Angular 2.0
Router and DI Preview

Exciting, Flexible and Modern Routing and DI for 2.0

Presented by
Peter Pavlovich
Principal Architect
EnerNOC, Inc.
pavlovich@gmail.com

Introductions ...

- Peter Pavlovich
  - Front-End Evangelist and Technology Addict
  - Current: Principal Architect, EnerNOC, Inc. (Energy Intelligence Software)
    - Angular/Meteor/Polymer/Aurelia, Node, Scala/Java, OrientDB/Titan/MongoDB
- Entrepreneur, Open Source Contributor and Community Member
- Instructor, Mentor, Speaker and Leader.
- http://linkedin.com/in/peterpavlovich
- pavlovich@gmail.com

Session Plan

- Component Router
  - General Discussion
    - Using the new router!
      - in 2.0
      - in 1.X
- Dependency Injection

Component Router

Outline

- Brief History
- Current State of routing
- General
  - Basic Features
  - Dynamic Loading
  - Embedded Applications
  - Screen Activation
  - Auxiluary Routes
- Design principles
  - Construction via pipeline
  - How to customize for your app

History Lesson

History of the project

- Brian Ford + Rob Eisenberg (project leads)
- Researched Routers from other frameworks:
  - ngRoute 1.x
  - uiRouter
  - Ember.js's router
  - Durandal's router
  - Passport (express middleware)

History of the project

- Detailed design -> asked for feedback.
- Solicited Use cases from the community
- Ported to 1.3 -> 1.4 -> 1.5

Current Routing Options

NgRoute

Simple

No Nested Views

No Sibling Views

ui-router

Useful now

No migration path

Component Router inspired by it

Basic Features

Basic Features

- Configuration
- Default/missing route handling
- History Manipulation
- Configuration by convention
- Navigation Model
- Document Title Updates

Additional Features
(rumored/partial)

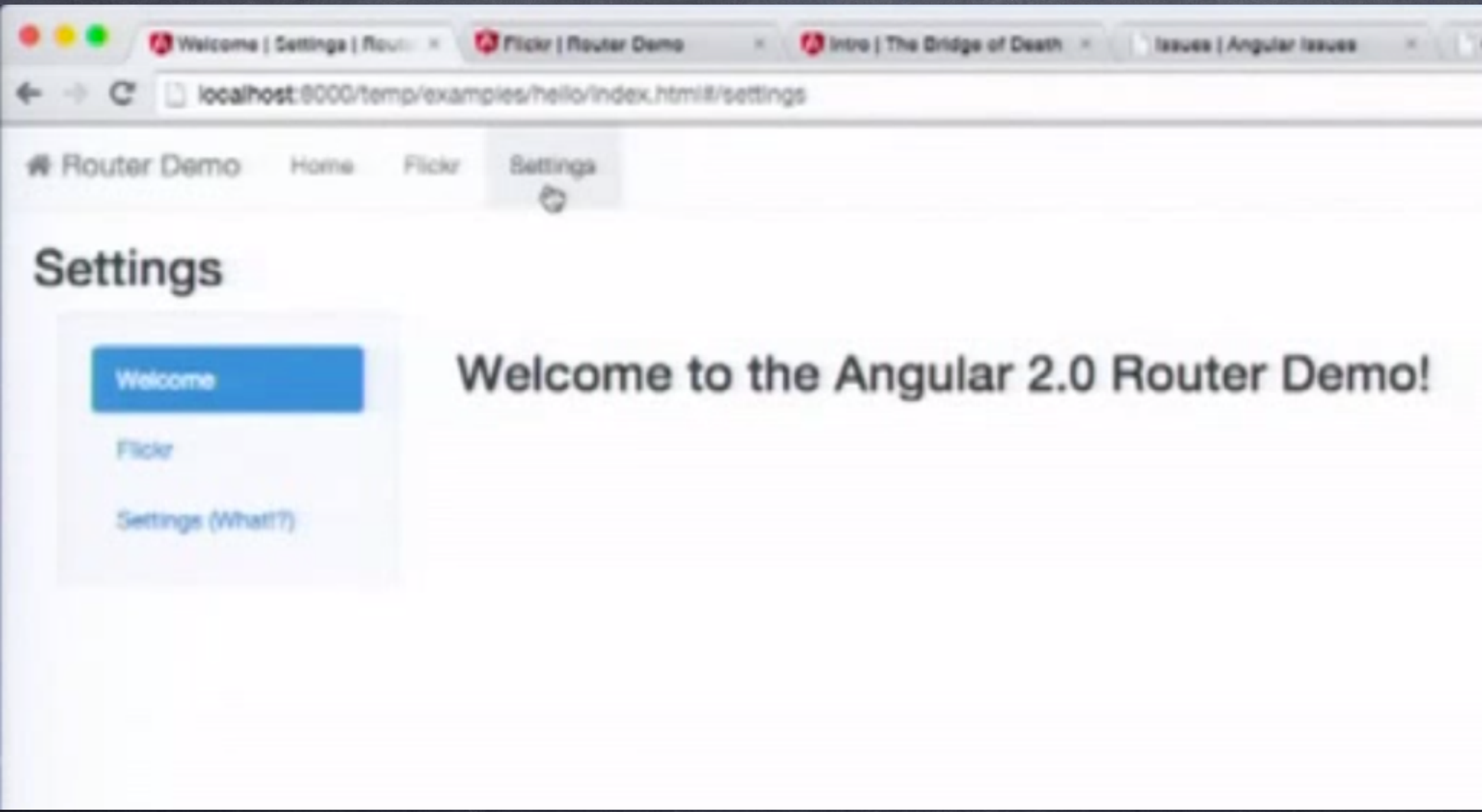
Dynamic Loading

Embedded Applications

Embedded (Child) Apps

- Top-level, App Router
- Child Routers
- State Management
- Component Reuse (optimization)
- Sibling Components/Controllers

Home | Router Demo
Flickr | Router Demo
Intro | The Bridge of Death
Issues | Angular Issues
localhost:8000/temp/examples/hello/index.html#/
Router Demo
Home
Flickr
Settings
Welcome to the Angular 2.0 Router Demo!


Welcome | Settings | Rout
Flickr | Router Demo
Intro | The Bridge of Death
Issues | Angular Issues
localhost:8000/temp/examples/hello/index.html#/settings
Router Demo
Home
Flickr
Settings
Settings
Welcome
Flickr
Settings (What!?)
Welcome to the Angular 2.0 Router Demo!

Welcome | Settings (What
Flickr | Router Demo
Intro | The Bridge of Death
Issues | Angular Issues
Comments | Issue 9742 |
localhost:8000/temp/examples/hello/index.html#/settings/settings/settings/settings/settings/settings
Router Demo
Home
Flickr
Settings
Settings
Welcome
Flickr
Settings (What!?)
Settings
Welcome
Flickr
Settings (What!?)
Settings
Welcome
Flickr
Settings (What!?)
Settings
Welcome
Flickr
Settings (What!?)
Settings
Welcome
Flickr
Settings (What!?)
Welcome to the Angular 2.0 Router Demo!

Welcome | Settings (Wha | Flickr | Router Demo | Intro | The Bridge of Death | Issues | Angular Issues

localhost:9000/#/issues/9742

Angular Issues

Number	Title
9743	docs(form.FormController): add 'date' token to FormController $error property
9742	$routeProvider - optional named group + trailing slash
9741	docs($templateCache): clarify inline template
9740	ngRepeat: $previous and $next
9738	docs(guide/Expressions): some punctuations are missing
9737	IE9 Angularjs not rendering the body and redirecting with hash # in url
9736	Docs: searching for $q only returns error/$q
9735	A scope question when transclude and isolate scope both exist
9733	Using a certain number of ng-src attributes on a page cause it to go blank
9732	(Performance) Array comprehensions - ish
9731	Route inherited param fix
9730	fix(location): throw a MinErr when base path has the wrong case #4082
9729	Add warning message when a module is reset/replaced

Welcome | Settings (What | Flickr | Router Demo | Intro | The Bridge of Death | Issues | Angular Issues

localhost:9000/#/issues/9742

Angular Issues

#9743 $routeProvider - optional named group + trailing slash

From ($routeProvider documentation)(https://docs.angularjs.org/api/ngRoute/provider/$routeProvider): > If $location.path conta
$location.path will be updated to add or drop the trailing slash to exactly match the route definition. It seems this does not work
~"/page/:name?"). In that case an encoded question mark will appear in the URL if you will navigate to ~"/page/index/ ~" (notice t
(http://plnkr.co/edit/hZ2pMLJuRCkkMc8Y!av0?p=preview).

Comments Events

Narretz

Seems to be the same issue as in https://github.com/angular/angular.js/pull/8200 But I'll keep it open in case the PR g

Screen Activation

Navigation lifecycle hooks

- canDeactivate: Allow deactivation?
  - deactivate: allows clean up after deactivation
  - canActivate: allow activation?
  - activate: component activated ... initialize?

Return values for hooks

- Hooks can return any of the following:
  - Boolean: true - allow, false - stop, revert.
  - NavigationCommand: take control of router
  - Promise: async return Boolean/Nav Command

Auxiliary Routes

AngularConnect: Routing in

Gmail

https://inbox.google.com

Inbox

Search

Today

Yesterday

This month

To

From briantford@gmail.com

Subject

Say something

SEND

Router as a Pipeline

Navigation Pipeline

- Navigation Instruction
- Navigation Context
- Pipeline

Customization

Customization Options

- Configuration/Conventions (use as you will)
- NavigationInstruction (steps: write your own!)
- NavigationCommand (write your own!)
- Pipeline (pull out / insert / replace steps)
- History (DI : pull it out and replace it!)
- Viewport

Using the Router!

Components in General

- consist of:
  - View
  - Controller
  - Router (optional)

Use in 1.4+

<script src="https://code.angularjs.org/1.5.0-beta.1/angular-route.js"></script>

ui-router vs comp router

ui.router	ngComponentRouter
$stateProvider.state()	$router.config()
<ui-view>	<ng-outlet>
ui-sref	ng-link

ui-router config

```
$stateProvider
  .state({
    url          : '/articles',
    templateUrl : 'components/articles/articles.html',
    controller   : 'ArticlesController',
    controllerAs: 'articles'
});
```

Component Router Conventions

```
|-- components/
    |-- articles/
        |-- articles.js
        |-- articles.html
|-- app.js
```

Component Router configuration.

```
$router.config([
  { path: '/articles', component: 'articles' }
]);
```

Convention

Template

```
./components/<component_name>/<component_name>.html
```

Controller

```
<Component_name>Controller
```

Controller As

```
<component_name>
```

Example 1.4 'component'

Template

```
./components/articles/articles.html
```

Controller

```
ArticlesController
```

Controller As

```
articles
```

Migration strategies

- What if my app doesn't match this convention?

This is my app structure

```
|-- controllers/
    |-- articles.js
|-- views/
    |-- articles.html
|-- app.js
```

Custom Controller Name

```
$componentMapper
  .setCtrlNameMapping(function(name) {
    return name + 'Ctrl';
  });
```

Custom Template Location

```
$componentMapper
  .setTemplateMapping(function(name) {
    return './views/' + name + '.html';
  });
```

Controller configures where to find its view

Module

```
ArticlesCtrl.$templateUrl = 'views/articles.html';
function ArticlesCtrl {}
```

Component Mapper Config

```
$componentMapper
  .setTemplateMapping(function(comp) {
    var ctrl = $componentMapper.controller(comp);
    var template = $controllerIntrospector(name, '$templateUrl');

    return template;
  });
```

Upgrade options

- Move/rename the files to match conventions
- Change the conventions globally
- Use $controllerIntrospector to have the controller define where its view is.

Use in 2.0

http://goo.gl/DFSq5o

Components in 2.0

```
@Component({
  selector: 'articles'
})
```

```
@View({
  inline: "<div>Look! I'm at {{ confName }}!</div>"
})

class ArticlesComponent {}
```

Angular 1 with $controllerIntrospector

```
ArticlesCtrl.$templateUrl = 'views/articles.html';
function ArticlesCtrl {}
```

Angular 2 Component

```
@View({ templateUrl: 'views/articles.html' })
class ArticlesComp {}
```

UI router -> Component Router

- Add new dependencies to our module def
- Set routing configuration
- Change the directives we use in our template

Angular 1.5 Component

```
angular
  .module(...)
  .component('articles', {
    selector: 'articles',
    controller: 'ArticlesController',
    templateUrl: 'views/articles.html'
  });
```

Using the Router now

Basic Configuration

- <base href>
  - Added as an HTML element below <head>
    - Tells the router how to compose navigation URLs.
- Optional service == not part of Angular core
  - part of @angular/router

Basic Configuration

- Configure using @RouteConfig decorator
  - apply to the router's host component.

```
@Component({ ... })
@Routes([
  {path: '/crisis-center', component: CrisisListComponent},
  {path: '/heroes',        component: HeroListComponent},
  {path: '/hero/:id',      component: HeroDetailComponent}
])
export class AppComponent  implements OnInit {
  constructor(private router: Router) {}

  ngOnInit() {
    this.router.navigate(['/crisis-center']);
  }
}
```

```
}
  }
```

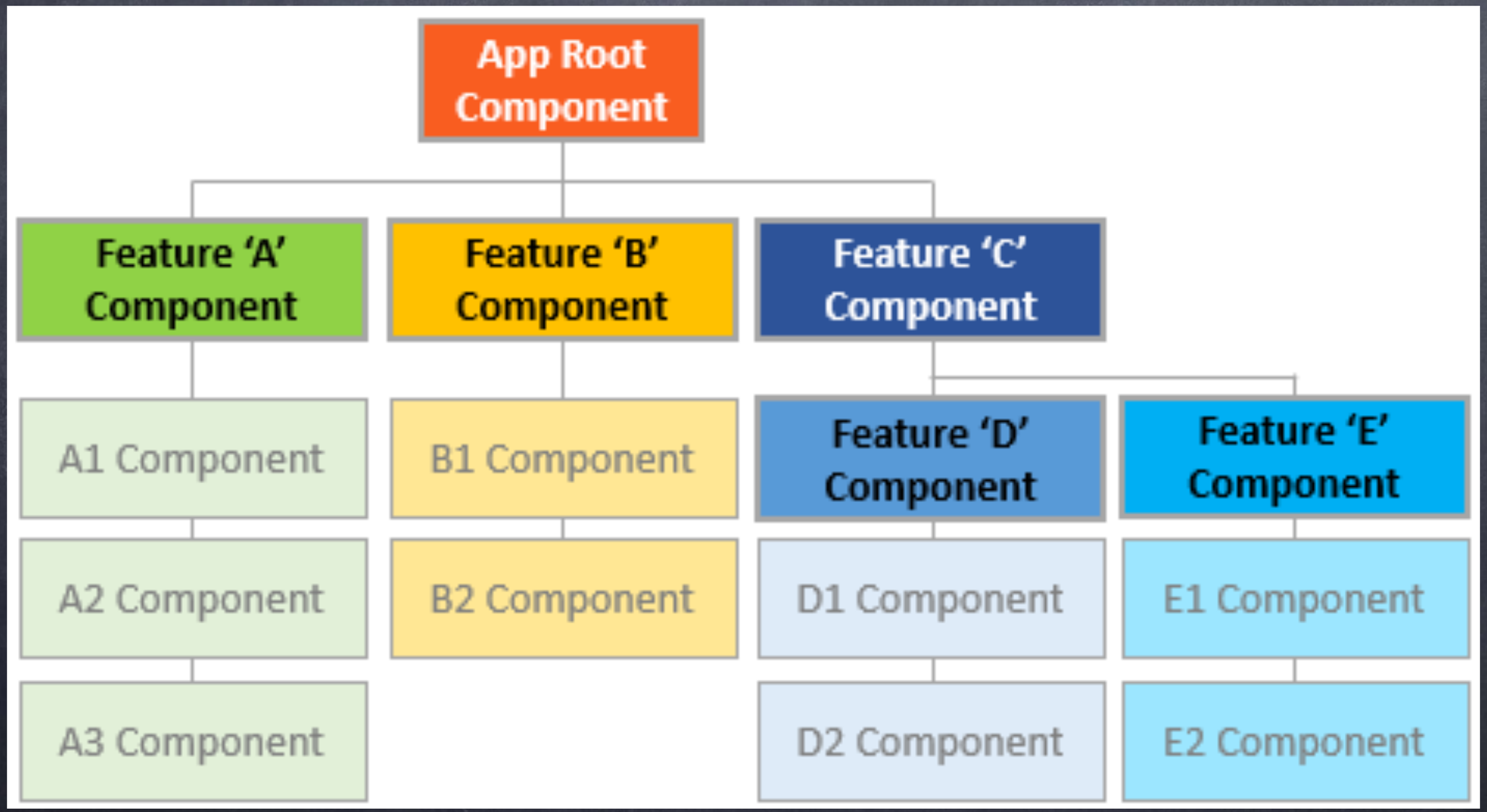
Route or Query params?

- There is no hard-and-fast rule.
  - Prefer a route parameter when:
    - the value is required.
    - the value is necessary to distinguish one route path from another.
- Prefer a query parameter when
  - the value is optional.
  - the value is complex and/or multi-variate.

Basic Use

- Set up a <router-outlet></router-outlet> in your main component's template to receive the route-specific content.
- Set up some basic routes using the Routes decorator.
- Navigate either in HTML or JS
  - <a [routerLink]="['/heroes']">Heroes
  - onSelect(hero: Hero) {this.router.navigate(['/hero', hero.id]);}

More advance structure



Router Terminology

Router Part	Meaning
`Router`	Displays the application component for the active URL. Manages navigation from one component to the next.
`@RouteConfig`	Configures a router with `RouteDefinitions`, each mapping a URL path to a component.
`RouteDefinition`	Defines how the router should navigate to a component based on a URL pattern.
`Route`	The most common form of `RouteDefinition` consisting of a path, a route name, and a component type.
`RouterOutlet`	The directive (`<router-outlet>`) that marks where the router should display a view.
`RouterLink`	The directive for binding a clickable HTML element to a route. Clicking an anchor tag with a `routerLink` directive that is bound to a *Link Parameters Array* triggers a navigation.
Link Parameters Array	An array that the router inteprets into a routing instruction. We can bind a `RouterLink` to that array or pass the array as an argument to the `Router.navigate` method.
Routing Component	An Angular component with an attached router.

Dependency Injection

What is DI?

- Look at DI as a design pattern
- Look at DI as a framework

As a design pattern

```
function CoffeeMaker() {
  var grinder = new Grinder();
  var pump = Pump.getInstance();
  var heater = app.get('Heater');

  this.brew = function() {
    grinder.grind();
    heater.on();
    pump.pump();
  };
}
```

As a design pattern

```
function CoffeeMaker(grinder, pump, heater) {
  this.brew = function() {
    grinder.grind();
    heater.on();
    pump.pump();
  };
}
```

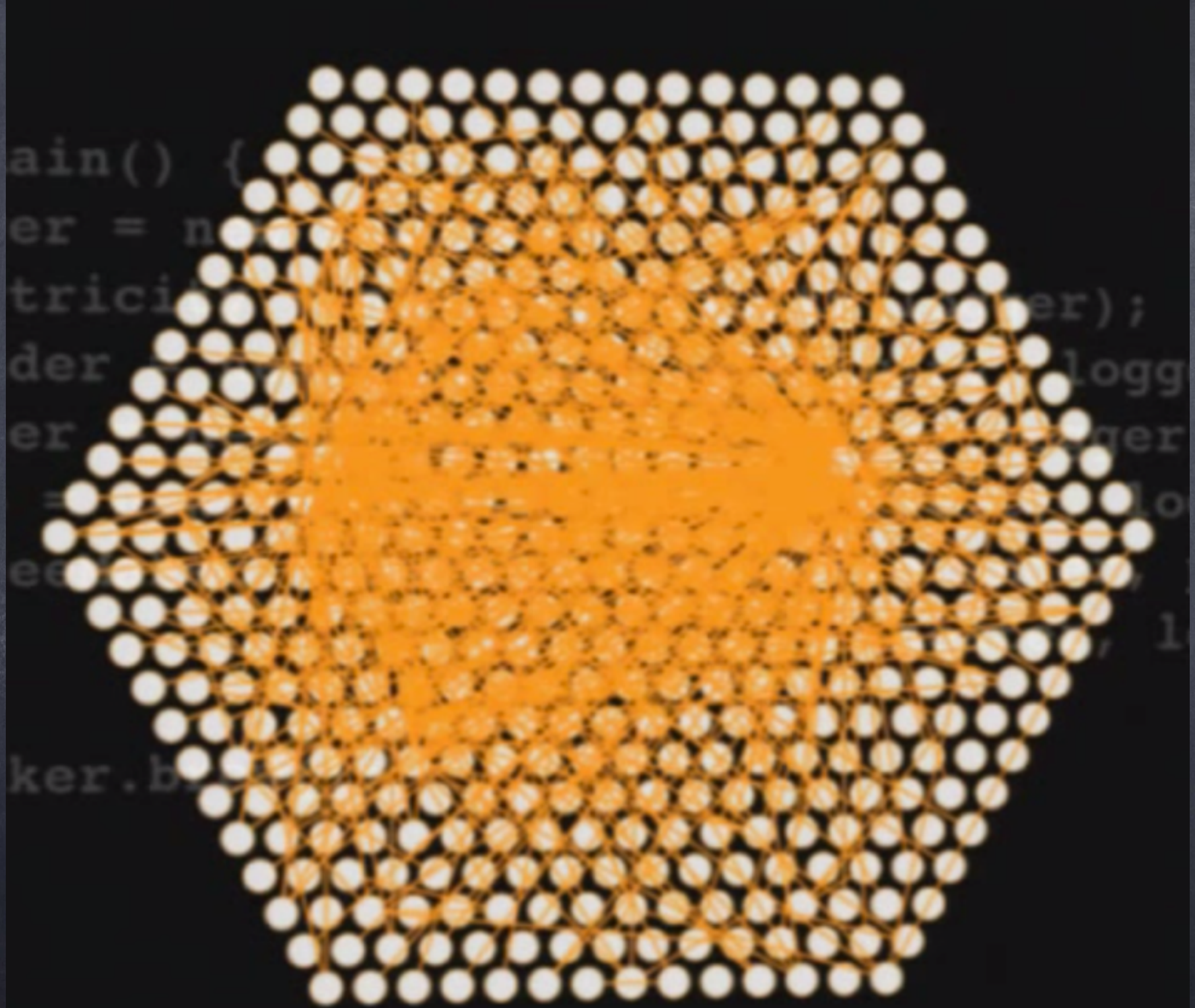
As a design pattern

```
function main() {
  var electricity = new Electricity();
  var grinder = new Grinder(electricity);
  var heater = new Heater(electricity);
  var pump = new Pump(heater, electricity);
  var coffeeMaker = new CoffeeMaker(grinder, pump,
                                    heater);

  coffeeMaker.brew();
}
```

As a design pattern

```
function main() {
  var logger = new Logger();
  var electricity = new Electricity(logger);
  var grinder = new Grinder(electricity, logger);
  var heater = new Heater(electricity, logger);
  var pump = new Pump(heater, electricity, logger);
  var coffeeMaker = new CoffeeMaker(grinder, pump,
                                    heater, logger);

  coffeeMaker.brew();
}
```

Dependency map of e.g.



So, what's wrong with 1.X's DI?

- Uses an internal cache
  - Dependencies are served as singletons.
- Requests for dependencies are synchronous
- Limited Namespace: all keys are strings
- Integrated in framework

DI in V2

```
import {Inject} from 'di/annotations';

import {Grinder} from './grinder';
import {Pump} from './pump';
import {Heater} from './heater';

@Inject(Grinder, Pump, Heater)
export class CoffeeMaker {
  constructor(grinder, pump, heater) {
    // ...
  }

  brew() {
    console.log('Brewing a coffee...');
  }
}
```

DI in V2

```
import {Injector} from 'di/injector';
import {CoffeeMaker} from './coffee_maker';

function main() {
  var injector = new Injector();
  var coffeeMaker = injector.get(CoffeeMaker);

  coffeeMaker.brew();
}
```

No need for angular.module

- We used angular.module to group 'stuff' to inject as dependencies into other modules.
- New framework can inject anything that is a component. Period.
- If you want coarser granularity, create a module that declares dependencies (through injection) on other modules and inject that.

What about mocks?

```
import {Provide} from 'di/annotations';
import {Heater} from './coffee_maker/heater';

@Provide(Heater)
class MockHeater {
  on() {
    console.log('Doing nothing...');
  }
}
```

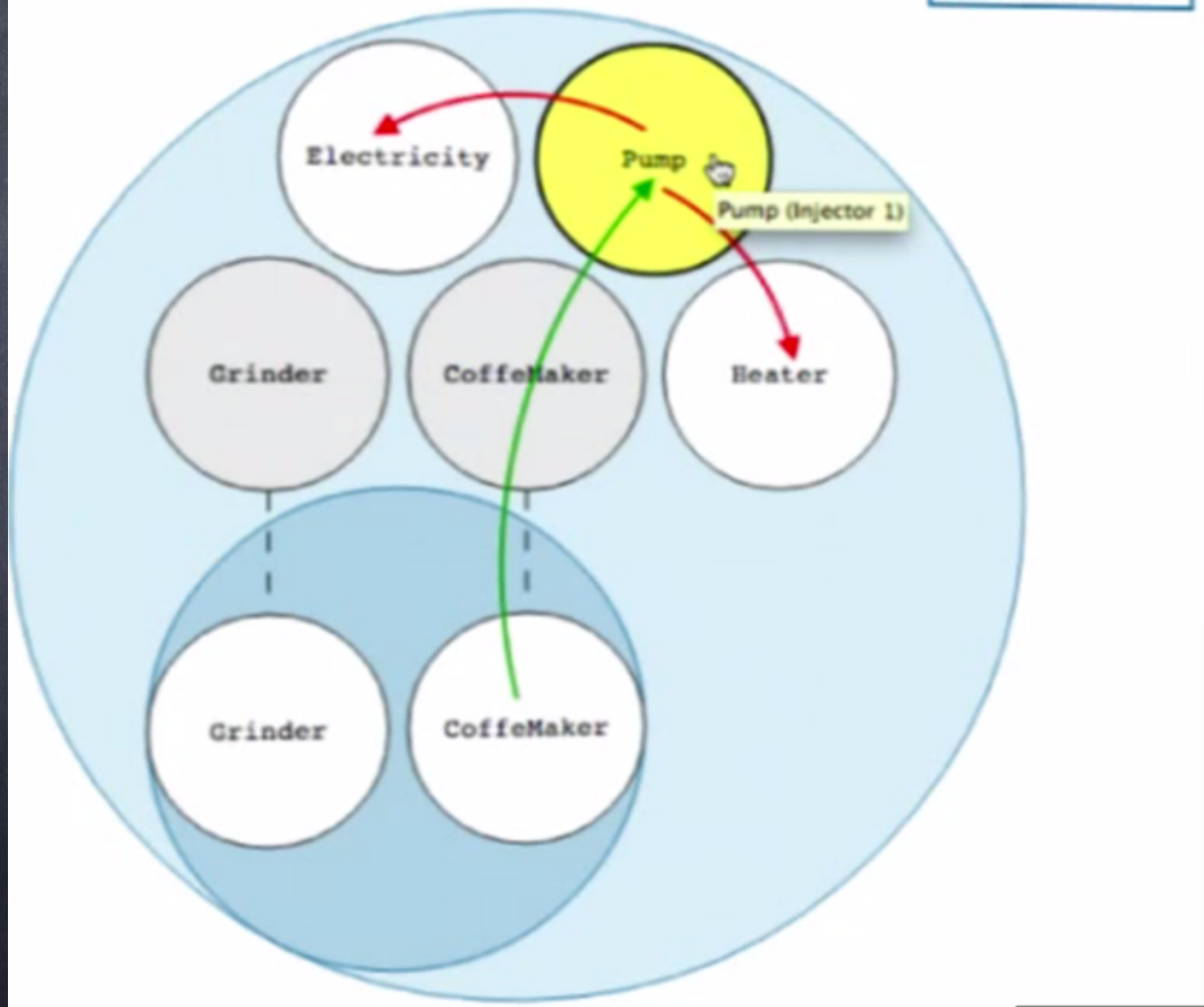
'Main' function for mocks

```
import {Injector} from 'di/injector';
import {CoffeeMaker} from './coffee_maker';
import {MockHeater} from './mocks/heater';

function main() {
  var injector = new Injector([MockHeater]);
  var coffeeMaker = injector.get(CoffeeMaker);

  coffeeMaker.brew();
}
```

And … we can nest them!

localhost:8000/visualize/injectors/

CoffeMaker
Heater
Pump
Electricity
Grinder



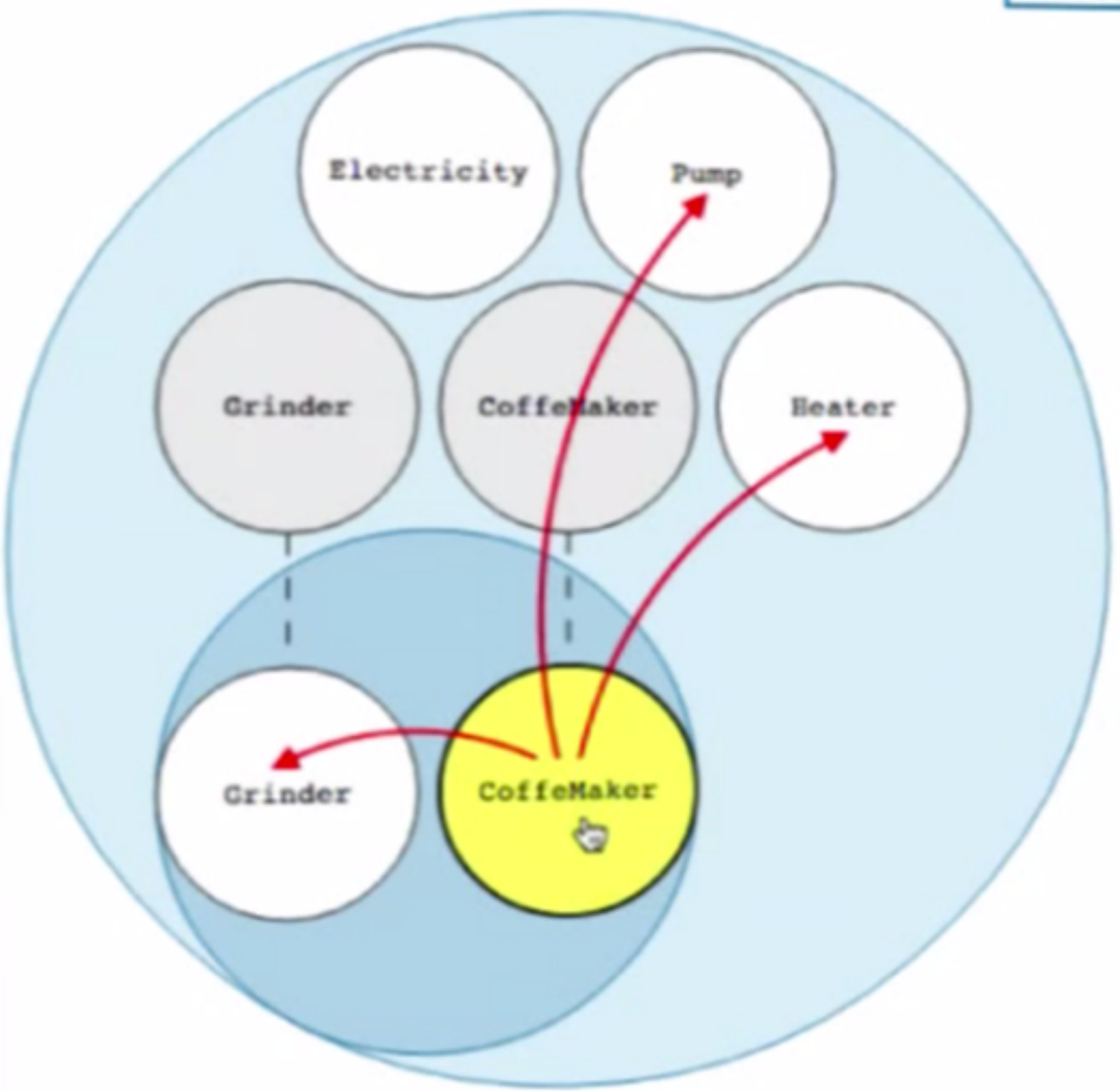
localhost:8000/visualization/injectors/

Visualization of injectors

Thu 4:55 PM

localhost:8000/visualization/injectors/

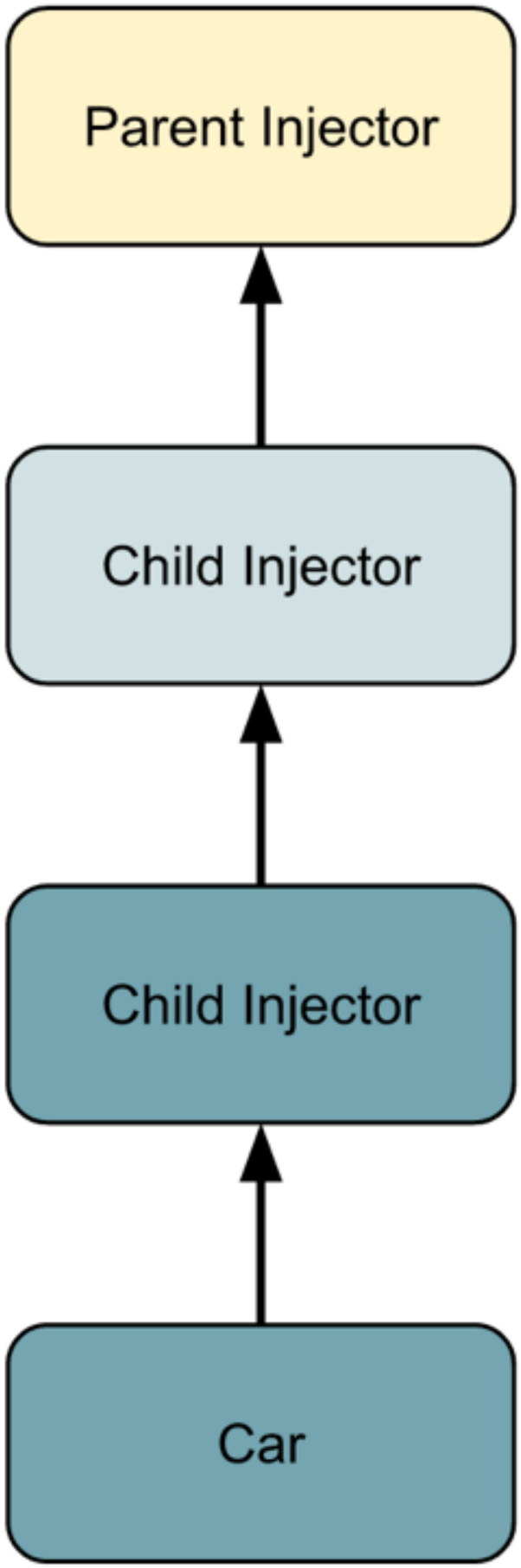
CoffeMaker
Heater
Pump
Electricity
Grinder



<top frame>

▶ Object {name: "Coffe
dependencies: Array[
instance: true, type
ider", title: "Coffe
Injector 2)"…}
app.js:501

>



Injector.resolveAndCreate([Car, Engine, Tires, Doors]);

Injector.resolveAndCreateChild([Car, Engine]);

Injector.resolveAndCreateChild([Car]);

Engine

Tires

Doors

This allows memory management by route

- As you change routes, the router can work with the component injectors as well as its own (each gets one)
- You can load/unload/reuse injected components depending on how you define things.
- No longer needed components can be unloaded from memory!

Questions?

Materials Sources and References/Thanks!

Component Router:

Angular 2 Weekly meeting notes: https://goo.gl/JKeMe5
Rob Eisenberg: New Router Talk @ ng-europe: https://goo.gl/zGatYQ
Chris Sevilleja's ng-vegas presentation on the new router: https://goo.gl/Ua9aJJ
Good blog entry on the new router: http://goo.gl/dd8922
Good slide deck on the router: http://goo.gl/zZcVRq

Dependency Injection

Vojta Jina's talk on DI in 1.X: http://goo.gl/KLlzNO
Great article on DI in 2.0: http://goo.gl/9Ca02H