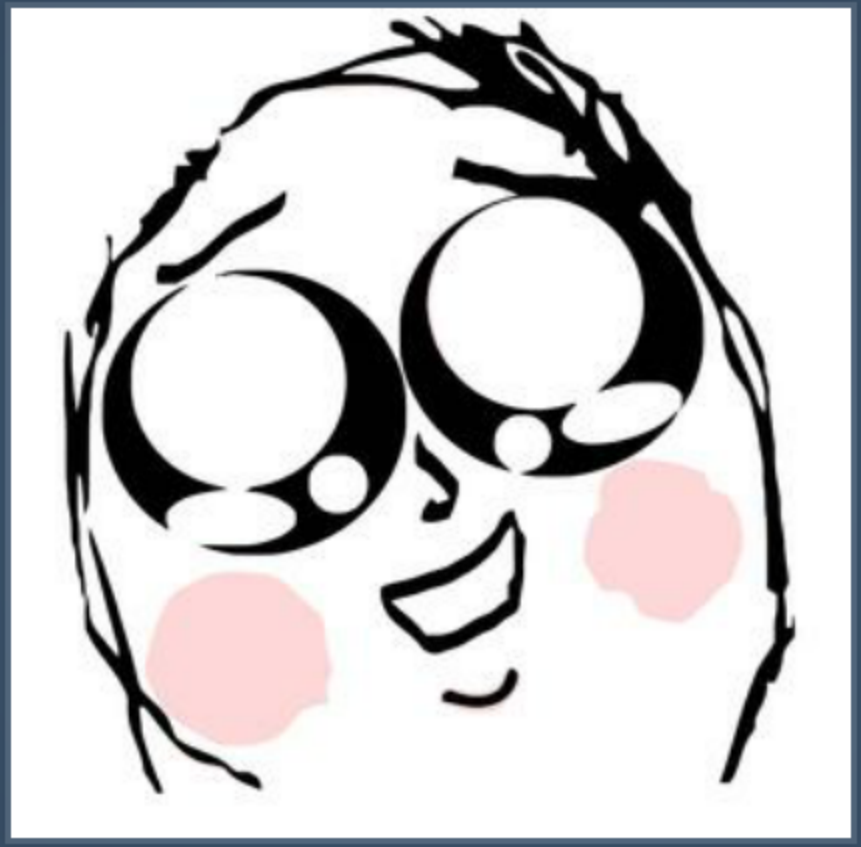
Building LARGE apps with Angular.js

@DobryJason

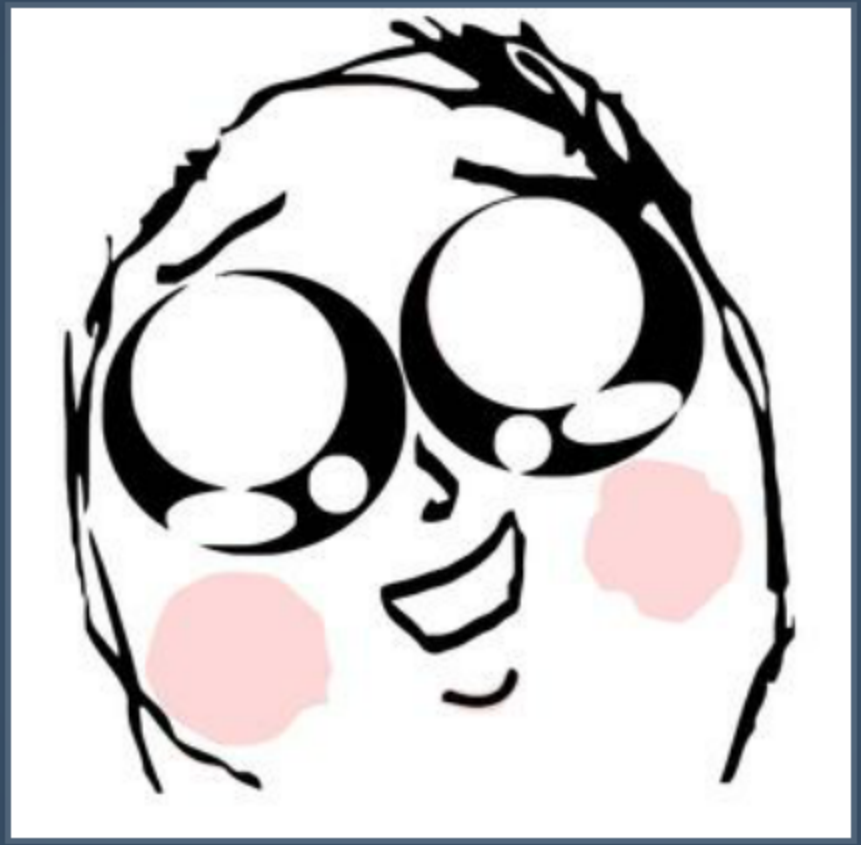
On GitHub: jmdobry

www.**pseudo**bry.com

New to AngularJS?

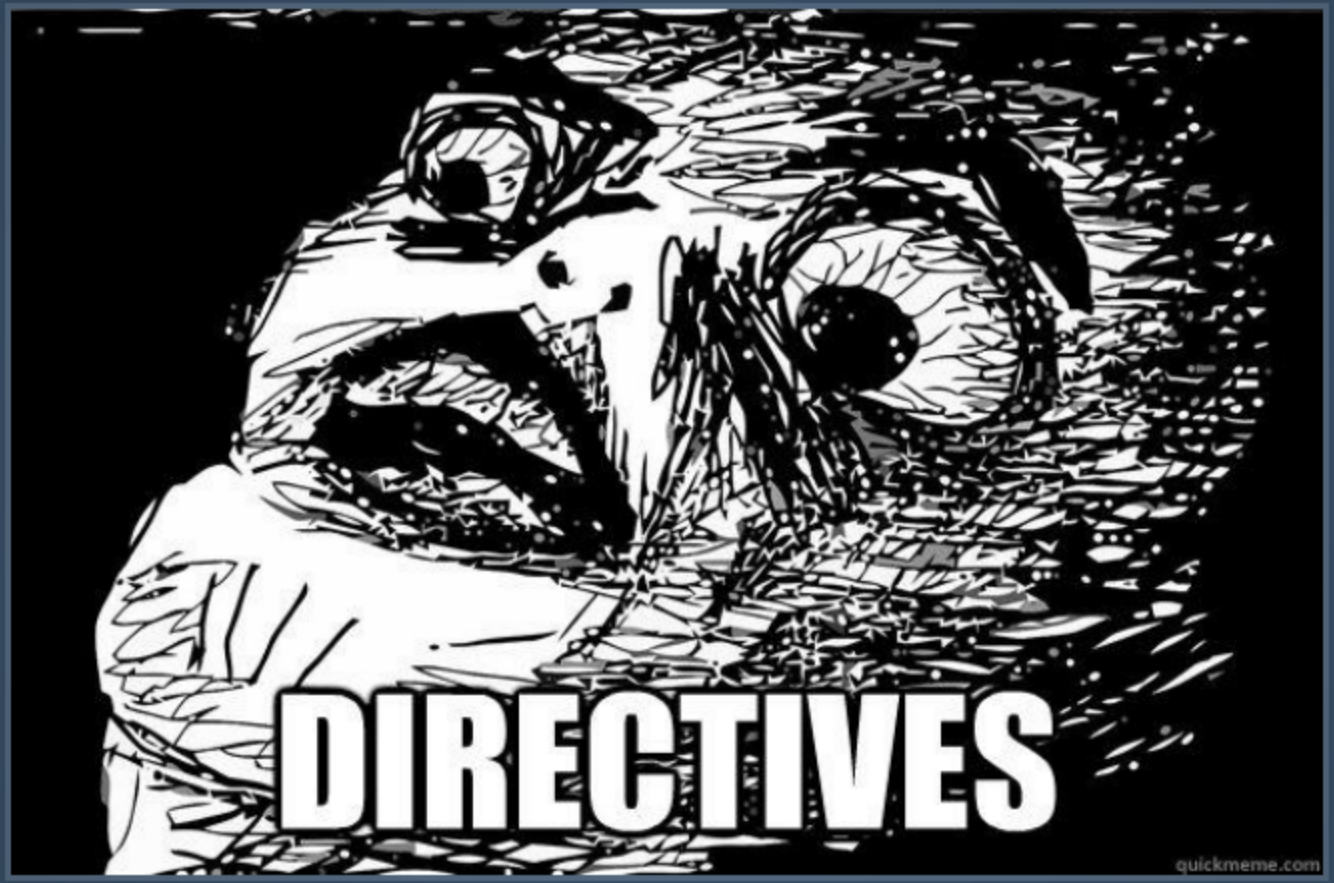


New to AngularJS?

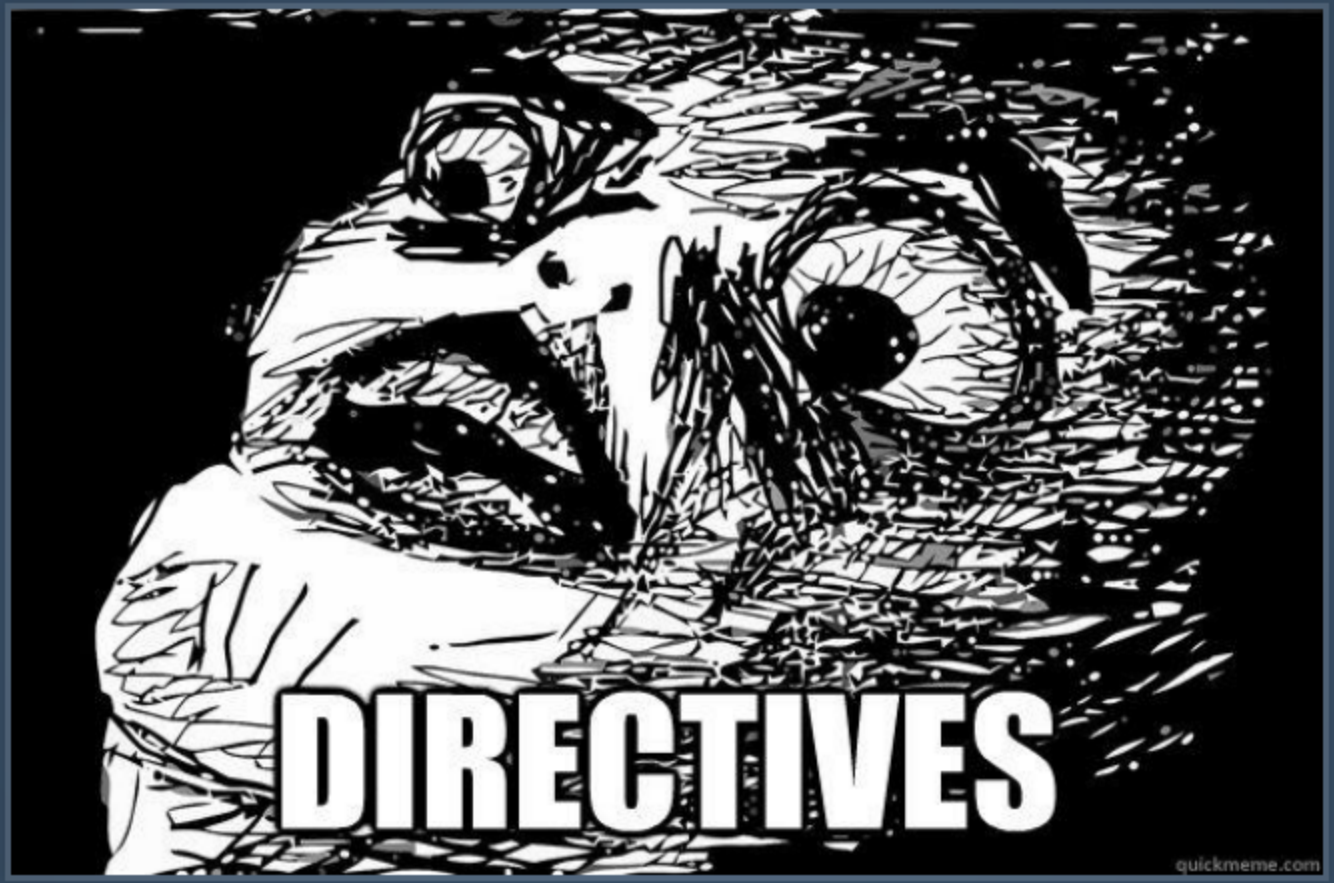


IT HAPPENS

Confused by?

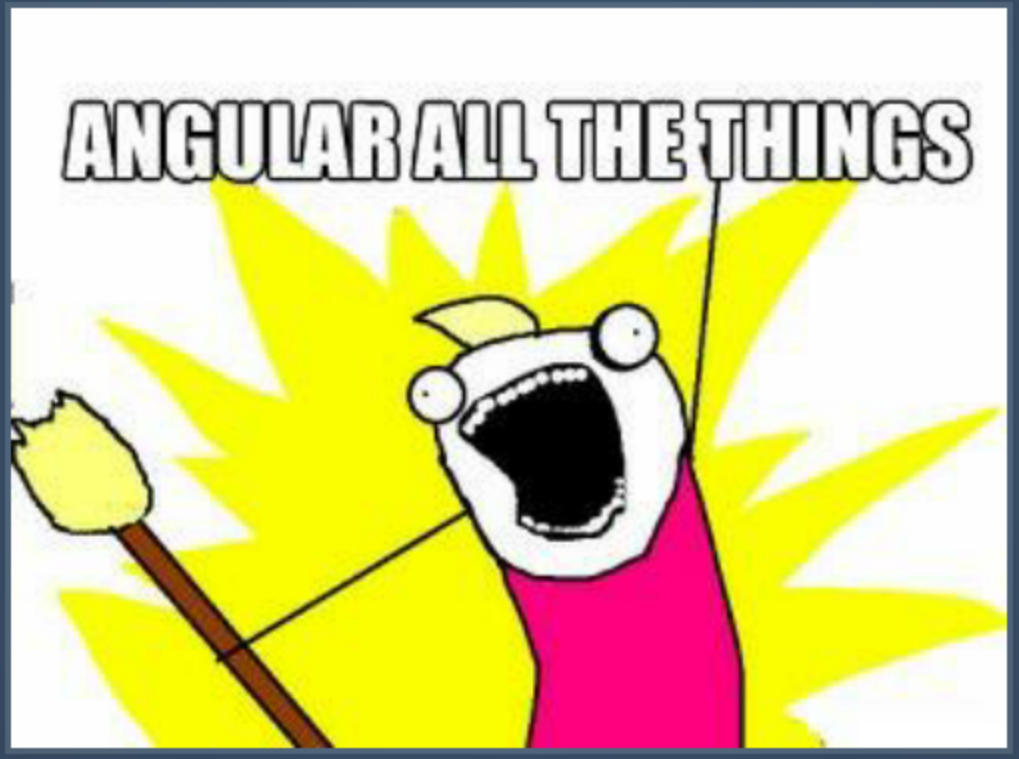


Confused by?

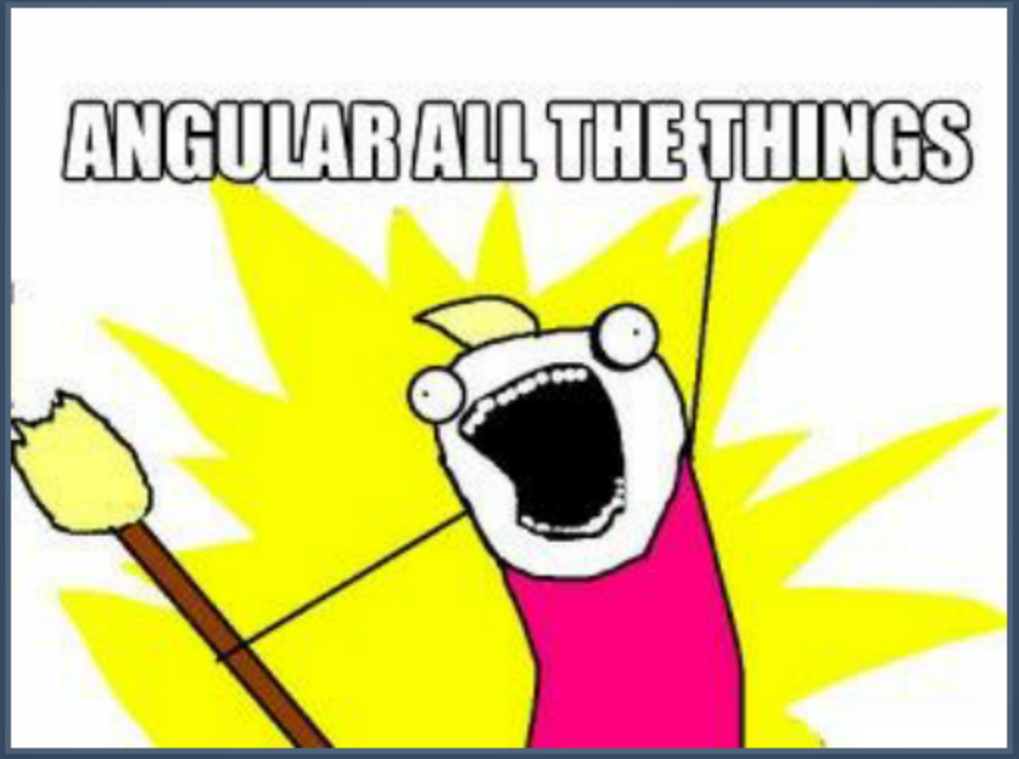


IT HAPPENS

Loving the magic and your new productivity?



Loving the magic and your new productivity?



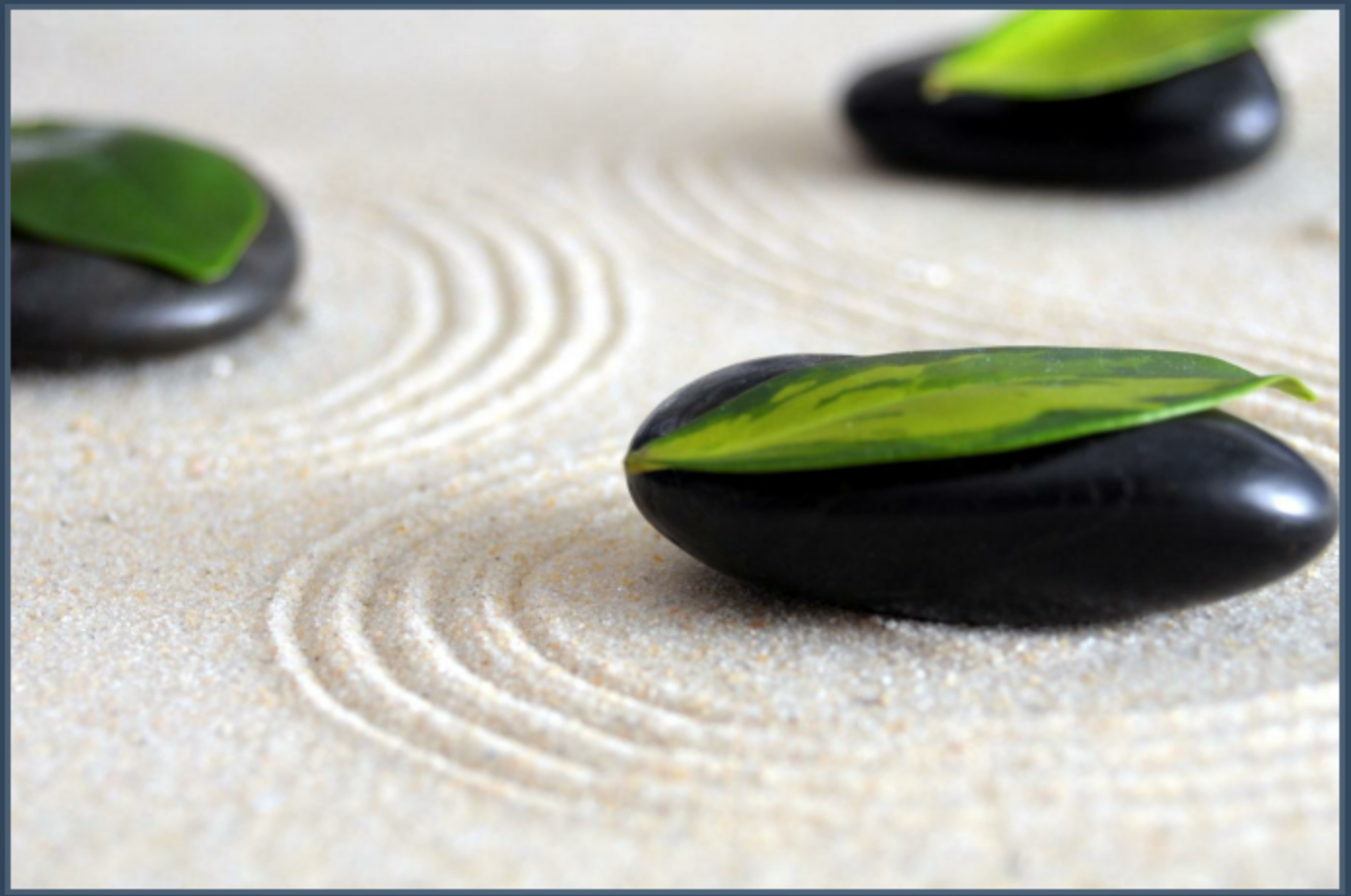
IT HAPPENS

Jaded by gotchas?

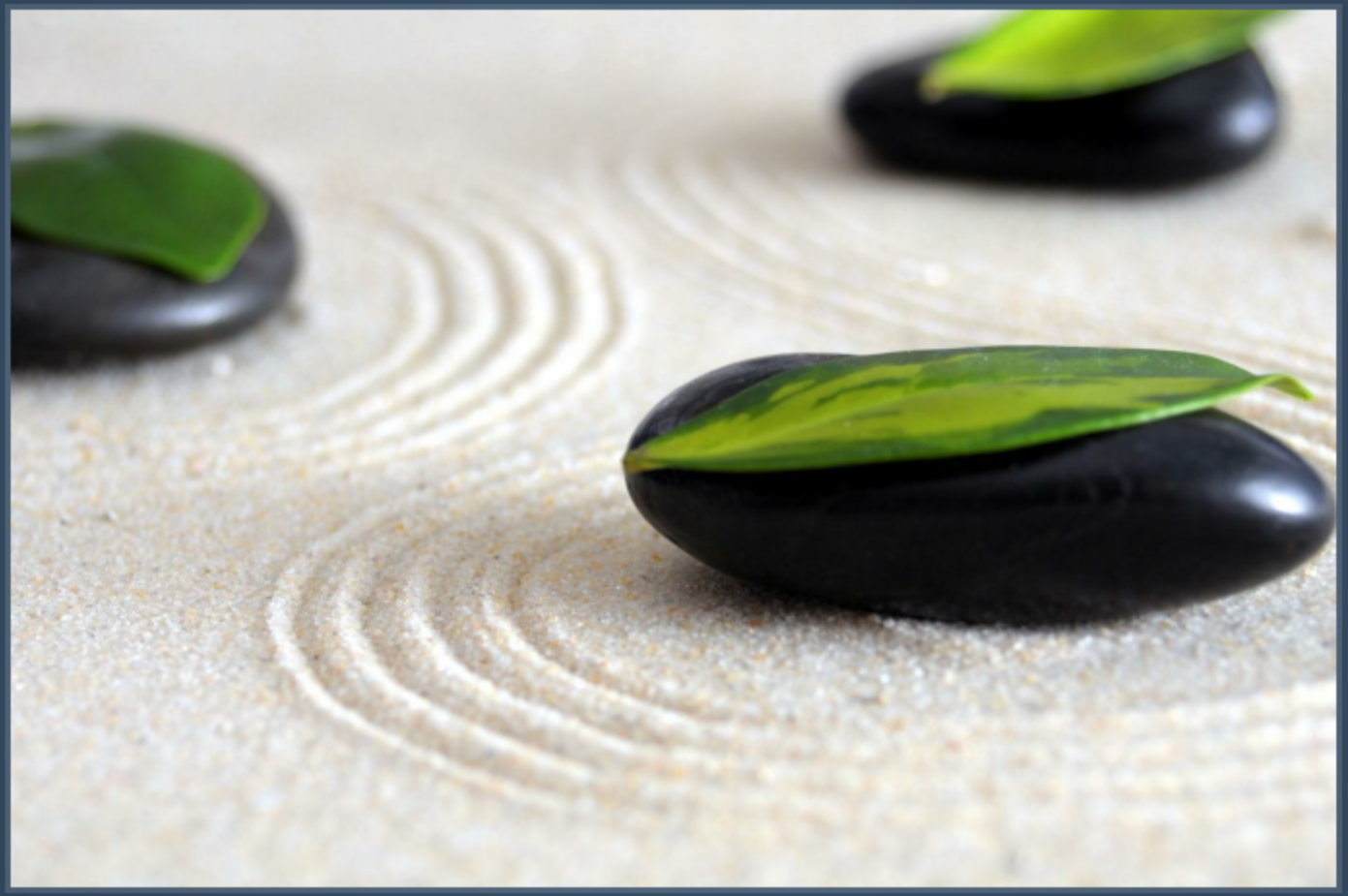
Jaded by gotchas?

IT HAPPENS

Enjoying the zen of Angular?

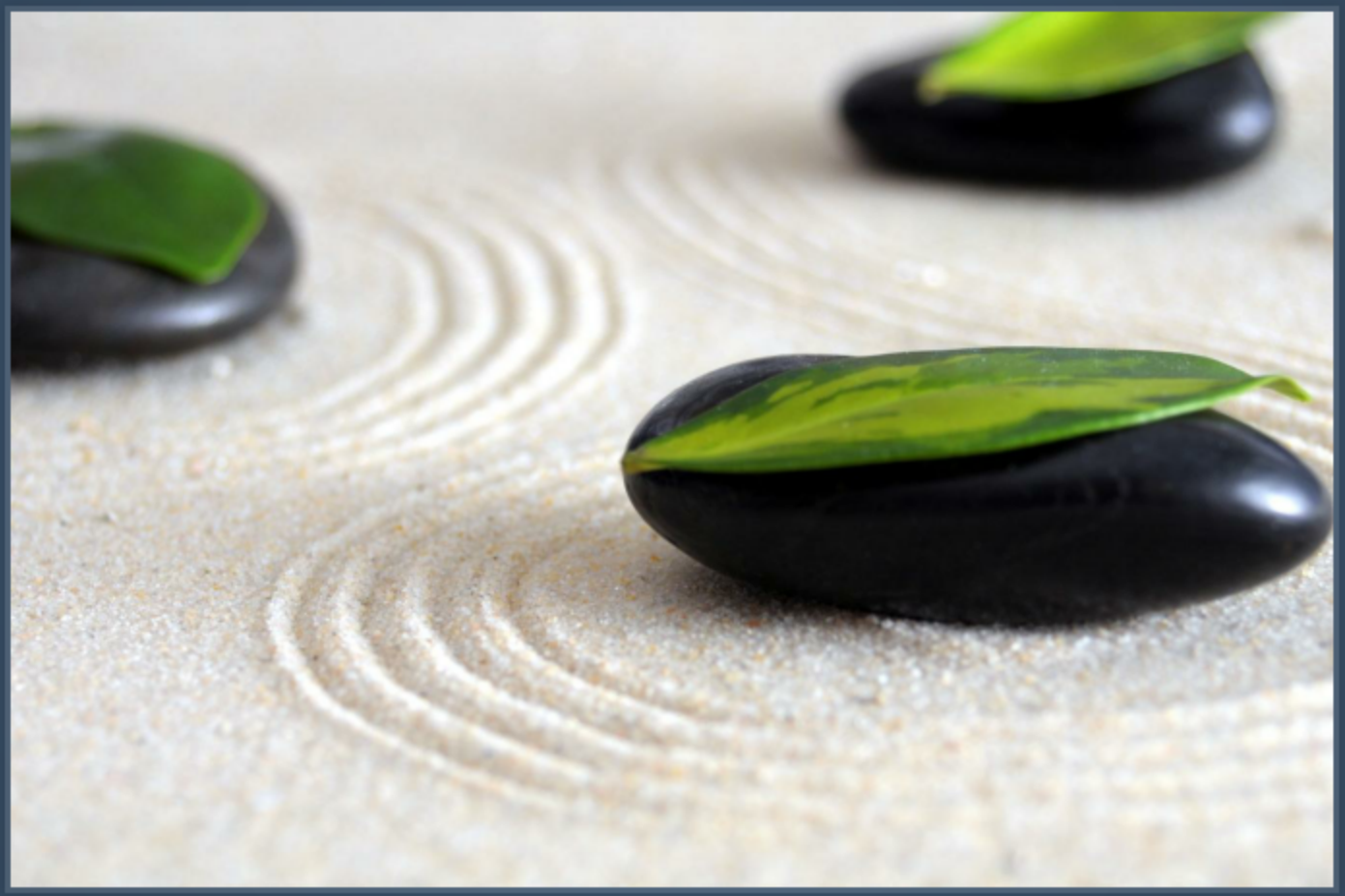


Enjoying the zen of Angular?



IT HAPPENS

Enjoying the zen of Angular?



IT HAPPENS…usually

A word of caution

You're bound to encounter Angular in the wild being used on a large project

A word of caution

You're bound to encounter Angular in the wild being used on a large project

You will inevitably run into a problem Angular does not solve for you

A word of caution

You're bound to encounter Angular in the wild being used on a large project

You will inevitably run into a problem Angular does not solve for you

Will you survive?

If you've needed to...

If you've needed to...

Organize your files

If you've needed to...

Organize your files
Name your files

If you've needed to...

Organize your files
Name your files
Lazy-load your code

If you've needed to...

Organize your files
Name your files
Lazy-load your code
Serialize application state to the URL

If you've needed to...

Organize your files
Name your files
Lazy-load your code
Serialize application state to the URL
Derive application state from the URL

If you've needed to...

Organize your files
Name your files
Lazy-load your code
Serialize application state to the URL
Derive application state from the URL
Manage a Model Layer

If you've needed to...

Organize your files
Name your files
Lazy-load your code
Serialize application state to the URL
Derive application state from the URL
Manage a Model Layer
Do smart caching

If you've needed to...

Organize your files
Name your files
Lazy-load your code
Serialize application state to the URL
Derive application state from the URL
Manage a Model Layer
Do smart caching
Work with a RESTful interface (for real)

If you've needed to...

Organize your files
Name your files
Lazy-load your code
Serialize application state to the URL
Derive application state from the URL
Manage a Model Layer
Do smart caching
Work with a RESTful interface (for real)
Use a 3rd-party lib with Angular

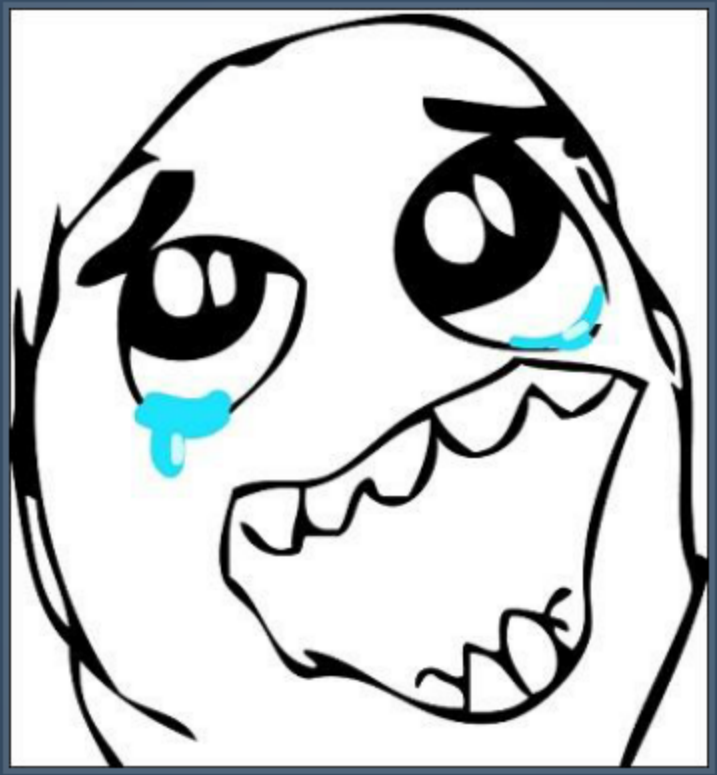
If you've needed to...

Organize your files
Name your files
Lazy-load your code
Serialize application state to the URL
Derive application state from the URL
Manage a Model Layer
Do smart caching
Work with a RESTful interface (for real)
Use a 3rd-party lib with Angular

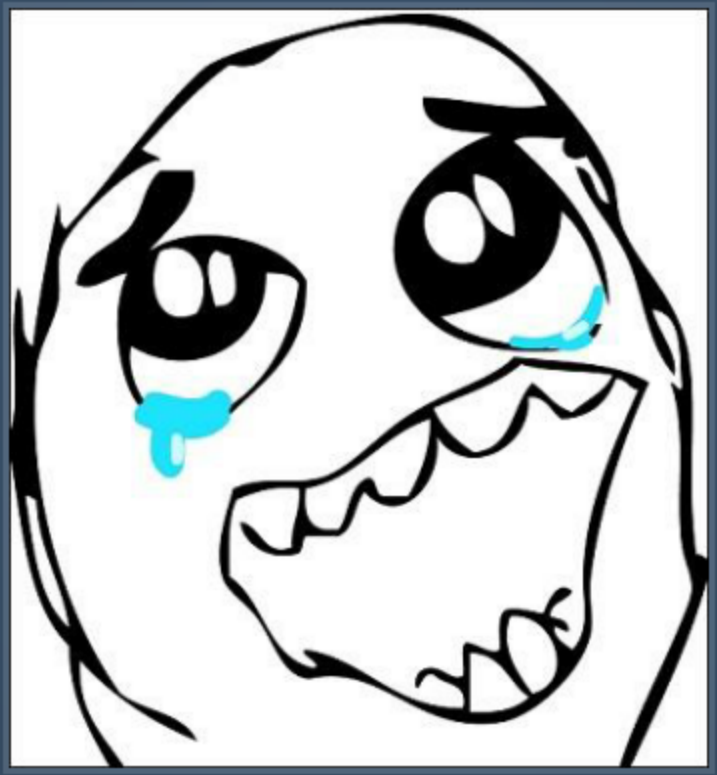
THEN YOU'VE ALREADY MET THE BEAST

But there will come a day when it saves your life

But there will come a day when it saves your life



But there will come a day when it saves your life

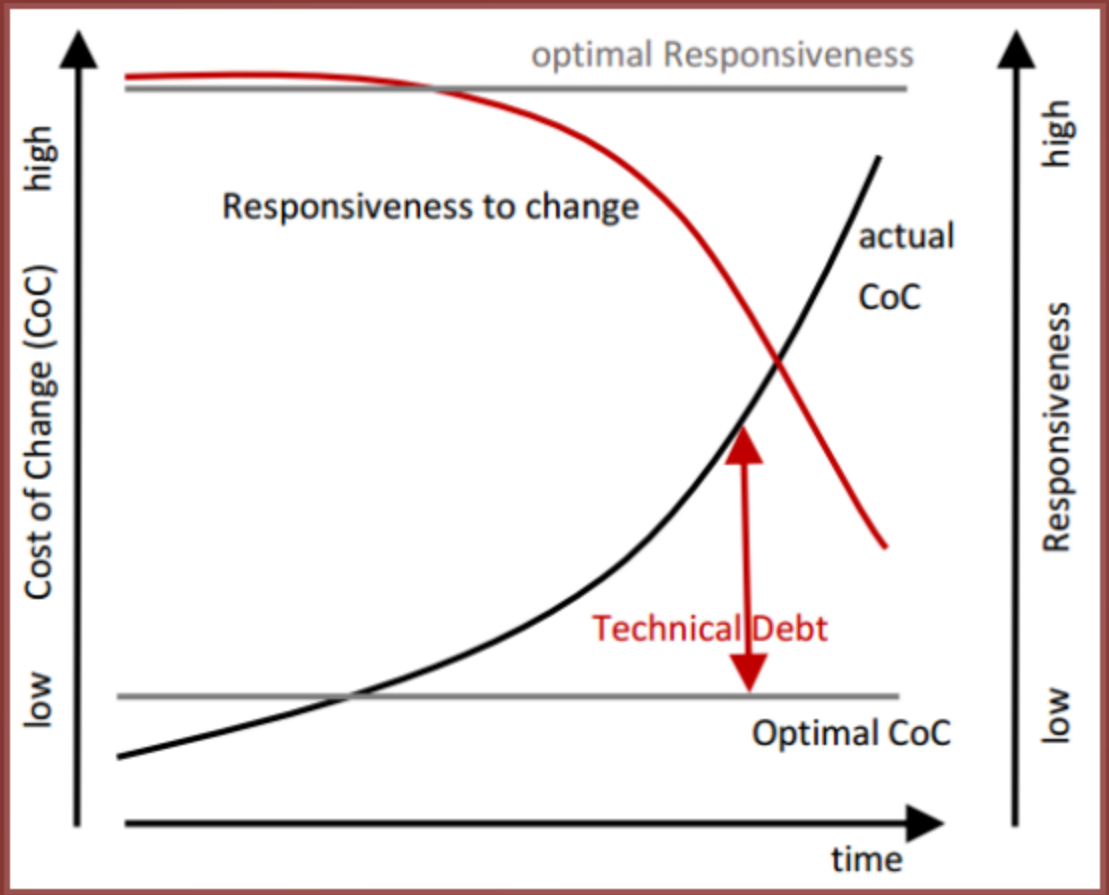


and you will cry

Survival Kit:

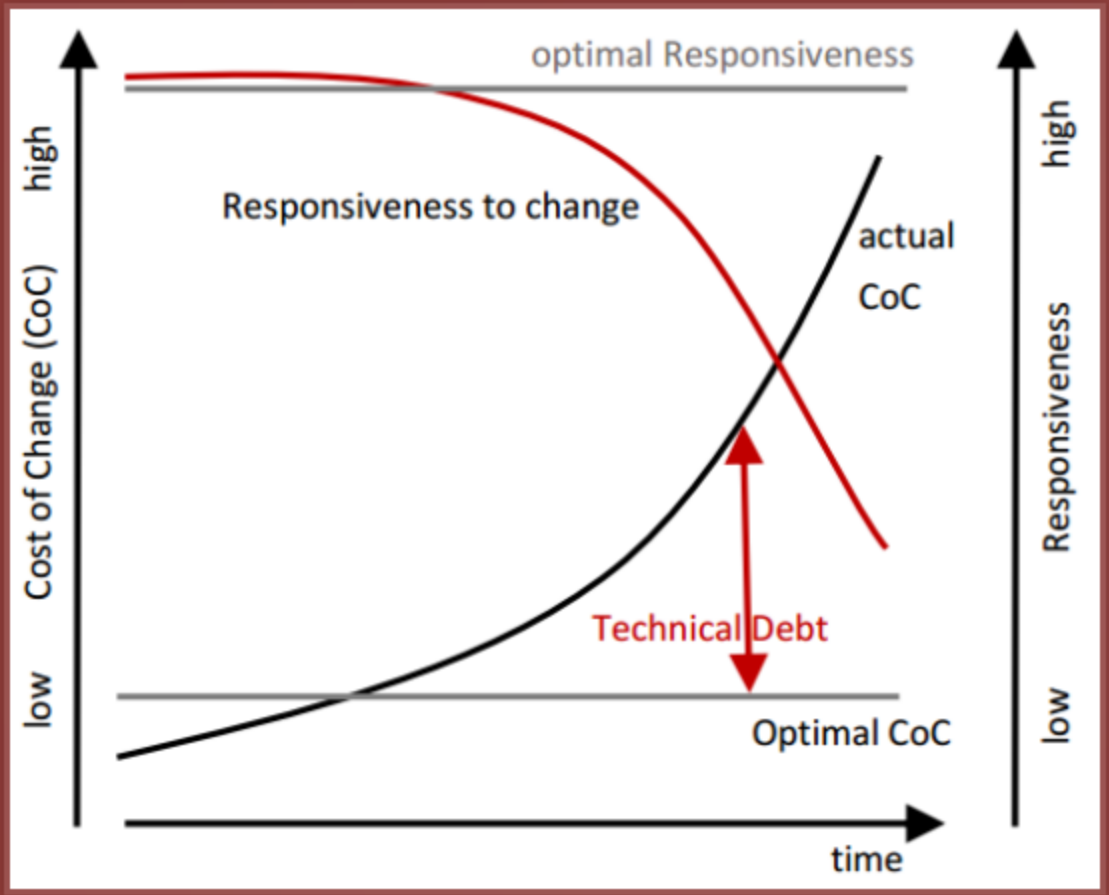
Building LARGE apps with Angular.js

Convention vs Configuration



Source: Clean Code Cheat Sheet

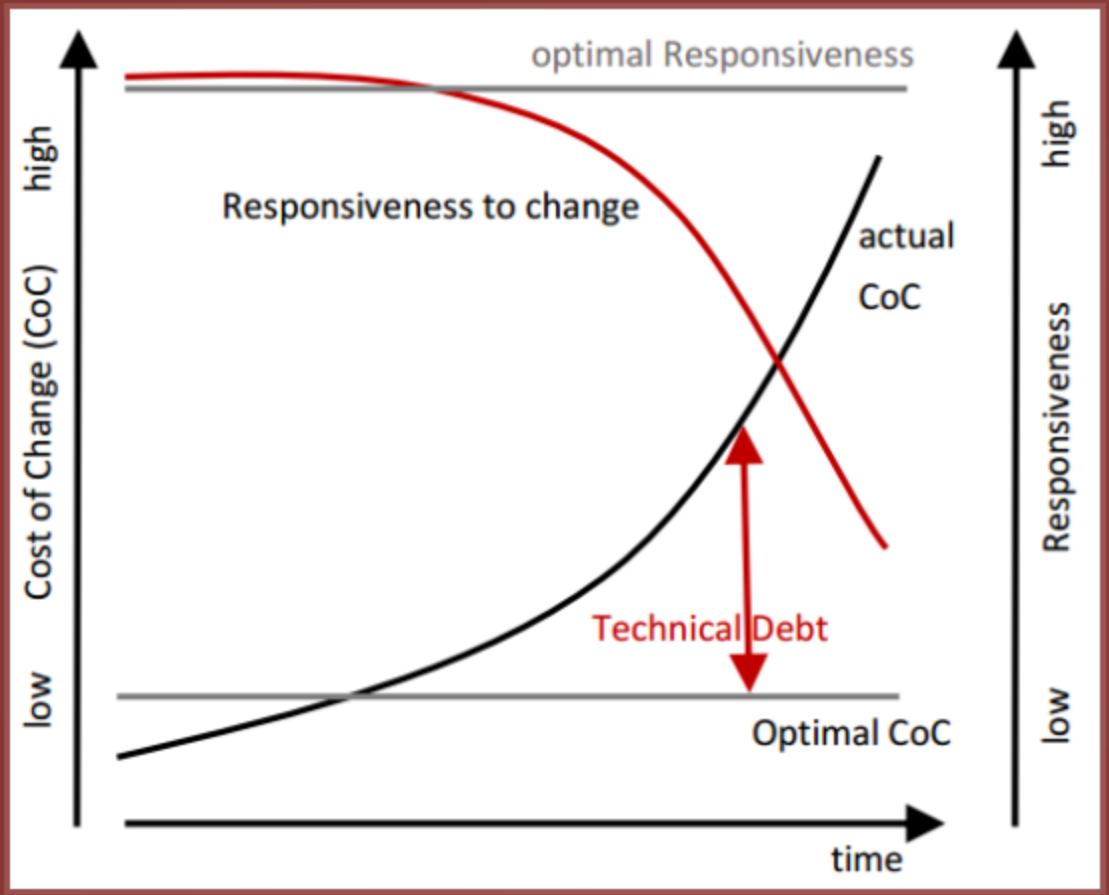
Convention vs Configuration



Testability

Source: Clean Code Cheat Sheet

Convention vs Configuration

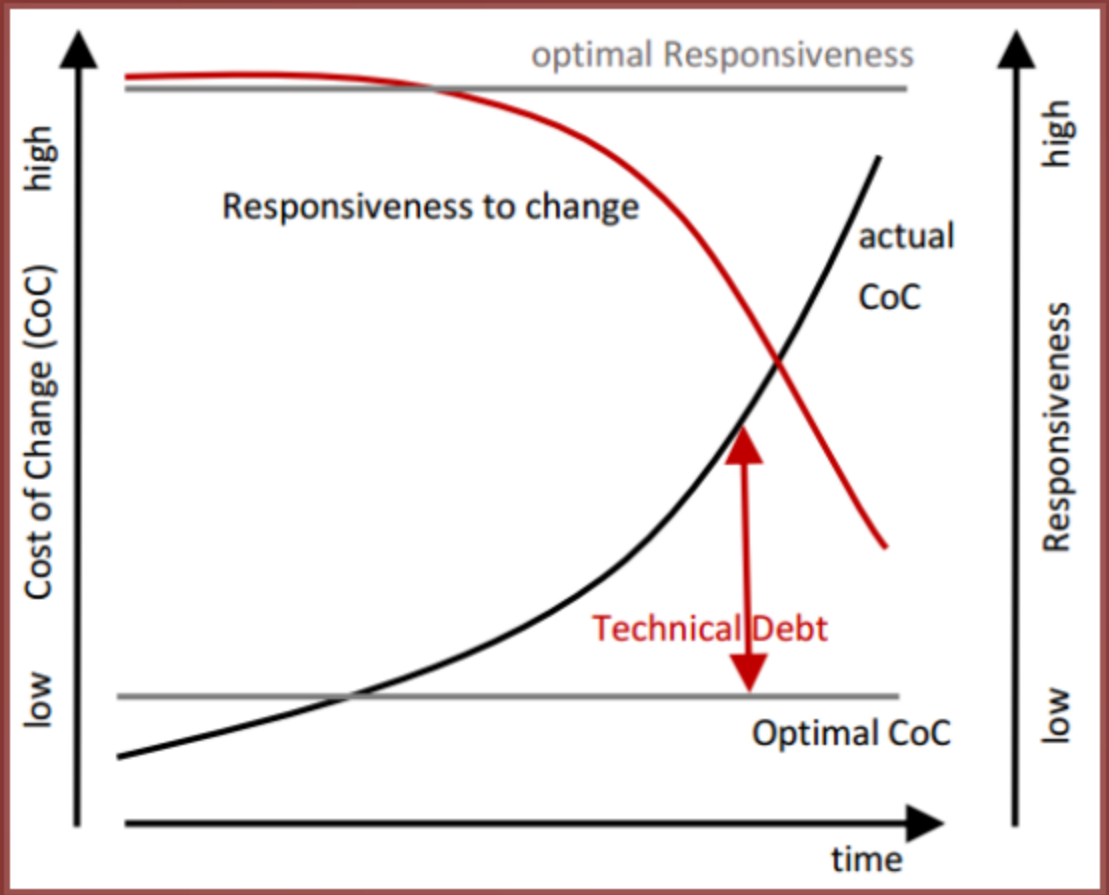


Testability

Changeability

Source: Clean Code Cheat Sheet

Convention vs Configuration



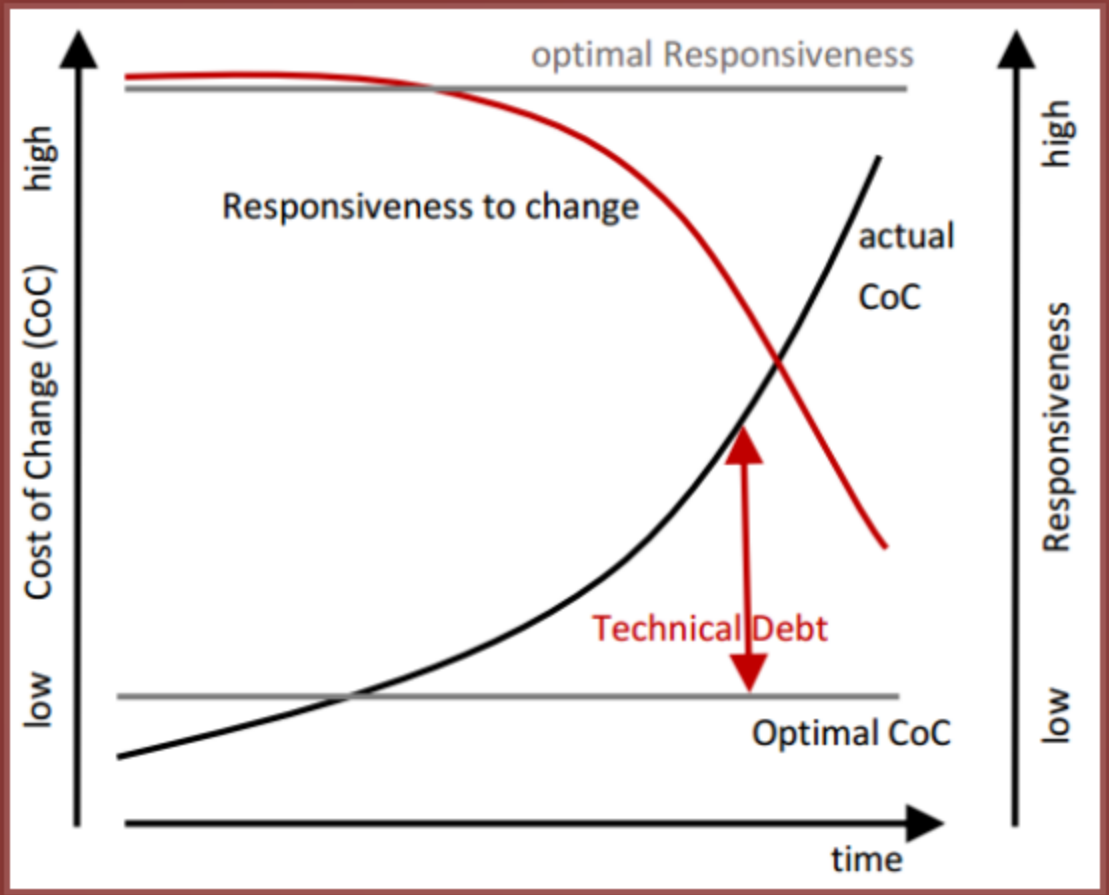
Testability

Changeability

Maintainability

Source: Clean Code Cheat Sheet

Convention vs Configuration



Testability

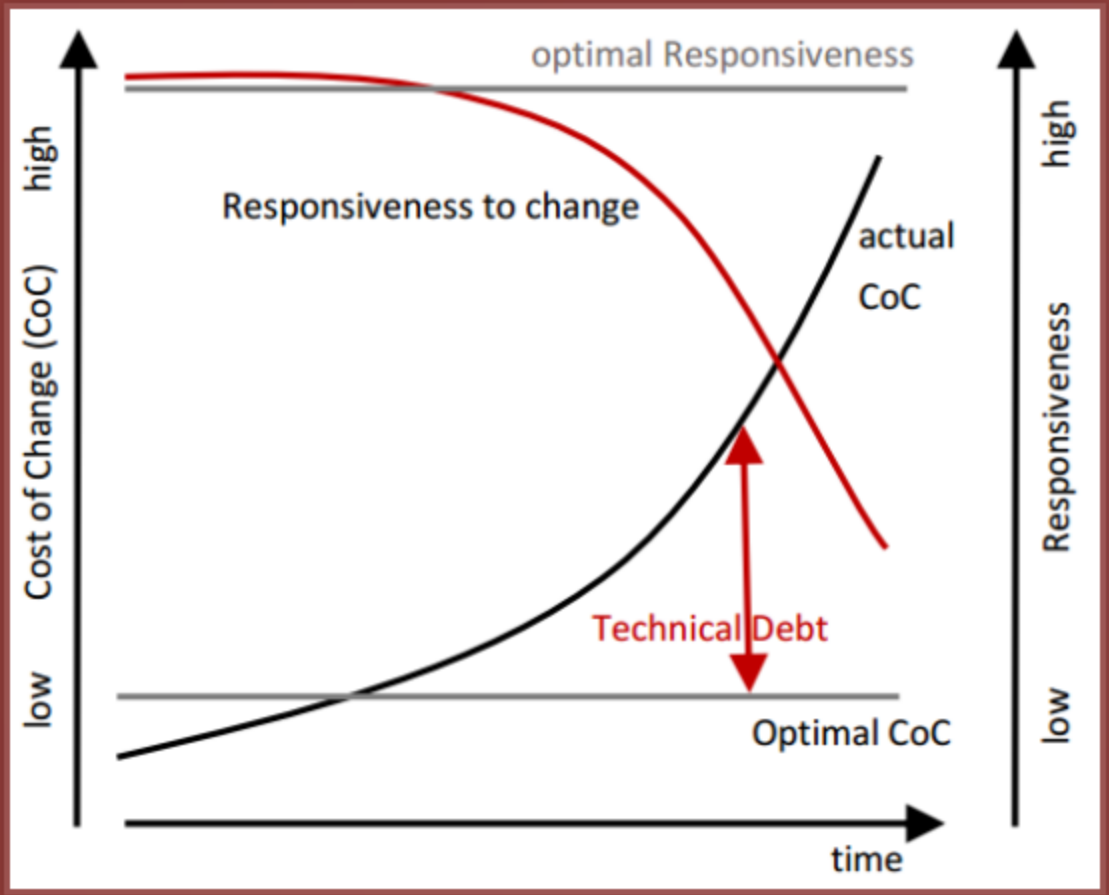
Changeability

Maintainability

Extensibility

Source: Clean Code Cheat Sheet

Convention vs Configuration



Testability

Changeability

Maintainability

Extensibility

Readability

Source: Clean Code Cheat Sheet

Convention vs Configuration

$service

$logProvider

$http

$log

$scope

$routeProvider

$controller

?

$location

$rootScope

$factory

$httpProvider

$q

$locationProvider

Convention vs Configuration

$service

$logProvider

$http

$log

$scope

$routeProvider

$controller

?

$location

$rootScope

$factory

$httpProvider

$q

$locationProvider

What is Angular's stance on anything?

Convention Example

```
app.controller('MyCtrl', function ($scope, MyService) {
    var _count = 0;

    function _privateUtil() { // function decomposition
        _count += _count + 1;
    }

    function _handleClick(data) {
        _privateUtil();
        $scope.answer = MyService.doSomething({
            data: data,
            count: _count
        });
    }

    function _init() {
        // expose "data" to the view
        $scope.data = 10;

        // expose _handleClick as a "public" method on the $scope
        $scope.handleClick = _handleClick;
    }

    _init();
}
```

Convention vs Configuration

Building LARGE apps with Angular

Keys to success:

Convention vs Configuration

Building LARGE apps with Angular
Keys to success:

- Establish/follow conventions

Convention vs Configuration

Building LARGE apps with Angular
Keys to success:

- Establish/follow conventions
- AngularJS docs = no help there

Convention vs Configuration

Building LARGE apps with Angular
Keys to success:

- Establish/follow conventions
- AngularJS docs = no help there
- Read what experienced Angular devs write.

Convention vs Configuration

Building LARGE apps with Angular
Keys to success:

- Establish/follow conventions
- AngularJS docs = no help there
- Read what experienced Angular devs write.
- Pay attention to the mailing list , the GitHub project , and the opinions of Angular's core contributors

Convention vs Configuration

Building LARGE apps with Angular
Keys to success:

- Establish/follow conventions
- AngularJS docs = no help there
- Read what experienced Angular devs write.
- Pay attention to the mailing list , the GitHub project , and the opinions of Angular's core contributors

don't abandon good programming design

Model Layer

"Angular makes no restrictions or requirements on the model"

Model Layer

"Angular makes no restrictions or requirements on the model"

Controllers

Model Layer

"Angular makes no restrictions or requirements on the model"

Controllers
$scope

Model Layer

"Angular makes no restrictions or requirements on the model"

Controllers
$scope

<--------->

Views

Model Layer

"Angular makes no restrictions or requirements on the model"

Model Layer <-------- > Controllers $scope <--------> Views

Model Layer

"Angular makes no restrictions or requirements on the model"

Model Layer <-------- Controllers $scope <--------> Views
>

$scope !== Model Layer

Model Layer

"Angular makes no restrictions or requirements on the model"

Model Layer <-------- Controllers $scope <--------> Views
>

$scope !== Model Layer

$scope === glue between Controller and View

Model Layer

```
app.service('MyService', function ($http, $q, $angularCacheFactory) {
    var _dataCache = $angularCacheFactory('dataCache', {
        maxAge: 3600000 // items expire after an hour
    });
    /**
     * @class MyService
     */
    return {
        manipulateData: function (input) {
            var output;
            // do something with the data
            return output;
        },

        getDataById: function (id) {
            var deferred = $q.defer();
            if (_dataCache.get(id)) {
                deferred.resolve(_dataCache.get(id));
            } else {
                // Get the data from the server and populate cache
            }
            return deferred.promise;
        }
    };
});
```

Model Layer

Controllers consume your data

Model Layer

Controllers consume your data
and massage it into your $scope

```
$scope.$watch(function () {
    return MyService.getDataById($scope.selectedId);
}, function () {
    $scope.currentData = MyService.getDataById($scope.currentData.id);
});
```

Model Layer

Controllers consume your data
and massage it into your $scope

```
$scope.$watch(function () {
    return MyService.getDataById($scope.selectedId);
}, function () {
    $scope.currentData = MyService.getDataById($scope.currentData.id);
});
```

Two-way data binding takes care of the rest

```
<h3p>{{ currentData.name }}</h3>
```

Model Layer

Controllers consume your data
and massage it into your $scope

```
$scope.$watch(function () {
    return MyService.getDataById($scope.selectedId);
}, function () {
    $scope.currentData = MyService.getDataById($scope.currentData.id);
});
```

Two-way data binding takes care of the rest

```
<h3p>{{ currentData.name }}</h3>
```

Your Services are the "source of truth" for your data

Model Layer

Large apps need a real Model Layer

$ngResource

Restangular

BreezeJS

Treat your data with respect

File Organization

Organize by type, form, function, route, or feature?

File Organization

Organize by type, form, function, route, or feature?

Sometimes "just search for it" isn't enough

File Organization

Organize by type, form, function, route, or feature?

Sometimes "just search for it" isn't enough

Logical organization is important

File Organization

Monolithic Files

```
partials/
    home.html
    login.html
    users.html
    orders.html
js/
    controllers.js
    directives.js
    filters.js
    services.js
    app.js
```

Recommended by angular-seed project

File Organization

Monolithic Folders

```
js/
    controllers/
        homeController.js
        loginController.js
    directives/
        usersDirective.js
        ordersDirective.js
    filters/
    services/
        userService.js
        orderService.js
        loginService.js
partials/
    home.html
    login.html
    users.html
    orders.html
app.js
```

An improvement - One "thing" per file

File Organization

Organize by feature

```
orders/
    directives/
        orders.html
        ordersDirective.js
    services/
        orderService.js
users/
    directives/
        users.html
        usersDirective.js
    services/
        userService.js
home/
    controllers/
        home.html
        homeController.js
        login.html
        loginController.js
shared/
    services/
        i18nService.js
    filters/
        i18nFilter.js
app.js
```

Modules

Modules

GOOD FOR NOTHING...so far

Modules

GOOD FOR NOTHING...so far

except loading 3rd-party angular plugins

Modules

GOOD FOR NOTHING...so far

except loading 3rd-party angular plugins
and mocks during testing

Modules

GOOD FOR NOTHING...so far

except loading 3rd-party angular plugins and mocks during testing

Misko Hevery on multiple modules:

"[you] should group by view since views will be lazy loaded in near future"

Modules

GOOD FOR NOTHING...so far

except loading 3rd-party angular plugins and mocks during testing

Misko Hevery on multiple modules:

"[you] should group by view since views will be lazy loaded in near future"

More info on that someone please?

Lazy Loading Now

Not *officially* supported, but possible

```
var app = angular.module('app', []);

app.config(function ($controllerProvider, $compileProvider, $filterProv

    // save references to the providers
    app.lazy = {
        controller: $controllerProvider.register,
        directive: $compileProvider.directive,
        filter: $filterProvider.register,
        factory: $provide.factory,
        service: $provide.service,
        animation: $animationProvider.register
    };

    // define routes, etc.
});
```

Lazy Loading Now

Load dependencies on a per-route basis

```
$routeProvider.when('/items', {
    templateUrl: 'partials/items.html',
    resolve: {
        load: ['$q', '$rootScope', function ($q, $rootScope) {
            var deferred = $q.defer();
            // At this point, use whatever mechanism you want
            // in order to lazy load dependencies. e.g. require.js
            // In this case, "itemsController" won't be loaded
            // until the user hits the '/items' route
            require(['itemsController'], function () {
                $rootScope.$apply(function () {
                    deferred.resolve();
                });
            });
            return deferred.promise;
        }]
    }
});
```

Lazy Loading Now

Lazily-loaded components use the saved providers

itemsController.js

```
// reference the saved controller provider
app.lazy.controller('ItemsController', function ($scope) {
    // define ItemsController like normal
});
```

items.html

```
<section data-ng-controller="ItemsController">
    <!-- a bunch of awesome html here -->
</section>
```

Kudos to Ifeanyi

Understand the $scope life-cycle

Understand the $scope life-cycle

$watch expressions should be fast

Understand the $scope life-cycle

$watch expressions should be fast and idempotent

Understand the $scope life-cycle

$watch expressions should be fast and idempotent

Too many/cpu intensive $watch expressions results in

Understand the $scope life-cycle

$watch expressions should be fast and idempotent

Too many/cpu intensive $watch expressions results in

in $digest ion

Understand the $scope life-cycle

$watch expressions should be fast and idempotent

Too many/cpu intensive $watch expressions results in

in $digest ion

(poor performance)

Understand the $scope life-cycle

Understand the $scope life-cycle

$scope is prototypal

Understand the $scope life-cycle

$scope is prototypal ...use with caution

Understand the $scope life-cycle

$scope is prototypal ...use with caution

```
$scope.fruit = 'banana';
var $newScope = $scope.$new();

$scope.fruit; // 'banana'
$newScope.fruit; // 'banana'

$newScope.fruit = 'apple';
$newScope.fruit; // 'apple'

$scope.fruit; // 'banana' // Did you think it would be 'apple'?
```

3rd-party integration with Angular.js

If something happens (DOM or data change) and Angular doesn't know about it...

3rd-party integration with Angular.js

If something happens (DOM or data change) and Angular doesn't know about it...

in $digestion

3rd-party integration with Angular.js

If something happens (DOM or data change) and Angular doesn't know about it...

in $digest ion

No $digest loop

3rd-party integration with Angular.js

If something happens (DOM or data change) and Angular doesn't know about it...

in $digestion

No $digest loop
No $watch callbacks executed

3rd-party integration with Angular.js

If something happens (DOM or data change) and Angular doesn't know about it...

in $digestion

No $digest loop
No $watch callbacks executed
No automagical updates

3rd-party integration with Angular.js

Use Angular counterparts where possible, e.g. $timeout, $http, etc.

Manually call $scope.$apply (if you need to)

```
$scope.$apply(function () {
    $scope.result = thirdPartyLib.goDoSomething();
});
```

I DON'T ALWAYS BUILD LARGE APPS WITH ANGULAR
BUT WHEN I DO I LIKE IT
quickmeme.com

The end

@DobryJason

On GitHub: jmdobry

www.**pseudo**bry.com

/building-large-apps-with-angularjs.html

also /designing-for-testability.html