Preparing for AngularJS 2.0

Dr. Mike Hopper

WDI Instructor

General Assembly

About Me

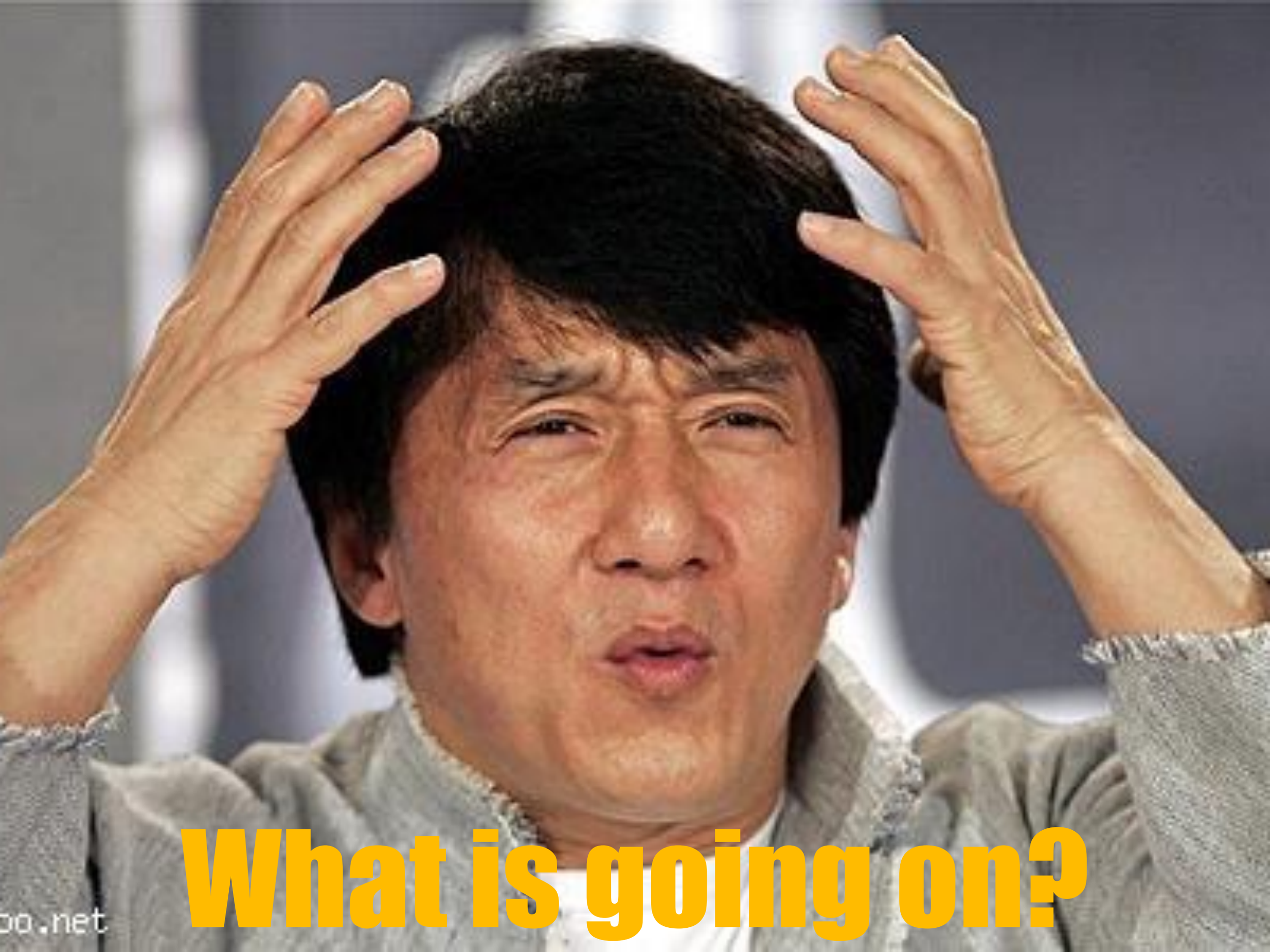
- Instructor at GA
  - Teach full stack web app development using JavaScript, RoR, NodeJS, PostgreSQL, MongoDB
- Over 20 years of experience as a Software Architect and Lead Developer
- Love to Learn and Improve
- I've been using AngularJS for about 2 years (since March 2013)
- Twitter: @drmikeh

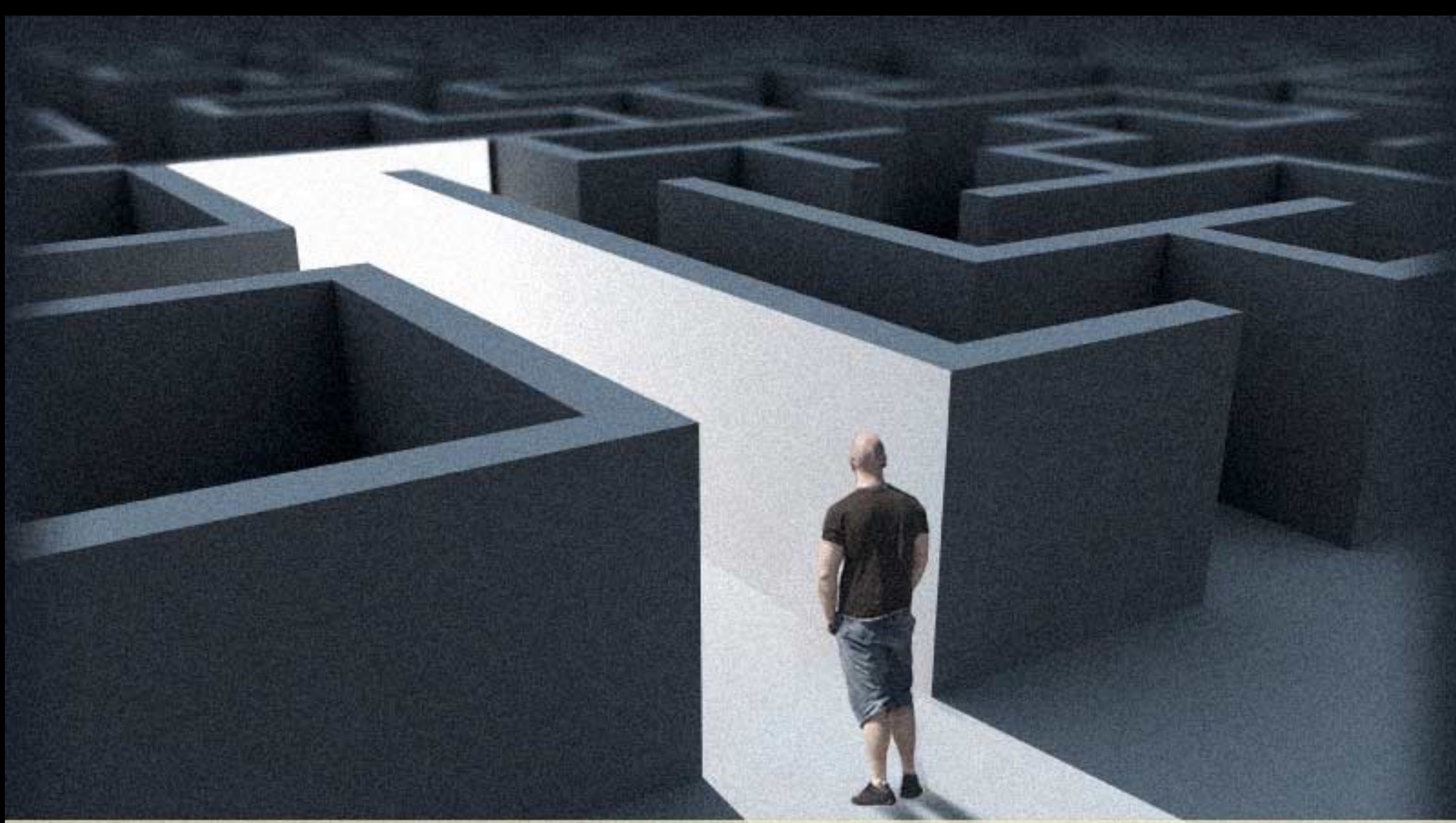
Angular 2.0

- Is it better?
- Why is it so different?
- When will it be released?
- Can I use it today?
- What language should I use?
- Is there a migration strategy?
- What about tooling?

The Web, it is a changin'

- ~~HTML 6~~ (uhh, ehh)
  The New HTML
  - Web Components
  - Shadow DOM
- ECMAScript 6
  - classes, modules
- TypeScript
- AtScript, Annotations
- Transpilers
  - Traceur, Babel


What is going on?



where do I start?

Benefits of AngularJS 2

Simpler

Less moving parts

More consistent

Simpler Mental Model

Better Encapsulation

Thus better support for
(web) components

Improved Performance

Easier

Testing

Better Integration with Underlying Technologies

ES6 Module System

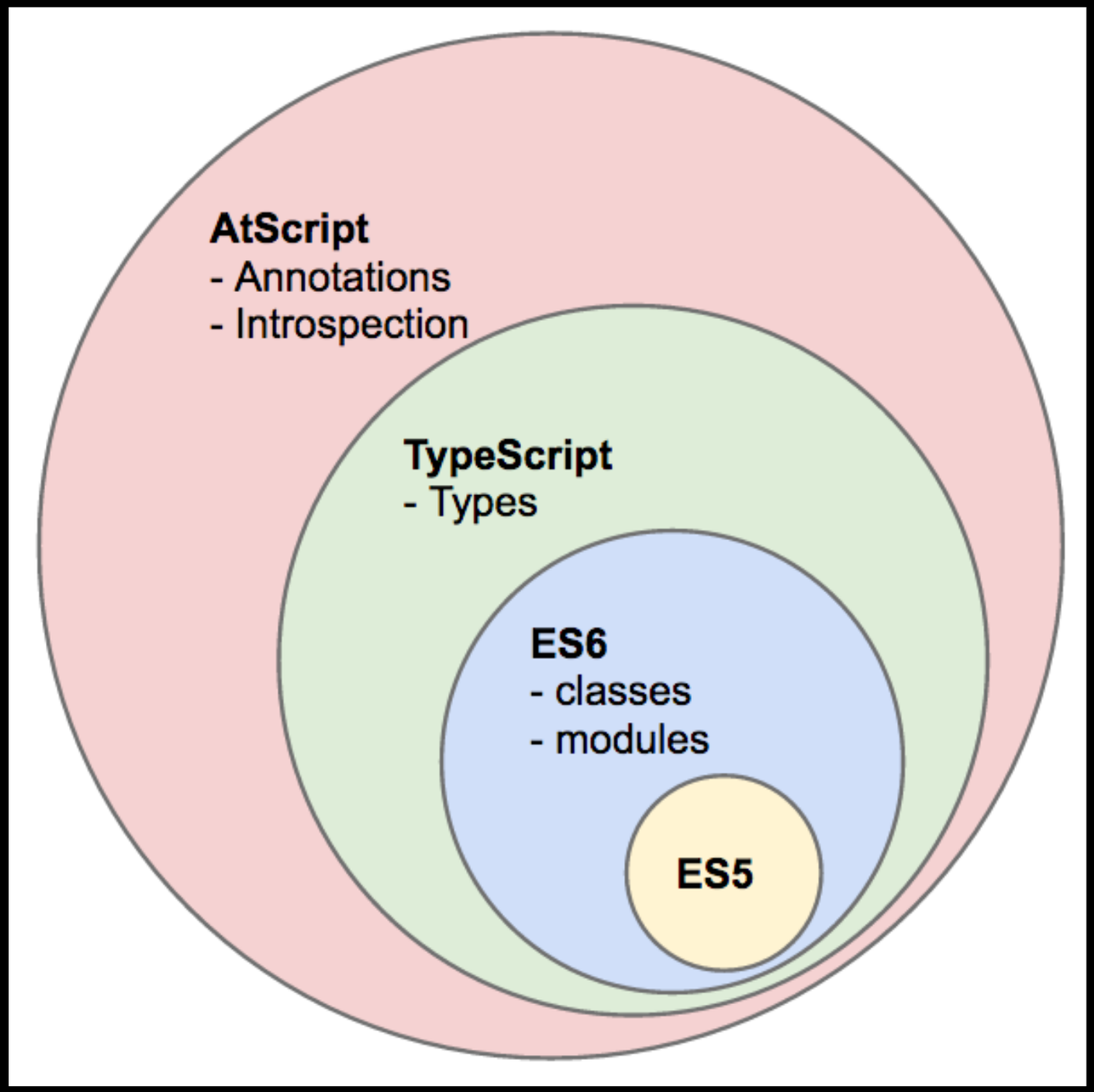
Binds to Native DOM properties and events

I’m so excited.

Agenda

- Languages
  - ES6
  - TypeScript
  - AtScript
- AngularJS 2
  - Classes, Components, Annotations
  - New HTML Syntax
- Demos
- Wrap up

Languages



- The AtScript features have been rolled into TypeScript.
- The annotation / decorator metadata and some type metadata is available at runtime.

This diagram is slightly out-of-date.

Let’s Look at Some Code!

ES6

ES6 Classes and Modules

```
1. // items.ts
2. var items = [ 'Groceries', 'Lawn', 'Exercise' ];
3. export {items};
```

```
1. // item_mgr.ts
2. export class ItemManager {
3.   items;
4.   constructor(items) {
5.     this.items = items;
6.   }
7.   print() {
8.     console.log(this.items);
9.   }
10. }
```

```
1. // main.ts
2. import {items} from './items';
3. import {ItemManager} from './item_mgr';
4. var itemMgr = new ItemManager(items);
5. itemMgr.print();
```

TypeScript

TypeScript

```
1. // items.ts
2. var items : string[] = [ 'Groceries', 'Lawn', 'Exercise' ];
3. export {items};
```

```
1. // item_mgr.ts
2. export class ItemManager {
3.   items : string[];
4.   constructor(items : string[]) {
5.     this.items = items;
6.   }
7.   print() {
8.     console.log(this.items);
9.   }
10. }
```

```
1. // main.ts
2. import {items} from './items';
3. import {ItemManager} from './item_mgr';
4. var itemMgr : ItemManager
5.   = new ItemManager(items);
6. itemMgr.print();
```

TypeScript

TypeScript is a superset of ES6.

Existing JavaScript / ES6 code that passes jshint is valid Typescript code.

```
1.  function greet(name : string) {
2.    return 'Hello ' + name;
3.  }
4.  // this makes TS happy!
5.  console.log(greet('Mike'));
6.
7.  // This makes TypeScript angry
8.  // but it will still play along!
9.  console.log(greet(123));
```

$ tsc angry.ts; node angry.js

angry.ts(9,19): error TS2345: Argument of type 'number' is not assignable to parameter of type 'string'.

Hello Mike

Hello 123 <— it still works (at least this time, but we should fix it)

```
// student.ts
class Student {
  fullName : string;
  // the public keyword declares properties!
  constructor(public firstName : string,
              public middleInitial : string,
              public lastName : string){
    this.fullName = firstName + " " + middleInitial + " " + lastName;
  }
  initials() {
    return this.firstName[0] + this.middleInitial[0] + this.lastName[0];
  }
}
// an interface
interface Person {
  firstName : string;
  lastName : string;
}
function greet(person : Person) {
  return "Hello, " + person.firstName + " " + person.lastName;
}

var student : Student = new Student("Michael", "A.", "Hopper");
console.log(greet(student));
console.log('Your initials are: ' + student.initials());
```

What about Components and Annotations?

Annotations

- Annotations are now called *Decorators*
- Are currently transpiled into ES5 properties that are available at runtime for introspection
- Used by AngularJS for:
  - dependency injection
  - `binding` a component to a custom HTML element
  - other uses as needed

A Simple AngularJS Hello Component in TypeScript

```
1.  @Component({
2.    selector: 'my-app'  // A CSS Selector
3.                        // binds to the <my-app></my-app> element
4.                        // we no longer have to match element and directive names!
5.  })
6.  @View({
7.    template: `<h1>Hello {{ name }}</h1>
8.               <label for="nameInputId">Name</label>
9.               <input id="nameInputId" #name_input
10.                     (keyup)="keyPressed($event, name_input)">
11.              `
12. })
13. class MyAppComponent {
14.   name : string;
15.   constructor() {
16.     this.name = 'World';
17.   }
18.   keyPressed($event, input) {
19.     if ($event.which === 13) {
20.       this.name = input.value;
21.     }
22.   }
23. }
```

Question:
Is this simpler than writing an AngularJS 1.x Directive Definition Object (DDO)?

That’s it for
ES6 and
TypeScript

What about the new AngularJS Template Syntax?

```
1. <div style="padding:5px" *for="var item of items">
2.   <input type="checkbox" #chkbox
3.          [checked]="item.completed"
4.          (click)="setCompleted(item, chkbox.value)">
5.   {{item.text}} <a (click)="removeItem(item)">Remove</a>
6. </div>
7.
8. <label for="description">New Item</label>
9. <input id="description" #desc
10.        (keyup)="keyPressed($event, desc)">
11.<button type="button"
12.        (click)="addItem(desc)">Add Item</button>
```

Noooooooooo!

Why?

```
1. <div style="padding:5px" *for="var item of items">
2.   <input type="checkbox" #chkbox
3.          [checked]="item.completed"
4.          (click)="setCompleted(item, chkbox.value)">
5.   {{item.text}} <a (click)="removeItem(item)">Remove</a>
6. </div>
7.
8. <label for="description">New Item</label>
9. <input id="description" #desc
10.        (keyup)="keyPressed($event, desc)">
11.<button type="button"
12.        (click)="addItem(desc)">Add Item</button>
```

- This syntax is actually *much better*!
- Notice how the bracketed attributes map to the standard DOM properties.
- [ ] is used for *expressions* (data binding)
- ( ) is used for *statements (events)*

The Following are ***Not Needed*** in AngularJS 2.0

- Properties
  - ng-bind ng-bind-html ng-bind-template
  - ng-class ng-class-even ng-class-odd
  - ng-style
  - ng-disabled ng-readonly ng-selected
  - ng-show ng-hide
  - ng-src ng-srcset ng-href
  - ng-value
- Events
  - ng-blur
  - ng-change ng-checked
  - ng-copy ng-cut ng-paste
  - ng-click ng-dblclick
  - ng-open ngsubmit
  - ng-keydown ng-keypress ng-keyup
  - ng-mousedown ng-mouseenter ng-mouseleave
  - ng-mousemove ng-mouseover ng-mouseup

Other Stuff ***Not Needed*** in AngularJS 2.0

- $scope
- angular.module
- controllers,
  services, factories
- DDO

Simpler

Less moving parts

More consistent

Simpler Mental Model

Better Integration with Underlying Technologies

ES6 Module System

Binds to Native DOM properties and events

I’m so excited.

Demo Apps

- Quickstart
  - Prints a "Hello, {{name}}“ greeting.
  - Transpiles AtScript at *build-time* via *tsc*
  - https://angular.io/docs/js/latest/quickstart.html
- ng2-ps-webinar
  - Joe Eames TODO app
  - Transpiles AtScript at *runtime*
  - video: *https://www.youtube.com/watch?v=-8P8NO8X-mQ*
  - code: https://github.com/joeeames/ng2-ps-webinar.git

Getting Started

- Learn ES6 and TypeScript
- Do the AngularJS 2 - Five Minute Quickstart
  - https://angular.io/docs/js/latest/quickstart.html
- Take a look at the Yeoman Angular2 generator:
  - https://github.com/swirlycheetah/generator-angular2
- Look for interesting tutorials, articles, and videos, especially those from ng conferences.

Resources

- AngularJS 2 - Five Minute Quickstart
  - https://angular.io/docs/js/latest/quickstart.html
- AngularJS 2 Alpha Releases:
  - npm install -g tsd;
  - tsd query angular2 --action install
  - https://code.angularjs.org/ (scroll to the bottom)
- Joe Eames TODO app
  - https://github.com/joeeames/ng2-ps-webinar.git
- Building a Zippy Component in Angular 2
  - http://blog.thoughtram.io/angular/2015/03/27/building-a-zippy-component-in-angular-2.html

Migration Strategies

- Angular's New Router - Component Router
  - Coming out with Angular 1.4
  - https://angular.github.io/router/getting-started
  - Will work with Angular 2
- "controller as" syntax
  - Make your existing controllers look more like their Angular 2 Component counterparts
- Run 1.x and 2.x Side by Side?
  - Rumor has it that Angular 1 and Angular 2 code will be able to work together
- For more info:
  - ng-vegas Conference Videos - www.ng-vegas.org

Questions