PRAGMATIC WORKS

Power Apps Cheat Sheet

Power FX Common Expressions to Know First

PROBLEM	EXPRESSION

Show the Current Logged in User	**The User Function Examples**
	`User().Email - Shows email address`
	`User().FullName - Shows user's name`
	`User().Image - Shows user's photo`

Filter the Rows to a Certain Criteria	**The Filter expression can filter the rows based on an expression of your choosing.**
	`Filter(DataSource, Expresssion)`
	`Filter( DataSourceName, Column > 0 )Filter( DataSourceName, Column > 0, Column2 = "Value" )`

Lookup a Value in Another Table or Source	**The lookup function can see the value from a different source based on an expression.**
	`Lookup(Source, Condition, ValueToShow)`
	`Lookup(DataSourceName,CustomerID = 5,CustomerName)`

Create a Global Variable	**Global variables can be seen on any screen. They can be created when you start an app by selecting the App OnStart event or on any actionable event.** `Set(VariableName, value)`

Navigate to a New Screen	**Navigation can be done two ways: either go back to the previous screen you came from or navigate to a specific screen..**
	`Navigate(ScreenName)`
	Optionally, you can specify an effect during the transition:
	`Navigate(ScreenName, ScreenTransition.Fade)`

Power Apps Cheat Sheet

Power FX Common Expressions to Know First

PROBLEM	EXPRESSION
Send Data From a Form to the Source	**A few ways to send data from a form are SubmitForm or the Patch commands. The SubmitForm is easiest with he smallest amount of flexibility.** `SubmitForm(FormName)`

Searching Multiple Columns for a Row	**The Search command does a contains search across multiple columns and is usually used in galleries or data tables.**
	`Search(DataSource, SearchString, Column1 [, Column2, ... ] )`
	`Search(DataSource, txtInput.Text, "FirstName", "LastName")`

Updating or Inserting Data Without a Form	**The Patch function is one way to update or insert data without using a form. For example, you can tie it to an OnChange even of a drop down box.**
	For inserting a record:
	`Patch(DataSource, Defaults(DataSource), {ColumnName:"Value1", ColumnName2:Number2})`
	Similarly, you can update data when you select an item in a gallery:
	`Patch(DataSource, ThisItem, {ColumnName:”Value1”, ColumnName2:Number2})`

<table>
<tr><th>PROBLEM</th><th>EXPRESSION</th></tr>
<tr><td>SharePoint Variations With a Person Picker</td><td><b>Lookups and the Person Picker in Sharepoint require some variations to your code in Power Apps since it’s a packed field, meaning that the one field has a bunch of items inside of it with references. This gives you the ability to navigate the hierarchy easier in Power Apps.</b>
<b>For filtering based on a Person Picker based on the person who’s signed in:</b>
<code>//Put this code in the App OnStart</code>
<code>Set(varUserEmail, User().Email)</code>
<code>//Gallery code or wherever you want to filter</code>
<code>Filter('Assembly Work Orders', Owner.Email=varUserEmail)</code>
<b>Patching is especially challenging into a Person Picker and a choice column. Both are shown below showing you how to unpack the column before inserting into your list. The below is showing inserting with Patch into a SharePoint list with a choice column called Status and a Person Picker column called Owner. A title column is required also for SharePoint.</b>
<code>Patch(‘SharePoint List Name’, Defaults(‘SharePoint List Name’), {Title:”Your record title here”, Status: {‘@odata.type’:”#Microsoft.Azure.Connectors.SharePoint.SPListExpandedReference”, Value:”Complete”}, Owner: { ‘@odata.type’:</code>
<code>“#Microsoft.Azure.Connectors.SharePoint.SPListExpandedUser”, Department: “”, Claims: “i:0#.f|membership|” &amp; User().Email, DisplayName: User().FullName, Email: User().Email, JobTitle: “”, Picture: “” } })</code></td></tr>
</table>

<table>
<tr><th>PROBLEM</th><th>EXPRESSION</th></tr>
<tr><td>SharePoint Variations With a Person Picker (Cont.)</td><td><b>Filtering on a choice column requires that you specify the value of the choice as shown below with a drop-down filter.</b>
Filter('SharePoint List Name',
ChoiceColumn.Value=Dropdown1.SelectedText.
Value)

<b>Defaulting a drop-down box in a form to your name in a person picker column</b>
Set the DefaultSelectedItems property to the following if your form is called Form1 and your Person Picker column is called CreatedBy. Note the code is going to check to see if the form is in New mode and do this or if it’s in edit mode, it will keep its existing value from the list.
If(Form1.Mode = New,{
'@odata.type':"#Microsoft.Azure.Connectors.SharePoint.
SPListExpandedUser",
Claims:Concatenate("i:0#.f|membership|",User().Email),
DisplayName:User().FullName,
Email:User().Email}, ThisItem.CreatedBy)</td></tr>
</table>

PROBLEM	EXPRESSION

SharePoint Variations With a Person Picker (Cont.)	**Filtering on a choice column requires that you specify the value of the choice as shown below with a drop-down filter.**
	`Filter('SharePoint List Name',`
	`ChoiceColumn.Value=Dropdown1.SelectedText.`
	`Value)`
	Defaulting a drop-down box in a form to your name in a person picker column
	Set the DefaultSelectedItems property to the following if your form is called Form1 and your Person Picker column is called CreatedBy. Note the code is going to check to see if the form is in New mode and do this or if it's in edit mode, it will keep its existing value from the list.
	`If(Form1.Mode = New,{`
	`'@odata.type':"#Microsoft.Azure.Connectors.SharePoint.`
	`SPListExpandedUser",`
	`Claims:Concatenate("i:0#.f|membership|",User().Email),`
	`DisplayName:User().FullName,`
	`Email:User().Email}, ThisItem.CreatedBy)`

PROBLEM	EXPRESSION

Get the First Day of the Month	**Some systems will require you insert the first day of the month vs today's date.**
	`Date(Year(Today()), Month(Today()), 1)`
	Would return 3/1/2021 if it were 3/16/2021. Replace Today() with your own date picker if you want the customer to pick the date instead of always getting today's date as shown below.
	`Date(Year(DatePicker1.SelectedDate),`
	`Month(DatePicker1.SelectedDate), 1)`
	
	For the last day of the month, you can use the With statement to construct your logic. The below code finds the first day of the next month then subtracts one day from that, giving you the last day of the previous month. With the With statement, you essentially, create a variable that's only in scope of this one command. In my case that scope is only in the scope of a text label.
	`With({DateSelected:Date(Year(DatePicker1.SelectedDate),`
	`Month(DatePicker1.SelectedDate), 1)},`
	`DateAdd(DateAdd(DateSelected, 1, Months),`
	`-1, Days))`