AngularJS 101

Connor Simmons – connorlsimmons@gmail.com, LinkedIn
Geoff Alkire – geoff.alkire@gmail.com, LinkedIn
Jesse David – jesse.p.david@gmail.com, LinkedIn

Introduction

Core Concepts

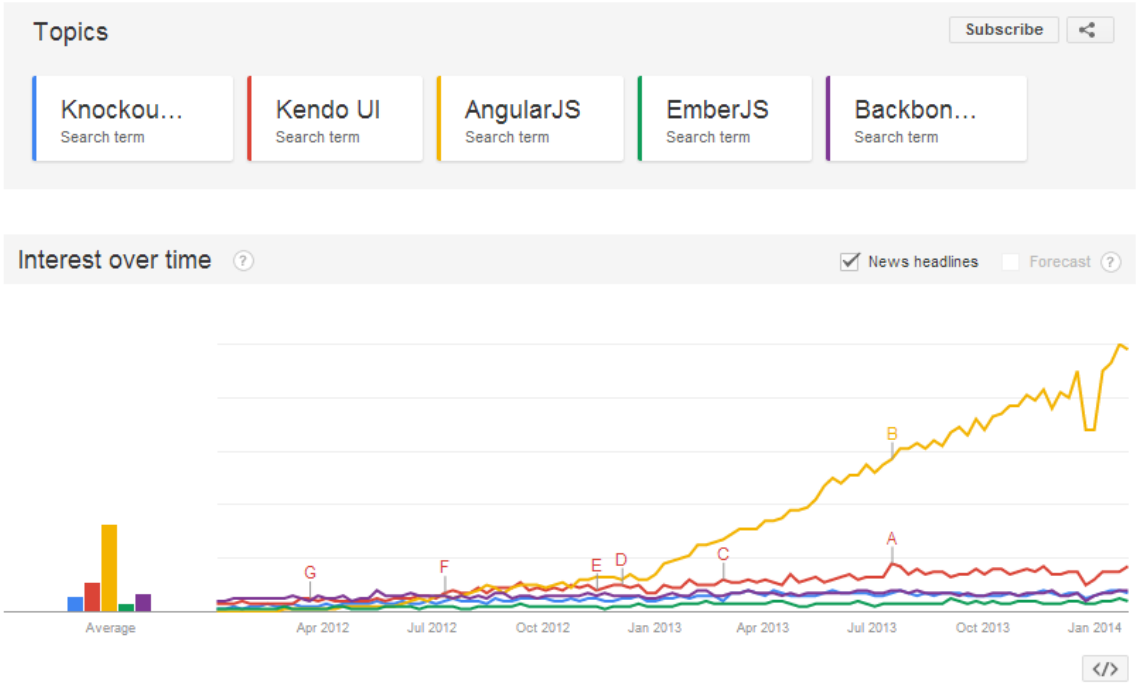
Directives*

Conclusion

Introduction

Angular is a powerful, rapidly growing, client-side development framework

- Maintained by Google – open source
- Huge development community



Google trends search

4

Why JavaScript frameworks?

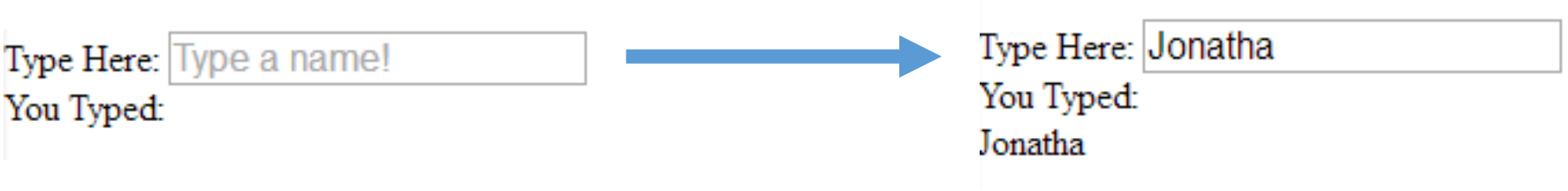
- Modern browsers & REST APIs
  - Browser JS engines
  - HTML5 and CSS3
  - Mobile browsers on par with desktop
- Base JavaScript & jQuery are too low level
  - jQuery is great for DOM manipulation, doesn't scale with app complexity
- Modularity, abstraction, testability
- Need for cross-platform
- Single-page application architecture
  - Gmail (Mail.google.com) great example of this

5

Core Concepts

Like many frameworks, two-way data-binding is a core concept of Angular

- Data is shared between the DOM and JavaScript
- Keeps view-related logic out of JavaScript



7

Like many frameworks, two-way data-binding is a core concept of Angular

jQuery

HTML

```
<input id="myInput"></input>
<h1 id="myInputTitle"></h1>
```

Javascript

```
function watch(obj, prop, handler) {
            var currval = obj[prop];
            function callback() {
                        if (obj[prop] != currval)
                              {
                                          var temp = currval; currval = obj[prop];
                                          handler(temp, currval);
                              }
                  }
                  return callback;
      }

var myhandler = function (oldval, newval) {
 //do something
};

var intervalH = setInterval(watch(myobj, "a", myhandler), 100);

myobj.set_a(2);
```


8

Like many frameworks, two-way data-binding is a core concept of Angular

AngularJS

HTML

```
<input ng-model="myInput"></input>
<h1 ng-bind="myInput"></h1>
```

Javascript

```
None.
```


9

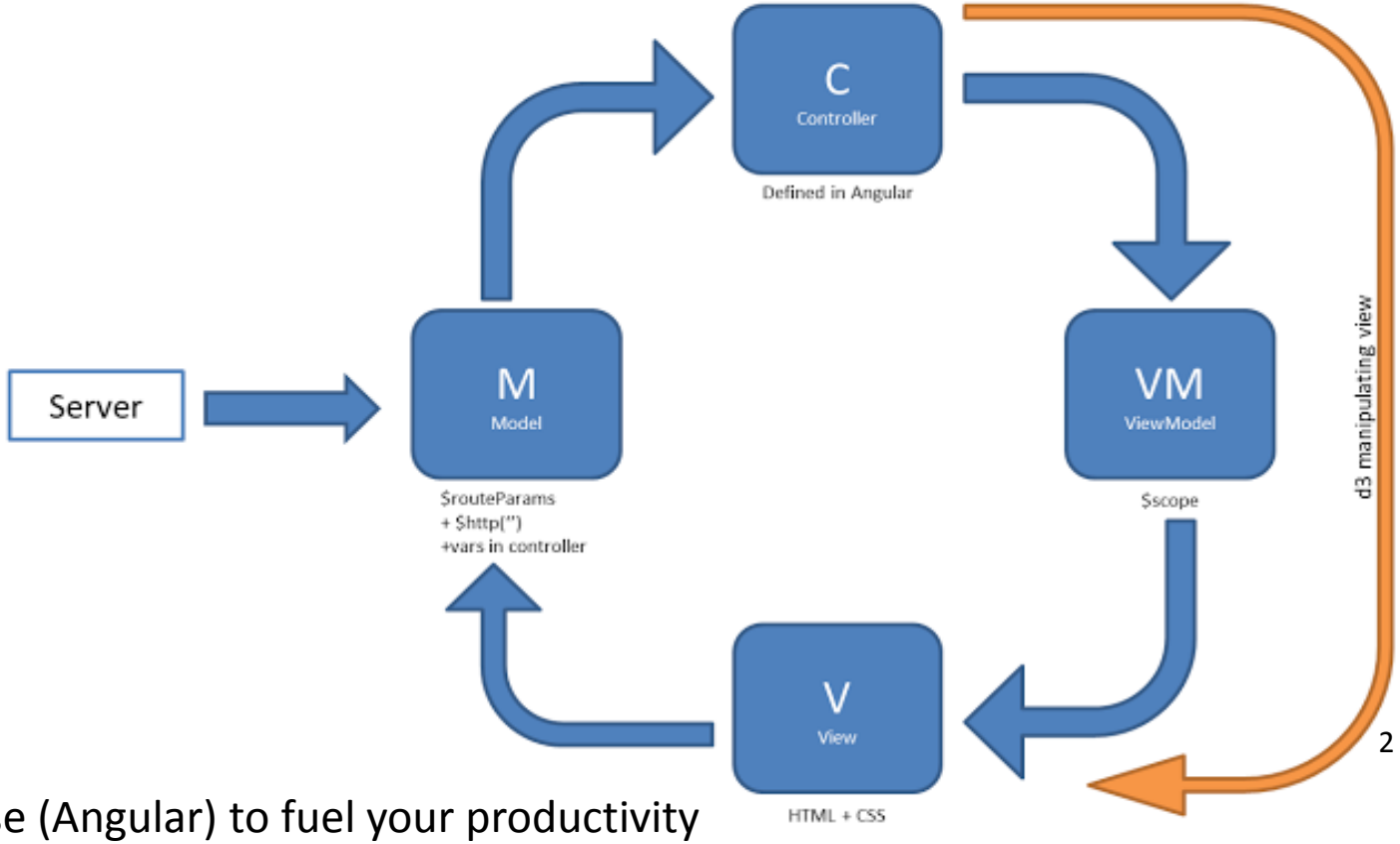
DEMO

Examples:

http://jsfiddle.net/az7r88za/

http://jsfiddle.net/w9yt1r61/3/

Angular considers itself a “Model-View-Whatever” framework



“Please use (Angular) to fuel your productivity and application maintainability rather than heated discussions about things that at the end of the day don't matter that much.”[1]

1. https://plus.google.com/+AngularJS/posts/aZNVhj355G2
2. http://chuckstangledweb.blogspot.com/

11

Dependency injection is provided out-of-the-box

- Simple Definition: Easily include references to other helpers, modules, interfaces, etc.
  - Has been around in “older” languages like Java
- Lays the Angular framework
- Simplifies adding/removing of modules
  - At run-time or compile-time
- **Provides for easily testable code**

(Boring definition from Wikipedia)

Dependency injection is a software design pattern in which one or more dependencies (or services) are injected, or passed by reference, into a dependent object (or client) and are made part of the client's state. The pattern separates the creation of a client's dependencies from its own behavior, which allows program designs to be loosely coupled…

DEMO

Directives (the best part)

Directives are Angular's way of defining reusable web components

- Arguably Angular's most powerful feature
- Markers on a DOM element that grant new behaviors
- Modularizes HTML

```
<div ng-repeat="customComp in getCustomCompetencies()">
    <custom-competency-slider
        color-group-id="colorGroupId"
        custom-competency="customComp"
        delete-competency="deleteCustom">
    </custom-competency-slider>
</div>
```

Angular comes with many useful directives already defined

- ng-show & ng-hide

```
<div ng-show="user.IsManager">
```

- ng-click

```
<div class="skills-charts" ng-click="magnify()">
```

- ng-bind & ng-model

```
<input type="text" ng-model="user.FirstName" placeholder="First Name">

<h1 class="inline"><span ng-bind="user.FirstName"></span>'s Profile></h1>
```

Developers can also create their own!

Directive Usage

```
<div has-role="Resources" itacademy-skill-detail-resources></div>
```

Directive Definition

```
.directive('hasRole', function (securityRoles) {
    return { restrict:'A',
        link: function (scope, element, attrs) {
            var value = attrs.hasRole.trim();
            function toggleVisibilityBasedOnPermission() {
                if (!securityRoles.hasPermission(value)) {
                    element.remove();
                }
            }
            toggleVisibilityBasedOnPermission();
        }
    };
})
```

Security Role Service Definition

```
.factory('securityRoles', function () {
    var rolesList;
    return {
        setRoles: function (roles) {
            rolesList = roles;
        },
        hasPermission: function(role) {
            var permission = false;
            if (rolesList != null) {
                for (var i = 0; i < rolesList.length; i++) {
                    if (rolesList[i].Name === role) {
                        permission = true;
                    }
                }
            }
            return permission;
        }
    };
});
```


17

DEMO

Testing

- Angular provides for unit-testable code through dependency injection.
- This means that our controllers, services, factories, and directives can be unit tested.
- Many libraries are available to assist in this task:
  - JasmineJS – A unit testing framework
  - KarmaJS – A unit test runner
  - PhantomJS – A headless browser

```
describe("myFunction", function() {
    var myfunc = NS.myFunction;

    beforeEach(function(){
        spyOn(myfunc, 'init').andCallThrough();
    });

    afterEach(function() {
        myfunc.reset();
    });

    it("should be able to initialize", function() {
        expect(myfunc.init).toBeDefined();
        myfunc.init();
        expect(myfunc.init).toHaveBeenCalled();
    });

    it("should populate stuff during initialization", function(){
        myfunc.init();
        expect(myfunc.stuff.length).toEqual(1);
        expect(myfunc.stuff[0]).toEqual('Testing');
    });
    //will insert additional tests here later
});
```

http://kroltech.com/2013/11/javascript-tdd-with-jasmine-and-karma/

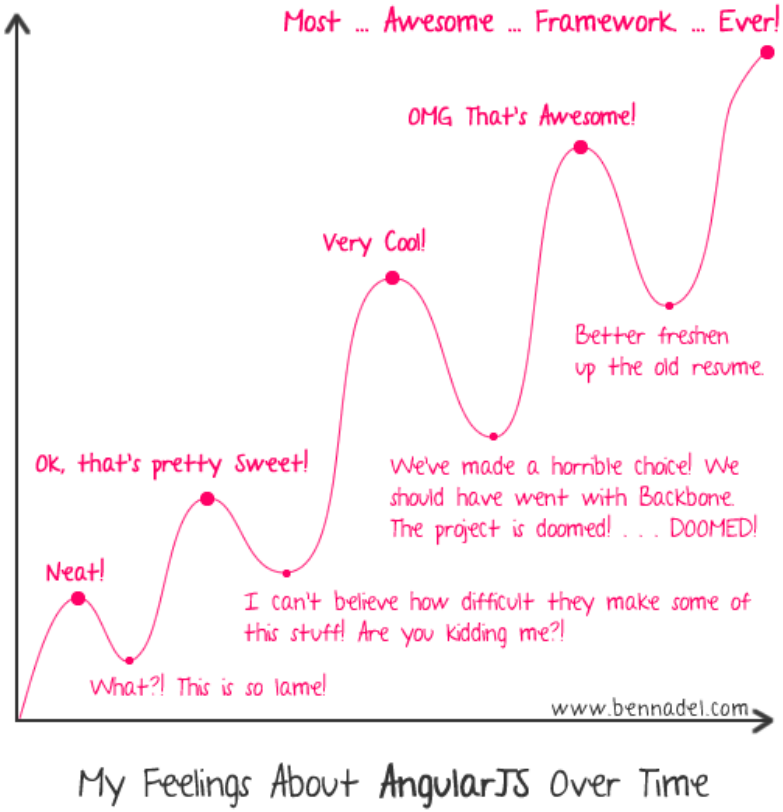
19

DEMO

Conclusions

While powerful, there have been gotchas and problems encountered

- There is a learning curve
- Error messages are not always clear
- Lack of support for IE8 (in newer versions)
- Bugs



My Feelings About AngularJS Over Time

22

Practical experience: Lessons learned

- Learn, understand, and use directives from the get-go
  - No matter how intimidating they seem, and how easy it might seem to build your app without them, you will thank yourself later!
- Pick a unit testing library and do TDD
  - Again, this will pay big dividends later.
- Determine a good way to manage your data interactions
  - This could mean picking a library to manage it for you (BreezeJS) or architecting your own leveraging Angular's built-in components

Questions?

24

Appendix

References

- http://www.egghead.io/ - Excellent Angular JS video course by John Lindquist
- http://angularjs.org/ - Angular JS official web-site
- http://docs.angularjs.org/api/ - Angular JS API reference
- https://groups.google.com/group/angular/ - Angular JS Google Group
- http://angular-ui.github.io/bootstrap/ - Twitter Bootstrap Angular plugins
- http://joshdmiller.github.io/ng-boilerplate/ - Angular Boilerplate

26

References

- https://github.com/simmons6/angular-demo - Demo application used in presentation

27