Angular 1 to Angular 2 migration ideas  
(Deprecated See: [ng1-2-strategy](https://docs.google.com/document/d/1xvBZoFuNq9hsgRhPPZOJC-Z48AHEbIBPlOCBTSD8m0Y))

*Status: (Draft)*

*Authors: misko@google.com*

*This document is published to the web as part of the public* [*Angular Design Docs*](https://drive.google.com/#folders/0BxgtL8yFJbacUnUxc3l5aTZrbVk) *folder*

# Objective

Describe the strategies which we will use to minimize the porting distance between Angular 1 to Angular 2 simpler.

# Background

Angular 1 and Angular 2 have different syntax and semantics. Attempting to convert the whole app wholesale would not be feasible for most production applications. This document will discuss ways in which an Angular 1 app can be upgraded/refactored piecemeal.

# Basic Strategy

The basic strategy will be to bring Angular 2 features into Angular 1 which would allow an Angular 1 application to move to Angular 2 syntax and semantics as much as possible, while still executing on the Angular 1 platform.

Once the application is refactored in the new style the migration to Angular 2 should be greatly simplified.

# Detailed Design

Migration strategy will include three prong approach:

* **Core**: Add minimal set of features to the Angular v1.5 release to support migration a migration library.
* **Library**: Create a stand alone migration library, which will include Angular 2 style directives as well as directive declarations which will translate to Angular 1x DDO.
* **Tool**: Development time tool which can convert Angular 1 template to Angular 2 template with the help of a schema.

|  |  |  |
| --- | --- | --- |
|  | **JavaScript** | **HTML** |
| **Core** | Dependency Injection (?)  Attributes ( [#12198](https://github.com/angular/angular.js/issues/12198))  Compiler Rewrite Hook [#12208](https://github.com/angular/angular.js/issues/12208) | (event) Syntax [#12197](https://github.com/angular/angular.js/issues/12197)  [property] Syntax [#12198](https://github.com/angular/angular.js/issues/12198)  [(property)] Syntax (?) |
| **Library** | Angular 2 DSL to DDO translator   * selector * properties * events * host | Static conversion rules  \*ng-for  \*ng-if  \*ng-switch/case |
| **Tool** |  | Syntax translator |

## JS Syntax / API

There are four main differences which need to be tackled in an Angular 1 to 2 migration

1. Provide directive declaration syntax which matches Angular 2 on top of the Angular 1 DDO (Library)
2. Removing calls to $watch/$digest/$eval from the application code base and reference to Scope (Library)
3. Bring down Application Scope.TTL to 1. (User)
4. DI annotations. (Core)

## HTML Syntax

1. Support (event) binding natively in Angular 1 (Core [#12197](https://github.com/angular/angular.js/issues/12197))
2. Support [property] binding to element properties natively in Angular 1 (Core [#12198](https://github.com/angular/angular.js/issues/12198))
3. Allow Attributes to store current property values (Core [#12198](https://github.com/angular/angular.js/issues/12198))

## Angular 2 / 1 Nesting

1. Allow Angular 2 application to nest Angular 1 application

## Process

1. Convert all code to component as syntax
2. Upgrade to Angular v1.5
3. Include the ng2migration.js in the Angular 1 application and install its module
4. Replace all template directives with their angular 2 equivalent ( ng-repeat => \*ng-for)
5. Convert Angular 1 component directives into Angular 2 syntax (with explicit comprehension flags)
6. Convert Angular 1 decorator directives into Angular 2 syntax (with explicit comprehension flags)
7. Run a template migration tool on the Component templates (with the help of a schema) to convert Angular 1 template syntax to Angular 2 template syntax and drop the comprehension flags.
8. Write a new Top level shell component for your application in Angular 2
9. Bootstrap the Angular 1 application inside the Angular 2 shell component
10. Move Services from the Angular 1 module to the Angular 2 Shell component
11. Migrate the Angular 2 / Angular 1 boundary lower (towards the leafs).

# Example

## General

|  |
| --- |
| // Write in AngularJS 1 using Angular 2 syntax  var Greeter = module.  Component({  selector: 'greeter',  properties: ['name']  }).  View({  templateUrl: 'greeter.html'  }).  Class({  constructor: function () {  this.ctl = this; // This is needed to run in both 1 & 2  this.name = 'World';  },  greet: function() {  alert('Hello ' + this.user + '!');  }  }) |
| // This will be generated.  module.directive('greeter', function() {  return {  restrict: 'E',  scope: {},  bindToController: {  'name': '='  },  templateUrl: 'greeter.html',  controller: function() {  this.user = 'World';  this.greet = function () {  alert('Hello ' + this.user + '!');  }  },  require: 'greeter',  link: function(scope, element, attrs, controller) {  }  };  }); |
| // template  <div (click)="ctl.greet()">  Hello {{ctl.name}}!  </div> |
| // this is how it can be used  <greeter name="World"></greeter>  <greeter [name]="expr"></greeter> |

# 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 you’ll be fast.

# Work Breakdown

Description of development phases and approximate time estimates.

<div ng-app="">

<ng2-component (click)="" [prop]=""></ng2-component>

<settings [user]="expr" (done)="expr"></settings>

<settings user="expr" done="expr"></settings>

</div>

@Component({

selector: 'settings',

properties: ['user'],

events: ['done']

})

@View({

templateUrl: '...'

})

class Settings {

constructor(@Inject('$http') http) {

}

}

module.directive('Settings', ng2adapter(Settings));

module.directive('Settings', function(ng2Facade) {

var protoView = ng2Facade(Settings);

return {

restrict: 'E',

terminate: true,

link: function(scope, element, attributes) {

var settingsProtoView = …;

var cd;

protoView.instantiate(element);

scope.$watch(function() {

cd.detectChanges();

});

attrs.$observe('user', (v) => component.user = v)

component.done.observe((v) => element.fireEvent(new CustomEvent(e)));

scope.$watch(attrs.user, (v) => component.user = v);

component.done.observe((v) => scope.$eval(attrs.done, {$event: e}));

}

}

});

----