Questions and Answers from Lukas Ruebbelke's "AngularJS In-Depth" Course

What is the best way to load large amounts of data with infinite scrolling? Would you recommend ngInfiniteScroll or is there another way of doing it?

I definitely recommend ngInfiniteScroll - Brandon Tilley is an excellent engineer, and I know he are using it in production. You can also accomplish a similar effect by creating a collection that is a subset of the main collection and populate the subset collection by slicing from the main collection.

Is there a plan to support sub-views in the future and what way would you suggest is the best way to create a sub-view structure at this point in time with angular?

I am not aware of a plan to support sub-views but that problem is solved easily enough with ui-router. This library is incredibly effective, and is one of the reasons that ngRoute is no longer part of the core. You now have the option to use one without the other without overlapping.

If you have have multiple REST sources that need to be compiled into one view, (example: for each city from source1, get current weather from source2), what is the best way to do this? In a factory, call source2 on success of the source1 promise?

You are correct in that you want to call source2 on the success of source1 from a service.

What is the best way to use $resource; should it be loaded directly by a controller, use with $q or cached in a factory's model?

Controllers should not own the domain model and so a service should store the server results and pass them to the controller as needed.

Should I inject my service directly into directive or I should work with service from directive through Controller?

Inject your service into the directive and it will be available to not only the controller but the link function as well. This is handy when you want to use the $window service for instance.

How to work with custom classes in Angular?

I am not sure what you mean precisely by "custom classes" so I am going to guess that you mean custom JavaScript code that has already been written that you want to interface with. In most cases I would expose it via a service so that you can control how it is being accessed in the application. An example of this would be how I handle socket when working with socket.io. It technically exists on global scope but I prefer to wrap it in a service so that I can mock it out and test it later.

With that said, AngularJS is just JavaScript and so technically you do not have to do anything to work with custom code at all. I use underscore all of the time and because it is such a general utility library, I will just access it directly.

What is considered best practice for navigation code in angular when using tabs and menus? Are there menu/nav libraries or do you manually use routes?

Best practice is to use anchor tags where possible so that you are not introducing unpredictable behavior to your user ie open in new tab etc.

My route table is completely independent of the UI that I use. I am a huge fan of Bootstrap and so I will often times define my routes first and then set up my Bootstrap navigation to match the routes. Routes drive the navigation and whatever UI library you want to put on top of that is secondary.

Any shortcuts to implementing pagination for large result sets?

ngInfiniteScroll is a great solution.

Is there a performance difference between $emit and $broadcast (if I already have $scope should I just do $scope.$emit('event') or should I also include $rootScope and do $rootScope.$broadcast('event') assuming the event is attached to $rootScope)?

I prefer to use $rootScope as my event bus because when I do $broadcast I know that it will reach every $scope object below it since it is at the top. I would use $emit if I knew that I only needed to send an event up a few levels to a parent scope but I believe the performance difference would be imperceptible.

What is the proper 'angular' way to achieve the goals of nested routing without using the angular-ui router?

You would accomplish this by creating a custom state service to watch $location and manually control the views by communicating the necessary information to the appropriate controllers. This is essentially what ui-router is doing so I would default to that but if you could do this manually as well.

Is $timeout(fn, 0); the right way to write directive code that requires css values from the DOM the directive generates (think of a grid that automatically set columns width based on content in the columns)?

I have used $timeout to create a buffer long enough for me to do DOM manipulation so I believe this approach is valid. I am really careful about using timers to sequence events because it is based on assumptions and could fall apart if those assumptions end up being wrong.

When should someone use a factory vs use a service?

Service will provide you a new instance of the function you pass into the service while factory will just provide the value that is returned by the function you passed in.

If you are declaring your function as a class ie using this then I would use a service.

Real Talk I RARELY use classical paradigms when writing JavaScript as I think functional paradigms work MUCH better. That means I see my code as simply data structures and functions to pass those data structures through. As a result, I hardly EVER touch the prototype chain or use this. I am a huge fan of the revealing module pattern which works very well with module.factory and 99% of the time that is what I prefer.

For more clarification see http://stackoverflow.com/questions/15666048/angular-js-service-vs-provider-vs-factory

What do you use for building your apps? Grunt.js? Yeoman?

Yes. :D Definitely Grunt.js but now that Yeoman is growing up I use them together.

It seems that having things like "ng-click" in the HTML would present problems in on-going maintenance of those applications. How do you address the maintenance of many copies of a single angular app?

I am not sure I understand the question. It is certainly less maintenance then having to manually define your own event handlers every time. And every ng-click should serve a specific purpose and so I would imagine the only maintenance you would need to do is remove them as they become obsolete. Feel free to clarify if I have misunderstood the question.

Is there a way to access data on the $rootScope (for example, a session object) inside the template of a directive that uses isolate scoping?

Sure. Just inject $rootScope into the directive.

"I am trying to run the example https://github.com/simpulton/noterious.
I git clone it, npm install, bower install, grunt server.
Everything works, except when I try to run the code. Do i need to point it to my version of firebase?"

I should probably make that requirement since firebase instances are free. I will pull it down and make sure everything is in order. Keep your eye on the repo for updates.

What happens to the template? Specifically what does Angular do with the bindings after it fills them in?

The bindings continue to operate behind the scenes to make sure that the view and $scope are in sync.

When and how does HTML pre-processing (the compile phase) of Angular happen?

The compile phase is kicked off after the $injector has been configured. The $compile service runs through the DOM and generates templates to be linked with $rootScope. The end result is dynamic DOM that is bound to $scope.

This is a really good explanation here http://docs.angularjs.org/guide/concepts#startup.

How does $scope relate to the HTTP Session object used in JSPs? What happens with the session times-out?

As far as I know there is no relationship between $scope and an HTTP session object. I would set up a service to keep track of that but ultimately defer to the server to control that.

Where is the init code of module should reside ? For example I have the panel where I drop the files and I need to prevent the drop event on other places, where should I put stopPropagation?

I would put the stopPropagation code in the directive.

Here is a good example http://stackoverflow.com/questions/10931315/how-to-preventdefault-on-anchor-tags-in-angularjs

"what is the best way to use bootsrap 3.0 with angular?

I use Bootstrap for layout only and so Bootstrap and AngularJS generally just coexist peacefully without having to know about each other. So I believe the best answer to that is to use Bootstrap in the best possible way Bootstrap was meant to be used.

is there a lib? or is it better to write down the html ? but what is with the bootstrap features like tabs modals or carousels? is there a workaround?"

For the JavaScript component parts of Bootstrap, I generally use the excellent AngularStrap or AngularUI libraries.

You can find them here
http://mgcrea.github.io/angular-strap/
http://angular-ui.github.io/

As a general principle, I recommend not reinventing the wheel if someone has already created a library that works.

How do we feel about Angular to build mobile apps? Any pointers from lessons learned regarding mobile?

This is a very good question(s). If there are distinct and unique functionality within a large deployment, I would break them out into their own modules with common functionality being grouped into a common module for use between them all. I have seen this effectively done on multiple occasions.

AngularJS is great for mobile apps and works very well with PhoneGap. The largest thing to focus on with mobile is not necessarily the AngularJS part but the user experience part of it. With that said, be cognizant of the objects and collections you are binding to. For the most part everything should work pretty well but occasionally you have to massage how you are updating your model ie replace the items in an collection one by one instead of en masse.

I would like to know if Lucas think Angular is a good choice for game that use heavily drag and drop that is many layers of drop?

Game development is an entirely different beast and I think that is best done in pure JavaScript and as close to the metal as possible. I would recommend keeping the game engine portion in JavaScript but the rest of the UI ie gameboards, login, etc would do very well in AngularJS.

How much modules the app can have?

Infinite.

With binding methods, that only works for methods on the scope, right? If I have a POJO on the scope called user with a method isActive(), I would not be able to do ng-show-:user.isActive() cause angular would not know when isActive() changes, right?

If you have a POJO then you need to attach it to $scope so that AngularJS can bind to it. In 1.2 you can bind to a function object using the ‘as’ syntax.

question about firebase and the https://github.com/simpulton/noterious

i can not make the code work after npm install bower install grunt server

I changed noterious for my firebase name and it does not work?"

I will look into this and update the repo.

Is there anything available for layout in Angular similar to ExtJs flex? (flex uses grid totals to get size so for 3 columns of flex1 flex2 flex2 1+2+2=5 5 becomes the base so flex 1 is 1/5 of total size flex2 is 2/5 this extends out as you add more)

Bootstrap has an excellent grid system as well as Foundation. I feel like ExtJS tries to solve problems in JavaScript that are best solved in the HTML/CSS space.

"$compile:
tpl = '<div>{{stuff}}</div>'
$stuff = angular.element(tpl)
A.
$compile($stuff)(scope)
angular.element('body').html($stuff)
B.
angular.element(""body').html($stuff)
$compile($stuff)(scope)

A or B better? Does it even matter?"

element.html(tpl).show(); // Add it to the DOM
$compile(element.contents())($scope); // Bind it to $scope

The format is essentially $compile(HTML)($scope);

If $scope is always there for the 2-way binding, why does it have to be passed in?

$scope is empty until you set up your model and methods on it which is why it has to be passed in.

Is $scope part of HttpSession or the same thing?

They are totally unrelated.

When/how does the pre-processing of the HTML happen?

Pre-processing of the HTML happens during the compile cycle. This can happen during the initial load of the AngularJS application or when an AngularJS directive is being instantiated.

How much modules the app can have?

An AngularJS can technically have infinite modules.

Is there a difference between $scope and $Scope? I've seen him write it both ways

It depends on whether or not $scope is being provided via dependency injection but I always use $scope or scope if I am in a directive's link function.

what about a modal pop up - do you not need to declare the controller you are using for the modal from within the controller you are in?

This would depend entirely on how the modal is implemented. It is possible for a modal to simple use the same $scope object that the controller is using that spun up the modal. I have also created a modal service where you passed in the $scope you wanted to use and the service created a child $scope object based on that. When the modal was closed, the child $scope object was simply destroyed.

Did everyone understand linker?

The link function is the place within an AngularJS application to do DOM manipulation. One of the most predominant themes in AngularJS is separating declarative markup and imperative logic from each other for testing. When you manipulate the DOM from within JavaScript you are forcing your imperative logic to know about your declarative markup. Separation of the two can for the most part be accomplished entirely with two way databinding but occasionally you need to do DOM manipulation. This is where the link function comes in.

The link function is essentially the workspace where you are allowed to manipulate DOM. AngularJS makes this easy for you by providing everything you need to effectively do this. You are provided with the scope object set up by the controller or parent controller, a jQuery object that is the element your directive was defined on so there is no need to do query the DOM to find it and an array of attributes to get pertinent values.

What are some of the greatest performance dangers? Too many watches?

The greatest performance danger is is binding to too many objects and/or binding to objects that are too deep. $watch does a deep comparison and so binding to complex objects is going to slow down the $watch cycle.

What's a good place to learn more about promises?

egghead.io has a pretty good screencast on promises as well the documentation on the $q service http://docs.angularjs.org/api/ng.$q is fairly helpful.

Here is a list of resources that the students shared with each other throughout class:

http://www.google.com/trends/explore?q=angularjs%2C+emberjs%2C+backbonejs%2C+ruby+on+rails%2C+nodejs#q=angularjs%2B%22angular%20js%22%2C%20%20emberjs%2B%22ember%20js%22%2C%20%20backbonejs%2C%20%20knockoutjs&cmpt=q

https://frontendmasters.com/workshops/web-workflows-tooling/

http://stackoverflow.com/questions/14324451/angular-service-vs-angular-factory

http://onehungrymind.com/

http://iffycan.blogspot.com/2013/05/angular-service-or-factory.html

http://joelhooks.com/blog/2013/08/18/configuring-dependency-injection-in-angularjs/

http://joelhooks.com/blog/2013/07/22/the-basics-of-using-ui-router-with-angularjs/

https://github.com/angular-ui/ui-router

http://jan.varwig.org/archive/how-to-do-nested-views-in-angularjs-hint-dont

http://jan.varwig.org/archive/angularjs-views-vs-directives

https://github.com/simpulton

http://sanjaal.com/java/tag/why-tdd-is-bad/

http://bennadel.com

http://www.yearofmoo.com/2013/08/remastered-animation-in-angularjs-1-2.html