Practice using functions in Power Apps

You may already be familiar with functions and formulas in Excel. In Excel, you use a formula to return a result or value to a specific cell. You can use similar calculations in Power Apps and as an Excel spreadsheet recalculation happens automatically

As you have already seen, when creating a canvas app, you can add a variety of user interface (UI) elements so that users can interact with, and respond to, the data presented. These UI elements are called controls. You can then use formulas to customize the control or to set the app to provide different responses depending on what the user has selected. For example, you could use a **Filter** function calculation on a control to ensure that Users only see rows in the data that are relevant to their geographical area.

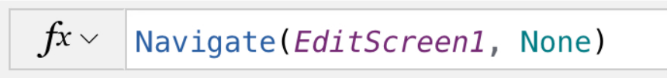
In the previous exercise, you have already used a function called **Navigate** to make the app move to the feedback screen and another operator called **ThisItem** to populate a control with entries from a specific column. You entered those formulas in the formula bar at the top of the screen.

There are some key differences between formulas created in Excel.

- In a canvas app, you do not need an equals sign at the beginning of the formula
- When typing in function formulas in an app, you do need to pay attention to capital letters. For example, if you were to type EditScreen1 without a capital E and S, the app might not recognize the screen name. Similarly, if you keyed in ThisItem without the capitals, again the app might not recognize it.

Syntax

Let's examine the syntax used more closely.



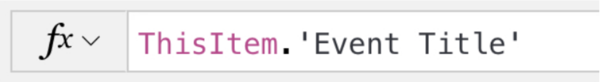
```
Navigate(EditScreen1, None)
```

In the example above, **Navigate** is a function name. A function requires an opening and a closing parenthesis. Inside the parentheses are arguments, which are the pieces of information the function needs complete its task. The arguments are separated from each other by a comma.

You can combine functions to expand their functionality. One of the arguments for one function could be another function generating a result. In that situation, each function will have its own opening and closing parentheses and arguments. Some functions generate results or values and others might generate actions or other effects.

The app will provide information to help you as you key in a formula. In the previous exercise, when you typed in the Navigate formula to customize a button, you would have seen the formula bar help by showing the description and the expected arguments. Also, in a formula, different syntax elements appear in different colors. This is an important visual cue since different types of elements require different syntax and have different expected behaviors.

For example, in the formula to customize a label box, the term **ThisItem** is in red rather than blue.



```
ThisItem.'Event Title'
```

ThisItem is a **Named Operator**. **Named operators** provide access to information from within a container. In the above example, ThisItem provides the name of the event from the called Event Title in the Excel source.

Another element that can be used in a formula is a **Signal**. **Signals** return information about the environment independent of how the user may be interacting with the app. For example, the **Location** used in a formula returns the device's current GPS coordinates. Unlike functions, **Signals** don't have parameters or side effects.

Enumerations return a pre-defined constant value. For example, **Color** is an enumeration that has pre-defined values for **Color.Red**,**Color.Blue**, and so forth.

Changes Based on Input

In Excel, a cell can hold the formula =A1+A2 which means that it will always show the total of whatever values are entered in those two cells.

An app does not contain cells like an Excel spreadsheet, but you can achieve a similar effect in Power Apps by creating two TextInput controls on the canvas to hold the values the user will input. If you also create a separate label control, you can use the formula syntax:

TextInput1+TextInput2

In the FX area. The label box would then display a constantly updated total for any values that the users input in the two TextInput boxes.

Conditional formatting is also possible by using formulas. For example, in the label box mentioned earlier, an IF formula combined with the Value function can check the number that it contains and customize the color if it is below a certain value.

If(value(Label1.Text)<5,Red,Black)

An **IF** function here operates exactly as it does in Excel. It requires three arguments, the logical test, what to do if the test is true and what to do if the test is false. In the example, the logical test takes the number returned by the value function from **Label1.Text** box and then checks to see if it is less than 5. If it is, the number will appear in red. If it is not less than 5, the number will appear in black.

And finally, some shortcuts that you might be familiar with from Excel are possible in-app formulas also. Power Apps contains both a Concatenate and a Concat function, but you can use the **&** symbol as a quick alternative. The formula:

ThisItem.Speaker&” “&ThisItem.Title

would take whatever is in the column **Speaker** in the data source and then combine it with the first space and then with whatever is in the column **Title**. Surrounding the space with double quotes tells the app that it is a text character.

Mark as completed

Like Dislike Report an issue