LESSONS LEARNED: ANGULARJS IN LARGE BUSINESS APPLICATIONS

BY THOMAS SCHEINECKER

ABOUT ME

A SHORT INTRODUCTION

Name: Thomas Scheinecker
Age: 25
Working at: Catalysts GmbH
Coding since: 1999 (age 11)
Coding Contests: CCC, CH24, Google Code Jam

MY PREVIOUS PROJECTS:

WILLHABEN.AT®

LINZ AG

PORSCHE
INFORMATIK GMBH

core
smartwork

eurofunk
KAPPACHER

WHY ANGULARJS?

COMPLETE JAVASCRIPT MVC FRAMEWORK

- 100% JavaScript
- 100% client side

WHAT IT PROVIDES

- separation of concerns (encourages mvc)
- testability (complete control about bootstrapping / injection)
- 2 way data binding (no selectors for setting / reading values)
- promises (no more registering of callbacks)

...

WRITE LESS - GET MORE

Interactive 'Hello World' without a single line of JavaScript!

```
<!DOCTYPE html>
<html ng-app>
<body>
<div>
    <label>Name:</label>
    <input type="text" ng-model="name" ng-init="name='world'">
    <h1>Hello <small>{{name}}</small>!</h1>
</div>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.16/angular.m
</body>
</html>
```

LIVE EXAMPLE:

World Hello World!

WHY IS IT BETTER THAN $*/=?!.JS?

Here are the GitHub stats from 6. May 2014:

commits / contributors are from a 1 month period

(the values in brackets are the diff to the stats from 31. January 2014)

	Angular	Ember	Backbone	Knockout
Stars	23432 (3173)	10087 (758)	17862 (813)	4921 (288)
Watches	2371 (307)	853 (41)	1473 (55)	440 (29)
Forks	7806 (1748)	2186 (207)	3934 (249)	814 (54)
Commits	397 (249)	38 (-83)	29 (21)	1 (-15)
Contributors	137 (116)	22 (1)	15 (11)	1 (-1)
Releases 2013	17	14	3	4

LET’S LOOK AT SOME CODE

ANGULAR MODULES

```
angular.module('myApp', []);
```

myApp.js

SERVICES

FACTORIES: STATIC CONFIGURATION

```
angular.module('myApp').factory('myService', function($log) {
    return {
        myServiceFunction: function() {
            $log.log('called myServiceFunction()');
            // do some fancy stuff
        },
        addSomeStuff: function(scope, name) {
            // augment external data / scope
            scope.someFunction = function(value) {
                $log.log('name: "'+name+'" value: "'+value+'"');
            };
            scope.someValue = 'someValue';
        }
    };
});
```

myService.js

SERVICES

PROVIDERS: CONFIGURABLE

```
angular.module('myApp').provider('myProviderService', function($log) {
    var logMessage = 'defaultMessage';

    return {
        setLogMessage: function(msg){
            logMessage = msg;
        },
        $get: {
            log: function() {
                $log.log(logMessage);
            }
         }
    };
});
```

myServiceProvider.js

SERVICES

PROVIDERS: CONFIGURATION

```
angular.module('myApp').config(function(myProviderServiceProvider) {
    // note the name: 'myProviderServiceProvider'
    // instead of 'myProviderService'
    myProviderServiceProvider.setLogMessage('myMessage');
});
```

app.js

SERVICES

USING THEM

```
angular.module('myApp').controller('myServiceController',
    function($scope, myService, myProviderService) {
        $scope.action = function() {
            myService.myServiceFunction(); // call method from service
        };

        // extend scope
        myService.addSomeStuff($scope, 'myController');

        // use some function (added by above call)
        $scope.someFunction($scope.someValue);
        // logs: 'name: "myController" value: "someValue"'

        // expose service on scope
        $scope.myService = myService;

        myProviderService.log();
        // logs: 'myMessage'
```

myServiceController.js

DIRECTIVES

```
angular.module('myApp')
    .directive('confirmClick', function () {
        return {
            restrict: 'A',
            link: function (scope, element, attr) {
                var msg = attr.confirmClick || 'Are you sure?';
                var clickAction = attr.onConfirm;
                element.bind('click', function (event) {
                    if (window.confirm(msg)) {
                        scope.$eval(clickAction);
                    }
                });
            }
        };
    });
```

myDirective.js

LETS GO BIG!

HOW TO

- organize your files
- *keep your code DRY*
- cache efficiently
- lazy-load your code
- use 3rd party libs

FILE ORGANIZATION

FILE ORGANIZATION

Monolithic files

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

This gets messy quite quickly!

FILE ORGANIZATION

Monolithic folders

```
partials/
    home.tpl.html
    login.tpl.html
    users.tpl.html
js/
    controllers/
        homeController.js
        loginController.js
        userController.js
    directives/
        usersDirective.js
    filters/
        i18nFilter.js
    services/
        loginService.js
        userService.js
    app.js
```

A little bit better - not everything mixed together anymore

FILE ORGANIZATION

Organized by feature

```
home/
    partials/
        home.tpl.html
    controllers/
        homeController.js
login/
    partials/
        login.tpl.html
    controllers/
        loginController.js
    services/
        loginService.js
user/
    partials/
        users.tpl.html
        userDetails.tpl.html
    controllers/
        userController.js
```

Even easier to find what you are looking for!

FILE ORGANIZATION

Organized by feature and module

```
public/
    home/
        partials/
            home.tpl.html
        controllers/
            homeController.js
        home.js
    login/
        partials/
            login.tpl.html
        controllers/
            loginController.js
        services/
            loginService.js
        login.js
    public.js
private/
    user/
```

Now with module separation (more on that later)

DON'T REPEAT YOURSELF!

why should you?!

INHERITANCE

```
window.BaseDetailController = function($scope, $routeParams, $location) {
    $scope.edit = function() {
        // switch to edit mode
    }

    $scope.save = function() {
        // save
        // show validation / view mode
    }
};
```

```
angular.module('myApp').controller('myInheritingController',
    function($scope, $injector) {
        $injector.invoke(window.BaseDetailController, this,
        // we pass all object which shouldn't be resolved by angular
            {$scope: $scope});

        // $scope.edit and $scope.save are now defined
    });
```

MIXINS

```
function DetailMixin() {
    this.edit = function(model) {
    // switch to edit mode
    };

    this.save = function(model) {
    // save
    // show validation / view mode
    };
});
```

```
angular.module('myApp').controller('myMixinController',
    function($scope) {

        angular.extend($scope, new DetailMixin());
        // $scope.edit and $scope.save are now defined
    });
```

JAVASCRIPT OBJECTS

```
function Details() {
    this.edit = function(model) {
        // switch to edit mode
    }

    this.save = function(model) {
        // save
        // show validation / view mode
    }
});
```

```
angular.module('myApp').controller('myMixinController',
    function($scope) {

        // no dependency injection!
        $scope.details = new Details();

        // $scope.details.edit and $scope.details.save are now defined
    });
```

myMixin.js

ANGULAR SERVICES

```
angular.module('myApp').factory('myDetailService', function($http) {
    return {
        edit: function(model) {
            return angular.copy(model);
        },
        save: function(model) {
            // http post / put
            // return promise
        }
    };
});
```

```
angular.module('myApp').controller('myServiceController',
    function($scope, myDetailService) {
        $scope.model = {name: 'MyName'};

        $scope.edit = function() {
            $scope.editModel = myDetailService.edit($scope.model);
        };

        $scope.save = function() {
            myDetailService.save($scope.editModel).then(function(data) {
                console.log('success');
            });
        };
    });
```

HELPER CONTROLLERS

```
function DetailManager(myService) {
    var myModel;
    var myEditModel;

    this.init = function(model) {
        myModel = model;
    };

    this.edit = function() {
        myEditModel = myService.edit(myModel);
        return myEditModel;
    };
});
```

```
angular.module('myApp').controller('myHelpedController',
    function($scope, $controller) {
        $scope.detailManager = $controller(DetailManager);
        $scope.detailManager.init({name: 'myTestName'});
        // call $scope.detailManager.edit(); without parameter
    });
```

CACHING

reduce the amount / payload of requests

STATIC RESOURCE CACHING

revision your files!

just append a content hash to your file names:
app.js -> app-cd8al3f.js

allow the client to cache these files indefinitely

TEMPLATE CACHING

convert

```html
<div class="container">
    <!-- some html -->
</div>
```

myView1.tpl.html

```html
<div class="container">
    <!-- some other html -->
</div>
```

myView2.tpl.html

to

```javascript
(function(){
    angular.module('myApp').run(
        function($templateCache) {
            $templateCache.put('myView1.tpl.html',
                '<div class="container">\n'+
                '<!-- some html -->\n'+
                '</div>');
        });
    <!-- myView2.tpl.html -->
})();
```

myViews.tpl.js

DATA CACHING

utilize 3rd party libraries like angular-cache

(part of angular-data starting with version 3.0.0)

provide similar possibilities like guava cache

HOW TO LAZY LOAD YOUR CODE

not officially supported yet!
but still possible with some workarounds

FIRST STEP: SAVE REFERENCES TO PROVIDERS

```
window.providers = {};
angular.module('lazyLoadingApp', ['ngRoute'])
        <!-- We also depend on ngRoute -> see 4th step -->
        .config(function($controllerProvider, $compileProvider) {
            window.providers.$controllerProvider = $controllerProvider;
            window.providers.$compileProvider = $compileProvider;
        });
```

app.js

SECOND STEP: USE SAVED PROVIDERS

```
window.providers.$controllerProvider
        .register('lazyLoadedController', function($scope) {
            $scope.aLazyLoadedFunction = function() {
                alert('called a function on a lazy loaded controller');
            };
        });
```

lazyLoadedController.js

```
window.providers.$compileProvider
        .directive('lazyLoadedDirective', function() {
            return {
                restrict: 'E',
                replace: true,
                template: '<div>Lazy loaded directive</div>'
            };
        });
```

lazyLoadedController.js

THIRD STEP: USE LAZY STUFF IN VIEW

```html
<div ng-controller="lazyLoadedController">
    <div>Let's use our lazy stuff:</div>
    <lazy-loaded-directive></lazy-loaded-directive>
    <button ng-click="aLazyLoadedFunction()">Call function!</button>
</div>
```

lazyLoaded.tpl.html

FOURTH STEP: LOAD OUR VIEW / INITIALIZE OUR STUFF

To initialize our lazy loaded code we use the 'resolve' property of routes.

For the actual lazy loading of the files itslef we use $script.js for this example.

```
angular.module('lazyLoadingApp')
        .config(function($routeProvider) {
            $routeProvider.when('/', {
                template: 'dashboard.tpl.html'
            }).when('/lazy', {
                template: 'lazyloaded.tpl.html',
                resolve: {
                    deps: function($q, $rootScope) {
                        var deferred = $q.defer();
                        var dependencies = [
                            'lazyLoadedController.js',
                            'lazyLoadedDirective.js'
                        ];

                        $script(dependencies, function() {
                            // all dependencies have been loaded
                            $rootScope.$apply(function() {
                                deferred.resolve();
```

app.js

3RD PARTY LIBS

don't reinvent the wheel!

- lazy loading -> OcLazyLoad
- data caching -> angular-cache / angular-data
- Twitter Bootstrap -> angular-ui / angular-strap
- templates 2 js -> gulp-ng-html2js / >grunt-html2js

...

THINGS NOT COVERED

but still worth mentioning

BUILD SYSTEM

WHAT IT DOES

- test execution
- concatenate JS files
- minify/uglify CSS/JS files
- revision files (unique name for caching)
- wrap angular template in js files
- live reloading during development
- js dependency management
- ...

BUILD SYSTEM

WHAT WE USE

- TeamCity
- gradle
- node.js
- gulp.js
- bower
- karma

LETS LOOK AT AN APP

IF YOU WANT MORE

LINZ - TECHNOLOGIEPLAUSCHERL

- various topics - not only angular
- details and info at technologieplauscherl.at

VIENNA - ANGULAR JS MEETUP

- focus on angular topics
- details and info at www.meetup.com/AngularJS-Vienna/

THAT'S IT!

THANKS FOR YOUR ATTENTION

Any questions?
I'll give my best to answer them

Github: tscheinecker
Github: +ThomasScheinecker