentation for their own Angular applications and libraries (e.g. services and directives).

It would be interesting to consider standardizing the format for AngularJS documentation so that libraries of AngularJS documentation from diverse sources could be federated into a single documentation web application - a kind of AngularJS Documentation Library.

# Detailed Design

## Use Cases

There are a number of projects that are either using Dgeni already or could/should be using it in the near future.

* Protractor
* Ionic Framework
* angular-translate
* sofa.io
* dgeni
* dgeni-packages

## Documenting Dgeni

The Dgeni tool should be eating its own dog food and using Dgeni to generate documentation for itself. To achieve this goal a number of things need to happen:

* Documentation needs to be written for the dgeni and dgeni-packages projects - i.e. the actual text content of the documentation.
* One or more dgeni packages need to be improved/built to support generating documentation for Dgeni and Dgeni packages.
  + **jsdocs** package - we need to improve this package to be able to extract/cope with documentation in comments of node.js style JavaScript.
  + **dgeni-package** package - we need a new package that knows how to extract dgeni package specific information from dgeni packages. This will understand things like processors, tags, templates, etc.
* A strategy for serving up the generated Dgeni documentation.

## Dgeni Document Processors

Dgeni works by reading sources and then applying a pipeline of “processors” to this set of documents. A number of standard processors have been developed that support documentation of the current AngularJS 1.x code. In order to support AngularJS 2.0 and 3rd party AngularJS code, we will need to provide additional processors. Here is a list of some of the things that these processors will need to be able to handle:

* ES6 specific concepts such as modules and type annotations.
* The current ngdocs package supports Angular concepts such as directives, filters and services, but does not have a good story for documenting other app specific Angular concepts such as controllers, routes, http interceptors, decorators and so on.
* …

## Documentation Web App

The current documentation web app is very specific to the AngularJS 1.x project. It is also hosted inside the project. We should create a new web app that lives separately from the AngularJS project. This doc app should only display documentation. The documentation that it displays would be provided to it as data. This may be a bunch of partial HTML files or may also include other meta data. If the definition of this data was well defined then it could support viewing multiple versions of the docs and to display documentation from a variety of disparate projects.

# Caveats

…

# Security Considerations

…

# Performance Considerations / Test Strategy

…

# Work Breakdown

…