Readable Code and AngularJS

John Papa
http://johnpapa.net
Twitter: @john_papa

pluralsight
hardcore developer training

App Structure is Important

```
/client
    /app
        /avengers
        /blocks
            /exception
            /logger
            /router
        /core
        /dashboard
        /data
        /layout
        /widgets
        app.js
    /bower_components
    /content
    .bower.json
    index.html
```

More Efficient When We're Organized

Finding Your Code in a Mess

What's Inside Really Matters

```
4   function SpeakerDetail($location, $scope, $routeParams, $window,
5                           common, config, datacontext, model) {
6       /*jshint validthis: true */
7       var vm = this;
8       var entityName = model.entityNames.speaker;
9       var logger = common.logger;
10      var $q = common.$q;
11      var wipEntityKey;
12
13      vm.cancel = cancel;
14      vm.goBack = goBack;
15      vm.hasChanges = false;
16      vm.isSaving = false;
17      vm.save = save;
18      vm.speaker = null;
19      vm.speakers = [];
20
21      Object.defineProperty(vm, 'canSave', { get: canSave });
22
23      activate();
24
25      function activate() {
26          onDestroy();
27          onHasChanges();
28          datacontext.ready([getRequestedSpeaker()]).then(onEveryChange);
29      }
```

Choose Your Coding Style

Chaining

Hoisting

Anonymous functions

Named functions

Global variables

Extending modules

IIFE

Module Variables

```
// create a module variable
var app = angular.module('app');
```

```
// create a variable using the setter works too
var app = angular.module('app', ['ngRoute']);
```

Variables or Chaining

```
// reference the module via a variable
var app = angular.module('app');

app.controller('AvengerDetail', function(){
    var vm = this;
    vm.name = 'Tony Stark';
});
```

```
// chaining to reference the module
angular
    .module('app')
    .controller('AvengerDetail', function (){
        var vm = this;
        vm.name = 'Tony Stark';
    });
```

Anonymous Functions?

Anonymous or Named Functions

```
// chaining and anonymous functions
angular
    .module('app')
    .controller('AvengerDetail', function (){
        var vm = this;
        vm.name = 'Tony Stark';
    });
```

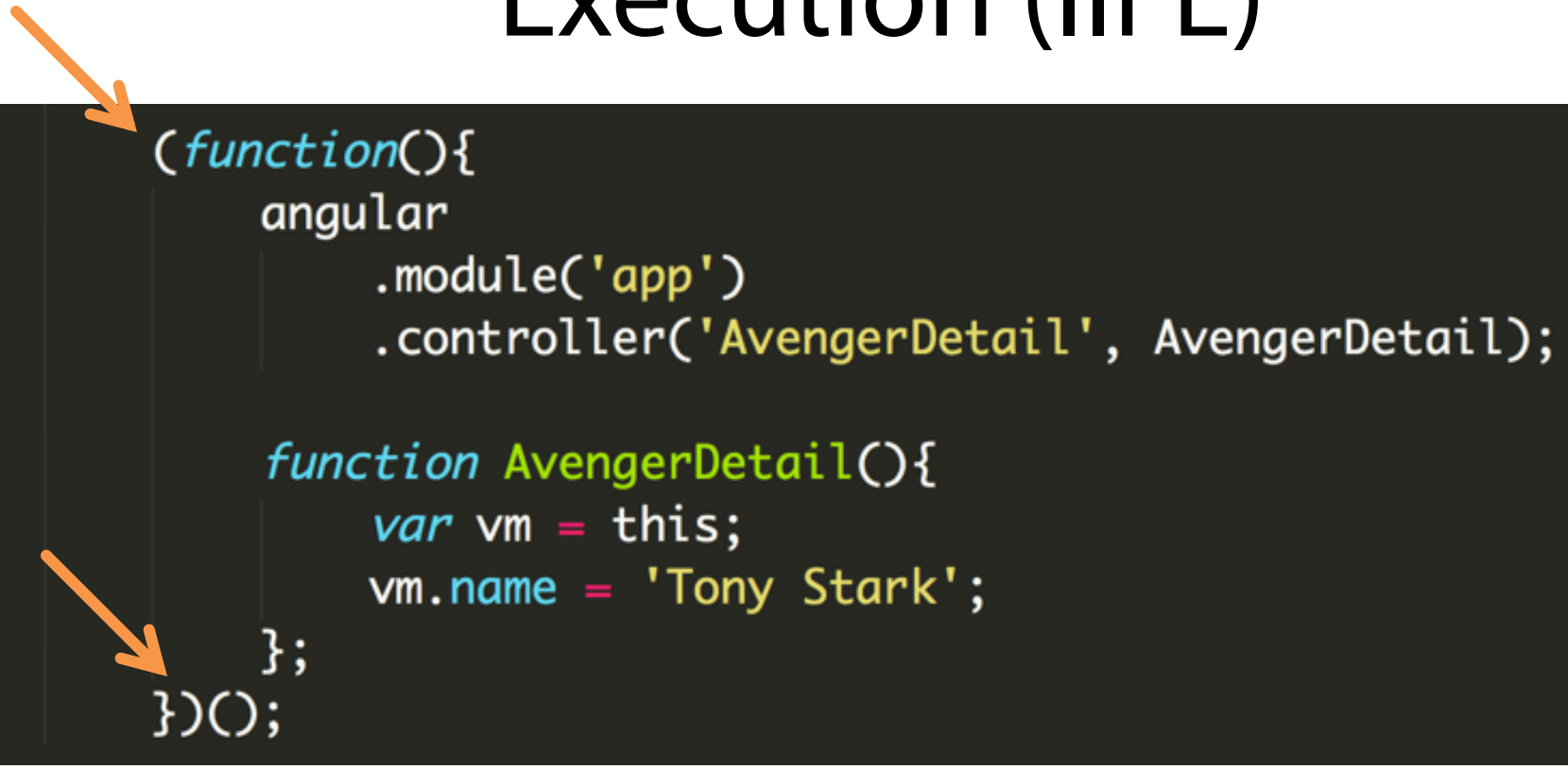
```
// chaining and a separate hoisted function
angular
    .module('app')
    .controller('AvengerDetail', AvengerDetail);

function AvengerDetail() {
    var vm = this;
    vm.name = 'Tony Stark';
}
```

Global Variables Running Wild

How Do You Contain Globals?

Immediately Invoked Function Execution (IIFE)



```
(function(){
    angular
        .module('app')
        .controller('AvengerDetail', AvengerDetail);

    function AvengerDetail(){
        var vm = this;
        vm.name = 'Tony Stark';
    };
})();
```

Safely Minify Dependencies ?

```
angular
    .module('app.dashboard')
    .controller('Dashboard', Dashboard);

function Dashboard(common, dataservice) {
    // controller code goes here
}
```

When minified these may become a and b, which won't be found

Inline Dependency Array

```
angular
    .module('app.dashboard')
    .controller('Dashboard', ['common', 'dataservice', Dashboard]);

function Dashboard(common, dataservice) {
    // controller code goes here
}
```

Minified

```
angular
    .module('app.dashboard')
    .controller('Dashboard', ['common', 'dataservice', Dashboard]);

function Dashboard(a, b) {
    // controller code goes here
}
```

Many Right Ways, Choose Your Pattern

Register and Inject, then Function

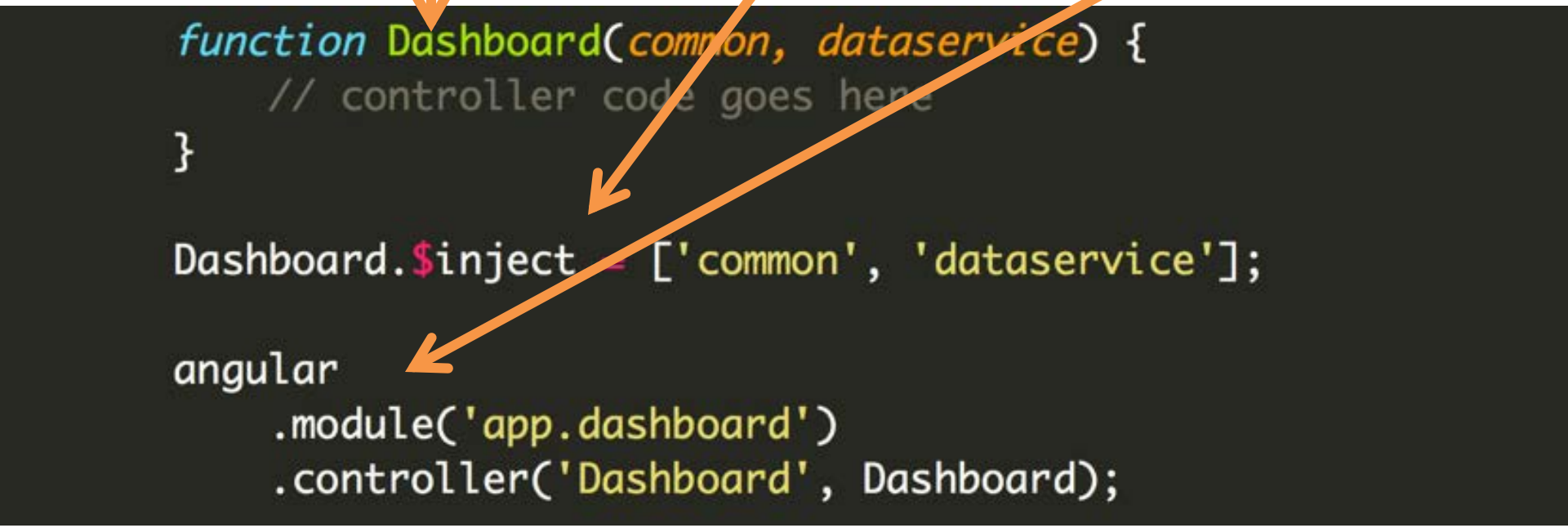
```
angular
    .module('app.dashboard')
    .controller('Dashboard', ['common', 'dataservice', Dashboard]);

function Dashboard(common, dataservice) {
    // controller code goes here
}
```

"I prefer this option. The mapping to the function is straightforward, it is commonly used, and the injection is implied rather than explicit"

Jesse Liberty @jesseliberty
Falafel Software Master Consultant

Function, Inject, Register



```
function Dashboard(common, dataservice) {
    // controller code goes here
}

Dashboard.$inject = ['common', 'dataservice'];

angular
    .module('app.dashboard')
    .controller('Dashboard', Dashboard);
```

"I prefer this sequence: write my function, inject the dependencies, ship it off into the app. It makes sense from hoisting and declarative perspectives."

Todd Motto @toddmotto
Front End Developer

Register, Inject, Function

```
angular
    .module('app.dashboard')
    .controller('Dashboard', Dashboard);

Dashboard.$inject = ['common', 'dataservice'];

function Dashboard(common, dataservice) {
    // controller code goes here
}
```

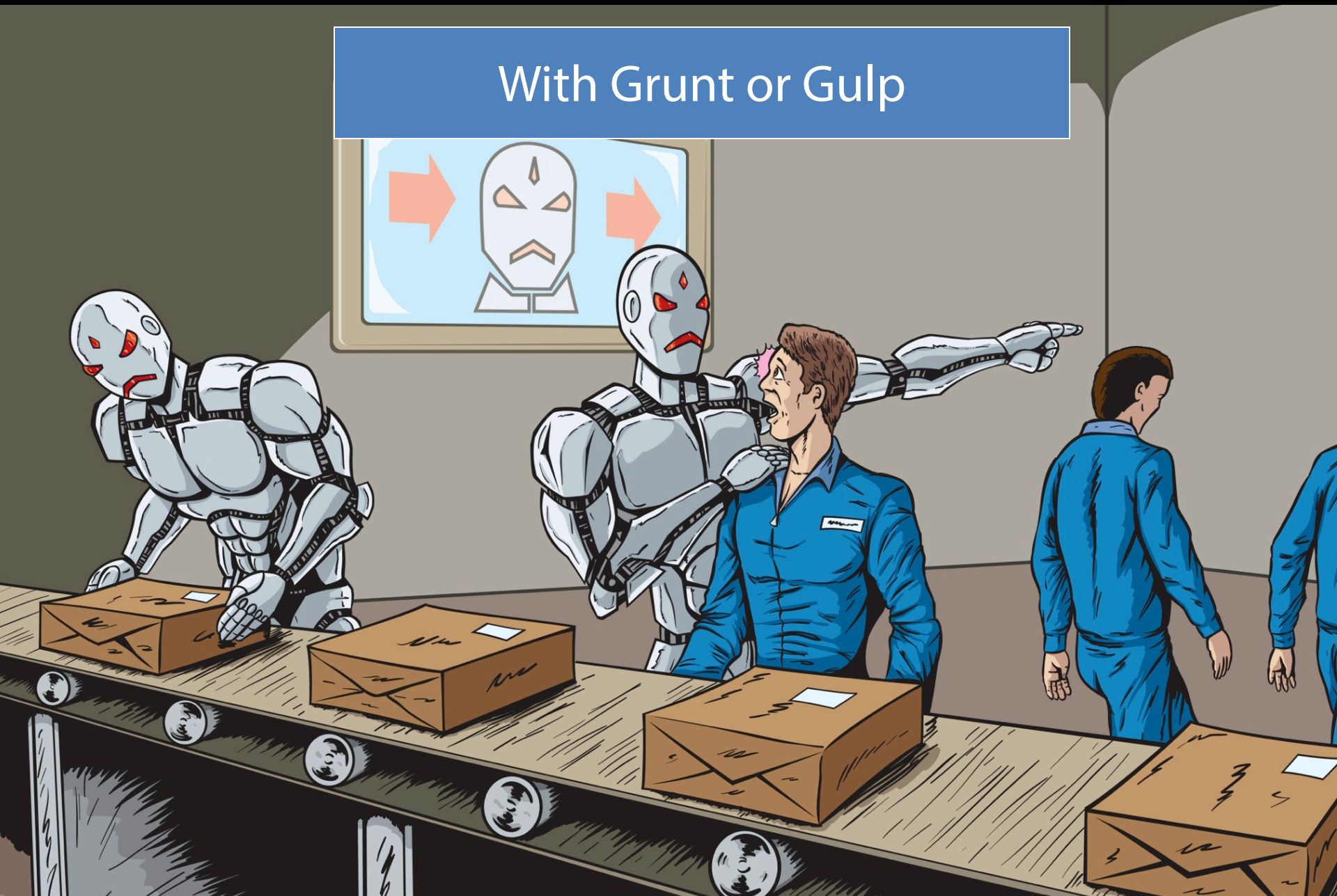
"I prefer to register, then see the injection list right with the function dependency list. Followed by the implementation details"

John Papa @john_papa

Which is Best ?

Choose a pattern that is easier to read and write, and be consistent

Automation is Great Option



Inject Later with ngAnnotate

```
angular
    .module('app.dashboard')
    .controller('Dashboard', Dashboard);

/* @ngInject */
function Dashboard(common, dataservice) {
    // controller code goes here
}
```

Enhancing Readability

IIFE's help reduce global variables and functions

Chaining and indentation

Named functions help readability

Keep important code up top including interfaces

Inject consistently

Consider ngAnnotate using Grunt or Gulp