Source: <https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md>

# Single responsibility :

Define 1 component per file & < 400 lines of code:

* Easier to read & maintain
* Easier unit testing & mocking
* Avoids hidden bugs when combining multiple components

Vb:

Flappernews combineert Factory + Controller

Verbetering:

3 aparte files:

* App.module.js  
  angular.module(‘flapperNews’, [‘ng-router’]
* someController.js  
  angular.module(‘flapperNews’).controller(‘someController’, someController)  
  met de function someController() {} erin
* someFactory.js  
  angular.module(‘flapperNews’).factory (‘someFactory’, someFactory)  
  met de function someFactory() {} erin

Define small function, no more than 75 Lines (less is a lot better)

* Easier to test, especially when they serve one purpose
* Promotes reuse
* Easier to read
* Easier to maintain
* Help

# IIFE

Javascript scopes

Wrap angular components in an immediately invoked function Expression (IIFE)

* These remove variables from the global scope. Helps prevent longer living functions & variables 🡪 no variable collision
* IIFE prevent test code from reaching private members like regular expressions or helper functions

Bad example:  
// logger.js  
angular.module(‘app’).factory(‘logger’, logger);

// logger global declaration  
function logger() {}

//storage.js  
angular.module(‘app’).factory(‘storage’, storage);

Function storage() {}

Good example:  
//logger.js  
(function() **{** ‘use strict’;   
angular.module(‘app’).factory(‘logger’, logger);  
function logger () {}  
**}**)();

//storage.js  
(function() **{** ‘use strict’;   
angular.module(‘app’).factory(‘storage’, storage);  
function storage () {}  
**}**)();

# Modules

Avoid naming collisions!

Use unique naming conventions with separators for sub-modules:

* Unique names help avoid module name collisions  
  Example app may be the root while app.dashboard & app.users may be modules that are used as dependencies of app

Definitions (aka setters)!

Declare modules without a variable using the setter syntax

* Not needed with 1 component per file

Bad:

Angular.module(‘app’, [‘ngAnimate’, ‘ngRoute’, ‘app.shared’, ‘app.dash’]);

Good:

Anguldar.module(‘app’).controller(‘SomeController’, SomeController);  
function SomeController() {}

Setting vs Getting

Only set once & get for all other instances

* Modules should only be created once, then retrieved from that point on

Set: angular.module(‘app’, []);  
Get: angular.module(‘app’);

Named vs anonymous functions

Use named functions instead of passing an anonymous function in as a callback.

* Produces more readable code, easier debugging & reduces nested callback code

Bad:

Angular.module(‘app’).controller(‘dashboardController’, function() {}).factory(‘logger’,function() {});

Good:

//Dash.js  
Angular.module(‘app’).controller(‘dashboardController’, dashboardController);  
function dashboardController() {}

//logger.js  
Angular.module(‘app’).factory(‘logger’,logger);  
function logger() {}