Angular Boot Camp transcript

This is a very rough transcript of the Angular Boot Camp (angularbootcamp.com) AngularJS training offered as a public course, and on-site by Oasis Digital (oasisdigital.com). This transcript is from a class in early 2014; we revise the content frequently, so a class scheduled when you read this, will have updated content. The main speaker is Kyle Cordes, one of our primary trainers. This transcript is not meant to take the place of the class, which it can't do with visuals and interactivity; rather it is meant to show the depth and breadth of the class.

- -

Day 1

Kyle: Hello, I'm Kyle Cordes. I work at a company called Oasis Digital. Paul is here; he is also from Oasis Digital. We do a lot of development with AngularJS, and we decided a while back that maybe we should try offering some training in AngularJS, since we've done training in the past. We strongly believe that training should best be done by people who **do** things, not by people who only train, if that makes any sense. We are 90 percent people who make things, 10 percent people who tell other things about them. Hopefully that will make this more useful than if we were going through a book saying, "I've never tried this, but you should believe it works".

I already asked a question, I'm trying to figure out so what your guys background. What kind of things do you guys work on? What kind of software, operating systems, platforms, languages? Somebody tell to me something.

[Student listed some technologies]

Kyle: Okay. You make EJBs? Any Spring? Some Spring? Any Hibernate? Some Hibernate. Only Hibernate or do you also make some ORM stuff? iBatis? Okay, even though it's MyBatis now, right? Yeah. Databases, SQL server, Oracle? Oracle and what? MySQL? Be careful, it's the only database I know of which has the feature that if you do something wrong, it says, "Yeah, whatever. That's fine." You just got to be careful. Don't make mistakes.

And then on the website, JSP? Two nods. How much JavaScript? Okay. Wicket's pretty cool actually. It's good stuff. Raise your hands if you've written at least 500 lines of JavaScript in your life? Keep it up if you've written a thousand lines of JavaScript in your life? 5000, you've written at least 5000 lines. 10,000 lines?

But if I do the same thing for Java, you guys will all be way over 10,000, right? Okay, okay. So probably you're familiar with like how JavaScript works pretty broadly, right? Well let's start talking about AngularJS.

Has anybody used AngularJS? Are you using AngularJS yet or just evaluating AngularJS? Has anybody like gone through the tutorial yet?

[Students, various answers]

[00:03:02] Introduction to AngularJS

Kyle: You're mostly right at the beginning. I guess we'll start at the beginning then. AngularJS is a library for making single page web apps. I like to start with this page from the AngularJS website because this phrase right here is really important. **HTML enhanced**. There are many, many cases in AngularJS where you wonder, "Why does it do it in a certain way?" It's because the fundamental approach of AngularJS is to use HTML, not to ignore HTML, but enhance it to make it better. There are many ways that you can enhance HTML with AngularJS.

I said that AngularJS is a tool for making single page web apps. With the kind of web apps you've been making if you're using mostly JSP, each time the user takes some action, you're navigating to another page, right? You're doing an HTTP request. Everybody's familiar with that? When I say request, everybody look over in here, like in the Network tab, everybody's familiar with this? Now you click on something here. You get a whole bunch of HTTP requests to go to the next page.

[00:04:15]

A single page web app is basically one that does not do that. A single page web app is an app where you load up a page - you load all those requests to get a page working, and then most of your interaction happens without reloading the page. I say most because if you get really pedantic here I said single page app, and some people are pretty pedantic that a single page application is literally just a single page, right?

Don't look too hard at the word single. It's very common for people that are making an app to call it a single page app, while it may actually have 5 pages or 3 pages, etc. A lot of times the login is a separate page. A lot of times if you build an application that's 4 major functional areas, each those functional areas is a "single" page. But we still use the word single page web app because the great majority of the work the user does happens on the same page. Just don't get hung up with the exact singleness.

Has anybody here built a single page web app with any tool so far? [only a few]

Back to AngularJS. The other really important thing on this page here is where it says "By Google." The way that we came to AngularJS was from trying, reading about, investigating, building sample apps, and evaluating 4 or 5 different tools in this space: JavaScript Library/Frameworks for building single page web apps. We like this one technically but we also like a couple of others technically. Among the ones we like technically, this one said "By Google" on it. That's important because Google is pouring money into this thing.

When we adopted this over a year ago, I think at the time Google had 6 people working on it, and then some of us were at the AngularJS conference a couple of months ago. Was anybody here at the AngularJS conference? Google was saying there they had16 or 17 people fulltime at Google working on this. Even though it's not a Google product per se (they don't sell it, there's no buy me page) it's very much a strongly Google-supported project. We feel like it's very unlikely it will disappear. We think that it's very likely 3 years from now this will still be around and going strong. That's important to us because we mostly serve outside organizations, and if we make some software for an organization and then a couple of years later they say, "All the tools you picked are obsolete and abandoned," that increases their costs and makes us look bad.

We like to pick tools that have some life to them, and this is possibly even more important inside a larger organization. Larger organizations tend to move people between projects a lot, so you don't want to be stuck in a project forever because you picked something that got abandoned after you picked it. You're the only one who'll ever know it. We think it's pretty safe that way.

- -

The code is all copyright Google, but it's all under the MIT license which is an extremely permissive license. Although it is open source, it doesn't have any little scary licenses that corporate lawyers get all worried about - if they see the word LGPL, sometimes they get worried. Or GPL, they get really worried. The MIT license is free to use whenever you want.

The JS on here is important too. Google is working on a variant called AngularDart which is a rewrite of this in their Dart programming. Anybody heard of Dart? Okay. They seem pretty committed to going forward on both. There was a lot of angst at the conference and challenging questions asked to the Google people. "Since you announced AngularDart, are you just going to make this be Dart next year?" The answer seems to be no. Dart is a project at Google, but there's a whole bunch of people at Google who love JavaScript. They're staying on JavaScript. So it seems pretty safe from that point of view too. (Of course I can't speak for Google!)

[00:08:49]

Personally I'm not convinced yet that Dart's going to take off because even though it has some great technical features to it, it's always going to be a second class citizen if it's only supported natively by Google's browser. And what's the chances of Microsoft deciding to natively support Google's language? These companies don't like each other. My feeling is that Dart is probably going to remain like kind of a sideline thing for quite a while. I have reasonable confidence that it's not going to take over the AngularJS world.

The next question ism where is AngularJS a good fit? Is it only the single page applications? I would say that for the kind of applications that we make for medium to large companies, which often have many, many screens, many forms on those screens, many entry fields, many data displays. Everybody knows what CRUD stands for, right? So applications that are CRUD and then some CRUD and business logic. That's all really good fit for AngularJS.

There are a couple of areas that are not quite great fits. There's kind of a perpetual flow of people asking if it's a good fit to make games with. It's probably not a good fit to make games with. I wouldn't recommend it there.

If you want to make like public web pages, it is possible to use AngularJS effectively like on a public phasing website. But depending on some nuances, you may have to do some extra work to make it properly visible to search engines. Although there's no reason you couldn't use AngularJS on the open web, AngularJS is overwhelmingly used and is an obvious fit for: business applications that live behind a login screen. Hopefully that matches the kind of applications you all work on.

[students agreeing]

That's not to say you couldn't use it as part of a public web property, but it's likely you'd use it for one section of a screen on a public web property. It's unlikely that you'd say we're going to replace our whole web property with an AngularJS app. It's more likely you'd use it selectively. Whereas for internal apps behind a login screen, you often just make the whole thing AngularJS.

AngularJS is predominantly used for large, complex things. I'll illustrate later. It's not necessarily bad for small things either. On many pages where you're using jQuery now, you could use AngularJS instead and it's really not any bigger. AngularJS is about the same size as jQuery. So it's also very suitable to use for little bits of interaction on the page. If you don't mind putting jQuery on the page, you really shouldn't mind putting AngularJS on the page. They're about the same size.

[Student]

A big complex web page, right? You got a whole lot of content a lot of people will put on. But there's kind of one section that you need to do some interactivity in it. You can just use AngularJS for that. If you're already loading 17 big JavaScript libraries, who cares if you load one more, right? We'll see a couple of examples. Go ahead.

[Student]

Kyle: That's the problem I cannot solve.

[Student]

Kyle: [chatter about training room setup] To answer your question, in terms of jQuery as a library from manipulating the DOM, AngularJS is also about manipulating the DOM. In terms of an ecosystem, there are currently far more widgets that are packaged as jQuery libraries that are packaged as AngularJS add-ons or jQuery plugins. But most of the popular jQuery plugins are also wrapped with an AngularJS directive we're going to talk about later. If you just do some trivial thing and jQuery is working really well, you ought to use just jQuery. But generally speaking, AngularJS is if you're building something substantial.

Any other introductory questions? I have a couple of things to point out, but we're about to switch to code. So if anybody has any questions about what is this thing in the general sense? Anybody?

[Student]

Kyle: You will have many opportunities to escape because we repeatedly switch between me talking and everybody coding, and you will be able to escape very easily to go grab food and bring it back. So don't worry. You will not be trapped in here for all day straight.

A couple of more things you should be aware. There's a tutorial that we're going to pick up and use in a minute. There are a ton of great talks about AngularJS on YouTube. Watch the dates when you watch them because sometimes things you watch that are a year old are pretty obsolete. This thing is evolving fast. Just be aware if you watch things: if the date is not really current, if the video presenter is doing something, you think “Is that really how you're supposed to do it?” The answer might be that **was** how you were supposed to do it. So keep an eye on that. But I would definitely recommend just watching these things at random from time to time. Broaden your knowledge greatly.

- -

This is always a good page to have handy, this Built With page. Whenever people say is it okay to use this thing? you can always point to lots of things that are complicated and some of them are well-known that are using it. This makes it feel like a safe choice, so be aware of that.

The documentation is excellent. I will consult the documentation and show bits of it many times.

[00:16:28] AngularJS Tutorial

Kyle: Our next step is we're going to do the AngularJS tutorial, and Paul is actually going to take a stab at stepping you through the AngularJS tutorial. What do we need for that? Just Git and Node. That's it. Everybody needs to get a hold of Git and Node. Does everybody have Git already? Does everybody have Node already?

We get to work together in getting Node up and running. Everybody, this is a great time to escape for a couple of minutes and grab the food while Paul gets plugged in and we will run and help everybody get Git and Node going.

[00:17:23-00:43:23 setting up Node / short break]

We're on the step zero. You need to go into that directory if you're in this command. Should I explain this Git command or is everybody already familiar with Git?

[Student]

Kyle: I'll give it 10 seconds. In Git, Checkout does not mean "make a reservation" like it does in some tools. And it doesn't mean pull the code from another server. What it means is: go to a certain place so all of the repo I have locally, go to a certain point and that point is a tag called step-0, and that -f means "force, throw away any local changes". You probably don't want it all haphazardly running in your real projects but we do this kind of thing often just because as we go through the tutorial, we want to make little ad hoc edits and then resume to the next step.

You do that command, then you need to get a terminal up and I will bring up a Mac terminal. I will adjust this point size a little bit. Kind of need to rely on, I need anybody to tell me if the size is too small.

You need to get into your phonecat directory and then "git checkout -f step-0", which takes us to the first step and then there's a command which you can copy from down here. It's where… I'm going to get the one that shows you the… Here we go.

This is the command if you're in a UNIXy context. Is anybody using Cygwin on their Windows machine? If you're running UNIXy things to do a command like this. If you're running Windows-ey things, you got to put the slash the other way. But roughly speaking this sort of command I'm running to get a little local web server running. Of course we could put this content somewhere over Apache or whatever server but for the purpose of working on a tutorial, it's nice just to be able to serve the code ad hoc out of wherever we are which is just right here in this directory. I run this command and now I have a web server running, localhost: 8000, then I go up and go up here, 8000/app/index.html. I think is the—so nothing here yet. Pretty basic app. This is a starting point.

Let's talk editors. Somebody tell me the editor or IDE they use?

[Student]

Kyle: IntelliJ, good choice. Do you have WebStorm?

[Student]

Kyle: If you like IntelliJ you will very much like WebStorm. It's their web tool complement to their Java tool. Other stuff, Eclipse? I heard another Eclipse. Any other editors anybody likes? Netbeans. Okay, VI. Maybe We'll do this whole thing only in VI.

I usually use a tool called Sublime Text for classes because it's really easy to see what's going on with the files in Sublime Text. We click File. Start. Here we go. This file, this content we see has this source behind it. Like I said tell me if font sizes are wrong. It's very hard to tell when you're right next to the big screen, if you're big enough. Okay so this is pretty simple. Can anybody spot the AngularJS-isms in here? Somebody point out an AngularJS-ism.

[Student]

Kyle: This guy here, okay any other AngularJS-isms you want to point out.

[Student]

Kyle: Where?

[Student]

Kyle: Yup line two, okay. And then there's of course the obvious AngularJS-isms right? The script tag. Those are some important things from AngularJS that can be readily demonstrated right here. The first is AngularJS comes as one JavaScript file that has no dependencies. You note that I didn't have to load jQuery first. I didn't have to load anything sooner or later. This guy here should give you a hint something is going on. This is how you tell AngularJS to come to life. You have to label some point ng-app. And then there's a substitution templating mechanism activated by these handlebar style templates. Does anybody use handlebars? No handlebars? What? Right okay so there's a number of bits about AngularJS that can be easily demonstrated, you can prove them to yourself using this, just this tiny bit of file. One is try changing this in some way. And I put a couple of zzzs at the end. Doesn't really matter what the change is and then look what happens when I make that little change. See that? I like to leave my debug console up while doing this. Oops, not what I meant to do. I've been playing with this new search tool Duck Duck Go. Google's been around too long. Try another thing. I leave the console up when I see these errors. This guy here is an error. It's a warning you get. It's unrelated to the change I made.

Any important thing I will tell you about AngularJS probably 20 times in the next three days: AngularJS, much like HTML, has this policy that if you do extra stuff, it just goes on right past it. So if it doesn't recognize something you did that's not an error or even a warning, so much like HTML, saying some things that it doesn't recognize, no warning, no error and furthermore, not activating AngularJS by not saying ng-app is not a warning and not an error. So the number one common problem you see of people first learning AngularJS is I can't get my page to work, it's because they left off ng-app at the top. So I'll spell it right. I will reload it and then it will work again. But no error, it's not a warning or an error either way. So that will bite you many times in your first month of using AngularJS. Then it will stop biting you. But I control a little more than this.

AngularJS works at the DOM level. Who knows what a DOM is? Only one person didn't raise their hand and I think he's just not paying attention. Imagine that this were something like JSP. So imagine this were PHP or ASP, whatever, means there are little other kinds of brackets. If they were, then I could make a change like this. I could take something like I could maybe take that part and just move it after the P. I could do that. Right? So if this were PHP, that change would have no effect. And everybody see why I say that? I see you nodding. So has it not occurred to anybody why this should have no effect if we wrote in like PHP? You look skeptical. Okay that will have a breaking effect. It will do unexpected ugly things in AngularJS. So what this illustrates is that AngularJS does not walk over the text of the page, right? So if you were using JSP and before the browser ever sees it, there's this template engine that walks through just a little text you type and it does textual substitution.

But AngularJS doesn't do that. AngularJS will never see this HTML. AngularJS works after the HTML so the bare HTML goes into the browser. The browser parches it into DOM which is an end-memory object representation. And then AngularJS steps through the elements of the DOM performing its work. That means that these little substitution things only work if they're sitting inside one DOM element. And right now they're not because like this is one DOM element and then that's a different DOM element. So the curly braces will not span DOM elements. That sounds like a problem. You'll see later why it's not.

But for example, if I'd try to do something in here, some sort of bold in here, AngularJS runs after HTML's been turned into DOM. But the results of AngularJS get fed back through the browser to get turned into DOM again. So you catch all that from there? This is text inside of a spring which is inside of an AngularJS expression which became substituted. AngularJS has this delightful recursive property where the output kind of gets fed back in to the process again. So take that little bit out. Okay.

And a few more things to illustrate at this level. Somebody asked if you have to use AngularJS for the whole page, the answer is no. I could put my ng-app down there on body and it would work. AngularJS works for the part of the page in which you put that ng-app. So I could even do something like put a div in here. I love typing. So I could put a div in there and I could decide that I want AngularJS to work for this portion of the page, and then if I had another one of these out here, and this build doesn't work, it doesn't matter. So we'll see it didn't work out there and it did work down in here. The answer to that question is yes, you can use it for only part of the page if you want.

For any real AngularJS app, if you're building like a big complex application, you're essentially always going to put it up here for reasons that become clear later. But if you were just using a little bit of AngularJS in an existing complex page, you could remove a lot of possible problems by just putting all them in part of the page you care about. The reason why you always put it up there is that you want to be able to do data-binding up in the title.

Let's do something simple. You want to be able to do data-binding at the title, and so you have the seven appear up there. So frankly speaking. if you put ng-app anywhere other than the very, very top of the page, so you put it down here for example, everything else will still work but the title will not have data-binding work. Yeah? Say again?

[Student]

Kyle: Yes.

[Student]

Kyle: This HTML comes from the web server in exactly this form. The browser turns the page into DOM. The browser runs this AngularJS source code in here and then it actually ties in the on-DOM ready event under the hood. So when the on-DOM ready event fires, AngularJS will walk over the DOM and that's when AngularJS activates. So it activates after the HTML comes over the wire, after it gets turned to the DOM, then AngularJS loads. And we'll see the effects of that repeatedly. And that's one of the hardest things to get your head around if you've been working with systems at work like three steps back from that your whole life.

Okay so one more thing I could point out here. A long time ago, that was the sort of recommended place to put this thing but nowadays, it's most broadly recommended to put your script tags at the end of the body. So it works the same way. And the idea here is if you put it up here, the browser, depending on the browser you're running and the version, sometimes will stop processing the page while it loads this JavaScript and you don't want that. You want the browser to be able to process the whole page and then load the JavaScript, not the other order. Any more questions on the tiny, tiny bit of AngularJS we've seen here? Good.

[Student]

Kyle: But if you replace it, use the AngularJS mechanism yes. If you jump into yourself into it, you have to call line code AngularJS. If you do that, the person reviewing your codes should not [00:57:29 inaudible] It should never make into your code. You shouldn't do that. A process that we will repeat many times is to go to the next step. Looks like I… I'm trying to show things nice and big. So I will leave my server running there and then here, I will go run the other. Check out step one. So many times we will do this and I'm at the next one. The other here I'm using will automatically reload the content. There it is. Now there's some new stuff in here. This is a tiny, tiny step. See how they're just all static HTML in here. How's your setup coming along? Did you get that? Okay another step or two, it's all yours. I like to point out that they're showing bad practices again and I'm kind of pedantic so I always do this. But all we're seeing here is that this is this trivial page that shows a bit of data which we're not yet using AngularJS for yet. Then you're going to want to go on to step two.

[Student]

Kyle: Step one it was just a static page. We're advancing to step two and then I always do this just because I'm trying to show the nice way to do things.

Paul, are you ready to explain this one? Okay Paul's machine is cooperating now.

[Student]

Kyle: No because it's a little server made for developments so it disables caching and so you don't have to. It just cooperates nicely.

Paul: [01:00:49 inaudible, probably asking something about step 2]

Kyle: That –f unfortunately is throwing away whatever changes I made. Just go to the next starting point. Going to need a bigger font, much, much bigger font. What tools do you use? Is that WebStorm you're using?

[Student]

Kyle: WebStorm doesn't use that. There's a setting you can turn on the WebStorm preferences to enable scroll size. I think we had to do this last time.

Paul: Go to your Settings, your Preferences. Settings. Then under Appearance Editor, you have to go to Editor. Click on Editor. There we go. Is anybody using WebStorm or IntelliJ tool? You have to turn that on before it lets you scroll, control scrolls or adjust your font size.

[Student]

[01:02:19] Step 2: AngularJS Templates

Paul: Let's do this from top to bottom. The first thing you'll notice is we have directive that's we had before this time. It has a string set value. We also have a new attribute here on the body called ng-controller, also called PhoneListController. You see another attribute down here on this list item. And then you've seen these before where AngularJS uses this for binding to take this expression and fill it out whatever AngularJS parts is developed. So let's go ahead and start at the top. Before we were adjusting ng-app and that whole [01:03:03 inaudible] we're going to start and we're going to look through the rest of the document from here.

Now we're specifying that this is an ng-app called PhoneCatApp. In order for AngularJS to run and use this app, we have to actually declare this app somewhere. See we've added a new source file, controllers.js. And inside the controllers we have a PhoneCatApp which is an AngularJS module. Given that a PhoneCatApp and inside it it has a controller. So we have a whole lot of new things going on here. So we're just going to step through this slowly we'll take a look at what this actually looks like when run.

When this is run, we see a list of three items. They're just a list phones with some [01:04:19 inaudible]. So what's going on? We know what this is. This takes whatever's inside here as they substitute it in. This is a directive, is an AngularJS word you're going to hear pretty often from here on out. These are keywords that AngularJS picks up on when it's browsing the DOM slipping through these ng- as specific AngularJS directives. And it knows once it reaches those, to go ahead and do some processing.

ng-repeat is a special directive that Angular knows to take each object down in this expression in making new copy of this element. What's doing this is looking for an object called phones. That's why we make additional phone for each one. Then here it says could be the name of that phone and the snippet. Now where is it getting these phones from? Right here we have another directive it's called ng-controller. A controller is this specific directive that is used to start up and create a scope for you to place data for that particular element in any element within it. Now we have a flow in this controller which we specified in our app.

Let's take another step back. We've declared this page to be an ng-app called PhoneCatApp. So AngularJS is looking for a module called PhoneCatApp and increase the application from that module. And inside this PhoneCatApp, we've declared a controller called PhoneListController. You can see now these adds up. When AngularJS is run and it parses this yellow script, it adds this PhoneListController to this module and places it in the name space set to be read by the strip and it's parsed.

[Student]

Paul: On my number I can't tell you. But you can just remove the script source part. Does anybody know how to run my numbers [01:07:13 inaudible] into the script tag on line nine. Take that source thing off of there. Rip out some blank lines. Put it on the page. We try to go as far as we can with fitting everything on one screen. It's much easier to have it flip back and forth while understanding.

[01:08:39]

So we just kind of start at the top again I guess. New script is just a way to tell JavaScript to perform nicely in a way that it basically just requires you to use better practices that are available to JavaScript. And it will throw more complaints than if you're not using those. We create this variable called PhoneCatApp. We're sending through AngularJS module named PhoneCatApp. And then we're sending [01:09:17 inaudible] And that we're calling controller on PhoneCatApp to add a controller to this module. Naming that controller, PhoneListController, and we're providing a function. And this function is really what the controller consists of. This is run the moment that

- -

AngularJS realizes that this controller is being used. We see that we've also provided this funny thing called $scope to this controller. Here saying $scope.phones find an array of objects. And before I said that, a controller is a place to kind of set up and establish your data for a particular element on the page. The scope is what's used to house that data. So this is where you will place your models for your application on the scope.

Now down here is where we access this controller. See your PhoneListController, Phone List controller. And now we have access inside this body at run time to this object. Now down here, this directive is asking for phones which is right here. Saying for each phone in that array perform this translation and give me the name and the snippet. Look again at what this looks like to see if it repeats a line item every phone in phones.

[Student]

[Student] Paul that thing on line two, where it says PhoneCatApp, what is that matching? Because I see two like it's confusing because on line 11, there's two things named PhoneCatApp. Can you change one of them to make it not confusing?

Paul: Absolutely. We can. This is the JavaScript out there that is holding our module. This is not what the module is called. This is what the module is called. So here what we're looking for is the name of a module, not the name of a variable that holds the module. So here we can say [01:12:15 inaudible] and then we can make it to match at this so that make…

[Student] Does it still work?

Kyle: Okay you've got a question out here pending, too.

[Student] Could you explain what a module is? And what, in this case, the scope would be? I see that you've received a scope as a parameter on that function you are defining. Explain a little bit more of that.

Kyle: I can talk about that one. JavaScript was invented in 1996 in something like a week and a half by a really, really sharp guy named Brendan Eich, who's actually still involved in the evolution of JavaScript today. He had this challenge, he could get a new language ready to go in browsers in a week and a half or we'd be using something like Java as our language inside the browser. which we had for a while in the form of applets which have kind of fallen by the wayside. He had one shot to get something. He left a lot of some things out. One of the things he left out was the module system. In Java you have packages and so on. It's a very rich module system. JavaScript doesn't have a module system. That means that people have to implement a module system in JavaScript. So have you ever heard of RequireJS? Ever heard of AMD? Ever heard of Browserify which is part of NPM?

[Student]

[01:14:26]

Kyle: There are various add on JS module systems. The painful thing about modules not being in the language is that everybody can invent their own module system. Which means there are multiple competing module systems. This problem will go away in ECMAScript 6, the next major version of JavaScript. Then we'll get a module system in the language and then everyone else will stop using theirs. One of theirs is one built into AngularJS. AngularJS has a module system built into it, it's kind of somewhat analogous to Require, somewhat analogous to AMD. And you activate it by saying angular.module. My app is making a new AngularJS module which is behind-the-scenes just an object. It's a container to put things in just like a module and just like a Java package is a container to put Java classes in. So that is what that line means. You're seeing the first tiny bit of the AngularJS module system with one line of code at this step.

[Student] And this variable scope in this regard here is what? Something a scope that's controlled by Angular?

Kyle: An AngularJS scope is just a JavaScript object. It's a JavaScript object which is published, made visible, to the template. So down there on lines 24, 25, 26, that's an AngularJS template. It can only see things that are published, made visible to us. It doesn't just look at all your JavaScript code. But it's seeing whatever's in the scope. When he said body ng-controller= that's saying make a new scope here. The purpose of a controller in AngularJS is to initialize a scope. That line 12, we're defining the function. We're saying here is a controller named whatever. It will initialize the scope. Well it's just a function it's called and it gets handed the scope it's supposed to initialize. That's just utterly plain JavaScript code on line 13. $scope is just a variable which is a JavaScript object. And it can put whatever in it, it wants. And whatever that is will be available to the template that lives inside that controller. That's a wiring up process that's going on.

[Student] What you call the template, where it starts ng-controller, is that what the template—I'm sorry below ng-repeat? That's part of the…

Kyle: At least you colloquially refer to some section of HTML that has AngularJS stuff in it as a template.

[Student] In that particular piece of HTML there, its only scope would be what you're defining up above in [inaudible]

Kyle: That's all that's published - that is what you just published to it. There's not a bunch of hidden magic but at least what's on the scope is just literally the code right there. Maybe you could explain on line 24 what that Phone [01:17:11 banging sound] means. ng-repeat is an iterator but the wrinkle is that that the thing on the left is a local variable so just give it a new name. Make it obvious.

[Student]

[01:17:39]

Kyle: That's just a local variable. It doesn't appear anywhere in the controller. See, that's an iterator with a local reference. More questions?

Paul: Ask them now. Now is the time to ask questions.

[Student] This scope variable, once the template is finished, well let's see. Can you continue to use any scope to add things to this code and use it in other templates below? So if you have more HTML code

Kyle: Where are you talking about HTML code?

[Student] Like for instance after line 29, after the UL and you can create another template there right?

Kyle: So still inside the body?

[Student] Right.

Kyle: Okay so Paul put up the notes on the scope and then do a little something here to use this. Name or whatever, put that on the scope.

Paul: Down here unless you need to look on the scope or something all day. Save it.

Kyle: Does that answer the question?

[Student] Right. Could you again continue from down below where it says let's say pass hello or say mine is hello, can you call again phones or is that not in the scope?

Kyle: Oh yeah. Paul copy that whole UL, that whole repeat thing. Just do another. Just do another repeat UL there and then turn down the HTML so it fits on the screen a little better.

[Student] This scope is really the pages scope? Is that what I'm…

Kyle: Well we'll see. Go down there and put another one of these after, not inside the UL. Well that's fine, yeah just right there that's fine. You should see if it will move over more than once.

Paul: Yeah so here we're asking for phones, and here we're asking for it again. We're doing the same thing with it but [01:19:56 inaudible] We can ask the same piece of data twice.

Kyle: Did it work?

[Student] Yes it works.

Kyle: So what was the question again?

[Student] So I guess the scope really becomes the pages scope?

Kyle: Where is the scope? What HTML element is the scope for? You can read it right? On line 23.

[Student] Oh the body.

Kyle: So Paul move that to a div inside the body.

- -

Paul: Move the controller on there.

[01:20:30]

Kyle: Yeah put that controller into the div inside the body.

Paul: So now to try and get to –.

Student] You got hello at the bottom.

Kyle: Hello hangs over outside the div so you already got … what happened?

Paul: We look at our console.

Kyle: So some errors, you actually do get errors for. Others, you don't.

Paul: Look up here, variable is getting an error. And the reason being is because at this point, we're not parsing HTML. The HTML's already been parsed. At this point, AngularJS is actually trying to look up a controller named PhoneListCt and not able to hide it. So at this point, there's actually an opportunity to throw an error and it does. We put that back to hide things. The error goes away. And just that before now, actually the repeats are still within the controller. They still have access to the phones object on the scope. Use display. Down here where we had hello name. This is outside of the div. Because of that, it only has access to any scope that could be declared as part of the body with the HTML. So if we don't specify the controller on either of those, there is a scope available to it. So we simply get them to split. Because it is still inside the AngularJS app, it still seems to parse that double curley brace. Rather than throw an error, it just ends up with nothing because that's a perfectly valid answer. Scope.name is nothing because there's no scope. So it just doesn't show anything.

[Student] Double scope you utilize?

Kyle: There is.

Paul: There is. Use it very, very, very . . . .

Kyle: It's like a global variable. So yeah you could just make all your variables global if you wanted.

[Student] And so presumably, you think we have multiple controllers that have different scopes for different pieces of . . . .

[Student] Can you briefly touch on why this is cleaner and easier as opposed to the brute force that you'd have to do with jQuery to do this, something this simple?

Kyle: It's not obvious here. It will be obvious in a couple of steps we'll be hearing some of you say wow that would be a pain to do in jQuery. Actually here's one if you want to illustrate it really easily. So Paul go up there on line 18. And type barx=$scope. That's it. Get a semicolon, there you go. You guys like automatic semicolon insertion? You rely on that or you type them in? I think it depends on what language you use. If you came from Java or C, you can't resist hitting that semicolon right? If you came from Python or something, you're like I don't type semicolons. Okay so now he's got something, he's going to have access to that bar. So now if you go to a different page to bring up the console. So if you type x.name= and just come on, give me a name. You can do it with us and type your controller for this demo to work. Move that little name thing up inside your controller. Yeah so then just do x.name and change that name to something else. Did that work? And then x.$apply. Here a colon.

[01:25:17] WORKSHOP

Kyle: Yeah that's fine. You actually have two var Y over the top. And you have said y= over here. I'm trying to put a hook in so we can have access in at the console for something that's defined inside. And take off the bar. This is a little bit of hackery that you often use when you want to play with something experimentally in JavaScript. You've probably done similar things with your jQuery stuff. Okay do the change. So that's the first hint of an answer of why this is different than jQuery because we're just doing data binding. He didn't have to write code like go grab something according to an ID or according to a class and change up text. It a transparent data binding process. Other questions about what we have on the screen so far? Let's go ahead and move on to the next. Step three. Google for some reason didn't take the time to use Bootstrap CSS scaffolding framework problems. I'm just going to change this real quick. Are you guys familiar with the Bootstrap?

[Student]

Paul: Okay so here they're using the scaffolding.

Kyle: Here you go four and eight or five and seven. One of those is usually good enough. And you still have to make your browser wider.

Paul: Let's go way back there. So now we have our list again and we have a search box, and we type in the search box. Put it here. Start by moving this over here. Okay. So we added some CSS. Nothing special about that. Our controller is the same as it was before. We have our module called PhoneCat. We have our controller on that module, our PhoneListController. We have our phones object that was placed on the scope. Down here we're declaring the controller on the body. So that means we're moving this down here. We've added a search. We created an input element and put the ng-model directive on it with a name of query. Let's see what that does in a second. We've got our repeater just like we had before but now there's a little bit extra in the AngularJS instruction. So this thing you probably wouldn't notice is that this and this are the same and the ng-model, this purpose is to type an input to an object on this code. So as we type into the search input, this actually taking this value, assigning it to query and saving that on this code.

What you're asking earlier about why we want to do this on the setup is because this value is automatically retrieved as it's updated and made available to this instruction right here. So this instruction, if you work in UNIX which I think we already have some doing that, you'll recognize this symbol as a type character and it takes a value and types it into a map. This is actually called filter.

Now we've got a little bit of confusion here because this is called filter. Why is the word filter here? Well, filter just happens to be the name of a filter. It comes with a set of filters that are readily available. So the purpose here is to take phones, filter it, using a filter called filter and supply the value of query to that filter made.

Kyle: Regard it in the more traditional invocation.

[01:33:03]

Paul: So the thing I like about is we have our phones list that we have up here and we want to filter it based on the word we type it that it just takes phones that fit that description and it supplies that to over here.

[Student]

Kyle: So you can see what's in there. You can read the JavaScript right there.

Paul: So we can see that we have both the name and the snippet. Over here you see the name snippet. So if you want to look for the word "fast," you found that in the snippet. We want to look for the word "Xoom," that's only available in the name. So this filter must be working on the entire text code of the object of the values.

Kyle: What if it was a field that wasn't even data bound? What would happen?

Paul: You just add a field. We're not displaying it but it's still part of the object. It located the Motorola Xoom without Wi-Fi because in the description properties of that object.

[Student]

Paul: So [01:35:07 inaudible] we have an input that is bound to a value called query that is automatically updated as we type.

[Student]

Kyle: So put our data binding underneath that using the curly brace and text, they've already seen this, to show what the query currently is. Maybe label it or something. Depending on your screen formatting, you might have to p it or br it to…

[Student]

Kyle: Curly. Can you make your font bigger? It's kind of fun being back here. I'm not usually back here. Things that are not obvious when you're standing up there are more obvious back here.

[Student]

Kyle: Double curlies.

[Student]

Kyle: Handlebars popularized that. So, Paul, something I found I have to do to make this process work is just chop out those comment, brutally chop out comment lines and blank lines. Blank lines are great. They make your code readable. But they're really crappy when screen space is in short supply. So I end up just chainsaw them out. Make an editor shortcut that just wipes them all out.

Paul: We can see that the two-way binding we're directing towards right here. By declaring ng-model directed, that tells AngularJS that this element is strictly bound to this object on this code. You'll notice that we didn't actually create that object in the controller. [01:37:15 inaudible] for you as needed.

Kyle: So put another copy of that input model, the whole line 30. How does that look?

Paul: These are all tied to that same object on this code.

[01:37:49]

[Student]

Kyle: Go try it.

Paul: So you're saying go ahead and put it right in here.

[Student]

Paul: Any data that you want to be available as a controller, module, that will be on this code.

[Student]

Kyle: What's not necessary? Well, you saw it work before. He's just trying to illustrate that it's okay whether or not you pre-populate it. You're not required to. It's not an error if you do it. It's okay if you do.

Paul: We did automatically filter this before you put that in. Any questions on this? Any other questions?

Kyle: Should we help and try to catch up on their own edits at this point?

Paul: Yeah. So go ahead and check out this example and go ahead and play with these values, adding some things to this code, adding things to the object.

[01:39:05] WORKSHOP

Kyle: There's no whiteboard in here. I just realized that. Wow. How do you ever train without a whiteboard? Oh, okay. 2 foot wide whiteboard. Little whiteboards for having little thoughts. To think big, you need a big whiteboard.

Paul: [01:40:56 inaudible] at this point?

- -

Kyle: Yeah. If there's a way to make it… I've always had a hard time getting it big enough, but we have a good, high-res screen here. So yeah, if you can make that take up most of the screen, then it's a really good illustration because it's not readable, the current size.

[01:41:41 Trying to fix the display on the screen]

Kyle: Is that filter case sensitive? Do you have it up and running? Well, you can try it, right? Try it.

[Student]

Kyle: Well, there you go. There's your answer.

[Student]

Kyle: It's a little confusing here because you're using a filter called filter. AngularJS has a concept of a filter and there are 6 or 8 or 10 of them built in. One of them is named filter. So the filter filter which is built in is case-insensitive and it looks at all of the properties directly on the object. But you can easily write your own filter. You wouldn't name it filter. You'd name it something more reasonable for your need. But it's arbitrary JavaScript. You can do anything you want. You can make it case-sensitive. You can make it look at some properties and not others. You can make it do numeric matches like the ranges. It's all JavaScript at your fingers ready to use. Just this example uses the built-in one because [01:42:56 inaudible] being demoed, you get this dynamic filter list by putting any code. Last time I had somebody make a really complicated filter.

Paul: We really need to make sure that we bring our own water bottles.

Kyle: Our own water bottles?

Paul: Everyone seems to have a fountain.

Kyle: Yeah. Some places have real nice water readily available. I have to go fill.

[01:42:44 Water talk]

[01:45:23 Intro with a guy about the training]

Kyle: Is everybody able to run this, make a couple of tweaks? Everybody made a couple of tweaks at the AngularJS code up on the screen? If you don't do that Checkout stuff, you get the ending code which is there's a lot more in it.

[Student]

Kyle: Yeah. I had built that on the older one. Yeah. It's the same idea. They're saying go to the title there, right?

[Student]

[01:46:49] Step 3: Filtering Repeaters

Paul: So this is on step 3 of the tutorial's image. It's a really great image about what are these scope objects are coming from and how they're being used and how they're going on and populated for you. So here on the left we have our template. This is the HTML that was written. [01:47:12 inaudible] We have HTML that has ng-app. At that point, that declares the root scope [01:47:21 inaudible]

Here on our body, we specified our controller. We already built that controller that sets up the scope. But what we haven't talked about is how that actually is [01:47:40 inaudible] from the root scope. That happens a little bit because when we talk about our repeaters, we see that each of these is able to access our scope that was on our PhoneListController. But what we're not seeing yet is that each of these repeaters create their own scope that declares from that phone was [01:48:05 inaudible] So each of those list items that dissolved actually have this whole copy of the scope.

Now over here, we see our input box and how that was tied to the query that was specified, and then all these repeaters, each of these have a phone that split up their own set of the phone. That's why we can say phone in phones and have each of them act individually. Just like here where we have query showing up and fire different places, so here the value for query was you specified each of these values that we're seeing here.

So why doesn't this show the same phone? The answer is because each of these sets up its own copy of the scope with its own value of the phone. That's why they're not all the same phone. And we'll get into more of the nuances of that here very shortly.

[01:49:29] Step 4: Two-way Data Binding

Paul: So now we're going to move on to step 4. [inaudible] the –f says [inaudible] So if you want to say something, you can probably do that. If not, go ahead and…

Kyle: Let's take a break part through this one and I'll switch to me for the end of this one and start with the next one.

[01:50:00 SHORT BREAK]

Paul: [01:51:02 inaudible]

Kyle: You're not pulling in another template that way? You're saying there's an ng? Show me the exact thing you're pointing at.

[Student]

Kyle: Yeah, I get that but I'm saying it replaces whatever. Yeah, yeah, yeah. Is there a way to get bold in there or…? [01:52:32 inaudible]

Paul: So on this program, we have a [inaudible] changes the order here. So let's look at what we have. PhoneCat app the same. CSS. Our controller now has AngularJS property on our phones objects called age. And we specify a orderProp on our scope and set it to the screen age. Down here on our template, we have a new input. So say select input, we use ng model as we did before for binding on orderProp which you know is set to age to begin with. Specifying our values. And down here, you see that we're using another filter specified by character and this filter is called orderBy. This is another built-in filter to AngularJS. We supply the value of orderProp.

The interesting thing is you already need to kind of start to figure out what this is going to do. We know that this might take the value of orderProp, supply it to the orderBy filter which is going to order this list of objects by that property. We've got property page or they can specify name, so they have orderProps of the name, orderBy the name of the phone. So the age of these phones is you do it by age. There's Nexus, Xoom with Wi-Fi and Xoom.

[Student] Where do you put the minus to reverse it?

Paul: Okay. I have to point out a few things here. Like that line 30 says you can duplicate line 36 for example. You can put –age instead. To make it clear what's going on, I usually go after the select and my data bind it to show what that orderProp means. The orderProp is, orderProp will data bind it. On line 43 take that –f out and let's see if that works. Take the other ordering. Labeling matters. The other reversed. So it's a really interesting fit. A lot of people get hung up on here. So this name is something on the scope. You know that, right? Because you can see that very obviously on there. But the thing after query, this is the name of something on the scope. So it's not solely the filter called parameter. This is the name of the parameter on the scope. This is the name of the parameter on the scope. But what if want to hard code this to sort by age? Do you know how to do that? Hard code that to sort by age. You had some answers for that? Do you remember? Age in quotes. I'll take us anyhow. Yeah. So see that it's sourced that way. There's an error out. But that doesn't matter, right? It's hard coded. We're not changing. So if we go back in here.

[01:57:44]

So the thing that can pull us on 43 there's an AngularJS expression. An AngularJS expression is not JavaScript. It's JavaScript-like but it's not JavaScript. It has these features JavaScript doesn't have like this type thing that's really a function call. The thing I wrote in green on there kind of shows what it's equivalent to. And then the colon is kind of like parameter to a function call. And then if you want it to a literal just like if you put food (var, var being a variable menu, you put var in quotes to make it a literal so it's very analogous. And yet in single quotes because if you use double quotes for the overall expression but you can actually reverse that. You chose to use single quotes for the overall expression, then you could use double quotes inside or you can use whatever the other escaping math. What's the escaping math that we do? It's like an ampersand and quote or something. It's really ugly. But it doesn't matter because AngularJS only sees it once it plants in the HTML attribute. So it's just a plain old HTML attribute that has a string value that AngularJS then parses. If those were a JSP page, you can even use JSP substitution here if you wanted to.

[Student] What happens if you reverse that order filter?

- -

Paul: [not sure if this is Kyle or Paul 01:56:05] Possibly. It would be very, very hard to create a situation where that's obviously true. Does this program still work? Yeah it still works. The reason why it still works is that the filter filter is order preserving. Filter filter doesn't randomize the order of things which [02:00:25 inaudible] That's why it works to sort it and then filter it. Filtering and sorting. So I wrote that little s in filter thing up on the whiteboard there. But they have the same idea. It just has to again here. The key thing because you actually don't have to do this if you want to. This ability to order and filter invariably we find this very useful for simple things, and we basically never used it for complex things. Because the complex things we'd rather might quite more standalone testable kind of pure JavaScript code. You put the right data, right? Because you can imagine you can scroll up. You put a written code just to put the right data in but you can use underscore to sort this filter in if you wanted to. Just put the right data on the scope and then data bound to it. This is not the only way to sort a filter. It's just some shortcut stuff built in, but you can easily replace those with something that's more specific to your needs or you can do it in JavaScript and put it out there. So is there anything else to see in this step? Sometimes I go to web page and stuff and scroll to the end to point out a few things. Let's take a short break and I'll do the next step. Five minutes enough for everybody?

[02:02:01] SHORT BREAK

[02:23:11] CLASS RESUMES

Step 5: XHRs & Dependency Injection

Kyle: Thanks. Someday, somebody will invent a way to make these things work when the tech guy is not standing there. That will be amazing technology. Okay so we are on step five, I just went to. I had made my font smaller when I was not broadcasting. So I'll make them bigger now. So this is step five by the way. The index file looks the same as before. This time because I want to highlight the JavaScript more prominently, I'm now going to leave the JavaScript in a separate file. You see that? That source? So in the past I always moved it in so I can see it all at once but this time I'm not going to. And then to get a reasonable display without having to have a full-size window, I almost always adjust these a little bit. See what it looks like. So we have a nice long list now. You might want to know where that long list come from.

Now we have a new thing happening here. So the only change from last time is this file. Does everybody see that? We have several new concepts here. The first is and the most basic way if you want to get data from a server resource you see on line eight there. HTTP, get a relative URL, relative to wherever this page is and obviously does a get and then you provide a success callback function. Some data gets passed to it and then data gets put on the scope. So the interesting thing about this is what you don't see. So you don't see any code to do json decoding because by default with AngularJS, if you use the HTTP service to get some data and the data comes back and it's json data and I think it actually keys off the HTTP content type. But it recognizes that it's a jso data and it knows that you want to work with it as JavaScript and so it decodes it for you and so your callback is not—you don't need to bring a json library. It's already in there.

You also note that since scope.phones, we didn't have to do anything to trigger the page repopulating because we've already seen that when the data on the scope changes, the page automatically repopulates. We'll talk about the magic behind that later. This is trimmed down to much shorter than before but I'm actually going to keep chopping it. This is just sort of a bit of silliness because obviously it's a controller. That's what's in there is a controller. The only thing that cut down can possibly do is be wrong. It couldn't possibly add any information.

I like to show a more idiomatic way to format the code. And when you look at real AngularJS apps in the wild, you'll typically see a format of more like this. "angular.module" returns the module and then you just chain on whatever you want to add to it. This is a slightly more idiomatic way to type this code because it chops off that variable. That variable is giving it a second name to the same thing. If you have two names of the same thing, it's a potential source of confusion. This removes the confusion.

Now there's a couple more new bits. You see these guys here? This code works, right? Let's try to see if it works. This code works. So if you haven't used AngularJS before, you might find the next thing somewhat jarring. But watch this. Let's say that I didn't want to do that. I wanted to do this. This should obviously break, right? But in fact it still works. So does anybody know why that works? Anybody far down the road with any AngularJS to know what's going on here? So this is the first thing that should be a little startling, the sorcery going on here. This is called dependency injections. You guys use Spring, right? How does dependency injection work in Spring? How do you activate dependency injections in Spring? Somebody tell me. You guys use it? How do you make dependency injection happen in spring?

[Student]

Kyle: Say again?

[02:27:24]

[Student]

Kyle: The XML okay. . . . .

[Student]

Kyle: How does it figure out what kind of thing to inject?

[Student]

Kyle: Right, by annotation. So there's names and types of annotations that always replace the trigger dependency injection in Spring. This is what I would write on a whiteboard if we had a whiteboard. So we're in JavaScript though, and in the JavaScript rule that you brought on the framework and you propose people write in XML file, if you go up their json file, you would get laughter not adoption. You can rule out writing a side by side XML file. We would not be having this conversation if they had proposed that. That's kind of the boring one.

But the two more interesting ones we have to rule out are it's an annotation, right? The JavaScript for language does not have annotations yet. It's likely that in a few more years some future version of JavaScript will have annotations. But that is we can't use annotations to trigger injection. The next one is data type. The JavaScript language does not have data types which means it can't use data types to trigger injection. So unfortunately, we're only left with one thing which is the name of a thing. So at this point yet, you kind of wonder the name triggers injection? So JavaScript the language has a really interesting property which is that if you or the programmer have a reference to a function, you can get certain metadata about the function. And one of the things you can get are the names of its parameters. I don't remember the exact invocation, but sometimes I'm able to just sort of remember it. If I said that x=function, that takes to and that just returns not anything because it nil. But if I have this function and I knew what to ask, is it arguments? No. If you know what to ask, is it parameters, prototype? Anyone remember the…

[Student] Args.

Kyle: Args? Is it args? Where's args? Args is what you can do inside. You can do args inside a function to get all of your arguments about value enter an array-like object. But there's a way you can ask. No that's not it. I don't remember. Doesn't matter. We don't have to do it by hand. The important thing is if you have a reference to a function, you can ask the names of parameters. So there's Angularisms going on here. Whenever you see this pattern, angular.module. You have some module.something. Right now, right here it's controller. But there's like 10 other things that could be here. Just remember angular.module.something. You give a name and then you give a function. In that function, the purpose of this function is both to provide a place to put the code behind it. But more importantly, this is a hook so that AngularJS can do dependency injection. So that's going to do dependency injection on that function right there. The dependency injection can only be based on parameter names because that's all we have in JavaScript. Which means that you can infer from those things. There's some kind of lookup table inside AngularJS that has $http with some value attached to it. So there are things happening behind the scenes here that are not immediately visible. Once you've been working with AngularJS for a few days, this becomes very second nature. You just recognize there are certain points that you see all the time every day where AngularJS dependency injection happens. And you get to learn what a bunch of common injectables are. And you can add your own things to be injected and we'll do that.

[02:31:34]

But in this case, the ones we will cover are HTTP and scope. Scope we've already seen before. So previously this text was not here and the injection is why that happened. But we skipped talking about it then because we knew that you might just assume that the control always gets past the scope. That will be a reasonable thing to assume. But it's actually not true. If for some reason you didn't need the scope and the controller, that would be an exceptionally unusual controller but it will be totally okay to leave off $scope here.

There's a lot of care about scope http, this is built-in service that wraps the XML HTTP request. So the key things to know about that is it's like a jQuery AJAX, similar set of functionality. It's newer than jQuery AJAX. So it's maybe a little bit less wrinkly than the jQuery AJAX. But most importantly, it's tied in to the rest of AngularJS, and so it supports being [02:32:37 inaudible] Remember when we got a hook and we set a variable to console and we had it do that scope apply thing to make it appear on the screen. You guys remember that? Because this is an HTTP thing in AngularJS, it knows that after it calls your function, callback, it will call that apply mechanism. So I didn't have to think about doing something to activate the screen updating with this data. Does that make sense?

[Student]

Kyle: There's a whole bunch. We're not going to hit this much because you actually should never write this code. We'll talk about mostly tomorrow. With AngularJS, you should be using promises, everywhere. So you actually should not be using. This is a special form of HTTP that uses a success callback instead of returning a promise. But you really should be using the one that returns a promise. You shouldn't do this. But yeah. We'll point out that you should be doing failure handling. And the promises can reject, not only resolve.

[Student]

Kyle: Say again? I love promises. I use them very heavily. You will learn. Were there any questions on this code? Because there's nothing changed here so make sure that we're all caught up. So I'd like everybody to just make some trivial change here to make sure you understand it. Just go in and change. I'm going to make for mine because I'm going to put in a wrong URL here and just see what happens. And of course it's not going to render. But I'd like everybody to just make some trivial edit. Make sure this works in your machine. Kind of play with it a little bit. Make sure you understand how it works. I'll try making a different trivial edit. What if I just put it the wrong place on the scope? Is that an error? Nope, not an error. Because it's not an error. Put things on a variable to find. Does it still work? That's a good question. So it still works. I'm sure somebody will try this. Like what if I named this HTTP instead of $http. You get an error that you're going to get familiar with in AngularJS. See if anybody can reproduce a similar or different error.

[02:35:16] WORKSHOP

[02:36:41] RESUME LECTURE

Kyle: So the AngularJS Help is really good. What it's good at is adding something you search for a common Angularism. So I put this back before I broke it. You take almost any token name of something in the AngularJS world and you go to the Help and you just paste it in up there. You almost always will go to the right, relevant, helpful documentation for that thing which usually populates. Somebody asks can you put a failure function, yes. You call it error and it gets past data status setters and config.

Something worth pointing out here. These are not named $. AngularJS is not doing injection here so these are not the magic name based parameters. These are just ordinary JavaScript parameters. You put them in that order. So I can play with that a little bit. I can go to the code and it said I could just chain here. Put an error call. And then what was the data status? All I care about is data status. In JavaScript you can just ignore what you don't want. So there's a status. I could put something on, like on the scope. Status, so if I did this, I could go back over on to the page template. Maybe over here is a good place as any. Like that. And then I actually want it to fail so if I make the URL obviously wrong, it should 404 on me. Let's see if I can get that to work. Yeah there we go. See? That worked the first time.

Kyle: So any more questions on–is everybody able to make this thing work? So the error I wanted everyone to see, did anybody try this? Raise your hand if you tried this. Have you tried this? Putting the name up there that's not the one that you were told to use. You get a different error. So get to know this error. You will see this error many, many times in your first month of AngularJS use and you'll see it occasionally thereafter forever. When you see this unknown provider, that means you should look at the thing after the little arrow. That is a parameter you tried to pass somewhere. And there's been no provider registered to be able to provide that thing in AngularJS. So now that I know that. Oh it's not called HTTP. It's called $http. This is as good a time as any to point out is a minor variation. There's a thing called constant. You can just give a name and then you can just put a value so that should be still valid code. And it is. Wait is it?

[Student]

Kyle: Oh there we go. Yeah. So there's a thing called constant. Now this is just the simplest, possible example the least number of characters to demonstrate the injection mechanism. A name that you put here is available in here. So I could say scope. I'll just put it on the ERR there because that's the easiest way to get it on the screen. And if I did that right, see how that number appeared there. So when I mentioned, so this injection mechanism that's being introduced here, the very simplest way to get your own thing able to inject is with that constant. Remember I said there's a whole bunch of angular.module.something. That's one of them. Constant. There's others, too. Okay. I can probably get through one more step before lunch.

[02:41:34] Step 6: Templating Links & Images

Kyle: So step six. Okay so what is new here? It's always fun to look and see what's new because I always manage to forget between each time. Oh yes this is a fun one. ngSrc is being introduced here. Let's have it on step six. I probably ought to actually catch up here. Step six, okay. So step six, this one illustrates a really important point. See these images? So you should be able to run this and just refresh it and see the images. And all the search stuff is all the same you saw before so it's not particularly informative.

- -

[02:42:25]

Here's the informative bit here. I actually don't think this is in the documentation, strangely enough. If you go in here, and instead of using ngSrc, you just use src. So now this image tag is completely plain HTML. Let me put it on the line. So very plain HTML. This illustrates that the AngularJS template process happens inside of arbitrary attributes of arbitrary tags. You see that? So you can use those curly braces almost anywhere. But I set AngularJS runs after the browser loads the HTML. Well there are a bunch of really interesting behaviors that are hardwired into numerous browsers and they were done this way in the early days of the web to make pages load fast. And one of those is as soon as the browser parses an image tag, it's going to start loading something of that URL. It's not even going to wait. The hook where JavaScript gets to run happens after the browser starts loading the contents of this. So if you look over here and refresh it this way, you will see an alarming thing. Is this readable at all across the room? Does anybody know of a way to make the fonts bigger in the debug portion of Chrome? Say again? Yeah I know. I can zoom in part of the page, right?

[Student]

Kyle: Yeah that's what I'm doing and it's not… I may have that feature disabled on my Mac. Yeah I have a hackie way to do it. You want to see a really hackie way to do it? Yeah just lower the resolution temporarily.

[Student]

Kyle: Well this works. Like if I just tell it Ihave a low-resolution screen, it'll come back up. Hopefully with a lower resolution and I'll put it back if it doesn't because I don't want to fight it. Yeah there we go. So it's kind of hideous but now you can read it, right? Fairly? So see that %7B? Has any of you been around long enough, you have an ASCII chart laying around somewhere? That's a left curly brace. So this is the URL in that src before AngularJS had a chance to run and do the substitution. At various places through AngularJS, of which you'll see at least four or five in this class, they just sort of make these compromises to how HTML works. And it's because that they want to keep this basic notion of it's HTML enhanced and the browser goes first, then it runs and does its processing. So if you set src={, it will work. That didn't break the whole thing and it worked, src but they provide an alternate thing you can do.

So let's say I did anything else that looked like, so just blah blah blah src. You know what we would see? Anybody knows what we would see? It just wouldn't work. And so if you don't set src on an image, the browser doesn't try to load anything. That's a really useful property. Well AngularJS adds a thing called ngSrc. What that does is it performs a substitution normally and then it puts that value in src. So this is just a little wrinkle of a workaround to say the browser's going to try to load the source so I'll have it start out empty so it doesn't do anything. And then after substitution, then we set the source so then the browser will load it and this time if I scroll up and down there are none of those extra HTTP requests. The specific which is that you have to know how to use image ngSrc, that's not very important. You can get that from the docs. But the thing that's not obvious is that there's a fundamental reason behind it, which is that AngularJS doesn't get to run until after the browser does its thing. So there's HTML, goes into browser, becomes DOM. Some kind of DOM-ready event loads, then AngularJS jumps in. So you'll see that repeatedly.

[02:47:01]

Okay. It's 11:39 so let's see what's on the next one. And then we'll see from there. We want to try to sneak in a little more before lunch or not. On this next one we'll also have everybody… Oh yeah this is the fun one. I'm going to talk about this in a little bit and then we'll go to lunch. And then everyone will get to reproduce this guy.

[02:47:25] Step 7: Routing & Multiple Views

Kyle: I just went to step 7. So step 7 introduces several new things at once. I use this tutorial because it's readily available for people to refer to. It's out there. You kind of know it's correct because it's from AngularJS. There's a number of things I don't really like about this tutorial. And one is that we'll go through a couple of steps that introduce one tiny little concept. And then we'll hit one like this and there's like four major new things riding on you all at once. But that's why I'll split it over lunch. We'll all be happy.

Okay so here's a new thing, you see. It's the angular-route. AngularJS is packaged as a main file and a bunch of add-on files. This is because you might not be using all the features and they don't want you to pay for the bytes of what you're not using. So if you're not using Angular route, you just don't load this file and you don't pay for the bytes. It's not a huge file. I can probably like if I can figure out where to click, I'll even show you how big it is. Okay so it's an app lib. So Angular itself is… Useless little screen res. So it's 716 on Minify but it's only a hundred minified. For comparison's sake, I guess jQuery isn't in yet but jQuery, here we go. So jQuery minified is 93. Angular, how big is it? It's about the same size as jQuery. Interestingly the unminified one is much huger that's because the jQuery has relatively Spartan comments. And the angular JS file if you read through has huge explanatory comments of how it all works. But for minification right, we don't strip the comments. We strip all comments up, throw them away, so it's really not a big deal to strip.

What I was getting at is the reason it's not a big deal to strip is they took all these things you might not be using out. So Angular route when minified it's only 4k, so it's actually kind of questionable whether it's worth splitting it out but they did. This is Angular 1.2.something. And they said for 1.3, a whole bunch more things are going to be split out by the way. Anyway, so if you want routing you go to include this file or this one if you're minifying. We'll see resource in a few minutes or in a few steps rather.

So the order matters. You need to have Angular before you can add an Angular add-on. And you have to have Angular before you can have your own files. We'll see those files in a minute. That's necessary because these files which I'm probably going to combine in here, they will say

Angular.something and you can't say Angular.something before Angular is defined. I really like to combine these things. So I am going to put this one right in here. Because I think it's easier to explain when combined. Okay well actually that didn't help as much as I would have thought. I won't be able to do that this time. They have divided into two different files like this. You don't have to do that. I'm going to show that you don't have to do that. That's just a convention that this example follows, not a particularly good convention. It's not a convention I would necessarily recommend you guys use. It doesn't make any sense to me that you divide your controllers into their own file. There's many, much better ways to do that. Then we can follow this convention of doing method chaining. What happened?

[02:51:15]

[Student]

Kyle: Where? That guy right there, right.

[Student]

Kyle: Yup, okay. Yeah okay. Oh I know why. You guys know why it changed on me? It's because I didn't change my fonts after doing this bit. See if it works.

[Student]

Kyle: Okay that's fantastic. I will definitely do that. So I don't need him visible. I'll use thse features so if I'm in Chrome and I go in the console, then I press Command +, yeah. That's fantastic. Okay. Where were we? Don't let me get too distracted by cool features. I put this on the one screen so I can talk about them all at once. That means I don't need to look at this controller file because we're actually not even loading it. If you look in here, I'm just loading one JavaScript file. I like to stay closer to production practice and put these down here where they are intended.

So multiple new ideas. Let's start on top of the page. I don't think we've mentioned this one yet. This is the dock type for HTML5. Angular does not require HTML5 but you should probably be using HTML5 in 2014. This is the name of a module. You can always look here. It will be something that appears on an angular.module line. ngView is a new one. Simple example apps usually have some template right here on the HTML page, but complex apps very often have many different sort of screens that work in one page. That's a single-page app idea. ngView is a placeholder to say something will land here. Does anybody use Rails like Django? What web frameworks are you using on your Java JSP stuff? So Spring MVC, you use annotations. Say this block of code serves this route. Familiar? I think the way this one works you'll see it came more from the Rails world where you typically have a single file that says here are all my routes. Because right in here, this is how you can figure your routes. Now you'll see what routes are in a minute. But you say config and then here this syntax, I will defer it, testing with that syntax for a moment. So you say config and then remember I said whenever you have and angular.module.something, that next function you put inside usually has dependency injections. So our dependency injection a thing called route provider.

Slight detour. Angular has a kind of two points in time when things run: when you're first loading a page and then as interactions happen. Configs happen when you're first loading a page and controllers run when you need to prepare a scope. So this could run once, this could run many times depending on how you navigate. We can figure the route provider. Now this is a list of routes. I find it a little bit hard to read when they're written as objects. If you're familiar with any of these other tools, if we combine these together to make it look more like a table, it's a little more clear what's going on. So now is this a little more clear? If we're on a page whose URL looks like this, and we'll see what I mean by URL in a minute, then we want to load some templates. This is a relative URL of some file and then we want to activate some controller to control what goes in that template. Or if the URL looks like this, you want to run this other one. Remember that I said a lot of new stuff going on, so uScript. You talked about uScript right? This baby activates modern JavaScript. So you don't get the ancient 1996 JavaScript. angular.module, the name of our module. If your module depends on other modules, you list the dependencies there. So it's sort of like it uses clause and languages that it uScript uses or import in a Java. So that's what lines three and four mean.

[02:56:10]

Merely putting Angular route on the page is not enough. You also need to say right here that you need to load that module. Okay so PhonecatController. That's another module. It's just a module down here. The division of JavaScript among files is totally separate from the semantic division of JavaScript and to this module system. So you could, in theory, write your entire JavaScript program in one large file and still use the module system if you wanted to. I don't recommend that but you could. Tomorrow we'll talk about what we do recommend. Okay so this is a way of saying this module depends on that module. If you didn't say that, then this other module, it would not actually be loaded. By loaded I mean it would not be registered in Angular's module system. So you might object. But wait. The JavaScript will be executed and that's true but for reasons that are not entirely clear to me, merely executing this JavaScript does not make this module visible to this module. You have to actually declare the dependency here as well. Everybody follow me so far?

We talked about config already. We're configuring routes. Now this looks like a little bit of a lookup table. You might wonder what do these look like. Well I think the word partials comes from the Ruby on Rails world. You guys ever use the word partials in Java? Yeah I never did either. It's the Ruby is I'm leaking over. So PhoneList. This should look familiar. This is the HTML that used to be sitting inside here. So everything that used to be right here, has been extracted to sit in a separate template file. So when this route is activated, the contents of this will be parsed. It will become DOM and will get stuck in right there. So Angular route, ngView works with ngRoute and you use routeProvider to configure it. So those four names all go together unfortunately. We haven't seen the second one yet.

- -

We'll get to that in a minute but if we just ignore line 10, we concentrate on line nine, this is getting us roughly back to where we started. We're doing method chaining here. So routeProvider got in. You can tell it's an injectable. You can tell that this must be a JavaScript object that has a field called when which is a function. Is anybody not clear on why? I could reason that out. So if you're not super familiar with JavaScript you're going to end up learning JavaScript really well. And then we've chained on here another one. So it says routeProvider.when.when, so you can infer from that but this is the method chaining pattern where they've intentionally made it so that the when function returns the original object again. So you can keep depending things on to it. Put that up there for clarity. And then there's also something called otherwise. You can probably guess this means that if you're not at any of these routes, it's going to take you to that route by redirecting to that.

So everything you saw there sounds parallel to what you've been doing on the server side at Java JSP but it's actually not because it's all happening in the client. We'll see that in a minute. And then the rest is code, this code, this is the same as before. We already had a PhonecatController's module. And we already had a PhoneListController in it. We've added a new one for detail and we'll see how that works n a few minutes. I think if we compare the last time we used to directly invoke the PhoneListControllers module up here and now we're saying this overall page uses the PhoneCat app. PhoneCat app uses PhoneListControllers. There's a transitive dependency there.

[03:00:37]

Okay so now you got to read the URL bar. Is there a way to make the URL bar bigger? I already did that command. It's like Ctrl +? Command +? Nope. Anybody having trouble reading the URL bar? The important bit which I'm going to bring this over into a text editor where I can make it big. So this is the bit I'm trying to call out. This is a page URL and this is the marker between the page URL and an anchor on that page. So we're using these in the default way, you're actually talking about a fragment here, the anchor fragment is what those routes refer to. As we navigate, we'll be seeing on the same page, from a loading a page over HTTP point of view, we're navigating the fragments on the page. Now, these URLs are really ugly. I don't like these URLs. So I don't use these URLs. There's a way to not use the URLs. Paul, do you know what the name of that is? Maybe you can look up one of our apps what the name of that is. But there's a simple invocation you can do. If you have control over your servers, you can make your server serve the same page regardless of the URL fragment at the end. And there's a way to make it where—this one is just app/phones—but to make that work you need to do a little serve side program. So you need to either use an Apache directive, or if you're serving directly from Java, you got to have a little servlet filter that says ignore the part at the end and serve the same content. The server phase is kind of out of scope for this. I just take the default. Did you find it? Okay.

So when I say route, I mean this part.

Paul: Location provider?

Kyle: Is that what it's called?

Paul: Yes.

Kyle: That was incredibly unhelpful Help. Wait, didn't I say HTML 5 up here? And where was that?

Paul: It just says at the end of the…

Kyle: Does the route help? I'm trying to find the page of the Help that talks about it. I looked on locationProvider and it wasn't there. And I looked on location, which is what locationProvider returns is it returns the location, and I tried to look for HTML 5 here and I didn't find anything down in here either. You said you searched on stack overflow to find the… Say again.

[Student]

Kyle: No, it won't work because you have to modify your server side to serve it. I'm just trying to give you guys a hook so you can figure this out later. So what do I do it for?

Paul: Removing the # symbol from Angular?

Kyle: Remove like that?

Paul: Yes.

Kyle: That did not.

Paul: The name of the question is removing the # symbol.

Kyle: So I actually type that. Like that? That is awful how hard it is to find this even though I know it's there and you're telling me how to find it. That is really terrible.

[03:05:05]

Paul: I think the question [03:05:10 inaudible] 14771091.

Kyle: That's completely intuitive. So this is the line of code I was talking about. For some reason, it is not documented in the Help for locationProvider which is really, really terrible. But if you do this, it will change the whole URL back to the slash. If you did that line of code, then it would do this URL is what it would mean. See? And so to make this work, you need to be able to go program your HTTP server to serve that content even if the user loads this page. Follow me? Which of you does things like servlet filters? You could easily make a servlet filter that recognize a pattern like this and you wouldn't issue an HTTP redirect but you would make it serve this content instead. It's pretty straightforward.

So where were we? So this stuff here is the same. I want to just point out how this works. So in here, I'm going to click on 1. You see the URL up the top changed to /phones/someID. These are just alphanumeric IDs. Nothing special about it. The cool thing is like the back button works, and if you want to really see what's going on, what you really want to do is get over here, clear this out. And then if I click into one of these, you're seeing it reload the images. This phone's json and the images reload. That's a little bit confusing. That's because the node HTTP server I'm using for development reason serves everything with immediate cache expiration. These pictures could be cached. But what you don't see is the page being reloaded. So see as I go back and forth or if I click into one of these, you don't see anything appear over here that loads the index HTML again. When I come back, again, I don't see the index HTML appear anywhere. This is a single page app. It has multiple navigational states.

Those navigational states are coded into URLs. That's good web practice. It will be bad web practice to have it where the URL just stays the same like that, and then you click around and the URL never changes as you go through the app. There are tons of apps out there that make that mistake like banks web app makes that mistake. It's incredibly obnoxious. If you make the URL always stay the same, that means it's impossible to bookmark a navigational state of an app. A good web app is one whose navigational state can be bookmarked because when it has that property, if I click on one of these things and then I were to take that bookmark and go to a new tab, I would land right back where I was before. This is a good web app in that sense.

It's twelve o'clock. It's probably time to eat. Why don't I answer questions on this and then we'll eat and then we'll come back and try to have everybody take a step forward. Questions about any of this code? This is all new.

[Student]

Kyle: What are you saying shouldn't break?

[Student]

Kyle: Yeah, it did break. I didn't even notice that CL's not bound. That's because it's broken. I didn't… See? Ooops, sorry. It's kind of obnoxious. It's not obvious, right? I only had to click over here and notice it was broken before. I end up leaving the console open all the time when I'm working. I also don't notice when I break things. Questions? I didn't get the right thing up. About all the new stuff that's going on here.

[03:09:36]

Does anybody notice the thing on line 10 yet? See that thing on line 10? Yeah. That's what I'm talking about. So what happens with that? Can you see what happens with that? I'll take this off so that it's… So you see where that value flows? So this guy is a route param. There's a route param service you can ask to have injected just by listing it in your function parameters, and that route param service just comes through as an object that has these names on it as fields.

[Student]

- -

Kyle: So when routing occurs, it will remember this. It will send you in to this template with this controller, and then route params, which can be injected into that controller if you want to, will know about the route params for wherever you are right now. Does anybody notice anything about this at the top? It does not say ngController up there. So that ngController is what you use if you want to kind of manually stick a controller in the middle of the page. When you're using this routing mechanism, there's this nice declarative way to list out here's my route, here's my template, here's my code behind the template. Here's my route, my template, my code behind the template. The real application could easily have 50 of these here. Somebody ask me a question. No questions? Hungry? Let's go eat.

[03:11:43] LUNCH BREAK

[04:09:43] CLASS RESUMES

Kyle: Everybody ready? Is everybody here? Who are we short? Well, we'll start doing a little workshop in those minutes. So the only thing that we ever need to do to just kind of get a clear at this is just pull up the example. You don't have to make any of the changes I made if you don't feel like it. Just add another route. So just make any route you want. I'm going to make mine Kyle. And then just kind of get into habit and work through the motions, make some mistakes and fix them or whatever you want to do. So I'm going to do something like this. I'm not going to a controller on mind just to show that you don't have to.

[04:10:28] WORKSHOP

[Student]

Kyle: I don't really have a partials directory. What I do is I organize my code as part of the application it is, and I make the directory for that part of the application, and I put a JavaScript code with my template so whatever I need led into that. You'll have a chance to do that. The complexity increases each time, right? Anybody got their own router on yet?

Kyle: Someone will make this mistake before the class is out. Where's my mouse pointer? There we go. So in these routes, all the keys are optional. In JavaScript, there's no data type, right, so you can just throw any key you want into any object. So if you think that URL is spelled like that, that's how I think URL is spelled, you get this spectacular behavior but it's just that not working. It routes there. It just doesn't bring up the template. Isn't that fun? So this is not an Angularism. Just pure, standard, straight JavaScript issue, and all the alternatives you have this over on the server with node, not that, which is that it's all varied as I type. It's not an error to say something extra you don't use. So now it'll work again.

[04:13:43]

[Student]

Kyle: Is your IDE trying to help you? Cool. Want to see how much a smart IDE can help? Your ID is probably pretty smart but my IDE is not that smart. On mine, I could probably do this now, and because it doesn't have like a data type, right? See what my editor did for me? It doesn't know that. It just looks at what you use before, so it actually shows me built answers. So it's not as helpful as it first appears.

[Student]

Kyle: Yes, and I'm in a JavaScript file so I probably could use it. Is it Shift Command P? Indent, re-indent lines. There we go. So yes. I don't know if it has full detailed formatting if it only has indenting.

[Student]

Kyle: I switch back and forth because I end up having to work with people who use all kinds of different things, so I grab whatever's handy. We've been using one called Light Table recently. It's really nuts. Okay, everybody got route edits? You should have a route edit or you should have a question for me about probably why you're not having a route. You have a route? Add another route? Anybody having any trouble adding another route to this thing? Got a route edit?

[04:17:52] WORKSHOP STILL ONGOING

Kyle: While we're waiting, I'll point out one little thing. So this code right here, everybody see this code? Remember I said these names are being used for dependency injections? Is anybody doing uglification or minification or their JavaScript code? You guys doing minification yet? So if you minify, it does things really helpful like make all your variable names shorter like into single letter variable names to make your code shorter. That would break this mechanism. Angular provides an alternative way to do this where wrap the function in an array and then you put the names as strings in the array. So if you do this, this is the exact same meaning as before but we elevated the names into strings which will survive minification. So al lot of the examples, they always type it this way so it survives minification. But I actually don't like typing it this way. So what we do is our build process includes a tool that goes back and inserts these automatically during the build process so that we can write clean, simple code without duplicating these names. In the build process, it doesn't ask you to make it minify correctly. I think that's a better way to do it. But you should understand the syntax.

[Student] In this way, if you were to swap the two parameters and functions?

Kyle: Yep, because this order makes you this order. So these become positional. When you use this syntax, these become positional to have to match. And it's really hard to never make a mistake changing to this. Like if you had to change to this and keep them perfectly and sometimes they have 10 + items, you're going to make mistakes. So that's why I don't like typing it this way by hand. That's why I end up changing it when I see it to make it shorter and easier to understand. But for a while the tool didn't exist to automatically add those. Everybody really was typing it this way so it would minify. But now that that tool is out there, it's like one grunt operational way. There's really no reason to be typing this by hand anymore. It's just not a useful way to work.

[04:22:16] Step 8: More Templating

Kyle: Next step, we're on step 8. In step, I'm going to leave this code help coms. I'm not going to do that reformatting again because I already showed. You guys should be able to understand this at it is now. I just rearranged it to make it understandable. It's not much different here. So the controller style, this list thing was already here with the http.get. Now we have a similar http.get here in the detail controller. So here's the pattern I just showed where they have the names, then I repeat them. So it survives minification. All this is at is the same as what you already saw. It's a get. This time we're doing string concatenation in the form of the URL. Not a good idea. And so success function we put it in a thing called phone.

The smart thing though is that this is not much code, and it could be less code, but this populates all the data. It's making this real complex detail view work. So if you refresh and you click into one now, you get this real complex detail view for a phone. It says a bunch of HTML. The details of the HTML are not super important. But if you want to look at them, they're in phone detail. So like here's this image repeat thing we saw before using the ngSrc. Here is the first image, right? So phone.image is zero. Gets put into the ngSrc of the big one. See, there's one. Here we are doing an ngRepeat with a bunch of these DDs. I hardly ever use DD and DT in my HTML. Do you guys use DD and DT much? I actually had to look up what they meant because I never use them. Yeah. And then DL. It's like a definition list or something? Definition title? I don't know. It doesn't matter. But there are a couple of wrinkles in here that are kind of fun. You haven't seen this before. You can go deeper there. This stuff.the other.the other and that works.

What we're seeing here is so you already know that if you do something at the end and it's not recognized, that's not an error, right? So if I did this, I will not get an error. The interesting thing is again these are Angular expressions, not JavaScript expressions. They're incredibly tolerant. If this first thing is not defined, like it's not it finds it and then it tries to look for something in size and weight and throws a no pointer exception, which is what happened if this was Java. It's very tolerant. So even adding that little bit, it's not going to emit any kind of error. It shows that some piece, the size and weight which used to be formatted or populated nicely, that dimension is now no longer populated. So it just skipped it. So since this didn't exist on the scope, if there's anything that doesn't exist, it just goes on merrily to the next piece without error. See? No error. I just skipped it. Put it back.

[Student]

Kyle: Well, if you just go to the starting point URL, it automatically redirects you to/phones. And I just click on phone which we did before but should direct you to one phone. Did you do the… Yeah, you have to do refresh because remember just clicking up on the links doesn't make it refresh. Those are all intra-page links. So you have to tell it to refresh.

- -

[Student]

Kyle: I think that would still be equally true on this. Yeah. This has some really nice properties. I think Wicket shares this property. It's been a while since I did much with Wicket. But the slick thing is that unlike JSP but like Wicket, this information, these extra add-ons, end up in attributes that would be ignored. And so if I… I got to do this.

[04:27:37]

[Student]

Kyle: I'll think about that in a minute. Let me get this first one up. These things are okay as HTML which is kind of nice. You just see the little placeholders. But if I wanted to actually make a page that it includes that, then apply CSS or something. I'm really in just the HTML so I don't have the CSS working. But the nice thing is that it's actually correct working HTML. I think it's a nice feature.

For placeholder images, I'm not sure exactly how I would do that. I would probably just have it working in my dev environment and have the images inserted that way. But someone had asked along the way the advantage of using, what was it, ngBind include? Is that what it was?

[Student]

Kyle: Right, so the slick thing about—actually ngBind, that would be good enough. The slick with ngBind is that it lets you take things that would be there, so that could go away and could go here, and this could be…

[Student]

Kyle: So you can do something like that and it would work.

[Student]

Kyle: So if I did this, I would see the 27 pounds there. Practically though, I would just work on it in this way, and I would edit the CSS and I would refresh it. Or if I wanted to edit something right in here, you can even do the thing where you can just maybe start style or get right here or something, right?

[Student]

Kyle: Yeah. That's why I type this way because this one will probably work. I think if I bring the page back up and find that dimension is weight, where is that? Here, see this? It got replaced correctly. Yeah because the way the ngBind works is that it means replace the contents of this element with the result of pulling that off of the scope. So the scope had this in it, and that ends up replacing this.

[Student]

Kyle: Well, there's a thing called ngBind, the template, and that lets you type expressions in here this way where it's more like a template, right, and you could do something like that.

[Student]

Kyle: They have quite the same effect because they had a bunch of instructions jump to it but it looks like this now. So it takes this. It treats it as an Angular template and does the replace, and then it replaces the contents of this element with the result of the value in that template. These 27 pounds goes away, and it gets replaced with the result in evaluating this.

[Student] Can you also use a mechanism like that to set, for instance, things like a classified element?

[04:30:56]

Kyle: Yeah. There's a whole bunch you can do. But just like you could say, I don't know, like wells. Wonder whether that's built in. You can do something like this. Why is my scrolling not working? There we go. See we got that box right with the well built into bootstrap. So there's a thing called ngClass and then you can put a something scope here. So this would actually be the name of something on the scope. Am I using this right, Paul?

Paul: Yeah.

Kyle: And then over here on the scope for the detail, you could say scope.class=something like this and I'll break it first. Make sure I really see it working there. So there it is, broken. And then if I decide to programmatically assign some… [04:32:00] I can't scroll my page with my wheel. So there it is working again. See? And there's a whole bunch more. I'm just showing you just a couple of the things. The important thing is to understand the mechanism because there's a whole bunch of other features, way more than I could possibly talk about in three days. You can really easily add your own features if you don't like those.

The thing we're seeing on here is only that there's just a bunch more complexity possible in here. It stays valid HTML. There's one little bit of invalidity that we'll cover, but it's almost valid HTML. There's features like this. This again looks like JavaScript. It's not really JavaScript. It's an Angular expression. So if you misspelled features here, it won't crash. It won't error but just go on. So be aware of that. It's probably enough for that one.

Paul: Did you start the recorder?

Kyle: Yeah, I did.

[04:33:04] Step 9: Filters

Kyle: On to step 9. This is kind of a clever one. It adds an interesting bit. You see that? You've seen this syntax before, right? You saw it before when we were taking a collection and passing it through a filter function and a sort function. Well this is a general purpose mechanism. You should read this so when you see this, what you should think about would be it's really more like that's a function.

If you thought about it in a more traditional language syntax, think about it as being more like that. They're piping a bunch of these things through checkmark. Checkmark is not built in. The idea here is to see what new things have been added. On this index page, in this example, they put the filters in a separate JavaScript file, which I typically wouldn't do if I had a filter that was for some specific purpose, I'd put it with the other code that's for that purpose. But we'll take that just to see how it works.

So it filters the JavaScript file and here is a filter. To make it a little bit more recognizable, I will do that. So angular.module, you've seen this syntax before. Remember when I said when you add a new module, you have to declare dependency on it or it won't work. So if I go look in app, the main module now has a dependency on the filter's module. To make the filter, you say .filter. You guys guessed that pretty easily. Remember when I said whenever you say angular.module.something and there's a function, right there you could do dependency injection although they didn't for this trivial filter.

[04:34:57]

This function is supposed to return a function which is the filter. See how you can kind of get your head around that. You don't put the filter right here. That won't work. The purpose of the outer function is to provide a place where Angular can perform dependency injection. That function is supposed to return another function which is the actual filter. So if you haven't done a lot of functional programming, you will because JavaScript is a somewhat functional programming language, and you end up using a function that returns a function that returns a function that returns a function. That is a thing you will do in Angular from time to time.

So you call that filter with an outer function whose purpose is it could do dependency injection. It returns an inner function. That function takes and input, and if it's truthy, it gets this, and if it's falsey, it gets that. If you look up these Unicode characters, they'll just be a checkbox and an x. So just a very minor improvement. You see here where it says gps true, infrared false. Now they're x's and checkmarks. It's a tiny bit of functionality. But I had mentioned that you can use this to develop a sort of a vocabulary, a company level vocabulary. You can imagine a person with a good understanding of Angular writing this code and there being some set of these that's readily available that you kind of decided company-wide here are all the different ways that you want your special data representations. You can imagine somebody who's really working more at the HTML level understanding enough to write this. Understanding that they can look at some example json, understand the data structure, understand how to write some dots to navigate it, and then understanding that they have a vocabulary of things they can type with this, and they have an example of about 15 of them, one of which is a checkmark.

[Student]

Kyle: Yeah. So for example, when you see this pattern of array of an element and all it has in it is a single double curly brace, it's really more idiomatic to not type it there. It's more idiomatic to type it here. I'm told that there is some microscopic performance advantage too. I don't know how much it really matters. It's more idiomatic to do that, and it will still work. So if I refresh the page… When I first refresh the page, I can't scroll with my scroll wheel. gps checkmark still works. And then you are asking if you could type anything you want here, yes because when the binding runs, ngBind means replace the contents of this non-element with the results of this expression.

[Student] Why is ng…?

Kyle: So Angular. This is your hint it's an Angular thing, but it's not a reliable hint because as you will see later there are things in here where Angular has changed the meaning of built-in HTML elements. There's not some rule that says every Angular thing has to be an ng on it. It's just that most of them have an ng on it if they're all new. There are ones that are built in to the changes, and we'll do that.

[Student]

Kyle: Don't name yours ng. So you guys use Oracle and MySQL right? Anybody hear of the .NET SQL server world? There's a great example from that.

[04:38:34]

In the SQL server world, they're really into Hungarian notation. Do you guys do Hungarian notation to your stuff? I see a yes and a no. It's pretty funny. It's where you put like little prefixes in front of everything, a little i in front of all your integers and so on. I hate Hungarian notations. In the SQL server world, all of the built-in stored procedures are named like that like sb food or sb something. And so people that are in that world think, "That must be the Hungarian console to sname all sorts of procedures." But the problem is that's not true. This is the Hungarian for system procedure. The sb_ triggers a different behavior in SQL server. If you name your procedures with sb_ it breaks certain things because it tries to treat them as system procedures. I hate it. Any more questions on the tiny bit of functionality here? Everybody to pull this code and either edit this filter or make your own filter that does something. It can be really simple. It can be very primitive. Just go through the exercise of adding a filter. Do some kind of filtering.

[04:40:05] WORKSHOP

Kyle: Here's one way to break things. Want to see how to break things? I can take this and I can think, "I need to do the same thing again." Which by the way, the reason why this is a bad way to type the code is because this mistake I just made is much easier to make. Maybe I'm going to change mine to be a y and then a nope. And then what will happen? So you have a y and a nope. What happened? What happened is that I redefined the same module and it overwrote that one, so the old one got overwritten by the new one. So you really don't want to overwrite the same module. What I could do is I could chain. I could say define a module and I will put this filter in it and I will put another filter in it. I want to call this one yn. And now I have two filters. And then I can go into my detail and I could say that sometimes I want to use yn. Of course, checkmark use. Here we go. So for the connectivity, I want to use yn. The gps, and then I go here and my gps is using my yn. My other one's using the checkmark.

[Student]

Kyle: The things it can return? I think it can return pretty much anything.

[Student]

Kyle: So you wrapped it in a fan. So you did something like this? Like that you did, right? And then when you ran it, the word span appeared? How come I can't scroll? Broke my Mac. That is the problem, right? That's because you're getting automatic encoding. It's automatically encoding as it comes back. The reason why it's automatically as it comes back is it assumes its data. The data that's coming out of a binding expression is potentially untrusted data. And so it's trying to prevent cross site scripting attacks. If you really want to do that, if you want to let it happen, then there is something you can look up, which I don't know exactly offhand what it would be, but it's something like ngBind html unsafe. See if I can guess it. Unsafe might be enough to get it. html. ngBind html. We have to make sure that sanitize is available. ngBind html. This functionality has changed recently. Like that. So let's see if that was enough to make it work. Yeah. I think the span went away. So the bind html can do certain things but not all things. There's an unsafe variant. It's not in the box. We've had to use it before. It lets you put arbitrary HTML. But this one is a somewhat sanitized HTML.

[04:48:05]

[Student]

Kyle: So if I wanted to do it once, I would do a… I will go right here. The nice thing is you can break lines in HTML, right? It's like it makes this easier to see. You can do ngClass=. You've seen this before. But there's more you haven't seen, and that's you can put a name value situation in here. So you can put a class name followed by an Angular expression, and the Angular expression will be evaluated for its truthiness. So if I do this, I would get a well class only if gps is true which it is and it got a well class. So that's one way you can do it. That's how I do it if I was going to do it once. If I was going to do it more than once, I would make a custom directive, which we'll see when we get to custom directives, and that custom directive would do like my company-designed way that we're going to theme across the whole app to represent true and false. And I will just say instead of ng, it would be sc or whatever, sc bind truth or something and then just encapsulate all the functionality there once and it just happens seamlessly across the whole thing.

Paul: I will also recommend doing that in place of returning HTML.

Kyle: Yeah. I just showed you that to show you that it's possible. You shouldn't do this. This is not considered a good way to do it. Did everybody get at least some tweak to their filter? Wave or something if anybody's having any trouble. Paul will help.

- -

[04:49:52] WORKSHOP

Kyle: Filters have to tolerate any old crap somebody sends to them. So this filter could be called but darn near anything in this input. So this isn't JavaScript but we can declare that this is a string. Or this isn't Java but say it's a string. This is any JavaScript value could be passed. Undefined is being passed. That means whatever code you put in here has to tolerate undefined, the JavaScript value undefined is the input.

[Student] So in this case why is it being passed?

Kyle: I can answer that. It's really cool why it's being passed but we won't fix it until considerably later. But here's why it's being passed. It's because here's what the template looks like. Blah, blah, blah and a checkmark whatever. If we look back at the top level code, here's the template, here's the controller. Let's go look at that controller. Here's the controller. When this controller runs, what will the scope look like the moment this controller is finished running? Anybody know? Just say so if you know what the controller will look like the moment this finishes running. The scope, what's the scope going to look like? Because this is JavaScript, everything is asynchronous. It will not contain scope.phone line 19 will get executed when the HTTP call returns, which is going to be sometime, hopefully during a little time later.

So when the controller function is executed to initialize the scope, it's actually not going to do anything at all, which means that this template which we just saw, this will first be rendered with nothing in the scope whatsoever. So again, it's all very tolerant so no errors are going to come out. This is not going to error just because the scope does not have phones in it. Nothing in here is going to error because the scope doesn't have a phone in it. But that does mean that all of these things are going to get called at least once with undefined because if this doesn't exist, this certainly can't. This can't. This call button detolerates that, makes that undefined, calls a closer function on it and renders that.

[04:57:03]

Now you can't see it very much because it doesn't take very long. But yeah, it's taking time. So did everybody get their some simple filter function running? Next.

[04:57:22] Step 10: Event Handlers

Kyle: Step 10, we are almost to the end of this built-in thing which does not take us very far. It's amazing. Image swapping and ng-click. Let's see what that does. If you look at this code, you'll see just one new thing in it, actually two new things. This now says main image URL instead of being hardcoded to be the first image. And down here, there's an ng-click. You should recognize this. It's like an onclick. It's an Angularism. An Angular when you see some name, you have to think where is that name being resolved? So a plain JavaScript the answer would be that that's being resolved in the global namespace in JavaScript which does not scale well with complex apps. The answer in Angular is always that names here will get resolved in the scope. So from this,

- -

I can tell you that the scope that serves this must have a field in it called set image which contains a function which takes an image value of some kind.

If I go look closer, that will turn out to be true. There it is. So scope contains a field called set image which is a function that takes a URL. It sets the main image URL to be that URL. We initialize the main image URL to be the first image URL. So there's not a lot to this. But if I click on one of these images, I guess I got to give up on my super wide thing here, if I click on one of these images, it changes the main URL. So you might think it's pretty easy to do in jQuery, right? But the slick thing is that it's all an implicit automated binding, which is pretty slick. So all this set image does is it sets some value on the scope. It doesn't actually have to go tell anybody to do anything with that. The thing is automatically going to notice that that value changed and going to change to the source to this image.

[Student]

Kyle: Yeah. All you have to see here is this guy and this, so if you just remember those two names, here's the code. Almost no code for that. We set an initial image URL to be the first image, and all the set image does is it sets something on the scope to be the parameter those passed.

[Student]

Kyle: Over on this template, this is a new piece. ng-click, this is the Angular version of an onclick, click handler, except instead the onclick just looks on the global scope and you're kind of on your own to make all the right things happen, I mean the global JavaScript scope. In the Angular world, things look on the relevant scope for where they are. But we already know the scope for this file is the one that gets set up in the controller for this route, right? So this stuff, it's not just all flying all over in a single global namespace. It's going to land in a reasonably controlled namespace.

[05:00:31]

[Student]

Kyle: Yeah. So this is the ngRepeat mechanism. So it's the iterator. If I have a look in the raw json file of this pulls from, I'd see the images must be in array of URL strings. This takes each value in succession, makes a small image for it, and each [inaudible] that says click handler on it. It's just the tiniest bit of codes. The only part of this step is to demonstrate how ngClick works.

[Student]

Kyle: Yeah, everything you could ask for. Yeah. The ones you want are not already there. It's usually some add and you can grab like that to get it like there's a touch add-on. It gives you all the touch events, for example, if you want those. I think I mentioned earlier. The thing that I don't like about the built-in tutorial here is some of the steps are big and some of the steps are small. This is a pretty small step, right? We didn't want to introduce like six new concepts for routing. This introduced the ngClick. So carry muddily on.

- -

[05:01:44] Step 11: REST and Custom Services

Kyle: This one is complicated. Okay. I have new things to show you here. Remember I said that HTTP with manual code to assemble the URL is not a great idea. There are more abstract ways to do it. This is one of the abstract ways to do it. I don't really recommend this Angular resource because there's better third-party stuff that kind of does more for you. But this is some value and it's in the box. It's worth as a gradual introduction. So there's a part of Angular called Angular resource. You can probably guess the official name of resource. Anybody guess the official name of resource? Come on.

[Student]

Kyle: Yeah. $resource because all the Angular stuff has $ in front of it. If we look at the top level JavaScript file, we will see I guess—no we don't see it here. We see it when it uses it. Did I pull the right code? Where's my resource? Oh there it is. So there's a change here. See this Phone thing? It does not have a $ in front of it. That means that it's being defined by this application. It's not a built-in thing in Angular.

There are some evil hackery going on here that I will explain in a minute. This controller used to have a $http call on it and a success handler and so on. It now illustrates an anti-pattern in handling asynchronous data in JavaScript. So this is evil, bad JavaScript code. You should never do this. They made a thing called phone.query, and look. It looks like it returns the data. Isn't that cool? Looks like it returns the data. But does it just return the data? What is the answer? Does anybody know? You should be able to know the answer to this. It's just because you haven't been in JavaScript.

[Student] Is it how long it takes to run?

Kyle: The answer cannot ever be true. So yes to what I said. Because there is no means in the browser to write JavaScript that blocks. It just doesn't have that feature. There's no way to say fetch http and wait for it to come back. If there was like the web wouldn't work, right, because bad programmers all over the web would write little bits of JavaScript that block the browser to wait for something to load.

[05:04:31]

You can't do it, which means that this absolutely does not wait for the data to come back because it can't. What it's doing is evil stupidity. It returns an empty array but it holds on to the array and it changes it out from underneath you later. Contemplate the programming model there. I write a function. You call my function. That function is some kind of data structure and you start trying to write codes that does something with that data, and some unknown amount of time later, I held on to that data and I just change it up from underneath you. This is not good. It's supported for backward compatibility because this is like somebody's first idea of probably how to solve this asynchronous problem.

Anyway, so move on here a little bit. The PhoneDetailCtrl uses $http and do the concatenation to build the URL. Now it has this kind of opaque thing. It has Phone that calls get, so you can guess that that's an http.get. It has this way of passing in a parameter like a named parameter inside an object here. You guys already know what route param means, right? It's from the thing up the topof the URL. You've seen this much before. But there is this new thing going on where we have a way of passing it as an object with this name phoneId and then there's some function. You should be able to guess this is the function. This is a callback. The function that happens at the end, right? This is doing the same thing, by the way. So this phone.get, it returns an object. But then that object it's changed out from under you when the http call completes. That's why I said this is an anti-pattern. I just teach it because it's in the box and you will encounter a code thfat does it. And then this code is the same as before.

So what is this Phone thing? Here, I can take this out to de-confuse. So what is this Phone thing? Well, this is a new kind of Angular thing we haven't seen before called a service. Here are some unavoidable complication. In the Angular world, we often say the word service but we often type factory to mean service. There's a thing called Angular service and Angular factory and they're about this far apart conceptually. But for various reasons, people always call them both services. But for other reasons, most people prefer how factory works as a way of typing them. And I'll show you later what the differences are but just kind of accept for the moment that we'll often say service and we mean service or factory.

So you're pairing this down a little bit. So remember I said resource is really named $resource. It's an injectable. So whenever you say someangular.module.something and it takes a function right there, it's going to be injectable. So it is injectable and here.

As long as I'm in here I might as well pair this down to the more idiomatic format. So $resource builds a function for you which is a wrapper around the RESTful resource. So this is a template for what the URL works. You got to give it a template and then you say… I don't remember what this next one is. But you can give it some kind of built-ins. So there's a built-in variant called query which uses a get and it passes in phones so the phoneId param. If you took this right here and you passed in phones for the phoneId param, you would be doing this which you should recognize as the URL to get you all the phones. So that's what this thing means. And you have to tell it whether it's going back as an array or not. There's just details with this guy. If you're really interested on how $resource works, the standard approach works, you go in here, you go to the Help, you type in the name of the thing that you want, and then you… Why is not—I used to go Paste and press Enter and it worked. Now I have to click. And you can learn all the details about how this thing works. The details aren't super important.

[05:09:18]

Kyle: The important thing though just to recognize it when you see it. So this is a really dense declarative way of saying there's a resource you want to talk to, here's the URL pattern. Here's the

- -

standard way of talking to it. This query this is just a name but it's not a built-in. I could name this foo. I'm just saying make me a shortcut called foo, and then when I want to use it, I can call it this way. And then regardless of this kind of thing, built-in it always has get and post. The standard HTTP verbs are already in the box. There should be no change in functionality. It should just work the same as it did before. Yeah. See? Worked the same as it did before. Any questions about $resource? The reason we're covering this briefly is that I don't recommend it. It's semi-obsolete and we're going to cover this stuff in-depth under promises. But still, I don't want anybody to feel lost. So questions?

[Student]

Kyle: Yeah, promises.

[Student]

Kyle: I'm promising it. Yeah, but you're going to hate it when it gets here because it's deep and complex. You guys use any promises? I already asked. Somebody said you had a place where you had to like fetch 10 URLs and use jQuery promises to wait on them. Yeah. You use promises all the time in Angular in real code. Any more questions about this code though? About this new $resource thing? I guess I should put the code up that talks about $resource. Questions about $resource.

[05:11:14] BREAK

[05:20:08] CLASS RESUMES

[05:20:56] Step 12: Applying Animations

Kyle: Sure. So if you pull up the next step which is step 12, the last step of what's built in here. This is Angular support for animation. So the interesting thing is Angular itself doesn't really do that much with animation. It just provides some hooks you need, but they're really cool hooks that make it really easy to get really nice effects. So the way CSS animation works, you trigger a CSS animation by adding a class to a DOM element. And then you write some hairy CSS to say what should happen when that DOM element is added. Angular support for that is like everything else separated. So separate files, you got to add this Angular animate file to your page. Then if we look in the app/js we will see some animations I've added with the module here. We will see those in a minute. But even if you didn't do that half of this would work. There's an animatejs file but ignore this. This is just the way of wiring up if you want to use jQuery style animations. See? It's wiring to jQuery animations.

What's interesting is how much happens without that. And we'll see that. I need to close some files. I can't show you anything because I can't. Okay. So the controller is the same as before. Did not write any code there to make the animation work. The template is the same as before. There's nothing there to make the animation work. Filter is obviously not. The app/js, it mentions loading an animation-related module but that's it. That is all it has to do.

- -

[05:22:37]

Kyle: There's no code in here to make the animations occur. The service, well that obviously doesn't need any code to make the animations occur. But over here, see it fade when I click on one. See? If I go back out to the list, as I move in and out, they fade in and out, all that.. The interesting thing is how little is needed. You enter this CSS, the apps here. Yeah here we go. They put all the animations in a separate CSS file. The main thing you need to know is enough to teach whoever cares about animation how they trigger the animations. See this .ng enter? That should be a hint that it's Angular related because it has an ng in front of it.

All you need to do is add this to your page, so you add one script tag. You get one file in your page. And now, as Angular adds and removes things to the DOM, it's going to automatically add and remove these trigger classes. Follow me? So without caring whether you use the trigger classes, it's going to just pervasively throughout whenever it manipulates the DOM, add and remove trigger classes. Well if you know that, you can just trigger off of those. So this phone-listing, that's that big long, this thing, so that class in combination with the enter class or the leave or the move, this spins as they move around, automatically triggers a CSS half-second linear animation, which is pretty slick. And here we are. So when we enter the phone, we automatically set its opacity to zero. So we make it transparent. And then the move or move active, so this is the case while it's moving, so if it has to rearrange them in their order while they're moving, it sets a move active trigger class. And then leave means that it's taking one away. So leave active is while it's moving it away, we have some opacity right? So the details don't matter but this is just using trigger classes without having to change the code at all. And that is enough to get this behavior.

So like if I sort them alphabetically, see like all that kind of behavior all happens without any code other than one file to say you want animations and then just write some CSS using totally standard ways except knowing the names of the classes that trigger things. So I think this is really cool even though I barely use it.

Here's why it's cool. If you teach it to a design person once, then they can do fairly pervasive animation without you ever writing a line, a code to support that animation. And I really like being able to make something somebody else's problem. So that's the CSS part. That's not so important but I wanted to point out that if you really do like JavaScript animations, there's a couple of hooks here and I see down and up. I don't remember which one you do for down versus up. See down here you—oh here it is, add class, remove class. So this just says when this class gets added, Angular is going to watch for that class being added. And when so, Angular is going to call this function. And then this function could call a jQuery animation. So it's basically a hook. So it's a hook provided here if you want to follow that same style of just the class triggers in animation. They provide this little bridge between that way of thinking and the jQuery animation way of thinking. Because with Angular, you're not going to be writing the line of code where you would normally make this line. Because you're not going to be writing a manual click handler whatever anyway. And so they'd give you this hook so you have a place to write it. My advice is to only use this if you really have to because these jQuery animations will not necessarily benefit from the graphic card on the computer. Whereas CSS3 animations on pretty much any modern browser actually get performed by the GPU in a computer which means they'll be fast and smooth and won't have any CPU impact and all that stuff. Any questions about this animations stuff? Do you guys even use it? I keep wanting to use it. I never get around to it. Very smooth, very easy. Sometimes it's a little bit tricky there. See like a little bit of a flash.

[05:27:52]

[Student]There is this weird issue with this example in so far as there's like an image behind these that I think point to it.

Kyle: Yeah.

[Student] It looks like the main unit just keeps uploading in the back.

Kyle: It is but then that image source gets replaced when I do this. It's almost like the image source needs to be manipulated correctly also. Like there's some wrinkle like they're not clearing or setting the source at the right time. And I'm not deep enough into it to track down the details of the animation glitch. I'm not a big animation guy. But I do know that if you have somebody who knows CSS animations well, they should be very happy with the hooks provided to where it should be very low development impact. It's a designer's problem.

[Student] So I have a question.

Kyle: Yup.

[Student] So if you go to that phone detail by your signal…

Kyle: Give me a second. I got it, okay.

[Student] Yeah up there. So like it's going ahead and creating a place sort of for all of its images and then determining which one is–.

Kyle: Yeah that's right.

[Student] So does that mean when it comes to other one, so does that mean if it's not active it will actually remove the class?

Kyle: Yes. So whenever the digest cycle, we'll talk about that later on, runs if it sees that this is no longer the case, it will have active removed from it.

[Student] Okay.

Kyle: So there's some wrinkle here that I just don't understand of what you're supposed to do to make it not do that. So again it looks like they're actually putting all the images in here so they can animate that it can have both visible at once. So you have to understand something about this. So

- -

here's the class phones. So if we looked over here, we can probably find the class phone, not phone-listing but phone. Where's phone? This one did they link the JavaScript in?

[Student]

Kyle: Yeah. It figures that the JavaScript won't be the one that has a problem. Okay so there. This code is being triggered. So remember I said, add class and remove class? So when this becomes true, and so there's a bunch of these images all sitting there.

[05:30:15]

And when that becomes true, it'll have the active class added to it which will trigger this animate up which will cause this function to run, which will immediately set the position and then it will trigger jQuery animation to change the top from 500 to zero. What's going on?

[Student] This is actually taking the thumbnail image and moving it over there.

Kyle: No it's not. It's not the thumbnail. See because they loop. The thumbnails are untouched. It's these, these class phone. Yeah sorry. I'm not a jQuery animation guru. I have to find a jQuery guru to tell you this one. I'm mostly then trying to get rid of jQuery. Use more modern things instead. But anyway, that's the important part. It's the hook you need. Here we go. Here we're saying it's a start out at zero and then animate it over. I think the default time like one second or something if you don't set it, which they didn't. Then animate it over to that time to send it, so send it up, out of the way. So I start with mine, this 500 is bringing it down. Yeah to animate, animate down , that makes sense. So as soon as this triggers, it puts it at position zero and then when it runs…. Why it's named up and down? Does anybody have any idea? Because this sort of animate down, but it starts up putting it at zero and it animates it toward minus 500 which is sending it up. So I don't know why they're named the way they are, that's not real important. Any more question of the animation stuff? It's not an area of depth here but it's worth knowing about the hooks.

[Student] One of my question is kind of like scope of all this. As far as like when we come here, you've got perks about cleaning up the code and all that. For those things, all these files as we're at the global scope right now, do you have the habit of wrapping your code in…

Kyle: This code is crap because it pollutes the global scope. So this is not merely a preference. This is objectively crap because polluting the global scope is evil and it should not be done. What we actually do is almost as evil. So if this was our file, first it would look like this. That's pretty good, that's less evil. It's not polluting the global scope. But there's actually a problem here which is that you're really supposed to say use strict. And this is a particularly irritating bit of JavaScript magic because they didn't extend the JavaScript syntax that have a way of like a pragma or something that other languages have. So then we say, if you put this magic string as the first thing all by itself, that triggers different behavior, kind of evil. But this really isn't good because use strict is actually global from this point down forever, not just to the end of this file but forever. So if I appended 25 of these together, and the 26th was some really old JavaScript that would break with use strict, this would break that code. Now I think the initial answer is you shouldn't have any code in 2014 that doesn't use use strict so that should be strictly theoretical. But the strictly correct thing to do is to wrap every one of these things and then immediately invoke function. So you do something like that and at the end of the file, you do something like that and then you call it. So that's strictly correct. And if you use JSLint, its default settings it'll line if you do anything other than this. So do you guys do this? Wrap each one in a…

[Student]

Kyle: So it's not my favorite thing but it's just sort of a consequence of the JavaScript language being how it is and eventually you get to sort of ignore it visually. They have these few extra lines at the top and one at the bottom of every file.

[05:34:31]

[Student] And then there's like a minor performance, something you have to figure. If you pass an Angular, it only has to go to that top of that function to look for the Angular object. It doesn't have to go all the way back up to the window.

Kyle: If you do what?

[Student] So when we write this, you would pass an Angular as a parameter to that function.

Kyle: So if you do that, and then you do that, then that makes Angular a local variable which is theoretically a faster lookup but I think the JavaScript jets sort of to the point today where you probably would have a hard time measuring that. The main reason I'd see people lint it is because it makes a dependency explicit. Because it's explicit that this thing relies on something defined elsewhere. Because you probably most often see it like this, right? I mean jQuery in that way. So something like the jQuery object that you probably call 400 times in a file if you're using it, that's more likely to be a measurable, meaningful thing. With Angular, I actually doubt you would ever write an Angular app where it would matter doing this. Because we usually don't type Angular 400 times in a file like you'd have with a dollar sign. Okay anymore questions what we did so far because we're about to step off of the end of their tutorial, which I think really doesn't take you very far at all right? I don't feel ready to go start writing apps right now and step into uncharted waters. Yeah?

[Student]

Kyle: The next things are we're going to work with Java, these scopes a bit more, and the fact that they're using JavaScript prototypical inheritance, everybody's going to work a little example to understand the impact with that in your code. We're going to talk about bindings just a little bit more. Though I'm thinking I've already covered that pretty well. Let me see here, the dependency injection we did pretty well. We're going to talk about modules. So we're going to rearrange these files like this pattern I think is a bad pattern. We're going to have everybody rearrange our code into a more reasonable pattern. And then if we get through those things quickly, we're going to attack promises and that will be the big. Promises will be the big topic for the end of today or the start of tomorrow. Then we have some lightweight topics mostly for tomorrow like we're going to look at a debug tool called Batarang. People have already seen. We're going to talk about that digest cycle that I've hinted about a couple more times. We're going to talk about these services and factories. I see there's kind of some oddity with how we name them and talk to them. I'm going to bring up the samples of a service and a factory. Have everybody write a service and a factory to make sure you understand the syntactical differences.

Let me see here. What else is on tomorrow? Tomorrow we're going to talk about Angular has an invent system, we're going to talk about a little bit. We're going to talk about cores which comes into play if you ever host your app at a different place to where you host your services. We're going to talk about REST Angular that I already mentioned. And then the big one for tomorrow is directives which are like the big topic in Angular.

[05:37:44]

Kyle: And then on the third day, we're going to talk about like dependency management, retrieving these dependencies automatically. We're going to dig into Karma for unit testing. If the time goes really well, we'll do a little bit with Protractor for end-to-end testing. And then we have a variety of kind of miscellaneous topics that will end up using up the rest of Friday. And then throughout these things, somewhere around the end of today, start of tomorrow kind of everyone will start building your own little app to try us to illustrate all these ideas that we're hitting. Is that a pretty good roadmap? Okay. Well the next topic is the Angular scope stuff or the scope inheritance stuff we probably should take at the five-minute break, maybe a little less just because we've been talking for many minutes straight. So let's take like a three-minute break and just stop talking for a moment.

[Student]

Kyle: Well if you want to get started, what we'll do is these are going to be examples that I'm going to make little examples and you're going to make your own little example. So what I'm going to do is I'm going to just go back to one of early steps of this repo like steps zero to one and copy that. And then take the old get stuff out of it to make my own new repo. I feel like everybody learns better when I do it that way. I could grab one of the… I've kept three past efforts [05:39:12 inaudible] of the ones I haven't kept. But I think it's more educational to just kind of type it on the fly.

[Student] You could do a lot of troubleshooting with that way also?

Kyle: Yeah the idea is to hit problems that other people are hitting.

[Student] So if I go to step one and just strip it down a little bit.

Kyle: Yeah. That's what I do. I'll just show that.

[Student] It's a static HTML.

Kyle: Yeah. Not much in there but it really probably matters.

[05:39:44] BREAK

[05:43:00] CLASS RESUMES

Kyle: So, Paul, you think 2 is the best starting point? It's not as hideous as 1.

Paul: Well, 1 there's nothing there.

Kyle: Yeah. So what I'm going to do here is I'm going to take the contents of phonecat and then I'm going to paste them into my starting point which I'm going to call K4 this time. And then I just happen to know there's a bunch of stuff in here I don't care about. I don't really care about package json under Read Me. I don't care about [05:43:42 inaudible] And then here, I only actually care about the web server. And then here, this stuff is fine as it is. And go over there. I'm going to separate it with old get history that I might have brought over than to do its own get on it. And then I'm going to go back here and I want to kill this thing and go run that same command in my program instead.

[Student]

Kyle: Say again?

[05:44:35]

[Student]

Kyle: So what I did is I.... See if I can find the command. So I did a check out to step two because Paul is pointing out that's probably the best place to start from. It's like the closest to a kind of clean empty…

[Student]

Kyle: Say again?

[Student]

Kyle: Yeah right because as I get check out -f means four so ignore wherever I'm now and get me to that step two. And then I went over to my finder to which you do something different on Windows, and I copied the contents of that Angular phonecat into this new thing I made that I called K4. And then I deleted a bunch of stuff that I know that I don't need. And then I like to show kind of good practices of keeping your history. So I don't just start randomly edit a thing without having it under source control so I get and edit it. And this is how you see what files would be added. And it's probably a reasonably good set of files. That lib stuff I know that I'm going to replace it later. So just to show a good practice here, I'm going to make a get ignore file. You don't have to do this. You can just delete it later but I know that I don't want app lib with source control because tomorrow I'm going to use something called bower to fetch my libs instead of checking them in manually because it's 2014. And so if I did this, it'll give me a somewhat shorter list of stuff I'm going to add. I won't seem to want app lib in there. I'm bringing all this phone stuff along because it's nice just to have some data to work with and I'd rather just see some data have handy than have to go type my own json. So I can do something like this. So now I'm going to actually add those to get.

Step two, something like that. So now I have a starting point committed. See that as I start doing stuff, I'll be able to go back than make a horrible error. So if I refresh the page here, this is kind of where we left off of step two. Now I want to start pairing things down a bit further. See here, in the libraries, what I'm going to do is I think I'm going to keep just this controller stuff. I'm going to put all my stuff in this app file because I find it easier to illustrate what's going on with it all in one file. So I'm just going to delete these other empty files because I don't need them. So then if I go in here, I can probably get by with just one use strict. And then I'll put this in a more idiomatic form. I'm not going to use this index async thing, that's irrelevant. I don't think I'm using this [05:47:47 inaudible] either. And on index, I only want to load appjs. And you guys, you don't have to make these same changes. I'm just making some changes because I know that I kind of want to clean it up a bit. But if you're just learning, don't really worry about cleaning it up a bit. Just try to understand the new concepts going in. You can clean up once you understand more.

This is pretty good but I think I had mentioned that it's really better at least through the last five years to do this down here. Make sure that really works. I think it's actually bogus to have these minified files checked in. So while I'm here I'm going to start a couple of minified things and remove them. Again it doesn't matter. I'm just doing clean-up to illustrate what can be done. I'm not using these modules thing, I might as well delete it. I'll make real partial when I need it, so I'll kill that.

[05:48:53]

And this is all just data that's handy to have around to play with. Say is anybody here a heavy get command line user? Nobody? Not a single get? What do you guys use?

[Student]

Kyle: I know you use get. What do you use? Somebody ask me what I use and said sourcetree.

[Student]

Kyle: You let IntelliJ do it?

[Student] Command line.

Kyle: Command line.

[Student]

- -

Kyle: So like you start get status to run that command. So this is all the stuff that I deleted. When I use sourcetree, let's see if I can get sourcetree to come up here in a nice way. It's kind of funny I have a previous version of this in there. I got to get this into my sourcetree somehow in a second here. There we go. So I would use sourcetree if I wanted to nontrivial commit just because it's easier. So yeah this is all the stuff I want to get in there. I hit one button to get it all staged. And then I can just do a commit here. It's a nice tool.

So the first thing I wanted to illustrate is how these scopes nest. So I think that all the examples we saw here is it only had one controller. Or they had two controllers but they were used for entirely different screens. So if you're building nontrivial apps, you might ask, "What happens if I want to use more than one controller on a screen?" And that's a very good question. So what I will do is I will take this existing junk here and I'll just put this in a div. So now I have scoped this controller down to something a little tighter. So on this I'm trying to make a hole here to cut my own stuff. So I want to be able to do my own stuff here and I want it to be obvious where's my stuff so when I use HR, it's like not a bold enough line, so I'm just going to do a pre-formatted row of arrows. It's pretty low-brow, right? Okay so above there, I'm going to type my stuff and I'm just going to leave this existing controller running below. Everybody follow me so far? So the question was, how does it work if you have more than one controller on a page? So let's see. If you misspell it, nothing happens. Here, try to keep things short. Short names, easy to type. So that's a controller. And then that's another controller. And then while I'm at it here, let's just put another copy of that controller. That seems like a nice little example.

Now what's going to happen when I try to run this? Anybody know? Yep. Not going to work. So I need to make this controller exist. So where would I do that? I only have one JavaScript source file, so I have to do it in that file. So this is the file where I'll do. I don't really like PhoneCatApp. I'm going to name it kyleApp which means that the change up here to point to the same thing. And then after that controller, I'm going to make another controller, and I think I called it acr? Then after that I get a function where I can do dependency injection. I'm going to do nothing at all because it's completely valid to have a controller that doesn't do everything. And now I can just easily add another one that doesn't do anything called b. Type this right. Bcr. This is not any kind of official naming convention. I'm just typing something short.

[05:53:39]

Kyle: So this is not expected. This thing should be up at the top. I'm trying to figure out why it's down below like that. Anybody know how I get this to be up there? I guess I just click on it. Yeah, there we go. PhoneCatApp is not available. I must have done something wrong. PhoneCatApp is not available, so what is going on there? kyleApp, okay. Oh, I didn't save it. See this little dot right there? I didn't save my index. So I got to save it before I reload it. There. So this is all the stuff below. That's my nice visible line. So my controller has done everything. This is a completely expected result.

I could do something like that. So this foo [05:54:44 inaudible] does not say anything. So you guys probably know enough to know what I type here. What do I type here? So that should work. Why did I get that? Right. It will only dependency inject what you give it. So the reason why it can't be implicit is that this is just JavaScript. This isn't like some language that gets compiled to JavaScript, and JavaScript does not have the ability to have like some function magically get past an implicit variable. So the only thing they could do is make a way where it can pass it if you give a place for it to land.

A reasonable question to ask at this point is how many [05:55:45 inaudible] are there? Is that one variable twice or two variables once? I can tell you it's two variables once, but I'd like you to have like a way of verifying and understanding definitively. And the answer there is… I can just go in her and put an input. I can do something like h3. I can make it obvious this is an acr. And then I could do something like an input. Does everybody remember what the model? Yeah. So if I do this, it should be obvious that they're two different ones because you can change them separately. Everything makes sense so far?

So when you say ng-controller, that means make a new scope here and call this controller to initialize that scope. The reason I try to keep my line short is because I click the right button here on this editor, I can do this and I can put these side by side. So I try to keep my line short. I don't know if I can do that.

The first question I asked is can I use more than one controller on the screen? Yes. Can I use the same controller more than once? Yes. Guess I had to go ahead to show that I can initialize it differently here. What happens? Scope does not define. Yeah. Sure enough.

So at this point, it's time for everybody to start making a little example like this. Paul and I will help you get through the little wrinkles. You don't need to do what I did. You can certainly type less than I did because I perhaps type excessively on this page. You can make it look however you want. You can make it have whatever data you want but just get some little example working where you put a couple of controllers next to each other and do some trivial bit of data binding just to kind of go through the motions and making it all work.

Once we get there, then we're going to start nesting them inside each other and get a hands-on understanding of how they all put together. Since I like keeping track of my work, I'm going to commit this as… I'm going to try to minimize my blank lines to keep this brief along the page.

[05:59:40] WORKSHOP

Kyle: Yeah. You can use two controllers. You can put one controller on other [inaudible]. You can do both like I did. You can just data bind an input or you can use some other data binding. Any variation. You can type through a filter. Just trying to get through everybody manipulating the code.

[06:12:57] LECTURE RESUMES

- -

Kyle: So is everybody up and running with the simple controller example because I'm ready to go on. We've already seen that these two side by side are separate. That's not really telling us much. So I'm going to simplify my example down by mixing that. But what I really want to get an understanding of is what happens when you put one inside the other. So I'm just going to clean up my indenting a little bit to make it obvious. So I have an A but I have a B inside of it. And over here for B, I'm just going to not set this at all. So that code is gone. What happened here? I didn't save. And save. Pain in the butt. It saved explicitly. There's no one from A, there's no one from B. And it looks like I hadn't labeled my B so I will label my B. Okay so we see how that name came through to both of them. See that? Inside B, the name is visible even though we did not set it in B. The reason why that works is when you have scopes nested, and in Angular you will always have lots of scopes nested, the inner scopes have Java prototypical inheritance from the outer scopes. That's the relationship. Its scopes are plain old JavaScript objects that have—I misspoke. They have JavaScript prototypical inheritance from outer scopes to inner scopes. So I had a raise of hands at the beginning on the number of people who've written at least a thousand lines of JavaScript. So somebody tell me something about JavaScript prototypical inheritance. Anything. Anyone tell me anything.

[Student] Hurts.

Kyle: It hurts? That's actually a pretty good thing. Anything else about Java prototypical inheritance? No one will answer or comment.

[Student] Scared.

Kyle: You should be because it's endless unpleasantness. [talking about clips] JavaScript has prototypical inheritance or prototype inheritance. This came from a language I think called Silk that Brendan Eich was looking at as he was making JavaScript. The way it works is let's say you had some parent object. And you have some child object. There's a prototype relationship. It's not inheritance the way you think of it in JavaScript. A prototype relationship is something between one object instance and another object instance. Can anyone tell me why there's no such thing as class to class relationships in JavaScript?

[Student]

Kyle: Exactly. They are only objects and so there are only relationship between objects. The way this works is the read and write paths are different. That is a source of endless stumbling, we will see. If the parent has a property X that equal 12 and you have a reference to the child and you say C.X, you ask what it is, you're going to get 12 because the read path goes that way. But there's a write path which does not go that way. The write path goes here. So if I say C.X=13, then I am writing an X property of this to be 13.

[06:17:17]

- -

Kyle: I think you had this exact issue that reading a property walks up the prototypical inheritance chain. I just have this written too deep but it could be arbitrarily deep. But writing always writes locally. I don't know if this is necessarily a great design feature of JavaScript but this is just how JavaScript works and Angular is pretty explicitly not trying to fix JavaScript. It's trying to just use JavaScript and so it does not attempt to fix this even though it's a source of annoyance. I now want to show you how to experience that annoyance.

[Student] So subsequent reads will give me 13.

Kyle: Yes they will.

[Student] So it will stop going up in this way?

Kyle: Yup. You wouldn't like it if there was a nice parallelism between reading and writing. I would like it but did not. You would like it if it wants you to have some path working if it keeps working that way consistently. But in fact this read path it'll work one way until anybody anywhere calls a write on X. And then all future reads will read that one instead.

[Student] So are you supposed to write to a local object and then read that?

Kyle: We'll see. I'll show you how you're supposed to use Angular in a way that makes you not stumble off of this. But first we need to stumble off of the hold mark. Okay so I have these two nesting things here. And again, the D, this is really just the placeholder. I'm just using this to make a new scope if you look at this. Because I'm not setting anything in the scope. I'm just kind of keeping this handy because it will be handy. They'll make a new scope. Now input an input here so. So is everybody reasonably convinced that they understand the relationship between the input and the data binding? Any question at all on that? I can edit type this and that changes, right? Because we understand well, we can now save some screen space and make sure we can keep making things big on the big screen. I can just start pairing those out and I could put one of these down here. And while I'm at it, since I've now saved some space, it would be kind of nice put to a second one of these so they're all binding to name.

The thing that I just described which is this kind of an abstract description, the cool thing is that you can not immediately play with that because if I change this to another name, it has the right effect you expect. And A, it's writing to this but then B, so the two different bcr's it's going to read path on that name operation. It's not calling right. So it's never setting anything on B scopes. It's only reading from the parent scope. But soon as you make any change at all to one of these, you notice the change didn't go up? Can you see from this chart why it didn't go up? I have now broken the relationship between this one and the others. Now I haven't broken the other part of the relationship. See? That still worked. But I've broken this much. And now if I type down here, I've broken that part of the relationship too and no further changes up here will have any effect on there. So the solution to this problem is to realize that this code is really wrong, find some things really fundamentally wrong. Before I show the solution, I like to commit the problem. Somebody's going to ask me to see the difference, and if I have it in source control, I can easily show you the difference. Show proto error problem.

[06:21:38]

Kyle: So the solution is, in Angular to use it correctly and idiomatically, you should not use the scope as your model. You should use the scope as a place to put your model. So right here, I'm using the scope as my model because my model is an object, has a property called name which is then. Shouldn't do that. What you actually should do is make this be an object which is the model. So now name is no longer a primitive sitting directly on the scope. It's an object which is a model object sitting on the scope. And then you need to bind correctly. So sometimes you see this referred to on the mailing list as the .rule and that rule is you shouldn't see bindings that have something without a dot. So when I had something without a dot before, that's kind of a red flag that I'm using the scope as my model instead of using the scope as a place to hook my or store my model. If you follow the .rule, you'll always have at least one dot. So this will work and I'll show you why in a moment.

So with this I now can make changes wherever and it does not trigger this problem. The reason why is that when I have name, so name.first, the .name part is the read path that reach up there. And then it's writing something hanging off of that. So let's see how the name.first is not going to write to this inner scope. .name is going to go find an object up here and you go change its contents. So if you always follow the so-called .rule, just put the model on the scope. Don't make the scope be the model. Then you will dodge this idiosyncrasy of JavaScript. And you will hit this problem because real apps have many, many nested scopes all over. So the next exercise is for everyone to create this problem, fix it and we're going to understand, tweak it a little bit, how you manage your skills so you get an understanding how this works.

[06:24:20] WORKSHOP

[06:36:24] LECTURE RESUMES

Kyle: Yeah. I'll point that out. That's a good…

[Student]

Kyle: I think she has a question. I don't remember your name. I'm sorry. What's her name? Shila? I'm going to write that down. I'm terrible with names. It's harder to be good with names with someone who is not physically present. Shila, probably spelled wrong but it's pronounced right. So what was the question on here?

[Student]

Kyle: Okay. So this is the JavaScript syntax for a literal object. So I said scope.name is this object. So we have to go back a step. Do you understand the difference between a value type and a reference type? You guys are all Java people, right? So in Java, you make a class force whatever,

- -

and I say horse force= new horse. That's the JavaScript way of talking. So the variable horse is a reference to an instance of horse, right?

[Student]

Kyle: In JavaScript, a string might be an object. But the important thing is it's a value type. If I pass you a reference type, right, you have a reference write it'll both talk to the same object. If I pass you a value type, you just get the value. You don't get access to what I had. So this is essentially adding a layer of indirection and saying scope.name, this is a reference to an object out there which happens to have a field first.

[06:38:07]

[Student]

Kyle: Yeah. I certainly could. But if I only read scope.name and I write [inaudible] inside it, now I'll never break this association as what I have drawn here on the board.

Paul: So another way to look at it is we've declared the object exists up here if they're a parent. Whenever we set a property of that object either here or here, we're still looking at the same object. We're not just manipulating the properties contained within that object that exists on the parent scope.

Kyle: Is that clear enough? So you're saying what if I just wanted to get a hold of the parent and do something to it? So what if I went right here and I had just like a last. And what if I had done scope. last= and you're saying what if I wanted to go down here and just reach up and change that, right? So I think there is a thing on scope called parent. I don't do this very often so this might or might not be the right way to implement it. It might take me a couple of tries. Where did it go? I didn't save. See? Not changed. These scope objects actually have a field in them called $parent. The $ should be your hint that it's an Angularism. So if you really want to get a reference one up, you can. It's not really recommended but you can. Any more questions on this whole prototype inheritance thing?

[06:40:34] WORKSHOP

[06:45:24] LECTURE RESUMES

Kyle: Next, I'm going to make sure I check this in. So what did I do most recently? I fixed the problem. There we go. This is kind of a bit of an ugly code. Put the model on the scope rather than using the scope as the model. So the next little thing that's worth doing, this is a topic where we turn to repeatedly is this whole module. Let's say this module has a name. Angular does not impose any rule on you about how you name your modules or about how you locate them in the file system. But what I like to do is I like to do a slight bit of a Java-ism here. I like to give the names like this, names that imply nesting. And what I'll do is I say if I have a module named kyle.app, then the place that I want to put that is in a folder called kyle in a file called app. So I guess I got to use the different tool to use it. So pardon my move over here on a different screen. So see I moved it inside there? What I like to do is say if a module has a dotted name, then all but the last one are folder names, and the last one is a file name .js. To make this work, I would have to do the right thing down here. So again, Angular does not require that, but we've got in the habit of doing that on all of our projects because that way there is an unambiguous question. An Angular module is one file, so I will not put more than one module in the file in real programs. The name of that module, if I have a big app, it will be sort of namespace with dots like Java, and I will put them in files that correspond to those names.

[06:46:34]

We do that across various projects, spell the same standard across all of them. We can have a whole bunch of files, a whole bunch of modules, and everyone can immediately understand where they should be and where to put them. I don't know if this is necessarily the best way. I think if you follow the mailing list, you'll probably find people talk about three or four different ways to do it. But this is at least one of them but it makes some degree of sense especially coming from a Java shop certainly since this is a very Java-like pattern.

One thing to point out that's not exactly Java-like though is that an Angular module is really more akin to a Java package than a Java class. So a Java class is kind of a smaller thing and an Angular module is more of a bigger thing like a package. You can think of these things as maybe being a little bit more akin to Java class although there are certainly plenty of differences. So I wonder if this works. I love making changes and not knowing if they'll work. Oh, look at that. Up here where I look for the name of the app, that needs to match the name of the Angular module. So now it works.

[Student]

Kyle: Yeah. If you think the code might like you have multiple apps kind of sharing the same file system, then yeah you could use the first thing as the name of the app, or not if you don't. We typically don't because we typically have different projects on all different repos like there's not really new room for them to conflict. But yeah, like if I have an address book part of an app, I might first name the module address book and put all the controllers I need and all the filters I need, whatever for address book I'd land in there. Now if it started to get too long, I just think about modularity like in the most general way. There's really nothing JavaScript-specific about this. How would I make this code more modular? Like what is it reasonable to divide this big glob of stuff? What smaller globs make sense? And I give each of those globs a name and I'd name space them under address book and I'd cut and paste the chunks of code into them. Make the module names match and check it in. And I'll probably get a chance to do some small version of that as one of these examples grows a little bit. But for the moment, it's really so insignificant that I don't have any.

- -

This is fine. it's like I'd argue that this phone stuff is kind of arbitrary. So if I just wanted to do that, I could go in here and I could say phone.js. That content becomes that. That can go up here. I could copy this into phone. And then I could say I only want that part in phone. Just a tiny bit of clean-up like that. Then I go over here and I don't want this one in this guy anymore. Got that. Now I think both of these are correct. But this is like a separate question if we're given actually load it on to the page, you have to get them loaded on to the page like this. I'm missing one step.

Can anybody guess what it is before I make the error appear? What? Model? Yup, dependency. You got it. So it's going to break because this PhoneList thing has not been defined. The reason the PhoneList has not been defined is because I only have an implication that ng-app to load this module. Nothing is actually telling it to get this module and memory. And the answer to that is actually declare the dependency. I'm going to use kyleApp as my top level for my application. And I'm going to have it invoke dependency on kylePhone so this should come back to life now without error and it did. And again I like many small commits. So I will commit this. That's weird. Did I not commit the previous one? Was it? Oh I actually did not commit the one where I renamed it. I'm going to go ahead and finish that commit. Rename module and move file. Finish the job there and then this next batch of changes, I get that in. This would be split out phone module. There we go. Everybody follow that?

[06:52:23]

That's a sort of operation. I'd done that in Java before. Package gets too big. Just split it. The cool thing is in Java, if you want to use the tools, you have all those refactoring tools available. You have tools that understand your source codes when you move things around. It makes all the right updates. We don't have any of that for JavaScript. And I even hesitate to say yet because it's a much harder problem in JavaScript. Because being so utterly dynamic, there's nowhere near as many hints in the language in JavaScript to help tooling do that. So far we're still mostly on our own if we want to do those move-arounds and cleaning things up unlike the Java world. That will be unpleasant as a project grows.

The next major topic is promises. But I think before I jump into that I will show one minor topic. I don't do a lot with this. I just like to point it out to each group because it's long and tedious. I just got to figure out which help topic is the good one. We don't use this stuff a whole lot. A lot of people don't use it a whole lot but you should be aware Angular has a bunch of stuff to help you with forms. Where's my example? Go to form directory. They change this so often. Forms, yes there it is. So there's a bunch of stuff with forms. This is really analogous to the animation stuff. It adds a bunch of dollar sign things to HTML for you. So if you have a bunch of forms, it's too much to dig in here now because it's very detail-oriented but if you have much forms, spend all day trying to understand this page and you will find a bunch of useful things in there.

So for example, form is the whole form. Size is long-control on the form. Dollar sign error, that's an Angularism for errors to pertain to the size. And this apparently, this thing is testing integer-ness and so this is what you get. You see this ng-show thing? This guy here becomes true if this is the kind of error that's occurring. And so you get these nice little… Is this page not working? They usually just demonstrate this right on the back page. This model equals sum. The documentation for Angular is currently broken. See how that size thing is appearing? Let's see if we can get to a non-broken version of the documentation. Let's see. It's pretty cool because the documentation is versioned. Where is that validation bit? Yeah there we go. See how it's green if I type something outside the range, it turns red. You get something to appear. See these little bits there. I'm not a guru of this stuff but I can point you in the right direction.

They have a way of testing for integers to regular expression. And then they have this $setvalidity thing that wires in to HTML5 form validity. HTML5 has this notion of whether the form is valid and Angular wires into it and enhances all that thoroughly. So the only code here is this notion of what it means to be an integer or a float. So all this stuff about the validity this is all just sort of automatic things that are being set behind the scenes and you can use ng-show. Ng-show says only show this if this expression is true. If you want to see that, we could go on to anything here at the ptag and say ng-show some expression. This ng-show works. If I type in a Janet here, it is okay up here. See that? So ng-show automatically shows and hides elements. There's magic going on behind the scenes. Like I said I don't use it all but I just like to give you like a really fluent explanation of it. But if you're doing forms heavily, you're going to dig in to this. It's a bunch of hairy messy stuff. But it's way easier than coding it one bit at a time running your own. How many people have implanted their own way of doing validations with jQuery? Only two of us? I think I've probably done it twice so I'll put both hands up. That's a little boring topic. We'll take like a five-minute break.

[06:57:58]

[Student] I have a request. Is it possible to cover the promises tomorrow morning?

Kyle: Well, so nobody misses it you mean?

[Student]

Kyle: Okay let me see if there's something else I could cover first.

[Student] You mentioned that there were some miscellaneous.

Kyle: Yeah what I can do is I can talk about the Batarang debug tool, go over some digest cycle stuff and talk about services and factories. And maybe even events and that would take us to the end of today. And start on promises in the morning. Just two of you have a…

[Student] That'd be great.

Kyle: Okay. We'll do that. We'll take a five-minute break and we'll dig in to those. We'll fit in four unimportant miscellaneous topics by comparison. And we'll jump into the big hairy difficult one at nine o'clock sharp. Does that help?

- -

[Student] Thanks, appreciate it.

Kyle: I'm going to talk about the Batarang debug tool, the digest cycle, that service and factory stuff that I hinted at and the Angular event handling system.

[Student]

Kyle: Batarang.

[Student] It's what Batman throws.

Kyle: Batarang, Angular Batarang. I'm trying to get a large print for you. It's a Chrome plug-in called Batarang. Anyway, in five minutes we'll talk about it.

[07:00] BREAK

[07:09] CLASS RESUMES

Kyle: Okay does everybody have this in? I'm going to assume that's a yes. So the cool thing about Batarang is if you're in any Angular apps, so any page, then… I'm going to put this one in. I'm going to put my dev tools in the bottom. You get this Angular.js tab, which you click enable if you haven't already, and you get some tools added to your dev tools that are Angular aware. The slick thing is that here are all your scopes. Remember we talked about their scopes and they're in a nested relationship. You're like a click or two away from seeing the nested relationship. This is the so-called root scope. I can click here to see one of the scopes I have inside it and sure enough there's that name and that last thing. Remember that how we're putting stuff in the scope, he was like a click or two away from just interactively inspecting your scopes and I find that very useful.

[07:09:54]

Kyle: Something that's not immediately obvious is that many, many directives in Angular created scope. See how we have these 4 and 5? Have to look and guess what they are, but I would guess that this ng-model made a scope and that this last made its scope. So even though there's nothing set in the scope, generally speaking everything in Angular makes another scope. That initially sounds kind of scary like oh no a lot of scopes. But an Angular scope is just a JavaScript object that prototypically inherits a bunch of stuff. And so practically speaking it's really close to free. It's like the overhead per scope is like what's the overhead of one Java object for example? By the way what apps do you guys use in web, in Java?

[Student] JBoss.

Kyle: JBoss. Okay, it's not quite as bad. In the early days of app servers, I was using WebSphere. We were curious. We pointed a profiler at WebSphere. Why does this thing take so long to initialize? From the time you start WebSphere, but before you deploy your own application, I would like to hear guesses of how many Java objects are created by WebSphere? So it's initializing, it's reading some XML, and that creates some, right? It's doing everything it does. Guesses.

[Student] 10,000

Kyle: Guesses.

[Student] Two million.

Kyle: How many?

[Student] Ten million.

Kyle: Ten million, okay. You're a lot closer. It's about a million objects. So it's about a million Java objects created by at least as a 15-year old version of WebSphere, late 90's I was fighting it.

Student: How are you actually fighting that? Logically that's not close…

Kyle: Because it was one order of magnitude away. Because in the most general sense, the scale of our universe is so vast that things were sentences to think logarithmically than linear. So that's why it's closer. I'm actually pretty impressed. Somebody got within the factor of 10, usually I ask that and nobody gets in the factor of a hundred. It's just mind-boggling that it'll be so vast. Those scopes don't matter. They're just a single JavaScript object. You could create a million of them and you probably wouldn't even notice. Here's the scope for the phones thing and then those inner scopes will be for one specific phone. Remember when I first said—actually I think Paul first pointed this out, how this ng-repeat works. It's actually making a scope for each one and it's putting, I think I call this a local, it's putting something on each scope that he makes called this. So that when you do a binding, it finds that on the scope. And the call thing is you can just see it's visually apparent what is going on right here. It's a very cool piece of functionality. It's one of the advantages that Angular has from its momentum. So there's like someone going through the effort of making this work. I'm not aware of a tool like this for like Ember, just competing library.

[Student] One thing I noticed is if in your controller you put something on the scope, it shows up here but I think your page, the page generates something on the scope, it doesn't show up.

[07:13:05]

Kyle: So you're saying if I go to controller B, and I put another input.

[Student] Yeah just like . . . .

Kyle: This will illustrate something interesting actually. If I go here and I type some last name here, question is where would I have to look? So there's last and that's a different last. Which is which? That's the root. That's A and that's B but where's B. How do I tell it? B is 6? No, no 6 is the phone. This is really hideous. How do you tell which is which?

[Student]I was just about to ask, when you're going to put the controller that created it.

Kyle: That sure would be a great information to have, wouldn't it? It must be one of these. I guess the question is if we find name, how come this didn't land in it? And that's a great question. I don't know. I don't know what triggers this thing refreshing. Say again?

[Student] In my app, I did name and age. Name is generated by a controller but age is auto-generated.

Kyle: Here's a question, what updates show up? So what if I did like this? If I did this, that's there. If I type up here, it's not updating down there. Even if I go back and forth. So I do not know that.

[Student]

Kyle: See how advanced? It's possible I have an older and newer version of the Batarang too. I don't know the detail. I don't know what the wrinkle is. I usually find the problem I'm having. Let me put it this way. Mine is usually broken before I type anything so I actually never hit this problem. My problems are so apparent, that you don't even have to type to activate them. There's already there. So a couple of things we're seeing. This inspector is kind of painful but if you trust the right button on it, see how it's highlighting, it's actually letting me, it's like showing me what the scopes look like. It's like letting the pointer on the screen. If you have complex page, with a whole bunch of stuff on it, you can't figure out how does this tie in to my Angular. And this little tool it's a really rough little tool. But it is better than nothing.

[Student] Can you bring that up again?

Kyle: Yeah.

[Student] I think they should update this. They tell you to select it.

Kyle: Yeah I don't know why it says that because I–.

[Student] When you press that, you highlight the B one. Okay so it's the scope. For a second there I thought this has no scope but it does. It's just the model.

Kyle: Yeah so the actual model goes up here and see that name. And then if I type in here. See if it updates. No it doesn't update up there either. Again the way I use this tool, I generally hit that problem and Ember doesn't have it at all. So that's where [07:16:24 inaudible] but that's just the first. There's a bunch of stuff in here. This performance thing, if we had a bunch of watches—and we'll talk about watches next—you would see things appear here to see how much of your total compute time is going to them.

[07:16:38]

Kyle: So this thing there's not a lot of time. This is not a very high overhead application but if it was, I might be alarmed to find that merely watching this expression is taking 22% of my total watching time. The reason I'm not concerned is that, it's still only taken 0.7 of a millisecond to do that. Although that seems like kind of a long time for checking one variable. This dependency thing, this is just bizarre and I guess I need to bring up a real app and see this although this usually populates for me. Something is wrong with my browser because this isn't working either. That's just bizarre. By the way if you turn that show scopes and bindings, then it will statically show a bunch of stuff. So it highlights it. That kind of adds a bunch of CSS to show how the Angular structure of your page looks like.

[Student] Unicorn zapper.

Kyle: Unicorn zapper?

[Student] The Angular version of it.

Kyle: Yeah they've been doing this thing where they add whimsical short names to each version. So this Help I have not found helpful but maybe there's something in here that tells me why the things that I expect to work don't. Anyways here's what it looks like. that dependency thing works, it just draws a line between all your modules and how they depend on each other. I only have two modules. But that's still enough to put them next to each other and draw a line. I have never found any practical use for this other than that, I guess if you wanted to express the complex application, you could screen shot this and put it in a report. The thing that I really use all the time though is this one. This is really useful. Let's just click on your scopes. It's not useful at all when you have seven of them because I can just figure out looking at the code. But all these features in Angular compose really well so it's really easy once you learn the basic tools to apply them again and again and again and again, and you have a whole bunch of controllers, and you have a whole bunch of scopes, and a whole bunch of models and a whole bunch of parts on your screen and you can get where you're like starting at it trying to figure out which one goes in which bit of code. And this usually answers it immediately.

Next bit, the digest cycle. I've been closing the window, I shouldn't. I closed the window of my own application. Here we go. So the digest cycle. Let's. So how do these things really work? How is it that when I type another letter there, all that other stuff updates. The answer to that, it's actually kind of unpleasant. You will not be pleased when you first hear the answer. The answer is every time you change anything, it checks everything to see if anything else changed. And in fact, it checks everything twice just in case checking made anything else change. So if you have 500 things data-bound, that will check all 500 things to see if any of them changed. So let's start illustrating that.

[Student]

Kyle: We'll see in a minute what the logic here. It's not based on a tree. It's based on trying it again and again until you get the same answer twice. Let me see what bit I can use to illustrate. The glue or that the mechanism underneath here is called a watch. I could go inside some handy controller like this one so I'm going to do this right here. Work as well as any other place. And I have a scope so anytime you have a scope, you think to yourself, what is it doing to make this data binding work?

- -

[07:21:02]

What it's doing is this. Internally it runs a code like this scope watch. And just to make sure that we're doing the same thing, so this chunk right here is internally doing scope watch of that and then it takes a function. Paul, I always forget, is it new old or old new? I think last time I guessed old new and I was wrong, so maybe it's new old. Internally it's doing this, and if the value of this thing changes Angular will call a function. Now my function is going to count as a log because I want to show you guys how it works, but what it's really doing behind the scenes is it's wiring up a function here that uses its own jQuery-like code to update the dom. So what's really happening is one of these is being created, and if this expression changes, it updates the DOM. So let's see this run. I need to hit Save to do this right, to go over here. I think we figured out that it's easier on a big screen if I put it on the right. [07:22:36 inaudible] So if I type something else, anybody care to guess why it's printing each one twice? What?

[Student] You say that it's checking twice.

Kyle: Yeah why do you think it's—actually it's not that. That's kind of confusing. The reason it's doing it twice is that where I put this code is in the controller B. So this gets called to initialize the scope that I use the controller B twice. So that that called twice. If I wanted to pare that twice down to once, I could watch this from up in A. Like that. So if I do that, then it will only do it once each time I change it. That part's pretty easy. What is this string? Angular has a service? So then all these things are often called services if they do something. Angular has a service called Compile. Compile is a slick little service. Remember I talked about Angular expressions? Compile compiles Angular expressions into JavaScript functions. So when I wrote this shorthand form, behind the scenes, it wrote something like this. It compiled that into a function, and the function is something like turn something. I think I typed that right. And it compiles it into something like that. Let's see that that really works. We should see them happen twice now. See how they happened twice? Because it's [07:24:50 inaudible] for the built-in one.

The reason I said something like this is really important to understand, Angular functions are super tolerant like HTML which means that the actual function has a bunch of safety checks to where if name was not defined, it would not throw a JavaScript error trying to look up first in the name that wasn't even defined. So the actual function is more complicated than what I put here but it's also safer and it's less likely to make your app start spewing errors. That's why if you're just doing something simple, then this is the preferred form but if you're doing something complex, where you need to write some function, you want to watch something more complex, you would do this. You can write your own function.

So another little variation on that. Watch this. Let's watch the name. What's going on there? By default, scope watch only watches this thing. It doesn't dig down and look inside it. Because it's more computationally expensive to dig down and look inside it. You have to explicitly ask. There's an optional extra parameter there to say I actually want you to dig down inside the thing that you're watching. And now each time I type this, you can tell that it's writing some object. We could inspect that if we wanted to. So if you have a watch that doesn't fire anything though it's pointing in an object. It's an optimization. I'm not sure that they made the right choice because I find that I have to type true almost every time I explicitly type my own watch. Because I'm usually doing a watch because I want to watch something complicated. To watch complicated things, I almost always have to look down inside them.

[07:26:45]

Let's play with watches a little bit more. At least with this functionality a little bit more. Here we go. Let me put something a little more interesting on the scope here. Let's say that last is not merely Smith but last is a function which return Smith. Yeah I guess I'm data-binding to last a few places. That's kind of an ugly way to do that. We'll go ahead and make last be Smith. It's a function that returns the age. I could data-bind to that so I could say that this is all an A, so I can just ignore all the rest of that. So I could maybe go in here and p, age. If I just did this far, this would work. By work, I mean it will not have an error. If I go in here, see my age, it doesn't actually say anything. That's because it does not automatically invoke functions. If you want to bind the value of calling a function, you got to be back. Okay. That works as expected. But if you want to make it obvious what's going on, what you need to do is create some side effective.

So now that we have a side effect, we'll be able to see. Do you guys notice anything? Yeah even though it's never changing, and it never even can change. It's still being checked repeatedly. If you put functions, and you plan to bind to their output, make them be functions that don't do a whole lot. You certainly wouldn't want to try to do some significant amount of computation there. If you want to do some significant computation you would watch the input that drives the computation and then only if the input changes would you calculate the output. So that's because Angular has no visibility inside the JavaScript language and it's library not a language little feature. So it has no way of knowing what this function might depend on. It knows no parameters are passed but in here I can write literally anything in here. I could have depended on any piece of data anywhere in the application I could have depended on from here. So with every change, it's getting checked repeatedly.

What if we made it actually return something different, right? We can probably do that. I could just make some var here. What if I made it return that and we know this would work, right? We'll just get a zero. See? But what if instead we actually counted how many times? So now we have a function which will return a different value every time it's called. You're not supposed to do this. But you will do this by accident and this is what will happen if you do this. You'd get a big spew of error and then it'll tell you the 10 digest iterations reached. So 10 is a built-in default constant of how many times it'll recheck everything on the scope until looking to get the same answer twice.

[Student]

Kyle: Yeah. It's actually possible to use these things sort of like spreadsheet cells. It's possible to implement calculations where this depends on this and this depends on this and so where you really like you want to take advantage of that but it's going to kind of go through waves again and again until it gets the answer. If you did a complicated one of those you might need to up that 10 to 20 or something. If you have to update it significantly you're probably doing something wrong. But if you want to use these things sort of in a spreadsheet cell sort of way, it is reasonable. One of the built-in services that you can call at startup, there's a way to pass it to a higher number to that.

[07:31:15]

So anyway, this thing you can see it got called 11 times. Interesting that it calls it then and here's a cool thing. It actually tries to remember what watchers it called and it kind of has some junk in here about some function performance thing that's kind of confusing. But if you just kind of wander through here, you got a big application. Sooner or later, you'd figure out there's some kind of age thing is what's going on. That's what's triggering this. It does give you at least a bit of a clue if you just dig through the mess, you would figure out what's going on. Interestingly it doesn't actually break the application. I can type and it still happened over here so the stuff that could work still did. It's just that every time I do that it's spewing out another error over here as it tries to, tries to check more times than it's allowed to. So the question you should be asking is, what is the performance cost of the way this works? Before I answer that, real question is what is the programmer time-cost benefit analysis of this? So there is an alternative tool you could be using called— oh what is it called? Ember. You probably heard of Ember. Did you look at Ember at all when you're considering this stuff?

[Student]

Kyle: With Angular, you get to write code like that. That's pretty nice. It's hard to get it wrong. It's extremely short. Doesn't cost you any extra time to write it. It's kind of native JavaScript but there's no way to interpose a hook in here. There's no leggage construct yet in JavaScript where you can say, "I want you to call this callback every time somebody does this." It just doesn't have that capability. You guys using any AOP stuff in your Java? Like Spring AOP or any of those? I saw one person shaking this way and one shaking this way. I'll take that as a uncertainty or disagreement. So in Java, if you're willing to complicate your build process or do run time code mangling, you can get a way of interposing that. In JavaScript I guess if you're willing to write in another language to compile the JavaScript, maybe you could. But Angular doesn't want to do want to do any. It just takes straight JavaScript in normal way.

The benefit of this is that it's really easy to write. The downside is since there's no way to get a hook to see if that changed. The only thing you can do is whenever it might have changed, go look at them all and see if they changed. Some code somewhere has to keep a list of all the stuff that might need to change to make those binding work. And this has to call each one and see if it's value's any different. But the alternative would be, if you're using something like Ember, you'd have to write this code instead. If you're willing to always write this code instead, it's possible to make this B some function that all your business model objects have that implement a hook to watch things and go kind of push the changes out. So that is the fundamental trade-off between Angular and Ember. If the question ever comes up, this is the things that's in your face the most often is the trade-off.

So far the trade-off seems to be in Angular's favor because your application has to get pretty big before this approaches a problem. I think the number that they float around is 2,000 2,500 is how many different things can be watched on a page before it starts to have a noticeable performance impact to the user. That's quite a bit. Let's create an example where this tests that. Let me see here. So what did I do since my last commit. I was messing with that ng-show. I was showing off some watch stuff. That's reasonable. So hold on to that.

[07:35:41]

Kyle: If you wanted to know how big of a deal is it to watch a whole bunch of things, that's actually a pretty easy thing to answer. Let's make a function that we can put on in here that will just spit out a whole bunch of stuff, spit out a whole bunch of little numbers on the screen. We know how to do that right? Everybody knows how to do that? Let's go up in here, we'll do a repeat. x in boxes. And that for each box, I'm just going to put box.size. So I need to put some boxes on my scope so this is the relevant controller. I'm not really using this age thing anymore so I can peel all that out. I said I wanted to put some boxes on my scope. Does anybody know a quick way to get an array of a whole bunch of boxes? Like an array of any kind of JavaScript, isn't there a way to do that? A way to get the same thing in array a bunch of times? It's fine if nobody knows. I'll just write it.

[Student]

Kyle: Yeah I know, that's what I was thinking. I'm sure underscore does. But it's just really easy to write zero minus less than 100, + +. And then it's push, right? Is it push? Is that the way you and then a literal object. I think we just wanted to have the size, large. That might be enough if I didn't make any horrible mistakes. Reference area. Large is not defined, oh yeah. Take it easy. Okay so we got a bunch of Ls but those are list items and that's a little bit hard to deal with. Let's make some stance. We had a whole bunch of Ls but that's not really very aggressive. Is 500 Ls too much? What I really want is I want a way to just change that value because I'm going to be watching every one of those. What if they just all changed? I could probably do that pretty easy. You could just say scope.change is a function. And that function probably does something similar to this. That's awful indenting. Why does it default to such terrible indenting? If I wanted terrible indenting I could use a Notepad. So scope boxes bracket I.size equal m. I've got an array of 500 boxes. We're going to get a scope for each one so it's going to be 500 scopes. And because I've done data-binding I'm going to be watching all 500 of them. So every time I make any change, it's going to reevaluate an extra 500 of these things. And if I call this function, it's going to actually change them all which should make them really change. And I guess I need a way to call the function. We'll put that in a ptag. Because as we learned a short time ago, you can do something with this. So let's see if I got that right. By the way, if this works, I'm going to increase that number until it becomes visibly slow. So if I type, certainly I see no... My CPU is making no activity and I'm seeing no extra costs. It's typing completely fluently. So 500 of them is not impacting me. And just to show that they really are being watched, check what I do to click on this.

[Student]

Kyle: Yeah I'm probably not doing something right. Did I not wire it up right? Did I put it inside B?

Student: You got to execute change.

Kyle: Oh got to do that, sure enough. Wow. I love making mistakes with it. So you can tell they really are being watched and still it's having no meaningful, no visible effect any way on my ability. Because with every key stroke it's re-evaluating that. See if I can find a way to make it visible. My CPU load is staying down there on the bottom. This crap running on my computer. Well if there's a core audio d why is it taking that percent of my CPU sitting here? Who cares?

[07:40:55]

Kyle: Anyway, this works. It's no problem. In my experience, if something works, that just means you need to turn up the knob higher until it doesn't. Isn't that like the general rule? If things work, you should probably turn the know higher until they break. That's certainly what I do with cars. Now we're going to have 2,500 of them being watched. And I can still type. As I type, there's no visible delay in the operation of my system and I can click and I opted immediately. Although for me 2,500 seems to be fine, that really is the recommended limit. depending on who you ask, what you read they'll say 1,500 or 2,500 or something. Certainly you don't want thousands and thousands and thousands of things being watched because eventually this will slow down. We have experienced this in real life. And I'll explain why on the board.

[Student] … 2,500 single characters.

Kyle: Right. It's not doing it intentionally. You don't accidentally create it like this. You have to intentionally create it like this. Here's how you accidentally create it. There's a tool out there called ng-grid. It's in the process of being re-written but the current one has this delightful design. It's made to be a grid control sort of like a lot of business apps that have a lot of grid controls. They have some headings here, maybe first name, last name, age, whatever.

The way ng-grid works is it makes a watch for every cell, so it independently data-binds every one of these cells to the value that would go in the point. And it actually makes another watch for each cell for the CSS to format each cell custom. So if you have M rows and you have N columns in your grid, and you use ng-grid, you're going to get MxNx2 which means that let's say you have 200 of those and let's say you have a really wide grid because we have some apps that have really wide grids. Users are like "Oh I have to see all the columns all the time." That might be another hundred x 2, how much is that? Well 0004 so that's 4000 watches. It's pretty easy when you're getting things multiplying together. So that ng-repeat, imagine the student, like let's see if they'd even know about NG grid. You just made that HTML table and do data-binding with ng-repeat and you ng-repeat it for each row, reptr, And in things like that, you ng-repeat it for each TV, for each column, and then inside there, you need a couple of these data-bindings. Well they'll be kind of slick because even a single value change in the whole table and just re-render only one little piece of the DOM. So it's kind of cool. But yeah you can really easily get cases like this.

[Student]

Kyle: What?

[Student]

Kyle: It is? I do? 40,000 oh yeah sure enough. So there's a reason I'm in the computer. Who invented math? Anyway, that's that. The answer is don't use ng-grid until they finish re-writing it. But the broader answer is you actually don't need to do a watch for each cell of the grid. The right thing to do for a grid is to just do a watch. You do a data-binding watch for each row of the grid. And there are grids that work this way. It's pretty easy to do this. So that basically it only watches each entire data element and in that way it re-renders that row if it changes and then this becomes 200 which we just saw that 2500 didn't create any problem.

[07:45:12]

Kyle: So if you just don't do it where you get the x times y thing, you will never hit this problem and you'll not be able to work with the problem. It's important to know about it because someone on your teams will eventually do this by accident. And they'll get a really sluggish page and someone will suggest the only solution is to abandon Angular. That's not true. That is a stupid thing. They should stop doing it. So questions about all that. We talked about how data-binding really works, scope watch really works, how they can add up. No questions? Answers? Criticisms? We hooked a couple but I don't, to my knowledge, I don't think of any projects that have adopted one yet. But there are tools like the competitors to ng-grid that know how to keep touching more data as you keep scrolling.

There's a tool called Beta Table like a generic JavaScript grid controller and there's an Angular wrapper around it. Corporate projects use that.

[Student]

- -

Kyle: If you just search around and see what's out there, you'll find something and we'll have an example. You'll edit the example to put some of your data in it. There'll be a real nice spike to that. Comments, questions on this whole data-binding thing? Is anybody surprised to learn what's going on behind the scene as it reevaluates every time and see if anything changed? Because I found that really jarring when I first learned it. That was really and then the second surprising thing is how little impact like how little it mattered that it did that. That was the second surprise.

[Student] So you compared it to Ember. Do you have comparison to Knockout because Knockout–?

Kyle: I love Knockout. I built things on Knockout before there was an Angular. And it's really cool. It does something crazy different. It looks like with Knockout, you'd say person.name Kyle so this is how you would set the name and then you would just do this to get the name. You build something with Knockout, you're familiar enough. Knockout it's kind of ingenious, because with Knockout if you make all of your pieces of data in your whole app, these Knockout observables, then amazing thing happens right. So you could say, person.full name = Knockout. They've changed the name so you wrap it with a Knockout computed. person.title. This is how you write a scope like this in Knockout. So it's a smaller syntactic penalty than Ember because with Ember, you have to write something like this. So the Angular has the least syntactic overhead. You just write it and read it normally. And you get all of the fun like sometimes getting unexpected results with the inheritance. Ember has I think kind of a worse case but maybe to the Java eye maybe not too bad. You can almost read these as if they're accessors. You just kind of ignore a little bit of punctuation in the middle.

Knockout has this different approach where you make each thing, each field be a function. You call it what the value or thought a value to set and get. Knockout does this amazing thing. Full name is a KO computed wrapper on the function which retrieves using Knockout functions. So when it runs your functions, it does internal bookkeeping. It keeps track of what other Knockout observables were referred to in the computing of your function. So it's doing bookkeeping. That means that it actually knows the dependency graph. It knows that it only has to recompute this if one of these changes, which is kind of brilliant. The syntax isn't too bad and the thing it enables is just amazing. I think it's really cool.

[07:50:12]

Kyle: But unfortunately, and the reason we don't use it and we do use Angular is that Knockout is this amazingly cool solution for this part of the problem which you then have to go compose with a whole bunch of other tools to build a whole application whereas Angular is even less to deal with. We just saw that the speed penalty is not that bad, not really a problem practically. And Angular is this big solution that has all these pieces. It solves a whole bunch of problems you need solved. If you know someone who knows how to use Knockout, they probably know how to use Knockout in some random collections of things with it, which is probably different than the set you know how to use. But if you know somebody that knows how to use Angular, then they probably know 80% in common with what you know and what you need. It's a more complete solution to a big problem. So that's why I think it has really a lot more momentum. But Knockout is technically just brilliant. I really like it. More comments and questions. Only one more thing and we're done for the day. We talked about service and factory. I'll just put that code up for a minute. These are either exactly or very almost exactly equivalent Does anybody know what a constructor function is in JavaScript? Constructor function.

[Student] We call it div. The function we call it div.

Kyle: And when you do a constructor function, you do line five. It's kind of hideous because there's just implicit this, right? So that function, we see function on line one, it's intended to use a constructor function and so it has the implicit this the constructor function get. Inside a constructor function, which by the way a constructor function is pretty much useless to call any context other than what's new because it refers to this which is not even a implicit parameter. A constructor function, it gets its implicit parameter of this and you're going to put things on that and that this is the object being constructed. So if you call app.service, you pass it a constructor function which will then get used in that sort of idiomatic way with JavaScript new and then that newly created object is the thing that will get injected when you refer to my service. You follow that? The factory syntax is much simpler. I think it's actually a line or two longer so I guess that's a reason to maybe not favor it. But in the factory syntax, you just return, just literally whatever you want. In the top one, there's an object being created. It has a say hello function on it. And in the bottom one, there's an object created that has a say hello function on it. You get the same thing either way. You get something you can call say hello on. They're just they're just syntactically different ways to do it.

[Student] And the thing I like about that syntax is that…

Kyle: Which syntax?

[Student] The bottom. If you had 10 functions that you worked right in there, you could define the functions up above the return. And then the return it kind of defines your API that if you were just to–.

Kyle: Absolutely. In fact here what I pointed out, there's this nice little bubble. It really applies either way. You can run some code here.

[07:54:08]

Kyle?: If I wanted to, I could define something up here. I could say say hello equals function and then down here, I don't have to take it. I could just cut and paste it. So you're saying you could do something like this and then you can go build the thing, right.

[Student] Or if you have 10 functions that returns succinctly.

Kyle: It's real concise.

[Student]

Kyle: You can actually do the same thing the other way so I'm not sure if that's really an advantage of one over the other. Let me see if I can at least type this right. I don't even have this in the JavaScript file. I'm just trying to illustrate you could do this and then you could have a bunch more in here and then down at the bottom. You could do this. So I think you could achieve that same thing either way. I prefer the second one because I find it more obvious and easy to remember. And I think the second one has become the most popular form, but for reasons that are not clear to me, we all say service even though we usually type factory. You can just remember that service and factory are nearly equivalent. You use them differently but you achieve about the same thing. And that everyone calls them services but most people type factory. Remember that. You're ahead of most Angular learners.

There might be one more thing to try to fit in and be done. This is a good one. I was thinking about doing is just making them work because just going back to where I started. So ignore that and then this I can ignore this, we don't need this. Go over here. And just paste those in for real and have them really work. Like that so then in theory I could put some code in here that would rely on this. Remember I said these are injectables. So I could take this and if I put it at a point where injection occurs, it will be injected so it's going to run this and it's going to give me one right here. And then I can here. Something like that and then I need to data-bind greeting so it will be visible on the screen and we'll put that right here. That's in D. This is an A thing, right? We'll put this right up here. That worked, the Hi Smith worked. That reminds me that before I really go any further, I probably should commit this because it has this nice example on it. I'll just let that thing sit.

Actually have you guys ever used the ability to do partial file commits and get? You guys ever done that? So I could do something like this and now it's going to understand that these are separate changes so I could say I want that one but I don't want that to go in here and say I don't want those, I don't want those. This and I do want so I do want this and then I do want this boxes thing. I guess I need to be a little more selective. And that age, that's all fine, so I do want all of this. There's a command line and kind of put you in an editor and you have to put marks in. It's not pretty.

So now I've staged the changes that have to do with demonstrating how scope and watch work but I've left these changes on stage. That's kind of cool. There is no subversion equivalent to this other than manually copying files around and editing them. Now I can do a commit and this is all about scope watch. Did I say scoop watch? Scope watch 2500. We did 2500 right? I think actually I can see right here right? 2500, yeah. I can commit that and then I can go back later and I don't want to commit this yet because I'm still working on it. Really user capability. It's real tricky to do the command line.

[07:59:30]

- -

Kyle: Some of the more primitive get GUI's do not have a way to do that but I think the one in IntelliJ is pretty good so I have a feeling you can do that exact thing in IntelliJ that I just did in SourceTree. SourceTree is free. All this, this stuff is just sort of junk left over now. There's no reason to let this pollute the space. Let's just put in two greetings and then we'll bring in my factory. I think I need to start sharing my space a little more aggressively. And then might as well show that they both work. If I've typed everything right, it's always iffy. Something's going wrong. Somehow I lost both of them and I'm not getting an error, isn't Angular great? So we have scope last. We're calling each of these, greeting one, greeting two. I'm binding them by greeting one and greeting two. I'm in the scope created by the same controller. I don't see any syntax errors down here. I don't know if this will have any effect. So why are these not working?

[Student]

Kyle: They work in both. I just haven't saved the file. But the answer will be is not a dependency situation because I put these all on the same app. If I wanted to, I could make some place to put these. I can call them services and put them both in there. And then there'd be the service dependency. This is an example. The reason I knew this would work is because I had copied those out of a working example in the past. That's why I knew it would work. They're just two different ways to make that service. As far as I know, it's primarily preference. There's a more general mechanism that you might investigate if you're ultimately curious called Provide. Provide is like a general purpose tool that sits behind a bunch of special purpose tools like controller and service and factory. If you want to really get into the nitty-gritty of Angular, there's a way to use Provide.

Any other questions on this because this is the last topic for the day? The syntax, the details of service and factory. Did anybody notice I was able to put these lower in the file and I referred to them? That's because running the code is just the finding ding all these things. It's only when it tries to actually use the controller but it does dependency injection and tries to look them up. And that could be an hour after I loaded the page and got all these registered in memory. More questions on this? This is what I'll put up on the screen when I get here on the morning because I'm hoping that everybody at least kind of go through this exercise of defining a factory and a service and being demonstrably familiar with making it work. We'll do this briefly in the morning. We'll jump into promises. Well thanks for suffering through a day.

[talking with Adam]

- -

Day 2

00:00 - 00:34 [Students talking]

Student

Did you know about the W3C widget spec?

Instructor

Ah yah. So that is coming down the pipe and it has a lot in common with angular directive model. Google has a competing project called Polymer and there's a lot of people working on it but you could sort and think that it's aiming to be, like how you would do angular in a world where widget spec is already implemented.

Student

'cause I know it was like published in 2011.

Instructor

I think it's just an evolving spec, it's not you know, it's not ready to use yet. So the question was, what year will it be, when you can rely on all the browsers you need to support, natively supporting widgets. An answer is not real sure. That's separate part of it but I mean fundamentally I mean, it's just HTML but…let's start getting this stuff up…

01:43 - 02:01 Seems like the students are discussing

02:00

When we have something you need to see I'll make it big enough so somebody in the back of this room to see so it's big enough for you to see. So I am gonna bring up a file... So that's probably not big enough right? 'cause I know I, reduce my font several times after. So that's more like the font I need to use. I realize that, for some reason people don't say in presentations or training when the font were too small widely enough, the only time they tell the truth is I have to actually physically went to the back of the room so I can read it, it's marginal I would say.

02:48 - 04:28 [Interaction with students]

04:29 [Students and instructor discussing about video conferencing system]

04:57 - 06:03 [Students talking and waiting]

Instructor

- -

Are enough people here to start? We'll start so another show of hands. Who is ever used a promised, even once in any language. If we take English out, Okay, so in JavaScript? And that's all jQuery right? Callbacks are terrible. Shouldn't do callbacks…… So deferred and promise mean about the same thing. A future is a slightly different thing. Languages would have threading of in everything called a future. Languages don't have threading, don't have a thing called future but they do have a promise or a deferred. I will start at the beginning to give a little bit of exposure. So what would you do if you wanted to fetch a computer thing over HTTP like Java and then like do something like that could... sorry different question... or you could do something like you know, A = fetch of you know some URL, then B = same thing, then you could say C = A + B. So imagine what could have that if just a number if you could have them together, it's very abstract. How did you do that is JavaScript? Anybody know? Yah so, if you just have callbacks, right, you got to write something that kind of ugly, something abstract and discreet than children can see it that if you write it into spurge you can slim it to that and stuff. So you would do something that is abstractly anyway something like, you wanna say A = fetch of URL1, right, so if you can't really do that because in JavaScript that there are no synchronous blocking codes, the language doesn't have threads, the browser doesn't have threads, there's a thing called web workers you should know about later. So you just can't write this line of code, there's no Java version, not like just changing the browser of anything, there's no special library, there's no library that you can get, you just can't do that, if you could do that, then only one thing could ever be happening at a time. That's why no browser implements that kind of call. What they all implement instead is something like a fetch [10:00] operation like that is very abstract or you have for the URL and then you give a function that will be called when it comes back, so again I am being super abstract so I am just gonna type it with an [10:13] so you can do something like this, you know, this data might be some sort of data and you could say something like A equals that data. So you could do that. you think will then this, that's pretty good, so I'll just do that twice and I'll just fetch the URL2 here and then I'll get B from that one and then I can just to do C = A + B, well would that worked. Why would that not worked. And how might is the might, yeah pretty sure, the reason the answer is pretty sure is that unlike Java, JavaScript is single threaded, like your codes is gonna run the completion before any these callbacks can even possibly run so when you gonna think about, when your codes run out, like you return from the last function, that's when JavaScript's running all the callbacks, if there are any, so this definitely will not ever work. So you get to think a little... what I kind of like to do is like put that here, right? That would be kind of nice, 'cause at least if I put that here I know B is gonna be available, but then A isn't. You think you know what I could really do; I could put that in here right? Like what if I put that in there? So now I am gonna fetch 1, then when it's done, I am gonna fetch the other and when it's done I can add them I can add them, so this will at least produce a logically correct result. What I ask in the beginning is that, is there a way I can just run these things of the same time, right? 'cause I really want to get them both started. Well that's kind of a pain, because to do that you kind of, you gotta go back, well you know, it was better before that's wrong operation. So you think well you know I actually like it better, when I started both of

- -

them, but you end up doing really complicated things like you know you make some sort of, some number and equals zero, and then on each of these maybe increment M, and when that's come back you increment N, and then maybe on each side, what if I said if, if N is 2 then now it's time to do this, and then what if the other one happens first so you really gotta do that on both side, so just try to reason out how you gonna use callbacks to start a 2 asynchronous operations in parallel and then do something when they're both done, it's kind of hard. You probably find some example on stack overflow, that's correct, but it's hard. And the reason is that callbacks are a crappy abstraction they're not composable, like there's no way to take 2 callbacks and make them together and make it one callback. You can make one call on the other but it's not really the same thing. So promises are a solution to the fact that this is hard. And when I say this is hard I am not only meaning it in the context of HTTP things, I mean just the general phase. Being able to have asynchronous things happen one and make it able to flow together is kind of hard. There's can be some like waiting for user to click the button, and like that's how generally I mean, asynchronous, right? If this is hard and a pain and doesn't compose well, what does? And the answer is either a promise or a deferred. If you read the promises A spec, these guys actually use both words promise and deferred to mean different parts of it, different parts, different aspects of the promise problem, but if those two words are use almost interchangeably, so it's a jQuery deferred of what they called a promise implementation, the promise implementation. I am gonna show you just one in angular uses both words for different parts of the problem. But if you, Here's what's where talking about deferred they mean the same thing as promised. It's a little bit different from the future. So, we won't talk about those because those can just line it up on threading. So now I do need to use this board,

14:40 - 15:46 [Audio inaudible]

I'll talk with a little bit with this so I can... Okay, so, promises is the general idea of an object that represents a computation whose value might not be available yet. So if I hand you a promise, you have this object that's handle, and it might already have an answer or it might not have an answer yet but it's a promise that you will eventually get an answer that what word promise means in the context, just for background there are something like a hundred plus different implementations of this promise idea for JavaScript. So this kind of like clever little idea and each person who rediscovers this who reads about these things, Oh would like to write one of those. And they'll put that on GitHub and then there's a hundred and one. Few years ago, a bunch of people whose responsible for the most popular ones get together and came up with this common promise spec called promises A, and there's a variation called A+, different variation called B, those don't really matter, what matters is that there's been kind of coming together of a general agreement of how promises should work like what the API should look at that means that if you learn one of them that works like this you kind of know all of them, even if you pick up a different uplighter for a different project so that's good, you're learning over head of the lesson than it used to be. So promises is a general spec, don't care about that much, the most popular of this codes then last time I look promise implementation, follow it's spec and it has the brilliant name Q. Just Q, try to google that, right? This is the thing we talk about yesterday it's like a search bubble. If your programming type Q, you probably find this thing in the top page of the search results, here in other program you need to type Q in the Google you probably won't find this in the top page of the search results. So Q is just a library, it's not an angular thing; it's just a general JavaScript thing. Like this Kris Kowal guy from Ambrose called Main, its Kris Kowal something. It became popular a few years ago, because it's specially written. And if you just want a general introduction, I would recommend reading this page, it's a few, forget everything I told you and you just go back and start on this page, there's nothing angular about it. It's just JavaScript, so I started making this tiny version of a pyramid of doom on point what they called and you gotta hold on to a callbacks and sight each other's and they talk about it here it kind of shows you the APIs so many of the things that I explain you can get to explain in writing here if you prefer. So Q is a general purpose promise implementation but that's not the one that were gonna work with because the people that made angular they like to Q a lot but they didn't need quite as much as Q has even though look, it's pretty microscopic, so they made a cut down version of Q called $q, this is a strange, strange documentation page, its ugly, so when in doubt in angular if you search for the thing you want in there, you will find the right page every time. So Q just by itself which is actually capital 'q' as its name is a general purpose JavaScript tool, there is a slight subset of it in angular called $q so they say is inspired by I think I read somewhere that its inspired in the sense that much of the codes is copy 'cause it's an open source thing. I don't know how much it's inspired versus extracted but... if you start here you see everything in an angular context you see dollar sign's everywhere. So this should be your starting point when to understand angular promises or deferred and all the details of the API are here, and I am not gonna read it to you I am actually gonna show it to you by the time you gonna learn to make it work. Ok, so, now is when I draw something, a Q promise is somewhat different than a jQuery deferred and that you just chop in two pieces. Draw the pieces above in the look. So in the Q promise world, there's a deferred which has promise, making that you understand this is that the deferred is where data comes in we have things called resolve and reject and then the promise is where things coming out a thing called then there's other variables will see later in some certain catch ID in there.

[21:00] A promise contractually has 2 pieces, I shouldn't say physical compose attraction of a computer that 2 parts of a computer the deferred and the promise. The reason why its split in half is that most typically right here there will be something like a function called return. So typically one part of the system will make a deferred with whole other system and then will return a promise that just connect it to that deferred, and we'll see that in a minute. So the puzzlement is okay, this whole background and the typing things make a little bit of sense. The reason its separated is that the promise only has answers to come out the deferred only has answers to go in, and if I hand you a promise I don't want you to be able to put the answer in, I hand you a promise so you can get the answer out of it, and I want to be able to drably hand out this promises, I want

- -

to be able to pass out many copies of this promise so all around my system, and I don't want a lousy programmer and want to, and some other, some part of the system that receives this promise out there to be able to put the answer into it, right, so its split in half to sort of a defense against people not understanding how they're supposed to use promises. So the part of the code, the answer is coming from holds on to the deferred and it hands the promise to the part of the code that supposed to be expecting the answer. So if somebody wants to, you know, hold that up from the camera for a minute, probably get file without it. Okay, so what if I wanted to do something like this, that's promises, let's do that. The place I am gonna do that is I am just gonna go right in here, this code already had going and I am gonna pair this example back by taking the service and factory stuff back out then this is where.

I use the series of factory, I am trimming this back because I wanted to be as illustrative as possible but when you guys as to working on, your example mean two or three just keep pilling more stuff in that same program, there's benefit in that too. I am just using the fact that a controller of some codes that gets executed by angular this is a good place as any as we start talking about how promises work. So I guess as long as I am trimming things I could take these grieving's out 'cause were not using those anymore and I am not really using much of this page as it is, let's just make sure the page still works. There we go, you know they never they get around to doing. It still says Google phone gallery at the top, there we go. Actually this a good place as any to show a bit of get fun, we'll take a slight diversion here. And we were to get, do you guys ever wish you would commit things on a different order, have you ever reorder things you get. Is there anyone knows to do that. Have you rearrange commits in a different order... I don't like to any of us since I like it better that the opposite. So this commit is purposes to trim out service examples. But then on this point, say you know, I actually made a mistake here, I should have labeled this Google phone gallery, I should've label this Kyle's angular JS examples, that will be a better name. You know that's a pretty easy commit, right? So I could just say, sure. Rename it at the top. So here's the fun diversion on this. What if I decided that I really don't like having done that at the end right? So this is a typical get kind of thing it shows you the, its time and reversal order so this is now and this is later. I really wish I had made that commit like, all the way back here or maybe right, that would've been a lot better of time to make the commit. Well its actually pretty easy to do that, so rearrange this, what I can do is I can just do a get rebates and I think let's make a commit number here its EFA3740 it does help when you type them right? This is a get interactive rebates, so the cool thing here is I can just edit the order of this lines of text, to edit to order of the commit they made. I can just take this line from down there and put it all the way up there. Put it up there, I just took a line of text and I move it up to the top it and seems these commits aren't stepping on any of the same lines of code .

This won't break anything, I can say that, and get just gonna rearrange all the commits and that's it I am done. So now next time I look over in here its gonna show having put the name on the top all the way back here the order I put it and then latest commit is now this form where I trimmed down the service example's. Part on the file that get diversion but that really useful to be able to do

- -

because if I were to pull this example, let's pull this commit back to show you something it would now be labeled correctly at the top. So I can pretend that I did the work and the right order. Sorry for the diversion, I didn't like seeing the wrong label of their outlet on that history. Back to promises, I mentioned that promises are named $q in Angular, so if you want $q you just ask for it to be injected like that. You can do that in numerous places throughout angular, I might need to, is right it hand. I don't need to add anything to the file list here, this list is fine because Q is very core to angular and that's the main angular file. You know that's just stuff. And I also don't need to go up here and add any module dependencies or anything, its core its right there don't need to do anything. So that was Q world well, let's get a deferred, that's how you get a new deferred, easy? I think there's a jQuery, it's like $ dot capital D deferred to get a new code in jQuery, does that sound familiar? What if I wanted to resolve this thing? And would you have any interesting value in mind, see if fire alarm on the wall who resolves this deferred fire alarm. And then I mention that there's a link when you have to defer to get a promise, that's just one line of code. And then we have P1 'cause its promise 1 to go with D1 and many things you want. Just that and notably it is not, not bad. Common mistake, you can get an ugly message which says find a way to make it a better message you can get an indecipherable message and you thought as a function it's not helpful, and then I said there's a way to get it out. Now, if we have futures, then you could do something like this, and that would mean wait to that thing to be resolve, just stop here and wait some other process that we don't have that in Java, we don't have futures in Java, we don't have threads, there's no blocking so if you don't have that capability look in another language. What you do have is a way to [29:28] The way exam works, it looks suspiciously like a callback because you put in a function here, here is the code that will be called when it has result. So see if that works, think there's, there's having a console lagged so for the sake of ease, I'll just leave this guy up, do you guys know why I just hit it twice? Yup, for the sake of this example. It would be more pleasant if didn't see it twice so I am just gonna go take more of the times where use the controller stops doing that. And what is this LP set though thing, that is, nothing that looks as promises but it's an annoyance. Let me read this, from extension on Load WFF. I don't know what that is. Part of this program extension I am running [31:00] minified code, I see some languages listed. Yeah, that would make any sense. That's just kind of last class right? I wonder how to move it 'cause I'll use it for certain things, here we go, when in doubt, you can just disable, a little down here in front of the button. So when you're having trouble with an extension, you just find it, and you disable it, I am loading this again see if it goes away. Good call. Okay. So, run this little bit of code, so what happen? So this is pretty boring already, right. 'cause we got the answer here and we printed it here. So the slick thing about it is that it still works if you move this down so here I have the code that's gonna run the after code, and then I put the before after it, at least the middle order of the file. And the slick thing of that still works is that of the same way. Any question so far, in this tiny, tiny, little bit of promise. Yeah, promise questions. Anybody see it worked? Yeah.

32:30 - 32-38 [Student Asking question]

If you started a process and you got a promise. Well, if you replace your browser in JavaScript with a language that had a process start, then you could do that, but there's no such thing as starting a process in JavaScript, it doesn't have threads, it's just not future is not there

32:57 - 33:06 [Student talking]

33:08

Instructor

Is there a way to ask the HTTP operation underway to abandon, right? I don't know, but that actually a question for the HTTP library, that it would have to provide in fixed, but there's definitely no way in the promise world to force it to do that. It's not a threading mechanism, there's not a way to forcibly stop the thread because there is no such thing as a thread. All this stuff that I have in the background has to be provided outside of this environment. Like some native code. So this native code in the browser that know how to do the HTTP fetch and that would just be a question on whether the people that wrote that native code provided the call to tell it abort HTTP call. Well you can decide not to do anything when it comes back but that's very different in making it stop. And don't let somebody sell you one but you want the other. So this is, pretty simple case, so there's P1 thing like for example you can easily do this down here, it's pretty slick, there we go, then of undefined what happened there oh, and I call then of undefined, oh here is said P1 of before I... So you can do this whole thing again, if you like that, you can do it then. So I could make another one of these things. Get this promise, and then on this one Let just give this a different answer to prove it more obvious what going on on the screen. So you should see that that they both run, right? So there's quite a bit of flexibility here when an order will write this lines of codes. I could write both this lines of codes way down here for example we'll know about if still works. That's pretty nice.

35:42 - 35:59 [Instructor talking with students]

36:00

So a promise or a misrepresentation a combination of deferred to promise represents an answer to some computation that might be available now, might be available later. If you knew the answer to point [36:13] although I would point out that you're not stopped from doing it, it says that there is lot of flexibility, if you wanted to use promises for whatever reason and you knew the answer right away, there's absolutely no reason you can go all the way right there and resolve it immediately. What you can use is this uniform interface for things that are known right now and things that aren't known right now. I give it around to change your code if you decide to make it available immediately. What am I asking? Well $q this is just a name that angular gives to that Q library or just the implementation of the Q library and Q.defer that's just the constructor function that actually gives you new one of this, the new empty one of these. Fetch things to it to do. Well no it's not a thread, it's not a thread of execution, that there is no there has something to do. Looks like there's no, like when you doesn't have any abilities to do something on the background, that's.

Well that's good, that's why you take in incrementally. I was pointing out on these things here that might happen all the way to the very end. And if we do that it still works. But they might happen later. So I could say this things are gonna happen at some later time, and that's gonna be I don't know, few seconds later if I do that when I reload the page I'll wait few seconds and then they'll appear. Ok, so I didn't really give it something to do because all it was doing was, was waiting, but there is a means in there that, that something happen later. But it's that, I mean I adjusted the time out like it's the shortest easiest thing to type something later. We already learned about HTTP right? So we know that there is a thing called HTTP its good from making HTTP calls. So I could do something like HTTP... something's going to right on my copy and paste. 'cause I should learn to type while looking away more effectively. Ok, so I think to the thing that called GET on there and I can pass is to URL, and I don't know I guess if I, if just to look on this file I'll definitely find it. So that should look familiar from what we did before, I think last time I use the phones.json or something and well, when this came up, so that's not available immediately, the browsers got to go fetch it, and that gonna happen some number of micro seconds or milliseconds later, and there is this success thing and you give it a function, and that function get the data from the call, so there's way to write some code that'll happen when it happens. Well like I could actually resolve these things there. So this would also be an acceptable way for things to happen later. See if this works, if type it right in the first time or not. No, 'cause there's a rows or URL to where my page is so this the right code for URL. So that's the step 4 of the question you are asking is there a way to give it something to do, the answer is yeah, right here, I gave HTTP something to do and the total when it was done, to run some codes and then I resolve this promises. See that is an incremental stuff. It all comes together over time. Have some data to one of the resolves, what does that mean? Oh sure yah, It's like something to do like it's data I think this will be, I don't quite remember what it looks like. But if cancel lag I can see what it look like what it looks like. So that is just to literally to test the page, so I think you can say length from that, and that its numbers, which means that I could use that as part of one of these, right? [41:00] like that that would work, See? So some computation something have to happen to take some time, and when that was done the data came back and that was used to resolve a promise so some code I previously set up then executes.

41:26 - 41:58 [student talking]

42:00

What you're saying what if this just play never happens, it never got called, right? Oops I need to come it out more efficiently. Well if it never happens then it just won't happen. So the important thing there is that there's not really a lot of differentiations between something that will never happen to something it hasn't happen yet. So there's no law saying every promise not to resolve eventually, I mean to say it's valid for some state of the system there were promise that never resolves. And mostly failure that it comes out to. So this is the point of that there is, there's lot that's can be done here, that these triggers can come from many places, you can resolve them from all sort of places right? So what if I set this guy here, maybe I'll resolve him differently and

- -

never where in a controller and I intent to use this in a controller because I can access the scope in a controller it is occasionally useful to do something like this. In the function that's on the scope, and we have function that's on the scope it's pretty easy to make an on click here, so let's go somewhere in this B, where I can do something like, when you say button directly to say input type equals button does anybody remember? Yeah, you got the input type = button, is that my button? So yah see? So once my button gets clicked then it resolves, right? You can't necessarily determine by statically looking at the code if it will ever result. It's a question whether this results is relieves the question where I ever click the button or not. Now I can numerously the few things to if I click the button repeatedly. It does not to happen again, so a promise is a one shot. So promise is an analog of a function return, and functions don't return more than once. A function is like the asynchronous version or a, promise is the asynchronous version of function returning. So to illustrate that, if I went here, and sort of pre resolve this look pairs then a pairs since it's already been resolve with pairs, attempting to resolve of something different, it doesn't, it doesn't work. You already follow me so far?

45:01 [Student talking]

45:10

Right here, HTTP.get right there. Yah, Wait, what's the question then? So it happens that HTTP is implemented with promises on the inside but that actually doesn't matter from this point of view. HTTP could be implemented by any means in the whole world as long as it has a way of being told about a callback that it should run when the data comes back, so this syntax to see, I think that's actually pre dates promises. If you've been able to use jQuery AJAX for long before jQuery had deferred So promises are kind of a higher level abstraction build on callbacks, you said callback is the beginning of the web. Long before that really. So this is just a callbacks, right? I saying, here's, and then a callback can be solve into promise. And that's exactly what this does too. So you see this in a minute. Just kind of played around a little bit here, if only played around the success, so there's still failure to look at, and if I've use this two promise more or less separately, and I mentioned earlier that they can be composed, so really important feature of promises, and I've mention that of I'm gonna tell you this now so you may need more times, the more you use promises the less you talk about promises in your code. You end up talking but you end up saying promisee thing at the boundary between promise-based codes and non-promise-based code and what I am showing you is a simple example it's all full of that boundary. But real code uses promises pervasively and the boundary is never, not visible, 'cause you're using it pervasively. Let's make that happen…….. If what? Well how would I not use promises here, like what would that mean……… A promise is just an object that represents the value of some computation when it's; it might be done now it might be done later. It's not an engine that performs the computation. In this case, well in this case the computation that I am using behind D1 is the fact that HTTP is gonna have to go and like talk to some low level operating system stuff and talk over the internet and....... Well sure, if wasn't gonna use promises I would, I could just

- -

you know, what I would just say, actually you're saying like put that in there right? Well it's not helping me yet but I am trying to get enough of the syntax in front of you that when I make it start helping you, it helps you. You think I've trying to do it on all alone so. What do I have? Okay, so this is the thing I was typing you'll say it's real pain that, set some calendar and try to track and two of them answer, So this is when you get to the good part, it's a, promises unlike callbacks compose, composing is a really important idea in computer science I need to tighten up this code so that I am not typing in small print at the bottom of the screen. Just to show where we're at, one of the promises resolves it for an HTTP fetch, and one of the promises resolves when I click a little button. You don't have to type $q very often, but you do in this case. There's a thing called all, all takes an array of promises, and gives you back the single promise that composes them all together, so what am I, my promises are P1 and P2, so back to promise, and what do we say that I could do at the promise when I said that I do a then on a promise and then print out the answer. So since this is a promise I could type them on it. And the do a function with an answer, so this will run after both promises have result, so this is the benefit, right? It's that the thing that I was pointing out is that's actually really hard to even reason about how to write it with callbacks is completely trivial one short line with promises, so I can just do a console log both are done then if I print the answer, this will actually be like API detail is that you pass it an array of promises and you get back an array of the answers from those promises and this should print out both of those in an array form. Why it isn't printed yet?

51:00

Right, 'cause the L is not result, so one of them results so that all you can think of it conceptually is being an threads that call through waiting for my button as if there are not threads, 'cause the language has threads, it's actually just some book keeping in some objects all written in JavaScript so that when the other side of the promise results which we saw here, then we got the combined answer there, that's a really big win for even the very simplest case. And to share the couple of other of the interesting tips, so, on this one I did this on key HTTP get thing with index HTML a success but the HTTP get it's actually better that, because if we turns a promise, which means I if just do this now P3 has a promise and let's see what else I can try, I'm already using index HTML so maybe I'll look at, let's see one file name here, I'll give you something different, CSS/FCSS so P3 is gonna get that guy, and now it's a promise so I can call a then on it. Call this A I guess, when this one resolves, you know, I'll put his together 'cause I am not gonna compose it yet. I'll bring at top on the screen, okay, so P3 is a promise because I happen to know that HTTP that I get returns a promise. So here, I am using promises and I am resolving it, but I never type this, and you'll find that when you're using promises pervasively, you're almost, always starting with a promise that someone else made for you, you're only rarely having to make one of yourself. When P3 comes back it's gonna print something, let's just see what that looks like. This is a sort of thing you just have to work through the details of the API that promise results with an object that tells you all about the HTTP response. I could have the status in there, I can get you some headers, there's lot of stuff in there, I happen to know that the part that really care about it is called data, I do data now it's gonna just print out the actual style sheet. And since I am trying to kind of boil things down I could just check for its length, there we go, that's 57. Remember that I set these things compose though so I could take that guy and I am just gonna put it up higher so it'll be in scope here, so I can go ahead and include that I could say, I really want to, I want all three things to be available. And so happens that the order that they're gonna come back in is that one on three are gonna come back almost immediately and two is gonna come back only once I click a button, it's as long as in here I could try making this button a little more obvious, is it value you type or label, what is the? somebody knows HTTP, HTML, that way, it will be a little bit easier on the eyes here to click, there's that one results immediately and printed that one aside from printed them I click this one, it results and they combine the answer results and then they, see it print says a, as an array as an object in it see it is a little bit of junk in there. Does it start to make a little more sense? Okay. But there's more, promises not only resolve but they have the stand out effect, as part of resolving, right? When I did this example each one of these guys could only go to one callback so you've ever like, maybe they already had a callback or something you wanted to add another callback, if you right a function that calls both callbacks, have we done that before? Promises have that to solve in the box, because you can call them as many times as you want. For a given promise, there's no rule that says you can only call them once you and you can call repeatedly. So I believe that P2 is the one that'll print out both of those THENs and upfront to combine the then. Okay, that means that it's not evident here 'cause I am just keeping them all in one file, but then it's safe to return one of this promises, remember I talked about this, there's often like a function return boundary right here, you can return one of these and pass it to other codes throughout your system, even untrusted codes somebody else wrote, because that would set them up this angulars on it doesn't break it, like it doesn't damage it in any way for all over your code you can set it up like analog or something. This D2 that I have resulting off the click D2, what else could I do, I could add another line, I could you know do something angularly in there. So I could say some scope thing here, here. Scope point blah blah blah. ……. Can I give P2 a new promise, so a promise, is the analog of the function call return. As if you don't know what's gonna happen now or later. It is single shot, you merely try to re resolve a promise 10 different ways and nothing will happen, here, that's not the right code, this is, it is the length in it. I think I showed this a minute ago, and I will just do it right here,……. so I could try to resolve a different times and nothing different is gonna happen it's gonna resolve with the apples and that's it, so calling resolves more than once will not make anything additional happen.

58:05 - 58:39 [Student talking]

58:41

Instructor

- -

What do you mean by P3 a different, P3 is, it is a promise...... Well, if you make a deferred this way, you can resolve it with anything you want, right? And this is all composible, I think we'll see something, I think I am forgetting that, we'll get there. So you've seen all, you seen resolve, you seen a P3 resolve more than once and it doesn't matter, that's just remember this is the asynchronous analog of a function call returning. Functions often return... yeah this is just to point out that this is the analog of the function call returning, a function call can't return 3 times, the only value, right. A promise has a 3 states it's either open, or its resolve or it rejected. We'll see rejection in a minute. And the only actions it can do, it's created in an open state if you call resolve it moves to resolve state and stays there forever, if you call reject moves to rejected state and stays there forever. That sounds limining but it's actually quite good because I mean this seems are very easy [1:00:00] to reason about. There are a, there's sort of pipes and filters models to get like flows of data or some people think of, I want a way to do a flow of data and there are libraries to do that, but there's much more general purpose than that, these things are not really as easy to reason about as promises 'cause they can have all retrieve data keep flowing through this. To see success we have to see failure, so I can go right down here, I am just picking anywhere no particular reason. And I could take like this D1, where I, that's gonna get resolve when the HTTP comes back, but what if I rejected it instead. So much of a function, can either return or can throw an exception. Those are the two thing two things, like let's do it in Java, let's do it in JavaScript, you can't do both right? Is everyone so I see it is just learning Java trying to figure out how to throw an exception and return something, it can't do that. A promise is the analog of that do it can either resolve or it can either reject, reject is like throwing an exception. So if told D1 to reject, then when this eventually happens it will not print anything so let's see. I'm still picking, how come my clicking isn't working, somebody know why my clicking is not working? It's because once I rejected D1 it's permanently rejected it can't change its mind to become resolve later, and I have this combined promise, well this only gonna resolve for all its constituents resolve. To my honor how can I like, doesn't this break my error handling? Well......When I compose this composition, ... You just have to keep going back to, it's like a function call return. If I have a function A = full and B=bar, right? And C = A + B, what will happen if one of these throws an exception, and now guess to that. If you have the C, you're composing and like yah if one of them fails the whole thing fails. And we can see that, well we did do see it, and if you want to catch it, you want to actually get something in front of the user, you can just go right here, it's as D1 that I am rejecting, you can just add another callback, right there, this is your error callback P1 failed, you can pass the details of the error if you want, and so if we do that, we will see P1 fail with that failed message. Now slick thing is that that actually composes pretty well too, so I can, catch all of them, so much like, if you're on exception handling, you can write a bunch of code that composes a bunch of operations together and put a single catch at the end, I can do a whole bunch of things and that, I then run a code compose them together asynchronously, and put the equivalent of the single catch and the end, so now because one of them fails, a combined one also fails. Right! You can do aggregated error handling just like you can retry catch. And it's exceedingly painful to get error handling right, to get a bunch of nested callbacks. Almost nobody does, 'because it's incredibly tedious. But it's pretty straight forward, okay, so now we're gonna switch to everybody take your example and do just some little bit of thing with promises, you don't have to do anywhere near all this, but make a couple promises, reject them, resolve them, change the order lines of codes, ask questions, Paul and I will help and this is like the first 2/3 of promises once I've already has this as little example working, prove it to yourself you understand it I have an additional really slick part.

1:04:44 1:04:59 [Student talking]

1:05:01

Instructor

Yeah, there is some naming thing I would recommend sticking with this naming, because this is not an angular thing, it's a Q thing and Q not even a Q thing, it's a promise A + thing. promises A + that it's spec that all the major ones are following so if you learn all these things and you switch to some other tool next year, the only thing you'll have to relearn is that something else that has the $ thing Q. Everything else will be exactly the same, if you switch to competing promise A+ notation. Hopefully jQuery will switch eventually then that I may add two three or something…… It may just hurt you to know that I left out the trickiest part of the core

1:05:54 - 1:07:01 [Instructor assisting the students on the given exercises]

1:07:01

Instructor

You just get to pass one parameter to reject if that's the only information available is that, and that's a bit passed up. No, this doesn't do something about HTTP it is just really a standalone. Well, P3 is an HTTP promise so what if I just made it fail like that. It's that what you're wondering? So the wrinkle is that I need to stop rejecting it here, 'cause it will just reject first, if I do this then P3 is gonna be that one that fails and there you go……..We'll see in a little bit how to dig more, dig better stuff out of that,

1:07:45 - 1:08:00 [Student talking]

1:08:01

It's pretty common to programmatically build up an array of promises, right, it might like have three things that you know and you've hand some more on dynamically based on some run time data, leave that up for a few minutes.

1:08:17 - 1:11:05 [Seems like students are working on their computer while the Instructor was helping.]

1:11:06

Instructor

That's just the browser console log, not anything angular specific. If you just go in chrome, and then you do anything for you to dug tools up but the easiest way for you to dug tools up is just right click and start and it will start the element and if it picks up the wrong part you can click over the console, that's the way to be able to take it to print statement.

1:11:30 - 1:12:19 [Students continue to work on their computer]

1:12:20

which was for two reasons, one of the reason is I wanted to be able to take use some sculpt click thing as part of the example, the other reason is that is the context of the controller it's a place where you can do dependency injection you can get a hold of that $q object from angular, If you would just try to incline jobs of codes, you would just be using the plain Q library, let's see it how far it is on a plain google search, there it is, like a form down, if you're using this library, just loaded this on your page and just put a capital Q object on the global scope, but since we're working on the angular version, I am using angular dependency injection to get it to me.

1:13:19 - 1:13:17 [students asking question]

Yah, if you've given an array of promises and thing it gives you is an array of the answer for those promises, the same order they are in the array,……. Yeah that'll be a tough o deal with right? ‘cause the whole point is you don't know if there are gonna be answers, so in the order you for them in. And you see, you get all the answers at once, so my A + B example if this is the time you cannot work with all the data ‘cause it's all right……… About what? Yah I am sure you can try the HTTP something where you have the course in there right? Of course, we'll get through this.

1:14:00 - 1:31:50 [The instructor is not audible, he's far from the speaker, and discussing about the next topic and assisting the students. And continue the exercises. Instructor answering students' questions.]

1:32:00

Instructor

These some codes somebody might want to look at, so when in doubt I will commit it

1:32:04 - 1:32:21 [Silence]

1:32:22

This code I would way, in the go. It's very manual, it’s talking about promises all over it saying the word promise repeatedly it's saying then repeatedly, it’s resolving and rejecting, that this code is very sort of tactical code, attacking its promises one line of code at a time. Real code in systems that

- -

use promises, it's actually a conference and the kind into my ears. They have. Real code tends to not do that, a real code in a promise centric system tends to do way, real codes tends to not say the word promise very often at all, it just sort of suddenly use promise. For example, angular has a time out service, it's kind of like set time out, but the return value is a promise which will be resolve when the promise is reached. You can call time out with a function, where is that, this is the thing where you test the, tell me how you pass the parameter, ooh that's ugly, I don't like that, delay, there we go, yes you can have a delay. You can go in here, and you can say, I am trying to ignore that, you can say that you want the time out service and that's like a, it's sort of like the JavaScript set timeout but it's more angularize it has extra angular stuff in it, and we talk about the notion of maybe having a time out for one of this things, and just to make it simple I'm just gonna, say, P4 the fourth promise, and I think time out is the function to call, here's the time out and then I'm gonna, I am just gonna give it an empty function and its gonna run after in say 500 milliseconds, this is just a promise that gonna be resolve it for 500 milliseconds and if I include that, now I need all these things to resolve before the work I believe I broke one of these, so I'm gonna put this back they will all resolve, so if I reload this now, just have to click there, there we go, now resolved, if were gonna say both there and, the longer accurate all are done, with normal JavaScript, you need to, you know wire up set time out, create promise right, it's awesome promise code appear right, I did a defer, and its promise and then I called set time out and all that but in, that just kind of thing you do when you're first showing promises you would never write code like what I have to read manually may have set timeout and may you resolve the promise and for timeout before the time out function built in to angular already is wired into the promise mechanism. So here I am using a promise without saying the word promise, the same way this HTTP you know yeah, but there's the success thing that you, you would never do this, this doesn't make any sense in the world when you have promises, what I really should do is I should call then there, treating it as a promise, right? But if it's already a promise I could just say that D1 is the promise or it should may, see resolving D1, I really, here I am manually creating D1 and manually giving a promise off of it, that doesn't really make any sense 'cause I actually don't need to do that because HTTP is already gonna give me a promise, I could simply say that, because, why is my clicking not working? It already is a promise I don't need to go and do a thing to make it a promise. Now when it resolves these things will happen, I don't care about printing anymore, I just took out one of the promises. Now what is this D2? This is the one that I am rejecting or that I am resolving with my click, right? Let's get this out of the way, now take off this, I would just demonstrate it here that you could put more than one, here, let's go click for example I am not aware of any built in way to make this thing automatically have a promise, but that means this is the case where I'm still gonna manually say Q to for, so gonna manually make a promise, it's pretty common in code were you, like maybe manually make one promise, but then your other promises this one, this one and this one are all just coming from some existing API. So you're gonna, even if you have to say defer a promise you often only save them a little bit of a time of I set this out of four promises on here I'm saying it once……. Say again? Line 19 and 20? Well, I have to hold on when you do that manually, remember this chart it comes in two parts, the part where you supply the answer and the part where you get the answer. So if you're gonna be manually resolving it, you have to hold on to this part if you cannot resolve it using this part, you have to hold on to this handle, it's got an in and out and resolve for it. To make that a little bit more clear I could put this code, next to this code See? The only thing I'm using D2 the variable four so that I can resolve it later when somebody clicks. The problem here is that, not an, this answer one is gonna be a giant beast of an object, somebody go run this and see there, so the answer to D1 is object object, see that thing at the top there? That's coming out of this line right here see that? Well, that's not really what I want, I don't want object to object to be there and let's say it has a thing inside that called data, if you get your header on this part, this is a promise, you've seen you can call then on a promise right? So you've seen this code we written already and you've seen that you put a function in here with some variable right? So that's all stuff you've seen so far but what I haven't told you yet is that then always returns a promise. So if you have a promise and you call then on it the thing that comes back is in fact another promise, these things compose together really really nicely. Now, I said it always returns a promise, I could write some code in here that would go make some promise and then return it, that would be fine, for example if what I really wanted to do is HTTP get something else it wouldn't make a lot of sense but it would at least be valid I can explicitly return something which is a promise, that's a promise if you call it again, if you call some other thing whatever, or if I really wanted to I could do something more like this right? I can go and make a deferred and grab a promise out of it and I could return that promise. [1:40:00] I said it always returns a promise though, so cool thing about the fact that it always returns a promise is if you don't do any of that if you just return some value, this is wrapped into a promise for you, which is pretty slick, this HTTP get returns a promise we call then on it, it passes that value when it's available through this function and then wraps the results in a promise so the cool thing is without changing the API, 'cause the API is the same it's always a promise API now P1 is in fact always gonna return ABC when it's done. So if I go run this, sees that the answer to D1 is ABC? This code right here, the P1, P1 use to be this promise but I change and now P1 is this promise, and the answer is ABC, 'cause I returned ABC, It's actually most common to build this sort of data pipe line in here, so this D is serve as an object that represent the overall HTTP call and D.data is the actual data from inside so now, the answer to D1 is all that, right? But remember I said that that's just another promise; well what you can do with the promise is you can call then on it. And then, it always works the same, so, the function here, I'm just gonna call this, I'm just getting into different, I have to give it a different number if it value then it get X just make that, 'cause the name doesn't matter. Well. the same thing applies here, so I could, I could do more, the results of this is gonna through that function the results of that is gonna through this function, I said that, that the thing that comes up there is the actual data of the page so I could continue with my, it's kind of building a processing pipe line here I could go ahead and yanked its length off there, and now this is gonna return a promise 'cause then always returns a promise and if promise is gonna be resolved with this data so without me saying a promise I'm building this chain of promises if you this it should now give me the length, it printed out as 1090. So what this all meant was, let's me see I can try to format this a little differently line them up maybe, HTTP get returns a promise run that through this function, run that through this function what we get back is a promise and I'm making call then on it so I didn't touch this code at all and I didn't touch this code down here at all, that's same value I could just, I could just build this pipe line when the data flows through asynchronously without changing any of the API around it. Now, it is pretty slick to though remember I said that you know, you chain this on well it knows chain errors through, like if I wanted to I could use that then syntax roughly you know, multiple functions but if I just don't do anything at all I automatically get chaining of the error so if I do this, the combine fail, P1 failed, it failed because this promise rejected 'cause the HTTP got the 404 and this chain promise rejected passing the error through, this chain promise rejected P1 was set to that time of chain promise which means that this part of the then got called 'cause this is the rejection handler, and then this was also combined into this all promise and made this one fail, so failure propagated all the way through its enormously easier to get error handling right in that complex match of asynchronous operations this way, giving this kind of error handling right by typing a bunch manual callbacks is just incredibly tedious, if you want some amusement you can watch twitter as people are learning how to use node where you like writing everything in JavaScript from the server side like you see some already discover it, you see some frustration, you see them discover promises, you see them say I really don't like code to use promises, that's where the cycle people go through to using node. More variations here, sometimes I find it helpful to draw a little diagrams, is anybody remember this diagram? Anybody written it down and like scraps it somewhere Sometimes you use to, kind of like draw little boxes or mark or something and kind of show the legends between them, so I have a promise of call H meaning my one on line 12, right? and then the alter of that pipes into a function so that next then I'll call this one the data1 it's just name I'm just a kind of give it a name and that pipes into a length1 and this is the one we call P1 it's a little hardly get that straight because like on line 12 it looks like P1 is the top promise just tell you kind of read the code partly and understand that thing has a function called on it so that whatever returns here will ends up in P1 , Is that clear? I know it will be better where to type it to make it more obvious that just sort of, reign full of ugly languages type, Ok, and what's that P2 do, so P2 is one I made by hand and when I call this to click one, P2 is the click guy. Now what's three, three is another HTTP1, three is this HTTP call, on line 39 this is actually returning a promise, see? Now when I said it doesn't matter, you can just abandon a promise, it doesn't hurt anything. I said then take a promise and you call then that always returns a promise, well the JavaScript of function if you don't explicitly return anything you just return, is it null or undefined, that you get, the numeric number, If you don't explicitly return in JavaScript, is that undefined? Remember its null or undefined, that doesn't matter because this thing here is its through that then its creating a promise which is when they get resolved its either null or undefined it doesn't matter 'cause I didn't hold on to that promise I just let that one fall on the floor. And then P4, this is the timeout so I have this timeout promise going over here and then I have all four of these coming together and just this all promise that I build out of then. In this code there is, one, two, three, four, five, so six promises that actually flow together and then we see these other than those are actually creating promises and they're being draw out of the floor, right? 'cause this one here, this promise, that landed right here got sort of a P1, but this then this created a promise who just abandon that one, and then this guy here he, created a promise we abandon that one same with that one, so there's four more and this, they are actually, you have to call thens, this another promises available here to unchain another one unto that, but so that's actually five more promises so in the code you see there I think we only see the word promise once is that right? On line 30, anybody see any other explicit mentions of promise? You see the word promise once and there's one, two, three, four, five, six, six promises, or seven promises that we're using explicitly and then there's five more that are being created with the trapping of toward that using so it's actually 12 promises in place and this a little example...... I was gonna like, illustrate here I guess sort of illustrate it all working into end. I think it's waiting for my click, then they all resolve then we'll see all the different parts of the answer off of that. There's one more interesting bit that I will paste in here, it's kind of a, challenge to understand. Go grab on another on one of my examples. There we go. Now let's paste the code in here. The name is a hint. Anybody guess what thing thing does?

1:50:06

It's ok if you can't guess what it does, I really like these things that end up with this kind of perfectly symmetrical indentation, isn't it nice? This is the kind of code you end up writing when you're using JavaScript in a very functional style end up with code like this, because it seems it just endlessly creating functions and returning them, their kind of parts this so we've seen what factory does. With factory you get the outermost function that you use for injection and then you're supposed to return something and that something is what people do get the factory injected will get, so this return right here, means that this is what I will get if I ask for KCsleep. So what will that, that will that, that's a function right? You're gonna get a function so the thing you get you should invoke, so I should invoke this KCsleep, well when I invoke it that should be in hand passing it milliseconds, well, once you'll get back from this function is another function which takes a value, and then it calls timeout with that value and then returns the same value after waiting some milliseconds but I already told you that $timeout returns a promise so, when you call this function you get a function which been called at a value returns a promise. So that's a lot, that's a lot to go through and I kind of put this up here this kind of a previewed it like two months from now this will make perfect sense if you use this between now and then. But the off chart is that I ask for this to be injected like that, ok? And then it has this great property that it already looks like this, it already looks like what is in these guys, I could take P2 and chain it on so P2 I could say equals P2.then of this function. I think I'll call this P2 prime so I'm reusing the variable name and can hard to follow, so down here I could to that instead, what this is intended to do if I haven't made any mistakes here in the copying and pasting and typing is that when I click this button it will resolves D2 which resolves P2 'cause D2 is my name for the deferred P2 is the promise. Right there. But, I called P2.then appear so it should immediately print something and should immediately call this, well what this is, it's this function what you call it and it gives you back a function because then expects a function and its purpose is to take value coming through and wait a thousand milliseconds and return a promise that will resolve after a thousand milliseconds. If promises are the analog of function call returns that asynchronous, then KCsleep is the asynchronous version in the promise world of like a sleep a thread sleep in JavaScript, and it's a JavaScript does not have thread.sleep, it doesn't have thread and there's no way to tell it just stop. But if you have a series of promises it's actually really easy to just understand the mechanics to put a weight in there. Let's see if I got it right. Usually takes tries to get it right but resolve, yup, see how we did when it came. Let's make it more straightly obvious_Out of 5 second, right, I click and it will go immediately print out from that promise resolving and 5 seconds later the other take the all promise finally resolves. I would not necessarily recommend an animation technic, but if you find yourself wondering yet how do I just put a sleep in here I can't do it threadsly, people yell at me if I just like burn CPU trying to sleep, well if you structured your data, dataflow as a set of promises it's like a pretty easy way to make an abstraction that makes it really easy to put a way then. And this is eminently reusable I could take it and put it, I could say as part of this, here's the HTML one right? Remember how I said you can just chain this together all you like so I could just say, before I chain that thing I want to chain an extra 2 seconds in So that HTTP answer coming back on that, usually once people play with this they say, we'll tell you what you gonna do we're gonna put this in version one of the software and we'll teach the version we're gonna reduce the amount of extra delay, so it'll, perception that our software is gonna be faster. Let's see if that compose properly. There it is, now that D1 I just, without changing any other code I was able to just state a sleep in there. If you are writing with timeout and manually nesting them and you wanted to add another step you have to go and radically reshape like indent a whole bunch of code, another indent and write another function rough around it with promises you sort of stand a straight line and just add another then in there. The next step for everybody you can mess with this if you want it's not important but like everybody just with playing chaining a little bit so chain a function through, So somewhere in your promise you probably have a chain of promises somewhat like mine here, use then put a function do some processing to it. If you wanted to for example the pretty easy to fetch two or four different URLs, take the length of each one, this kind of a process doing all and print out the total length of all the pages you fetch that kind of thing, and that would be exceedingly tedious to do without promises that any variation

1:56:34 2:17:45 [Students are working on their computer while the Instructor is assisting them]

2:17:46

I'll continue reasonably good hygiene by committing my progress, I must chaining examples. A question was raised I think by Adam right here, wait no, who raised the question? wait, I don't remember your name, you raised the question, so the great question is, should we be fetching this data is some services or something, the answer is yes, if we go to a simple controller case it's right up here, so I'm gonna take all this promise stuff for the moment, so I'm gonna say okay well this is

- -

find this is all the stuff we learn but I don't need to know all this right now right? That's all gone, the sleep thing I'm not using anymore I might use it again, I'm not using timeout anymore, I'll leave it queue there 'cause I know I want it, let's look over phone maybe. So over here I moved this to a separate JavaScript file phone stuff, This is doing the thing were it says scope phone and its having it words hard coded. Right? And we learn the best not really a great way to do it. Which are we really supposed to do is fetch this data so with the previous example that Don came in the box was something like an HTTP get of phone/phones.json I believe, and then I think it called success with the function and this data is already decoded that says scope.phones = P, so I think something like that is what the example code did so to see if it works, see I have my phones down there if I reload here and HTTP is not define I get those errors all the time but it's now that that's an injectable if you want it you have to ask for it through the like of Java import. Now I have a whole bunch of phones down here. There's several things wrong with this, one of the things wrong is that you really should be is in the promise API, like I should be saying then there and it's okay to use success but if I want to able to manipulate it and chain it and all that stuff, like if I said then instead here it's almost equivalent the only thing where it's not equivalent is that the D that gets pass is you use then is that bigger object that contains everything about HTTP request and you have to explicitly dig the data out of it by saying that data if I do this I should get the same result and before, see? Same thing. Is it reloading? It's so fast I can't even tell. It's interesting though that it reloads so fast that I can't tell. And we have a way to make it reload more slowly right? Because we just saw some little utility thing, And you guys are just welcome to copy this utility, sooner or later I'm gonna just put it on a webpage somewhere or something. But I've defined this KCsleep thing and... Well, the way this works module, factories, controllers all this injectable are actually global across an app, the fact that I defined it in this module that does not in any way stop me from using it from this module, if I want to use that from over here I can just say, I need one of those from over here, and then I said I could chain, right? If I do that, [2:21:38] promises here, I get all that chaining stuff back, 'cause I really like that chaining. If I do this, I'm sure I'll get a little bit of a delay before it comes in now. Any questions from what I just did. Questions so far? They already do enough of promises in your example to follow that. That was one problem, is that it's really; you really not supposed to call this APIs directly once you understand promises, like the success and its kind like your training wheels. The next problem is it's an abstraction problem here because controllers are supposed to be about setting up scope to make this interface work. And what I am doing here is that my controller has codes that's talking to a back end and understanding how the back end works. That's frowned upon the angular world its actually totally fine for example in a small app but what you're really supposed to do is use a service perhaps using the factory keyword. You're supposed to move that off of their, that's the more abstracted way to do it. It's to not let the layer of your code that make it brings the GUI to life which is what the controller does shouldn't be the same way the order code knows how to talk to a back end, start making non trivial apps that everyone rediscover that promptly If I want to edit thing that's kind of spread the code out and follow the standard way of typing it, make it a factory, naming is one of the hardest problems in computer science so I'm just gonna call mine phone data. There's a convention which is if you're making things that maybe like distributed or part of like some big shared thing then you want to make some kind of like two little prefix, somehow two little prefixes had become part of the angular culture if you're making something as locally or project people usually just skip the prefix. You're not required to do and there's nothing in there that cares, you can use a three letter prefix it doesn't care. So then you have sustained a function and this purpose is the first function here is always to do injections. That's an injection point. And then in here I'm supposed to return something, whatever I return here is gonna be what people get when they go to the factory. This is an injectable so I will put it up here and then it can have its own injectable so I will take these and move them down here, so I'm adding a layer of indirection. Then you were to see the problem of this code, it's a really obvious problem of this code, it's on line 14, you've already spot it? Yeah, so what I really would want to do is make this thing return scope.phone I could really lot adjust that, but I can't do that right? 'Cause this is code that's gonna run later. Well, the answers actually to do the digging out in here, so this is another prompt, this is the promise I'm chaining on my sleep and I'm chaining on a function that digs the data out of the phones 'cause that's the only part I really want and so this is gonna return a promise right? I could go right here, that's what I feel like I want to write, but I can't actually write that because this thing is a promise, and we know what to do with a promise right? What we do with a promise? Call then on it. In there, we could do this, let's see if that works. It's pretty big change, I'd probably broke something.......So this is a step four 'cause we build some abstraction but it's kind of messy, right? Because these things always return promises, like when you push this layer out and it knows how to talk to the outside world it's always gonna be a promise though, right? There's never gonna be asynchronous way to grab the data from the outside world right now so if there's any chance that you ever have to talk to external resource the only safe thing to do is to return a promise 'cause that gives you that ability to be asynchronous. But it's kind of messy to make this controller care about that being a promise, it's kind of ugly well there's a mechanism that's kind of write at hand but it's only a hand if were using a routing. I need to take a detour here and put routing in here, does anybody remember how we put routing in? Yup that's the first step. Angular route, what's another thing has to do add routing? Let's go over here and declare dependency on ngRoute. Then what's the next you have to do to use a route, you have to configure my routes, right? Let's go configure my routes, on the cheat let's go looking out I did it on the other one 'cause I don't feel like retyping at all from scratch, it's gonna be something, somewhat like this like a little bit last time this time be able to different. Somewhere here my application I need to configure my routes and from what I'm doing it can be like the worlds simplest route, next is done here 'cause that's part of an ad. Well, I guess if you want let's find URL to use and redirect into it automatically that's pretty convenient, I just need somewhere to put this thing and I call it partial1 last time and that's, like it's as good name as any. What I need to do is I need to take all this stuff that I currently have in the body and out that all into something called partial1. Especially hard to look at the right project over your head…… It's like move this off for this code expects it and now this thing, let me see here, expects are gonna

- -

be its either a controller listed here, a controller listed there, you will need to have a top level controller to use routing 'cause you want to build list of controller here so want I'm gonna do is I'm just going to assert that my top level controller is this one that means I don't need to put the controller here anymore, because conceptually the whole thing is getting wrapped in a controller and I happen to know that it's totally okay to have another controller inside 'cause we already established yesterday that these controllers nest really nicely. If I have some conflicting names on the scope I might have some interesting happen but I happen to know I don't 'cause this one only for phones in the scope and that other code doesn't care about phones at all so I know this is the safe thing to do just let this nest. Now I'm using routing if I start without being on a route I'm gonna go to view1 its gonna load this template which is gonna be this HTML and that's gonna be stopped in to where the ngView landed. [2:30:00] get ngView, there's the mark where it lands so all of this will get injected here and this is kind of nice 'cause that's suffered in my page that's all about loading web access for my page it's about the specific templates so I actually kind of like that change divides itself a little more modular way. I think I configure this correctly, so let's see if that works, it probably won't there's probably some mistake in there somewhere…….. This is the hint right? Right there is the hint……. So that works. So it's quite a bit transforming around, so that's transformation I just made right, it's something that wasn't using routing and made it use routing, like everyone goes through that transformation if you start a terminal project just graphing your angular template stuff right in here and as soon as you enter to anything non trivial you immediately find that you need to actually put it in a partial and used routing. Follow me so far? The last bit to add on here is real nice and there's a great feature as part of the routing called resolve, so if you say resolve you pass it in object you give a name value pairs here, you can give it a local name, I'm gonna call it pdata and you can map that to a global name something registered so this is saying go find something if after your service and in that global list of all the factories and services called phonedata and go get that data. If it's a promise wait for it to resolve. So this will delay the rendering of the template until this is resolved and then pass it in and inject it as pdata, the answer not the promise. This is a way to declared ably saying here's a source of data, here's what I want to call it, here's the controller I want to give it to so I go to that controller and instead of asking for phonedata I say it's gonna be called pdata but pdata is the answer it's not a promise anymore. Remember I said the more pervasively you use it the less you talk about promises. I don't have to talk about the promise anymore because the angular machinery in that router dealt with the promise. I can just say this because my controller will not run until after the promise is resolve without me having to really work at it. So my controller then goes back to being utterly trivial, this guy here is making a promise without talking about the promise it's saying then a couple times to chain it then there's declared of code saying that for this route I need this template for this controller here the data sources I need for it of which is only one. And then I just have to make these names match so this phone data have to match where the data is coming from and this pdata have to match where the data is going to inside the phonelist controller……… It actually won't render and there's a way to put an angular unto go somewhere else instead because it failed, you see here instead of rendering and re rendering, it waits for all the data and it renders. And you can do more, right? You guys are going to build a real world page and a complex isn't see app, you're probably not going to have this one little kind of data, we could ask another kind of data on page len data and maybe this thing doesn't use leak 'cause it’s really fast guy and it maybe talks to indexHTML and then we don't to do that chain but if I remember right if you pipe it through ddatalength that's how you get from an HTTP result to the length of that page so this guy here will return a promise which will resolve with the length of indexHTML. That's kind of unimportant code but here's the nice part is that this is starting to take on a very nice declared of flavor so I commit my ldata here, this is not a naming conviction thrill I’m just gonna typing something so I can recognize the name, so now I can make a list of all the kinds of data this page needs and somebody else is responsible for writing code to go make all that happen. Then find a template maybe right here. Let's see if I got that right the first time. No, something wasn't right. Length came up nothing on it. I wonder what, whenever I wonder what I tried to use the angular tools here to find out…… So if I poke around here, I can probably enable this inspector and then press this guy to get a scope, trying to figure out that's phone does on that scope so maybe that's the scope that has phones outside of it and it looks, no this is the one that has the names trying to figure out what happen to my len looks like my len maybe didn't make it on there so after phones, I have len, I don't something weren't right. Did I not reload the page? What happen?....... Yeah, so my

Page lendata it should resolve what the length of this file and then over here. Page ldata resolves into ldata and that’s gonna be a local name which is gonna be injected to the controller which is here so that came in there and that should be gone to scope.len which should have been on the scope 'cause this is part of the controller, this is the outer, something weren't right here this code I’m suspicious of, so what I could do is just to see if I’m working it all to see which half of the problem, to see which half is broken basically I could do that. That much works, so I know the binding is right. That means I must not be having the right thing coming through here. So this is somehow not arriving with the right data, maybe I made a mistake here, its namevalue namevalue so I have those names and values right if I have done a misspelling here I would get an error…….. This will throughout the HTML file that’s not locate in the js folder and plus through an error the route would not succeed. So it's a factory just like it is a function HTTPget, I think that's right. By the way there's another way you get, I don't know guys if you are familiar with this. If you say debugger that's an object, and that means it will actually stop right here in the debugger. Foremost a pretty good source of all debugger in it and so once I’m here there's all kind stuff I can do. I'd like to watch expression, let's just watch d what's that look like, so d is an object so this point I’m watching d, so it's looking for d as right now, so that's another d of this d and then I believe this is an object inspector, it does have data and data does have length, that much seems like it should be fine, I don't see any problem there. What I can do here is I could go over to this guy and I can set a break point, now if I continue running, it should get up to here and then I can add a watch expression to see what ldata looks like and ldata is 464 that all seems fine. Now, I could actually set a break point here and I could let it run again and at this point I could actually do a watch if I want to really dig in here a little bit of a time.

2:40:00

The scope.len, this all looks fine, the problem I thought I have there I don't need I could remove these break points here 'cause otherwise it will stop every time and to see what happen, wait but now it worked, what did I change, I didn't change anything did I? Maybe I haven't saved my file before is that possible?....... Debuggers in a way actually have typing something into the source that will cause it to stop at that point, where is that guy? Generally speaking you rather set your debug marks in the debugger but there's just this back door way of putting and typing line of the source, yeah it works now. I must just not save my file it's a most common thing. I need to learn to press the save all keystroke on my editor. This illustrates the point I was trying to make which is that a controller should only be responsible for setting things up for the GUI it shouldn't be doing things like talking to a back end, illustrates that most of the time you really shouldn't be manually dealing with resolving the promises for the controller, although there are cases were you might, depending on the page or building you actually might want a page that populates incrementally as the data arrives, like if you want that you know how to get that but now you also know how to get the behavior were it waits for all the data and then it populates the page and then there's a very nice construct were this starts to look like metadata bulk of program 'cause it's very declare. You know where this routing thing I set this before, before we have resolve that the routing ends up being sort of a map of your program so this template is as this controller in this route. Well, you can also have that mapped include what data flows to what part of your program, this mechanism…….Yeah, this is a way of wiring a service to a controller and instead of directly having the service know about or instead of having directly having the controller know about the service, you can add a layer of indirection here and the word resolve is your hint if this things return a promises this will wait for those promises to resolve. There's an implicit q.all in here happening behind the scenes. This mechanism only expects this guy to return a promise, but if you wanted, you could do it in a more generic way and you can wrap this and have return like some read methods and then you could simply dig inside it and call then on it in here and there's another alternative, you can type the function right here, this can be a function that returns a promise so if you don't mind this guy starting to get for both you could actually type, you just have a function there that gets phonedata and then calls a read method on it. 'Cause we're inside an injection point so if I wanted to get phone data available I could put it right there and I'd have phone data available and then I could write in here, we said phone data is a promise so this, that should actually work the same as before, see if it did. I might have missed something up there, failed to expatiate a…... This config happens at the initialization and these services are not available yet. There's a slight bit more work around you have to do, to do what I said, but I don't want to be late for lunch...

2:44:22 - 3:53:00 [Class took their lunch. Reassembles. Students starts asking questions and instructor answers.]

3:53:00 The answer for that is yes, because you can call this config from anywhere so I can go over to the phone and I could say that I actually want to configure my phone stuff over here and I could even say that I want to configure only the thing related to the phone over here and then in my main file I could put only my otherwise so that way all the stuff would have to do the phone including the routing, that prevents the phone to beat up and put over the phone, so if I did that right which I sometimes do, it should still work, there's an error down here I didn't do it right. Let's just puts this back to the other... Unknown provider route provider. That's because this one needs to just refer this guy and it is okay to refer to it from both that doesn't hurt anything. So to answer the question, yeah. And you really should do that because if you have, if it starts to becoming pretty complex, let's say there were five routes pertaining to the phone you really would want to push those to the whole section of the application put all your phone stuff in there and namespace it all under phone. And they all just compose together, well the rule is you can only say otherwise once if that's what you get if its unrecognized route, so you should do that in some central global place but yeah you could dobby up your routes in a way that makes sense for the modular to the very application…… The main convention that I like is to I think I say this the other day, I treat those like a Java package name and so I, like this should be in a directory called Kyle and a file called phones, maybe more like a Java class name, and put its conceptual like a package and so this would be in a directory called Kyle and file called phone so that's how I recommend placing the files, you don't have to do that it doesn't care at all because the execution model for JavaScript is just depend on the JavaScript together and put it on the page so it doesn't really matter but just to make it make some sense I tend to do it this way, there's people will do I all different ways. I mean if you wanted to you could actually divide up each of these things into a children file f really want to……. I don't do that because if you write this in a very modular way that tend to end up pretty short and I don't want to have a thousand tiny files that's hard to manage enough with consists of a thousand tiny files. I'd rather have a hundred modestly size files. Any other question?...... So angular.module this is creating and angular module named kyle.phone that depends on an angular module called ngRoute and when it creates it, it returns an reference to it, if you have a reference to an angular module you can call a bunch of different things on it to register various kinds of angular, I called my entities or things 'cause I don't know if they even have a word that means configs or controllers or factories or whatever but these chain, and so this config returns the same modules so you can just keep chaining more calls unto the same module and so it ends up looking a little bit like you use Java, it starts out with a basically this is like a package whatever and then this is a some imports and these are maybe conceptually like classes of a not really you can have a bunch of them in a package, and you know they're not all the same kind there, this are four different things of three different kinds and then the dependency injection that's sort of as if you were using spring annotations to do injections as if its name based. There's kind of analogs to what you're doing in Java that make it maybe a little easier to understand than if you started doing something totally adhoc. For the size of the application I'm doing here I could ignore all this and just do it completely adhoc it wouldn't matter but intentionally try to follow this maybe you know full fledge to way have typing the files just to tell people they'll have it doing it that way……. So where were left off? Was to have everybody take what you learned about promises and in your own little sample app that does whatever you want load some data in a way where you use an outside factory or service whichever you like, that loads the data and use this resolve mechanism to make it available to your controller I think a benefit of moving this over to this file is that you can see everything…….. So the only part you need to see on the app is this otherwise we'll flip back for that but for reference what you need to know is here. That would really code that. Take you sample you're working on and maybe start moving it toward a program that does something, so pick some little something you want to do and just maybe type a couple lines of fake data in a json file and load it with HTTP and if you need to transform it in some way use method chaining to transform it in whatever way, use resolve to get the data it your controller. There's a whole bunch of name matches to get right? Paul and I will help you get them right, like you have to you know, this name has to match this name but this name has to match this name, go for it.

3:59:17 - 4:55:00 [Students starts to work on another exercise while the Instructor assist them and answer their questions. Class also took a break]

4:56:00

So my app js is pretty simple I have this little fun thing in here, I have the empty controller and barely even using the only thing interesting here is the otherwise on the route. I was just illustrating here the idea that it makes more sense to define the otherwise in case you'll only have one in a global place even if you define routes for specific sub modules in the sub modules place

4:56:22 - 5:01:44 [Student continue to work on the exercise while instructor continue to assist them]

5:01:45

The way I showed this I wanted to use this maximally concise syntax for resolve which means that the thing on right side here has to be a string which is the name of a factory or service to return a promise. If you didn't want that conciseness and you want to make bigger factories that had multiple operations in them, here, since I'm gonna break things this would be a good time to commit……. Use routes and provide I like to commit before I break things. Let's break things. So you're saying, what if you wanted this guy here to instead of just directly returning the promise what if you wanted this thing to return a bucket of some kind. And it's gonna return a number it maybe has multiple operations in here, maybe it has an operation called get. And that get itself is a function and maybe that function returns the promise, so this is the scenario you're talking about?..... The question is where do you write get up here right? I think one answer is if you want to do this a lot, like if I was doing this a lot what I would do is I would make my own custom router that had the ability to take like do like something like this. So that's what I would do if I was doing a lot but this router doesn't have that capability so what I would do instead is take advantage that this router does know how to put a function here that's how to tell her to function here, you can put a function right there. And in that function, I'll just leave that text behind that function is one of the very few places other than here and here [5:04:00] that you get to injection, so if I do that now it just got phone data injected well that means I don't need to have that in quotes here it means that I have, I actually have phone data, well if I have phone data, I know that it has a get that I can call, I believe the only thing I'm missing is that I need to actually return this thing not just call it. I think that is efficient, let's try it. Yes, so that's the answer to your question, if you are working on a more normal width of an editor window, you weren't you know, obsess like I am with trying to keep this all within a narrow column and you're willing to do a little bit of ugly formatting you can still kind of argue ably put it on one line like that. The only thing that you really need directly expose for resolve is the load operation, the read operation...... I might well just make ones that return the data for the read operation and then just do something else if I want to do some writing somewhere else, so I wouldn't necessarily try to combine the read side and the write side because the syntactic overhead of doing another one is basically one line so I can make another one with one line so it's not like it's a major undertaking. So like I'm putting each factory on its own file, I mean if I was I might be more hesitant to make more factories but I'm more inclined just make one that already has the right behavior, did anybody make a view that took a parameter? And do you want to know how to get that parameter, did anybody find how to get that parameter to return database on it? The answer is for me you've already seen you can just skip the routeparams down here and then look at routeparams to return data for the correct phone or if you're doing this approach you could have routeparams injected right here, and then you could write code in this like right in here if this thing needed to call get or like if you want to call get for a specific one I won't make this complete to not to run it but you could do something like that, and then I send of this is to keeps it the definition of id in the use of id, just a few lines that code apart so its lot easier to make a names match if things are not to each other these are all right at your fingertips....... If you're going to directly data bind it, you just stick in the root scope and then it's immediately available for data binding anywhere in the app 'cause its scope inheritance. If you want it available to your code but you don't want it to be always data bound then you can actually use a factory, if I wanted to I could do a bar in here and I could populate this I could actually catch this thing coming back in there so you could make a factory that the first time you invoke it it loads the data and all the other times you invoke it it already has attached. You could go grab a library that somebodies written that creates data bindings that know about local storage where you can declared ably just say that this thing on the scope should automatically store in local storage also and then when you it'll automatically leave after it the next time you some up so there are ways

- -

with very little code to do what you just said. You're saying if this failed for some reason right?..... I think right now I'm not doing any error handling you're just seeing this happen, so this is not really helpful. There's a couple different ways you can do it, one is if you wanted to handle the error handling like as if it was a success so you'll not talk with this in the hallway. Let's say your loading ten times of data on the page if you would really like it to work if only nine of them load then what you really mean is that you don't consider the failure of one them to really be an exception that should just kill the whole thing you consider to be an acceptable thing, you could go in here and you could write a [5:10:00] then and the first function could just be empty and the second function which is the error handler this could return a placeholder of an empty array and I think if I do that right then if I fail to load it'll just come up empty and if I could have leave and put something else to pass back to put on the scope to indicate the failure, so this is the way of kind of failing forward so if you want to fail forward you have to use a construct like this, this is that then idea. Remember the first one is success the second one is failure but this much I put exceptions once you catch an exception you then go forward from there, you no longer exception handling you're running again and so the thing that I returned here went forward it didn't bail out. But if you didn't like that, if you wanted to have the whole page fail, then, I don't remember the exact syntax...... Find resolve and success. The other that you would typically do is you could catch this on route change error, we may haven't talked about this yet but I will point this out. This is broadcast of any of the resolve promises are rejected, I could put an event listener and we'll talk about event listeners on one of topics, as if for today I mean for tomorrow we'll hit event listeners, there's a global event mechanism so could catch that the intention is that you could make a global event handler that may be redirected to a different route if something fail completely or maybe grab the error and logged it or post it up to a server somewhere if you're essentially collecting client side errors which is a really good idea by the way. There are means to do that. If we have time when we get to event handling we'll catch this error instead of some random error that I make up……. The suffering is over for a little while get to talk about stuff. CORS, Cross Origin Resource Sharing. This is the browser, there's two tabs open, two windows if you want to be call it. This is mybank.com, this is evil.org,

5:13:25 - 5:13:30 [words inaudible]

Joe user innocently uses one other tabs or windows to log in to mybank and uses the other tab or window to go to evil.org. In the early days of the web the following was possible, you would log in to mybank and you get a cookie coming back and that cookie would say something like all=true whatever because some kind of indication that I am logged in to mybank and allowed to make calls to mybank and then can your other window you could evil.org. Evil.org could serve you arbitrary

JavaScript is that called CORS, I mean you go to a page and it says js or whatever script source it calls whatever and arbitrary JavaScript comes in to your page and so in the error that is the web before the same origin policy. Once you were logged in your whole browser was logged in and so if any like some JavaScript here try to talk to this guy the cookie would flow and will be okay, so if you open another window it could actually talk to your bank logged in as you, that's bad. So it's abuse badly in the early days of the web. Somewhere in the earliest days like just IE and Netscape somebody came up of this something called the same origin policy. Now it was too late to apply the same origin policy to most of the things that we do on web pages, on webpages you do a JavaScript SRC CORS or image SRC CORS. You can make any page load as some any other program on the web. But the same origin policy applies to the thing that is called XML HTTP request and it's used to the things like this to talk to the services from webpages around the time this guy was coming in, we'll figure out if they have with the same origin policy and xml HTTP request is subject to the same origin policy and what it means is a page can only talk to the domain about the origin which is the domain name and the port number and the connection type, HTTP and HTTPs. A page can only talk to where it came from, if you put JavaScript code here that does an xml HTTP request that works, if you put JavaScript code here the tries to talk mybank that is actually blocked by the same origin policy. Then you put to see the same origin type error trying to load a page so that is piercing. The thing, like if that wasn't there the web would be terribly broken more so than it is the same origin policy is a really good thing we couldn't do a secure web if we didn't have it. The problem is that sometimes it's not mybank.com and evil.org sometimes you're trying to make something that does this on purpose, like agreeing to. I don't know what's your domain name, you can easily imagine having some sort of page f.sm.com you load you app off of and then having some kind of like API server that you will load some data off of. You guys probably already wanted to do this in what you've been working on, right? You actually want this while process to just work, you want to be able to have, let's erase this and make clear again You want to be able to have a webpage that the JavaScript comes from here but you get the xml HTTP request to it from another server. The way you turn this back on is called cross origin resource sharing, the reason you can't do it is the same origin policy the way you can say you have permission here to do it is CORS. It's pretty easy to understand the mechanics of CORS by reading the webpages about CORS. If you would go on Google and type CORS you'll get to read all that. The thing that I found really had to understand is where you're supposed to put the CORS headers stuff, but where do you do that stuff. But if you reason through it it's pretty obvious where. Remember the previous example , well this is like the good side, so this is maybe, this is like the site where the page came from and this the site we're not supposed to talk to just maybe I kind of reverse them that and drop before. So imagine this is like an image pass them is actually like a bank, so imagine you're actually a bank. What would happen if it was possible to put something on the webpage saying this webpage could talk to that, what would happen? If there was any possible text you could type in a webpage to skip pass the same origin policy then evil.org would put that text on a webpage. That's why you could search all day you'll never find anything you could put on a page to turn it off 'cause if you could turn it off the evil could turn it off too, there's nothing you could put on the page. But more over this is the [5:20:00] server that the page is served from, but if this page could turn off CORS by serving some magic header then evil.org would serve that magic header. You can reason that, that means that the server that you're serving your application from there is nothing it can do to turn off cars either. The place where you can turn off CORS if on your API server, so these CORS headers your API server this guy here has to serve CORS headers, so that means that this CORS header thing is going to not be a JavaScript problems. It's gonna be code you write in Java or Apache config you do, something on those lines to make your API server that you want your pages failed to talk to, to talk to it. The good is it's pretty easy like once you know a parameter to use in everything you go find something on stack overflow that figure something you have to paste in the server filter, paste some stuff to copy your page and serve the right CORS centers. It's not difficult to do once you understand where you do it and the answer is on your API server. The interesting bit there is sometimes you want to talk to some API out of the internet. If people want their public API to be runnable to arbitrary webpages then they have to serve CORS centers. If they don't serve those headers it cannot be run by arbitrary webpages and you could run it through the Java program on their computer and talk to any server you want, it's not a restriction what you can do locally it's a …it's because they didn't think of that, they didn't think I want this to be called from a webpage, it came out that maybe API and everything, you're gonna call this from a Java server application who knows what.

5:22:10 - 5:23:03 [Instructor and student discussing something, Audio inaudible]

5:23:04

They have an official answer pop when they go back to IE and I forgot what the number is. They say that there's a next major version are like moving up one on what IE they support but promptly speaking, yeah, as if today it supports the right version of IE…… We should talk about IE took a different approach to the like what you have to do to make these CORS locally they didn't make it probably transparent to this stupid cross origin thing you have to do and so that should be somebody else' problem you shouldn't have to find that it's annoying you have to find that but if you use the right abstraction over it somebody else is already got it. I'm actually surprise you have to find the jQuery seems like the reason to do the jQuery what if abstract over that... You guys probably support at least chrome, Firefox and IE to your terminal apps. If you're within the IE versions that angular supports, which I don't have in front of me, you should not have to do anything to make that work…… Right now it supports IE 8 or later which is pretty good, you said IE 9 already had trouble, if I remember right I don't know if it's on here yet but for the next version of angular the core angular came is gonna stop caring about IE 8 and only test on IE 9 and above browser security restriction. If you come across some sites that got a fancy API but you can't get it to work on a webpage because of this but it sound like you're already saying IE 9 so it probably won't affect you. You shouldn't have any problems. I was hoping, as if there's a bunch of stuff about how HTML works or directives, we'll see all that in a bit, but I don't see anything here about xml HTTP request but as expected does the right thing. So any more questions about CORS, I can't really do the programming to show you how CORS work because we're all about the web

- -

browsing part before angular was. The same origin policy does not consider a file URL to be the same origin as running something from local host. In the old days about development would be very frequently just like directly open an HTML file in our browser, you really can't do that anymore you have to run a local web server and load it that way we get to work. If you open up by dragging it into the browser or something and you got that same origin error that's why. There's actually, there's nothing you could serve that will get pass that because there's no way. If you want to live dangerously, just to try something and I don't recommend this, remember these words, search for it and you'll find people saying, if you go run chrome from the command line instead of double clicking and clicking on something in your windows, you can pass this disable web security and that turns off the same origin policy, don't do that 'cause it never a way you forget you did that and you'll like start going to real websites and logging in to them and you'll be totally exposed. Don't blame me if you do this and forget you're doing it. And one thing to never ever do is go modify your chrome short cut to always run at this way.

5:27:51 - 5:29:29 [Instructor asking questions and discussing the answer to the student]

5:29:30 We saw a thing called $resource, for a long time $resource was completely unusable, resource it was broke in various, now it actually is useable but in the intervening time almost everybody I have talk to move to using this instead the thing called restangular. If you're trying to make a restful web service running behind angular app give this a really good try and manage to get to run this and just kind of point a few things about it, it's not part of angular it's just something this mgun till guy I did.

5:30:14 - 5:33:11 [Instructors chatting with students]

5:33:12

I'm not going to use restangular here in class because it's not part of angular, it's not in the box it's just a third party thing but it's so commonly used I think everybody should know about it. Here is why it's so slick if you're writing like if you have a restful backend you're trying to make restful URLs, then there's a sort of pattern that everybody follows. You're probably seeing URL that look like this before, maybe you have a user's return to json representation list of all your users, maybe users ? firstname = k, it returns all your users with firstname starts with k, users 1 2 3 returns a user with the id 123. Users 123/reservations, returns all the room reservations along with the user 123, is this seem familiar to guys working on json or restful backend. If you're using like this sort of pattern something that's general family then using this guy, you don't ever have to do spring concatenation to make those URLs, 'cause really clever thing you set the base URL somewhere then you just say that and if forms this URL itself, that's why, it returns a promise and so they called then on it but we already know we could just say return in front of this and the service and have the whole thing we just spent 45minutes finding work with one line of code the slick thing I that when you get one of these users it remembers where it came from in terms of the restful URL, if you take this thing you called getlist on it it will know that its URL is that and it will know that is you say getlist something on there to append that. There's no code written here to form these URLs, it just understands the pattern. Here's a much longer example, let's say that you have some user on your scope and then you say this one pattern here and it forms this guy, it forms the messages 1 2 3 and then this from thing forms this and the get list on the red does that, it's this like called the fluid pattern. And there's other more variations on this I just show a few examples, as said I don't want to dig in to it 'cause it's a pretty big thing and it's not the part of angular but you should definitely look at this and if the URLs and your restful API are close enough to this common pattern it should say you quite a bit of piping. What do your guys URLs actually look like?

5:37:06

See they don't show the definition that you can define custom thing, …..

5:37:54

Parameters aren't shown here but parameters are really easy to handle, let me see if they have an example, we use this all the time I think all of our projects use rest angular. You have some account and it remembers the URL it came from you say you want to give a little list of places and it figures that that must be done by putting /places on the end and then you could pass arbitrary query params just as an ordinary object and those could depend as query params, no string concatenation, far as chances to get it wrong. They're doing some kind of user to log in thing here there's a way to do an inner stuff to kind of globally put all your off stuff here. You shouldn't have to write this all the time. First account made this much, if you request this then in most restful ways of thinking this will due of the list of all the places so that's why they say it can't last. 'Cause it's not one, if it was one it will be another slash after this with an id. This is all the places for account 123, you got it. We're not gonna code this one I'm just showing it for understanding.

5:39:30 - 5:50:59 [Class took a break. And discussing something]

5:51:00

Have you start running directive yet? I would like to play with some directives and I think this time I'm gonna put them on a separate page, what I'm gonna do is if I'm just gonna put them on a separate file. Let's call this dir.js you can call it anything you want I'm not setting a directives 'cause I don't like following the way that angular c does it. If I follow the naming convention base on its location, I oath to name it this way and it doesn't might get routing; 'cause I'm gonna do a route just fail to get it to show up on the screen. I'm gonna put a dir there I'm gonna name that dir I'm not gonna specify a controller 'cause I don't need one yet, I'm not gonna specify resolve just I don't need it either, it illustrates that basically everything in a route is optional. I need a dir partial. Obviously I didn't do something right? Dir didn't work. I know why it didn't work, is because when you add a new module that doesn't make it magically appear on the page then it doesn't make it get loaded, I'm just making a place to play with the directives here. Then I got to go to my app then I got to declare a dependency here or somewhere it gets loaded. Just kind of go to some motions of how we add another view. Now I have, I just wanted to get a different template

- -

because when I'm talking about directives I don't want the kind of need this confusion of all this other files, maybe with that guy I just put these to side by side I'll probably empty one of these out we don't need this one anymore either that. You already know what this means and then there's nothing to do with directives, you already know how partial works this files, is only changed in the most simple way of like loading another JavaScript file. Everybody has that background, some context there. Directives are what you see when you see the ng, when you saw something like this ngClick that's a directive. I can make my own, you can probably guess the syntax directive, give my directive a name, anybody have a nice name for a directive call it water, since I have water in front of me, I'm thirsty so I call it water. And then you can probably guess what comes here write a function you could probably guess that in this function you could do injection and be right. That's the least we can do to have a directive. In a directive would be returning a literal object which has a great number of optional things in it. I guess this is the minimal directive it doesn't do anything, I want to active this directive, you've already seen this form of activating a directive, there it is again. This should run without error anyway, I'm not doing anything yet, but see I didn't get an error and no error printed out. Seems like a reasonable place to commit. I keep going back to the angular website, were about to do this, first two words or HTML enhanced directives or how you do that going to enhanced HTML. The simplest thing you can do on one of these is put a template. Look at that, that text appeared, so this is the simplest possible directive, you can put a directive in that when it's invoked some template replaces what was in here. See the dry went away, and got replaced with a template from the directive. That's the simplest possible directive. You try to make the names meaningful, when you're gonna see some ugly thing about the names in a minute. If you wanted this template to contain HTML you just type it. This will get inserted into here the JavaScript code will grab this text inserted and then the browser will take that text and parse it as HTML. Simplest possible template. You can get thousand different things were gonna the do ten of them. Another files that's a good idea, I could for example go in here save this here as my water template and if I didn't like putting the text in here I could put it out there and then instead of a template I could talk about a template URL that's got to type the name right……. The next step of a complexity is it pulling it from a template from another file. That was pretty pleasant. Now I'm gonna bring on some unpleasantness.

6:00:00 Let's give this a more interesting name a two word name like a water jug. What if I mention name of the template or of a directive in there in a no such corrective. You guys can probably guess what's gonna happen right? Nothing, it's not an error in HTML for not an angular. An angular to just save the name or something that doesn't define…… You think this might work, no. Here we have a feature of angular, I'm grown kind of fine with the feature but it will be a speed bump to your learning. Angular goes out of its way to let you write things in JavaScript to look like JavaScript but write things in HTML that don't look like HTML. In fact, you type that in HTML to match this in JavaScript…… Those are five different syntaxes that all resolve to the same thing we'll get there. Of these, the first one is the one you should use this is the one that communities have gathered around but most of these you should consider as like legacy support

- -

because somebody thought they are good idea at some point and they already wants to renew than not but the last one is especially interesting, especially when you type it a little more like this. The reason why this one is interesting is that none of the, like all these are not valid HTML5 and HTML5 you're not allowed to just put arbitrary attributes on things you're only allowed to put attributes that makes sense for that element that are defined as being valid attributes to the element. In HTML5 if you want to do arbitrary attributes they must be prefix with data dash. And there are even some JavaScript libraries out there that all of their attributes are always prefix with data dash. Now, practically I've never seen a web browser that would complain about any of these 'cause web browsers are incredibly tolerant but if someone in your organization decided it was really important that all your stuff validate HTML5 you can just put this in front of all of your attribute directive and locations and your angular stuff will then validate this clean pure HTML5. Maybe I'll put these two that are we're thinking about together, those two that were thinking about the rest were not really worth thinking about. But let's just say we did something, this implicitly says right here, restrict is another optional thing you can specify for directive and this restricts how the directive can be used. It is a set of four letters, that are described that length in the documentation but JavaScript does not have a set data type and to express a set they just use a string so you have any combination of these four letters in any order that's not order that's just a set. A means this directive is useable in the attribute position, but it actually more than that, if I listed an E in here then it's also useable in the element position and that's kind of cool. That means I can use this as a new element, sounds slick right? Interestingly this doesn't work. 'Cause that's an xml it is not an HTML. If we do this right, I need to make it obvious where to break is.

6:05:11 - 6:06:33 Instructor working on his computer and showing students something

6:06:34

This shows that you can use it this way, well there's more here, there is C, C means class. I'll start labelling these, used as a class. And now we get some like a div, and type that ordinary CSS class. I'm trying to think which syntax that class one uses, like uses this one. That didn't work, that means it must be when you are using it in a CSS you'll have to use that or you have to use that syntax instead. There's even a way to use it a comment, this is kind of an odd one. I've to show you why it's hard to make this one work……. In this use you have some sort of tag, this doesn't really matter what tag……. This one will initially appear to not work even though I will put an M there. I do this one even though you may never even see a directive use this way, but this illustrates something really important in angular. There's a reason why this doesn't work, it's because if I go look at the dom here, see there's a directive, If I look at one of these others, you see how the text got inserted inside the div, see that? If I want to make it more obvious the text is inside the div I can even go here and I could put another div and I can even just like make up some style or something. It doesn't matter I could even put some random style to see it has something to look at. If you look at that, if I look in here to just inspect one doesn't matter which one. That's the div where I activated the directive and the inside that it put that template context. Well when angular

- -

tries to do that on a comment an HTML comment dom element is not allowed to have sub elements. Angular will attempt to put this content inside there and it won't be able to [6:10:00] and it won't make an error just won't work. So the div here that angular is working at HTML dom level and it's subject to HTML dom rules. It's a straight forward work around you can optionally tell it to instead of inserting that template inside you can say you'd like this to replace instead. And if I do that then the comment when will work and if I inspect any of these I will see that there is my, so the original div is gone, it put this marker on the div that inserted so there is not an extra layer now just directly has a bit there. You'll probably never hit that problem, I just illustrate that 'cause inevitably if somebody tries it and hits it and they can't get it to work and think it's broken you kind of stack over flow for a while before you find the problem. There's a little more you can do here, I could something like... You're not restricted to making up your own here. It's actually totally okay if you hang a directive of a preexisting tag. And I'm just gonna always use it as an element so I'll just go ahead and keep that simple and I don't want to use that same thing here but I'll just try to keep this as minimal as a possible for to least confusion. If this is saying when you match an h3 element replace its content with this new title….. Although I think the browser where to tell it did, browsers will just amazingly tolerant. There's my new title, This is not a very useful thing to do making all h3 say new title but it illustrates that you're not limited to things you don't recognize, there can be tag as if you do recognize that you've been using forever, and either you or some else who wrote angular can other write their meaning…… For example if just had and done this I believe it still will work. I think its EA might just EA, so it is just A, that is the default if you want it to be EA you have to EA. The attribute position is by far the most common way to use. I can do another little variation here Imagine that I have some content but I wanted to put a picture frame around. Some arbitrary HTML. For all the ones you've seen so far the content inside has been replaced but there is a way to wrap things around what's inside there. I'm using this one as an element but I wouldn't have to, it's a feature, one of the few totally made up words in angular transclude and if you transclude you usually have something more complex going on so it's usually worth using a separate file. I'm going to call this picture frame to make it easy to match these things. I just go over here and save this I think that's all that. Let's go something like a div class well that's a nice one that everybody knows. I could do all sort of things around it but maybe a minimum I'll just a label here or something, maybe I what to put something at the end. There is this ngTransclude you can do. What the transclude means is go insert the original content in this spot; I called it picture frame for a reason because it's a way of taking some content and wrapping something around it programmatically. That's really useful if you frequently on a page want to wrap same HTML around a bunch of different things you can use this to rep duplication out of your application, Let me get this. Here we go. Here's that text and I automatically put the title in front of it and my HR is not visible. I don't know why my hr is not visible but I'm just gonna be low tech about it I'll just draw a line so it is visible. Maybe somebody else would do some nice CSS. Any questions about this? This is basically all the simple things to use directives for there's no behavior here, we'll do some behavior but first were just doing kind of basic templating with directives. What's you're asking is, if I put this here, will it work? No. It's an optional feature, it's a mandatory feature route angular you must use JavaScript file identifiers in JavaScript, and you must use HTML style identifiers an HTML and not only is it something you must do is pervasive in angular so if you go to angular and you look up any of this built in features like ngApp you'll see that the official name of it is ngApp but the way you use it is If you look one of these it makes more sense so if this is the official name of it in JavaScript plan and that's what it will be called inside the angular source code and this is how they refer to it. But when you use it you got to do it this way. So yes to answer your question, there's no dogging this convention affecting you every time you use angular……. That might be one of the reasons that they did it although in this case it's not really being made of variable, No, it never becomes a JavaScript identifiers, it's just a key in a look up table. I think the actual explanations is they couldn't bear to make their HTML look like JavaScript and they couldn't make their JavaScript look like HTML because sometimes you actually do want to type an identifier and a dash is not valid in identifier in JavaScript. This is a decision they made early on and you see that maybe early on they weren't quite clear what they want the syntax to be this is the one that everybody actually uses. Questions, comments. The question is, what would happen if you define something that was already defined by you or angular, I believe the last guy in wins to test that I could redefine an h3 with a new work title [6:20:00] No, it didn't over write it. Based on this I would guess that trying to over write it would not work, that would be my guess…….. It's possible that by doing this you're adding more behavior so tops what I'm doing another directive on the same tag and it doesn't replace the existing one but if I did some other things here other than the template it might work. I've never had occasion to define one more than once. A couple things worth pointing out here, you can do some useful stuff even with just a sort of static templaty stuff you see here, you can use this as a tool to slash duplication in your HTML, if you have some case where you have some chunk, some five lines of HTML that you're repeating hundreds of times in your application, just in a real straight forward way you can put that HTML on its own file and you make a directive for it where you can just sort of invoke it this way or even this way wherever you need it. You can kind of use it as almost like a macro facility for HTML and that a useful capability. If you have some widgets and each widget has the same HTML around it to kind of create this control mechanism you could do something like I did with my picture frame here and use transclude so that you can put arbitrary HTML before and after some chunk repeated many times during an application, that's another tool for removing duplication. You can also use it to build kind of other kinds abstraction, you could think of this water jug thing as being an abstract thing I'm saying about the div and then over here I'm specifying some specific behavior but you know maybe I say water jug a thousand times throughout my application and then I add some behavior to it here I'm not doing that yet I'm just trying to take this jetly by showing you one of the static templaty stuff. The next step is gonna be for everybody to make themselves a few directives of using just the tools out here and kind of get familiar with it. Any questions before we jump in?...... Make yourself a place to put directives make up a couple names or things, experiment with this naming scheme and how it affects you just make sure you can make it work maybe use a template for one, a template URL for another they can be just short or shorter.

6:22:43 - 6:38:29 [Students working on another exercise while instructor assist them and answering students questions.]

6:38:30

If I look at the DOM by inspecting I see the original div, this is the div were I have put the directive on it and the contents got stuck inside, but actually I have replace on I got to turn replace off to show you what it does differently. By default if you're not doing a replace so if you're not replacing then if you inspect one of these, The original div is right there untouched and then inside it is that one with the file which the one from here. So the default behavior is the directive replaces contents of the element that you have an act on but if you want to replace the whole element instead then you can say replace true and then instead of replacing the contents of the element if I look at one of these the entire element has been replaced with the one that has the div style. This becomes, depending on what you're trying to do, if you're trying to create a certain dom structure because in you want to like bootstrap or something and style it in a certain way, there's just two different options for how it creates the dom structure depending on what you want to do. But the default is to have replaced false, I usually leave it false and that I don't know exactly what that, I don't know if there's like a major performs it trade off or anything

6:40:00 - 6:41:14 [Continues working on computer]

6:41:15

The way I use this in the first chapter here was very very limited just to kind of show off some static stuff, what I really want to do is things that are little more dynamic. I want to make some additional directive and use it instead and to try to, I want to keep this old stuff on the screen in case anybody needs to see it. But I'm gonna try to put a nice marker across the screen just to separate the new from the old. I use the attribute form. There's this convention that if you're making directives that might get passed around and reused of making it two character prefix…… What I want to do here I want to make a directive that takes a parameter, that's pretty easy to do. Directive, got to do the translation here, that's the same as before. Got to come up with a template, I think I'll try to type my template in line so I'll copy them on this one. Let's make this be a div style =

6:43:29 – 6:45:13 [Instructor working on his working on his computer]

6:45:14

If I made this a name that was not already on the scope and you that it would not be substituted and further more there's no magic going on here it's not like I could say it first name. It's not like there's a magic going on here so if I do this still no John and it's not like it's just the funky name thing so this will also will not work this sill no John. There's extra work you have to do to tell it to pull parameters for a corrective. I'm just gonna say name up there 'cause I like that nice short one to start, the extra thing you have to do is scope. If I say scope like this I'm saying that I want a fresh new scope I don't want this thing to just have the parent scope I don't want it to just take the scope from the page it's in, I want to give it its own scope and there's this little many language right here if how you assemble this things own scope. So what you can say is you do this little name value pair, you can say I want this to be in my local scope that I'm forming, this was in the local scope in the template right in here. And then this is a literal object and you use the object notation and then there's little magic things you can type here. If you say att or attribute then you can type name of an attribute. This thing when you use this directive which I've done up here it will make its own scope it will look for an attribute called name where you used the directive and it will put that in the local scope called first name and then it will apply the template which will then work. Similarly to the resolve routing we just did there's like three to four steps you have to give them all wired up right and if you miss one it just won't work. Let's see if I got them all right… Error on the console. Line 45 I have an unexpected identifier. This is the simplest way of passing a parameter into a directive. Attribute means just grab the literal exact unprocessed contents of the attribute and you still have angularisms available to I actually could do something like this. And I could 'cause remember I have name on the scope right up here, so this would work, see, so I can still do angularisms but it's just like this is gonna get a substituted and then it became Kyle. And then when this thing looks at, it's gonna see kyle. Directive don't go here, we already saw the places you can use directive, you can use them as attributes, you can use them as elements, you can use them as classes and you can use them as comments. Those are the places where directives go. The things that go in here are data or functions on the scope. …You're saying what if I wanted to actually pull it from this, what if I just wanted to say that. Is that what you're looking for? …..That triggered off the name, yeah, you can trigger more directive off of the same HTML there's no limitation on that. There is a mechanism to tell this thing [6:50:00] that one directive can see another directive, you can make kind of groups of directives that cooperate but that is way out of scope for a three day introduction to angular, but yes to answer your question is yes there is a way to do that. …..It fairly common to see something like this and I don't remember I'd there is a dash version over there or not so I'm just going to experimentally determine, yeah that worked. It's fairly common to see this, that and then that. Just because if there really is only one parameter for directive it makes a lot of sense to just have it be right there on that attribute. This is like the easiest thing you can you can bind those in. I do want to make this just a little more complex though to demonstrate that it's one way because the way I type this here I'm kind of cheating it's a little bit too easy you can't tell whether it's one way or two way because there's no way to edit it. The way we've done that before is made it an input like this, now it's editable. To demonstrate what I mean by one way. ….Now I should see name up there and if I edit it you see that it does flow in, but if I edit it in here it doesn't come out. That's because this atribute is a one way binding. I could be a little fancier, I could make this an OD view edit or the percent edit. And then if I want to keep this convention right that, that. If I do an equal sign there you kind of think of the equals been like two arrows that point like this. The equal sign is how you do two way data binding and if you do two way data binding that thing that you out in the value is not something which is substitution it's the name of something of the scope, we've seen many many cases for example right here, were you have a directive and an equal and the name of something on the scope this is that pattern so its directive = and the name of something on the scope, just curiously named name. So this should set up a two way data binding to where this first name here its bound both ways to here and it's the name missing on the scope so that data can flow in and out. Let's see if that works. It doesn't because I have an unexpected took in the line 51 and that's because I didn't remove my semicolon. This first one, I guess I really have to label this right? This is a view of the name and this is an edit of the name. Make it a little more obvious what's going on so this one I'm using one way binding although if I do this they'll change, this one here I'm using two way binding, just like angular ships with things that do two way binding the tools are like one character away to make your own thing that does two way binding. For example you could make a template here that's like an address editing widget and you could just data bind it to any address so could do like a billing address and the shipping address for example. And you could sort of template out box of HTML which have their own custom data bindings just as if they have enough to the angular in the first place, so you have two way data binding there. The next step is for everybody to extend your example and make yourself a directive that uses one way data binding with the equal and a directive that uses two way data binding or with the @ and two way data binding would be equal. As a third variation which is the ampersand we'll skip because...

6:55:00 - 7:19:49 [Students working on their computer while the instructor is assisting them]

7:19:50

The next bit of directives is to get something behavioral going on a directive that actually take some action so let me get these in Directives data binding. The best short example I know of is actually in the angular documentation. ……Template expanding, we already did that. We already did this stuff. Here we go this is it. This is the simplest example I've seen of manipulating the dom in some useful way. I'm just gonna bring over the essence of it to our example. This is a span it's like a div it looks like we're gonna have a directive called my current time and it data binds perhaps to something called format. We can go over here and see what that looks like. They expect there to be something called format on the scope I can make that true just by going this controller I already have putting something called format on the scope. And then here's this, my current time directory I want to bring it over and then trim it down and explain it. So this is the big directive, directives that do things tend to be big. Fix that, first I just want to see if it works. See there's a clock coming right there. There's four, five ideas all mixed in to this one this probably why it's so big. But the only relevant code at the top of the file that I added was this one line. So I put something on the scope and the only thing that ever here is like put a spam just some kindly throw away element and I put the name in the directive. You see the same pattern I explain with the dashes here become camel case here. We've talked about interval before I think I've mentioned it when we were doing promises. But it's being used here for a little different purpose. And then date filter is a thing that's built in to, it's a filter. When we talked about filters like the filter filter and the sort I filter this is the date filter filter and then it translates like a Java date object to a formatted string, that's a little bit of functionality built in that we're calling. One of the things that a directive can do is that it can return what's called a link function, see this return down here it's analogous to all of these returns but for some reason when they typed in they decided to define a function up here and then use it down there to follow the way I've been typing this little more closely I'm going to modify this to look more like that. Now this directive looks a little bit more likely you've seen before. We return a bunch of things in this case we didn't return a template. This directive it has no template, has no template URL it has no restrict, it has no replace, it actually doesn't have any of the other features that's use of directives but does have a feature called link. People that made angular they really like the words compile and link 'cause those are like programming language sort of things so they redefine them for their own use. The process I've been talking about were angular processes some dom and applies its rules and runs directive and does substitution they called that process compiling. I'm not really convinced that's a great use of the word compile but that's what they called it. So if you compile a first the next thing you do is link. Link means you attach things to some dom after you've compiled it. In angular compile a link are pair of operations which are maybe vaguely slightly analogous to compiling and linking a C program as it probably mostly done. This thing declare of a variables no big deal, declare a function no big deal, all things you've seen before scope watch you seen $watch before but might notice that this scope looks different than the scope you've been there's not a $ in front of it. That's because this context in a link function, this is not a context where dependency injection happens this is just a plainald ordinary Java parameter list, they always come in this order every link function and every directive gets a scopes and elements and attributes this is all seen in there in the box. When you say scope here, I mean you could say $scope who wouldn't hurt anything 'cause its pass position, I could do that and it still work but it wouldn't benefit anything that would be you'll be seen confusing so you shouldn't name this guy $scope. This element, well this is simply a reference to that dom element that were running on so that's gonna be a reference to this dom element right here, Now this element is not a bare dom element it's a jQuery or jQLite wrapped dom element. Angular includes something called jQLite built in to it which is a much shrunk down jQuery if you're using jQuery angular uses your real jQuery but if you're not it uses a its built in jQLite but the important thing is that this element is always pre wrapped for you so you very often see people who haven't learn this particular bit writing code like this. Trying to wrap that element, but this is actually always wrong because angular will never hand you a bare dom element anytime hand you a dom element its already wrapped and there's totally ways to wrap it again. You should recognize this from dom or from jQuery if you have a reference you can say .text and then that replaces the contents of the

- -

dom element with the text you pass. This is when you call this function the element of this directive run on will have its content to write there replace with the result of calling the date filter on the current date with the format so here's the format, where the format come from, well to show off that all the full scope capabilities are available you can call scope so this will be the scope where this element, or where this directive was run, you can call watch on it to have. Now attr is that my current time so remember I said that the attributes that when I did this way you have this kind of high level syntax to grab access to attributes, well down inside a link function you have this low level thing it basically just hands you the attributes wrapped in the JavaScript object. This is a way of saying give me what's in there and it doesn't have all these capabilities data bindings and all that this is just this is the bare underline mechanism. This link, this is a low level maybe greedy dom level programmer feature where is this up here is like a template level programmer feature. Since all we can do is get the string value here if we want to go look that up in the scope and watch it well we actually have to call scope watch we don't have data bindings we're doing our own data binding by doing scope watch. This thing here gets called whenever that value changes it actually gets called once the beginning also when the value arrives, so we said that we remember the format when we update the time. Ignore this for a moment. Interval is a way of getting something called repeatedly and I could change this and that will called more often if I wanted every five hundred. Intervals are series for calling things repeatedly all it does is call update time repeatedly so this is sort of like a set time out that sets its time out by itself again and again. We hold on to what comes back so this returns some sort of a handle to that interval and right here we explicitly cancel it so this interval service has a way of cancelling an existing interval. I mention that there's an angular event mechanism we haven't talked about much yet but here's how you call that event mechanism. This is a reference to the element it's wrapped up there is a $destroy that's a hint that can angularism. Angular will emit this destroy event right before it actually removes this thing from the dom right so if I like change pages on the page widget and this is being pull out of the dom this is a warning it's about to be pull out of the dom this is actually important if you skip this. You're leaking intervals. [7:30:00] If you skip this and do some paging or some routing thing you'd be leaking this background processes running and if you entered that a whole bunch time should run out of memory Everything we did up to this point, everything from here up and all the other codes since yesterday morning we were supplying template and we never have to think about things like leaking references to thing on the dom. When you stepped in to a link function in the directive like this is where it's getting deep and you have to start being aware that yeah if you make mistakes you're gonna 'cause memory leaks that aren't gonna go away to the pages reloaded and so on. It's pretty typical for jQuery based apps to more or less ignore memory leaks 'cause if you're like just like interacting for a little while on a page and loading a new page well so what you know you build a new page to throw away all the memory anyway. But with angular you're typically running very long live single page apps somebody may have stay on your page for all day and so you kind of do need to be able to more careful about leaking resources. That's a pretty complete walk thru on what this guy offers just make sure I didn't break them in the process of working on him. And just to show that this data binding really works on the format we could do....... So if I didn't like this format I could maybe do something like that instead and then I'm sure to dynamically changing around how I want the date format to look and there you see each time I do it all the name will be updates, It's a little bit more, it's a lot more book keeping, this is all book keeping I never done until ten minutes ago. But this is the kind of code that angular is built in, angular is full of directives, all these things we views like ngModel and so on are all angular directive and angular is programmed in this kind of low level maybe greedy way so that applications can mostly be programmed in this high level template data binding sort of way. This is the interface this is the border between the higher level world and the lower level world…… Do I need to explain I need it again, the watch part. Things this is, I would say a slightly overly complex directive just to point out what a simpler one would be let me make a simpler one. I could have a simple time and then it's a used simple time, I could do this, I think I need to aligned break here then I called it simple time which that will be converted to HTML to look like that and then the simple time doesn't time this silly configurable format, rather than have a configurable format by simple this just gonna be hard coded. I'm not gonna have to do this, so all that goes away and then instead having to format the variable I just have it hard coded in there and I think that's it. Let's see if my simple time works. Simple time works and the complex time works. The thing you're asking about is the scope watch which is the difference between the simple and the complex. Before I explain that does anybody have a question with this simplified version that doesn't have scope watch? You really need to understand this a hundred percent to lead to the next one. Could everyone rewrite this on demand? Then ask questions. Ask questions about simple times…So, wherever you invoked the directive it took all the attribute on that element and it prove them in a JavaScript object that and add an object to you. This is its way of making the attributes on here available to you; the complex example actually uses that to get some data out of. The simple example that's in here it doesn't even use it and since this is JavaScript you can just ignore extra parameters you don't care about I could even just do this without changing the meaning of it……. The timeout id, when you called interval, interval is a service use for running it something in a regular interval it gives you back a handle to that interval which they called timeout id so that before that element that this directive is attach to is destroyed we can cancel it so that we don't leave the running on the background for no reason, that's what this guy is for, it's housekeeping to make sure leak resources like time resuming in the background….. This is the boundary between high level and low level this is where you can build your boundary between angular data binding and manipulating with dom if you're gonna manipulate the dom in the directive like this is where you're supposed to do it. You shouldn't write any code in the controller that reaches into the dom that's on the high level side of that boundary. We're doing it right? The thing you just asked about line 69 does it. This text dash replacing in your HTML of this dom element. This is where it's okay to use jQuery things to manipulate what's inside…… There's all kinds of hiding and showing for example, here we have a way editing the name if I just want to like hide something I could just do a div and there's a thing called ngShow and then you can do

- -

an angular expression here name not = to Wayne and then close this div. If you just want to o the hide and show that's at the top at the toolbox you don't need to go deeper that if I just make this be Wayne that's the hide. You don't need to pull out jQuery to hide and show stuff…… I mean if just want to just hide an show this whole thing I just put something I'd out an ngShow around and turn off that way I would generally only do things this low level way, that has to be done the low level way. So the pending question was to explain the scope watch again. This is the only difference between the top code and the bottom code. In the top code we have to hard coded the format in then in the bottom we have a var this variable is only visible inside here for the format and we're watching something to set the format, This scope watch is the hairy painful way that you have to do data binding down inside a directive link function. This is an implementation that something that we've seen previously we just did an equal sign. This is how that = thing is done under the hood. Attributes, this is a JavaScript object containing all the attributes of the element where the directive was used, here the element were the directive was used. All of the attributes is just one attribute and they get put by name translate it to JavaScript in an object which is pass here, attribute.mycurrenttime will just be the string format. That's seems more like that @ sign data binding where it's just a one way binding of a string that's way to just get that value. We are watching it, we are scope watching it [7:40:00] which is this will get executed whenever something changes just on case that string change. This thing, we want to actually know if, we actually want to watch the content…… To make this clear I will cut that out when I say that that is literally just a string what I mean is the meaning of thing would be the same if I literally put that string in here. That is equivalent 'cause its literally just the string, what were in is we're watching a thing called format on the scope but what scope is it I didn't do this thing to get my own special scope I didn't use that feature so the scope in question is the scope where this guy lives. This thing lives in a scope which is the one that was set up with this controller, when it looks for a thing called format it's gonna find this. That's the format it's gonna find. When that changes or when it's for set this function where gonna called 'cause we've already seen how scope works and we will set this inner formats so this variable you know it's a long live variable but its only inside right here and then we'll update because time uses the format that way as I change the format immediately we'll format this that's on the screen. Did answer the question of what's that scope watch is for?...... If you left that out, yeah that still works so let's do this and if I started just changing this in here to say d-m then it won't pick it put necessarily immediately but you'll see it pick it up next time the time changes, that's right, that was just a nicely that was done that they could update faster more responsively…… Absolutely, that's what this is this is a reasonable component with behavior, everything we saw before this last unit here was sort of reusable static things or reusable data binding things, this is a reusable component with arbitrary behavior. This is the escape patch you can do anything here….. I mention this previously this namespace for mycurrenttime lives and where simple time and od person of it, that's a actually a global application wide namespace so the answer is if you anywhere in your angular application if you make this directives up here you have this code anywhere, then on any of your templates you can use them. You don't need to do anything special at all you could make some module here that had your comment directives you use all over you app and they would just work you don't have to do anything special to make more color to your app…… Right! When I used it, this is not namespace I just said simple-time and that work 'cause there's a directive simple time. This is actually living inside a module kyle dir and a file dir.js and neither of those appears here you don't use the angular module name and you don't use the JavaScript file name those are irrelevant, It's match purely based on the name in here the name in there with that translations between HTML names and JavaScript names…… I just put them all in the same file 'cause its easier for me to scroll up and down but I absolutely could have made a separate file I could make a time directive file and I could move those into it and I could go reuse it to anywhere I wanted….. Definitely because when I do it I always make my each file via module 'cause I find it else to keep track off mentally. Right, 'cause when I added this file kyle.dir when I put it in dir js it landed right here I had to do just a couple of things I have to go to the index to get it unto the page and I have to go to my top level module and declare my top level module and declare dependency on it. If I added another one it will just a few clicks and a few taps…… Yeah we pull the JavaScript unto the page and then you have to actually declare the dependency to make it available inside angular. There's no direct correspondents if you would go in here and look at angular route.js, I need to adjust my, oh I see this feature in it's a blind text you could've map you file here; you kind of immediately see well this is a really big file. Angular route is actually 891 lines of codes but somewhere around here we've see something like directive. Provide. They tend to use provide 'cause provide thing is more general way to do directives and controllers and everything else and they tend to use that 'cause they're deepen to it but yes, here's the mechanism supplying it, there's the ngRoute module and that here's the $route service that it's providing. That name is the one that has to match that name right there. JavaScript doesn't have a module system yet and so we have this homegrown module system. It wouldn't have to be the same system as Java has it wouldn't whether they have the com pattern or not but just having any kind of module system, like in JavaScript in Java you could be critical if you don't like the module system at least you have the module system you not like and JavaScript don't have a module system so it's implemented in JavaScript and so the angular one like there's just require js and it's different from this. More question about this deep end of directives?.... We'll hit that tomorrow I'll walk thru the whole process the remaining 14 minutes of today is to answers any questions about directives. If anybody likes it so much you want to dig in want to write one I'll sit here and help you a bit but this is the, this is the deepen, you can kind of think there's two halves of angular programming, there's the using stuff and the making stuff. Everything up to this point is more of a using stuff. And this is the making stuff side. There's like a bunch already out there the great thing is the thing you have used in community so angular grid directive. There's this ngModule guide here that ngGrid somebody made a thing called angular table. If you run an angular table, this is a directive somebody made. You can actually pass on a model as a literal which I guess make for a cool demo and after you want to use it. And these people they wanted to be able to type HTML to define how your grid looks, they made directives caller hitter row and hitter column, this is not built in they actually made a bunch of directives to define what the table should look like, think there's actually a kind of slick. It has like little behaviors you could expect but I think you could click on things……. This is an example of somebody made some complex directives I bet if I click the right place here I can see it. Angular table they called it. How long is it? This is 407 lines of code that all of the same low level style I talked about to [7:50:00] implement this thing, that's what you normally see; when you see implementations they are a bunch of low levelish code. The ngGrid that's the one I mention its really rough they're working on re writing it but it currently creates way too many scope watches the one where I made the error on the white board here and I wrote 4,000 and that should be 40,000 which is like that should slow for us. This is another example of a directive or a set of directives if I figure out where to click. Let's show big this one is. These are all different files there's just one thing the divides this into. I bet I can find it, this is probably its main directive this is one of many. This is its main directive 173 lines but there's a whole bunch more codes. 'cause the code were just pre controlled scattered among like 20 or 30 source files Obviously understanding this is way outside the scope but the complexity you see there is introduced by this here. This is probably; this is just about the simplest useful behavior of directive. The directives that are before are more of sort of like templating stamping things out to reduce and tighten up your HTML and lets you write big complex thing in a hurry this kind of things by adding new behavior they're both called directives but they're very different in terms of the complexity of working on them. While I'm talking about grid its worth looking in here. This is a directory somebody put together if you want to know if there's angular module for something well there's 476 in them listed here. There's a whole bunch of stuff in here that are someway bootstrap related. If you want to work in bootstrap you're not gonna have to start from scratch start running in your own directives to make bootstrap work correctly. You're gonna be able to grab something that's somebody else's work quite hard on. But I think this one that's called angular UI, the UI right there, this is the main one I believe. Here's the JavaScript just to find some groups and here's some mark up and so that accordion accordion group that's pretty slick right? Because they implemented it as angular directives and instead of being you put a bunch of divs and you kind of annotate them and you make some call to tell jQuery to take over them you just get to write them directly as if HTML has accordions in it. Pretty slick and you even of composes, look there's ngRepeat sitting there compose with an accordion group. You probably don't want to start with just bare bootstrap and start calling it on things you probably want to grab something like this and get a good grasp of it and use this to avoid writing all that code yourself. So those bootstrap alert chose an angular wrapper On the bootstrap alert, that's you know exceedingly easy to use this…… There's not even any dom manipulation code here alert is just an array and if you push to add one to it it's all data bound so when I click that the data binding just makes another one up here. That's all I really have for today I'll cancel off on 8minutes early.

7:54:29 – 8:05:26 [Instructor dismisses the class 8minutes early and let the student work on their computer]

- -

Day 3

[00:09:47] CLASS STARTS

Today we have a few more topics, and hopefully about half the day available for everybody to build something while we are here to build. Something went wrong with my column width here. I don't know how to fix it. Resize columns with the data. That's not what I expected. Resize columns, data. That's not a very good spreadsheet. At least it's free. Big change from yesterday afternoon's topic and that is how should you get all the code for all this stuff on your machine for client-side development. It's not a strictly Angular topic but it's a topic you will run straight into if you build anything with Angular because Angular's part of this large complex tool ecosystem and you'll pretty much only use Angular alone today. Once you start building anything real you're going to be wandering the internet finding numerous add-ons that you want. Because the alternative is to decode it all yourself from scratch by hand. On our projects, which are of moderate, I'd say moderately complex business applications, it's not unusual to have 20 to 30 libraries on our webpage. We don't put each one separately with a script tag. There's a whole bunch of build process stuff we'll talk about. But the first step is just obtaining all the files. How do you guys obtain the files in your Java projects for all the libraries you use?

[Student] Maven.

Kyle: Were any of you working in Java in the bad old days?

[Student]

Kyle: Amazingly enough, the reason Maven survives the pain of using Maven is that not having it is actually even worse. I know it's hard to imagine if you're using Maven all day. In the bad old days, we would wander the internet finding web pages about libraries we wanted, then we would download that library and we'd put it in the library directory of our project and check it in. And then when we would read in the documentation that that library depended on six other libraries and then we would then wander the internet getting those versions of all those other libraries. And then we would make little lists on paper, or on whiteboards, trying to understand exactly what versions of exactly what libraries we needed to all work together in our application. And then we would manually, figure out how to get them all in the class path. Whether there's an order dependency we would get all that right manually. The reason you tolerate Maven's unpleasantness is that it's still a lot better than doing all that stuff manually. Have you guys heard the joke, you type Maven and it downloads the internet? Well before Maven, you would have to manually download the internet and check it into your projects.

I put a chart up here. I always put Maven first on this chart because it's the tool that most people or the greatest number of people seem to be familiar with. But it is one of a substantial family of tools. I label them dependency management up here. They're really dependency management and

- -

build tools on this list. With Java, you use Maven to do your dependency management and Maven to do your build. When you Ctrl-Alt the file, usually called pom. xml. Has anybody used C#? Any C# folks in here? Did you use NuGet when you're using C#? NuGet is kind of in that same space. You almost certainly use MSBuild which is built into Visual Studio.

[00:13:42]

Microsoft actually has a pretty good story around that. And that MSBuild is the build engine that's inside visual studio and so you can make a project with Visual Studio and you're immediately ready to build it at the command line without having to go investigate ways to do that. But that's pretty slick. We're a little behind that in the Java world. Has anybody done Ruby on Rails programming, that kind of thing? That stuff? Let me fix an error while I'm standing here. There's a tool called Gem. It does the same thing. Has anybody done Perl? No Perl people? Perl's actually the granddaddy of all this. It has this amazing thing called CPAN and it really pre-dates all these other tools. CPAN inspired all this stuff. It's really fantastic. I write some code enclosure just with a tool called Leiningen which is an awful name for a tool since no one can ever spell it.

But what's really of interest today are these for what we use in the JavaScript world. Someone here mentioned doing some coding in Node. Node ships with a thing called NPM, Node package manager. And the nice thing about it shipping with it is since you all have Node, you all already have Node package manager. It is possible to use Node package manager to also manage the packages you want in your web client code. But it's more common. On the client said JavaScript used a tool called Bower for that and that's what I'll show you. The next big change to make the little sample applications everybody's been tinkering with is to remove the manually fetched libraries. And we're going to issue a small number of commands to switch over to automated dependency management of Angular and angular-route and all that sort of thing. So the good thing about this is that although it's pretty imposing if you just wander through the web pages, if I just walk you through it, like in half an hour you'll all be fetching all your dependencies automatically for Angular. That's a pretty good thing to get past.

The way you do it, clear this guy out. Put this thing back to a generous point size. That's not what I meant to do. Tell me if anybody has trouble reading anything I type. I have a project here. I'm going to ignore that Node module and dist stuff for a moment just to prove that you can ignore it. I'll delete it. So you need to get your projects to this state and you'll notice there's not any extra files here at the top level. Like if they have a leftover package.json or anything like that, just go ahead and delete whatever package.json you have left over. Have all of you guys through starting your own project? I had mentioned doing that, I don't know if they already got help doing it. I'll just show the process. So this is the process that I went through when I started this little K4 project. You have to do something appropriate for what your machine has.

So let's say I was going to start a new project K5. I'm not really going to start it. Oh I did because I went to this project we were already working with, this PhoneCat thing and I just copied all of its contents into my project just to kind of use it as a starting point because it's a handy starting point. And the idea here is that it's not inside the thing called any other PhoneCat anymore because this is my project. And since it's my project, I know there's a bunch of things here that I don't need. I know that I don't need any of that. I don't need that. And in here, the only thing I need is that web server. I'm not using anything else. I just kind of pare it down to just the web content. You all want to do something like that. And the important thing is that you need to stop operating in the Angular PhoneCat directory. You need to make your own directory when you start your own project.

[00:18:12]

I'm using a Mac and it has some slightly unpleasant behaviors. Unless I press some extra buttons, it doesn't show me hidden files here. And so I can't tell whether I have a .git directory looking in finder easily anyway. And it's important to get rid of that if you have it. So let's see here. Let us go up and check with the terminal. Where am I? I don't know where I am. In my K5 directory, I'm just looking. I don't have a .gitDirectory. If I did, I would kill it. I would do something like that because I want to make my own. I want to start a new get project here. I don't want to add to the history that I brought from the Angular PhoneCat project. That's a big messy example project from an Angular website. I want to start a new fresh project. I think you guys are mostly familiar with Git if I remember from yesterday. Everybody knows about Git and start a new project. That's what I thought I heard so I didn't walk through it too much. I'm not going to use that. I'm going to use the one that I already started on Monday.

If you get your project to this state where it's your project. Mine is named K4. You can name it anything you want. I'm not working in the Angular PhoneCat project, got my own project. And I don't have a package. json file. I don't have Bower file. I have to clean out anything that I'd be left over because I want to go through the motions of setting up from scratch. I'll put all these commands back up at the end anyway. To start a new project, under npm package management, type .npm init. But I need to draw a diagram to make it clear why we're doing that. Dependency management is good so we end up operating in a nested onion of dependency management. Like if you're working on Linux, this outer box is your computer. If you're working on Linux, you might be using RPM or deb files to do dependency management at the machine levels. Is anybody running Linux? Which Linux are you running?

[Student]

Kyle: Okay so you're using .deb files. You already did an automatic dependency management of a software in your system and that's great. On Windows and Mac, we don't have those advanced features like 10 years behind Linux. Within a specific project, you might have a server level project so it's possible that you'd be using Maven to dependency manage your overall project. It's possible you might have a subdirectory inside that which is your client-side project or your client-side project. You might be using npm to manage the tooling, the client tooling, and then inside that we're going to use a tool called Bower to manage the client packages. The only parts we're going to look at today are this npm and Bower over there because this is what's relevant to product development that we were talking about. Just to get you some context. It's very common as it's nested onion of dependency management going on in a complex system.

The reason we're going to use npm is that every piece of tooling you want to use with Angular is written in JavaScript and needs the Node to run. Even though Angular runs in the web browser and it doesn't care about Node or npm, all the tools you want to use to help you use Angular and there's going to be many of them over time, all require Node and npm. So that's why even though we're not even doing Node development, I'm not going to write a single line of Node code. Nonetheless, I'm going to use npm and Node to do some package management. Does everybody follow me what I mean by that? I'm going to use this to fetch the libraries I need for tooling. Yeah go ahead.

[00:22:23]

[Student]

Kyle: Node is a server runtime environment much like a Java VM is a server runtime environment. There's a thousand ways to deploy it. You're going to need that runtime environment where you run it.

[Student]

Kyle: The client side stuff we do that all runs in the browser. So that's just HTML and JavaScript files. You can deploy to do that on any static web server in the world. But if you write logic using node, then yeah you have to have Node to deploy that. Just like if you write your logic in Java, you got to have a JVM to deploy it on. Like if you write PHP code, you can't just take a plain Apache. You have to add a PHP support through Apache to deploy it. So Node is like in that family of thing. I'm not going to cover any of that here. On script we're using Node and npm to do package management to get tooling into my project because that's way smarter than manually downloading all the tools I need every time. And it's a few commands. So it's just a little bit starting process.

I'm going to do an npm like that just to kind of set up my project and I'm going to mostly say take example in our standard answer, so Kyle's example project, I'll just give it a name. Version, I don't know 0.5.0, I'm just making something up. Description, I'm going to leave that blank. The entry point, I'm just using this for dependency management so I don't care about entry points so I'm going to just say presenter. I'm not setting up Node package automated testing because I'm not a Node developer. So I'm just going to ignore that. There's some slick features where it can know what Git repo package it comes from which I don't care about so I'm going to skip that. If I was making a Node module that I was going to publish to a Node module repository, then there's some search capabilities there where you have keywords. But that's totally irrelevant to how I'm using npm which is just to manage my dependencies so I'm just going to press enter. I could leave this one blank. I might as well put my name in there because it's pretty easy. And then license, I don't know. I'm just going to type none because I'm not sure exactly what the BSD 2 Clause says. If I wanted to I could type copyright 2014, whatever. You can type a license if you want. What happened there?

[Student]

Kyle: Oh.

[Student]

Kyle: Yeah it doesn't warn. I hit this before. The name has to be a single JavaScript identifier. Such a bummer. No description, no entry point, no test command, no repo, no keywords. I'll just do one word for the author this time. Type none because it's less to type. Yes that name needs to be a single valid identifier. All I did there is I made a trivial package.json file. Let me bring that up. That looks like this. And it's nice to use that npm init command but if you wanted to, if you had one of these files, you could just copy and paste it. It doesn't matter. It's just a text file. There's no magic here. That's the very first thing you need to do.

[00:26:06]

Then I said I wanted to fetch my dependencies with a tool called Bower. So I'm going to do an interesting little trick here. I'm going to use the outer tool to manage the inner tool. I'm using this to manage this which is not—it's not immediately obvious why we're going to do that but it'll become so. So that's just a npm install. And now this is where it gets a little bit annoying because the people that make these dependency management tools, they sort of semi-copy off of each other. But they don't do a real perfect job. So let me see here. I'm trying to copy this simple word here. With npm, this is the command syntax to install a package. And then save this dash dash save dev means to add it to my list of declared dependencies. npm install, save dev and I'm going to use this to install Bower. I'm using an outer tool to install an inner tool. And as long as I'm installing stuff, I happen to know that later, later this morning I'm going to show you how to use Karma, a testing tool. So if I like, as long as I'm typing commands anyway and letting them run, I could do Karma. And I'll probably do some more later but this is just getting through, just showing you how it works. And this should be real familiar. If you type in Maven in command you much download the internet. Right? Still waiting.

[Student]

Kyle: Yes it does. The typical fanout for Node modules seems to be like sometimes 10 or 20 to one. It's like you install one package. It installs a package. It installs a package. I've had 400 packages come in on one command before. Yeah go ahead.

[Student]

Kyle: So you sort of need to learn the verbiage. Does anybody know is there a Maven command up the command line to edit dependency? I always just edit my Palm file. You Maven users know of a command line way to add a Maven dependency? I'm sure there is one.

[Student]

Kyle: Yeah. There is one, but I don't know it. In the Maven world everybody seems to just bring up their editor and just copy-paste the block of XML and edit it.

[Student]

Kyle: Added to this project, there is. I think there is. I really think there is a way to say put this in my POM. I just have never—I just don't know what it is. Okay so don't worry, I'll put these, the commands up at the end. I'm just going to walk you through it and then I'll run my history and I'll copy-paste and leave the commands up so you don't have to worry about catching the exact command as it goes by. I'm just trying to show you the process. This is kind of painfully slow. This is because Karma knows I might want to use PhantomJS so I don't have to go to the PhantomJS website. It just says oh I'm going to grab you PhantomJS in case you need it. That's kind of slick.

[Student] So you grab Karma.

[00:29:36]

Kyle: Yep. It doesn't matter with the order. The reason I wanted to grab a couple is just to make this example a little bit more obvious. What I just did automatically added a bunch of dependencies to my package. json file. See that? When I first brought this up, this section was missing and just from running those commands I have now declared in a file my dependencies and this file, I will check in the source control. Which is slick because now I have this nice short text file that says exactly what I need.

[Student]

Kyle: Yeah go ahead.

[Student] This obviously put a directory node_modules through a whole bunch of stuff for mobile development so that you keep it clean and commit that to Git.

Kyle: You should never commit that to Git. That should be in your .gitignore.

[Student]

Kyle: See I just added command to my .gitignore. The reason is that it might install different things on different platforms. Remember when this scrolled by here, if you look closely, right here, see that right there? It's a somewhat intelligent package. It knows to get the right PhantomJS stuff for my system. If I check these in, these will be the wrong files if you're on Windows or Linux. So you would not ever want to check in your node_modules directory. It's worth noting though that you can get a lot of files. How many files did I get in Node modules? I get 6,000 files in my Node modules running a couple of those commands. So it's pretty nice automated dependency management. I didn't have to track down. I got all the right files I need right there. Just to prove that it's automated. If I had a fresh checkout, my node, I would not have a Node modules directory. I'll be in this state, no Node modules. If you just do npm install with no, nothing after it, that will look at your package.json file and go download all that stuff and put it in for your project right now. The only thing that gets checked in was one short easily readable diffable merge-able text file and that manages all of your server-side JavaScript tooling files. Are you using npm init with this old hat?

[Student]

Kyle: Okay. That was not real hard. By the way, you notice it went a lot faster the second time?

[Student]

Kyle: You should—okay let me talk about -g You should only use -g if you absolutely have to. Because what that means is, grab the current version. Install it globally for me. That means if you weren't on this project, nothing's going to work. You're going to go talk to him and say, "Type all these -g command" and you're going to get the current version of when you type install -g. you now have probably different versions of things and things are going to break. So you should only do global installs where you actually have to and there are a few things you have to. You should do local installs for as much as you can because local installs get version managed in that package.json files. It's up in this little window.

[00:33:07]

I was trying to illustrate here. I just did an LS in my .npm directory. In your home directory, it'll make a thing called .npm and it'll cache all that stuff. And that's why when I deleted my Node modules and I ran npm install again it didn't take nearly as long because it only checked to see if they haven't changed. It didn't actually re-download all of them. You should never touch that file by hand. It's just the local cache that it does to make things go faster in your computer.

So what I just did does not yet touch any real—any part of this application because I'm just pulling in some tooling that I'm not using that. So my next step is to start using the tooling. I mentioned that I'm going to use Bower to manage my client-side packages. There's a bit of ugliness here. You can type something like npm install -g bower. If you did the -g, that puts it somewhere where it's on your path, and I've actually done this. I don't think I'll do it again. I really don't recommend that. This is kind of a hole in Bower. There's a tool called Grunt. We'll talk about in a while. With Grunt, they separated the command line interface. You just need to do the command line interface global and you do the rest of Grunt inside your project. Bower's a little behind. It doesn't have that. So in the meantime, I'm going to take advantage of the fact that I just happen to know where Bower is. I happen to know that it's there. And I'm just going to run it from where it is in my project. That way I know that I'm running the exact version of Bower that I have version-managed in my project.

- -

So it's probably unimportant if Bower doesn't change very often. You could just type Bower instead of what I type there. And you get the global one you've installed already because you probably already have it. But I just like to show this because there is a way to kind of explicitly ignore the system-wide path and say, "Oh yeah I want this specific version of Bower that I have installed in this project right now." So I typed npm init right to initialize my dependency management stuff with npm. Now I want to initialize my dependency management of my client-side packages. And so the nice thing is that a lot of these commands are very nicely parallel. It's Bower and init and it's going to ask me a different but very similar set of questions. So I don't care what it's called so I'll let it be called K4. I will just make up a version number. Doesn't matter. I don't care about a description. I don't care about the main file or keywords. The author is fine. I haven't even thought about what license it'll be so I'll just type none. This project doesn't have a homepage. I usually say no to this. If you say yes to this, it actually did screw your project and looks to try to figure out what you're using and adds it to its dependency list, I don't like that because I like to set my dependencies explicitly. So I will go write that to no. That ignore thing is fine.

This especially will be confusing when you first use it. Bower and npm and Maven, they're not only tools for managing your project's dependencies. There's tools for publishing libraries for other people to use. It's like Maven—you guys know what Maven central is? You've done the Maven central? Yeah. Bower, they noticed some people were accidentally publishing things they didn't mean to publish. So they added this ability to accept this private flag so that in case you went through all the motions to go understand Bower publishing and make yourself an account on Bower, set up your authentication to let you publish on Bower. They're trying to give you a little safety speed bump so you don't accidentally publish a private project on to Bower repo.That's kind of nice of them.

[00:37:11]

Now the fact is that if you never use Bower to publish anything and you never go to the Bower website and make an account. Then you have no risk of ever publishing anything right? You have to go through some motions to be able to publish. But in case you're in that sort of thing, there's a way of marking a package as private. Make sure I don't ever accidentally publish this. That's fine.

[Student]

Kyle: The command is slightly different and I don't recall what it is. If you do a Bower install, if you do an npm install -g bower, and that puts a Bower command on your command line, if you look in the modules in your computer, you probably have something different. Because see I have a Mac version so I don't have a way to readily show you what's on your Windows computer. This stuff all works on Windows. I just don't know the exact contents of the files.

[Student] It's an executable Node file.

Kyle: So if you type node, and then that name, then you'll run it under Node. The other thing is if you get—you probably know Windows command prompt. Windows command prompt sucks. It's awful. It has like no features. It's a relic from 1992 or something. I always either use Cygwin or I use the… If you've installed Git, you have a Git Bash prompt you can use. If you use one of those, Unixy commands will work for you. When I do use Windows, I actually always use the Get Bash prompt so that everything works. Because it's just endlessly inconvenient with the Windows command prompt differences. It is possible to make it all work, it's just inconvenient.

The reason I brought this up is I got a bower.json file. So you might recognize this as a result to some of those default prompts I took. You won't really need to care about anything in here. Just kind of be aware there's some Boilerplate junk that goes in here. On the package.json, when I first made the file, it didn't have any dependencies listed. And with Bower, when I first made the file with init, I don't have any dependencies listed. I'm trying to get you guys to understand the parallelism of all these tools.

Let's say I wanted to get a tool. I wanted to install something. Well there's a command Bower search. That's kind of slick. See I think it shows something if you just type search. I think maybe just list a whole bunch of packages if you just type search. That cannot be too slow to even wait for it to do it if it's talking to their web service. I'm not patient enough to list all the packages. Did it come back eventually?

[Student]

Kyle: You can use Bower search to ask Bower, "Hey what do you have available? You have any Angular stuff available?" As you guys probably noticed, Angular is actually really, really popular. So there's a whole bunch of Angular-related packages published into Bower. That's a really long list. I really don't care about that list because I just want to install the one that I care about.

[00:40:32]

Kyle: Now if I just run this command, it will install it locally but it won't track the dependency. It is not clear to me why the default behavior is to not track the dependency. It's also not clear to me why the people that made Bower that almost certainly use npm chose a different parameter to help save the dependency of an npm. That the Emit command is the same. The Install command is the same. But the option to make the save the dependency is slightly different. I find that annoying but once you hit it a few times, you can endure it. So install. A command like this says, "Hey Bower go get me Angular." Bower has this odd default. Go ahead.

[Student]

Kyle: You can pick any name you want.

[Student]

Kyle: You could just press enter to all those prompts. It doesn't matter. You can pick any name you want or you can just take the default. That's why I was kind of walking through it repeatedly fast to show that I didn't do much. I just kind of said, okay okay. Like a wizard in Windows, you just keep pressing next until it's done. We'll walk through it again. I'm just trying to get through it once and then we're going to help everybody get through it on their little example project. See this Bower components directory? By default, that's where it puts the stuff and so we've put Angular. I actually don't want it there. I really want it to be in this lib directory. So what I'm going to do is I'm going to delete my lib directory because I'm going to replace it with stuff on Bower and I'm going to delete that directory Bower just made.

And this is where it gets slightly annoying. It's inconvenient to set the directory where Bower puts the files. You just have to know or you just have to search stack overflow to know that if you make a file called .bowerrc, you can put contents like this in it. You just have to know it's not my favorite thing. Using the editor of your choice—and by the way, you will hit problems on Windows. Windows does not like to let you make files that start with a dot depending on the tool you use and how you do it. We can help you through that. I'm using VI and I think it's an app lib this time so I'm going to make sure I spell it the same way. This is the inconvenient way that you tell Bower where to put the dependencies it downloads. And don't worry about catching this the first time. We're going to help everybody walk through this.

Now that I put that in, remember how last time I did npm install, and that just meant take all the dependencies you already know about and get them right now? Well over in the bower.json file, when I did that save option I now have recorded with a single line of text my dependency on Angular. So now if I just run Bower install, it will go fetch whatever was listed in the Bower file. So the Bower file is a simple short plain text list of all the libraries you need and then there's one command that just goes and gets them all.

While I'm in here, I know we've talked about jQuery so I think I can do that. A lot of these common packages you can guess their name in Bower they're just name of the thing in lower case. Now I have jQueryin my project. But look at that. I got jQuery2.1 that might not be at all what I want because a lot of people are still using an older jQuery. So I can just go in here and I can say I want 1.9. That's a jQueryI really want, a 1.9. I don't think I have to remove them. Next time I do a Bower install, I think it's actually going to get 1.9 for me. See now? It pulled my 1.9.

[00:45:05]

As long as I check in to my source control, this bower.json file stating the versions, this is a list of what we use on this project and Bower just does the right thing. If somebody changes the versions, then you would see a diff in source control, somebody changing this line, somebody upgraded jQuery. Somebody might be thinking why wouldn't I just check jQuery and Angular into my project. Is anybody thinking that?

Student: Sure.

Kyle: The reason why because in every company, there is somebody usually a really smart new hire who says, "Oh I have a problem I know how I'll fix it. I'll go edit jQuery." And like if you're trying to do development and scale, you want to use the libraries as they arrive. You don't want someone just making random hacks to them to fix things. You might have to monkey patch with your own JavaScript file. You just want to use jQuery. You don't want to use some local hack jQuery that somebody thought "Oh I'm going to fix it by editing the line in the middle of jQuery."

[Student]

Kyle: Oh yeah editing the minified file, that's awesome. Because then you look at the diff, you can't tell where the change is, right? And then you have arguments. Are we checking in the real file? Are we checking in the minified file? All those arguments go away. If you're doing automated dependency management, just like you never think about what jar files to check in where you're using Maven. You never think about what library JavaScript to check in when you're using Bower. If you like using Maven for Java, you should like using Bower for JavaScript because they're very parallel. I think there's a couple more tools because didn't we do angular-route? I think we're using angular-route. You do have to spell them, right? And then, what else are we using? What's that other Angular thing? Animate we were using? I don't really know if I care about Animate. I'm not an Animate kind of guy but if I did want to have that in my project, I would just do this.

[Student]

Kyle: That's what I'd do. Here I'll show you in a second if I figure out where my source tree is. There it is. Right now in source tree, it is going to… I guess I already ignored it. If I didn't ignore it, it would be telling me my .gitignore file. K4 node_modules. There's my .gitignore file. I'm actually already ignoring app/lib. Yeah. So there's one more file. There's some other Angular thing we're using. Do you remember? Like angular-route, angular-animate, I think there is another one. I don't remember what it was. I happen to know that when we do testing later, we're going to be dangling with our mocks. So I'm just going to type that now because I'm sitting here with the right command prompt anyway. So I just grab all the stuff I need using Bower and I can see right here a nice declare of list of what I need, the versions are nailed down. They're not going to magically change on me.

[00:48:25]

Kyle: By the way, if you want to know under what conditions they do change, all these little symbols have meanings. And you can go read the documentation for Bower carefully. There's a way of putting a plus in that says let it automatically upgrade beyond this. We've been bitten by that a number of times. I think I'd rather do manual upgrades than automatic upgrades. Any other common tools you can think of? What I want to put out here is that if you can think of it it's probably already in Bower. I bet there's a bunch of Bootstrap related stuff in Bower. I'd be willing to bet. Oh this is going to be a long one. I don't even know where to begin to look there. Wonder how long that was. wc -l, how any lines was that? 263? Ouch. There is one called Bootstrap. There we go. There's Angular Bootstrap which is a bunch of Bootstrap wrappers for Angular. There is the Sass version of Bootstrap. This is jlong version of the Sass Bootstrap. There's tons of stuff in here. Where I'm getting at is that it's not only JavaScript stuff. There's like CSS stuff off the shelf too.

- -

There's like bags of stuff you need for client-side development sitting in Bower ready to just declare dependency and then you have it. That doesn't matter.

There is one change you'll need to keep an eye on though which is when these files were put in to app lib. They're not just put in there flat. Bower has its own way of naming the directories. And so it uses the Bower package name and then it puts all the files underneath that. angular-route is not next to angular.js. It's in the directory called angular-route. So to keep this whole application working, I do need to make one real small change. I need to go into my index which I think is here. And I just need to know this. I need to do that. It's like one little change to get it back to life. It's pretty small impact. And then I'll see if my web server is still running. It's not. So I think I'll just run my web server again. And then if I go in here, local host, I think yeah my browser memory helped me out there. I'm trying to make sure this thing still works. It did. See this data loaded like before and I could type some stuff in. This is my example from yesterday afternoon. It all works exactly the same. The change I had to make was microscopic. I just had to accommodate that Bower has a certain structured system of where to put files instead of dumping it in an ad hoc way.

The next exercise, it's for everybody to make it through all that. You need to have your own project. You should not be working in Angular PhoneCat anymore. So if you have to, copy stuff over. Set up your own source control repo. git init. [Note: I listened to this again.] And then I will bring some commands up in a minute and paste them up into a file here. So you don't have to remember how to type them. You have to run about a half dozen commands to get up and running with automated dependency management of everything you need for client-side development. So to avoid hurting my neck I'm going to drag this over here so I don't have to look over there while I mess with this.

[00:52:15] WORKSHOP

[1:39:38] LECTURE RESUMES

Kyle: So there's reasons you shouldn't use –g. Over in the Grunt world, these guys know about those reasons, so Grunt is a little bit more mature in a sense. They have you do a bash g of what they call the CLI, the command line, just the word grunt basically. And then in the Grunt world, you declare dependency on the underlying Grunt libraries. But the Gulp world is not quite there yet. If you look back in six months, I bet this will say Gulp CLI. But in the meantime just to get up and running, run that command. If you're on Linux, you probably type sudo in front of it or Mac, you probably type sudo on front of it. So get Gulp on your machine. I've already done it so I don't have to do anything with this stuff.

[Student]

Kyle: Yeah you have to. It's because it puts it on your system path, so you have to sudo it. Once you've done that, you can run this npm install command below in your project. You should recognize this command. This is that npm install --save-dev that we've done n times before. And we'll do a different gulp file. So this npm install --save-dev I'll run this one myself because I don't think ever [01:40:56 inaudible] in this project. npm install --save-dev. This is going to go download

- -

it and it's going to update my package.json file. So this is exactly the same as when we did a karma and we did… What else did we do? Bower? So it's the same idea. And just to show you the practical effect, here's my source control changes. Because I run that command, it now shows me two more files that I have dependencies for.

Next time, like if I were to have somebody else pull the file down, the next time you get a fresh install out of source control, if you will type npm install, you'd automatically get gulp.

[Student]

Kyle: Say again?

[Student] Gulp and gulp what?

Kyle: Here. It's gulp and gulp-util. So a nice thing here is the extremely small step you must take to get up and running here. It's no big deal at all. I'll bring those commands up in a minute if anybody needs them. I'm going to get the next step up on the screen. So I, just to make sure things happen quickly, made a gulp file that actually works because I didn't want to use the time today to crate it interactively. And I'll talk through what this does in just a minute. It's part of K4 and it's called… And you don't need to type this yet. I'm just trying to get set up so I can run. So that gulpfile. js that name came from here. This is right on the Gulp website. You create a file and go to your project called gulpfile.js. So that's what I did and we'll get back to that in a minute. Don't bother to type all that yet.

[Student]

Kyle: Well, you can look at it and start setting like I'm going to put up one that's more… Like it actually has stuff in it. It's appropriate for the way we structure our projects. So I can actually talk through it and not really mean something. So did everybody get to these first couple of commands? Any troubles, errors, crashes?

[Student]

Kyle: Say again? They should type sudo and then type in the password of the user that has administrative rights on your machine. By the way, if you don't like having to install it as root, in the same way as I did that, that bin and bower json thing, I put the mouse in the wrong place maybe. Uh-oh.

[01:43:50] TRYING TO FIX THE SCREEN OR CAMERA?

[01:48:11] LECTURE RESUMES

Kyle: So that's one of them and then, that's one. And the SourceTree comes in handy. And Sublime I'm going to need. Okay so, after that nice interlude. So I explain the how you get to—anybody having trouble getting Gulp? Everybody is able to run one of these commands? Anybody had any hiccups? So Gulp is a build tool. Think of it as a slightly more modern alternative to Grunt. I have a nice simple but real realistic useful example. It's not really a minimal build because a minimal

- -

build is just an empty build. Why would you bother to do an empty build? So this is like a minimum useful build and I'm going to walk through what each piece means, but first you need to know this. Do you see this require? This is RequireJS. So when I said earlier that we weren't going to write any server-side node code, I lied a little bit. I lied by one file. This is server-side node code. It's node package manager system but it's basically RequireJS and you type require. So these first five lines, those are using the npm require mechanism to get a reference to this package. What that means is for this to work, you need to npm install those packages. So this will not work until you npm install the packages and give me a minute to so we can easily do that.

So what you end up doing is basically the same thing we've done before, npm install --save-dev because that way I'll get in automatically next time. And I already did gulp and gulp-util. So if I want to use ng-min I have to do that. That's right. And I think my font size changed in that. And if I want to get uglify, I have to list it. By the way, if somebody's noticed, you're allowed to list more than one package in this npm install command. Just like on Linux, you can type apt-get and list a bunch of packages. You can list more than one. And so I want this gulp-concat thing. In both Grunt and Gulp, you'll end up needing a bunch of plugins to do everything you want to do. Those plugins are packaged as node packages and they all run on node. And so like I said earlier, you're sort of stuck with the node ecosystem because that's where all the tooling lives and this is an example of that. So this is all tooling. Like none of this code that we're doing right here is going to run in the web browser. This is all stuff that runs in your dev environment or in your build-server when you work. All the tooling is based on node. All the tooling installs using npm. So I issue this command to go grab those additional packages and it downloads a portion of the internet. That wasn't too bad. So now I have the packages, so that when I run this node program right here, it will work. Those packages will be able to load.

So pretend I didn't type those lines in the middle. This is the simplest possible gulp now, right? Simplest possible gulpfile. It defines one task which is the default. That task has a source and it has a destination. You can see that? So the simplest possible gulp build which runs if you type gulp, if I look over in here, and now's the right place, got to close some stuff. This is my K4 directory. It made a directory called dist, and in that, you see my JavaScript files. These three JavaScript files got copied over. So that's the simplest possible work you can do in Gulp is like copy some files from a source directory to a distribution directory. Did everybody follow that? Anybody have any questions about just that much of it?

[Student]

[01:52:35]

Kyle: Can you guess what the command is to run gulp? Gulp.

[Student]

Kyle: Yup gulp because when you do the install-g, that gives it the ability to put that command in the drive directory to be on your path. So just gulp.

[Student] Is that file already there?

Kyle: Is that what file already there?

[Student] The gulpfile?

Kyle: I'm walking you through a gulpfile I made. You are free to try to do this exact file or you can do something different. I'm using this to explain gulp. A real gulpfile in a real application will probably be 10 times as long as this. But even with this tiny 13-line file, we can do some really useful stuff. But I want to write whatever gets to this point for real forward.

[Student] No gulp install found.

Kyle: No gulp install found, so did you do the npm install --save-dev gulp and gulp-utils?

[Student]

Kyle: Earlier though we had to install gulp itself. But we also had to do… Go back. On the Gulp webpage, I had this page up, I try to use like cite real pages as much as I can because I figure that's what you'll most likely stumble on later when you need to know this. You had to install gulp globally but you also have to install gulp into your project with this command. And if you forget, as long as you remember Gulp JavaScript, you'll find your way here and you'll get through it. You can forget me and remember Gulp and JavaScript. So I'm waiting to get to catch up to explain the last of this file. And you don't have to type this exact file. You could type a different file. I'm trying to teach you just enough gulp and you can make a simple gulpfile. So if anybody wants to follow along just to this point, you wouldn't need this line or this line, that line, or that line. So you could do just the uncommented lines and have a valid gulpfile that copies files. So like this here, if you type lines 1, 7, 8, 12, and 13 and if you have named your files differently, you'll have to put something different here. If you want them to land somewhere different, you'll have to put something different here.

[Student]

Kyle: Say again?

[Student]

Kyle: gulpfile.js. If you forget that, you just go to the Gulp website and it tells you to create a gulpfile and here is their simplest gulpfile. I don't like their example because it doesn't actually do anything. It just says to place code here. And I like to start with my simplest example doing something.

[01:56:33]

Kyle: If you forget how to write it, you go to the website and it's right there. Let me go grab a couple of command lines. And I'll put those in here, taking them all on the same page. Now all in one screen, you can see all the relevant commands. By the way, there's a really slick feature. I don't know if you guy have hit this yet. Isn't that slick? Sublime Text has that,. IntelliJ has that too. I really like that. Can everybody follow so far? Is there any question on what we've done so far? Because we'd picked up a new build tool you probably have never used before. We configured it in JavaScript. Did everybody get that much to run like go over to your wherever you're looking at your files and look that these files, that your JavaScript files that copied to a dist directory. So there's no questions so I'm going to go on.

[Student]

Kyle: We shouldn't be using ng-min yet, so it shouldn't matter if you installed it yet. See I have lines–.

[Student]

Kyle: Well it would only be looking for the module if you put this line in without commenting it up. I haven't commented out, so you shouldn't even look for the module yet. One pain point at a time. The cool thing is the thing we've talked about like automatically putting the bracket notation in, we're doing that.

[Student]

Kyle: For what?

[Student] On the Gulp website, for the global install.

Kyle: I'll just copy it over here.

[Student]

Kyle: There.

[Student] A little bit away from is like right now we would have a Maven POM littered with plugins to do various things to our JavaScript code like minifier. And we're going to eventually just have one plugin or a build set that runs the npm commands.

Kyle: There's a long history of build tools using JavaScript. There's multiple separate communities [01:59:50 inaudible] tool. So a bunch of the tools for example [inaudible] Maven's plugins. That's something you're doing now. There are good tools there, but in 2014 and really '13 and '12 and probably even '11, there's just enormously more activity of people making things including JavaScript-based tools built for JavaScript. So the amount of activity right now making Java build tools for JavaScript is very slow compared to the amount of activity making JavaScript build tools. So Grunt and Gulp both have these vast arrays of plug-ins actively developed like all the things you want to have happen. So my suggestion is start moving forward the JavaScript-based build process to your JavaScript code just because that's where all the stuff is. There's no reason to hack them. There's a little technical reason why you can't write Java code [02:00:41 inaudible] JavaScript, right? You have some of that one right now. But I think you'll find that the amount of gulp you write to get the job done will be like one line of this for every 30 lines of XML you put in [inaudible] So building something complicated, a complex build process, you must have this unimaginable POM. We probably have a 30-step build process. I bet that 10,000-line POM file to build with Maven. It's really painful. This is just incredibly dense and slick by comparison.

[02:01:14]

Kyle: By the way, there's a whole other stack people in the Ruby world, there are Ruby-based tools to do all the client-side build stuff, too. So people that can Ruby Rails stuff, they usually use a stack of Ruby tools to prepare their JavaScript and then file that. So if you ignore the commented lines, we have this really, really simple thing, it gets some source and it writes it to a destination. But let's start doing a little more. I want to have his ng-min. So ng-min that is a tool, the ng which precludes an Angular-oriented tool, this is a gulp plugin that prepares your Angular code for minification. Remember when I talked about in the Angular code, we pull one up. Like right here, you could type manually route provider. Remember you could do all that, that syntax. And I said, I don't like to type that syntax because I'd rather just have it play automatically. I don't want to save that. This tool ng-min, I got to close things I'm not using because I can't even keep track of it. That one's empty, okay. So this tool gulp ng-min, it is a node npm package, so that's why we had to type npm install --save-dev gulp ng-min and then, it's a one-liner with require—that's how you call RequireJS to get it, to get that thing loaded up and instantiated. And it stores it in the JavaScript variable. So in case it's not obvious, this is just JavaScript code. You could write arbitrary JavaScript in here if you needed to. You probably won't need to.

So now we have ng-min available. They have this extremely clever syntax. Node has this concept of a stream. A stream can have an arbitrary number of chunks of data flow through it. Gulp uses node streams as its way of seeing the world. So gulp.src, it does like a file glob match and then emits all those files as a node stream. And then pipes that node stream through a function. The function we're piping that stream through is ng-min which is this ngminifier. So that entire ngminification adds the syntax for you, it's packaged up as like two lines of code which is pretty slick. So if I save that and I go run it, I think I just want gulp to run it. It took six milliseconds. Now if I look at this dist directory, here I have—I need to split my screen. Here we go. If I go over to the dist directory app js and I put it side by side, it messes up the formatting a little bit so don't let that bother you too much. But if you compare, I'm trying to just get them side by side. This config function route provider, it automatically added that syntax around it. So ng-min automatically uses the alternate syntax so that this provides minification.

Then down here, when it said controller function scope, the alternate syntax was to put in a square bracket there and put in the strings, it automatically did that syntax. So it's pretty slick. It means that you never have to manually deal with that syntax to survive minification. I see a look of puzzlement. No? Okay. So if I wanted to minify these files, now I could and it wouldn't break but how would I do that? Well minifying it, that's pretty easy too. There's a thing called uglify which is a minification tool written in JavaScript. I sound like a broken record. It is packaged as a npm node module. You install it with npm install --save-dev gulp uglify. It's now available to gulp. You use this require command to get it loaded into memory, the variable reference. And then with gulp, you can easily just pipe your JavaScript through the uglify mechanism and then next time you run a build. Yeah. Any Sublime Text users? Is there a way to make it wrap this on the screen? I can look but if somebody knows I will be most pleased.

[02:06:18]

Kyle: There we go. It has a word wrap. It's close enough. It turns this into that. If you look, see how it says config route provider but then it says function n, it didn't preserve the string that that name used to be here. If you look right after config route provider, it's a function of route provider so that route provider became n and then route provider.otherwise, became n.otherwise. So that's minification. You guys are running minification, right? Yeah. I still have three separate files. But what I really want is one file. That's also just a trivial bit of gulp work. I guess I never did use g util. I just put it out of habit. concat, this is another gulp plug-in, it's packaged as a node npm module. You install it with nmp install—save-dev gulp-concat. You require it in the normal way, puts in the JavaScript variable and you pipe through it. It takes a name, a single file name so that all the stuff that came into separate files get concatted into a single file and then that pipes on to being written into the output. So now if I run gulp again, now I no longer have—I guess it didn't delete them—but I have an all.js file which now took all three of my JavaScript files , prepare them for minification with ng-min, minified them, concatting them together and wrote them out to a file called all.js.

[Student]

Kyle: There are a zillion things in there so there's a delete operation you can access and you could make a clean target that cleaned out your dist. So I'm not going to go through all the details because we're kind of getting past the edge of what's really Angular-centric. Let me try to give you, show you enough that you can pick it up and go but in a real project, yeah I would actually put a whole bunch of more stuff in here. I'd make a clean target. That's one example. I would take just the library stuff I used. So if I look in lib, this is going to have all the stuff that Bower put in and I would probably just grab the pre-minified ones because why minify something already minified. Because I'll just grab all the min ones, and then concat those together into like a library JavaScript file, something like that. There's a tool that can take your index HTML, so that now my index HTML points to all these separate files. There's a tool called usemin for which there's a Grunt plugin available of the same story which can automatically replace these script tags with like a smaller number of script tags to load the gulp outlet instead of a raw input. So all the tools you need are there but I think just these steps is enough to get you guys up and running and understanding how this all fits in the Angular dev world.

From now to lunch, and then for a while after lunch, I'd like everybody to get just some tiny bit of this running. You don't even need to do as many steps as I did. Just get something running so you know you understand how it works. And then decide you want your Angular application to do something so come up with something. It can be in your problem domain. It can be for fun. Decide on some really simple tiny program you want to make and start making it. So try to apply the skills from the last two days to make some small simple application that does something.

[02:10:16] WORKSHOP

Kyle: What you just said, everyone I talked to in this class has said the exact same thing. So here is we have seen. I've had people at very large companies from the day I've seen more or less the exact same thing is that before or maybe like this back-end and front-end because of their development workflow [02:17:02 inaudible] developers. Of course you're hiring a bunch of [inaudible] but their workflow looks like that. And then after, they're seeing that just about reversed. Because the back-end, before this really consisted conceptually of backend data and back-end GUI work, right? Like JSP and everything. This all becomes frontend work. And so the demand because of what people need to know shifts radically toward like a really good understanding of JavaScript and the DOM and CSS and then on top of that, stuff like Angular, you need a whole bunch of that.

[Student] Oh I don't disagree. I tell you the trick is matching the tools and solutions to the resources that are available right now or projects that are being kicked off this year.

Kyle: Yeah it's a mad scramble right, because if this is what you've been doing historically, then this is what the team looks like. And all of a sudden there's all this demand for all parts of the team, and so we've all this people that need to learn.

[Student]

Kyle: Yeah a simple page.

[Student]

Kyle: Yeah? I had no idea. With anything quite that extreme, so it's about the same thing as what…

Student: Yeah so it was working and then all of a sudden it . . . .

Kyle: Oh okay so technically the way you use clean, if you go to where you download the clean, that will probably give you an example of its use. So let's go back to where it was supposedly working.

[Student] Works.

[02:19:47] ONGOING WORKSHOP, OVERLAPPING CONVERSATIONS

[02:34:48]

Kyle: This is what the source code for Google Calendar, the page for Google Calendar looks like. They actually dump the CSS in JavaScript right into the HTML. So Google Calendar serves this one giant HTML page that has all the CSS and all the HTML in it to make the whole page populate something for being close to one HTTP request or not much passed.

- -

[Student]

[02:35:30] ONGOING WORKSHOP

[02:48:19] LUNCH

[03:39:00] WORKSHOP RESUMES

Kyle: You're looking at your files again.

[Student] I was inspecting the scope. What I did, I'd change the spec. I was trying to put the array anywhere else but I don't know.

Kyle: I think I know. Yeah I think I might know. I think you actually have to watch the thing because its data binding happened over time. But you need actually to watch it. So what do you got there?

[INAUDIBLE CONVERSATION/ASSISTING STUDENT]

[04:13:06] Kyle: Give me a minute here to make sure I've not missed anything. Unit testing is the large one. We'll talk about the end-to-end testing tool briefly but it's just too big to attack in the scope of the class. And then I always leave a section at the end for just general questions about how to build apps. People ask me questions about how they're supposed to really build apps. And I can talk about that. I can talk about interacting with servers, organizing the files, how to divide up a large application into parts but that's kind of driven by questions. So that's the remaining subject matter. So broad questions, testing, build something while we're here to help. We got to get out of here at 3:45 so that is… 3:45 so that's two and a half hours from now. That's enough time. It will work. We're fine. Try to close this stuff we're not using here. I'll put this code up. That's the last thing we were working on. Let's go.

[04:15:53 INAUDIBLE CONVERSATION/ASSISTING STUDENTS]

[04:53:25] LECTURE RESUMES

Kyle: When one person comes back we're going to talk about Karma, so prepare yourselves. I should see if I have anything to commit, so I always commit before I go on. So this is the change if I need to see the location. That's all fine, that's fine. So I added Gulp. Look I need to ignore the stuff in dists. I remember we talked about what to ignore. I'm trying to find my .gitignore file. Ignore dists. So the next step is to add Karma. I'm going to pare down my open files here a bit because we're going to be attacking new things. Karma is a server-side development tool for managing those with npm. So npm install, I think we all already did this. --save-dev karma. I already did this so it's not going to take me time. I thought I already did it. It was nice and fast. As long as we're talking about it, I might as well have this webpage up. So when in doubt, search Angular address or something. It's kind of a pain. It didn't give me a very good search result. Use Google instead. I can't look at the screen as well today because of the idiotic fire alarms going off

- -

in the middle of the night at my hotel. I didn't sleep very well. So I slept on my neck wrong and so I can't turn my neck to my left as easy as I'm supposed to.

[Student] You should have said fire alarm's going to keep going off. I'm going to give it reason.

Kyle: If this is what was on fire, I feel like it's worth it. Something is going on here. Why is this… It's not… I don't know what. Why am I not finding the right web page here. Something is awry.

[Student] You're waiting for Karma's homepage?

Kyle: Yeah. I'm trying to figure out why I'm not getting Karma's homepage.

[Student]

Kyle: Yeah that's bizarre that it doesn't find this. So there's a funny back-story here. It was called Testacular when it first came out. This is widely considered a humorous name. I think it might have been some kind of trademark problem. I don't know if they changed it only because it was humorous or because of the trademark problem. But it was re-named to Karma. They just haven't replaced this video yet. But it's very much worth watching this video sometime this here if you want to understand this tool well.

[Student]

Kyle: What?

Student: I'm just kidding. We have the same one but…

Kyle: I like your Mac sticker. It's pretty cool. What was this thing here? I think we just have to leave this off and that was the actual homepage. Okay so to get this tool you can either do this command npm install-g karma or if you did the install that I did, this is just going to put it in this one project with its version control. It doesn't matter which one you do. I think I've actually installed it both ways. Let me see if… Yeah okay. We've done npm init. We've done bower init. And we've done git init. So you can probably guess the next command, right? So karma init will make a karma configuration file for you by asking you questions interactively. As you might guess, I'm going to mostly just press Enter at this because I'm going to go edit those things later. So I just pressed Enter a bunch of times. It didn't really think. I just press Enter until it stopped asking me stupid questions. This is the file it made for me and I'm going to put this side by side with another file. This is one where the file it makes for you is not that good. Let me see if I can get this where you can see them side-by-side well. Now we got to talk about these files. One that it creates for you is put all these helpful comments in. So if you're working on this on your own computer, it could be a good idea to leave all those comments in. But I always remove them all because I like files that you can look at all on the screen at the same time. Actually like them, not only to show everybody , I just like it when all the relevant information is on my screen. So I started it something like this and I chopped it aggressively down to this. As I look at this, it looks like this last time using

Angular resource, so I'll probably leave that behind. But the reason I brought this one up on the right here, this is just from a different example I did. I'm just going to copy and use it over here.

[Student]

Kyle: This whole thing? I could maybe. I could email it to you.

[Student]

Kyle: No you can't. Want to see why not? Here we go. I'll do it that way. If I need to, I'll send it to you.

[05:00:00]

But the important thing is actually understanding it. So if you just wanted to have a file, you could pull up a repo somewhere, and I'll probably even put this repo up, where you can get to it, if you want. This is a good place to start understanding why order matters. So here's the index HTML for this program, and here's what I've done in Karma. You note that I'm not using jQuery over here. I think for the sake of understanding, I'm just going to start using jQuery over here. There's a whole bunch of relationships here where order matters. That's why I bring this up and talk through it. So knowing all that, over here, if you're going to use jQuery in Angular, you're going to use jQuery in your page also, always do it first. There's logic in Angular that if it sees the dollar sign already defined by jQuery, Angular will grab jQuery and use that instead of its built-in jqLite. Follow me? So this file order matters. Because this file order matters, this file order matters. You always want to put jQuery before you put Angular. I put Angular before I put Angular add-ons like and Angular route. So likewise, over here, I put Angular before I put add-ons like Angular route.

On the left here, you see Angular mocks. You should never see that on your page because angular-mocks is useless on your page. Angular-mocks is test-support for Angular, stuff to make testing go more easily. After I put all the Angular stuff, then I put my application code so this one line corresponds to these three lines. And then last, I put my test job as script code. I will probably explain that like four more times walking around and it's really tricky to get it right. But for the moment, we'll move on and make it work. On line six, I took the default which is Jasmine. Here's a bit of confusion. Karma, like oh this is a testing thing. It's just a test runner. It doesn't actually have any testing tools in it. Jasmine is an actual testing tool kit. This actually has testing tools. Who's used JUnit? Okay. The thing called like NextGen Test or something. It's pretty popular in JavaScript Java. Are you guys using that or just JUnit?

[Student]

Kyle: QUnit? Okay. So QUnit is an alternative JavaScript testing thing. You can use it with Karma. So if you prefer a QUnit, there's a way to keep using QUnit. I'm not going to bother because I'm just going to use Jasmine because that's the default for Karma. But let's just remember, Jasmine is a testing tool like QUnit. So whatever you know about QUnit, probably 90% of the same things are true.

- -

[05:03:54]

There's little bit different styles how you write them here. Like this style where you say tests for example. That's like the TDD flavor of language. There's like two schools of thought. There's people who say TDD, people who say BDD. You guys heard BDD? As far as I can tell, the difference is only in the words you say and what your name thinks. As far as I can tell, there's no actual difference in what you do if you're doing TDD or BDD. All I know is that the BD people that kind of hold their nose just a little bit high because BDD is a more recently named thing than TDD, say "Oh TDD, you're doing that in the 90's." So actually don't know if there's any real meaningful difference. But I do know that when you see words like test and okay, you're looking at something that is using TDD-style naming. And if you look at something that is saying describe it and expect using something that uses BDD-style naming. So this means test and this is a test suite or a group of tests, however you're grouping tests together in QUnit. So what do you call the thing that holds a bunch of test in Q Unit.

[Student] Module.

Kyle: So that's a QUnit module and this is a QUnit test and this is a QUnit okay. So hopefully that the difference in the syntax won't be too jarring. So back over here to Karma. Karma is a test runner. It doesn't have anything in it that does any actual testing. There is no modules or asserts or okays or groups or anything in here. It's just a way of executing tests. But it does have just enough knowledge to make the right call for each of the libraries. So here where I said Jasmine, you would say QUnit. And then we saw that there's a node module that enables the QUnits support in Karma. So you got to type npm install for that node module. If you're happy with QUnit, I would recommend continuing to use it. Not changing just because the default is here. This is the default. Angular itself has a thousand something test written with Jasmine. And so a lot of Angular people use Jasmine but you don't have to. There's no connection at all between Angular and Jasmine.

So I think the next step is to write a test. I had decided here that I was going to make a top level directory called test and put my tests in it. They go up here and make a new folder. And then, I'll go in there and make a new file. My tests. Something like that. My tests. This pattern will match my test. This will match all my code. Again, the order matters, right, because in my test, I'm going to do things that refer to my application code. If I refer to application code that didn't exist, it wouldn't work. Something implicit here, you might not realize, see how I'm listing all my JavaScript files here. This implies that I'm not going to use the HTML file. So this unit testing makes no use of your application's HTML file. So this is a different list of files. Now it happens to be almost the same list of files but it's not quite the same. For example, you get to do things like catch all your JavaScript with a single line in here.

Now I need to actually put a single test in. Where would I get one of those? I don't feel like making. I type one of those. I'm just going to grab one that I did last time. It's a good starting point. That's not what I meant to do.

[05:08:15]

Here's a couple of tests. That first one is pretty trivial. It's just testing that true is true. The second one is going to be a little more complex. It's checking that 47 is 47. You all already know TDD pretty well. I can go over a lot of stuff. That's good. If you want to know the details of all these things, just look at the Jasmine page. I don't know how it really compares with Q Unit. But it has a vast array of built-in ways to test things. Has things that it called spies, which I think they're a little bit… I think some people call those mocks or place holders. The language is all different tools you'll be familiar with. There's one interesting bit in here. Where is it? Yeah has a slick ability. You can mock-out the JavaScript clock. So if you have stuff that's interval-based, there's a way to put in a fake timeout mechanism. And you can make your timeout happen immediately. So your test, so you can still run immediately even if it's testing code that has timeouts in it. It's pretty slick.

So I'm ignoring all the complexity in Jasmine. I'm just showing you where it is in case you want it, and I'm writing a couple of trivial test. Now I want to run these trivial tests. So far, all I've added to this application is one configuration file and one file called Tests. So to run your tests, you give Karma a start. Actually, you know what I'm going to do? I'm going to make this a little bit easier to understand at the beginning. Do that. Karma start. See my warnings about not matching any files? It looks like I got some of my paths wrong, so I got to fix that. Look I'm inside the app directory. That's kind of odd. That's not what I meant to do. So I'm going to move that Karma file up one. I meant to put it up there. Get my Karma back open again. Okay, Karma start. I hope I don't get a bunch of warnings this time. Now I'm still complaining about what do they think about jQuery and Angular not matching any file. When you see these things, that's a pretty strong clue that I did something wrong in terms of my paths. I can tell you where it is. Example, where I copied this from, I had named my lib libs instead of lib. So what I need to do is did that. So if you get errors, track them down. This time I didn't get errors. Has anybody used—I know you use Karma. Who else uses Karma yet?

Here's the most jarring thing about Karma and the most common mistake is that when I ran Karma, it opened a browser. This browser has a green bar which initially made you think maybe it's telling you about test failure and success but it's not. This browser, it's only purpose, is to be a JavaScript runtime environment. This browser is actually not for you. It is only for Karma. So you, the developer, you're not supposed to look at it. You're not supposed to close it. You're not supposed to close it here. You're not supposed to open up new tabs and start browsing CNN in here. You're not supposed to open up and do your email in here or go to the Angular website in here. You're not supposed to do any of those things. The only purpose of this browser is an environment for Karma to run tests in. And you'll note that your application will never appear in this browser. It's just talking to some port number that the Karma is using. If I look at my web server, and I'm running, I can actually kill that web server, so I'm no longer running the web server for my application. If I stop Karma, run Karma again, that thing is going to start up even though I'm not running my web server. So it started on my monitor. So see it started up and it's not trying to fit. I'm not even running the web server anymore. So the browser Karma starts, it's not for you, it's for Karma. It doesn't talk to your web server. Rather Karma is grabbing your code that matches these patterns and injecting it into that web browser.

[05:13:09]

It's actually using a web socket. So if you dig around, you can find that there's a web socket connection between the command line program here and this browser. And this command line program is looking at your files, grabbing their contents and injecting them into this browser using a web socket. So there's some pretty cool plumbing going on behind-the-scenes. That plumbing is what Karma does. It doesn't do any actual testing. It does plumbing.

Now I have a few more errors. See this Angular is not defined. Wait, no. This is from before. I need to make this more obvious by cleaning my screen. Is it clear? Clear. Okay, this Karma's web browser and if I manage my screen space carefully I should be able to edit files and have you see the changes immediately. So let's try this. If I do this, and then if I do that, and if I do that. I'm trying to manage my screen space where you can see what's going all at once. This is much easier to do if you're not trying to show a whole room of people or you have multiple monitors and so on. So the intention here is that every time I edit the files, my source files, it's going to automatically re-run my tests. See where it says… Oh come on, really? There is the two over there and the other two is over here. So it's a little bit hard to read because of the size of the window. But if I go edit a test here, I'm going to hit Save and you're going to see two things happen. This briefly flashed. This program injected the updated JavaScript into this browser. This is just JavaScript written using Jasmine, so ignore the fact it looks different than QUnit. So it had already injected Angular and all the other tools. And then objected by code, injected this test code, never try to change it, it re-injects it. It re-runs it. It re-injects in there and re-runs it. And then record the output here. So the output does not appear here. So this browser is not for you, it is for Karma. It appears here. You can use [05:15:49 inaudible] by the way, if you try to have a window here.

So I had one fail. So if I go up over here, I could probably pretty easily get two failing. Now I have two fail. And then I could go over here and I could fix it here, fix both of them. And now I'm back to zero failing. If I was concerned that the JavaScript I was writing would fail, if I used a different browser I could just manually launch a different browser and it's totally okay to manually launch another browser. Karma has the ability to manage the browsers that are running but you don't have to use it. You can just manually launch another one. Now if I've made some change, it's going to re-run it on both of those connected browsers. If I wanted to make a slightly more complex set-up, I could get my phone on the same Wi-Fi so I could talk to the other and I could actually have my test, my unit test run on my cell phone at the same time as it runs there. That's really helpful to do by the way.

[Student]

- -

Kyle: I've had trouble getting any useful behavior out of that. I have not had to use it because I've kind of been doing TDD for a long time and I'm kind of trained not to use my debugger when doing TDD. The debug runner. What does the debug runner do?

[Student] And I think in the browser, you see here, I think I've used it once.

Kyle: One thing you'll notice is if you go over here, and you keep the console open, it says its console cleared. So the purpose of this is to remind you you're not supposed to be looking at this browser.

[05:18:02]

So this is not a bug. This is a feature. But there may be something going on here. There you go. I guess this is a way of getting it to show the output in the browser window so that if I really want to know what's going on, I could type debugger right there and I could say that and it's going to re-run it. And if I look in the right place, it should have got me in the debugger. Yeah file's in debugger. So it actually got. I can stop my test and don't mess with the debugger. I don't really care about that. I hardly ever do that. Okay. So now we're back to everything passing. So everything up to this point is just making Karma work. Now if everybody wants to take a stab at just making Karma work, we will help you make Karma work. Once we have Karma working, I will show you how to use it to Angular-specific testing. But let's just get it all to work. I probably need to leave a couple of files up.

[Student]

Kyle: Where it pops in the corner of your window? I just leave the window open. I'm really old school.

[Student] Well it's nice if you don't have dual monitors.

Kyle: When I'm developing, I use this with an external 27-inch monitor. So I don't have that problem. But hey, if you really like integration, switch to Web Storm. It has integration in the box so it knows about Karma. It will actually report the errors and it will take you to the test immediately. So everyone will need to put the right stuff in this file for your application. It could be a little different than mine and any questions about why things are there in what order, I will answer.

[Student]

Kyle: When you use a glob like that, you have no control of the order. So Angular needs jQuery but first Angular route needs Angular to be first.

[Student]

Kyle: If you had 20 or 30 library files, it was not worth the effort of [05:20:42 inaudible] The thing that gets you and I've seen people get bit by this, here's the thing to never do. Sometimes you see a very clever person think "Oh you know what? I'm going to name all my test files .testjs and then, I can just watch for those anywhere in my project." So never do this because your projects include no modules thing that has 6,000 files in it. So if you do this, it's going to keep looking through your 6,000 files in the node director to see if any of them matches this pattern.

[Student] I did that but I left the test on the front.

Kyle: Yeah oh it's fine to leave the test on the front. It's fine as long as you have the test on the front. Just don't ever start looking at the top of the project because you'll end up watching a bunch of files that are sort of there incidentally like node modules. You won't think of node module because you didn't personally put the files there but it can have six, seven, 10, 15,000 files in it. You can actually get an error. On a Mac there's a limit to how many files that you can watch. There's actually a command you have to go find on Google and type if you want to watch more than some number of files. You probably didn't hit it if you didn't hit the error.

[05:22:06] WORKSHOP

Kyle: Does Q Unit have this? Actually I'll say this. Jasmine has some cool features. If you're working heavily on one test, you can make it a D describe instead and then it just runs that one test. It doesn't run all the other tests. So if you're heavily working on one part of the application, you can scope it down and keep quickly re-running the test you're working on instead of re-running the entire test with every set. It's really nice. And this describes the nest. So you can group nest them as deep as you want and you can still turn them on and off with that D describe at any level. It's really helpful. I think you can put an X in front of it, if you want to specifically exclude one. Because x [05:24:02 inaudible] then you'll say I don't want to run this at all.

[ONGOING WORKSHOP/ASSISTING STUDENTS]

[05:38:56] SHORT BREAK

Kyle: So what we did here is we only did kind of trivial testings. We kind of rubbed this shiny tool that knows how to work with Angular. We didn't use it with Angular. So I'm going to check this in and then I'm going to add the bit that's needed to work with Angular. So go over here. All I did is I added jQuery because I wanted to be consistent. I put in this Karma comp that works and I put in one test with some trivial files. Pretty small stuff in terms of the changes to have unit testing. So unit testing but no Angular. Now I want to test Angular. This is when the mock thing comes into play. Over here I have a thing called Angular mocks. You should never see Angular mocks loaded on your page. It's only for unit testing support so it only goes right here. You should look in your lib directory and see if you have it. If you don't have it, you know how to add it. It is this command—. I got. It's hideously ugly. It's this command if you don't already have it. Do this to add it. Certainly don't go download it by hand. Angular mocks add some features to Angular to facilitate testing. The really interesting thing about how it does is it pollutes the global scope to do that.

Polluting the global scope is a really nasty thing to do for production code but there's kind of this convention that testing frameworks and testing code is totally allowed to pollute the global scope

- -

to make the coding more convenient. Because with automated testing,, the fundamental challenge of automated testing is getting any programmer to do it at all. So anything you can do to lower the difficulty including polluting the global scope is generally considered okay. For example, this Jasmine over here, I didn't have to apply Jasmine..describe, Jasmine.it. It didn't call it dollar sign or underscore or tilde or whatever. It just threw these things and some more stuff on the global scope so that I could call them with maximum convenience. Angular mocks does that with some additional things. If I want to look at some code I've written. I don't want to look in dips, right? It's minified. It's worthless. I'm going to look at some code I've written. It does almost anything at all here. We'll test this BCR controller. It's as good as anything. Pretend that this thing had to do something like that. I could write a unit test but this actually does it, but this controller does something useful. I'm going to put something even simpler to test though because controllers are slightly complex to pass and I want to start the simplest thing first.

So let's say I had a filter. The filter is called… Bad name for a filter. And that's a function for injection and it returns a function. And that function takes a value, and it returns the value plus up. So it's like the it's a really lousy filter but it's easy to test. So I want to write a unit test for a filter. How would I do that? This Angular mocks that we're already injecting on the page adds some capabilities. So let me try to introduce everything with its web page. Let me find our Angular mocks. Here we go. This takes me into the ng-mock but I can probably go to ng-mock. Here we go. So this is where we type NG mock which is the internal name for the thing. Notice it's in the JavaScript style. This gives you some instructions on what's available in the box of this thing. The important ones that we'll actually use are module and inject. We'll use those heavily and this is important note it's all published on Window, Window is where the global scope is. So it's published globally. So I don't have to say angular.mock.module or angular.mock.inject. I can just say module or I can just say inject. So Angular mocks added those two things to my vocabulary, that way I can just readily do. Describe, testing a filter, function. Now this is not Angular code right here. Failed to close my… This is not Angular code so this is not a point I can do injection, I should point out. This is just ordinary Jasmine and I can say it should add—what does it add? It adds up? Okay and that takes a function.

[Student]

Kyle: Yeah. I like making mistakes. It keeps everybody on their toes. So what I'd really like to do here is I'd like to have a handle on that function. So imagine that I was just doing ordinary JavaScript and I had some function called Up. So imagine I was testing that. Then I could go in here and say expect up of x to be x up. I should be able to do that. So since I should be able to do it, I'm going to see if it works. I left this running the whole time so I'll save that. I had one fail. Testing a filter should add up, failed. Okay so why did it fail? Undefine is not a function evaluating up. Undefined is not a function. What's going on?

[Student]

Kyle: No, it's right here.

[Student] This trip.

Kyle: Yea it's right here.

[Student]

Kyle: I don't think it matters. That still failed down there. Undefined is not a function evaluating. Okay so is this up thing the problem? Yeah yeah. I closed. I didn't follow the pattern right. It's tricky to follow the patterns on these things. You call expect on a value.

Student: You're missing a…

Kyle: I'm missing that. I call expect on a value and that gives me back this expectation object that I call to be on. It's a little tricky to use these things right but since I don't see any green down there, I know it all worked. This is how I just test an ordinary function. And I wrote it here for convenience but it would be also completely acceptable if over here, if I was just doing ordinary JavaScript none of this Angular module stuff, I could do it there. And I think if I can remember the way to do save all—oh it's kind of funny. When I first say one of these, if I saved them in the right order, I won't get any errors. So there we go. Now it ran even though I defined the function over there.

[05:51:47]

That's pretty easy right? What's the execution model, right? It just appends all the JavaScript together and runs it. So if I'm doing global link spaced functions, it's trivial to make them available inside the test. But the problem is I'm not. I actually want to test an Angular filter. So there is a way to do that. I could clean up my… We'll take that out of the picture. And now if I try to re-run this by saving, it's going to fail because there is no up so my function is not there. So how do I get a hold of that. Here's how you get a hold of it. So you call that module. I mentioned you have module and you have inject so what module does is module is equivalent to this. So module says go load stuff. It tells Angular, "Initialize those modules, make them available for testing." If I was going to be ng-apping Kyle.app, I'll just take a quick peek over here at app.js. This is the module that has the code in it that I'm testing. So I need two modules. Is that enough? No that's not enough because there still is no up but you have to do that first.

Well inject is the other thing we need to use. Inject is a function. Inject is a function that you call with another function. And as you can guess from the name, at this point, you have Angular injection available. So I can ask to have up injected and if I do that and I haven't made any mistakes, what am I doing if I made a mistake. Unknown provider, up provider. Oh filters are a funky exception. You have to manually get the filter tool and then you have to manually say that. There's couple of things that are not available in the normal way so now that succeeded. See that's all a success. And it actually tested on both of my browsers because I still have two browsers running. So this is way to test something, to test a filter. You need to use this inject function to get access to Angular injection and then you use that filter as a function you can use to get a specific filter. Everybody follow that so far?

[Student] So would factory work the same way?

Kyle: Would factory work the same way? What if I had a factory I wanted to test? Let me clean up this test a little bit then we'll test the factory. So there are some nice little features in here that you can use to make this cleaner. I don't remember if it's called set up or before each. Is it before each? Okay because every one of these stupid things, and I use like four of them, use different namings everything. So before each, like that, this is a way of saying I want you to do something before each function. So I could move my module call up in there. So if I did this, I hit save. So it still works. I don't see any red down there so it still works. So that works. That's nice because if I was doing more than one of these it tests, I wouldn't be loading the module every time. Remember I said these are nestable. I could create more groups, more sub-groups in here by nesting these things. You can nest them arbitrarily. One's enough for understanding. That's nice. The interesting thing about inject is that it takes a function but it also returns a function. It expects to be handed a function. Well here I'm just kind of pointlessly adding a function wrapper so I can actually remove that extra function wrapper which means that the syntactic overhead mostly goes away. I had to say function anyway so that the overhead for testing an Angular thing is now down to saying the word inject and saying the name of the thing I want injected because I had to say function anyway to make the whole thing correct JavaScript. So let's make sure that really runs.

[05:56:39]

I got an error. What did I do wrong? Syntax expected token. Oh yeah here you go. Set the bound through params, can't remove that requirement. So we're all green down here so it's all good. So the syntactic overhead is not too bad because even this part here I wouldn't be adding more of those if I was adding 50 more things to test. Now I have the question of testing the factory right? Well this factory is it's kind of a painful thing to test so let's make some simpler factory that's no big deal to test. So factory, what do you want this factory to do? He doesn't have anything for me? This will be a factory which maybe has a function in it. It's like some utilities so it's a bundle of utility functions. And it's not going to need anything because I'm going to put a trivial function in and it's going to return this literal object and that's just going to contain one utility function and that function is going to be called bang. That's going to be a function that takes a value and then returns. Did I do that right? Where's my cursor? Returns the value with the bang on the end. So I guess I'm short one. So I didn't see any errors down there. That means that I'm syntactically correct. Because every time I change anything and press Save, it's going to go through that motion of injecting the JavaScript and telling your browser to do it. The amazing thing is you can put a lot in here. I could put 20 libraries and a hundred of my files and it still wouldn't take that long to inject all that JavaScript and run it. Browser execution environments are just amazingly optimized. It really helps them do some TDD.

- -

So I want to test this thing. How would I get that? I want to test some Angular stuff because I don't want to repeat this before each stuff. So I should say that this—and if I want to be strictly correct here, you're supposed to make this read like English. It should add a bang and then I know I'm going to need to get a handle on that so I got to do injection. And that takes a function and I think this is one of the cases where you can directly say the name of the thing you want. It's not entirely like I always have to try it each way to figure out why I want to do the other. So I got to type this right. Over there's where I start the function and then I got to close. And then I'm going to have to do an expectation. Let's see if I got this one right. So I want to expect utils.bang because I expect utils to return a thing called bang. I'm not going to bother to check that it has a thing called bang because if it doesn't I know this is going to fail. Because I'm sort of implicitly testing that it's present and that it works. So let's see if I got that. Yeah I got that one. So that works. Filters are kind of a special case. You have to inject filters and manually look up the filter. But the things that you use all the time like controllers and factories and so on, they're just sitting right there and you can inject real easily.

The last workshop of the day is test some Angular things. So put some code in an Angular factory or filter, whatever, and test it end to end. And when you get back, go back to working on your program you're making to do something. And then at about 3:30 I have a couple of miscellaneous topics to talk about and wrap up. That could include events for example, if you're interested.

[Student]

Kyle: Get access to what?

[Student]

[06:01:05]

Kyle: So I can show you how to do that if you want to test the controller, right? There's a way to do that. It should set up the scope. And then I'm going to need to do injection. Inject. And then whenever I say inject, they have to say function. I'm trying to make this a little smaller so you can see it. So what I need to have injected here is the controller that I want to test which is BCR and a service called root scope, and you'll see why in a minute. With this root scope service, I can make my own scope. So I can say something like var s. I can make my own scope. =root scope and then there's a convenience function on here called new to let you make a new scope. So s is now a new fresh empty scope. Then I think I need the injector service for this one, but I don't recall for sure. Paul, do you remember did we use the injector service? Yeah. I did the same thing in one of my previous tests here that I have saved. I wish had saved them from the beginning. I only realized with doing this that I should be keeping my example. Let's see here. Does this one do it? No, I didn't do it on that one. Does this one do it? Yeah. Let me bring this one over. So here's some code that does it. I'll just paste this in here so I have to work from and save time. Yeah, I remember now. Instead of injecting the controller directly, you can inject the controller service which is a look-up service for controllers. And then you make a scope by calling root scope new like I did there. And then when you call this guy, you can override the injection. So this controller service has some test support stuff. So you can say do you want to test this controller and then you can say there's what it's looking for. So I can provide it, I can locally say don't do whatever global lookup you would do to get a dependency. I'm going to just give it something. So when it looks up that, it's going to get the one that I just made. And so the one that I just made, that means I have a reference to it.

The curious thing here is like this controller, this actually returns a reference to that controller. I didn't even need it so I didn't even keep it. So then I can do an expectation on that scope, and over here on this BCR, all I did was set a single field A on it, and then I can use that sort of test. Get all this crud out of the way. So what happened here? Should set up to scope fail. Oh look at that! See? The types are different because when I wrote this code for whatever reason, I made it through the string 44 and that picks up the difference. So if I do that, then I'm back to green. Does that answer your question? You can test the controller. Yeah.

There's a similar and longer invocation you can do to test the directive. You can take the HTML and say run the directive and you can verify that it did what you expected to the DOM. Everything you can do there is a way to test.

[Student]

Kyle: I got to do some juggling of my screen resolution here. I can give up being able to see the window there for a while. I can switch this over to notes. That's the command right there. You guys email me so that I don't forget. I can put this code somewhere so you can get it.

[Student]

Kyle: I would but the nice thing is that since everybody has that need, that need is already very well-met. So if you just know that it's there, and then it's called HTTP back-end, there's already this nice, fake HTTP implementation.

[06:07:14]

What you can do, I'll just show you the code here because they already have it. Yeah, okay. So here's a controller that calls some HTTP stuff, right? So you should recognize most of what you see here. It's like a before each. You've seen that before. You've seen inject. See, these guys are actually asking the injector for HTTP back-end but you can also just put HTTP back-end there. So you say HTTP back-end, when somebody does a get of this URL, respond with this JavaScript object. So there's right there in line you configuring the HTTP back-end to provide some data.

[Student] So when you're in tests, the act of getting that HTTP back-end then replaces the HTTP?

Kyle: Yeah. You need to know this. This guy, like this does a lot. This rewires Angular to support testing. So it's like HTTP back-end appears in there. You don't have HTTP back-end when you're running a real page. This adds all this stuff you need protesting. That's why you should never try to put this on your page or you will get bad results. Many have tried and gotten better results.

- -

[06:08:48 inaudible/too far] You want to see app.js? So I didn't do anything inside phoneapp.js. I just made up like the simplest filter only ever in the simplest factory and other trivial controllers so I can test them. So you should be able to test one of your real ones if you have one handy.

So how does this compare to QUnit?

[Student] As a language, there's more typing.

Kyle: More typing in QUnit? So what's smaller in QUnit?

[Student]

[06:10:31] TALKING ABOUT QUnit

Kyle: So this is only one of the alternatives? Some people say Jasmine is like the most. What's the other one? There's one called Mocha. The crazy thing about Mocha is that the expectation language is not even part of Mocha. You got to use a different tool to form the expectation language used inside Mocha and there are people who have very strong opinions about which of the two different expectation modules is better. Like the one called Chai, some people there's rapid [06:11:29 inaudible] between Mocha and Chai. So it's just crazy about it than Jasmine. I've only just used this one.

[Student] Anybody can come on and read the sentence and know what they're trying to test for.

Kyle: Well that's what they aim for here.

[Student] As far as mocking goes, what I use [06:11:57 inaudible]

Kyle: Yeah, I've heard of that one. So this has enough mocking built in. I'll just put this up here to see if anybody's mind is too blunt. So in our stuff, we actually felt like typing HTML because it has too much duplication and visual noise. So we actually type only Jade which is an indentation-based syntax for HTML. So we type things like this and they get rendered like this as part of the build process. It's indentation-based and all the closing is automatic. So it has the amazing feature like there's nothing you can type here to make your tags not balanced.

[Student]

Kyle: We mostly still use Grunt, Grunt watch. It's doing like Sass and it's doing Jade and that kind of thing. So we don't type CSS directly. We always type Sass instead. We don't type HTML. We always type Jade instead.

[Student]

Kyle: Yeah. You can do a Gulp easily.

[Student] That would creep some people.

Kyle: Yes. As I said like for programmers who have like maybe used or seen Python or something, they tend to be like "Yeah. I can kind of see that." Kind of like a straight designer who's been

- -

spending the last 20 years doing this, looks at this and thinks you are crazy. Anyway, I'll put the code back up here. It's kind of funny. I actually type more HTML when doing classes than when doing development even though I do way more development than classes because I only type Jade in development.

[Student] Once your application is set up, you work so much in your JavaScript and not the HTML.

Kyle: And we still type JavaScript. A lot of people type CoffeeScript instead. Okay, let's go around and see if there's…

[06:14:59] WORKSHOP/ASSISTING STUDENTS

End of class transcript.