EVERGREEN 2020 ONLINE CONFERENCE

WEDNESDAY TRACK 2: ANGULAR CLIENT INGREDIENTS

JUNE 10, 2020

CAPTIONING PROVIDED BY:

CAPTIONACCESS

contact@captionaccess.com

www.captionaccess.com

>> It’s 4:00. Thank you all for coming. This is Bill Erickson's Angular ingredients. I am Amy Terlaga. I am hosting this session. I am with Bibliomation. Bibliomation is sponsoring this session. Closed captioning is being sponsored by Equinox Open Library Initiative. We’d like to thank our captioner. This is session is in meeting mode, not webinar mode. So please leave your video off and your mic off and use chat when asking a question or commenting. I think you can ask questions verbally, but if you are comfortable with chat, use the chat. I am pleased to introduce Bill Erickson. Bill, take it away.
>> Thank you so much, Amy. Hi, everyone. I have no idea who I am talking to. I haven't seen you all in so long, and I was really looking forward to seeing everybody. So I will just look at your names instead and say hi.
I don't have the chat open. There is the chat window. I may not see everything that is going on there, but I will try to glance over there occasionally. I am going to talk about Angular -- today, as is often the case with my talks, it's pretty code heavy. And I do these as much for myself as something I can return to later as a reference.
Some of this may be a little eye-glazing at times. I will try to keep it moving along and kind of like. It is a little bit short attention span theater because there is no story arc or narrative here. It's a pretty random mix of stuff. So the idea was to talk about kind of the main pieces that we use when we are building interfaces on the Angular side. There is certainly a lot of that going on, but I also wanted to make a point of mentioning things that have come up and been problematic to me or certain themes that I see over and over that you know, might have some advice to offer. There is a fair amount of that going on, too.
And like I said, I'm happy to take questions at any time. In fact, it probably a good way to pause me for a second. Otherwise, I will go ahead and get started. I don't want to spend too much time on some of the obvious things. My first light here is about the grid. The grid is one of

the things that is in almost every interface. And whether or not you want to our care about it, you will eventually understand for the most part how it works because it is so ubiquitous.
I thought I would mention one or two less obvious things. One thing I make it a point of doing Now, anytime I built a grid, is I try to imagine the ideal layout for someone using the interface. And one of the things we can do is define specific columns by default and have the other columns that can be added as needed.
Another thing we can do is tell the columns how wide they should be by default. And I don't know how, knowledge this is, but there is a flex review, which I will highlight here with the mouse, which allows you to send a proportional with or -- set a proportional with the recall. So if you have a grid with something like an ID column which is generally going to be fairly narrow, it's just a narrow, something else like a title, you're going to want the title column to be wider by default. So you can just set a flex value, this is saying the title column is three times the size of the ID column.
And discus you a chance to create something that at first glance makes more sense to anyone using the system and it means they are less likely to have to go in and change the column size for themselves.
Something else that is a little bit less known that is fairly recent is this idea of a cell text generator. Someone just asked they should be hearing audio. I am talking. So I assume someone is hearing audio. But I would like to confirm that. Okay, great. Thank you.
It seems like audio in general is working.
One of the features that is available in the grid is the ability to print it in tabular form or export to CSV. But you can also create grid cells that have complex data that isn't just a text. And I have a picture at the top from the Angular catalog of the barcode column pointlessly -- the barcode and gives a link to do it and they link to jump to the edit interface. So if you wanted to print something like this, there would just

be a big check of HTML that would make.-- not make a whole lot of sense that column. So to allow for better formatting of those types of data to create something called this cell text generator on the grid. Anytime it finds a column that has attempted to find, we'll look to see if there's a generator for in the code and here we have a column named barcode and find the generator, a very simple one here that says when you need the data for this column instead of rendering this complex HTML, just render the barcode.

And we will see these also in the console, the Javascript console, where if you have a template and you don't define a matching generator, it will give you a warning.

Another handy widget is the orgfamilyselect. This allows you to specify a chunk of the orgtree. So typically we have library selectors that allow you to pick a library, a specific org unit. This lets you decide on an org unit and then say whether or not you want to include all of the -- and payment org units. This comes in handy in a number of places especially admin interfaces or in grid filtering contexts. Under the covers, this looks a bit like this. Just the main thing I wanted to point out is the model that allows you to track the value is selected in that selector. They are going to tell you the primary org ID which shows up here and whether or not either of those checkboxes are selected, and depending on the situation that either of these, it will include the selected plus an additional org units in disarray here.

So it makes building queries based on a user selection here pretty simple.

Date range selectors, there are multiple date pickers. There is a date picker, this is one of the more interesting ones because it allows you to select date range. You will see this in the looking reservation page. That is at least one place where it shows up.

And you can pick a start date and you can page through months to pick an end date to get the range you need. And then at the bottom you can

see what that model looks like under the covers. Once a value is set, then you will have an object with these two different components in it.
On the Angular side, the access keys, which are the keyboard shortcut commands, the code to do that is home grown. We don't have a third party dependent docket dependents for that. That was something -- that was a decision I made. Given the complexity, there are certain things that can be done in 30, 40 lines of code that we should do in the long run. That's where this came from. And if you are in all the Angular interfaces and you type control H, it will give you this list of keyboard commands that are registered in the interface.
And it will give you the command, the label or the action that is document with the command, for the command is defined, and whether or not it is active. If a command shows no as active, that means it command was registered later on that precedes or supersedes, I should say, and existing command. So it overrides.
And under the covers, it looks like this. Not two different from the Angular JS market that you might see with this. But we have a directive, context which is the navbarcode. Edit have at least two different key commands that will fire this action. Description, all of these are marked as -- so you can't change to work commands and discussion, and the discussion is that i18N. So it can be a click command or any of those would be fired when to in question is fired.
So jumping back here, we have two different check-in entries -- sorry, Patron search. 14 F4, alt S. I can see and are given for collapsing this down to one. This is just how it looks now.
This is probably the most eye-glazing one. But I thought it worth mentioning. Two things on here I wanted to mention, that are relatively new. On the Angular side. First is this thing called flesh selectors in that PCRUD. This is a replacement for manual flesh entries, Edison specifically related to when a class in the audio has a field in the link,

and the destination for that link is another class which has a field described as a selector.
And you have seen this kind of thing in a drop-down. So you want to list your copy statuses so you can pick a status. That label that displays for available or checked out or whatever, those are coming from its being labeled that we want this name field to be called a selector field.
It comes up in a lot of contexts, especially admin interfaces. So instead of how to think about how you want to do the selection, you can grab all the things you might want to display a name for them as opposed to an identifier.
The second part of this is there is a thing called the format service. You can pass all kinds of data and it will figure how to get it and display it. It's good with dates. Dates and times, you can't it's -- locale and times unaware, it has Boolean values and strings in all kinds of stuff. Want to keep -- one of the things he knows that it is if you pass an object that is defined in the IDL, it knows whether or not a certain field likes to a selective -- selected class. And it will look to see if that has a flash version. It does, it will display the selected value. So in this case, the conflict the consult log, meta bib field class, and it will transit the remainder of the value. Typically you would see meta bib field class and an identifier. In this when we are seeing the name of the object.
So anytime you have an IDL object that you want to display a calm, and you want to automate the process or not think about it, this formatter comes in handy.
Another new interface component, relatively new, this was added along with the Angular MARC editor, I keep looking at my face and drifting off camera. This was added the added along with anger Mark editor. It's essentially a list of key value pairs and you take an input and you tell it that you want it to be a context menu and you tell it what to do in an item -- and an item in here is selected.

As far as the entries, your giving it a set of values label pairs. So in this case I have a component or something with an array of things called stuff, and adjusts markers out and the end up in the context menu.
One thing of note on the chunk of code down here is I have these parentheses here. Some of you who are familiar with the arrow notation for fonts, you don't have to have function brackets if your just doing one line of code. But if your one line of code is an object that starts with brackets, you're going to confuse the compiler and us when to expect you to have -- that thinks that it's needing a function one sees this bracket here. You can get around that by putting it in parentheses, and it knows that whatever is inside the parentheses is a single return value. Or you can have a return value but either way works fine.
This, to me, and we have the same thing on the Angular JS side, as one of the more clunky and festering parts of the development process we have this thing called string component, which the reason we have it is because we don't want to just put their strings into code because it we can't translate them into other languages. So we take advantage of the functionality the built in, in Angular or -- and that way we can defined strings in the templates that are translated into templates and the strings are accessed by various mechanisms encode.
But this is a fairly simple example. Your just defining a string, given getting it and I don't fire, and giving it test, and then you can access the text various waves.
Here's a more propagated example. You're adding logic to the template that renders the string value, and then you're telling the string to use specific data at runtime to fill in the template so it can apply the logic based on the individual tab and query value.
So this is a lot to type. Just to say search, :, piano music. That's all this boils down to. I'm just here to save -- to say that there is hope for the future this that it will be a lot easier. As of Angular 9, which we have not migrated to yet, they have introduced -- I don't know if it's a service or

what they are calling it exactly, but there is a way for you can type code -- or excuse me, type strings into the code and tag that is being translatable.
So in the end, we would have all of this would be replaced with something pretty close to this. It would have a variable at the end. So that's extremely exciting. And I can't wait to start using that. There is a caveat, with Angular 9, at the time I was less reading about this, that the code which generates translations is not yet capable of reading the translations from the typescript code, so you have to manually add them to the eye because of the XMB files that will be uploaded to the translation service.
So that might get in the way of using it for Angular 9. But it does seem like there is at least -- this is great progress. That's probably hardest part that we wanted getting to get taken care of. So teaching that energy -- I forgot what it's called, but it's a command that generates the translation. So it is how it’s taught to read the stuff out of the code, and we can get rid of a lot of the stuff.
Something new that was just merged on our way to wrapping up the current batch of releases was a new addition to the IDL called configfield. You see it here in the example. And this is a handy marker in the IDL that says when you are rendering rows of this type of data in a grid, potentially other contexts, but for starters, this is for admin strata grids, then you can render this field as a link to another table which is a relationship to the first class.
So the example that came up in launch pad that was the first example working, was the C 39 source attributes field. So you will have a Z 39 server, LC, et cetera, and the link to that is a set of attributes that you use for searching. And before we had to such analogy, there was no way to jump from one to the other. In fact, there was no weight to get to the secondary interface at all.

So you can go to the source page, and we have a link for the attributes. If you click on the attributes for the source, it's going to add a -- first of all, it's going to take you to the attributes page and is going to filter the results to those that relate specifically to the LC source. I will show you what that looks like.
That's kind of a limited picture, but this would be the attributes grid and the reason is small is I wanted to capture this part along the top here. There are two ways to apply filtering to the grid. There is via the URL, so that's when talking about here, and it shows the filters that are applied via the URL and it gives you an option to clear this filter so you can view all the items.
And that leads me into the general more purpose grid filters. These have been around a little bit longer. But this is something that is turned on per grid, these are in-line filters and they provide different options for -- I should've had a picture of the drop down, sometimes it's hard to do that. Loses focus and it disappears. But it gives you different options for filtering so over here on the string value for example, will let you do start with, greater than, less than, those of things like that. On the hook column, this is a primary key value so all you say is equals or is exactly in effect.
But this functionality is there for grids. I want to point out if you are using this, you have to, as with all the grid data, you have to wire it up and get it working the way you want it to work.
So under the covers, this guy, when you are providing data for the grid, it's going to be an excerpt item on your data source called filters and is going to have one or more filters attached to it based on the name of the column. And there is a filter clause under there. And you essentially add these to the query that you're making to retrieve data from the server. It could be expanded out to other APIs to read this.
Another new thing is server print templates. I don't think people have had a chance to interact with this yet pick there are only two examples in

the matter server right now. One is just an example of how to -- that ports over the Patron access template. Since we are still using the angle JS for that, people haven't read interact with it yet. And the second is a hold for record, and that's the spirit metal staff Angular -- so not a lot of people have had the need to look at these.
But it will become more important as we move more interfaces to Angular, and the general concept, it's fairly similar to an Action Trigger type of thing, though it's focused specifically on printing. You had leased template toolkit things and they have a fairly narrow set of functionality desires they can do. From the client you pass a template data object and it just gets rendered into here, and return to the client is an HTML page of the client renders that either through hatch or -- level printing, that gets sent to the printer.
The main reason I bring it up. Like I said, it will be needed more for the interfaces. But it does allow you to do things like apply a locale to a template. So if you have a staff member logged in at a different locale, or say for example, you know a Patron's default link is different, you can take advantage of that different context.
So these will start showing up more as people set up coding interfaces. And this is the admin UI for managing those.
A topic that has come up quite a few times in recent memories, in launch pad, is the wonderful power that we have with being able to especially broadcast huge numbers of API calls to the server, through the XML stuff or through web sockets. It really kind of encourages you, for the sake of speeding up the interface to parallelize a lot of data collection, send it a lot of stuff off at once.
And for various reasons, this being one of them, we have been seeing certain cases where the server is just getting hit with way too many requests at once. So something that I have started doing is being a little more conscious of this, especially on any type of batch operation or page load operation. So this is something that I might have written some

time ago where I wanted to load four different things simultaneously is a certain pager component was loading. So these are get blasted off at once. The server is hit with four requests, and generally is not that big of a deal. Evergreen does well with parallelized processing. But if you're sending off too many, can be problematic because you are exhausting resources on the back end.
So this is kind of the alternate version of that that I started doing now, as I tend to by default and sure that everything is loaded in cereal. So that one client is making a request, given a response, et cetera. I will do this as my default. And this is a very typical -- most of my components of a load function that looks exactly like this. Of course, we have found scenarios where it really just needs to be a little bit faster, then one option is to do more paralyzed ocean of parallelization. Another option which I started doing more and more of his turning these four calls into a single API call that is custom-built for this interface.
That we benefit both from the speed of doing four things at once, while at the same time only having a single API call to the server.
Some of these things I'm suggesting, it's stuff I have come across, things that work well for me. Most of this, is not the gospel, it's just how I found it to be working well for me at what I think probably makes sense in those cases.
So toward the end, if there any questions, any comments about any of this stuff, and definitely glad to hear it.
Like I said, this is all over the place. It's basically, I take notes when I'm writing code of stuff I can use and I come back and talk about it.
As of Angular 8, you find it is hard to merge changes between Angular 8 and 7. Because there is a new required field on this thing called view child in the Angular components. The child is a way to get references to child components on the page. So just to kind of demystify the -- with Angular 8, the flag called static can be true or false. And if the value is set to default, that means that this my component variable is not

available until the after view hook. After view in it hook. Angular has various what they call lifecycle hooks so that you know of a certain point in Page rendering what is happened or not happened yet.
So after view in it is a little later on of the process and more the page component has been fleshed out at this point.
If the -- was set to true, then you have access to this variable in the oninit function.
My default going forward is meant to use false and that is based on the fact that as of Angular version 9, that will be the assumed default, which means that we won't have to have these in the code anymore and the migration tool well do them for us.
And the other part is, they basically say you only really need this under kind of atypical circumstances.
So I generally just move everything to false going forward. And then being aware of when this variable can be accessed.
And related to that, when we have child components like this, I will have a component within a page within identifier loading process of loading progress, you do have to be careful, regardless of all of this, when you do this -- my component or in this case loading progress. Because depending on how you have defined whether or not this component even exists, if it doesn't exist, then that is going to throw an exception. Because it comes into it out of existence based on its existence in the page.
So an example of why it wouldn't exist in the page, is if you are wrapped in an NGF block, and this variable is set to false, this would not exist. And if you tried to use it you get an exception.
As a comparison to that, if you did something like a hidden block, which does not prevent us from existing, it's a plea prevented from displaying, then you can reference this thing any time after the after view in it. Because it will disappear until has been put into existence. After one caveat about that, if this is a giant complicated component, then you

don't want to just hide. Because of it is hidden, it is still occupying resources. So that is a case where you want to prevent the component from existing all and do more of an that -- ngf thing. But this, I think you can get away with without too much trouble.
This is different from Angular JS. I will show you an example in just a second. The general idea is if a component exists in Angular and there is no reason for it to go away, while performing actions like changing your route or other variable data, then the component is not going to be torn down and rebuilt and reinitialized every substantiated and all that stuff for speed reasons going to stay exactly as it was. So if you want these sort of child components to respond to external stimuli, like input variable change, you can teach it to do that. I was want to show an example of that real quick.
This is the Angular catalog. And it's a search result set and I'm on the title detail page. We have two main components here. We have book summary along the top, and the tab that is currently being displayed, just the mark MARC HTML tab.
So this component was taught to respond to rat changes. Probably not easy to make out over the interface pick but here you can see there is a bib record ID in the route. So if I click next, it change, and the HTML changed.
And the reason changed as it was taught to watch for changes in the route and it re-renders itself using appropriate identifiers. The same kind of thing happens up here, but this is more of a general-purpose component that doesn't know anything about the route. So it knows to watch for changes to its input variables into re-render itself as needed.
So the way that looks, code, is the two different types, again, route level changes, the first one here, this is a difficult thing that I have a lot of interfaces that I make. You will subscribe to the activated route, parameter map, this is an observable that fires every time anything up and that URL changed. Anything I care about is going to show up in this

subscription. And it does fire once on Page load. So you don't have to do this logic once and do it again inside of here. You can do it all inside of the subscription. Basically I'm saying, what record ID does not appear in this URL, is it different from the one I already loaded? Let's go ahead and load it. We are going to reinitialize ourselves.
And then going back to the alternate form, input changes, that would be kind of the record summary here. It has an input and instead of being a variable, it's a set function and a get function. The set function is watching for changes, of course, and it does have similar logic if the ID that was provided is not the one I'm already checking, that I will reload myself. One caveat here is a you don't want to do this before in it has run. Because typically that's when you do the first load of data. So you don't want to do this before that and then read it again because were justification effort. So have taken -- tracking whether or not this code has completed and only after that's done will I ever reload based on a change of input. You have to be careful. These will get undefined data. The photos are called at every level of the process. So if you regular ID is passed, you're going to get Noel and that he find here. So you have to be conscious of what is coming in.
I mention this only because it's going back to those themes and patterns that I see over and over. The introduction of observables is very powerful and use it for all the network stuff. Process streams of data unlike standard web promises, which are basically a true/false type of thing. They can return different things, but it is one or the other response. Success are not success. Where the observables give you results, success, or possibly an error.
Often you will see things for you have to process a stream of information, but then the color just wants a promise, yes or no, that it's done. You could do that with observables, but in some cases the library would be expecting across.

So this is a common workflow I use a lot in my code for a want to process one thing at a time because I'm expecting a large list of them because I don't want to grab them all in one chunk. I want to get a string of them. So is that string comes in, I can pass it off to some function using this tab function which is part of the observable API. So you are generating an observable, paving the observable, and saying I want to look what's in there. I want to do anything to it, just like don't look at what is in there. And then turn the whole thing into a promise and you can be compliant with the code that wants a promise instead of an observable. You'll see this quite a bit. Other times, you will see this mixed in with merging and merge mapping. So you're just taking multiple observables and coalescing them into one.

So this is just some suggestions that I have for development that I wrote down, the company for nearly now and then. So I thought they might be helpful.

First and foremost I always recommend that you symlink into your -- every time you hit save in the browser you just apply the updates. You don't do any think else. That said, toggle over to your browser, hit refresh. In the code changes. That's a lot easier than copying stuff around were certainly not doing and install. You can encode really quickly with this setup.

For those of you who have done work on the Angular side, you may have been greeted by this page and expected we sometimes or even better, this page. There is a certain level in the Angular rotting process where if it bombs out, then exception is thrown, it's unable to completely rotting, and it will send you back to the base page. Or in some cases it won't even do that. It will just throw up his hands and quit.

I always bookmark the URL I am looking that working on, so if this happened to get back to it instantly. Because when this happens you can't just click the back arrow to get back to where you want to go, because essentially won't lock to that because it knows it's broken even

if you reload, it's going to go back to the less successful route. It's a little bit annoying when this first happens, but if you can get through quickly, it's not too troublesome.
For those of you using vim as a text editor, answer their other similar options for other text editors. I recommend adding this to your vim RC file. Trends white space from script files. This is really handy for running ng lint. One thing that our lint configures and says as we don't want to know point spaces with the code. So instead of having to take us out by hand, you can set this up and it will clean it up as you go. It is very handy.
And the prod versus death builds, they archly build differently. The prod builds can do much more deep compilation. In particular they dig into the HTML. So if you do a prod built, you can do a lot more about problem the problems in the code. It will show you where you have a function defined in the text script and you called it in the code with a different number of parameters. And prod build will tell you, you are calling function correctly. And it will do lots of other things, too. So is like a secondary there a checking tool. And that's something that should always be done before or after merging file code.
And lastly, back in the early days of Evergreen, sometimes we would do programming to kind of learn how each other operated. I learned a lot from those guys and I decided to try to continue that tradition. So this is my pair of programmer for now. He's really good at telling me when I'm doing things the right way. His name is Booger. And he will let you know what he thinks.
So I can open the room for questions. That was a lot, I know. It was an onslaught of coder information.
Booger is a Boston terrier. Regarding the cases where you want to use view child, status and cost true, haven't found it yet. They mention any documentation -- let me find that again. Here's the URL. I will just pull it up. Is there a case where I should use static true? I will let you read that,

it had to do with the situation I haven't really encountered. Yes. The dog has amazing under bike. Crooked teeth. Yes. Pulling from Jason regarding the symlink trick, you have to be careful with that and installs, because I think it will choke when it -- that's an excellent point. My sort of local quick and solid script means that out of the way. And it puts it back into place.
The slides are fairly sparse, but I do have a link to them. That includes the slides themselves and the compile HTML, take single file you can load up in the browser. And just for experimentation, say I did this for markdown, I'd never used it before, so the bare text might look a little different.
Okay. It's fairly quiet over there. Someone just said my name. Well, looks like we’re doing good here. We are getting close to time, anyway. I believe we have another virtual happy hour at 15 past the hour. So if there aren't any more questions, I guess I will let Amy wrap this up.
>> There was one last one.
>> I missed that. There we go. How do you like Typescript? I absolutely love it. It's a pleasure to work with. Chris mentions Visual Studio code as a good editor. I have used that as well, and is -- it is really handy because it will give you typing suggestions and other kinds of suggestions. That are really useful. It will warn you if your importing stuff you don't need or referring to things that don't make any sense, all in real time. It's really cool.
I believe the sessions will be sent to the mailing list. Possibly all the attendees. I'm not sure, once they have been sorted through and edited. A lot of love for VS code.
>> Okay. Is that it? Okay. Thank you, Bill. I like your style, dude. (Laughter)
>> Thank you all. I'm glad you all could make it, and it's great to see you all virtually.

>> And that is the end of the sessions for the day. Bill, back to back, I had no idea. Good for you.
>> Fortunately, the first one was a panel. So not too much.
>> Okay. There is going to be a happy hour if I don't experience any diff dock taken the difficulties on my end. It's limited to 100. I have no idea, we might get 10, we might get 100. I hope to see people there.
>> Well that be on this channel?
>> I wish it had been this channel. If I had had some good foresight I would have kept it with this channel. But I didn't. I created a separate one.
>> You have about 25 minutes to do some pre-gaming.
>> Deathly think. I should ask that. If people want to just do some kind of games, if we are going to create breakout rooms. There are themed breakout rooms, but they aren't happening. I'm going to keep it loose.
>> So people are chatting at the end of the day.
>> The pre-gaming has already started. Everyone has gone loopy. Me included.
>> I'm going to turn off my video and take a break. Thanks, everyone.
>> I will say goodbye to anybody. Goodbye.