This document is now obsolete. Please use this template from the angular.io folder instead.

Design Doc Template

author: XXX
last update: 2016-12-19
status: [draft | in review | approved | implemented | obsolete]
doc url: https://goo.gl/pngvr0
feedback url: https://goo.gl/6cwSsy
based on template: *v1.0.0*
This document is published to the web as part of the public Angular Design Docs folder

Objective

In a few sentences, describe the key objectives.

Background

Stuff one needs to know to understand this doc: motivating examples, previous versions and problems, links to related projects/design docs, etc. You should mention related work if applicable. This is background; do not write about ideas to solve problems here.

Prior Art

List of existing solutions and their corresponding strong/weak points.

Detailed Design

Details on how you'll implement. Should be in sufficient detail for other engineers to materially comment on structure to affect the end result.

Caveats

You may need to describe what you did not do or why simpler approaches don't work. Mention other things to watch out for (if any).

Security Considerations

How you'll be secure

Performance Considerations / Test Strategy

How will you be fast?

Is there an impact on the payload size due to this change?

Documentation Plan

What types of documentation are required? [API | Concepts | Cookbooks | etc.]
Which folks will be needed to implement the docs?

Style Guide Impact

Does this change have an impact on the Angular Style Guide? Do any updates need to be made?

Public API Surface Impact

Are you adding new public APIs? If so, what's the change? (method signature changes, new classes/interfaces, new npm packages, config file properties or new config files altogether)

Are the new APIs consistent with the existing APIs? Do they follow naming conventions and use correct terminology?

Developer Experience Impact

How will this change impact developer experience and workflows?

Are we adding new tools developers need to learn? Are we asking developers to change their workflows in any way?

Are we placing any new constraints on how the developer should write the code to be compatible with this change?

Internationalization / Localization

Accessibility

Breaking Changes

Does this feature create breaking changes? Who will be affected? How can we make this easy on users to adopt?

Deprecations

Can (or should) any existing APIs, tools or documentation be deprecated as a result of this change?

When will the deprecated APIs be removed?

Rollout Plan

- How do we minimize impact to users? Are there migration tools, docs, GDE prep, etc. that would help?
- How do we maximize adoption and benefit to users? Are there other parts of the ecosystem that should take advantage of it?
- Should we have tools partners prepared/updated before general availability?

Maintenance Plan

How do you plan to maintain this feature for the next X years.

Can the APIs be cleanly evolved? What did you do to prevent big breaking changes should the implementation or feature set change based on the feedback we get within the next year.

If this is a service, what's the update and monitoring strategy?

Work Breakdown

Description of development phases and approximate time estimates.