# ng-material for Angular 2

The purpose of this document is to serve as a plan / reference / map / manifesto for building ng-material 2.0. The intended audience is engineers that work on ng-material.

## The goal of ng-material

Our goal is to build a set of **high-quality** UI components built with Angular 2 and TypeScript, following the Material Design spec, that are consumable by both JavaScript and Dart. These components will serve as an example of how to write Angular code following best practices.

### What do we mean by "high-quality"?

* Internationalized and accessible so that *all* users can use them.
* Straightforward APIs that don't confuse developers.
* Behave as expected across a wide variety of use-cases without bugs.
* Behavior is well-tested with both unit and integration tests.
* Customizable within the bounds of the Material Design specification.
* Performance cost is minimized
* Code is clean and well-documented to serve as an example for Angular devs.

### Feedback for Angular 2

ng-material will likely test the limits of what Angular itself can do and will provide valuable feedback for improving the framework itself. Whenever someone runs into an Angular issue while working on ng-material, we should work with the core Angular team to try to improve the framework.

## The parts that compose ng-material

### The components

These are components that users can combine to build modern Material Design sites and applications. Components can range in complexity from very simple (button, checkbox) to highly complex (autocomplete, datepicker).

#### Prefer more focused, granular components vs. more complex, configurable components

Example: configuring one component

|  |
| --- |
| <md-button>Basic button</md-button>  <md-button class="md-fab">FAB</md-button>  <md-button class="md-icon-button">pony</md-button> |

Example: multiple components

|  |
| --- |
| <md-button>Basic button</md-button>  <md-fab>FAB</md-fab>  <md-icon-button>pony</md-icon-button> |

#### Be cautious with use of display: flex

* The [baseline calculation for flex elements](http://www.w3.org/TR/css-flexbox-1/#flex-baselines) is different than other display values, making it difficult to align flex elements with standard elements like input and button
  + Component outermost elements are never flex (block or inline-block)
  + Don't use `display: flex` on elements that will contain projected content.

#### Component completion checklist

* Behavior and appearance are consistent on all supported browsers and screen-readers.
* Fully accessible, including keyboard interaction, color contrast, and ARIA labels.
* Conforms to the [public specification](https://www.google.com/design/spec/material-design/introduction.html), modulo performance / complexity trade-offs.
* Themed via SCSS for both color and sizing.
* API surface is fully documented.
* Documentation and demo, including a demo set to RTL and a demo with custom styles.
* Robust unit tests, some [e2e](https://angular.github.io/protractor/#/) tests, and [benchpress](https://www.npmjs.com/package/angular-benchpress) tests for performance regressions.
* Code is fully-typed.
* Integration with [Web Tracing Framework](http://google.github.io/tracing-framework/) (advanced components only)
* Build any related testing mocks / stubs for consumers to test their use of the component.

### Style definitions

See the separate [design doc for themes and styling](https://docs.google.com/document/d/1UUQDh4Jw8eS5gf4iCLvluT6HUABMSZRt1wS5yViFOaE/edit#) for ng-material v2.

* SCSS style definitions are not only consumable by the ng-material components themselves, but by developers that want to consume the style definitions for color, shadows, animation, etc.
* SCSS source files will be bundled with each release so that users can build their own themes or style overrides.
* By default, the library defines absolutely no global styles. Any reset behavior must be scoped within a component on elements that are created / owned by ng-material. However, a set of optional, "easy-mode" style sheets can be made available for developers that want a quick start. This plays well with the idea of providing project templates.
* Support styles for Windows high-contrast mode. This is a low-effort task that makes a big difference for low-vision users.

#### Use lowest specificity possible

Always prioritize lower specificity over other factors. Most style definitions should consist of a single element or css selector plus necessary state modifiers. Avoid SCSS nesting for the sake of code organization. This will allow users to much more easily override styles.

Example: high specificity

|  |
| --- |
| md-calendar {  display: block;  .md-month {  display: inline-block;  .md-date.md-selected {  font-weight: bold;  }  }  } |

Example: low specificity

|  |
| --- |
| md-calendar {  display: block;  }  .md-calendar-month {  display: flex;  }  .md-calendar-date.md-selected {  font-weight: bold;  } |

### Docs & demos site

This is the gateway to the components and serves as the first impression to the user. The home of documentation, demos, customizable downloads, and other resources.

* Will be built with Angular 2 and ng-material 2. While not feasible at the start, the enitre angular.io site should eventually be built with Angular 2 and ng-material 2.
  + Determine our goal UX at the start and work towards that.
* Can be pointed to as an example of an Angular 2 application.
* Show total library size on homepage. Don't make people figure it out themselves.
* Offer a straightforward download for people that don't want to bower / npm / etc.
* Developers can, via the site, customize and download their own build of ng-material with only the components they need. Inspired by [jQuery UI](https://jqueryui.com/download/).
* Developers can, via the site, select a custom theme and download the corresponding CSS output. Inspired by [Material Design Lite](http://www.getmdl.io/customize/index.html).
* Code examples can be opened directly in CodePen (as today, javascript-only).
* URLs should be straightforward without unnecessary information in the route
* Individual pages can be crawled by search engines
* Shows component "report cards", grading on various metrics (a11y, i18n, perf, etc.)

### Starter seed templates

Set of skeleton applications / sites for common use-cases. Inspired by [Material Design Lite](http://www.getmdl.io/templates/index.html).

Include at least one "enterprise" example template (forms).

### Layout module

* The layout module should be completely uncoupled from the rest of the components. This is critical to supporting existing applications that want to adopt the ng-material components.
* Avoid attribute selectors due to [performance issues](https://github.com/angular/material/issues/3455) with Internet Explorer.
* Avoid SCSS loops for generating style permutations, preferring to apply as much styling via JS as possible to keep CSS payload size low.
* Focus first on supporting Material Design specification
* Need to benchmark performance vs. other popular layout systems
* Enables view changes based on device size.

## Process: attaining super high quality

### Critical user paths

These are extremely important paths users are going to take that will leave a lasting impression of the library. As implementation marches forward, these paths should be reexamined to make sure that the experience meets our standards. We should also be *asking* our actual users about how these workflows are for them. See [list of critical paths](https://docs.google.com/document/d/1pbzcm5mNTZPvhP044Ce35c8LfgDi_wgGSNfXsTmDG4I/edit).

### Library Structure

Principles for code organization and design.

#### Small, focused modules

Keeping modules to a single responsibility makes the code easier to test, consume, and maintain. ES6 modules offer a straightforward way to organize code into logical, granular units. Ideally, individual files are 200 - 300 lines of code.

#### Less is more

Once a feature is released, it never goes away. We should avoid adding features that don't offer high user value for price we pay both in maintenance, complexity, and payload size. When in doubt, leave it out.

#### Documentation

* Documentation and source code live in different files. This makes the source files cleaner and easier to grok, while also making it less intimidating for community members to contribute to the documentation.
* Documentation lives in the same directory as the source (with the TS, SCSS, HTML)
  + Have an "Edit these docs!" button
* Implement some kind of git hook to warn if the code has changed (with some epsilon) and the documentation has not.
* Write documentation with markdown. A build tool will convert to HTML and bundle into the documentation site.

### Code health

Code health is critical in general, and even more so for our project because developers will look to the source as an example of how to write code with Angular 2 code. There are steps we can take to keep health up.

#### Formal code review process

Code reviews offer an optimized way for multiple engineers to collaborate and end up with something that reflects the sum of their knowledge. Reviews look at design, functionality, complexity, style, and anything else that may warrant feedback.

All code going into ng-material master should be reviewed and approved by at least one other member. For bigger or more complex submissions, additional reviewers are a plus. Tooling can be used to streamline the review process.

#### Design review

For all major feature work, the implementor should put together a brief design doc and implementation plan to be reviewed by the rest of the team before starting coding in earnest (throwaway prototypes notwithstanding).

The most important thing here is to give team members an opportunity to give feedback on a design / idea.

#### Style guide adherence, clang-format, and TSLInt

The team will agree upon and formalize a style guide and use [clang-format](http://clang.llvm.org/docs/ClangFormat.html) to adhere to it. We will use [TSLint](http://palantir.github.io/tslint/) to capture some rules that aren't covered by clang-format.

#### Code coverage

A high code coverage does not automatically mean that the code is well-tested, but a *low* code coverage signals that tests are lacking. We will capture and monitor code coverage primarily to ensure the coverage level doesn't fall too low. The goal is to keep coverage above 90%.

#### Commenting

Comments that explain what some block of code does are nice; they can tell you something in less time than it would take to follow through the code itself.

Comments that explain *why* some block of code exists at all, or does something the way it does, are *invaluable*. The "why" is difficult, or sometimes impossible, to track down without seeking out the original author. When collaborators are in the same room, this hurts productivity. When collaborators are in different timezones, this can be devastating to productivity.

Example: not-very-useful comment

|  |
| --- |
| // Set default tabindex.  if (!$attrs['tabindex']) {  $element.attr('tabindex', '-1');  } |

Example: more useful comment

|  |
| --- |
| // Unless the user specifies so, the calendar should not be a tab stop.  // This is necessary because ngAria might add a tabindex to anything with an ng-model  // (based on whether or not the user has turned that particular feature on/off).  if (!$attrs['tabindex']) {  $element.attr('tabindex', '-1');  } |

#### Performance testing

We will use [benchpress](https://www.npmjs.com/package/angular-benchpress) to write performance tests with two goals: guarding against performance regressions, and comparing the ng-material components to similar components in other libraries (such as Polymer and Material Design Lite).

### Environments supported

Browsers: IE10+, latest two version of Chrome, Firefox, and Safari (including mobile Chrome and Safari).

Screen readers: NVDA and JAWS with IE / FF / Chrome, VoiceOver with Chrome / Safari, Talkback with Chrome.

Document usage and testing process for these screen readers.

## Release process

TBD - Will likely copy the Angular 2 process.

## Upgrading users from ng-material 1.X to 2.X.

TBD

## Completion Roadmap

Component *completion* will occur in the following stages. However, experimentation with more advanced components should happen in parallel with development of the simpler components in order to find places where Angular 2 makes progress difficult. The roadmap below is broken up in such a way that more advanced components are later in the pipeline in order to give Angular 2 itself more refinement time.

### Stage 1 - Simple components

Start off with simple components and focus on nailing down all aspects of what [completes](#id.d2v0htwhecj3) a component and dependencies that comprise the foundation of the library. Once this set of components is complete, ng-material 2.0 is in *prototype*.

#### Prerequisites

* Needs Angular 2 build process to be extracted into reusable piece
* Needs a working version of the new router

#### Requirements

* Docs & demos site

Documentation for theming the library

* Components: button, checkbox, radio, input, list, progress linear, progress circular, card, ink ripples

### Stage 2 - Intermediate components

Introduce need for gestures, dynamic component loading. Once this set of components is complete, ng-material 2.0 is in *alpha.*

#### Requirements

* Solution for handling gestures
* Pop-up components: dialog, toast, bottom-sheet, tooltips
* Container components: toolbar, side-nav, grid-list
* Form controls: switch, slider, icon
* Contributing guide

### Stage 3 - Advanced components

Complicated compositional components. At this stage, both Angular 2 and the team using it will be more developed. Once this set of components is complete, ng-material 2.0 is *beta*.

#### Requirements

* Components: Autocomplete, tabs, chips, select, menu, fab toolbar and speed-dial, virtual-repeat
* At least 3 starter / seed templates.Starter project & dev guide
* At least 3 starter / seed templates, can be based on the starter project.
* "Choose your own build" tool on ng-material site
* "Create your own theme" tool on ng-material site
* Upgrade guide and tooling for ng-material 1 to ng-material 2

### Stage 4 - Even more advanced components

These components will be released after the official release of ng-material.

Requirements

* Components: datepicker, data-table