Beginners Guide To PowerApps

Freelancers: Dries V, Laura GB

C365 MICROJOBS

Contents

Introduction

Citizen development, PowerApps, what is all the fuss about?

Citizen development is the hottest term at the moment in the Gartner dictionary, but what does it mean and what is a citizen developer? Gartner describes a citizen developer as the following :

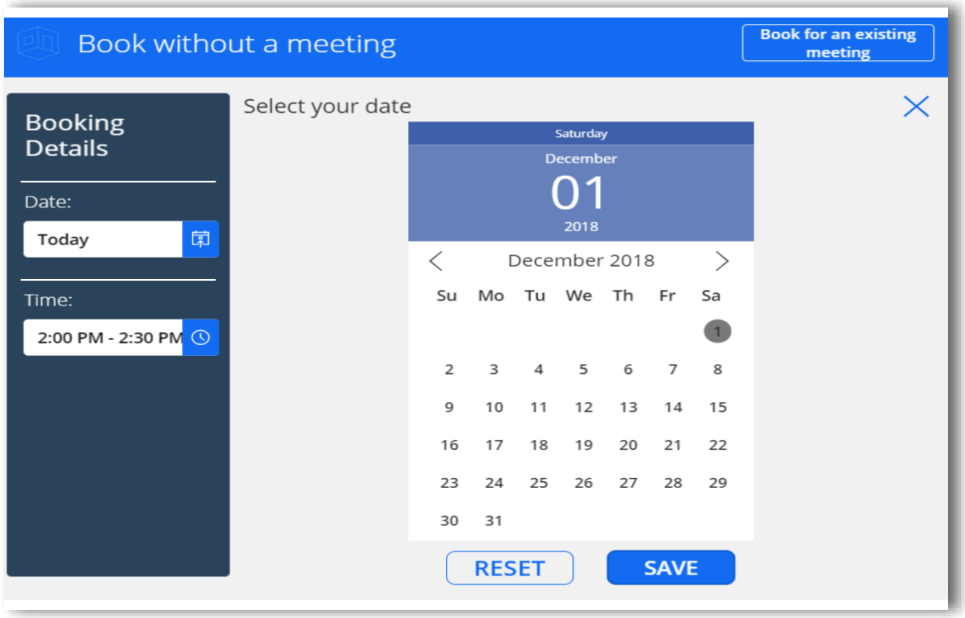
"A citizen developer is a user who creates new business applications for consumption by others using development and runtime environments sanctioned by corporate IT."

In short, this means that "citizen development" refers to non-developers who have the skills to create a business applications, tools or processes.

Is citizen development a good thing?

Yes. Who else other than the business owner himself, knows enough about the requirements to build the business app he/she wants to use best? No longer thinking about: who I should hire to make my PowerApp, how much will it cost or whether the developer will understand our needs.

To help get us started, here's an example of an app in PowerApps:



Getting Started with PowerApps

In the past business users who wanted to create an application to ease their daily tasks didn't have the tools and ended up creating complex Excel spreadsheets with macro's, advanced formulas, and even sometimes some low-end code. These spreadsheets were often then passed on to the IT department to be maintained and supported going forwards.

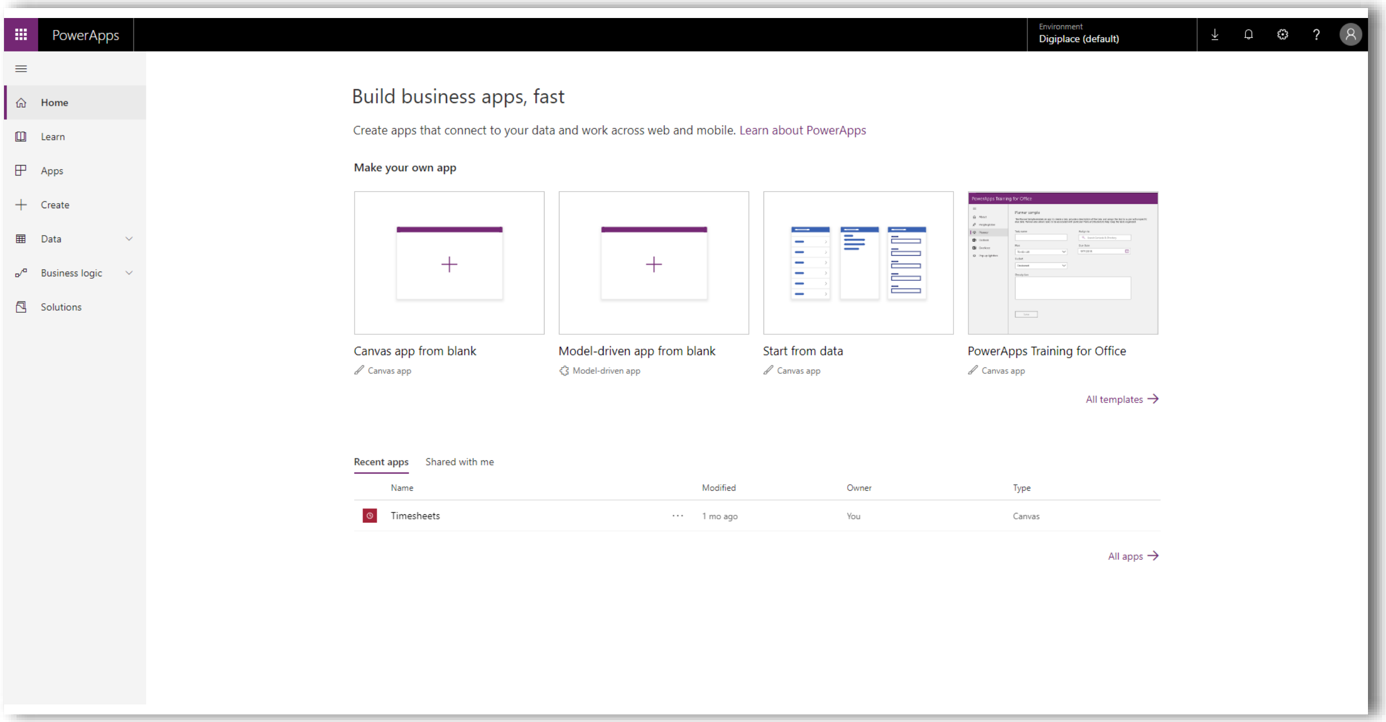
Modern companies want to have a grip on the tools their employees use and are happy that citizen development is now a 'thing'.

Because of this reason, power users of PowerApps offer a much greater value to their company. PowerApps is part of Office 365 and allows power users to create business applications in as little as a couple of hours. Don't waste any time and use this article to start learning the basics of PowerApps.

How do I start developing a PowerApp?

PowerApps development can either be completed in PowerApps Studio, which is a downloadable client application, or directly inside your browser.

Previously, PowerApps Studio had much more functionality, however, at the moment the browser has caught up and has virtually the same features. The browser version has also become much faster to use than ever before.

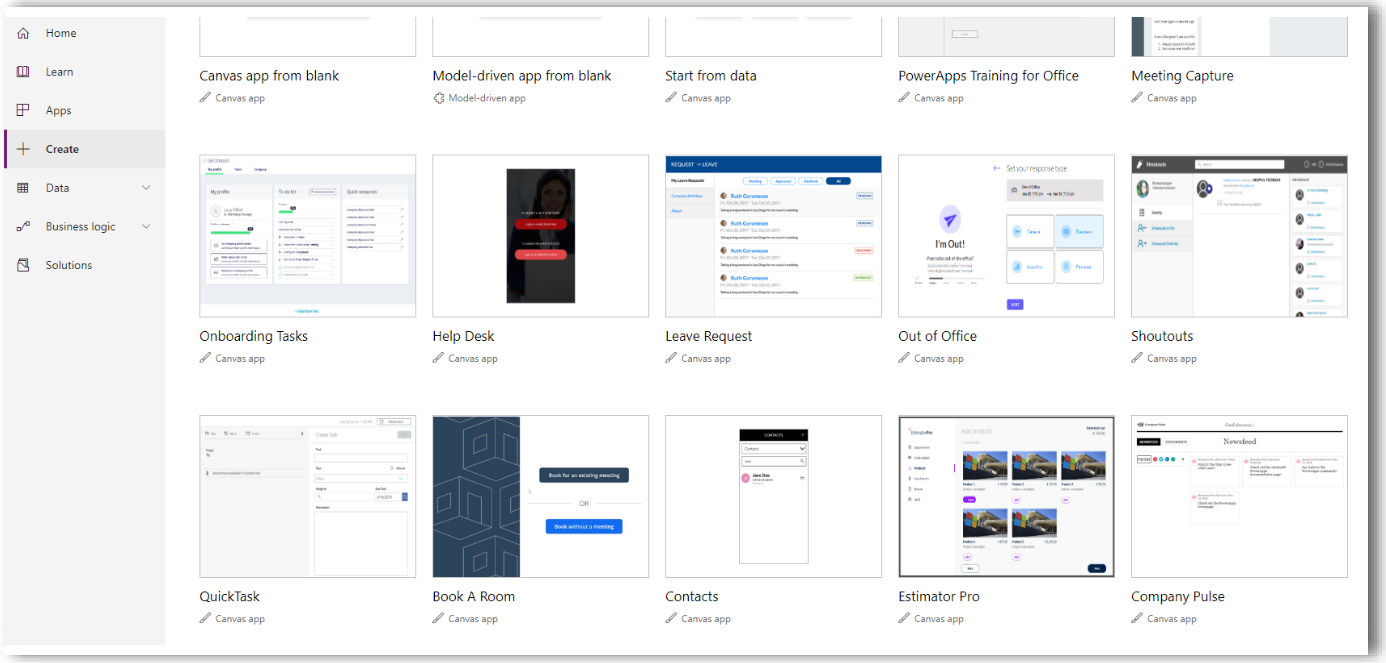


	Name	Modified	Owner	Type
	Timesheets	1 mo ago	You	Canvas

Start with a template ...

My recommendation, if you're just starting, is to create your first PowerApp from one of the templates included in the product.

There is no need to worry for hours about how your PowerApp is going to look. Just click on the "create" button (on the PowerApps start screen) and pick a template to use in your business environment. Even if there is no need for you to use this business application, it can be an excellent way to learn about what's possible and share ideas with your team.



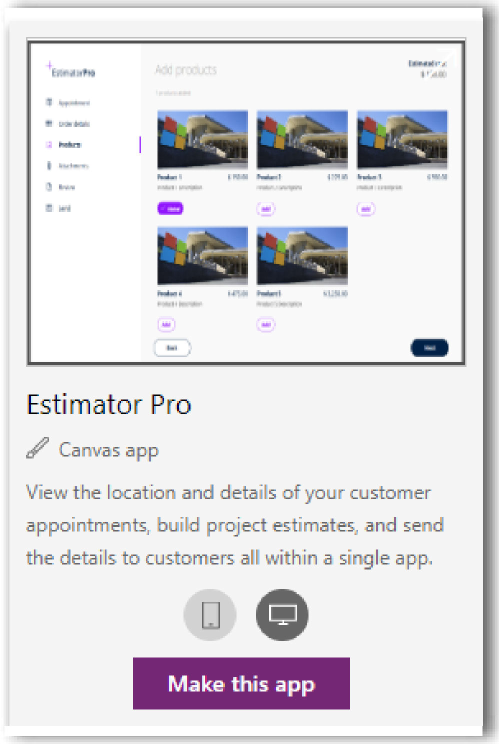
Branding and building for mobile devices

You will notice that some applications in the templates section have multiple design modes. In the example of the Estimator Pro PowerApp, there are two design modes: the phone factor and tablet factor.

A phone factor doesn't mean your PowerApp cannot be displayed on a PC or tablet; it means it will work optimally for a phone (having a small rectangular design).

It is up to the PowerApp creator to choose the design mode, the orientation of the PowerApp (landscape or portrait) and also consider whether the aspect ratio and orientation should be locked.

I am sure you will agree, this is a great start! In just a few clicks it gives you, the creator, a working PowerApp. However, keep in mind that this is only the start of your PowerApp creation. The branding and responsiveness factors are up to you.



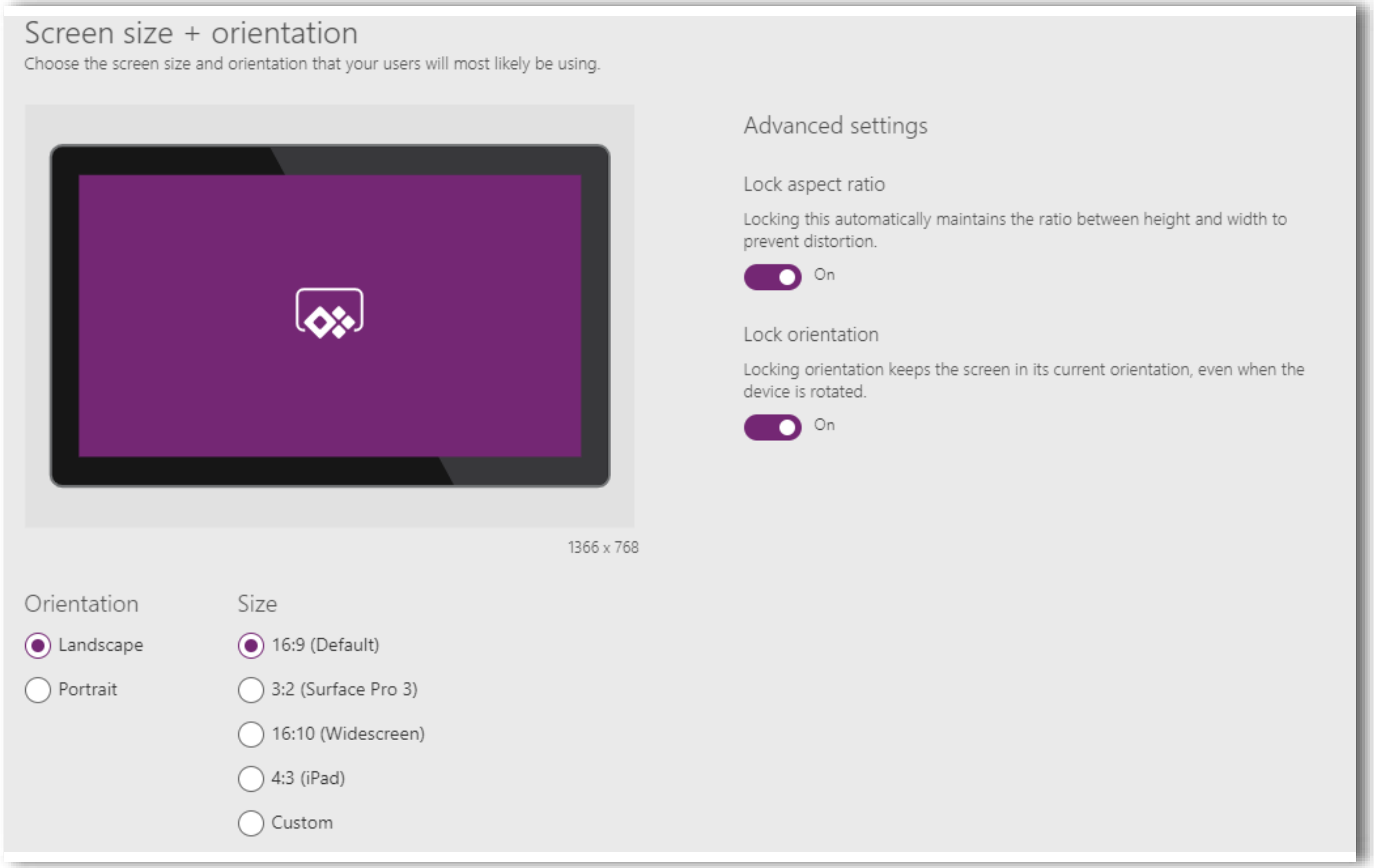
Estimator Pro

Canvas app

View the location and details of your customer appointments, build project estimates, and send the details to customers all within a single app.

Next to the "orientation" option, the user can also choose the required "size" of the target device.

Branding and Mobile cont ...



Screen size + orientation

Choose the screen size and orientation that your users will most likely be using.

1366 x 768

Advanced settings

Lock aspect ratio

Locking this automatically maintains the ratio between height and width to prevent distortion.

On

Lock orientation

Locking orientation keeps the screen in its current orientation, even when the device is rotated.

On

Orientation

[✓ Landscape]

Portrait

Size

[✓ 16:9 (Default)]

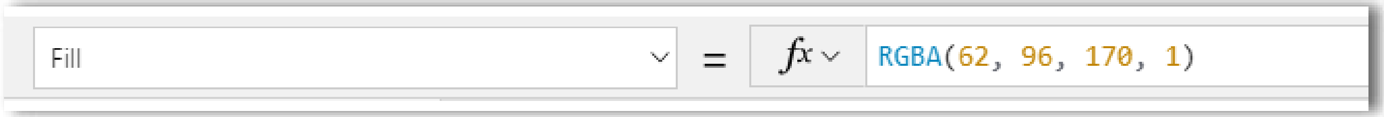
3:2 (Surface Pro 3)

16:10 (Widescreen)

4:3 (iPad)

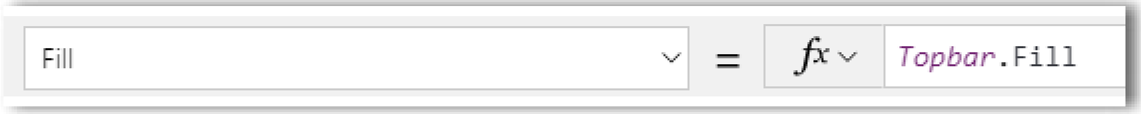
Custom

When branding your PowerApps, I recommend that you have a consistent brand and colour. Be sure to set your colour only once (for example on the top bar of the first screen). On the other screens, don't copy this colour. You should copy the attribute ("fill") from the first screen's top bar. Using one value for your colour is how you maintain consistent branding throughout your PowerApp.



```
Fill = fx RGBA(62, 96, 170, 1)
```

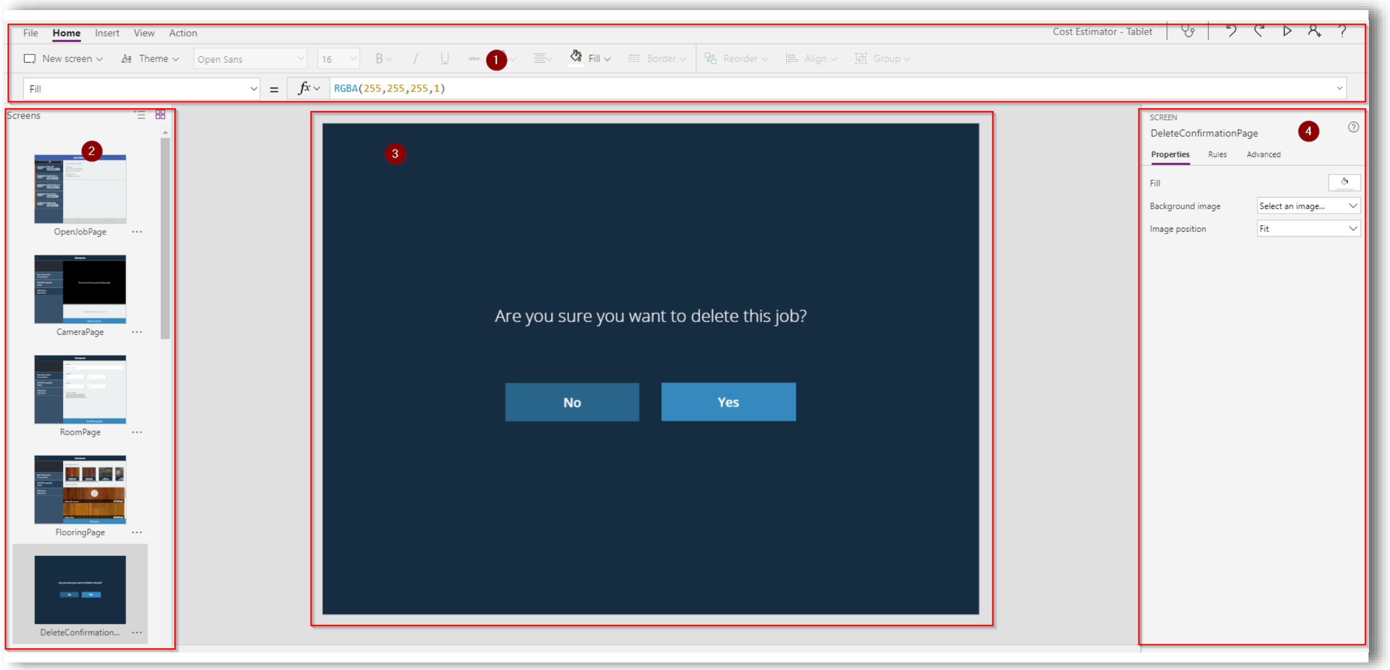
In the screenshot above, I have set my fill to a colour, then referenced it as shown below:



```
Fill = fx Topbar.Fill
```

A tour of the PowerApps user interface

The PowerApps interface looks as follows, where each section has its purposes.



The first section (*top*) is the simplest to describe because this is the most familiar one: the ribbon. Microsoft introduced it in Office 2007, and since then it has never left. Now it is also available in the PowerApps designer and allows you to achieve the same functionality that you are used to in Excel or Word. A few examples of what you can do are underlining text, aligning content and setting formulas on your objects (in comparison to formulas on cells in Excel).

If you are used to Excel, there will be a mind-shift you will have to go through when creating mobile applications instead of creating excel worksheets. The most significant difference is that you will have to start thinking in "screens". These will be shown in the second section (bottom-left). The first screen in this section will be your start screen with subsequent screens displayed in order below. To change the order of these screens, simply drag them up or down to your preferred location, think of it being similar to the slide sorter in PowerPoint.

In the third section (*bottom-middle*), your PowerApp screens are displayed. You can use this section to select your controls (and afterwards setting formulas on these) or to simply drag objects to another location.

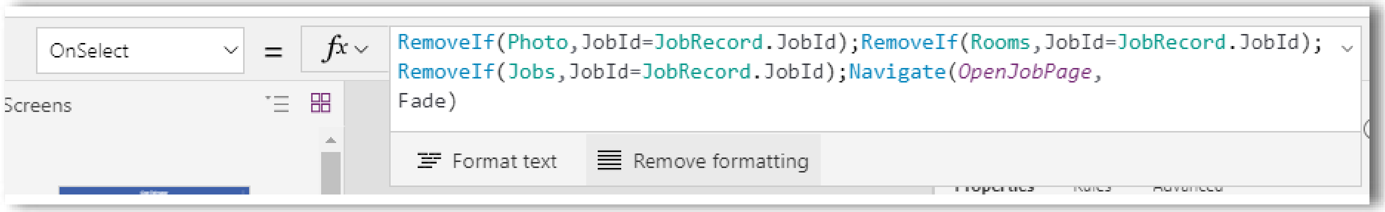
In the fourth section (*bottom-right*), you can define the attributes for the selected control, like in the formula bar. You will see there is quite some overlap between these two sections. Some attributes can only be selected in this section though, for example connecting to your data and picking the correct data layout template.

> When holding the alt key, your PowerApp is not in developer mode anymore, but in run-mode. Allowing you to test the app without the need of pressing the play button at the top right.
>
> — **PowerApps Tip**

Formulas and Attributes

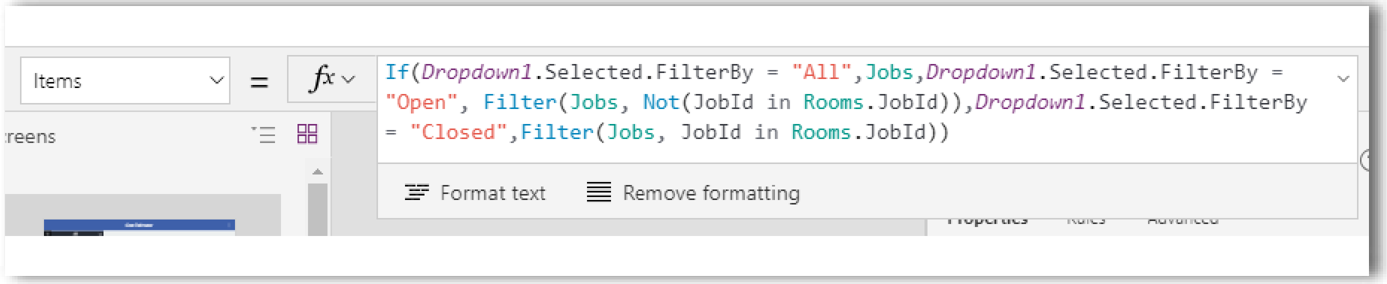
PowerApps contain many formulas and attributes. Formulas are always coupled to an attribute; attributes are dependent on the object you have selected. For example, a screen will have attributes such as: BackgroundImage, Fill, ImagePosition, LoadingSpinner and LoadingSpinnerColor allowing you to set some design preference. It will also have some start handlers like: OnHidden, OnStart and OnVisible allowing you to execute some actions like refreshing data sources etc.

Other objects will have different attributes like OnSelect where you can define what should happen when you click the object.



```
RemoveIf(Photo,JobId=JobRecord.JobId);RemoveIf(Rooms,JobId=JobRecord.JobId);
RemoveIf(Jobs,JobId=JobRecord.JobId);Navigate(OpenJobPage,
Fade)
```

Objects that allow you to show data will have the attribute Items where you can define your correct data source.



```
If(Dropdown1.Selected.FilterBy = "All",Jobs,Dropdown1.Selected.FilterBy =
"Open", Filter(Jobs, Not(JobId in Rooms.JobId)),Dropdown1.Selected.FilterBy
= "Closed",Filter(Jobs, JobId in Rooms.JobId))
```

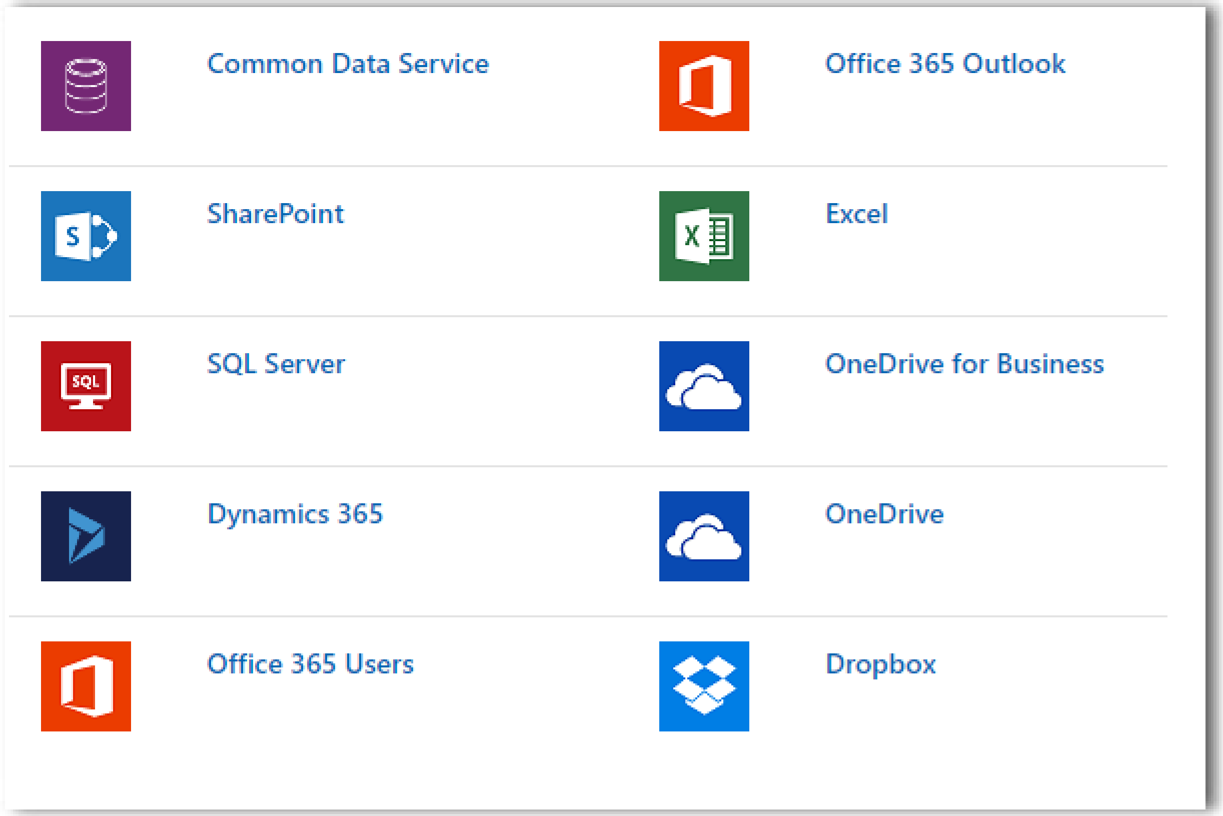
Top 10 Formulas

There are around 150 formulas that you can use with PowerApps, details of which can all be found on Microsoft's documentation site. I asked our PowerApps group on Facebook which ones they used most frequently and listed them below.

Formula	Description
If	Determines if an expression evaluates to true. If it is then a specified value is used, otherwise a default value is returned.
SortByColumns	Allows you to sort a table by one or more columns.
Sort	Sorts a table based on a given formula and sort order.
SubmitForm	Saves the contents for a form to the underlying data source.
Filter	Allows you to filter a set of records based on a given formula.
Search	Allows you to search for a set of records based on a given search query.
UpdateContext	Allows you to store any useful value in a context variable. Scoped to the PowerApps Screen.
Set	Similar to UpdateContext only this time the variables stored are globally scoped.
Lookup	Finds the first row in a table matching a specified formula. Returns a single record.
ClearCollect	Clears all records from a collection and then adds a different set of records to the same collection.
UpdateIf	Update a value if a condition is true.

Connecting to external data

PowerApps has great support to connect to data from other systems. There are already more than 180 connectors available. Examples of the most common ones are shown in the screenshot below:



Connecting to on-premises data

It's also possible to leverage data that's stored in your on-premises data stores. This is achieved by using a gateway. To setup a gateway you will need to use a data source from this list:

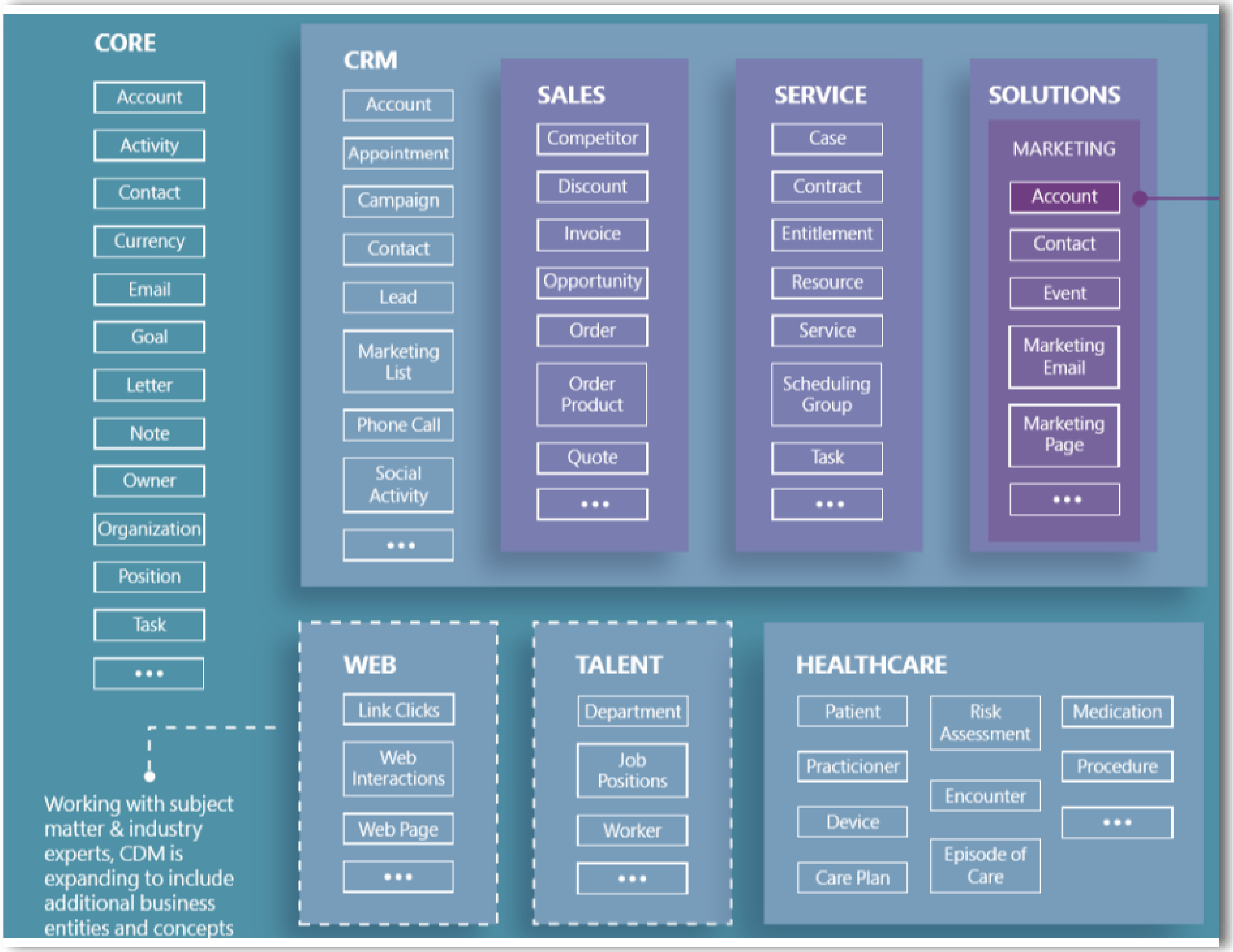
SQL Server	SharePoint	Oracle
Informix	Filesystem	DB2

The process for configuring and managing your gateway can be found in this article.

Connecting to data in the Common Data Service

Microsoft PowerApps can utilise data via a service known as the Common Data Service. In Microsoft's words - "The Common Data Service (CDS) for Apps lets you securely store and manage data that's used by business applications. Data within CDS for Apps is stored within a set of entities. An entity is a set of records used to store data, similar to how a table stores data within a database. "

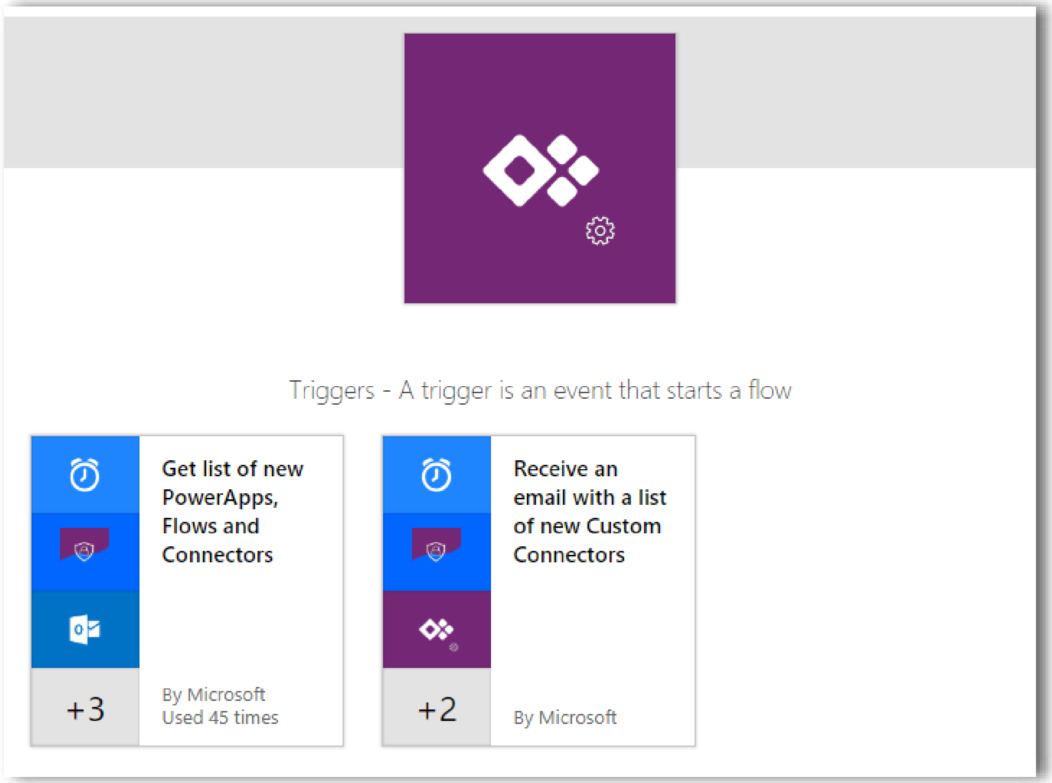
The poster below gives an overview of which entities currently exist within the common data model.



Do more in PowerApps with Microsoft Flow

PowerApps allows you to create mobile applications easily and quickly. Along with your mobile application, you will probably also need some automation to be done in the background, like for example, sending e-mails when a user clicks on a reservation button. Simple tasks like this can easily be done in PowerApps, however when more advanced logic is needed, Microsoft Flow offers more flexibility to handle this.

PowerApps integrates very well with Flow, and they make their connection via the PowerApps and Flow buttons, each one allowing the other to be started. You can even use MS Flow to handle some PowerApps maintenance and governance.

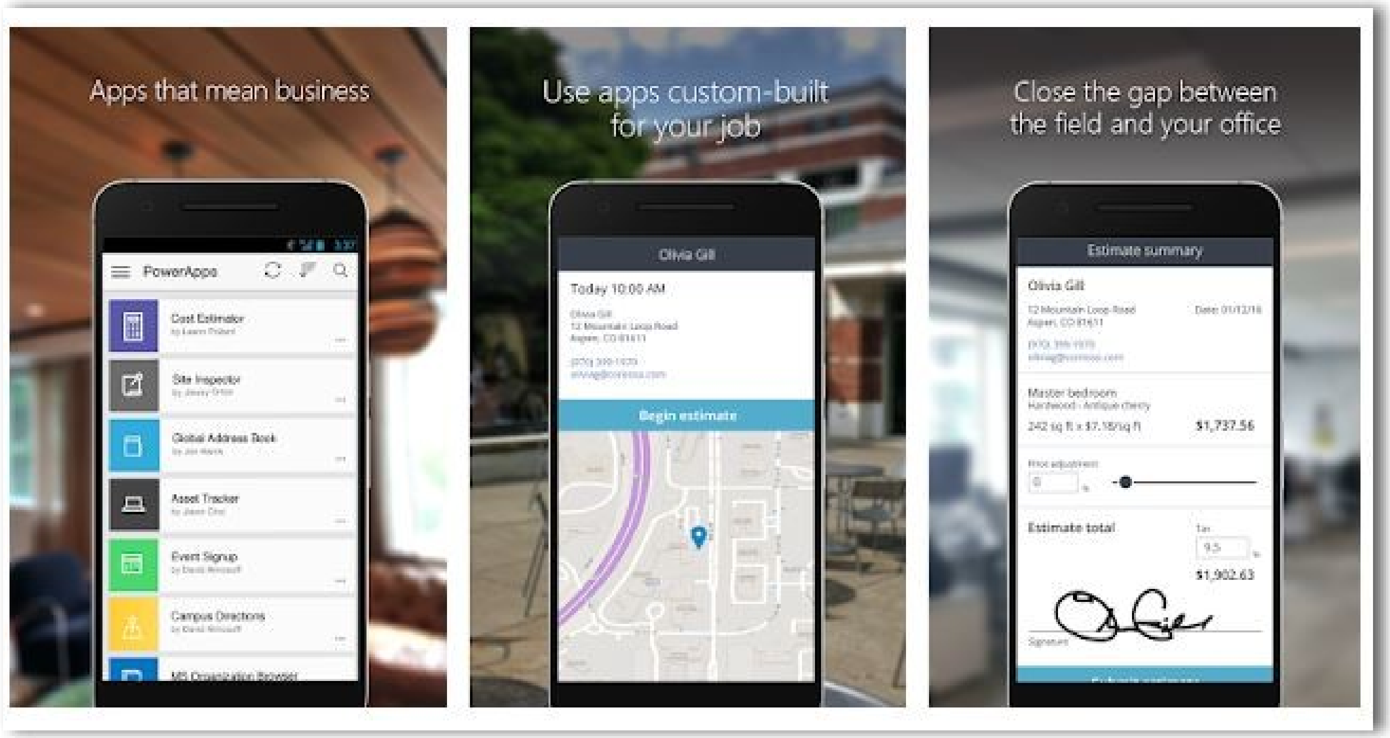

Triggers - A trigger is an event that starts a flow
Get list of new PowerApps, Flows and Connectors
+3
By Microsoft
Used 45 times
Receive an email with a list of new Custom Connectors
+2
By Microsoft

Mobile App

Using PowerApps on a mobile device

To run a PowerApp on your mobile device simply install the PowerApps application via the Google Play Store (for Android devices) or Apple App Store (for Apple devices). The first time you start the PowerApps app, it will ask you to sign in to your organizational account. When this is done, all your company apps will be displayed here (at least all the ones that are shared with you). If you have a couple of apps that you use frequently you can add them to your home screen (pulling them out of the PowerApp app) – however, this functionality only works on Android at this moment.

The screenshots below are sourced on the Google Play store (provided by Microsoft) and give a great example of what a PowerApp looks like on a mobile device.



If you don't want to install another app, then don't worry, you can access them via your browser.

Going Offline with PowerApps

One of the major use cases for PowerApps is to allow workers to use apps on the road. One common scenario for a remote worker (such as a travelling salesman) is that they may not always be connected to the internet.

You will be pleased to know that PowerApps does offer some support for working 'offline'.

To build support in our PowerApp to handle offline data can be achieved by utilising a few useful expressions as follows:

Expression	**Description**
Connection.Connected	Allows us to test if our PowerApp is connected to the datasource. If it's isn't then we need to save and query the data from the local storage (on the mobile device).
Clear	This will remove records from your collection.
Collect	Allows you to store records in the local cache, if we have no connection.
LoadData	Allows you to load data into a collection locally.
SaveData	Allows you to save data to a collection locally.
Patch	Updates/creates a record in the data source. Ideal if you have a connection.

As you can see, offline support can be quite a complex subject if you're not a developer by trade, however, once you've grasped the concepts it's quite straightforward.

Multiple language support

How to support multiple languages

It is entirely possible to create a PowerApp that renders text in the language of the logged on user. Note: If you think your PowerApp may need to be multi-lingual in the future then it's wise to provide support from day 1. Delaying it will result in you have to refactor every label, text, tool tip and message which can be a painful job.

How to return the language for the current user?

Before we do anything, we need to retrieve the users language. The language can be determined by calling the "language" function. "Language()" returns the "language", "script" and "region" for the current user as a tag, e.g. "en-GB" would return for someone living in Great Britain.

Change all of your hard-coded text to the current language

The golden rule when adding support for multiple languages in your PowerApp is not to hard code text. Instead, you should use a function that looks up the language from a local dataset that stores the key name and string.

Your dataset can be imported from something like Excel and would be in this format.

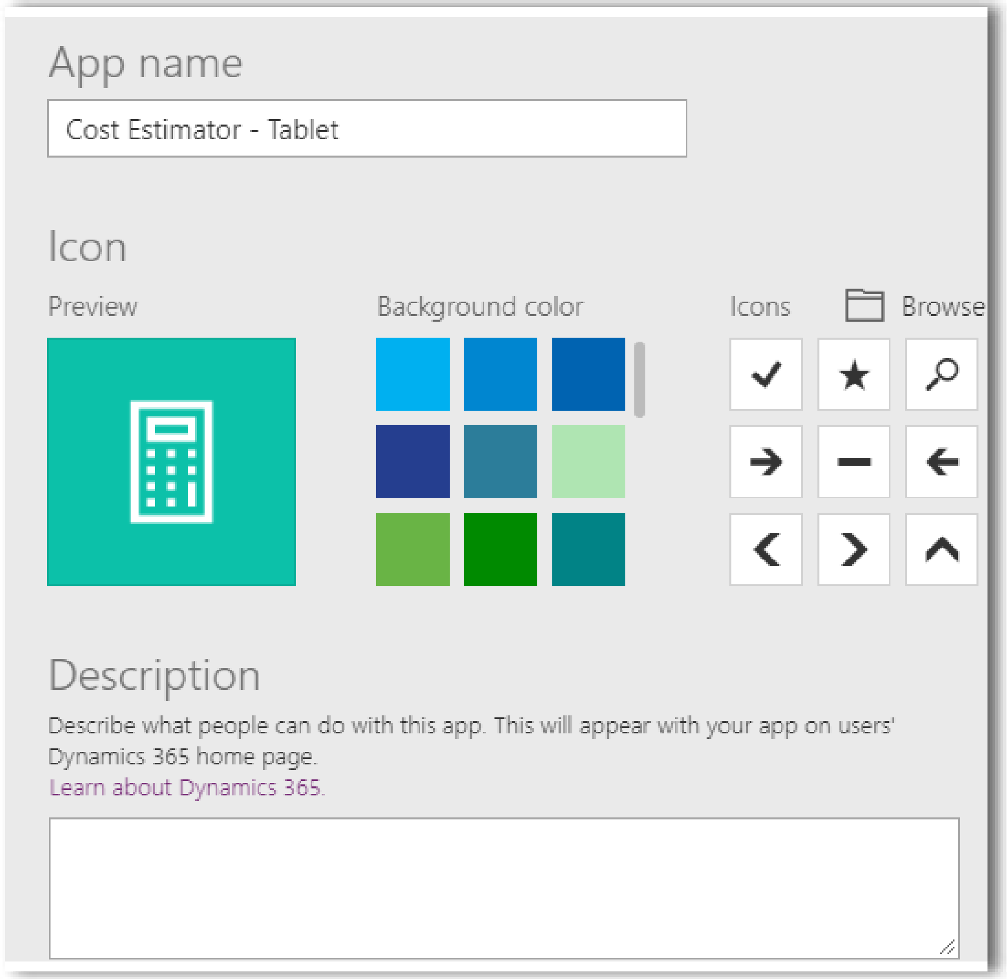
Language	Key	StringName
en	SaveTooltip	Save the customer
pt	SaveTooltip	Salve o cliente
fr	SaveTooltip	Sauver le client

Now that we know which language the user needs and have a language dictionary, all we need to do is perform a lookup like this (note, you will more often than not store the users language somewhere locally):

```
LookUp(LanguageDict, Key = "SaveTooltip" And Language = "en")
```

Publishing & Sharing your PowerApp

When your PowerApp is ready, give it a good name, icon and description and then publish it. After publishing don't forget to share it with the correct users so they can access it from their PowerApps app on their mobile device.



App name

Cost Estimator - Tablet

Icon

Description

Describe what people can do with this app. This will appear with your app on users' Dynamics 365 home page.
Learn about Dynamics 365.

PowerApps also has versioning enabled, which means users will only be able to view and access published versions. This means you can make changes to your app in the background without affecting their use of the tool. For example, you could work your way up from version 1.0 to version 1.7 before publishing this to version 2.0. Users would continue to use version 1.0 of the app until version 2.0 was made available to them.

Purchasing PowerApps

Licensing of PowerApps

If your organization already uses Office 365 or Dynamics 365, PowerApps is included in your license so you can start using PowerApps to create canvas apps and model driven apps (only for Dynamics 365 users).

If you don’t have one of the subscriptions mentioned above, you can use PowerApps via Plan 1 which costs 7$ per user month.

If you are a developer and need to create some custom connectors or need to use the common data model, you can use PowerApps via Plan 2 which costs 40$ per user per month.

Some connectors are also premium connectors which are available at an extra cost.

Note: prices are at the moment of writing and can change. For up to date details or more information go to https://powerapps.microsoft.com/en-us/pricing/.

Note: You maybe wondering if it's possible to share your PowerApps with external users. Unfortunately, (at the time of writing this), it's not currently available. There is a user voice request here if you wish to vote for this feature.

Part 2: Tutorials

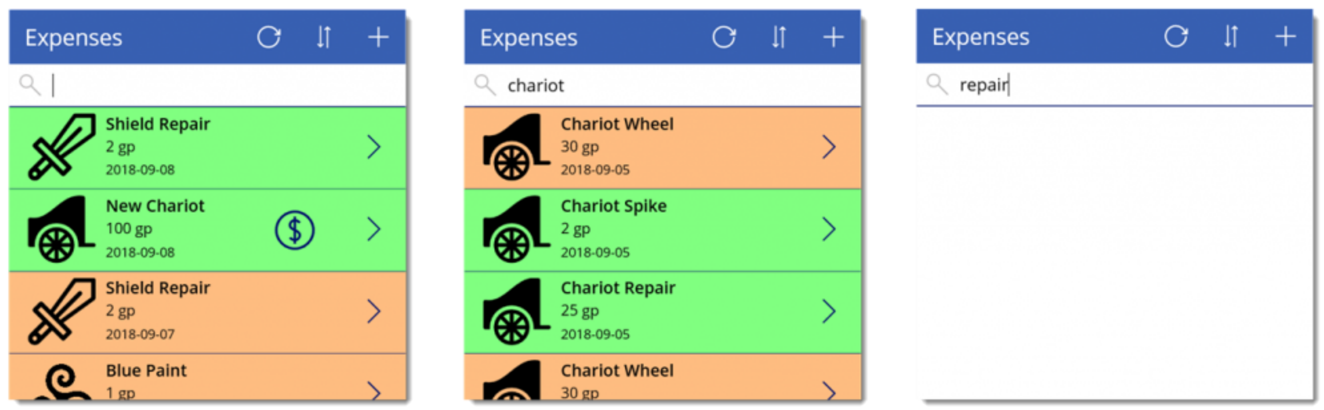
Tutorial #1 - How to search and filter data

This tutorial explores the default search available when you choose the "Start from data" template. Using this template, it's possible to connect to a SharePoint List (amongst others) and have a "Browse", "Detail" and "Edit" built for you.

When the three screen PowerApp is created (using a SharePoint list as the data source), it also includes a search box allowing us to perform a straightforward search.

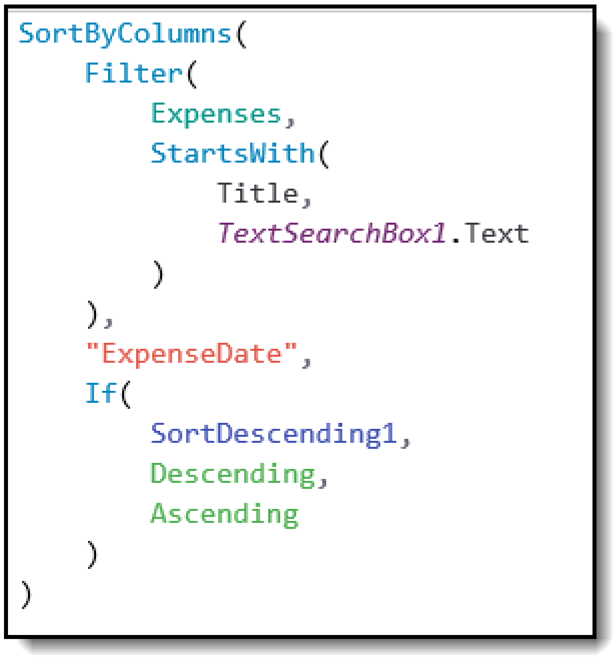
In the screenshot below, you may notice the following:

1. **Left** - A default search screen showing all results.
2. **Middle** - A search results screen filtered requesting results beginning with "Chariot"
3. **Right** - A search results screen that returns no results based on the word "repair" (more on this later).



The PowerApps control that allows us to browse the list items, is called the "Gallery".

The order of the sort (ascending / descending) is determined by the variable "SortDescending1" (see below) which toggles between true and false by clicking the sort icon. Changing the variable instantly changes the gallery, no refresh is required.



```
SortByColumns(
    Filter(
        Expenses,
        StartsWith(
            Title,
            TextSearchBox1.Text
        )
    ),
    "ExpenseDate",
    If(
        SortDescending1,
        Descending,
        Ascending
    )
)
```

By default, the Gallery has a data property called "Items" which includes a "SortByColumns" function allowing us to sort the data. By default this will sort by the Title. In the code example above, I show you how to sort by the column named "ExpenseDate".

Now let's try to improve the search (to filter on "repair")

To improve the search we need to examine the "Filter" function being used above. This function takes at least 2 parameters, firstly "Expenses", which is the field data returned from the SharePoint list. The 2nd parameter allows us to set a new filter. The filter we used in our original example was "StartsWith" (which explains why “repair” gave no rows). No rows had a "Title" starting with "Repair".

There are two obvious alternative options. The first is to swap "StartsWith" for the "in" operative.

```
SortByColumns(
    Filter(
        Expenses,
        TextSearchBox1.Text in Title
  ),
  "ExpenseDate",
  If(
    SortDescending1,
    Descending,
    Ascending
  )
)
```

Changing to use the "in" operative works as expected. We can now search for "Repair" and get all the matching repairs. The filter also only searches the "Title" column. Hence, if we searched for "Travel" we would return no results. However, we can combine different conditions using the or operative which is denoted as an "||".

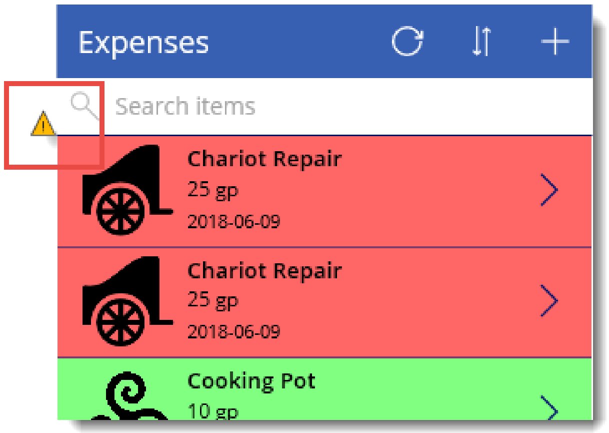
```
SortByColumns(
    Filter(
        Expenses,
        TextSearchBox1.Text in Title || TextSearchBox1.Text in Category
   ),
  "ExpenseDate",
  If(
    SortDescending1,
    Descending,
    Ascending
  )
)
```

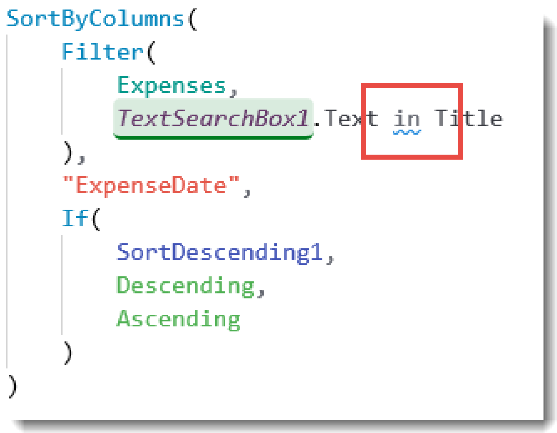
Another way to solve our problem is to use the "Search" formula which allows multiple columns to be specified and offers more flexibility.

```
SortByColumns(
    Search(
            Expenses,
            TextSearchBox1.Text,
            "Title",
            "Category"
        ),
    "ExpenseDate",
    If(
        SortDescending1,
        Descending,
        Ascending
    )
)
```

Dealing with the warnings in the editor

You will notice, as soon as you change the function to use "Search" or "In" you will see a blue wavy line under parts of the code along with a warning triangle.

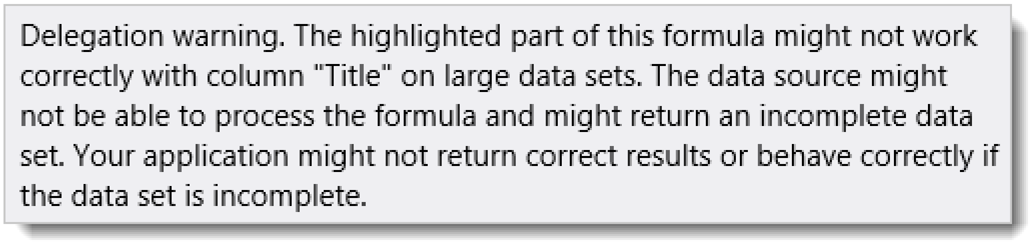




```
SortByColumns(
    Filter(
        Expenses,
        TextSearchBox1.Text in Title
    ),
    "ExpenseDate",
    If(
        SortDescending1,
        Descending,
        Ascending
    )
)
```

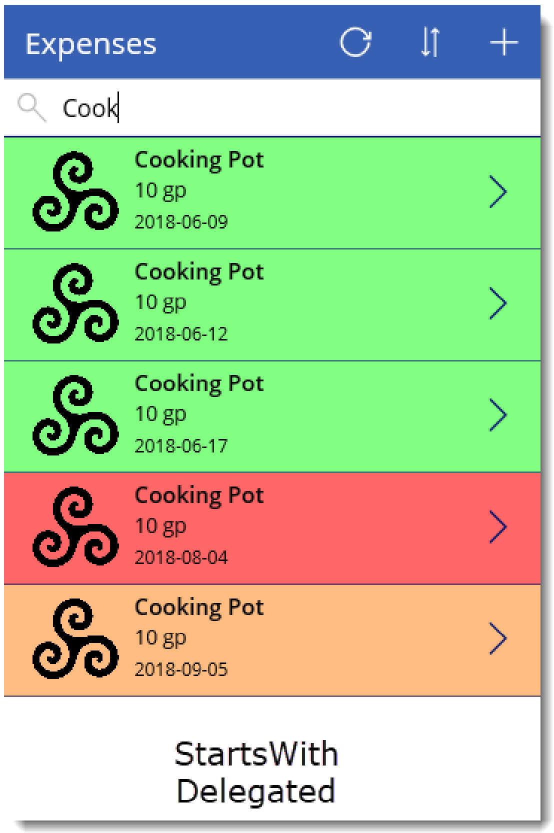
If you click on the triangle or the blue wavy line you will get a warning message concerning "delegation". The message is telling us that large data sets might not work correctly. By default, PowerApps defines a large recordset as being 500 rows. This effectively means that if your search returns 501 rows, none of the rows will be returned. This is for peformance reasons as you don't want to return 10000s of rows to the client, especially on a mobile connection.

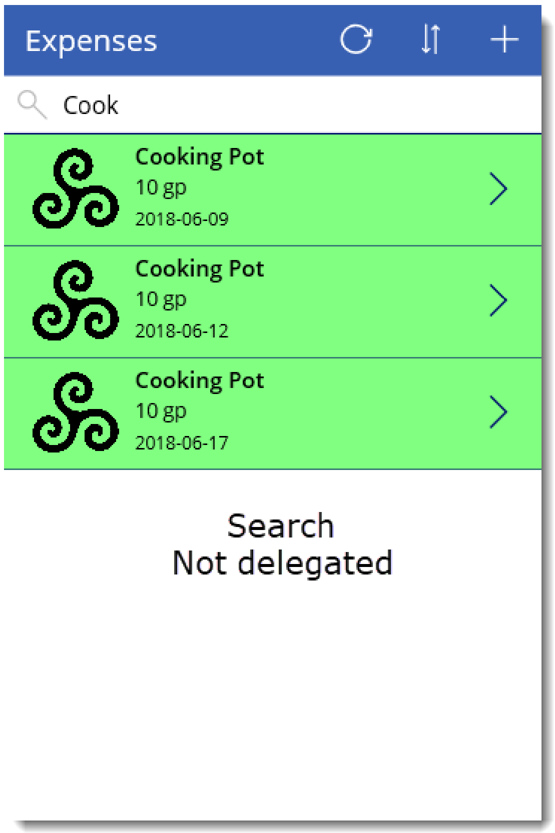
Here's the warning message that you will see:



Delegation warning. The highlighted part of this formula might not work correctly with column "Title" on large data sets. The data source might not be able to process the formula and might return an incomplete data set. Your application might not return correct results or behave correctly if the data set is incomplete.

The example below illustrates the difference between a delegated and non-delegated search:





What is Delegation?

Now that we've seen the effects of 'Delegation' in PowerApps, let's examine what it means. Delegation refers to the process where the filter or sort is sent to the backend data source and then it's the responsibility of the underlying data source to query the data and return the filtered/sorted data. The impact of this means that less data is sent to the PowerApp and the data source which was built to filter and sort takes on the burden of this often expensive processing.

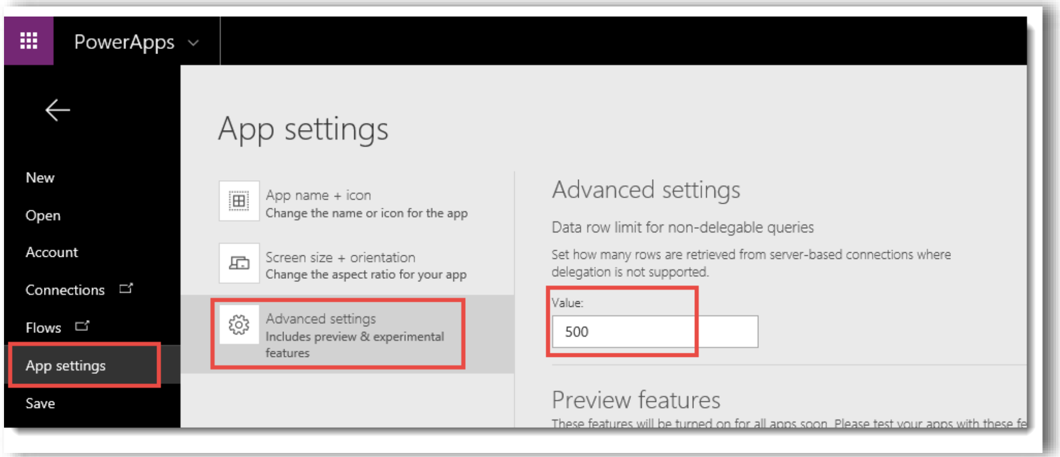
One thing to note, different data sources have different rules regarding which sort and filters can be delegated. This list of rules is constantly expanding and can be found at the following link: https://docs.microsoft.com/en-us/powerapps/maker/canvas-apps/delegation-list

Scrolling down the list you will notice a list of "Top Level Functions" that can be delegated. It's also pertinent to mention that SharePoint allows you to sort the results, but you cannot delegate the search. Whereas, SQL Server (as you'd expect) supports all delegatable functions except the predicate - "StartsWith".

Changing the default 500 row limit.

The default number of rows to be returned (via delegation) is 500. This can easily be changed by selecting the File ribbon tab and then selecting "App Settings" and finally "Advanced settings".

However, you should be aware of the effect of increasing the limit. If you choose a number that's too high, this can cause major performance issues.


PowerApps
New
Open
Account
Connections
Flows
App settings
Save
App settings
App name + icon
Change the name or icon for the app
Screen size + orientation
Change the aspect ratio for your app
Advanced settings
Includes preview & experimental features
Advanced settings
Preview features

Data row limit for non-delegable queries

Set how many rows are retrieved from server-based connections where delegation is not supported.

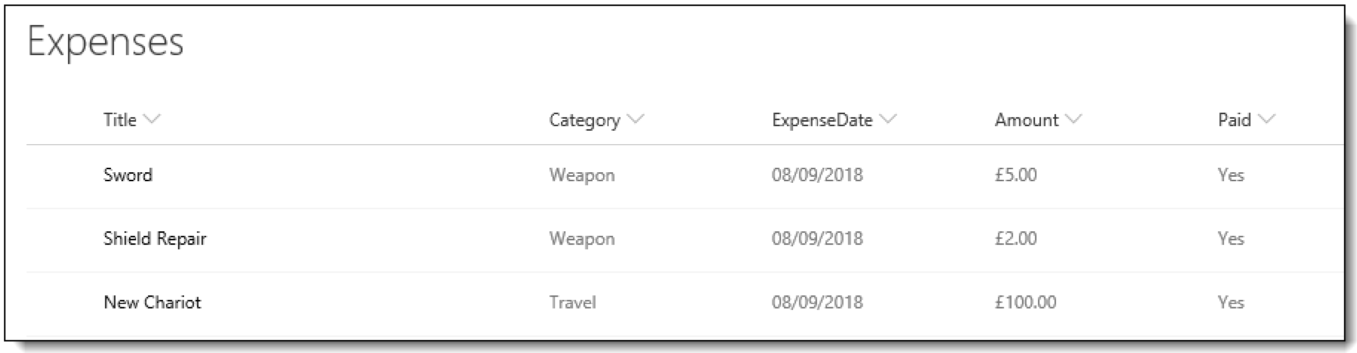
Value: 500

These features will be turned on for all apps soon. Please test your apps with these fe

Tutorial #2 - How to do conditional formatting

This tutorial will show you how to change the look of individual rows (in a PowerApps Gallery) based on logic you specify. The data is based on the theme of Queen Boadicea and managing her expenses within a SharePoint list.

The list below is a modern SharePoint custom list with a few text columns, a date column and a currency column.

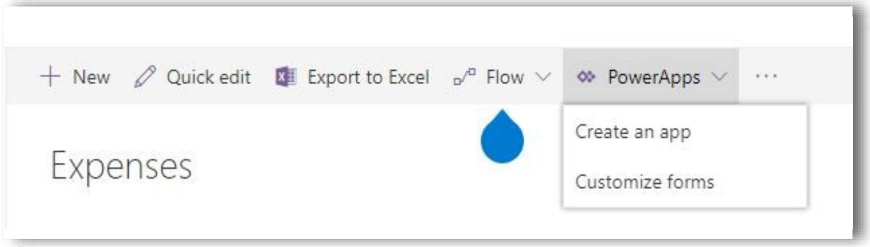


Expenses

Title	Category	ExpenseDate	Amount	Paid
Sword	Weapon	08/09/2018	£5.00	Yes
Shield Repair	Weapon	08/09/2018	£2.00	Yes
New Chariot	Travel	08/09/2018	£100.00	Yes

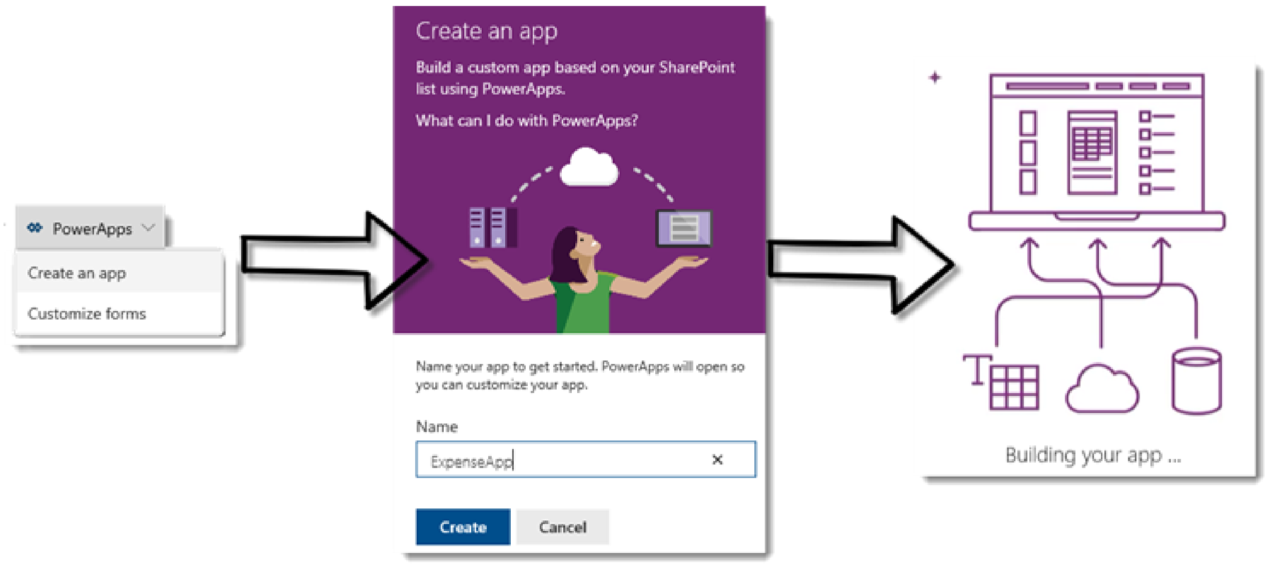
Create an app from the SharePoint List

To create the example used in this tutorial, for speed, let's create a standard PowerApps app by navigating to the list in SharePoint and selecting "Create an app".

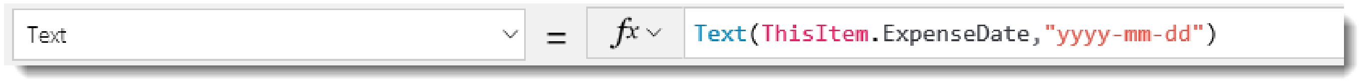


Conditional formatting

After a few moments, a default 3-screen PowerApp will be ready for use on your mobile device.


PowerApps
Create an app
Customize forms
Create an app
Build a custom app based on your SharePoint list using PowerApps.
What can I do with PowerApps?
Name your app to get started. PowerApps will open so you can customize your app.
Name
ExpenseApp
Create
Cancel
Building your app ...

Before we begin, change the gallery layout to image, title, subtitle and body. Please also change the date to a format of “yyyy-mm-dd” using the "Text" function for the "Text" property of the expense date. As below:



```
Text = Text(ThisItem.ExpenseDate,"yyyy-mm-dd")
```

To keep in with our 'Celts' theme let's tweak the value to show Amount field as gp and sort by the Expense Date.

Conditional formatting

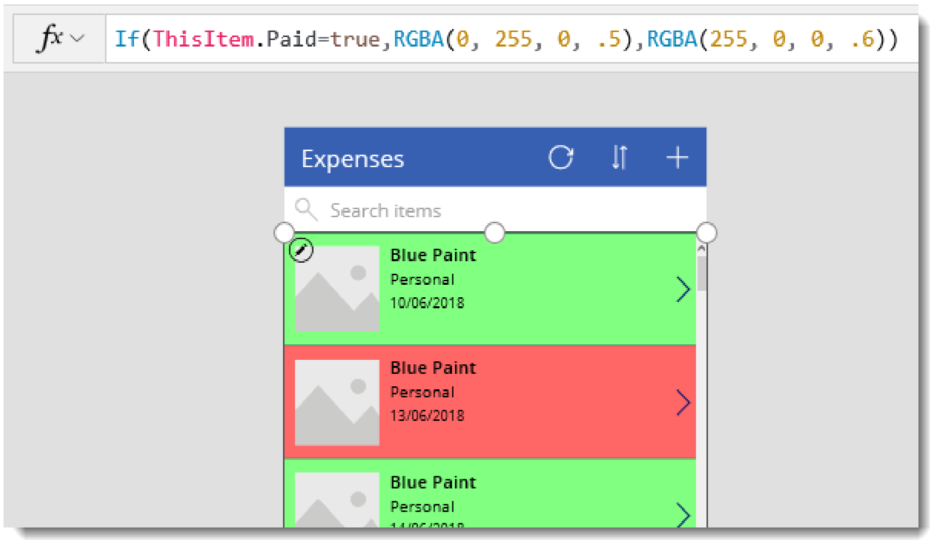
Colouring rows based on a condition...

In this section, let's set a colour for each row in our Gallery. We want the row to be displayed in green if they have been paid and red if not. Colours in PowerApps are specified using RGBA functions that take 4 parameters (red, green, blue and an alpha channel, which specifies the opacity of the colour between 0 and 1). The colour numbers are 0 to 255.

Step 1: Select the gallery object and in the property drop down select TemplateFill.

Step 2: In the formula bar tweak the colour to be bright Red and change the opacity to 1. This changes every row to be bright red. Tweaking the red value or changing the opacity will change the colour. Bright red would be "**RGBA(255,0,0,1)**"

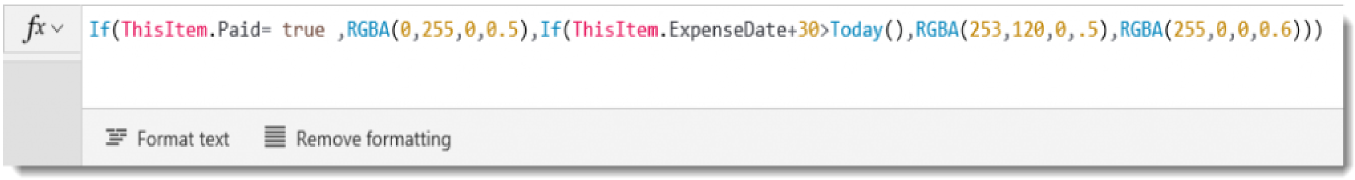
Step 3: Now we will add an If function into the formula to change this. The If function is just the same as Excel's; if, the test, what to do if it is true, what to do if it is false. In our test if the Paid column, which is a yes / no column is "true", i.e. yes. In other words the list item will display green if it has been paid and red if it hasn't been paid.



```
If(ThisItem.Paid=true,RGBA(0, 255, 0, .5),RGBA(255, 0, 0, .6))
```

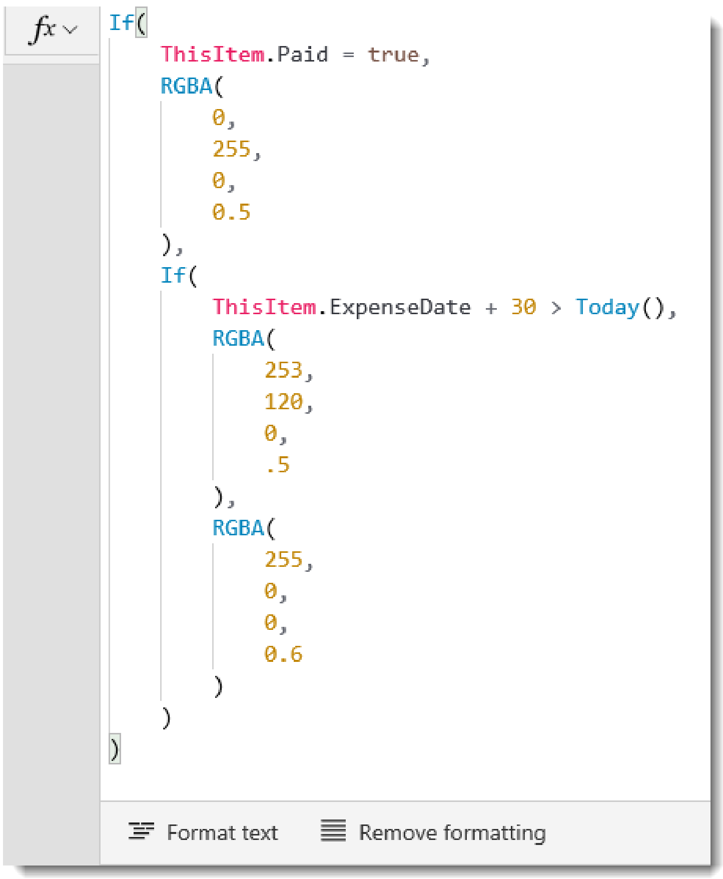
Step 4: Now only unpaid rows are coloured red. If we want to add another colour to represent a different status, e.g. amber to show if an expense is recent, then we need to nest another "if" statement inside the first one. If the expense date plus 30 is greater than today, i.e. the date was in the last 30 days then it will be coloured amber for an upcoming payment.

Conditional formatting



```
If(ThisItem.Paid= true ,RGBA(0,255,0,0.5),If(ThisItem.ExpenseDate+30>Today(),RGBA(253,120,0,.5),RGBA(255,0,0,0.6)))
```

If you prefer the functions laid out then click Format text.



```
If(
    ThisItem.Paid = true,
    RGBA(
        0,
        255,
        0,
        0.5
    ),
    If(
        ThisItem.ExpenseDate + 30 > Today(),
        RGBA(
            253,
            120,
            0,
            .5
        ),
        RGBA(
            255,
            0,
            0,
            0.6
        )
    )
)
```

Changing the picture based on a condition …

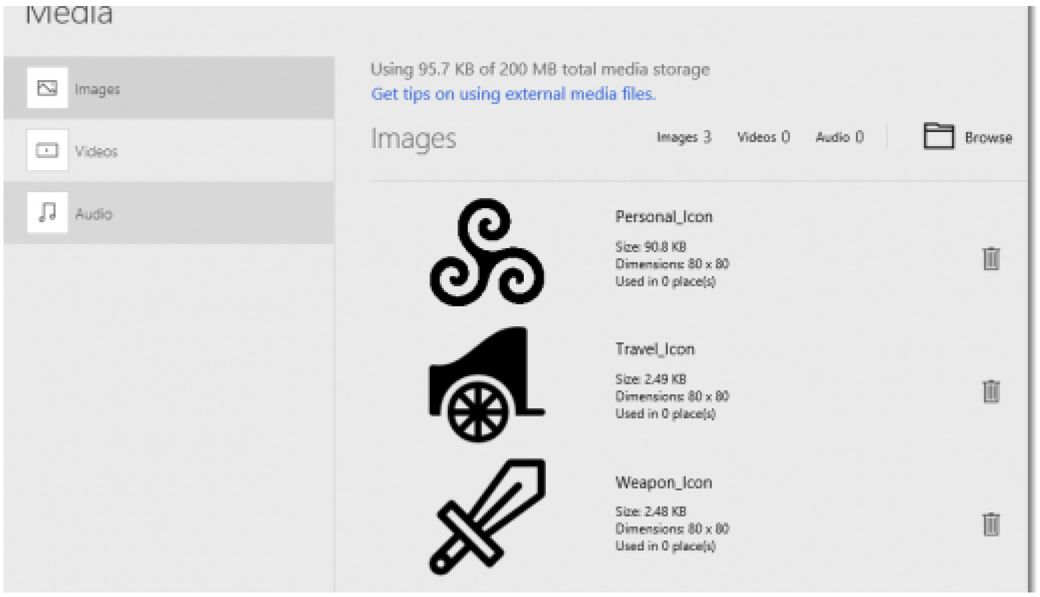
The list of expenses includes a category which is either Personal, Weapon or Travel. I want these to be shown using small pictures. The first step is to load the media.

Step 1: On the View ribbon, click Media and make sure you have Images selected on the left.

Conditional formatting

Step 2: In the top right click Browse and select your images. You can use the Ctrl key to select multiple images from one folder.

Step 3: Click Open to load the images into your app. Note you are limited to 200 MB of media storage.



Using 95.7 KB of 200 MB total media storage
Get tips on using external media files.

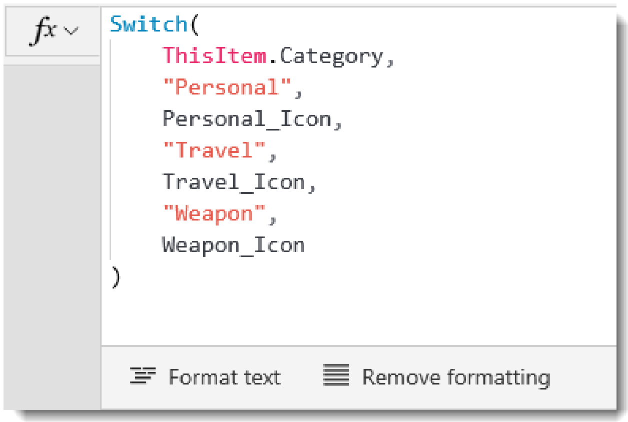
Personal_Icon
Size: 90.8 KB
Dimensions: 80 x 80
Used in 0 place(s)

Travel_Icon
Size: 2.49 KB
Dimensions: 80 x 80
Used in 0 place(s)

Weapon_Icon
Size: 2.48 KB
Dimensions: 80 x 80
Used in 0 place(s)

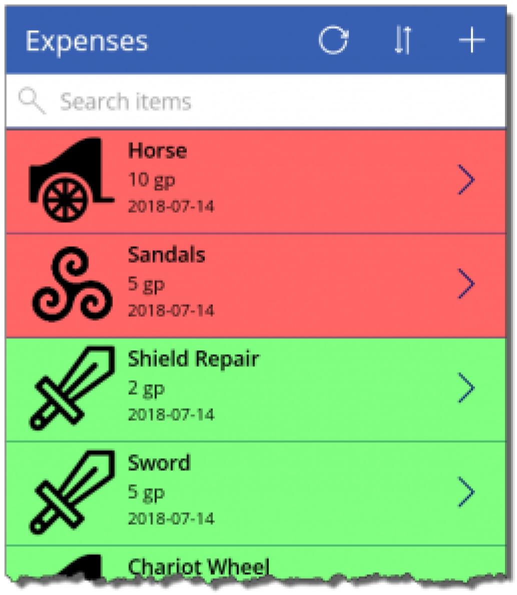
Step 4: Click on the image in the top gallery section and select the Image property. It currently is set to SampleImage.

Step 5: To replace the images we'll use the Switch function. It takes a single value and then pairs of matching values and the result.



```
Switch(
    ThisItem.Category,
    "Personal",
    Personal_Icon,
    "Travel",
    Travel_Icon,
    "Weapon",
    Weapon_Icon
)
```

The list should now show images based on the category as below:



Horse
10 gp
2018-07-14

Sandals
5 gp
2018-07-14

Shield Repair
2 gp
2018-07-14

Sword
5 gp
2018-07-14

Chariot Wheel

Hiding a value based on a condition ...

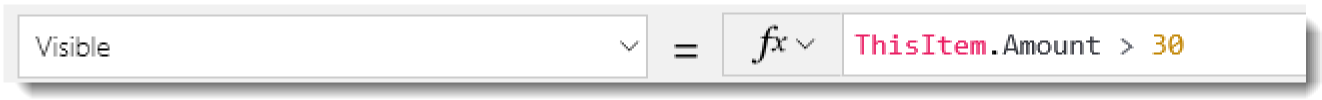
The final section in this tutorial is perhaps the easiest idea. Hide or show an icon based on a simple logic test. In this section we will show an icon if the value is over 30.

Step 1: Select the "BrowseGallery1" and click the edit icon to select one row of the gallery.

Step 2: From the Insert ribbon, select an icon. Position where you want it to be in the row. One will be shown on every row.

Step 3: Select the "visible" property, it will be set to "true".

Step 4: As the visibility is just true or false it ca be a test and doesn't need an if or switch function.

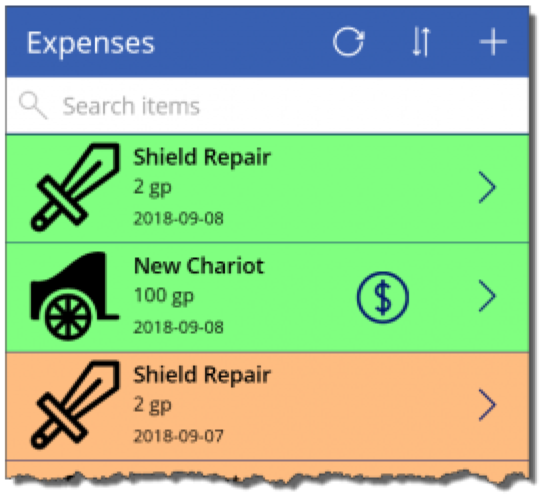


Visible = *fx*

```
ThisItem.Amount > 30
```

Conditional formatting

By applying the formula above you will now notice that there are only rows visible where the value is greate than 30 gp



Shield Repair
2 gp
2018-09-08

New Chariot
100 gp
2018-09-08

Shield Repair
2 gp
2018-09-07

Tutorial #3: Use scrolling Text in PowerApps

Now we will see how to use a timer control along with a touch of maths to scroll a text message across the screen. It's a very simple example but illustrates some basic animation which can be used in many applications.

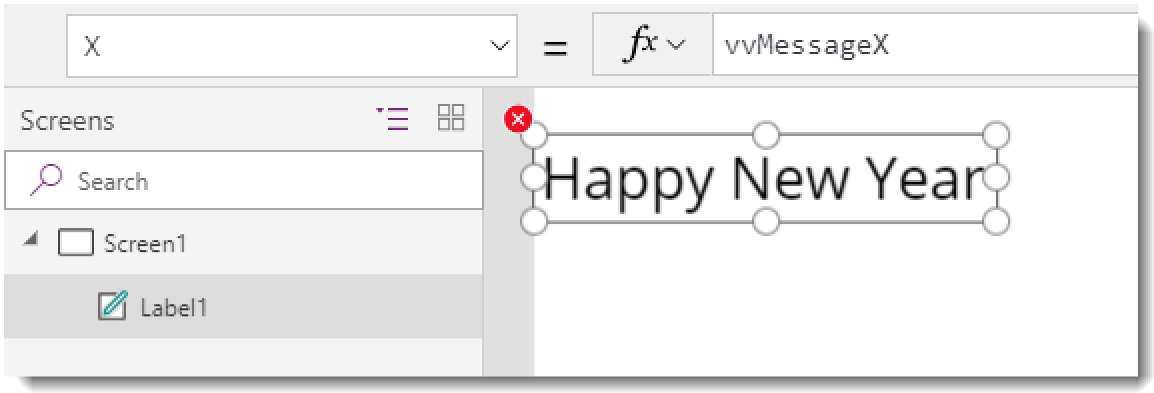
First, add a simple message to a new PowerApp

Before we start looking at timers we need to add the message which will scroll to the PowerApp.

Step 1: In the PowerApp, add a label that contains your message formatted how you want.

Step 2: Resize the label so that it fits the text exactly.

Step 3: Set the "X" value of the label to a variable, e.g. "vvMessageX".



```
X = vvMessageX
```

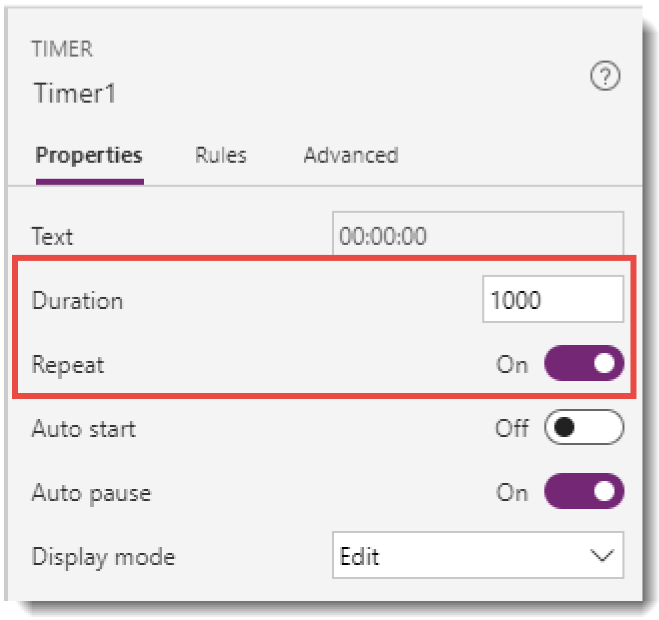
The UI will show a red cross error, but that’s okay it just means the variable "vvMessageX" hasn’t been set up yet. We will do that in the timer.

Setting up the timer

Timers in PowerApps allow you to have some code run after a period of time and repeat this forever or until stopped. You get to specify when it starts, the period of time, and what actions it does every time. We will start by just increasing the variable vvMessageX every half second.

Step 1: On the Insert ribbon, from the controls drop down, add a timer to your app. It can be hidden later but for now, it looks like a button.

Step 2: The duration of the timer is in milliseconds, this means 1000 = 1 second. Change the duration to be 500 (half second) and Repeat to be on.



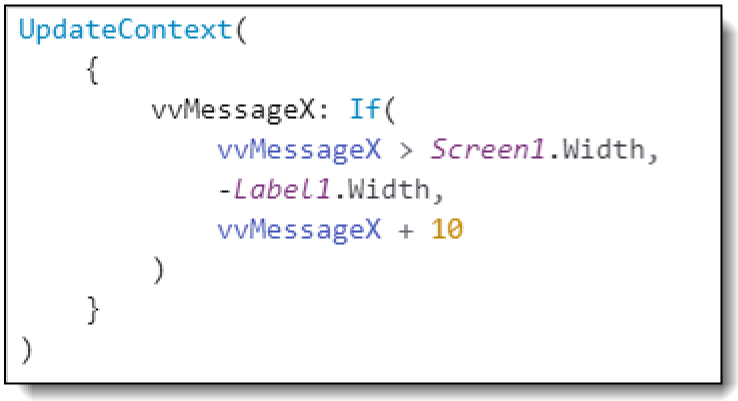
Text 00:00:00
Duration 1000
Repeat On
Auto start Off
Auto pause On
Display mode Edit

Step 3: A timer control has a property "OnTimerEnd", this happens at the end of the duration and repeats because we turned on repeat. Change the property to just add "10" to the variable "vvMessageX". For this, we are going to use "UPDATECONTEXT", which means "vvMessageX" is a local variable to this screen.

```
UpdateContext( {vvMessageX: vvMessageX + 10} )
```

Step 4: Preview the app, click the timer and your message will slowly jump across the screen. We need to speed that up and if you wait long enough your message will vanish off the screen never to be seen again.

Step 5: So to make the message loop around we need to increase until it reaches the screen width and then reset back to the start position. Update the time OnTimerEnd property to include an IF statement inside the JSON statement.



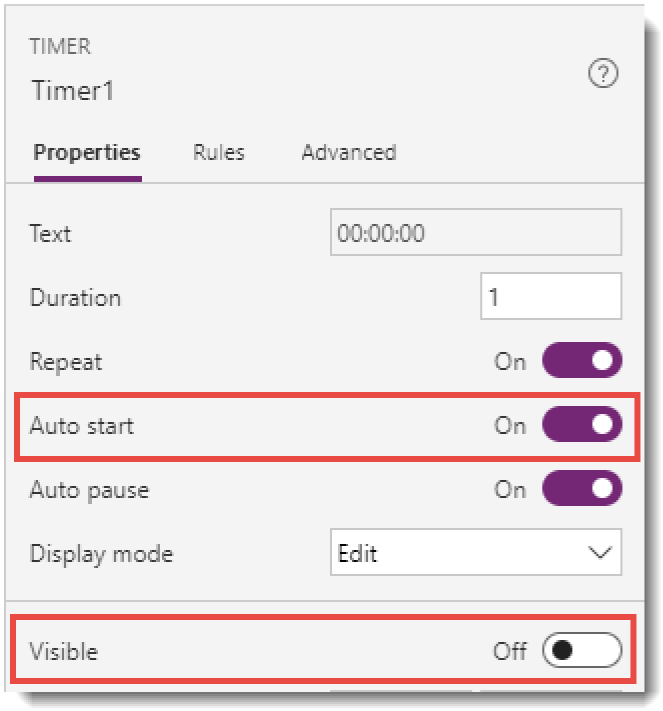
```
UpdateContext(
    {
        vvMessageX: If(
            vvMessageX > Screen1.Width,
            -Label1.Width,
            vvMessageX + 10
        )
    }
)
```

If the variable gets larger than the screen width, i.e. has vanished, then reset it to be off to the left of the screen so it can re-enter, i.e. minus the width of the message label.

Step 6: Now you need to adjust the duration of the timer and how much you increment the variable to balance speed and smoothness of the movement. After a little playing around with numbers, I got the duration down to 1ms and increased by 5.

Finishing touches

At this point the timer should work when you click on it. Now let's hide the timer and make it start automatically. To do this, select the timer and change "Auto start" to "true" and "Visible" to "false".


TIMER
Timer1

Text	00:00:00
Duration	1
Repeat	On
Auto start	On
Auto pause	On
Display mode	Edit
Visible	Off

Preview and test your app and you should see the scrolling text.

Tutorial #4: Creating Tabbed Forms

Tabbed forms are ideal when a form has more controls than screen space. It really helps to use a tabbed form to group items and keep the visible size of your forms manageable. Unfortunately, PowerApps doesn't ship with a tabbed form so we need to create one using a gallery and some groups of controls.

The steps to create this form are:

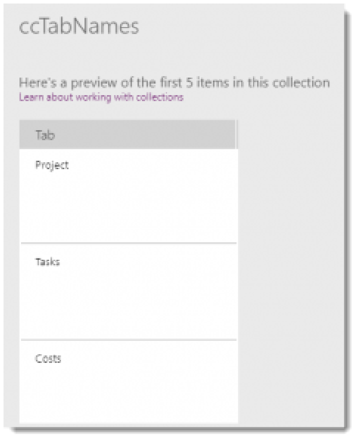
- Create a collection of tab names
- Create a gallery
- Create a group for each tab with a visible property

Create a collection

For this form, I'm going to hard-code the tab names into a collection. This code needs to be in the OnStart of the app and for testing purposes in a button as well. The collection just needs to contain the tab names.

```
ClearCollect(
    ccTabNames,
    {Tab: "Project"},
    {Tab: "Tasks"},
    {Tab: "Costs"}
)
```

This creates a collection with three rows of data.


ccTabNames

Here's a preview of the first 5 items in this collection
Learn about working with collections

Tab
Project
Tasks
Costs

Build a Gallery

A gallery will create the tabs, each tab will be a label that changes colour based on which gallery item is selected.

Step 1: Insert a blank horizontal gallery.

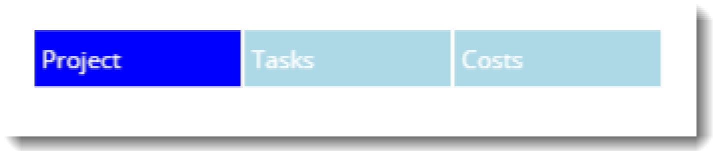
Step 2: Make the data the collection "ccTabNames" created in the last part.

Step 3: Update the gallery to have the following properties.

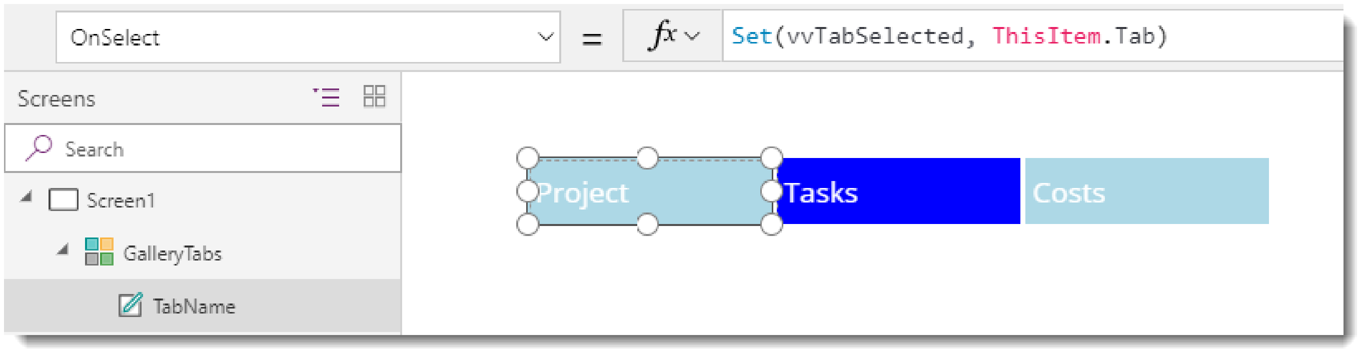
Gallery Name	**Gallery Tabs**
Width	GalleryTabs.TemplateWidth * CountRows(GalleryTabs.AllItems)
Height	40
Template Size	150
Padding	0

Step 4: Add a label to the gallery template and update to have the following properties.

Label Name	Tab Name
Text	ThisItem.Tab
Height	Parent.TemplateHeight
Width	Parent.TemplateWidth – 3
Fill	If(ThisItem.IsSelected, Blue , LightBlue)
Color	White



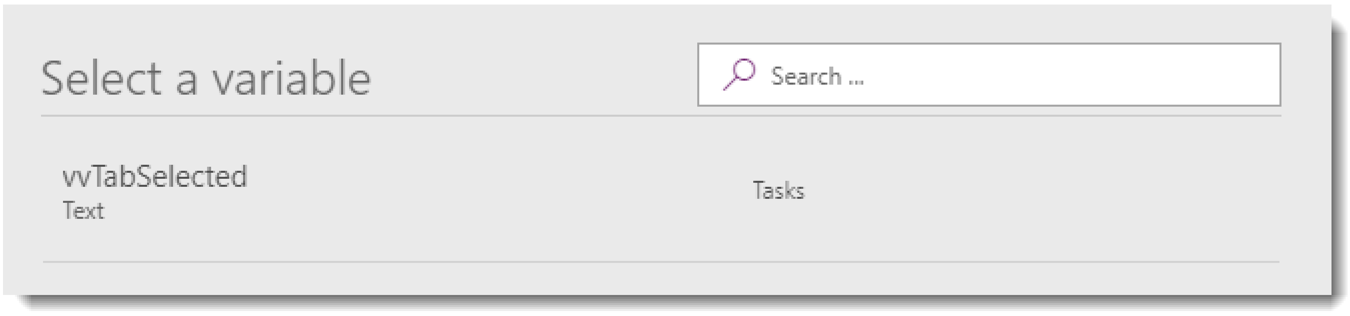
Step 5: Change the "OnSelect" of the label to update a variable, "vvTabSelected", to the tab value.



```
Set(vvTabSelected, ThisItem.Tab)
```

Test your tabs in preview mode. You will need to click the button to run the code created in the previous section.

Tip: View the variables to check the tab value is being saved.



vvTabSelected
Text
Tasks

Groups of controls

The last stage is to create the groups of controls associated with each tab, they will be surrounded by a rectangle with a white fill and a coloured border that matches the selected tab.

Step 1: Insert a rectangle from the Icons drop down, and modify the properties. In this example, the rectangle hasn't yet been renamed.

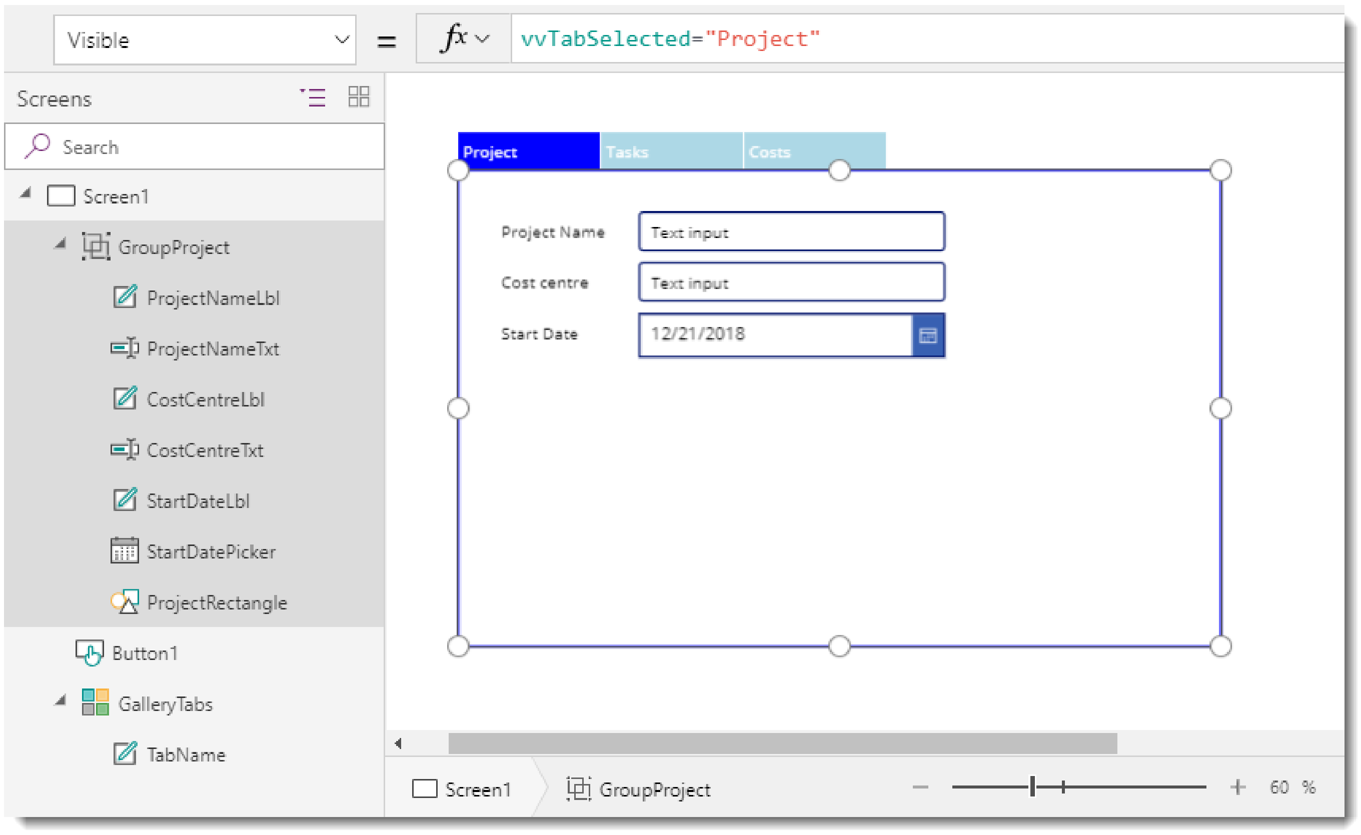
Property	**Value**
X	GalleryTabs.X+Rectangle1.BorderThickness/2
Y	GalleryTabs.Y+GalleryTabs.Height
Width	800 or whatever fits your app
Height	500 or whatever fits your app
Fill	White
Border Color	Blue
Border Width	2

Step 2: Add the controls for that tab.

Tabbed Forms

Step 3: Select the rectangle and all the controls for the tab and group them together. Rename the group to match the tab, e.g. GroupProject.

Step 4: Select the group which will select all the items and change the visible property to **vvTabSelected="Project" .**



Visible = fx

```
vvTabSelected="Project"
```

1. Repeat steps 1 – 4 for each tab required and remember to test as you go.

Tutorial #5: Using a Google Map in PowerApps

Maps are awesome and adding a map to an app (even a static one), adds that special touch. This post is an introduction to adding a Static Google map to a PowerApp.

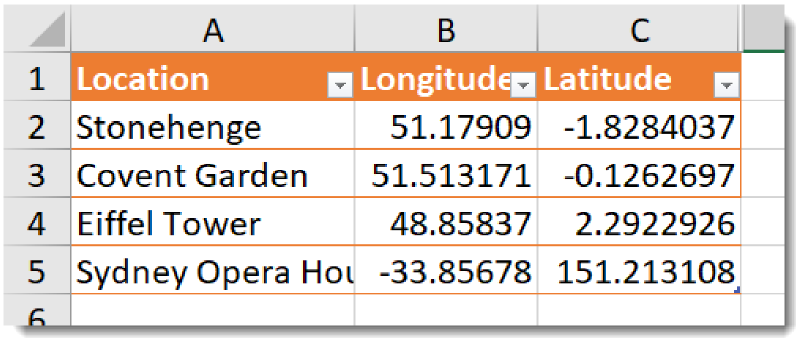
Create your Google API Key

In order to connect to Google and request a static map image, you will need an API key. This is much simpler than it sounds however, Google really helps.

Visit https://developers.google.com/maps/documentation/maps-static/intro and scroll down to find the Get Started link. This will walk you through getting an API key. It will give you a key something similar to "AIzaPyCC6Jfzjo50meU9DRsf-duxS7_VfPmzc-s". Copy your key as you will need this later.

Adding a Static Google Map

For this demo app, I have created an Excel file for my data source (named "Locations") containing a list of locations along their longitude and latitude. Shown below:

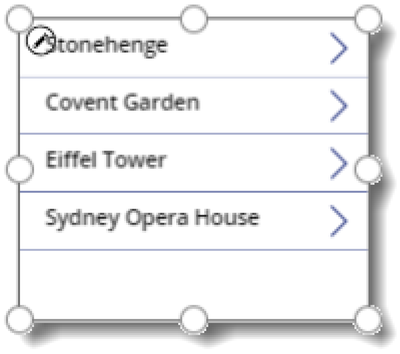


	A	B	C
1	Location	Longitude	Latitude
2	Stonehenge	51.17909	-1.8284037
3	Covent Garden	51.513171	-0.1262697
4	Eiffel Tower	48.85837	2.2922926
5	Sydney Opera Hou	-33.85678	151.213108
6			

Using a Google Map

Step 1: Build an app and add your list of locations with longitude and latitude as a data source.

Step 2: Add a gallery with Locations as the data source.



Step 3: Add a button to your app to setup up some variables. These variables are the various parts of the URL needed to get the static image from Google. Add the following to the OnSelect property

```
UpdateContext({ vvHTTPStart: "https://maps.googleapis.com/maps/api/staticmap?",
vvKey: "YOUR-API-KEY", vvMapZoom: 15, vvMapSize: "400x400" })
```

Step 4: The image we are going to show on the PowerApp comes from a url.

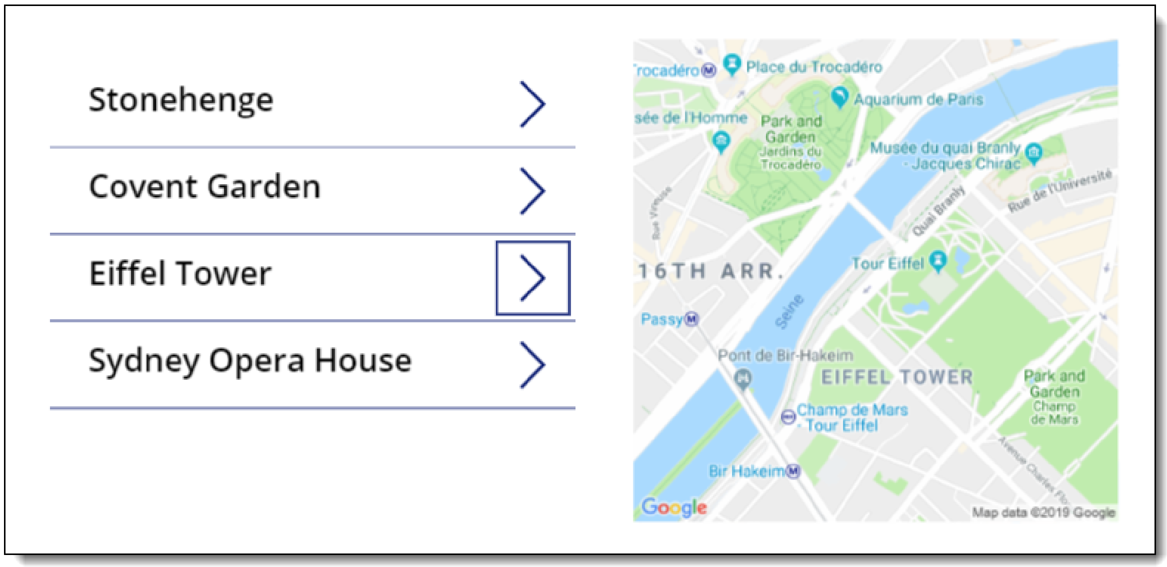
An example of the url is: https://maps.googleapis.com/maps/api/staticmap?key=YOUR-API-KEY&size=400×400&zoom=13&center=51.17909,-1.8284037&

We are going to store this URL in a variable vvMapAddress which will get updated by clicking on the gallery. So change the OnSelect for the Gallery template to the following:

```
UpdateContext({ vvMapAddress: vvHTTPStart & "key=" & vvKey & "&zoom=" & vvMapZoom
& "&size=" &vvMapSize & "&center=" & Gallery1.Selected.Longitude & "," &
Gallery1.Selected.Latitude })
```

Step 5: The final step is to add an image control from the Media drop-down on the Insert ribbon. Resize the image to match the vvMapSize, i.e. Width 400 and Height 400. Set the Image property to vvMapAddress.

Step 6: Preview the image, click the button to set up the variables and click on the gallery to select a location.



Part 3: Tips

My PowerApps Tips

Hopefully, if you've read this far you will feel comfortable with the basics of PowerApps and realise the power they offers you. In this final section, we wanted to share some of the tips to make your life easier.

Sometimes your app is just not suitable to be a PowerApp!

Not every scenario asks for a PowerApp, let's say you are a company with 1000 employees and you want to roll-out a mobile application which is used 4 hours per day by half of your employees, your data is living on-premises as well as in the cloud and next to that it should work completely offline and should be shared with externals...

This scenario is not meant for a PowerApp, you do want to have a fully customizable application which is available to external persons, so you should go for a stand-alone apk. Creating a stand-alone solution is likely to cost more to develop than a PowerApp, however, it will provide a more sustainable solution in the long term.

Don't overcomplicate it!

It's ok to put a lot of data connections in your PowerApp, this is one of the things it was designed for! However don't overcomplicate your functions and automations. If you do want to do some advanced logics and automate your business logic, don't forget about MS Flow.

Write basic functionality and automation principles in PowerApps, however, from the moment you think something will take a bit more time, hand over the tasks to MS Flow. PowerApps and Flow were launched at the same time because they are complementary, so use them both!

Make use of Collections and Variables

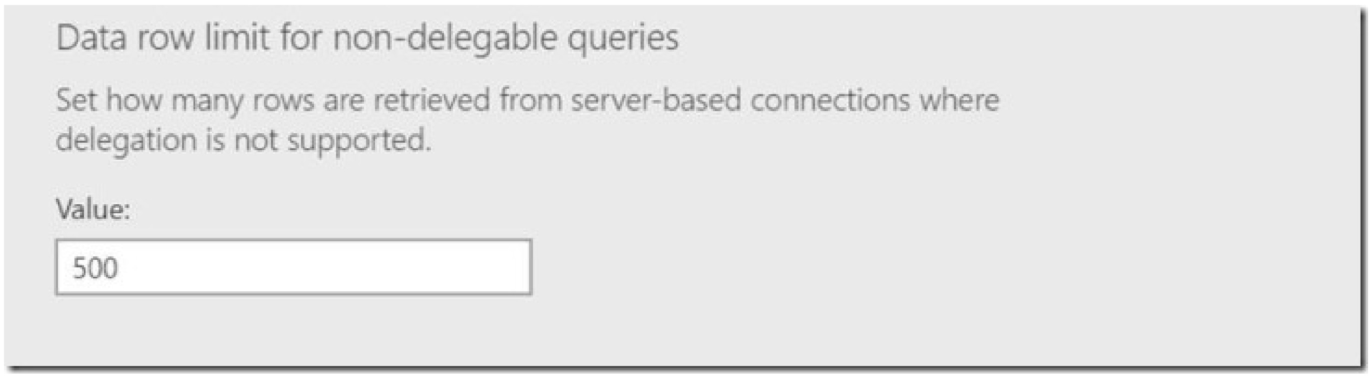
When laying your data connections, always consider if it is necessary to have live data or if this can be cached. You will see that only a small percentage of your data has to be live, so why execute a whole bunch of queries to live data when caching is much more performant. You will want to look at the Collect functionality here.

Use the function ClearCollect(cachedDataSource, DataSource), from then on use cachedDataSource instead of DataSource. When you think your datasource should be updated, simple execute the function ClearCollect(cachedDataSource, DataSource) again.

Delegation, delegation, delegation

PowerApps doesn't want to query thousands of records and perform filters on these records, this because of ... that's right, performance reasons. This is why you always have to think about delegatable sources. By using the correct formulas, you can delegate the processing of data to the data source, instead of retrieving all your data over the network and then processing locally.

When delegation is not an issue, or you are 100% sure your data source won't hold more than 500 records, you could consider forgetting about delegation for this specific data source. PowerApps allows you to increase the number of retrieving rows from 500 to 2000, however, don't touch this number when this isn't necessary.



Data row limit for non-delegable queries

Set how many rows are retrieved from server-based connections where delegation is not supported.

Value: 500

For more info: https://docs.microsoft.com/en-us/powerapps/maker/canvas-apps/delegation-overview

PowerApps are personal

When rolling out a PowerApp do not forget about correct permissions. Keep in mind that PowerApps is personal, this doesn't mean they cannot be shared, this simply means that when a person starts up a PowerApp everything will run under his own account. This means all your PowerApp users will need to have access to the lists/libraries that are used, but also the office 365 or other connections that you will use.

So be careful with this! Do not rollout your PowerApp before validating this with some test users, because nothing is more annoying than launching your first PowerApp with errors saying: "There was a problem saving your change. The data source may be invalid". At the moment of writing, impersonating or elevated permissions are not possible, so do spend some time on getting your permission matrix right!

Name everything!

Your PowerApps can easily grow to have many controls. It's important you define a convention and name everything so that it's identifiable. So no more "textBox1" and more "txtCustomerAddress".

Need help with PowerApps?

We really hope this E-book has been a useful guide to show you how to get started with PowerApps and also achieve some common things in our tutorials.

However, as with everything, you may need some help from time to time. Maybe a 30-60 minute call to discuss some of the concepts discussed in this book would be useful? Or, if you have a need for a PowerApp perhaps you'd like us to assist or even build it for you.

The authors of this book, (Dries and Laura) are both very experienced PowerApps Freelancers and are available to help you online via Collab365 MicroJobs.

Laura's MicroJobs

View Laura's MicroJobs

Dries MicroJobs

View Dries MicroJobs

Why use MicroJobs to hire Microsoft Freelancers?

At Collab365, we believe the way we work is changing drammatically. We documented our thoughts in the '[Future of Work and what will it mean for your job?]".

How does MicroJobs work and what about payment?

Find out how MicroJobs works [on this page].

Paying for online services with people that you don’t know can be worrying for both parties. The buyer often doesn’t want to pay until they’re happy that the Freelancer has completed the work. Likewise the Freelancer wants to be sure they will be recompensed for their time and commitment. Collab365 MicroJobs helps both the buyer and the Freelancer in these ways:

1. The buyer pays up front and the money is securely held in the MicroJobs Stripe Connect platform account.

2. The Freelancer can then begin the work in the knowledge that the payment has been made.

3. Once the buyer is happy that the work is complete and to their satisfaction, the funds become available to the Freelancer.

4. There’s even a dispute management function in case of a disagreement. But it shouldn't happen. As long as the deliverables are agreed up front and both parties keep talking the entire way through, you won’t be disappointed.