Power Platform
COMMUNITY CONFERENCE

Full Power!

Power App Optimisation

Keith Atherton

Power Platform
COMMUNITY CONFERENCE

SEPTEMBER 18–20, 2024 • *Workshops: Sept 16, 17 & 21*

MGM GRAND • *Las Vegas, NV*

Whova

The official event app for the **Power Platform Community Conference**

Join the event app to access:

- Event announcements
- Personalized agenda, session details
- Speaker & attendee profiles
- Networking, meet-ups, messages
- Event documents

Event Invitation Code: PPCConf2024

Full Power! Power App Optimisation

Keith Atherton

Who am I?

Keith Atherton

- Power Platform Solution Architect for ANS Group, UK
- Microsoft MVP for Business Applications
- Microsoft Certified Trainer (MCT)
- LinkedIn Learning Instructor
- Power Apps Super User
- 12x Microsoft certified

Microsoft® Most Valuable Professional

Connect with me

Microsoft Certified Trainer MCT 2023 - 2024

S1 23 | S2 23 | S1 24

Microsoft Certified: Power Platform Solutions Architect Expert

Microsoft Certified: Power Platform App Maker Associate

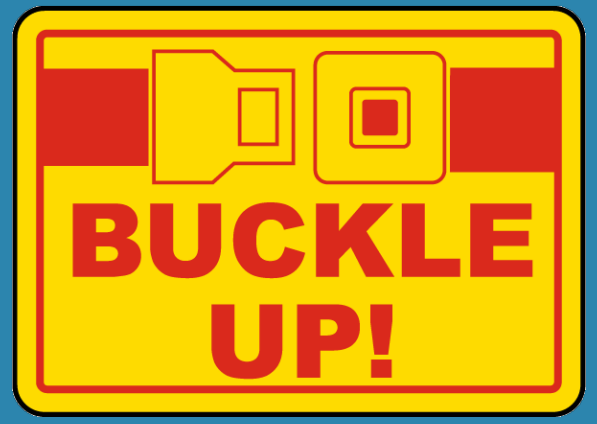
Microsoft Certified: Power Platform Functional Consultant Associate

Microsoft Certified: Power Platform Developer Associate

I’ve got stickers for this session! #FirstComeFirstServed

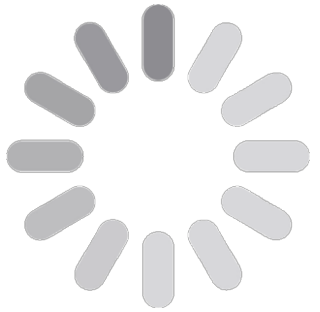
Agenda

- Code optimisation
- Architectural optimisation
- User experience (UX)
- Monitoring



Why optimise?

- Improved user experience
- Improved user satisfaction
- Improved user adoption
- Non-functional requirements (NFRs) may dictate app load and response time limits.



Loading, please wait… Almost done… Nearly there, promise…

The *RAIL* performance model

- Originated by the Google Chrome team in 2015
- Mantra: *“Focus on the user; the end goal isn't to make your site perform fast on any specific device, it's to make users happy.”*

R = Respond to user interactions within 100ms

A = Animations should render each frame in under 16ms (60fps)

I = Idle time non-essential work to be performed within 50ms chunks

L = Load page content within 1,000ms.

Power Apps steers towards performant patterns

- The default Power Apps behaviour guides you towards well known performant patterns such as:
  - Streamlined data loading at launch
  - Automatic incremental paging of data
  - Caching of data for collections
  - Loading only essential data for each page
  - Consider when work can be done asynchronously using a cloud flow, etc.

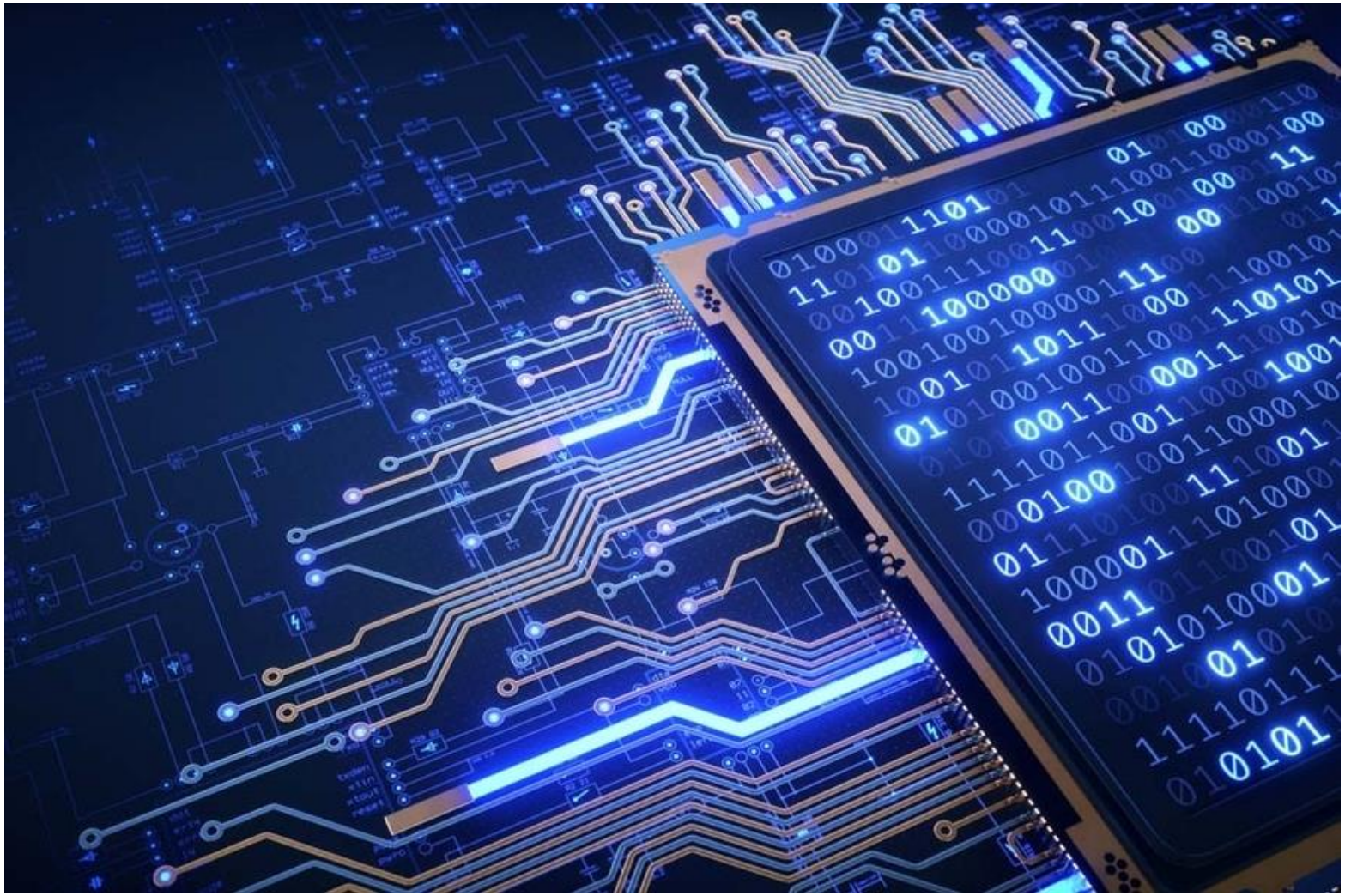
Falling into anti-patterns

- There's a risk of creating an app that performs poorly due to anti-patterns which can cause:
  - Slow loading times
  - Slow transitions between pages
  - Difficulty updating and retrieving data
- Some common examples of anti-patterns include:
  - Loading excessive amounts of data
  - Transforming everything into collections
  - Overloading the App OnStart property.

Four key performance design principles

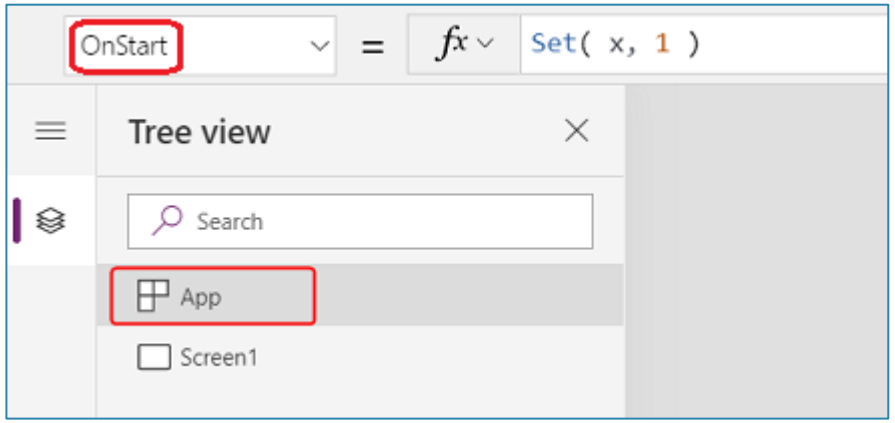
1. Optimise page loads
2. Small data payloads
3. Optimise query data patterns
4. Fast calculations.

Code optimisation



App OnStart property

- The OnStart property runs when the user starts the app

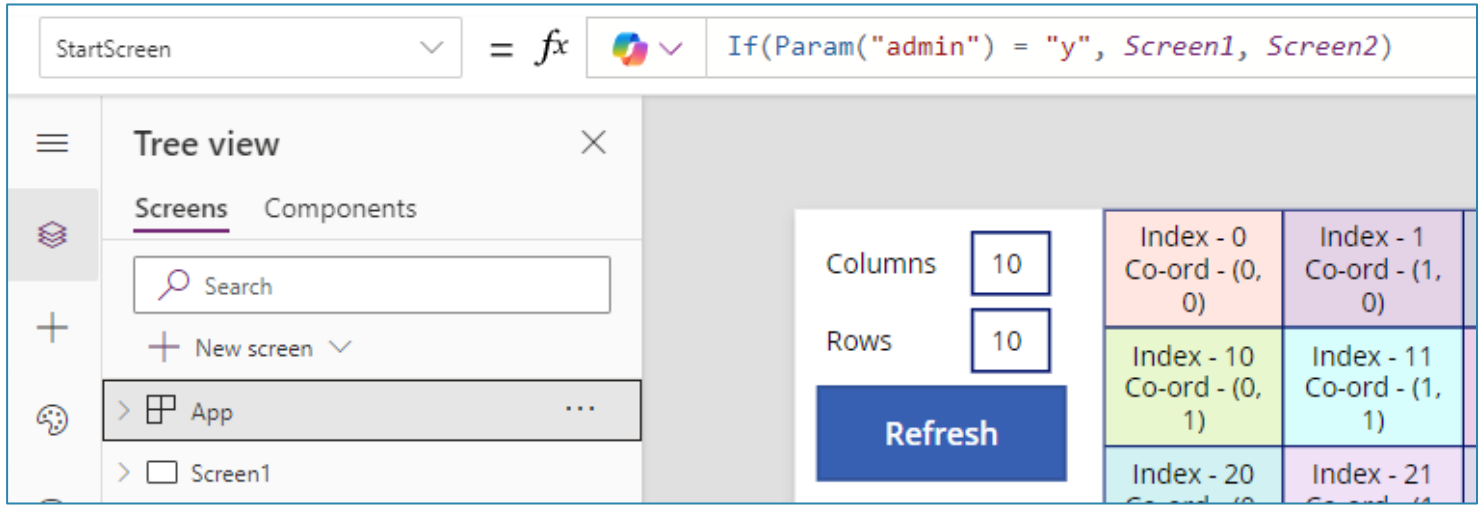


Note

The use of **OnStart** property can cause performance problems when loading an app. We're in the process of creating alternatives for the top two reasons for using property—caching data and setting up global variables. We've already created an alternative for defining the first screen to be shown with Navigate. Depending on your context, this property may be disabled by default. If you don't see it, and you need to use it, check the app's Advanced settings for a switch to enable it. The **OnVisible** property of a screen can also be used.

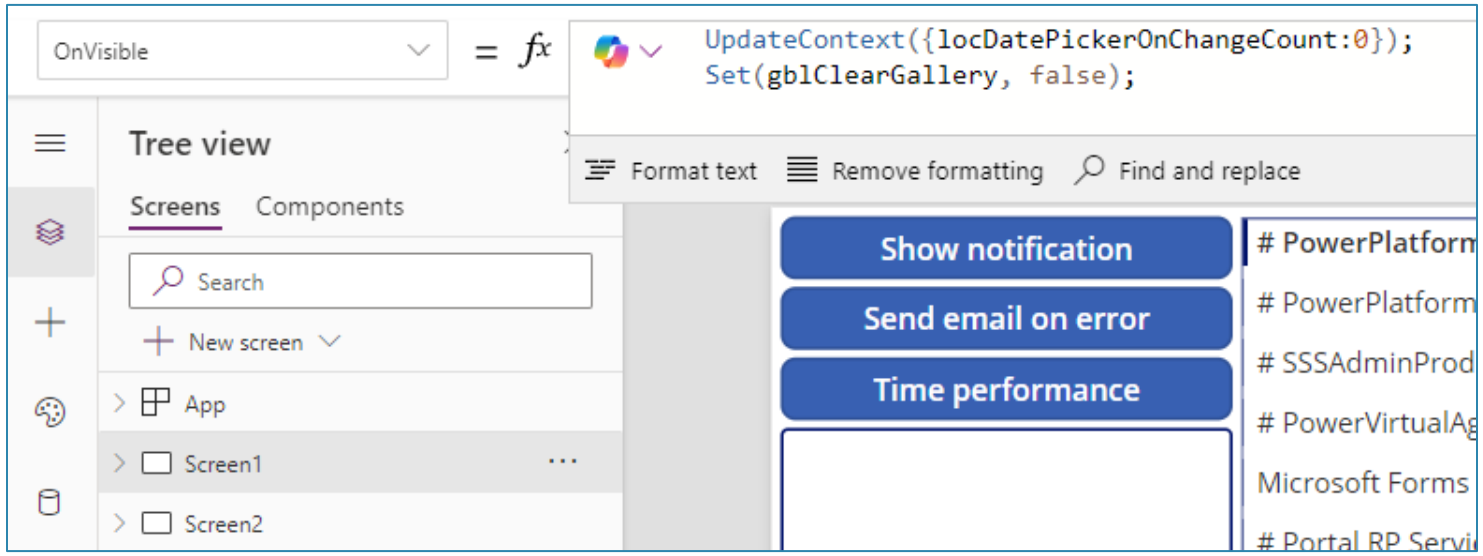
OnStart property alternative: App StartScreen property

- The App StartScreen property determines which screen will be displayed first
- You could navigate to a specific screen depending on a parameter
- You could display a custom loading screen



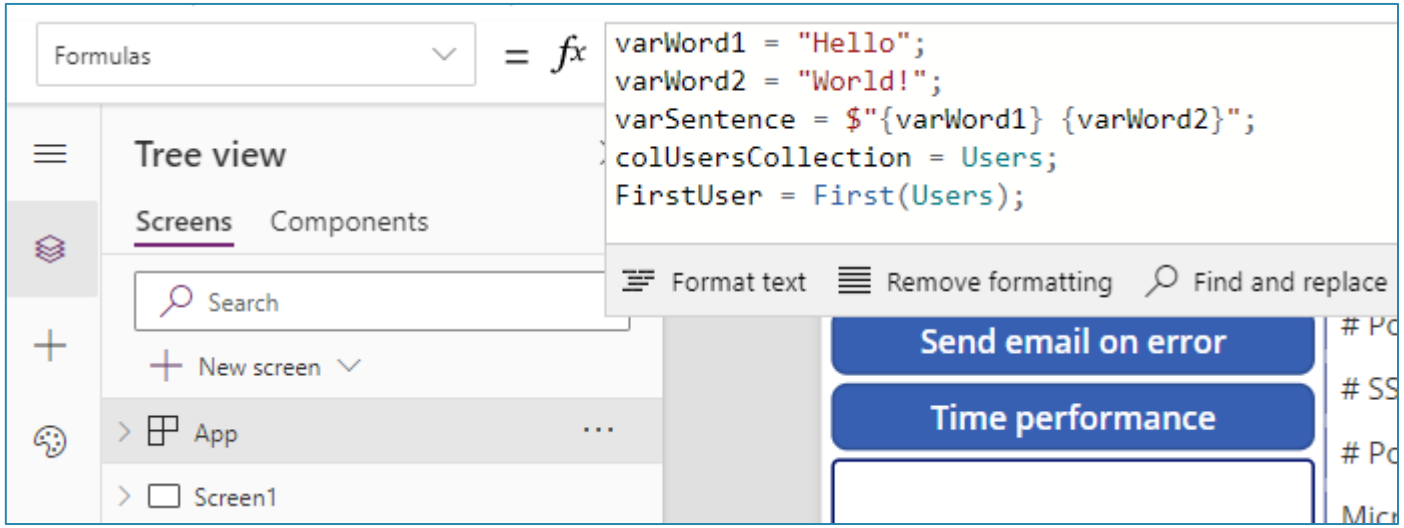
OnStart property alternative: Screen OnVisible property

- The behaviour of an app when the user navigates to a screen
- Use this property to set up variables and preload data used by the screen

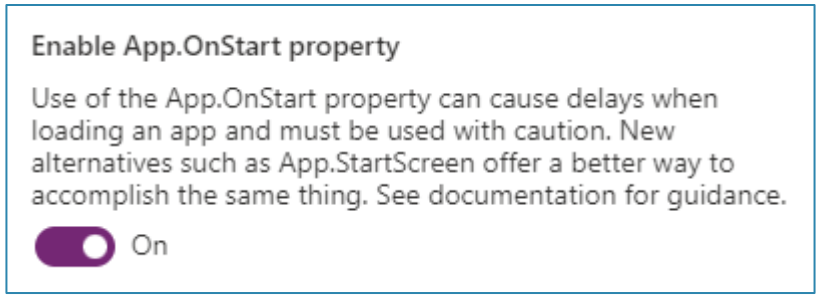


OnStart property alternative: Named formulas

- The formula's value is always available
- The formula's value is always up to date
- The formula's definition is immutable
- The formula's calculation can be deferred
- “Just in time” approach
- Also improves reuse

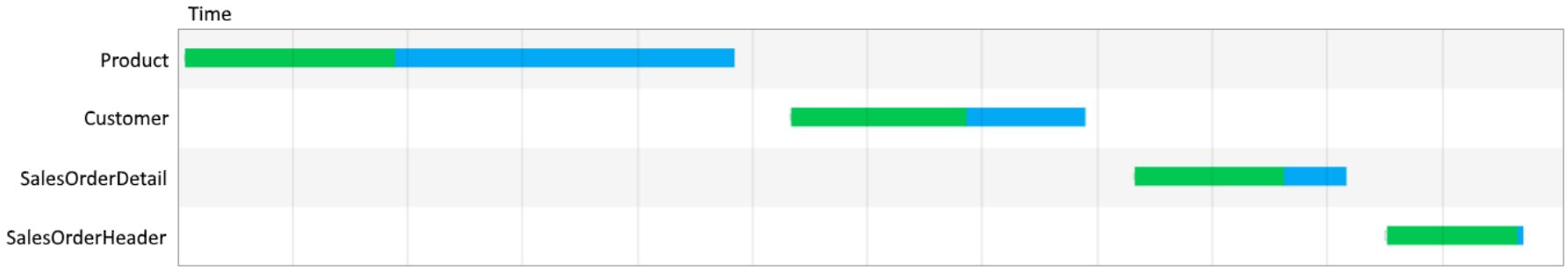


App OnStart property can be disabled



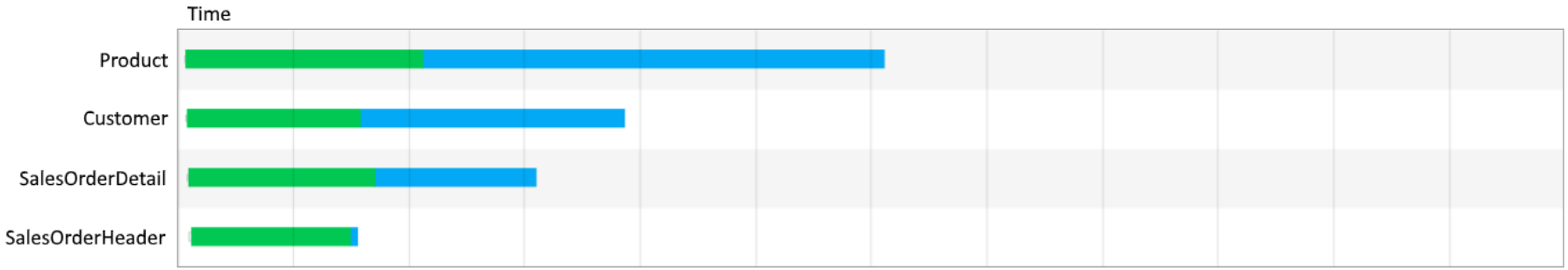
Concurrent function

```
ClearCollect( Product, '[SalesLT].[Product]' );
ClearCollect( Customer, '[SalesLT].[Customer]' );
ClearCollect( SalesOrderDetail, '[SalesLT].[SalesOrderDetail]' );
ClearCollect( SalesOrderHeader, '[SalesLT].[SalesOrderHeader]' );
```



😢

```
Concurrent(
   ClearCollect( Product, '[SalesLT].[Product]' ),
   ClearCollect( Customer, '[SalesLT].[Customer]' ),
   ClearCollect( SalesOrderDetail, '[SalesLT].[SalesOrderDetail]' ),
   ClearCollect( SalesOrderHeader, '[SalesLT].[SalesOrderHeader]' )
);
```



😊

Variable types

- Global variables:
  - App scope, can be referenced from anywhere in the app
  - Holds a number, text string, Boolean, record, table, etc
- Context variables:
  - Screen scope, can be referenced from only one screen
  - Great for passing values to a screen, much like parameters to a procedure in other languages
- Collections:
  - App scope, can be referenced from anywhere in the app
  - Holds a table

```
Set(gblName, "Oscar");
```

```
UpdateContext({locEnergyLevel:100});
```

```
ClearCollect(colAccounts, Accounts);
```

Variable types – *With* function records

- Formula block scope
- Records are cleared from memory when execution is complete!
- *With* functions can be nested.

```
With(
    {
        tmpTitle:LookUp(Users, 'Primary Email' = User().Email, Title)
    },
    Set(gblUserTitle, Title);
);
```

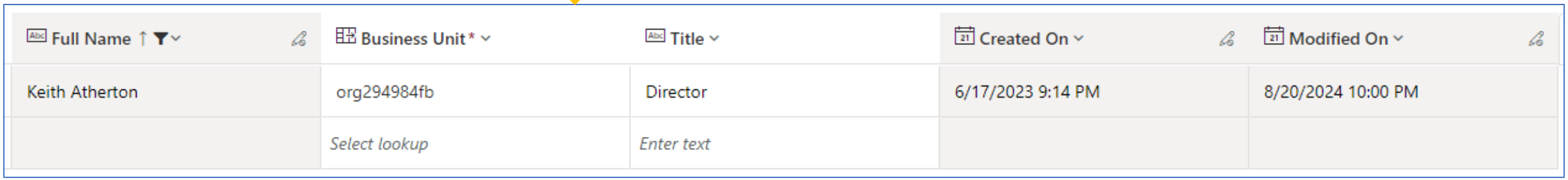
And another thing about those *With* function records…

- They can massively improve performance if the record values are referred to multiple times
- Can improve readability too.

```
// x3 data source calls.
With({
      Title:LookUp(Users, 'Primary Email' = User().Email, Title),
      FirstName:LookUp(Users, 'Primary Email' = User().Email, 'First Name'),
      LastName:LookUp(Users, 'Primary Email' = User().Email, 'Last Name')
},
   Set(gblTitle, Title);
   Set(gblFirstName, FirstName);
   Set(gblLastName, LastName);
);

// Only x1 data source call! 😊
With({
      tmpUserRecord:LookUp(Users, 'Primary Email' = User().Email)
   },
   Set(gblTitle, tmpUserRecord.Title);
   Set(gblFirstName, tmpUserRecord.'First Name');
   Set(gblLastName, tmpUserRecord.'Last Name');
);
```

Using the *LookUp* function to retrieve a record's value

```
// Get user title.
Set(gblUserRecord,
    LookUp(Users, 'Primary Email' = User().Email)
);

Set(gblUserTitle, gblUserRecord.Title);
```



Full Name	Business Unit *	Title	Created On	Modified On
Keith Atherton	org294984fb	Director	6/17/2023 9:14 PM	8/20/2024 10:00 PM
	Select lookup	*Enter text*		

Use the *LookUp* function *ReductionFormula* if needed

- *ReductionFormula* - Optional. This formula is evaluated over the record that was found, and then reduces the record to a single value
- You can reference columns within the table
- If you don't use this parameter, the function returns the full record from the table.

Syntax:
LookUp(Table*, *Formula* [, *ReductionFormula*])

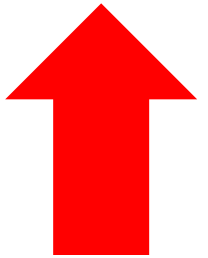
```
// Save user title to variable.
Set(gblUserTitle,
   LookUp(Users, 'Primary Email' = User().Email,
      Title
   )
);
```

😊

Retrieving table data

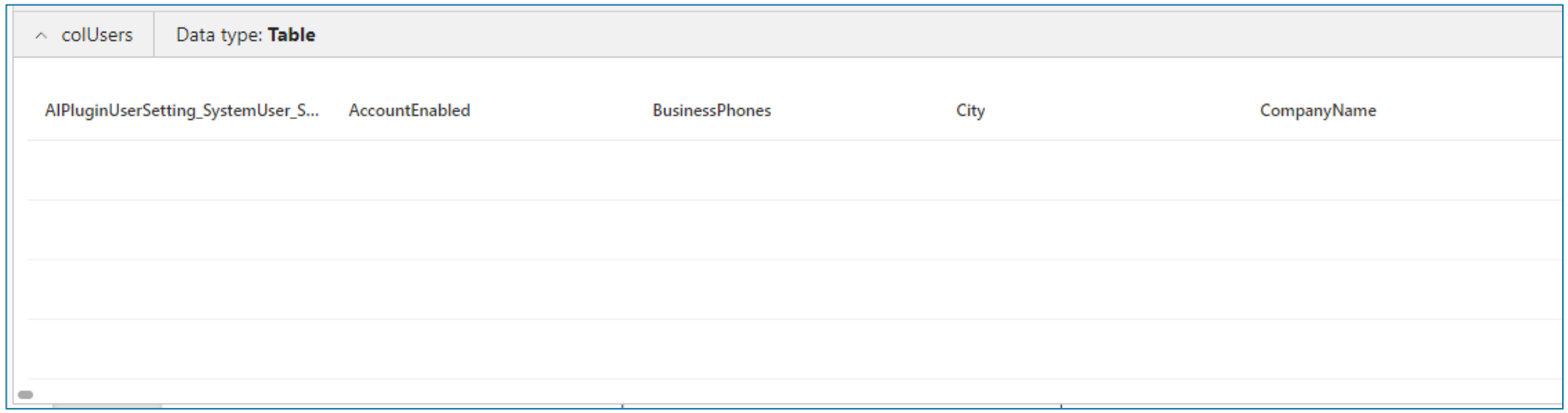
User columns and data

Full Name	Business Unit*	Primary Email*	+124 more
# AIBuilder_StructuredML_Prod_C...	org294984fb	AIBuilder_StructuredML_Prod_CDS@onmicrosoft.com	
# AIBuilderProd	org294984fb	AIBuilderProd@onmicrosoft.com	
# AppDeploymentOrchestration	org294984fb	AppDeploymentOrchestration@onmicrosoft.com	
# AriaMdlExporter	org294984fb	AriaMdlExporter@onmicrosoft.com	
# BAP	org294984fb	BAP@onmicrosoft.com	
# BizQA	org294984fb	BizQA@onmicrosoft.com	
# CatalogServiceEur	org294984fb	CatalogServiceEur@onmicrosoft.com	
# CCADataAnalyticsML	org294984fb	CCADataAnalyticsML@onmicrosoft.com	
# CDSAcisInfraAppGlobal	org294984fb	CDSAcisInfraAppGlobal@onmicrosoft.com	



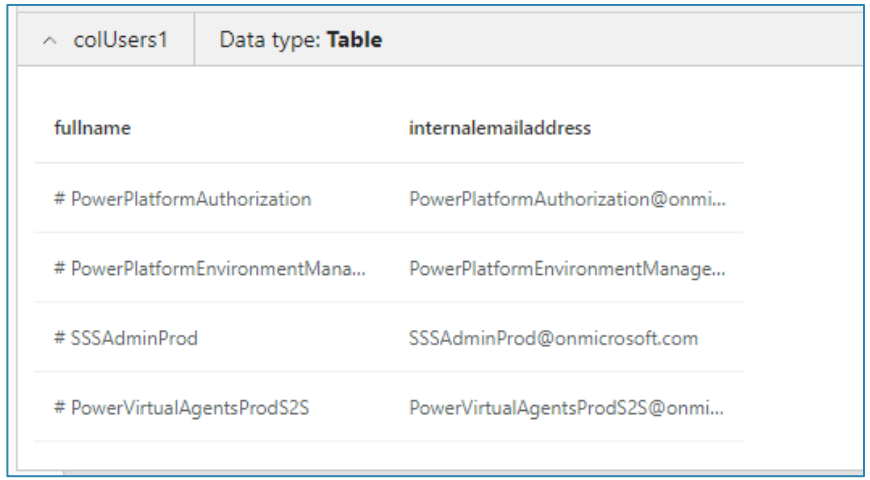
Retrieve just the table columns you need

```
ClearCollect(colUsers,
    Users
);
```



etc…

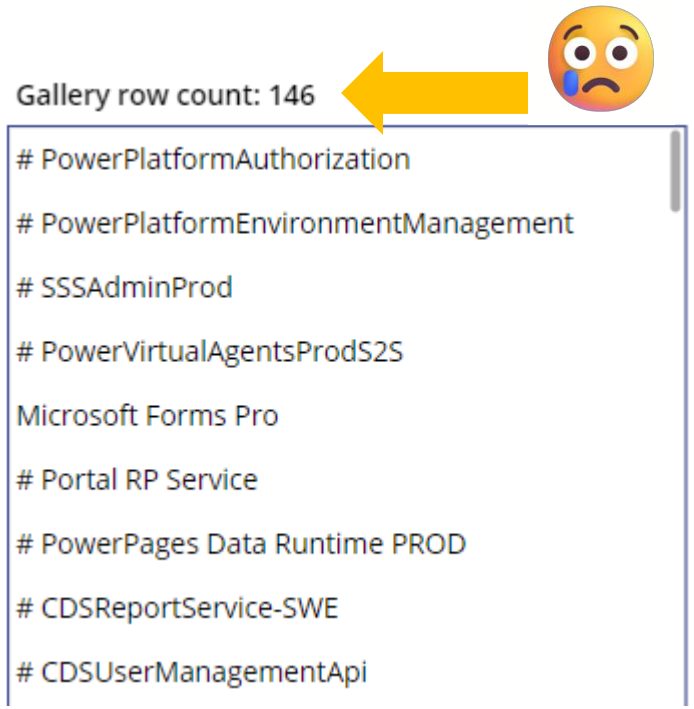
```
ClearCollect(colUsers1,
    ShowColumns(Users,
        'Full Name',
        'Primary Email'
    )
);
```



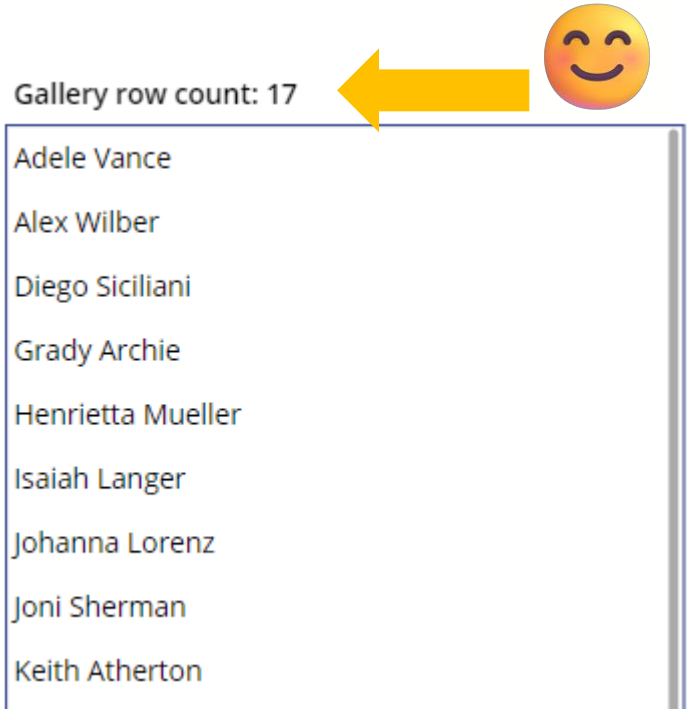
Note: This can be mitigated by Explicit column selection (ECS) which automatically computes which columns are necessary to retrieve based on their usage in controls (for example, galleries and forms.)

Retrieve just the table rows you need

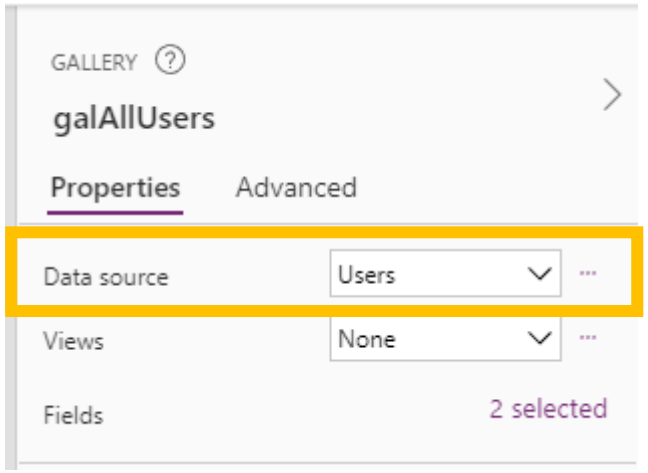
```
ClearCollect(colUsers,
    Users
);
```




```
ClearCollect(colUsers,
    Filter(Users,
        City = “Edinburgh”)
);
```

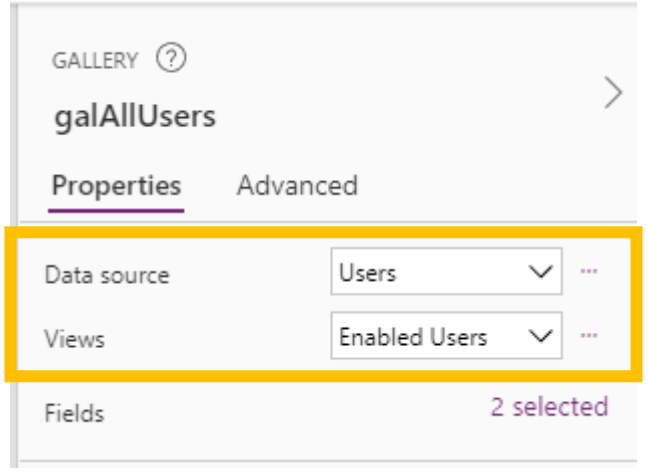


Consider Dataverse table views



Gallery row count: 146

PowerPlatformAuthorization
PowerPlatformEnvironmentManagement
SSSAdminProd
PowerVirtualAgentsProdS2S
Microsoft Forms Pro
Portal RP Service
PowerPages Data Runtime PROD
CDSReportService-SWE
CDSUserManagementApi



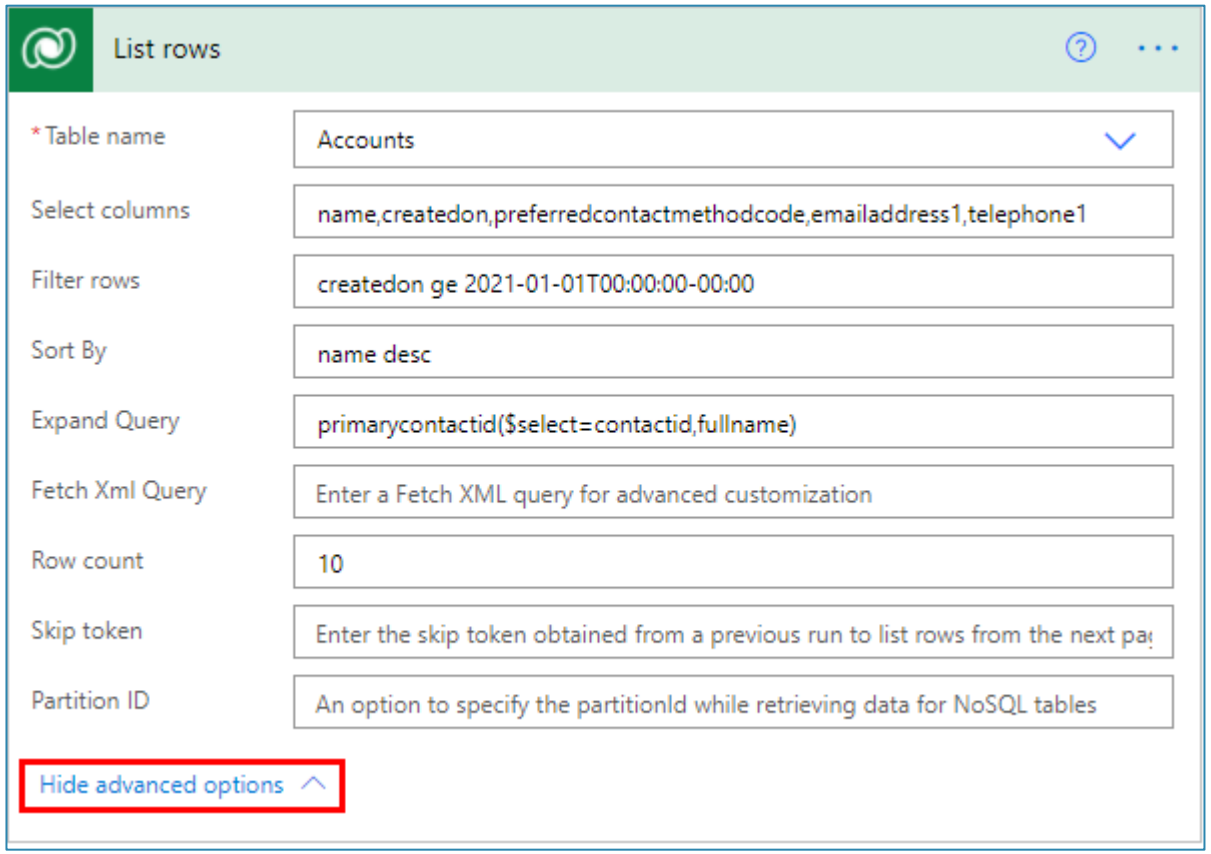
Gallery row count: 17

Adele Vance
Alex Wilber
Diego Siciliani
Grady Archie
Henrietta Mueller
Isaiah Langer
Johanna Lorenz
Joni Sherman
Keith Atherton

ℹ️ Note: SQL Server, Oracle and many other relational database management systems offer *views* too!

You could use Power Automate to populate a collection

- It offloads work away from the app…
- …but there's approximately a 0.6-second performance cost to instantiate Power Automate
- Power Automate must be independently launched each time it's called and have memory allocated.



Cache data where possible

- Retrieve then cache (store data in memory) using variables and collections where possible
- Referencing cached data is much faster than retrieving it from the data source
- Consider:
  - How often does that data change?
  - Is real-time data needed?

```
// Save user record to a variable.
Set(gblUser,
   LookUp(Users, 'Primary Email' = User().Email)
);

// Save user title to a variable.
Set(gblUserTitle,
   LookUp(Users, 'Primary Email' = User().Email,
      Title
   )
);

// Save users to a collection.
ClearCollect(colUsers,
   Users
);
```

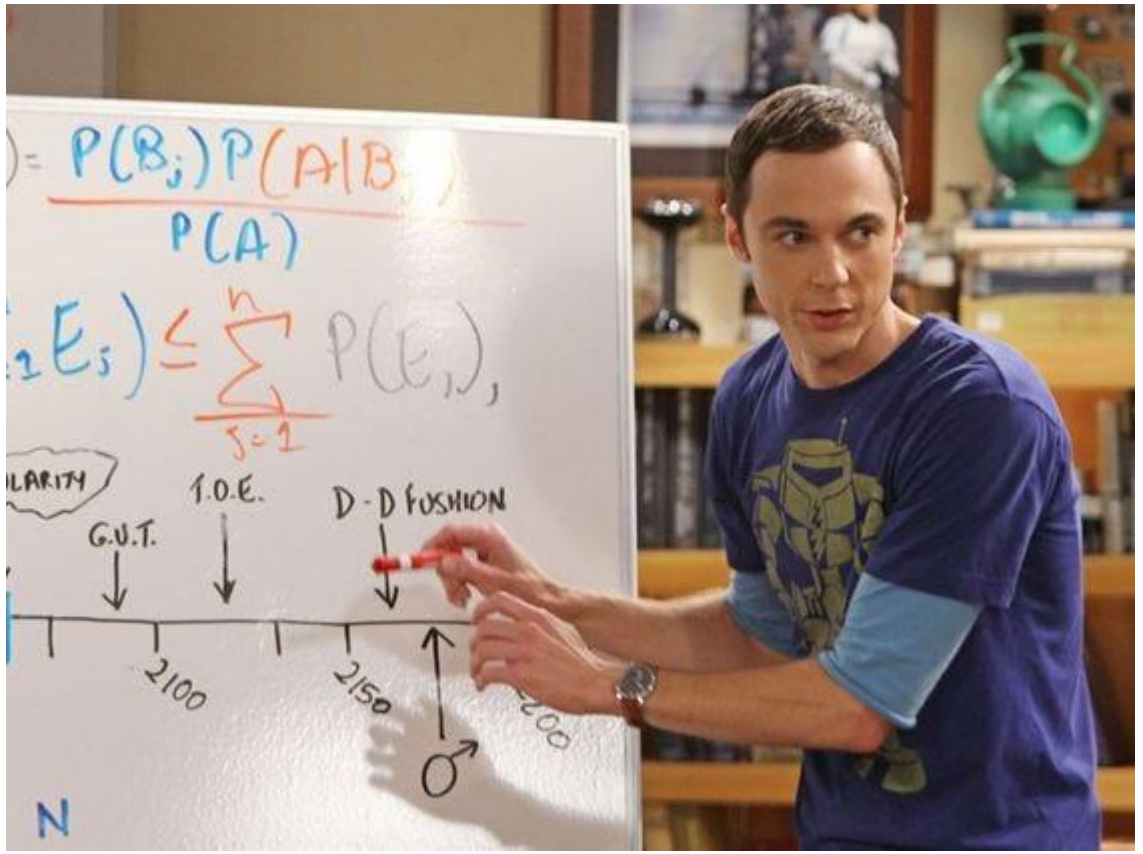
Patch records fast

```
// Patch each record one-at-a-time.
ForAll(
    colAccounts,
    Patch(
        Accounts,
        Defaults(Accounts),
        {
            'Account Name':"Test Account",
            'Account Number':"123"
        }
    )
);
```

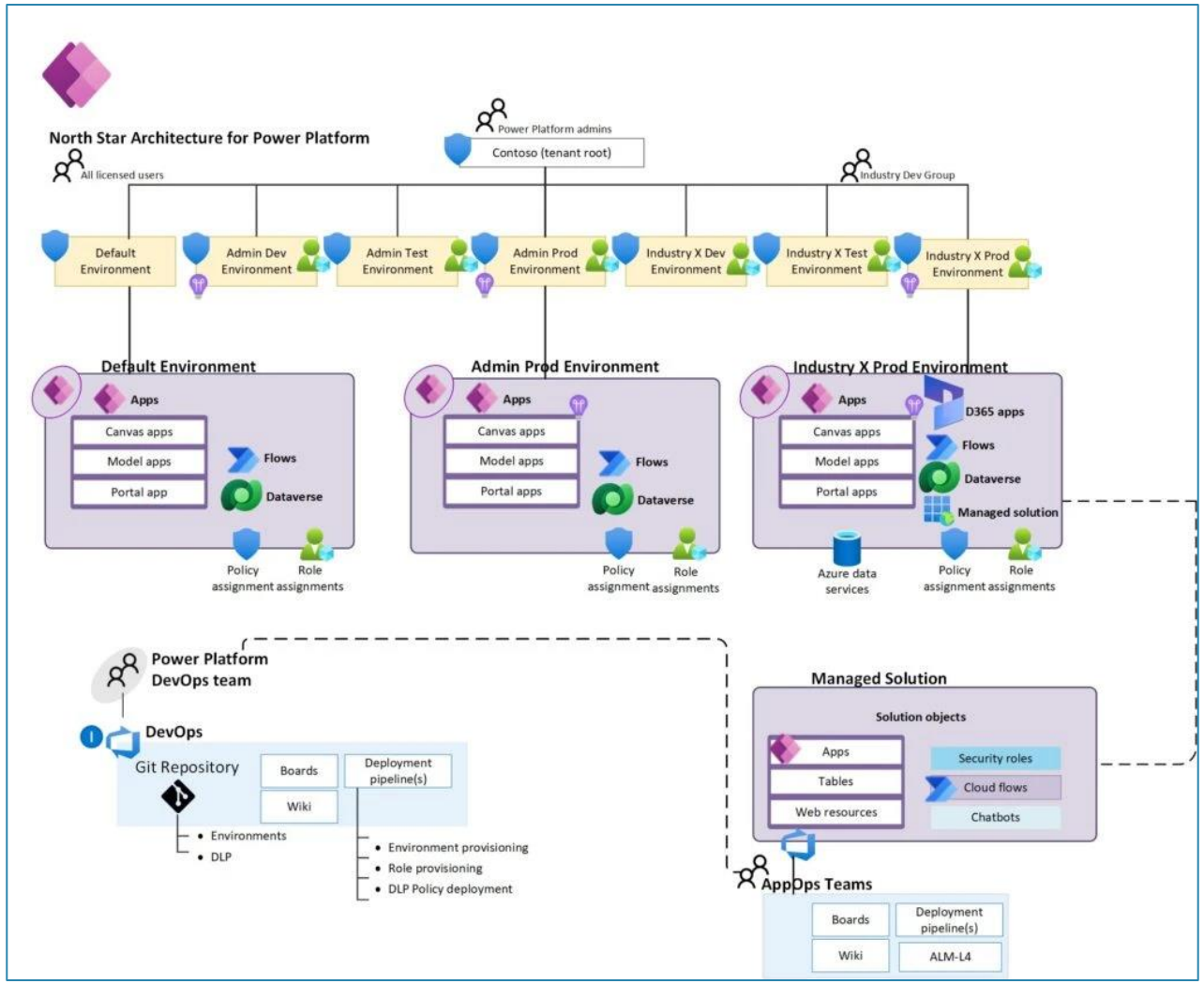
```
// Patch all records simultaneously using schema match.
Patch(
    Accounts,
    colAccounts
);
```

Avoid over-engineered designs and other tips

- Avoid cross-screen dependencies such as referencing a control from another screen
- Avoid galleries nested in other galleries
- Call Gallery *AllItemsCount*
- Consider screen transitions
- Avoid overusing timers
- Consider data source refreshes
- Use the Power Apps offline feature if required
- Publish your app regularly to get the latest features and performance upgrades.

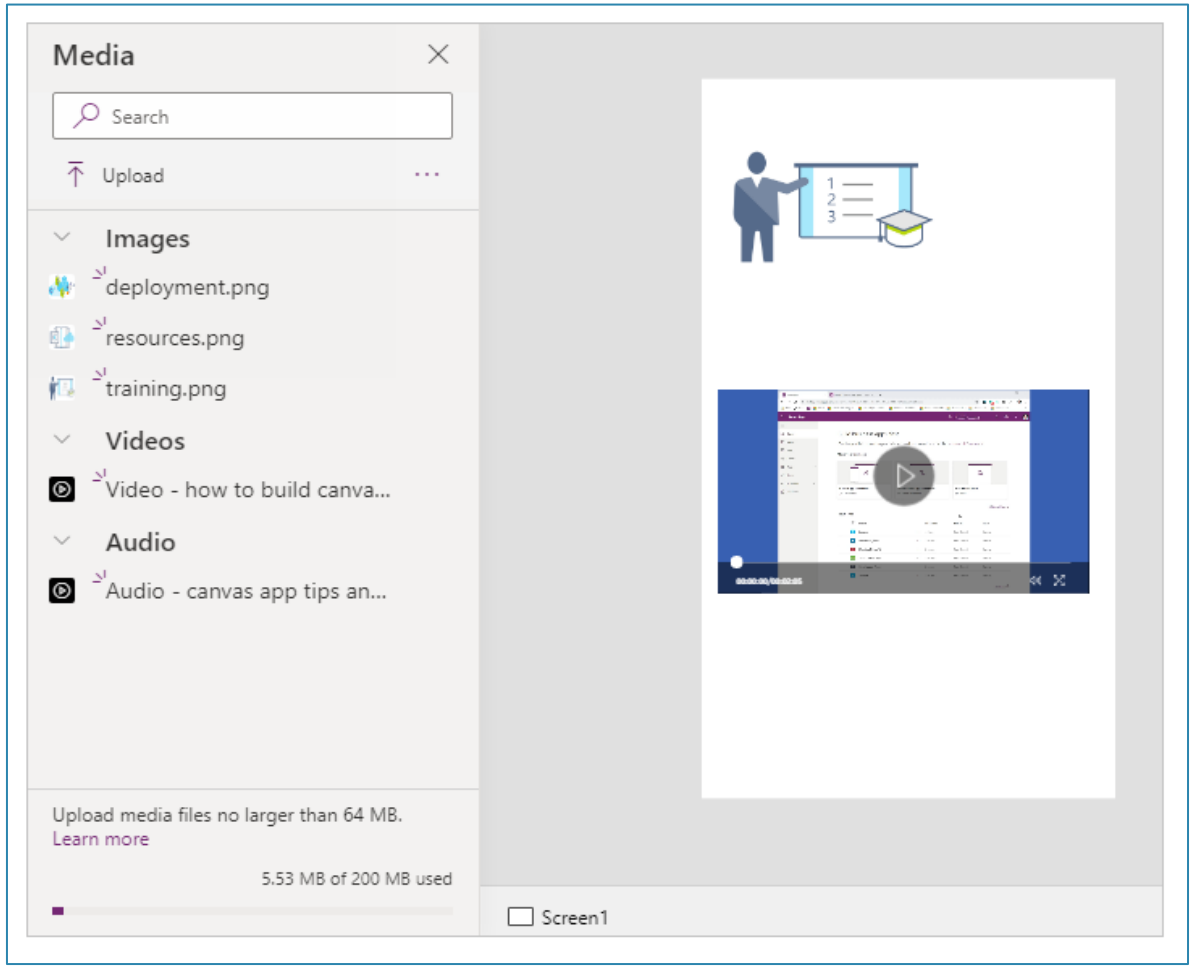


Architectural optimisation



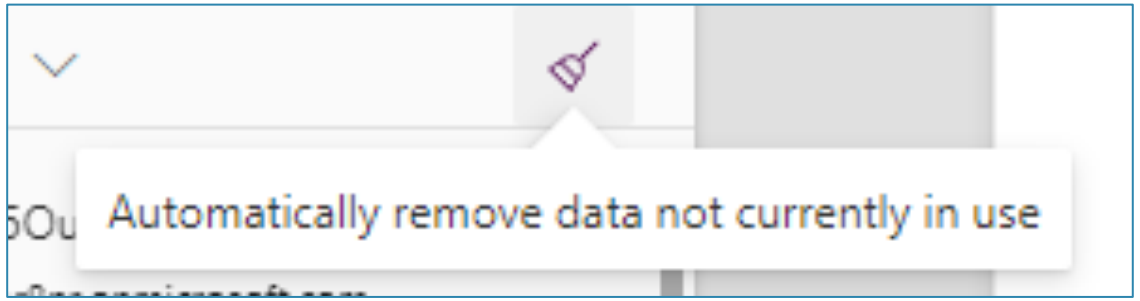
Consider uploaded media

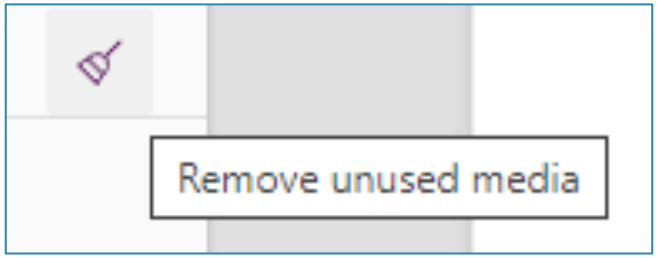
- You can add images, audio & video files to a canvas app
- Size restrictions:
  - Total size of all media files uploaded to an app can't exceed 200 MB
  - Maximum size of an individual media file in an app can't exceed 64 MB
- SVGs are typically lighter than PNG, JPGs, etc
- Use Dataverse image thumbnails where appropriate.

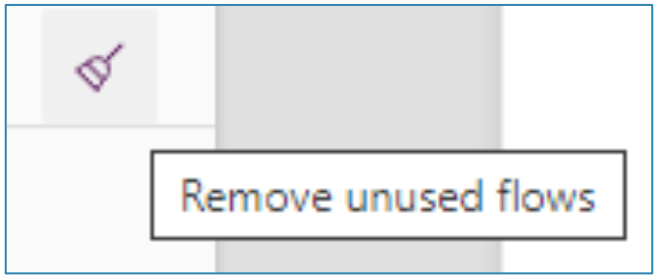


Remove unused elements

- Automatically remove data not currently in use
- Remove unused media
- Remove unused flows

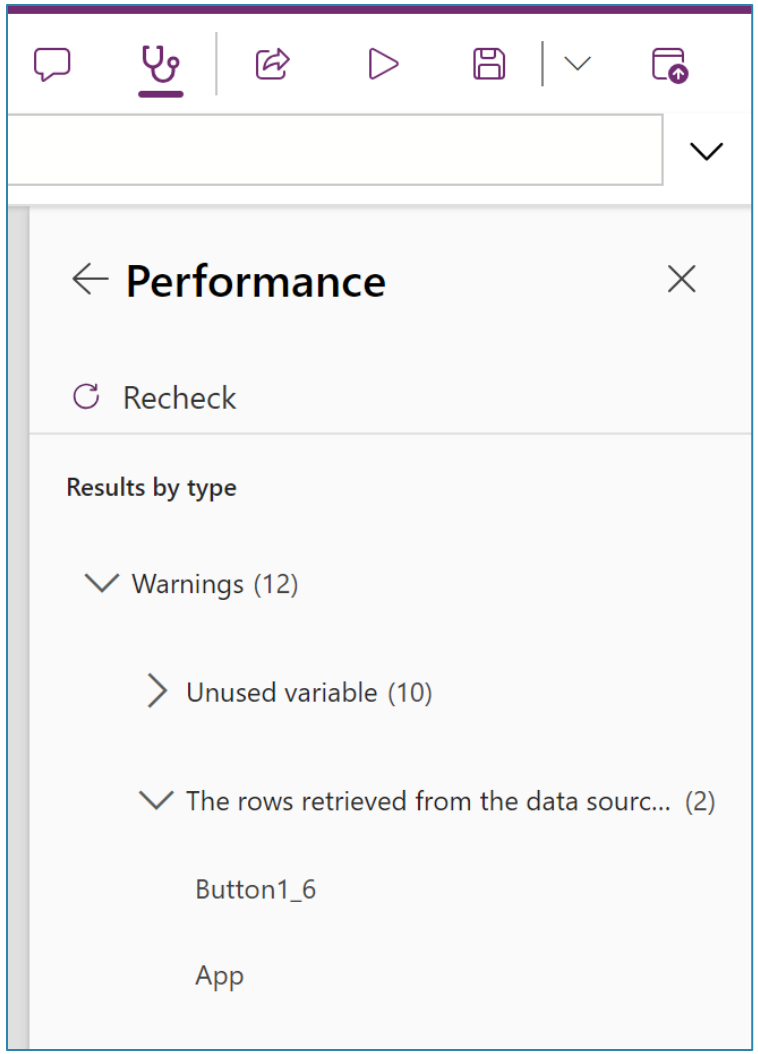


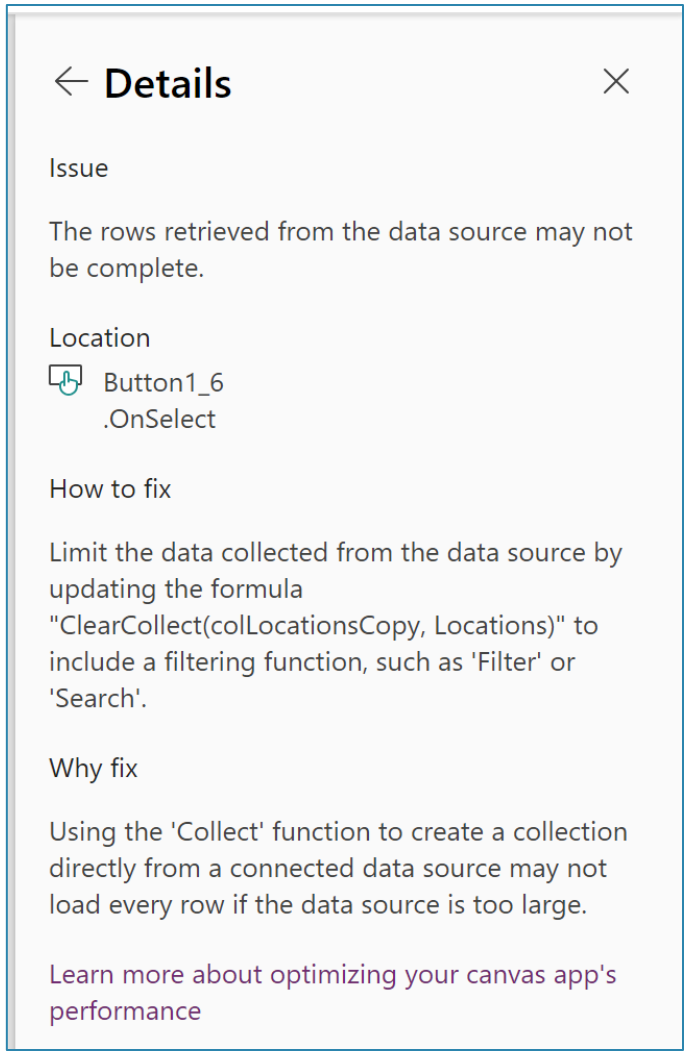




Use App Checker

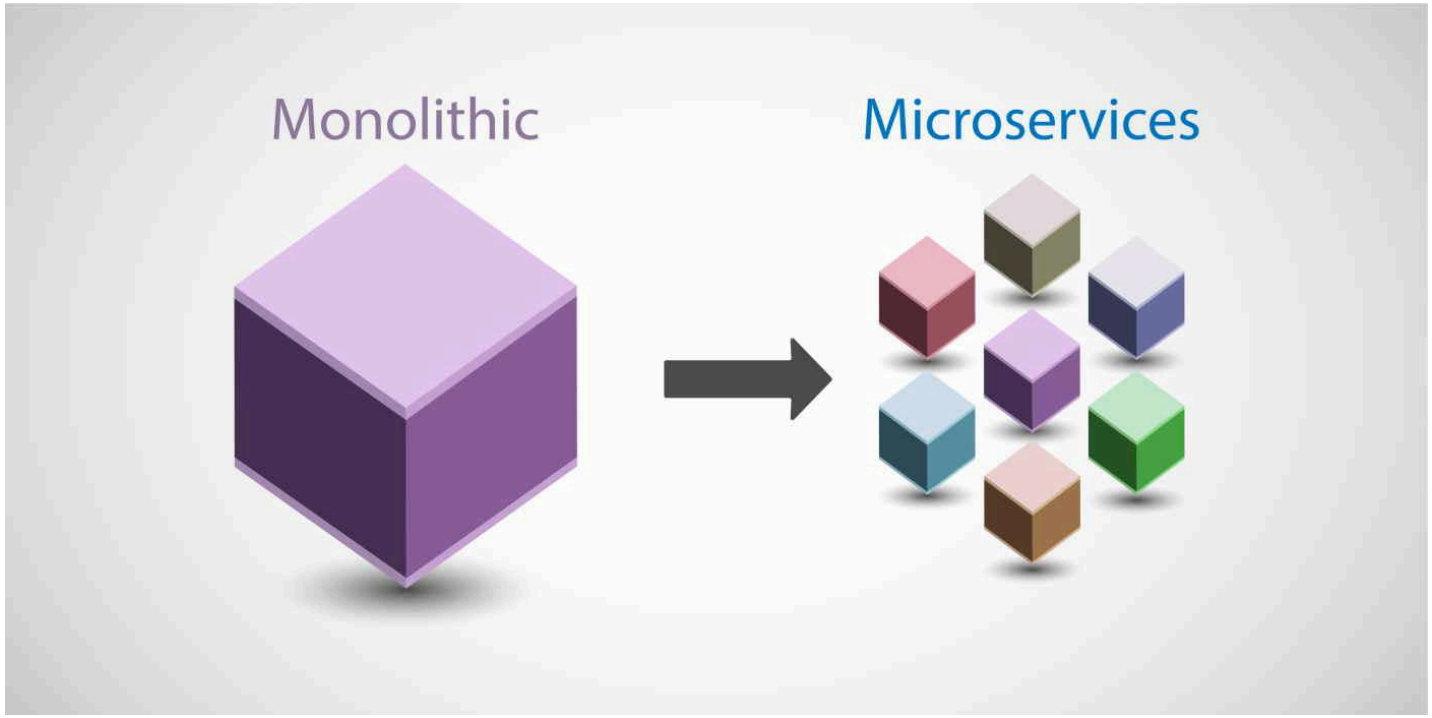
- App Checker highlights errors and warnings for the following and offers fixes:
  - Formulas
  - Runtime
  - Accessibility
  - Performance
  - Data source



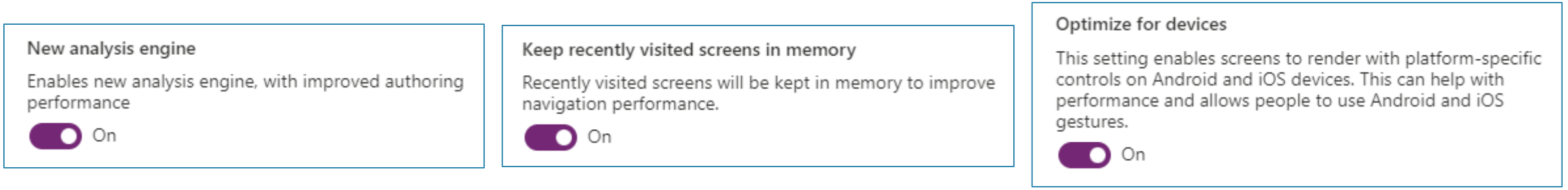


Monolithic design

- Consider splitting a large “monolith” app into multiple smaller apps
- It can be better having a smaller app which does one thing very well
- An app can launch other apps using the *Launch* function and passing parameters if required.

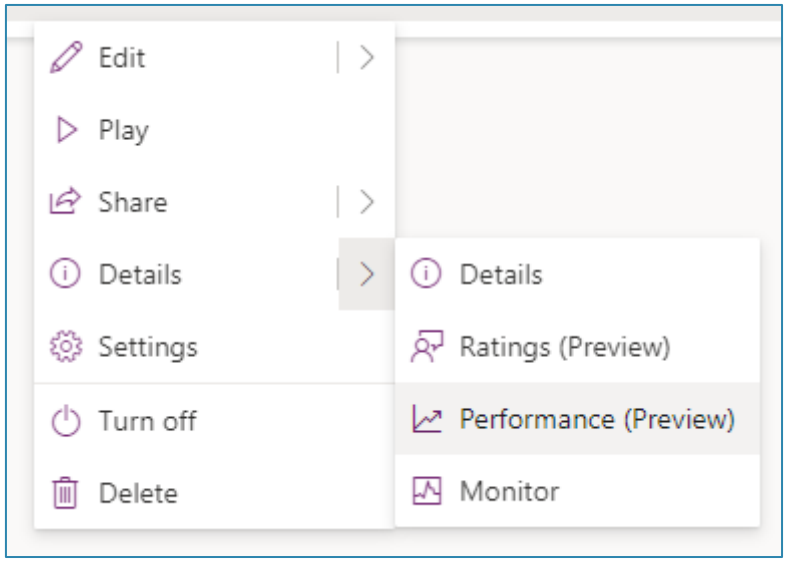


Useful Canvas App settings



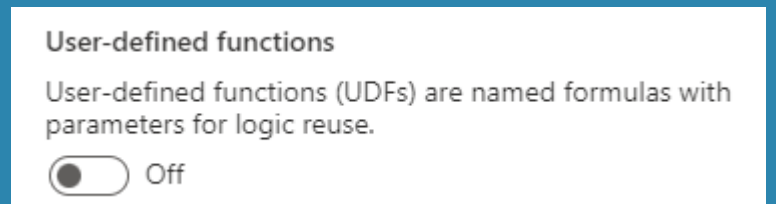
Model-driven app considerations

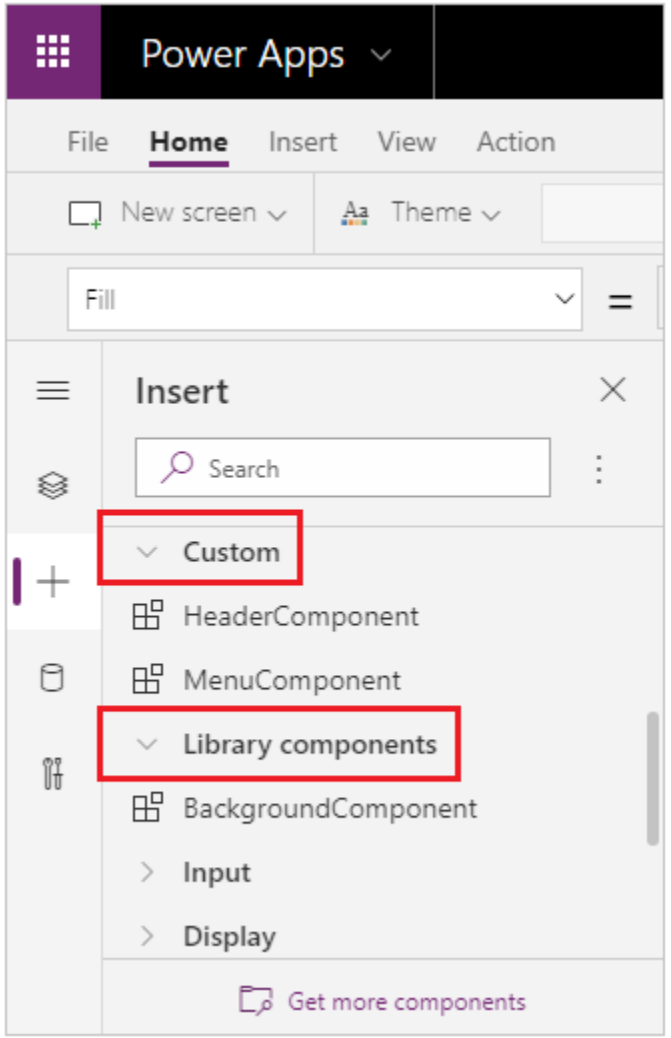
- Form default tab controls have initialisation logic invoked on load
- Data-driven controls can affect loading speed such as:
  - Quick view form
  - Subgrid
  - Timeline
- For JavaScript, use asynchronous network requests when requesting data
- Solution checker analyses client and server customizations for performance or reliability issues
- Performance insights (preview) 👉
- Object checker (preview).



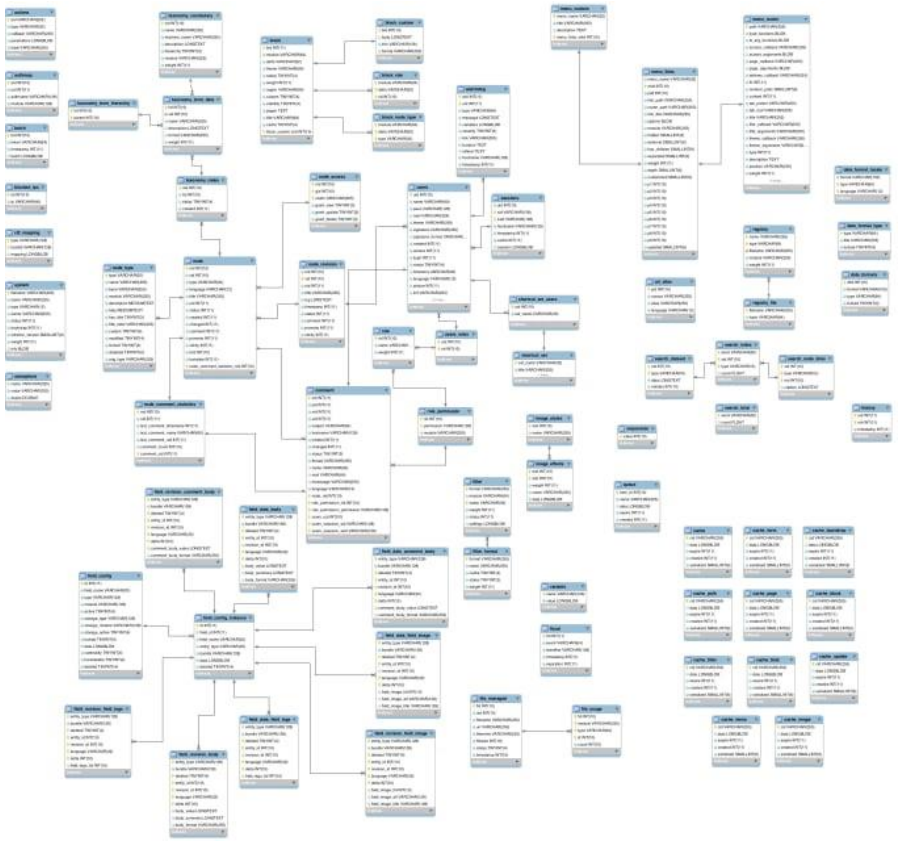
Development optimisation

- DRY = *Don't Repeat Yourself*
- Consider reusable items:
  - Named formulas
  - Components 👉
  - Component libraries
  - User-defined functions (experimental) 👇
  - Low-code plugins (preview).

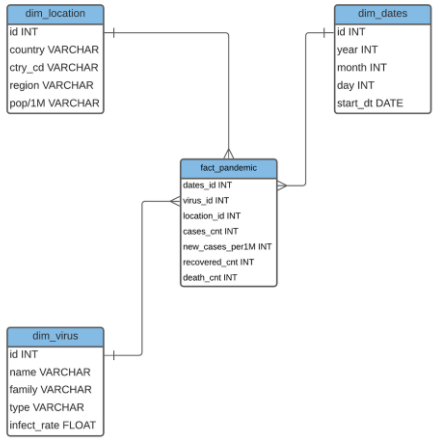




Beware of Dataverse data model over-normalisation



What I thought I needed…



What I *actually* needed

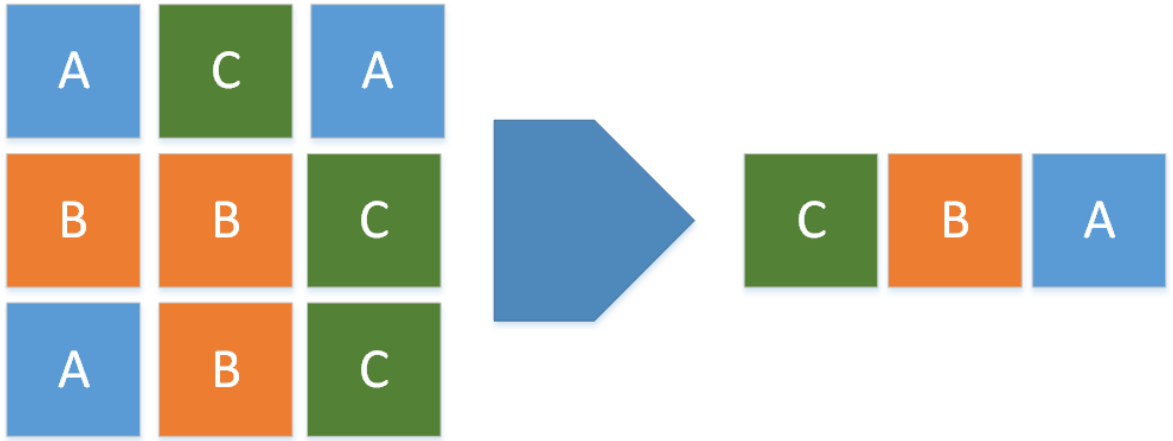
Data model normalisation

- Data normalization means that you store the data only once, and avoid duplication, this helps to keep the data consistent and accurate
- From Microsoft Learn: *"However, sometimes you need to duplicate some data to make the queries faster and easier"* 😮
- You need to balance these two goals in your app design and table structure

- Related to this: Avoid too many lookups on a gallery
- This anti-pattern is known as "N squared, (n^2)" or an "N+1" problem
- Consider de-normalisation or views.

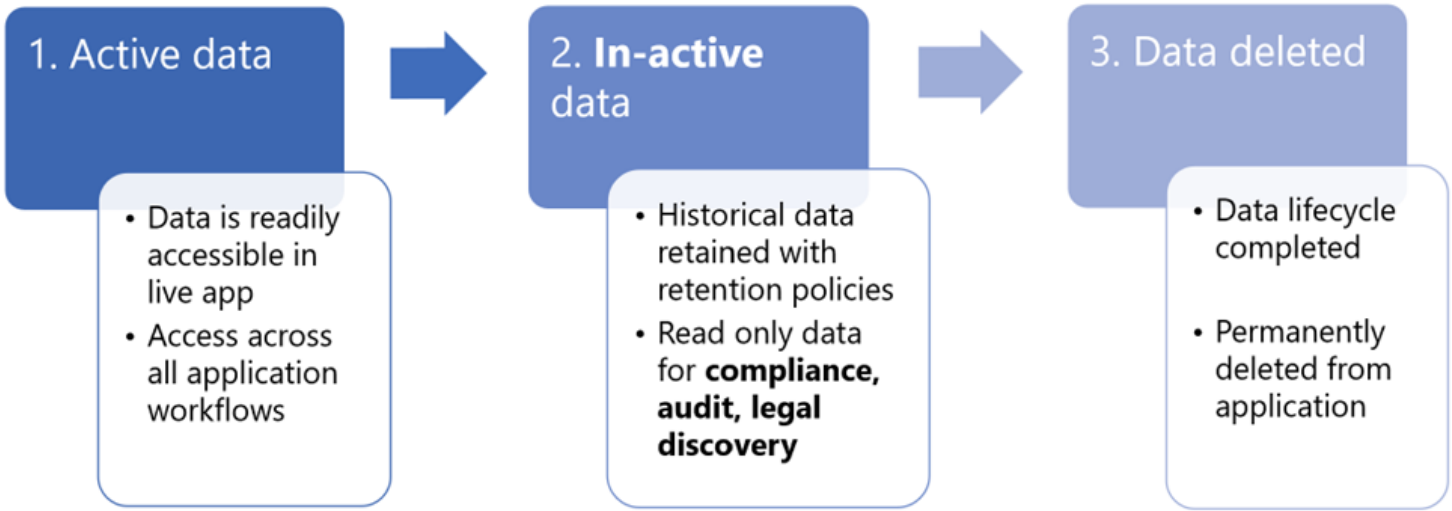
Consider duplicating data

- Sometimes data is slow to access if it's stored in a different location or format
- To make the query faster, you can copy the slow data and store it locally in a table that is fast and easy to query
- Run another process to update the local data periodically such as a Power Automate cloud flow, data flows, plugins, or SQL Server stored procedures
- Note: However, this means that the local data may not be the most updated version of the original data.



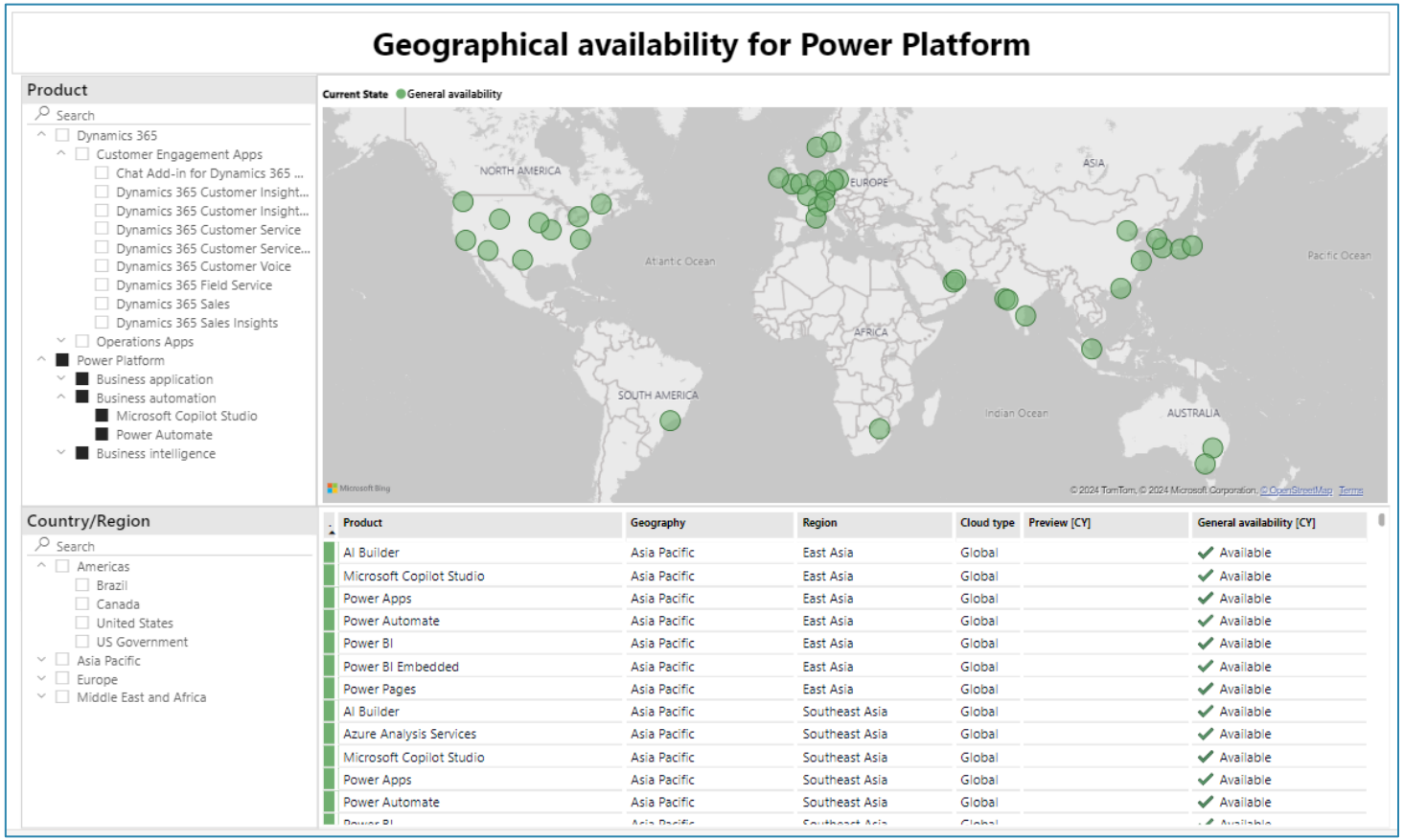
Dataverse long term data retention

- Consider moving inactive data to the Dataverse long term retention store
- Securely retain the historical application data long term for audit, legal, and regulatory requirements
- Note: Requires a Managed Environment.



Are your users close to your data?

- Consider the region for new environments
- Consider where any on-premises data gateways are located
- Consider where other data sources are located, e.g. Azure SQL Database.



Quiz time! Which canvas app data source is fastest?

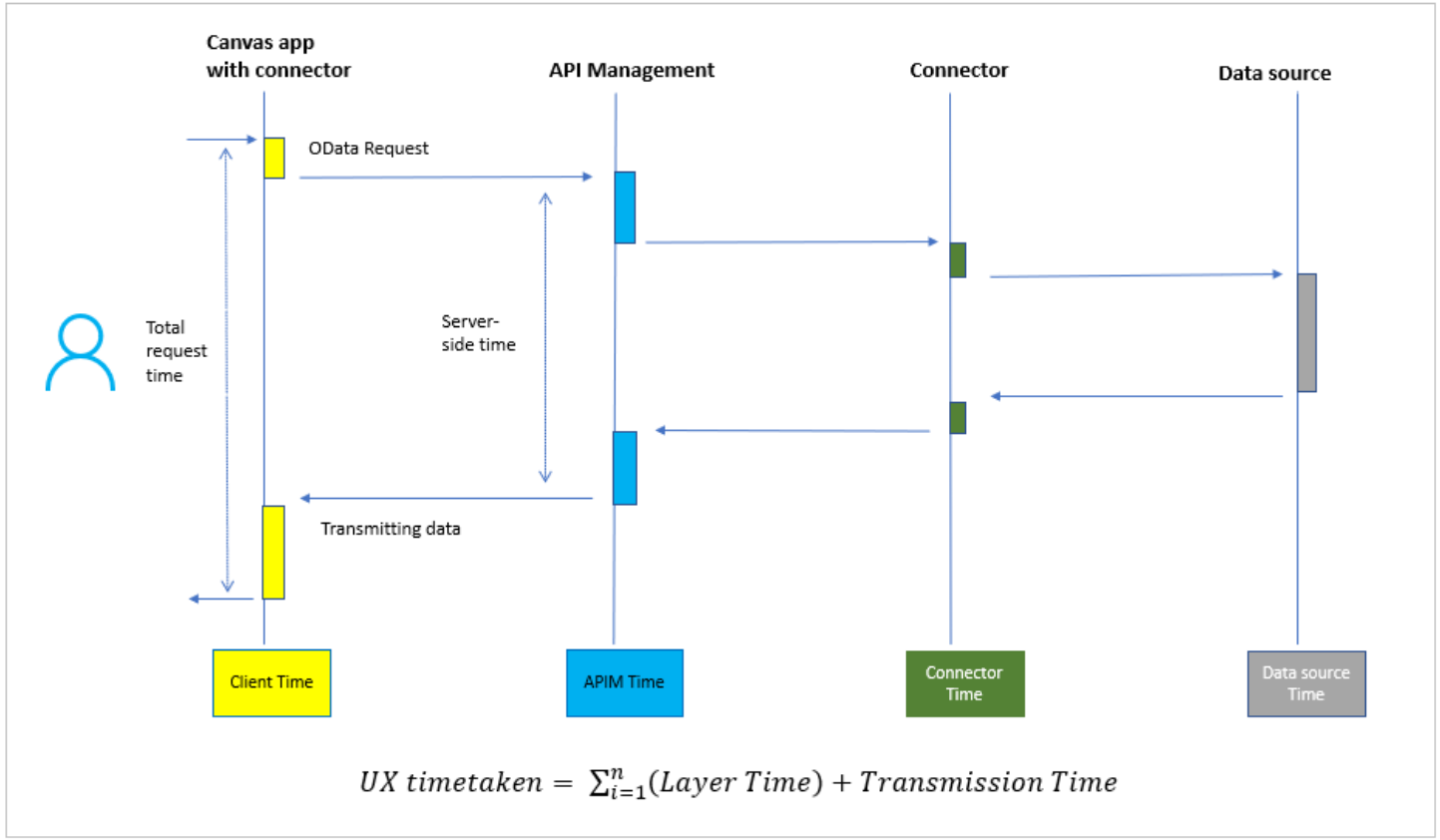
Online data source
E.g. Azure SQL Database

On-premises data source
E.g. SQL Server

Dataverse

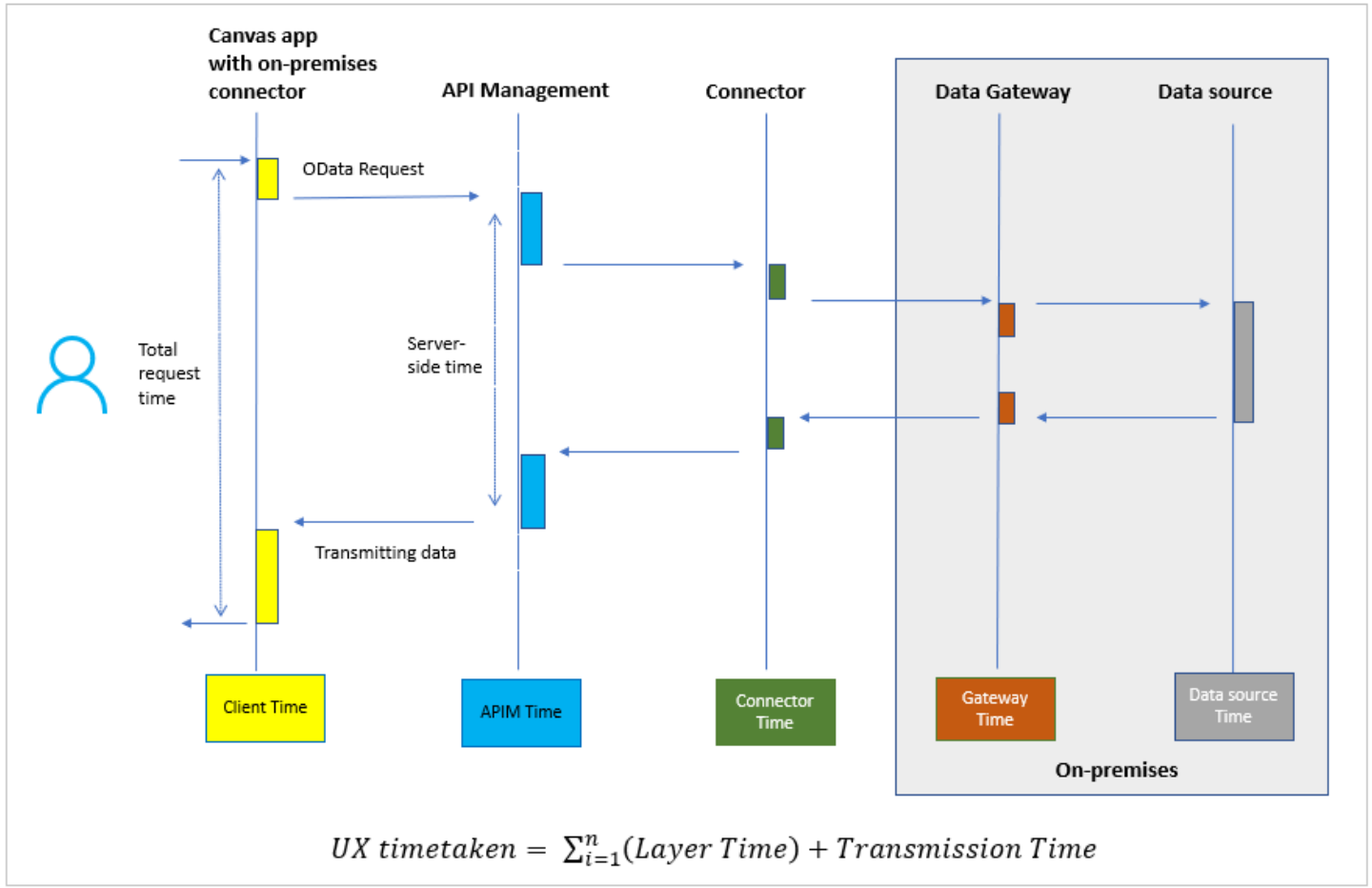
Online data source data call flow

- 2 layers to the data source
- Each layer can perform quickly or encounter some overhead while processing the request



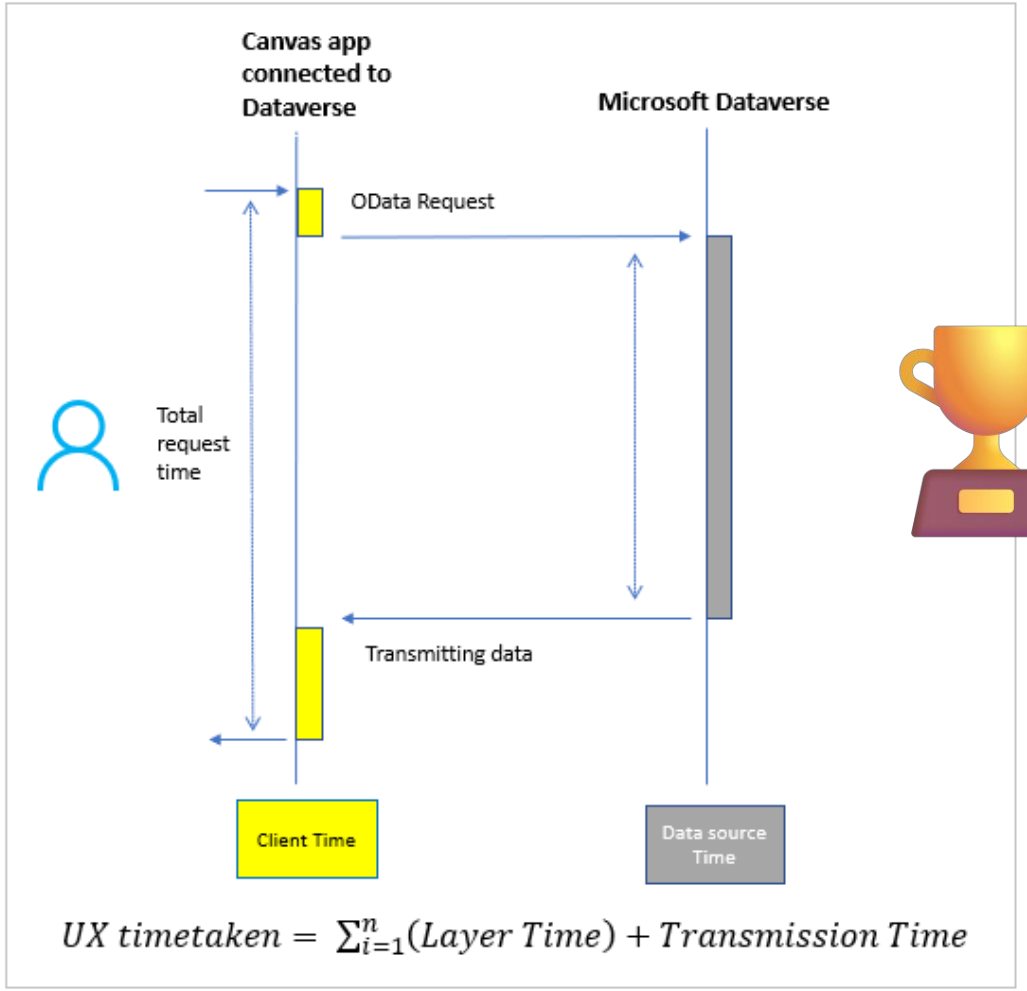
On-premises data gateway data call flow

- 3 layers to the data source!
- You need to have another layer called an *on-premises data gateway*
- The location and the specification of the data gateway also affect the performance of data calls



Microsoft Dataverse data call flow

- Data requests go to the environment instance directly – without passing through Azure API Management (APIM)
- Because of this, the performance of data calls is faster compared to the rest of the data sources! 😎



?

Implement delegation where possible

- Delegation is where a Power Fx query is performed at the remote data source rather than in the client app
- Only the required results are returned to the Power App
- Delegation is supported for data sources such as Dataverse, SharePoint, SQL Server and Salesforce
- Note: Power Apps is limited to the first 2,000 records from the data source.

Power Apps delegable functions and operations for Dataverse

These Power Apps operations, for a given data type, might be delegated to Dataverse for processing (rather than processing locally within Power Apps).

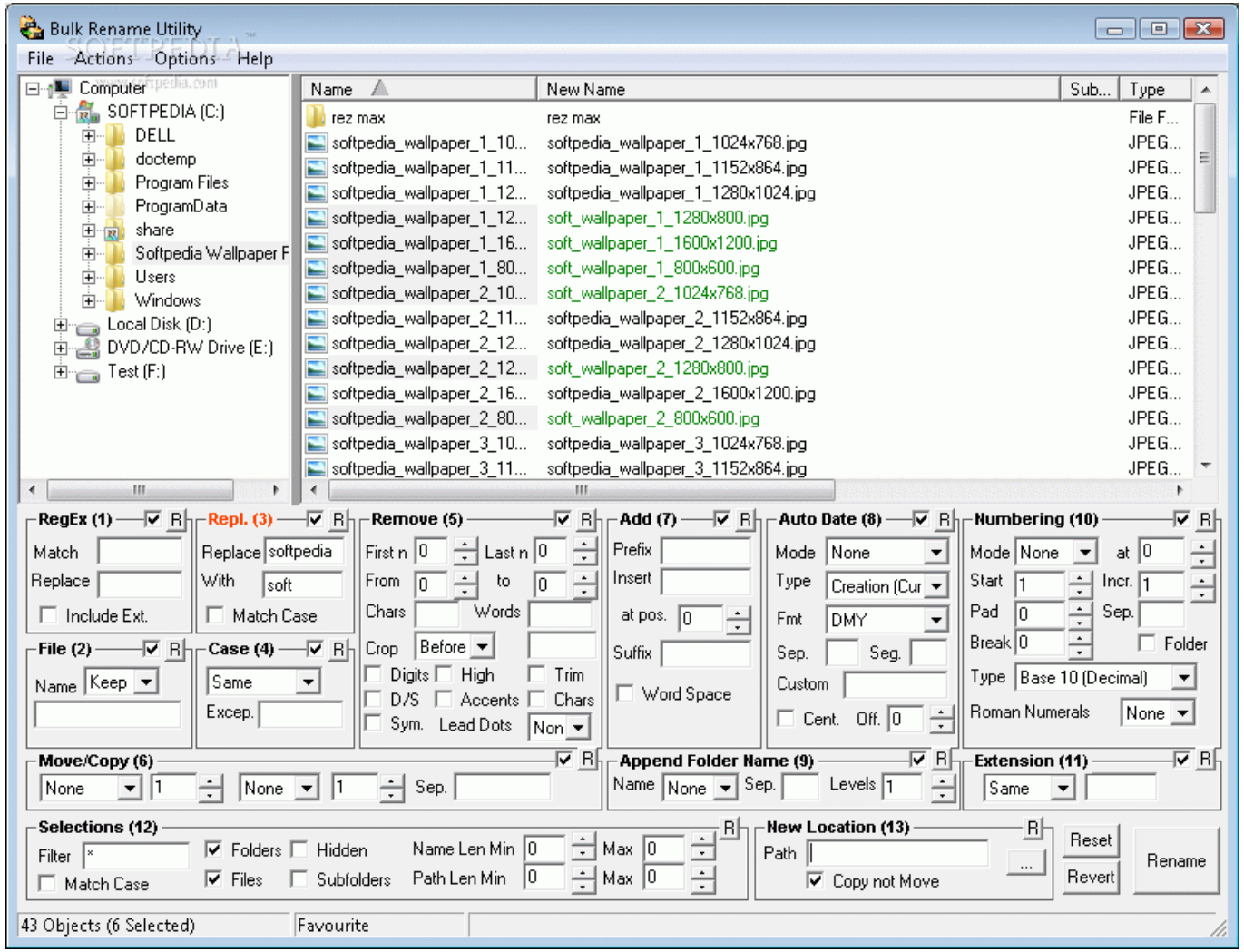
Expand table

Item	Number [1]	Text [2]	Choice	DateTime [3]	Guid
<, <=, >, >=	Yes	Yes	No	Yes	-
=, <>	Yes	Yes	Yes	Yes	Yes
And/Or/Not	Yes	Yes	Yes	Yes	Yes
CountRows [4] [5], CountIf [6]	Yes	Yes	Yes	Yes	Yes
Filter	Yes	Yes	Yes	Yes	Yes
First [7]	Yes	Yes	Yes	Yes	Yes
In (membership) [8]	Yes	Yes	Yes	Yes	Yes
In (substring)	-	Yes	-	-	-
IsBlank [9]	Yes	Yes	No	Yes	Yes
Lookup	Yes	Yes	Yes	Yes	Yes
Search	No	Yes	No	No	-
Sort	Yes	Yes	Yes	Yes	-
SortByColumns	Yes	Yes	Yes	Yes	-
StartsWith	-	Yes	-	-	-
Sum, Min, Max, Avg [6]	Yes	-	-	No	-

User experience (UX)

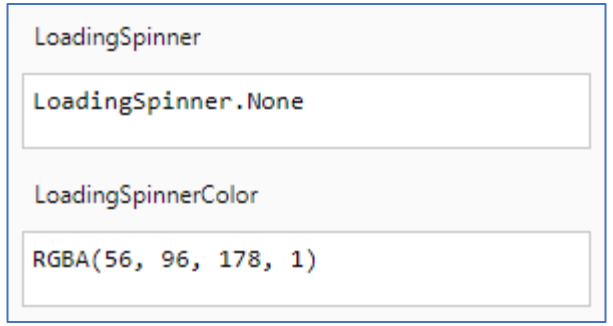
Busy user interface (UI) anyone?

- Controls require memory, therefore lots of controls need lots of memory
- Consider the gallery control for repeating controls such as grids of data, menus, etc
- Consider splitting busy screens into multiple screens.

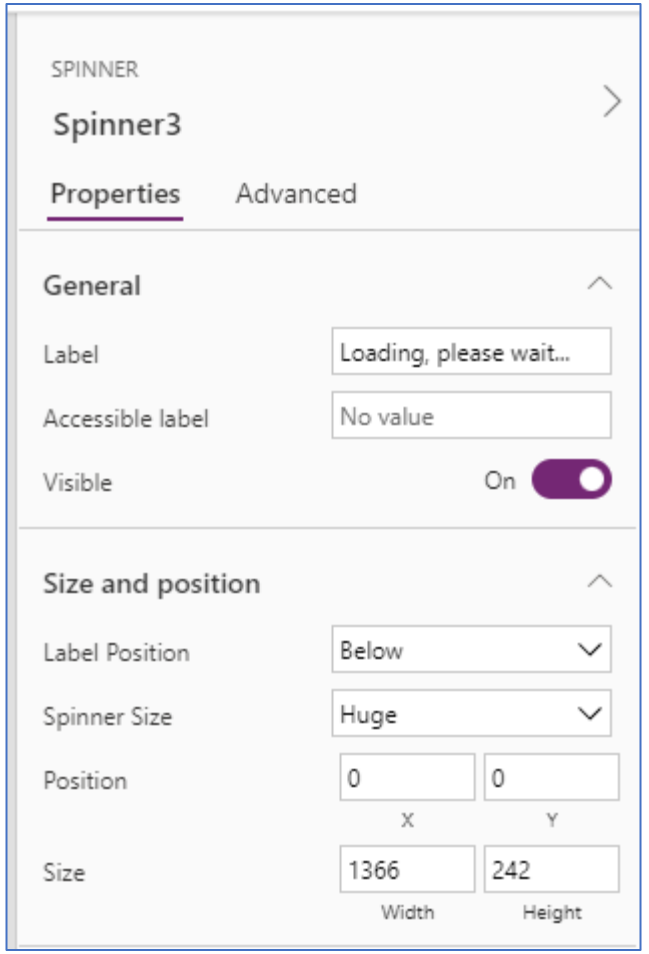


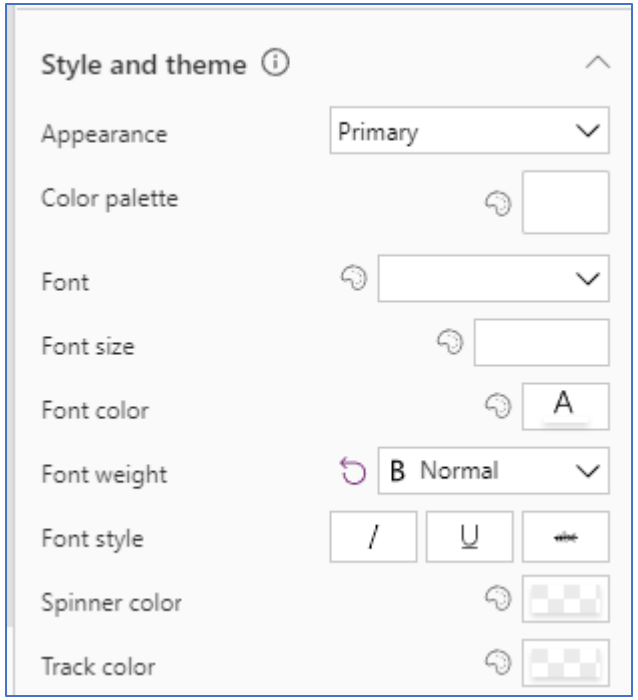
User feedback: Out of the box spinner

- Screens and galleries have the following properties:
  - LoadingSpinner:
    - None
    - Controls
    - Data
  - LoadingSpinnerColor
- Or you could use your own animated gif



User feedback: Modern controls spinner

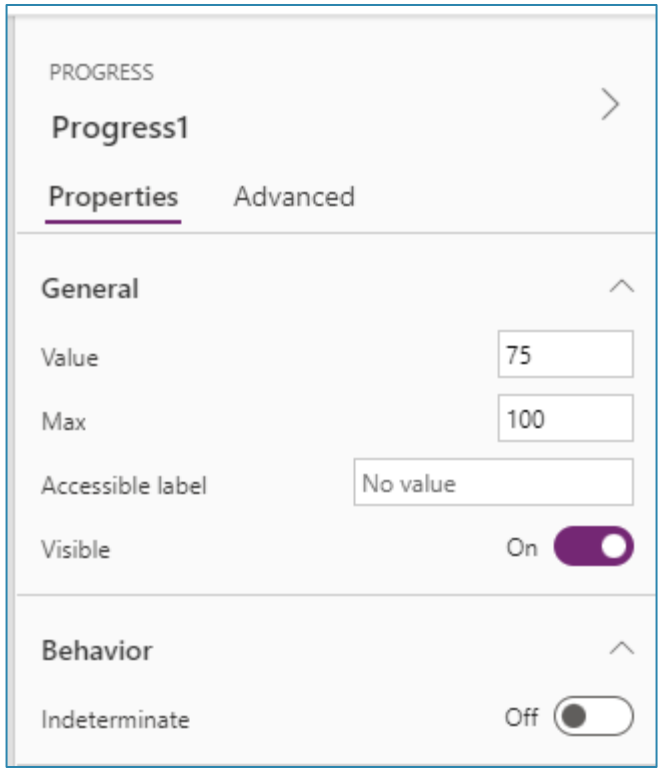


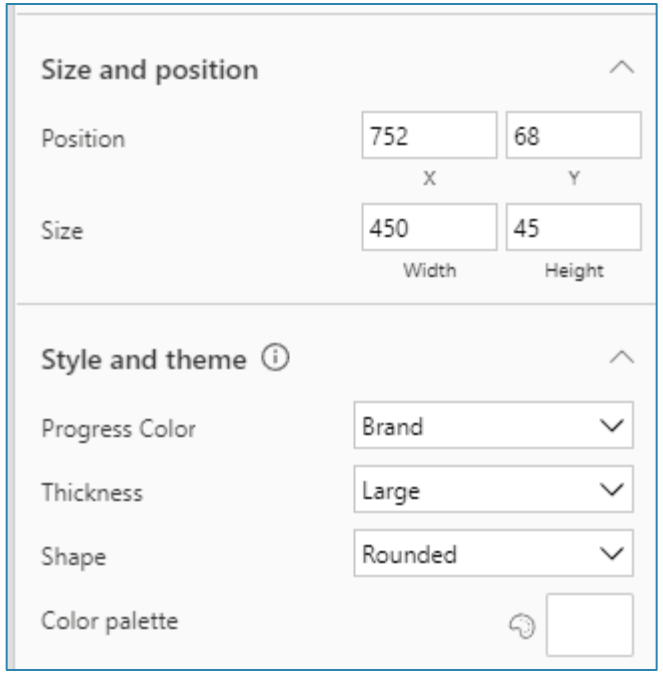


User feedback: Modern controls progress bar

Determinate:

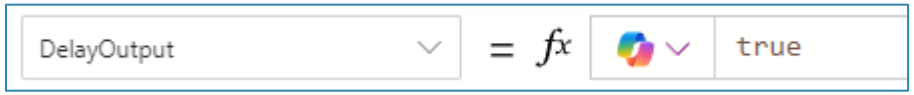
Indeterminate:





Text Input *DelayOutput* property

- When set to true, user input is registered after half a second delay
- Useful for delaying expensive operations until user completes inputting text rather than for each keystroke
- For example, when input is used in other formulas such as searching within a gallery.

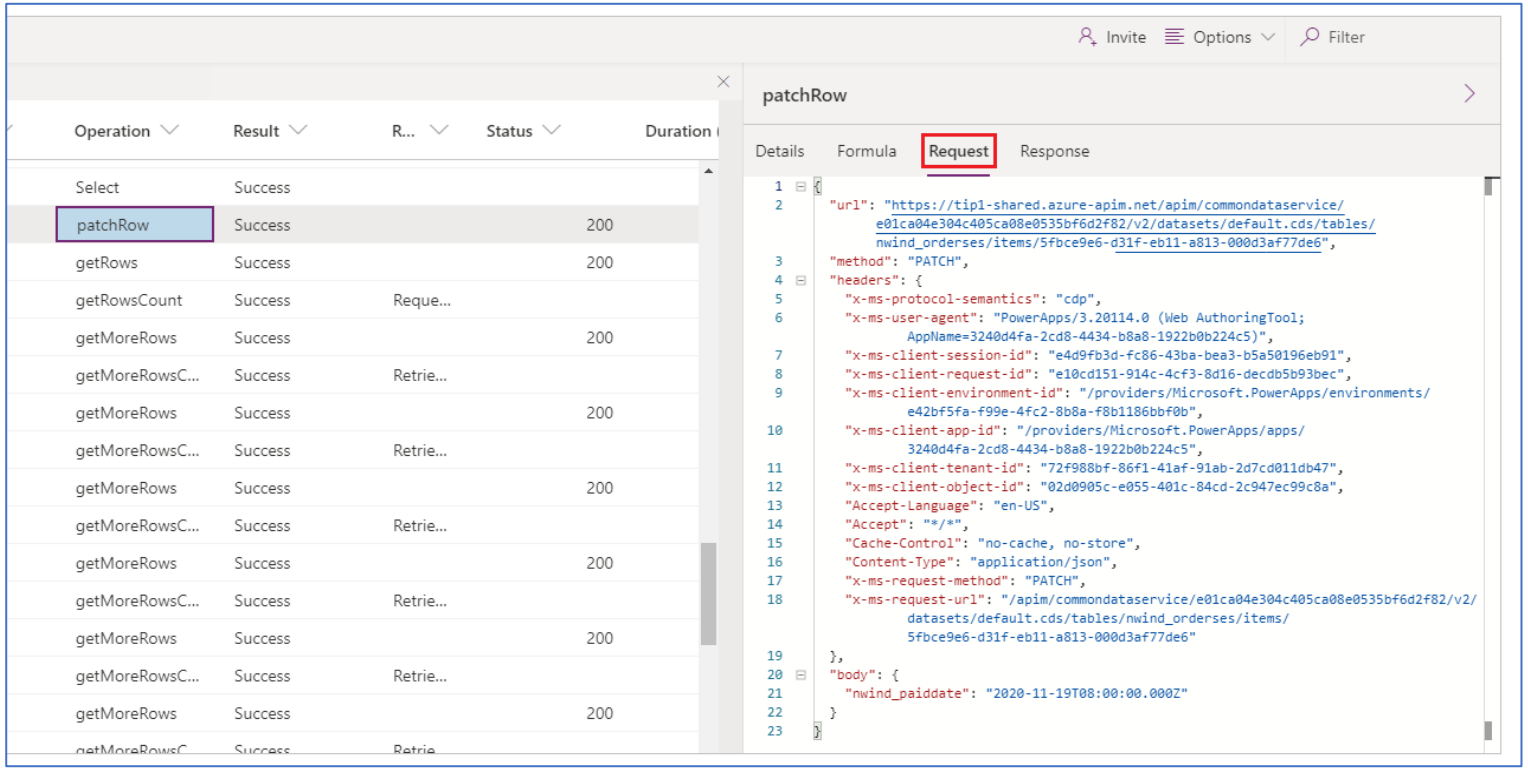


Monitoring


Brand
Reputation
CRM
Quality
Innovation
Strategy
Solutions
Analysis
Plan

Power Apps Monitor

- View events while building apps in Power Apps Studio
- View events for published apps during runtime
- Check:
  - Number of network calls
  - How often is data being retrieved
  - Response data size
  - Duration of requests
  - Errors
- Access using App Checker or Advanced Tools.



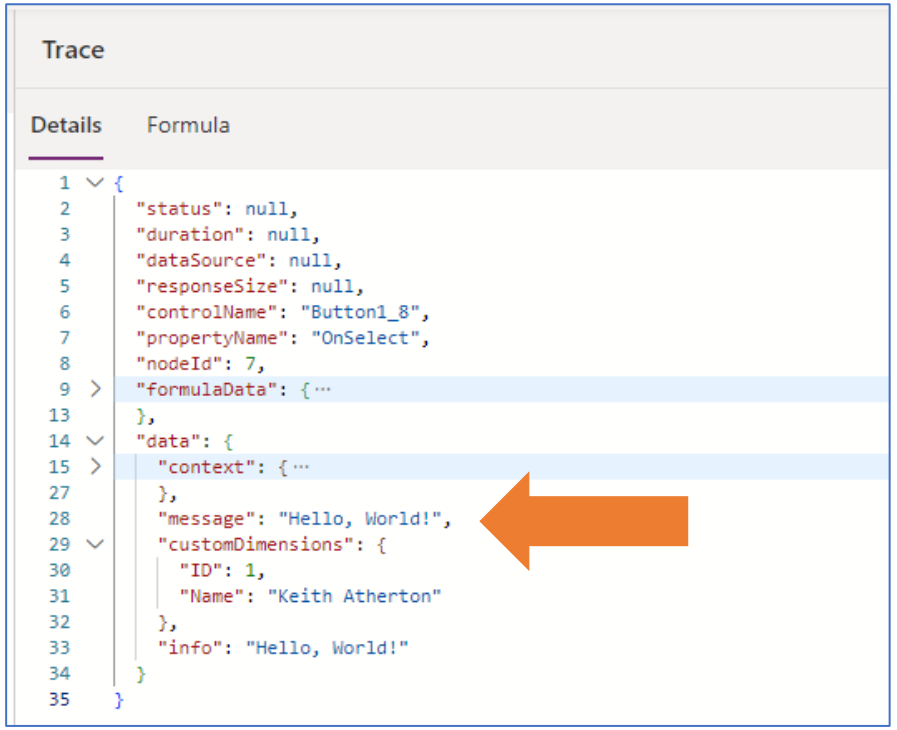
Trace function

- Record diagnostic information from behind the scenes
- Messages appear in:
  - Power Apps Monitor
  - Test Studio
  - Azure Application Insights.

Syntax:

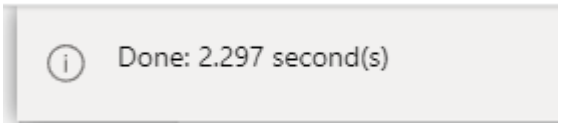
Trace(*Message* [, *TraceSeverity* [, *CustomRecord* [, *TraceOptions*]]]
)

```
Trace("Hello, World!",
    TraceSeverity.Information,
    {
        ID:1,
        Name:"Keith Atherton"
    }
);
```



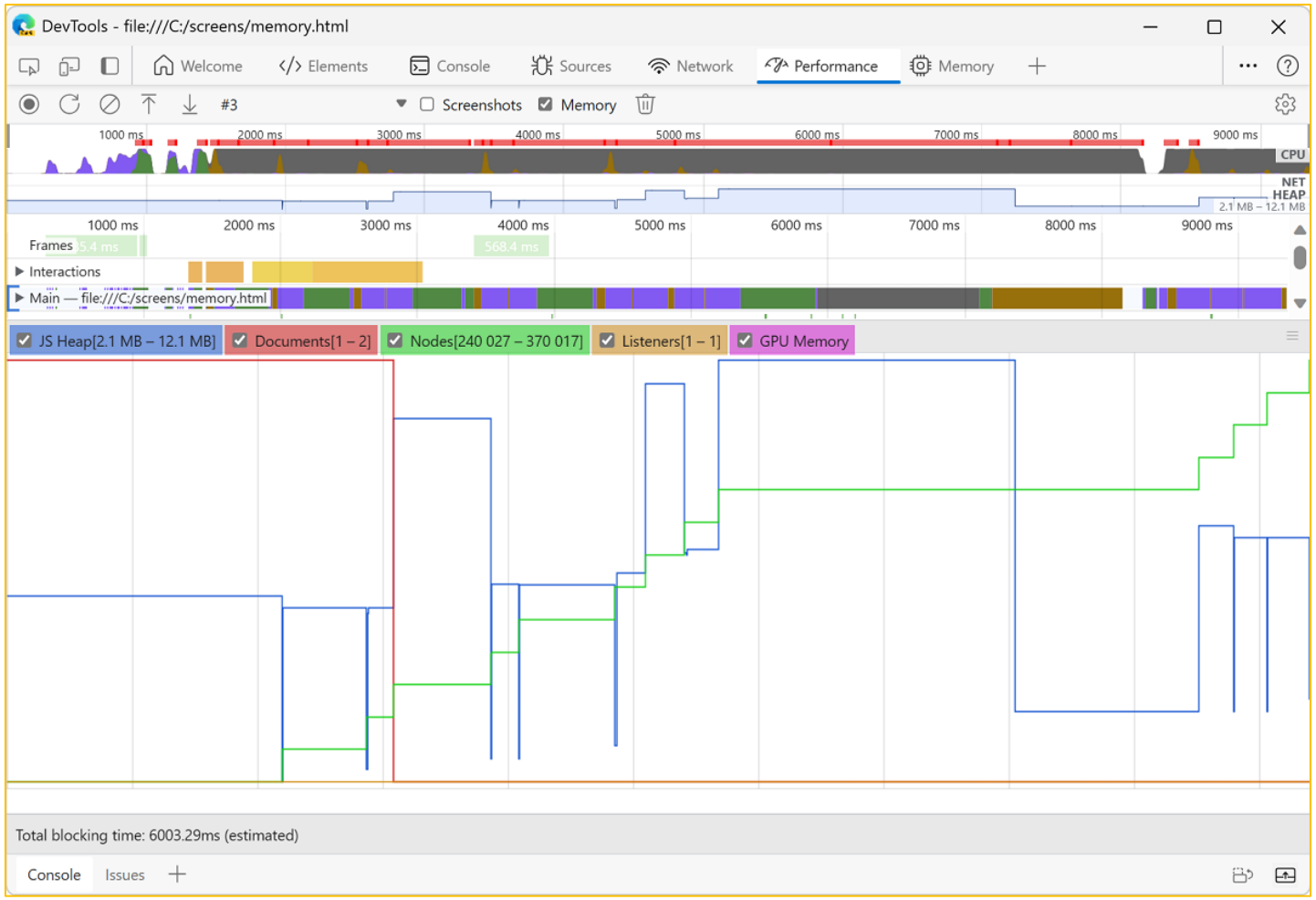
Power Apps performance monitoring Power Fx

```
UpdateContext({locStartTime:Now()});

// My wonderful (but slow) code...

UpdateContext({locTimeDiffSeconds:DateDiff(locStartTime, Now(),
TimeUnit.Milliseconds)});

Notify($"Done: {locTimeDiffSeconds / 1000} second(s)");
```



Measuring memory consumption graphically

- You can use browser developer tools for your browser to profile memory consumption graphically
- It helps you visualize heap size, documents, nodes, and listeners
- If you need to take it to *that* level… which most of us probably don't!



Agenda

- Code optimisation
- Architectural optimisation
- User experience (UX)
- Monitoring

Thank you!

Keith Atherton

- See me for stickers
  - #FirstComeFirstServed 👇

Microsoft® Most Valuable Professional

Connect with me

Microsoft Certified Trainer MCT 2023 - 2024

S1 23 · S2 23 · S1 24

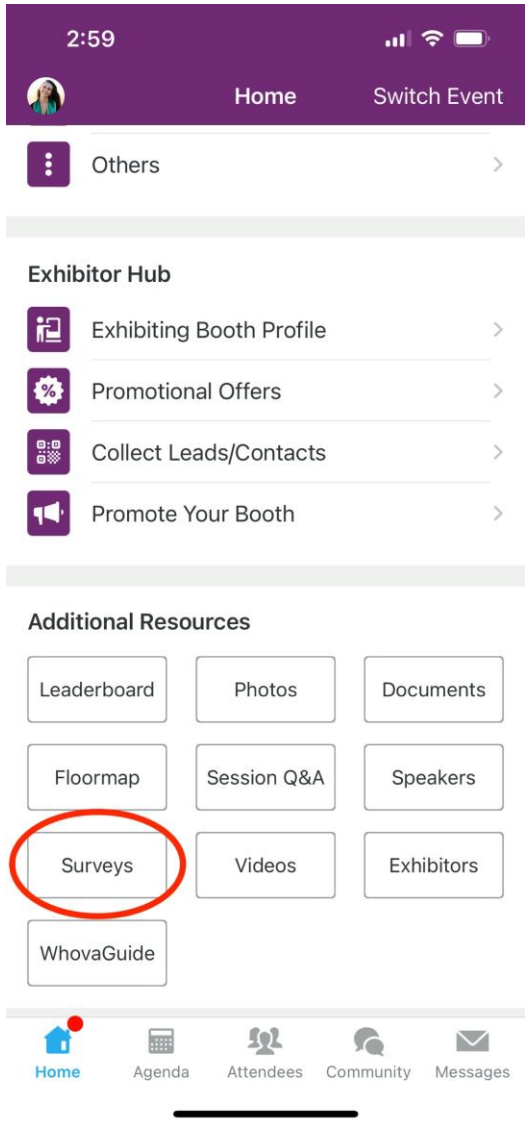
Microsoft Certified: Power Platform Solutions Architect Expert

Microsoft Certified: Power Platform App Maker Associate

Microsoft Certified: Power Platform Functional Consultant Associate

Microsoft Certified: Power Platform Developer Associate

Session Feedback Surveys



We really want to hear from YOU!

In the pursuit of making next year's Power Platform Community Conference even better, we want to hear your feedback about this session.

Here's How -

- *Simply go to the Whova App on your smartphone*
- *Scroll down on the Power Platform Community Conference Homepage to 'Additional Resources' to click "Surveys'.*
- *Click Session Feedback.*
- *Scroll down to find this session title.*
- *Complete the session feedback survey.*
- *Finally, click 'Submit'*

It's just that easy!