# Full Power! Power App Optimisation

Keith Atherton

# Keith Atherton

- Power Platform Architect
- Professional developer
- Over 25 years experience
- Microsoft MVP
- Microsoft Certified Trainer
- LinkedIn Learning Instructor
- 11x Microsoft certified
- *Power Platform Community High Five* user group leader
- *Women in Power (Platform)* mentoring program mentor
- *On Air in the Cloud* podcast host











# Agenda

- Code optimisation
- Architectural optimisation
- User experience (UX)
- Monitoring and troubleshooting.

# Why optimise?

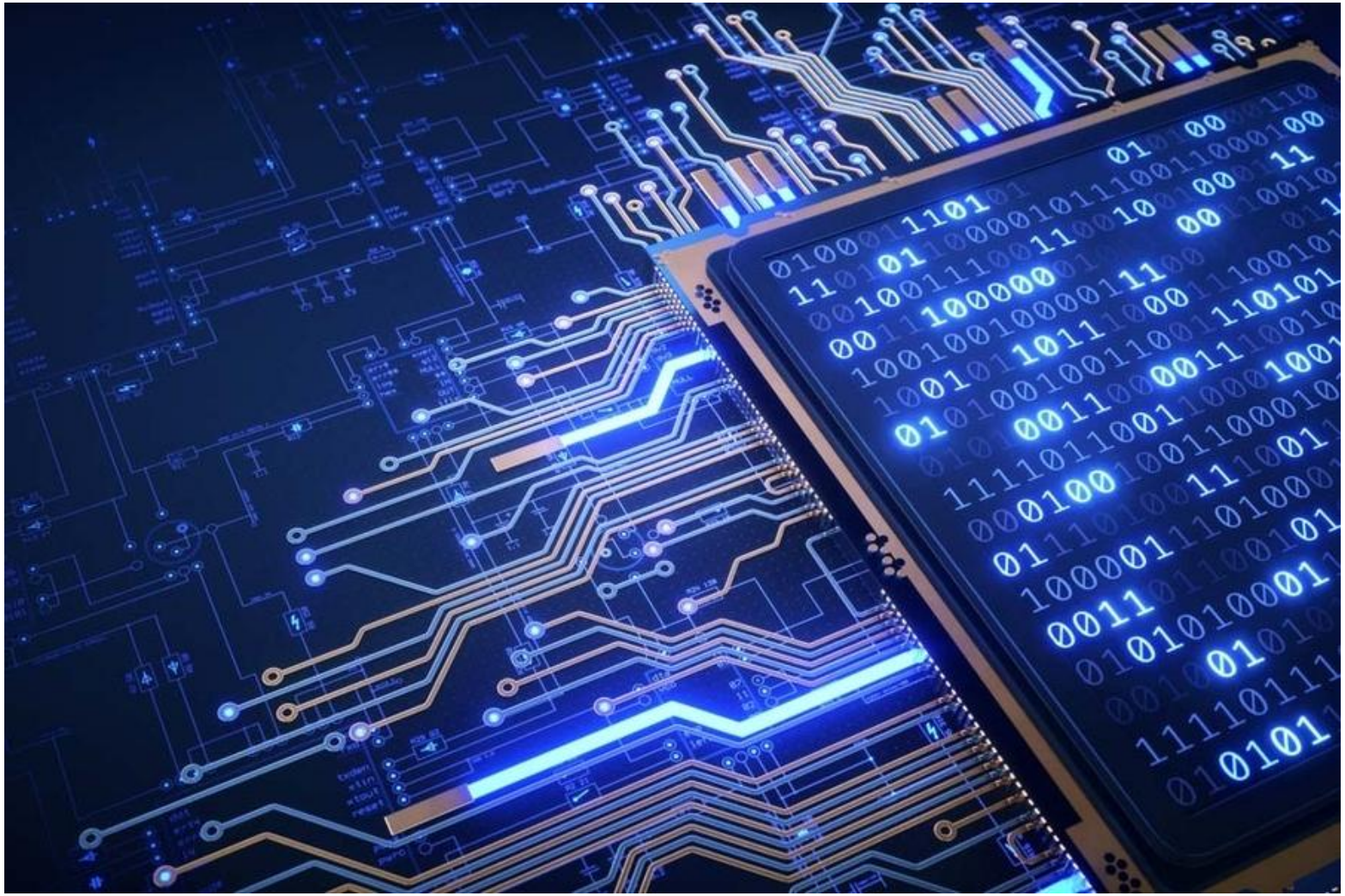
- Improved user experience
- Improved user satisfaction
- Improved user adoption
- Non-functional requirements (NFRs) may dictate app load and response time limits.

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

# The *RAIL* performance model

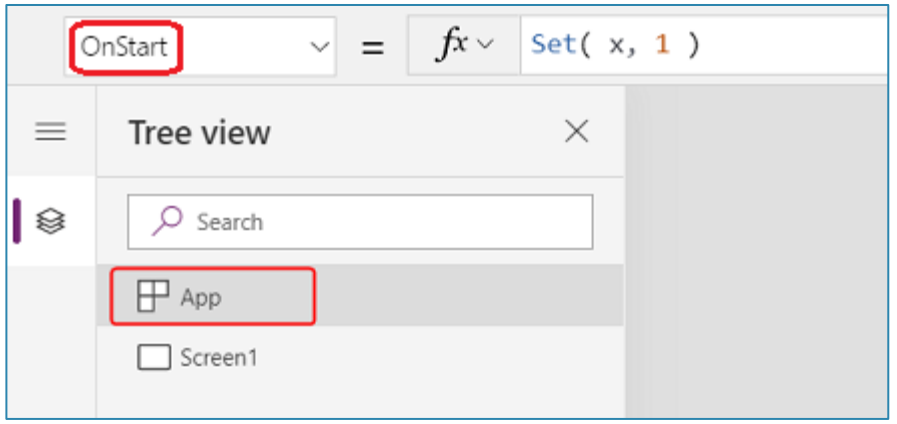
**Response** | **Animation** | **Idle** | **Load**

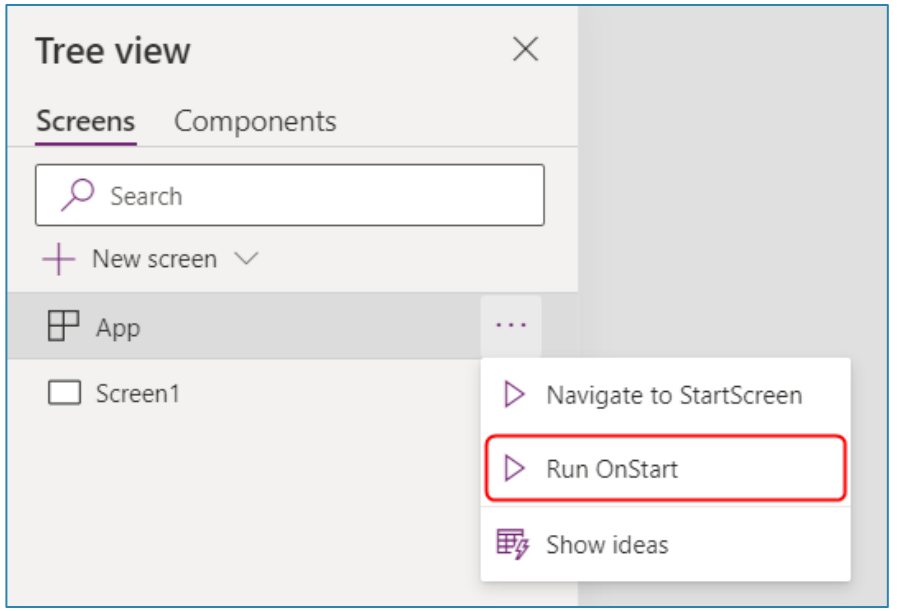
# Code optimisation



# App OnStart property (demo #1)

- The OnStart property runs when the user starts the app



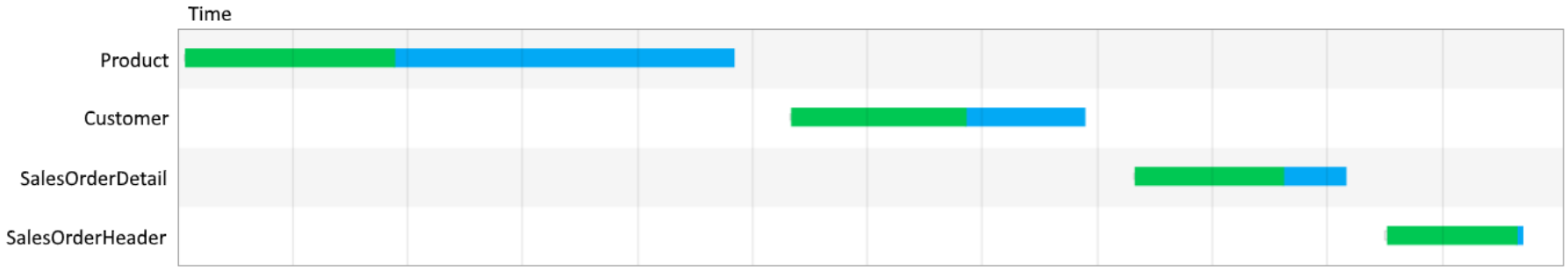

https://learn.microsoft.com/en-us/power-platform/power-fx/reference/object-app#onstart-property

# Power Apps performance monitoring Power Fx

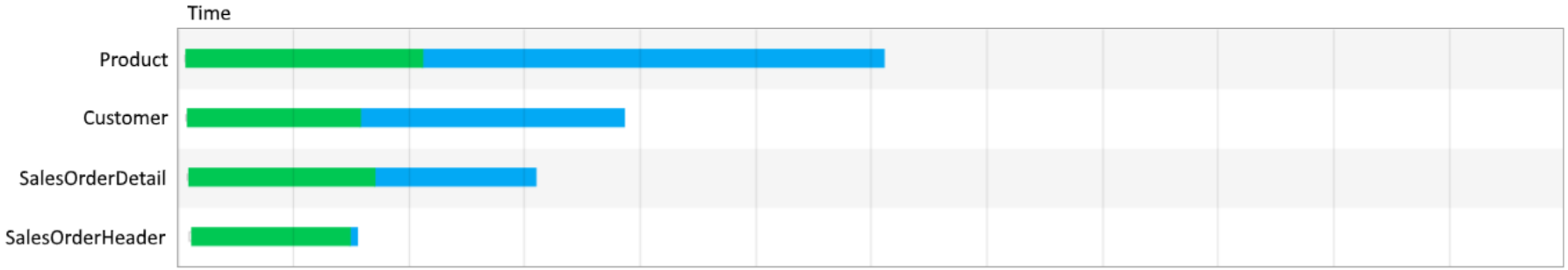
```
UpdateContext({locStartTime:Now()});

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

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

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


# Concurrent function

```
ClearCollect(colProduct, '[SalesLT].[Product]');
ClearCollect(colCustomer, '[SalesLT].[Customer]');
ClearCollect(colSalesOrderDetail, '[SalesLT].[SalesOrderDetail]');
ClearCollect(colSalesOrderHeader, '[SalesLT].[SalesOrderHeader]');
```




```
Concurrent(
   ClearCollect(colProduct, '[SalesLT].[Product]'),
   ClearCollect(colCustomer, '[SalesLT].[Customer]'),
   ClearCollect(colSalesOrderDetail, '[SalesLT].[SalesOrderDetail]'),
   ClearCollect(colSalesOrderHeader, '[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(gblPetName, "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(
    {
        Title:LookUp(Users, 'Primary Email' = User().Email, Title)
    },
    Set(gblUserTitle, Title);
);
```

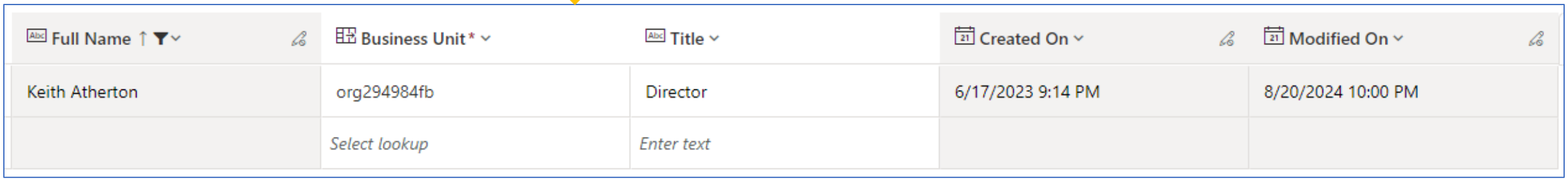
# And another thing about those *With* function records…

- 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({
      UserRecord:LookUp(Users, 'Primary Email' = User().Email)
   },
   Set(gblTitle, UserRecord.Title);
   Set(gblFirstName, UserRecord.'First Name');
   Set(gblLastName, UserRecord.'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.

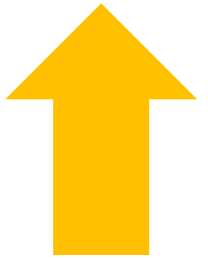
**Syntax:**
LookUp(Table*, *Formula* [, *ReductionFormula* ] )

```
// Save user title to variable the slooooow way. 😨
Set(gblUserRecord,
   LookUp(Users, 'Primary Email' = User().Email)
);

Set(gblUserTitle, gblUserRecord.Title);


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

# Retrieving table data

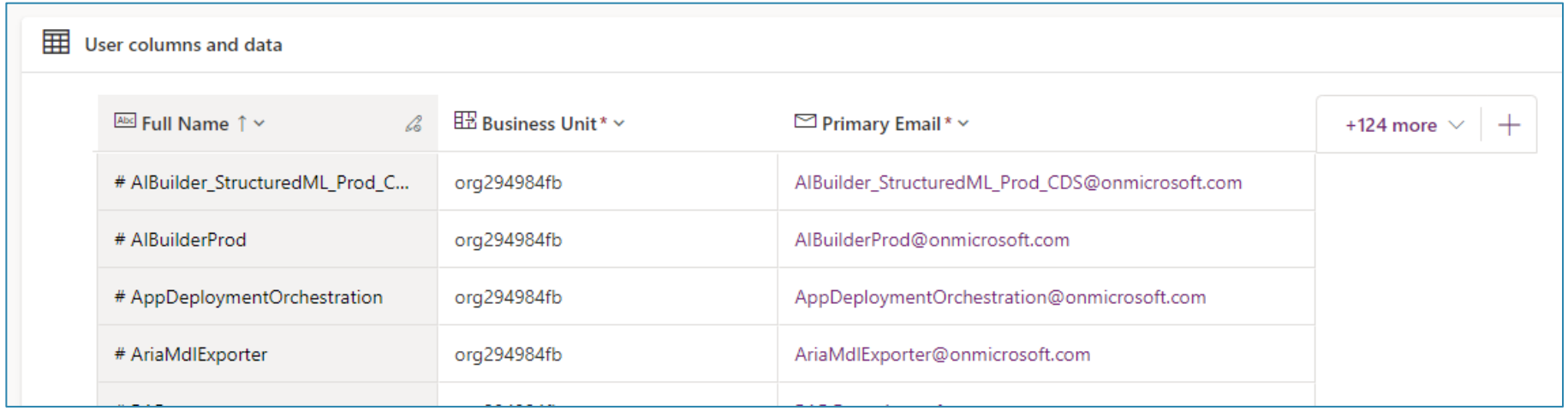
User columns and data

| 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 | |



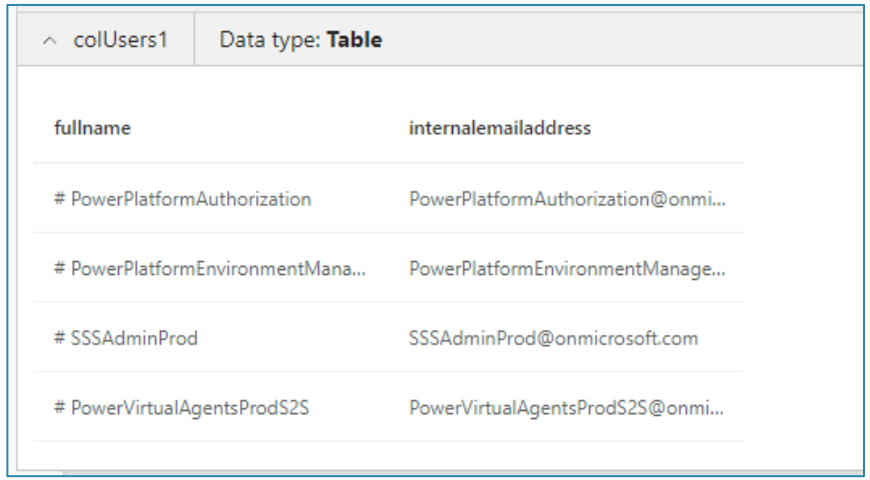
# Retrieve just the table columns you need

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




etc...

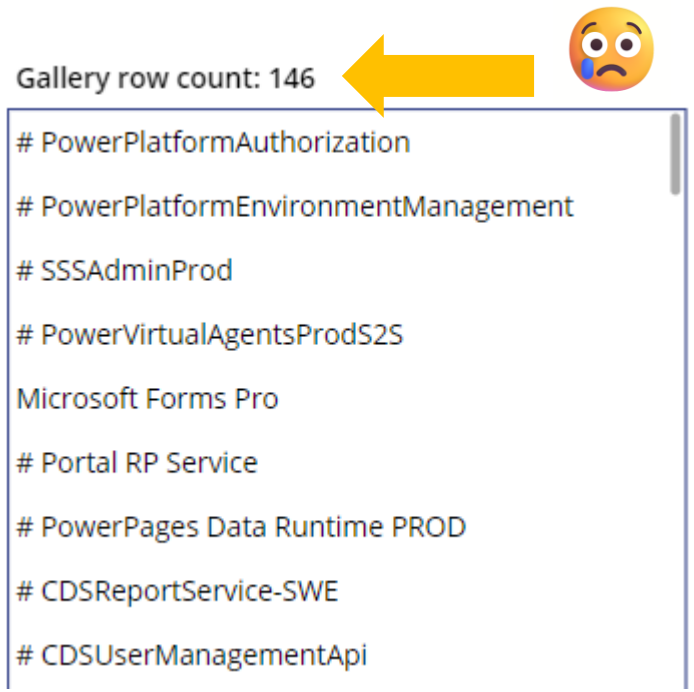
```
ClearCollect(colUsers1,
    ShowColumns(Users,
        'Full Name',
        'Primary Email'
    )
);
```



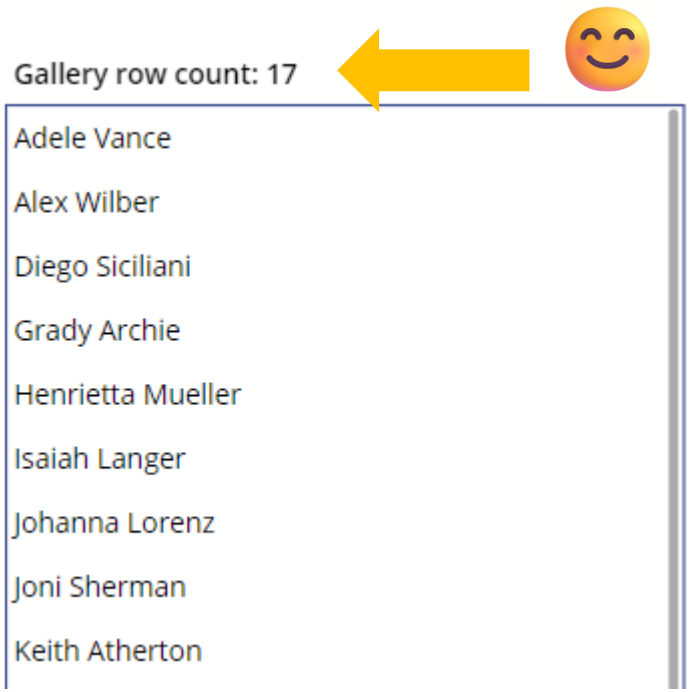

Note: This can be mitigated by *Explicit column selection (ECS)* which automatically computes which columns are necessary to retrieve based on their usage in controls (for example, galleries and forms.)

# Retrieve just the table rows you need

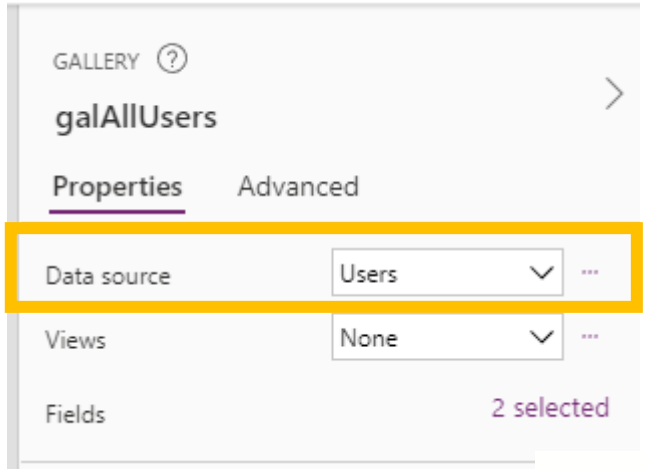
```
ClearCollect(colUsers,
    Users
);
```




```
ClearCollect(colUsers,
    Filter(Users,
        IsStaff = true
    )
);
```

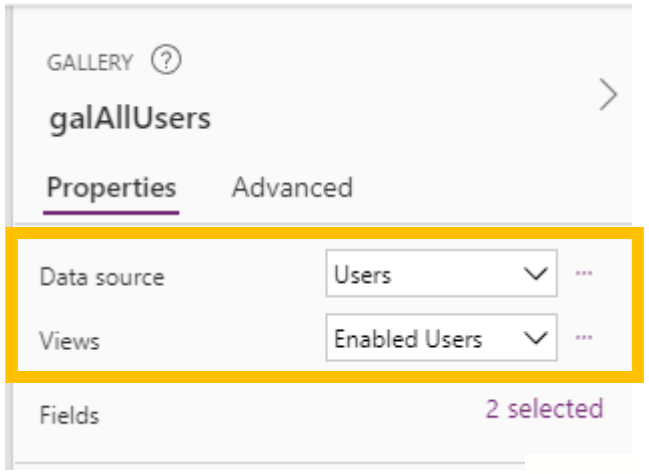


# Consider Dataverse table views for galleries




Gallery row count: 146

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



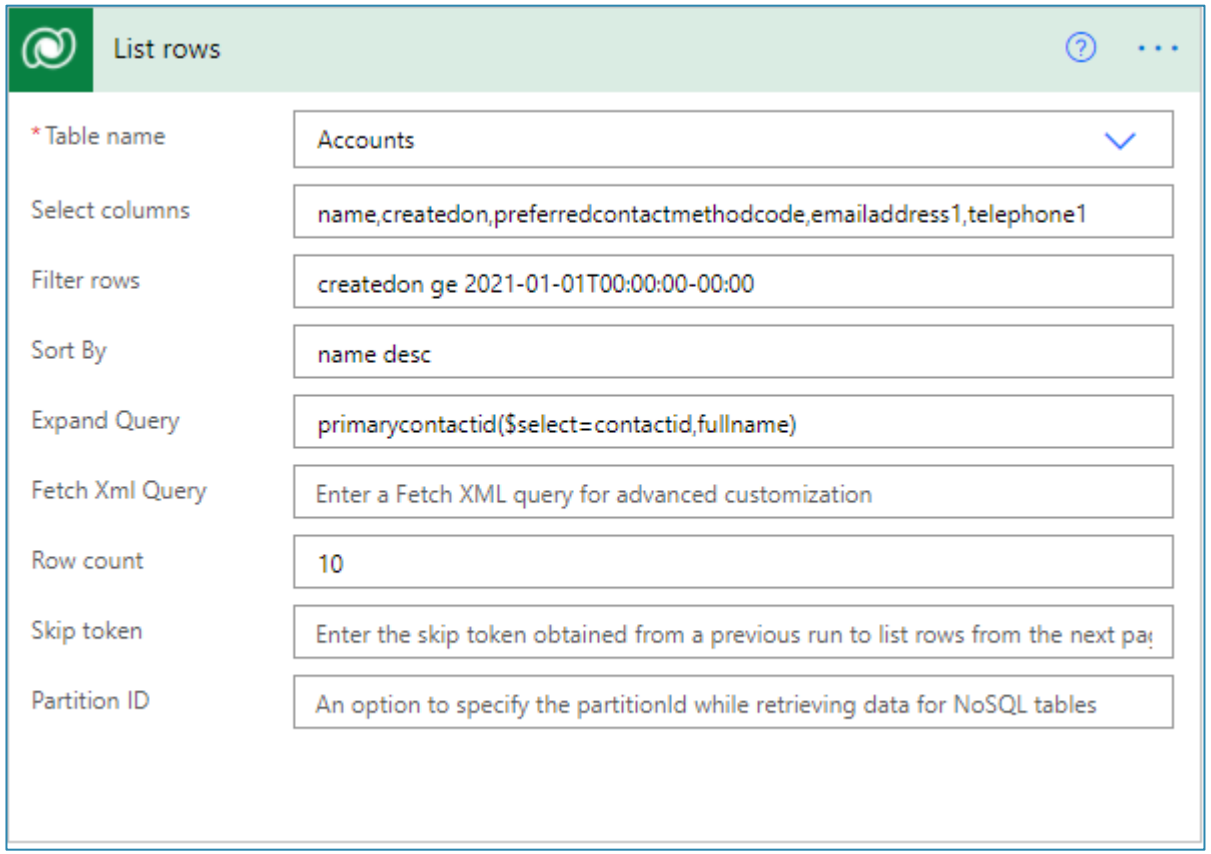

Gallery row count: 17

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

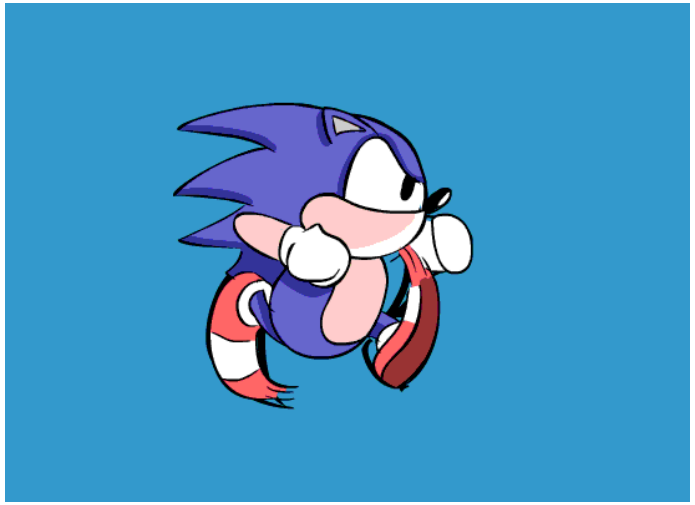
- Retrieve then cache (store data in local memory) using variables and collections where possible
- Referencing cached data is much faster than retrieving it from the data source
- Consider:
  - How often does data change?
  - Is real-time data needed?

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

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

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

# Patch records FAST

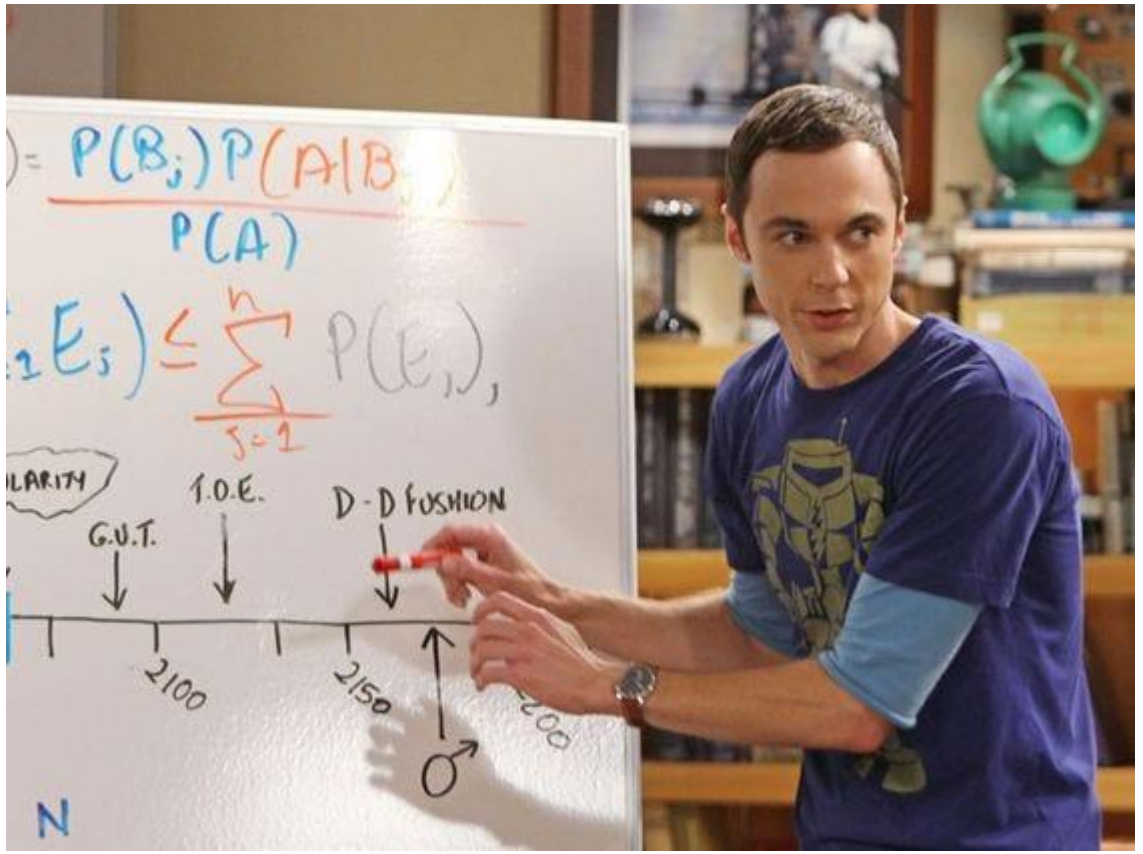
```
// Patch each record one-at-a-time. 😢
ForAll(
    colAccounts,
    Patch(
        Accounts,
        Defaults(Accounts),
        {
            'Account Name':"Test Account",
            'Account Number':"123"
        }
    )
);
```

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

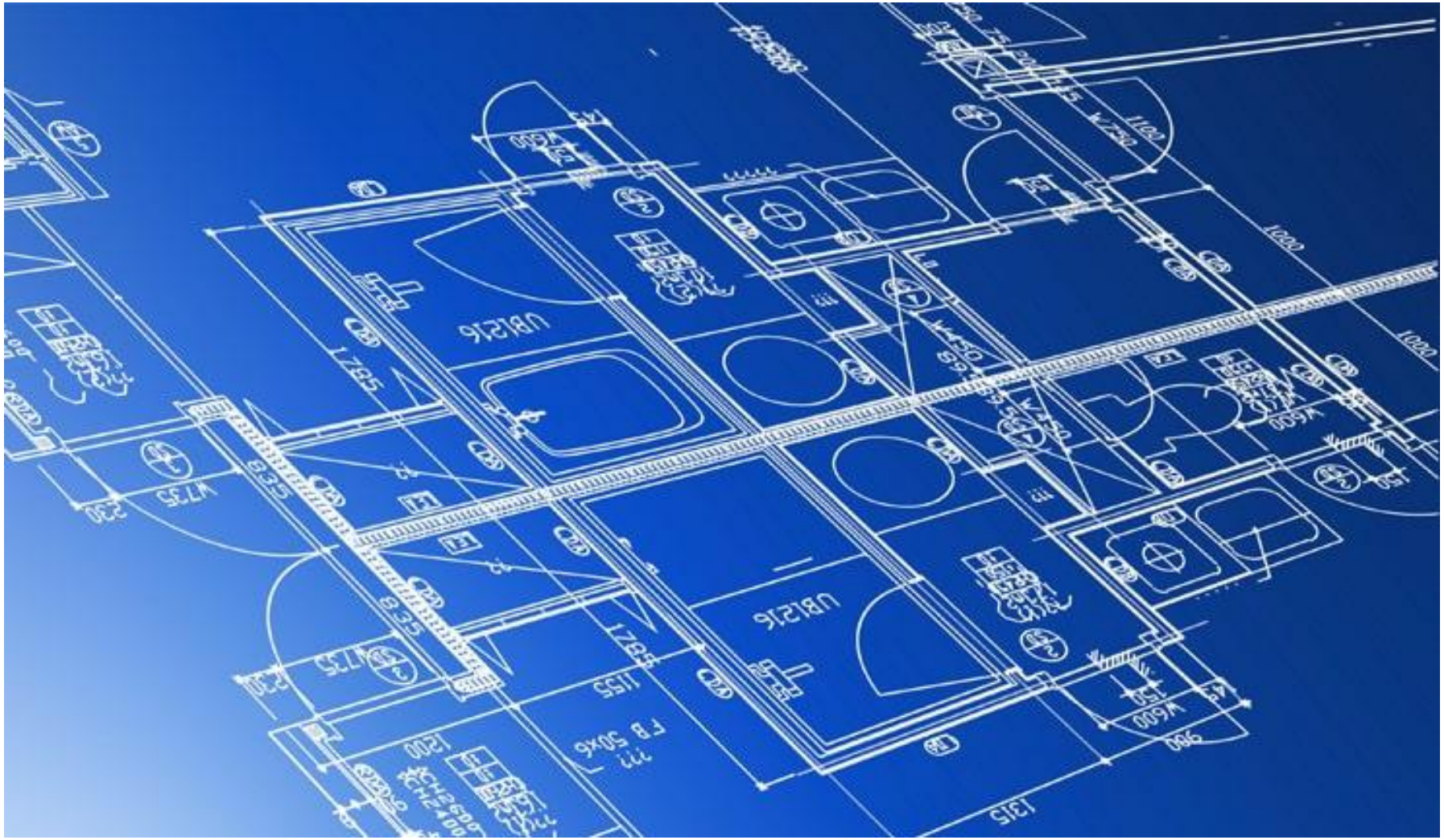


# Avoid over-engineered designs and other tips

- 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.

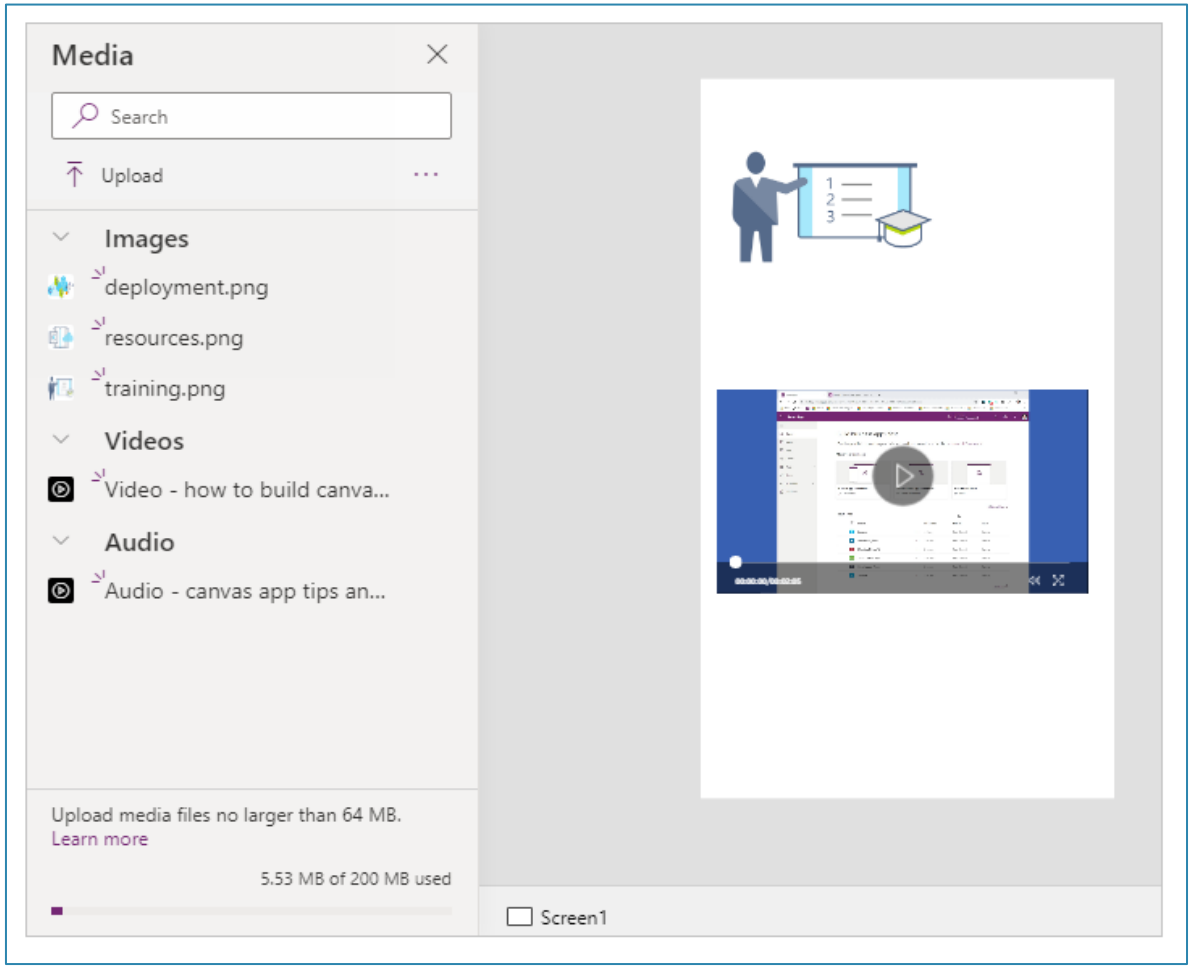


# Architectural optimisation



# Consider uploaded media

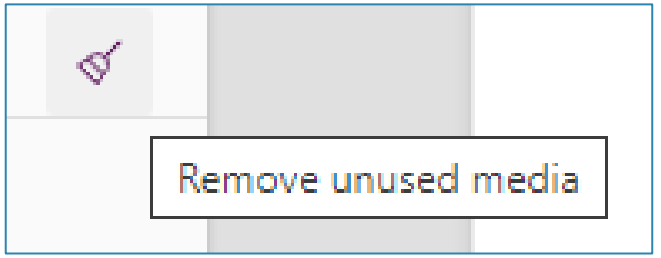
- You can add images, audio & video files to a canvas app
- SVGs are typically lighter than PNG, JPGs, etc
- Use Dataverse image thumbnails where appropriate, e.g. in a gallery.

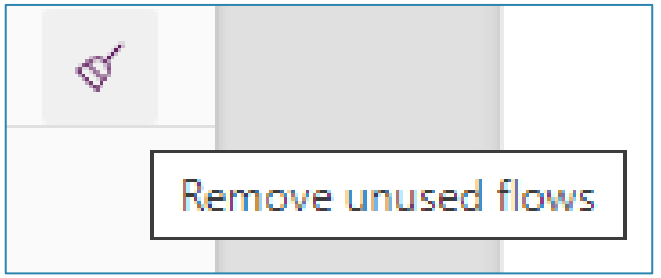


# Remove unused elements

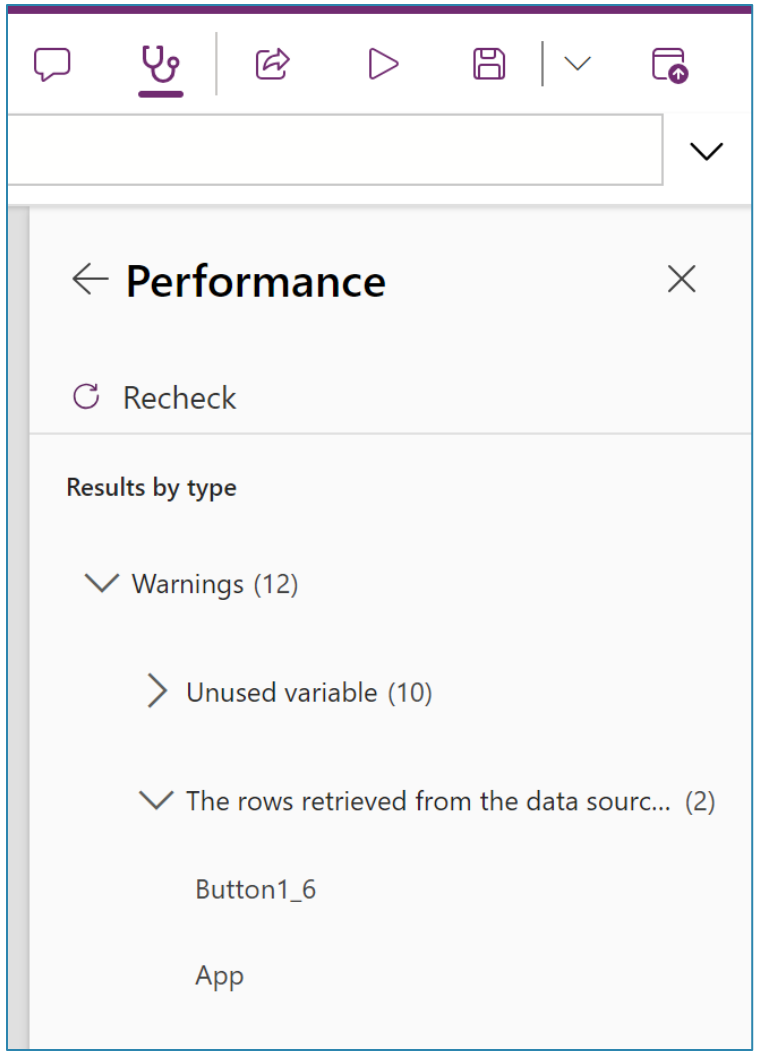
- Automatically remove data not currently in use
- Remove unused media
- Remove unused flows





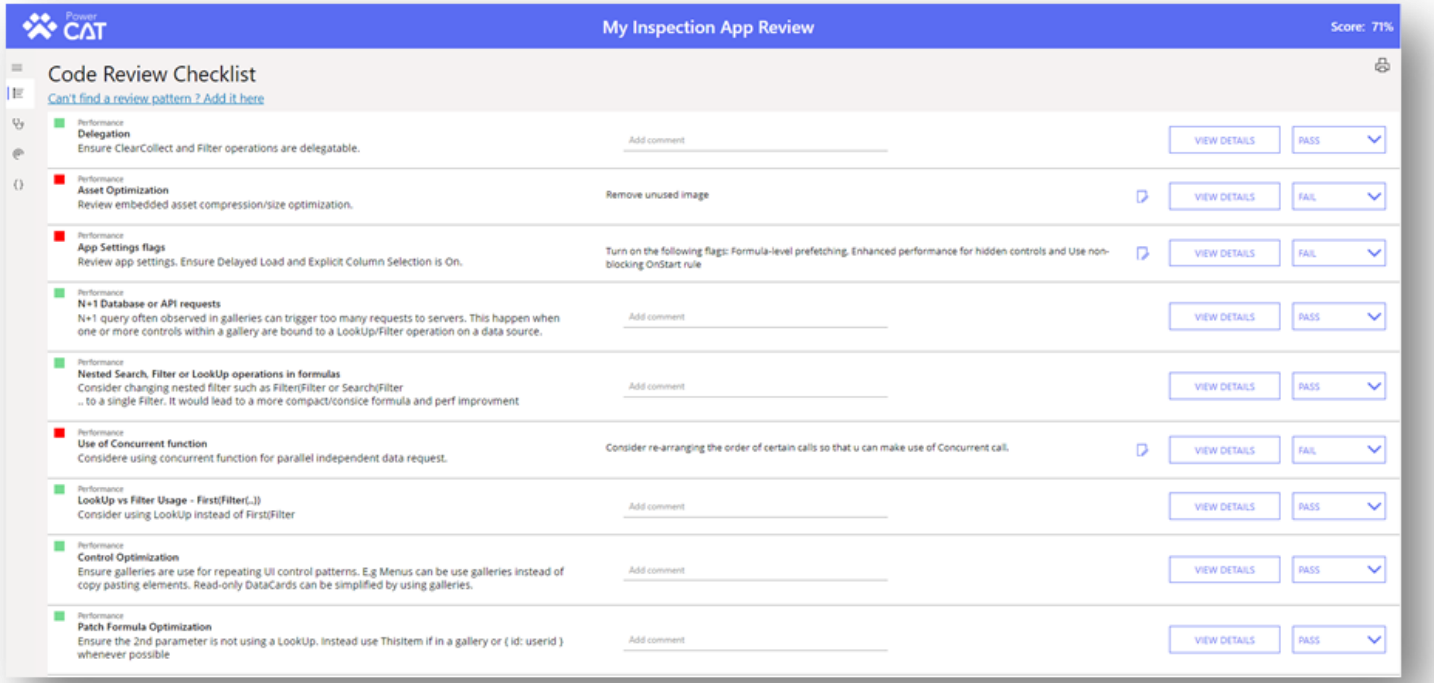



# Use App Checker (demo #2)



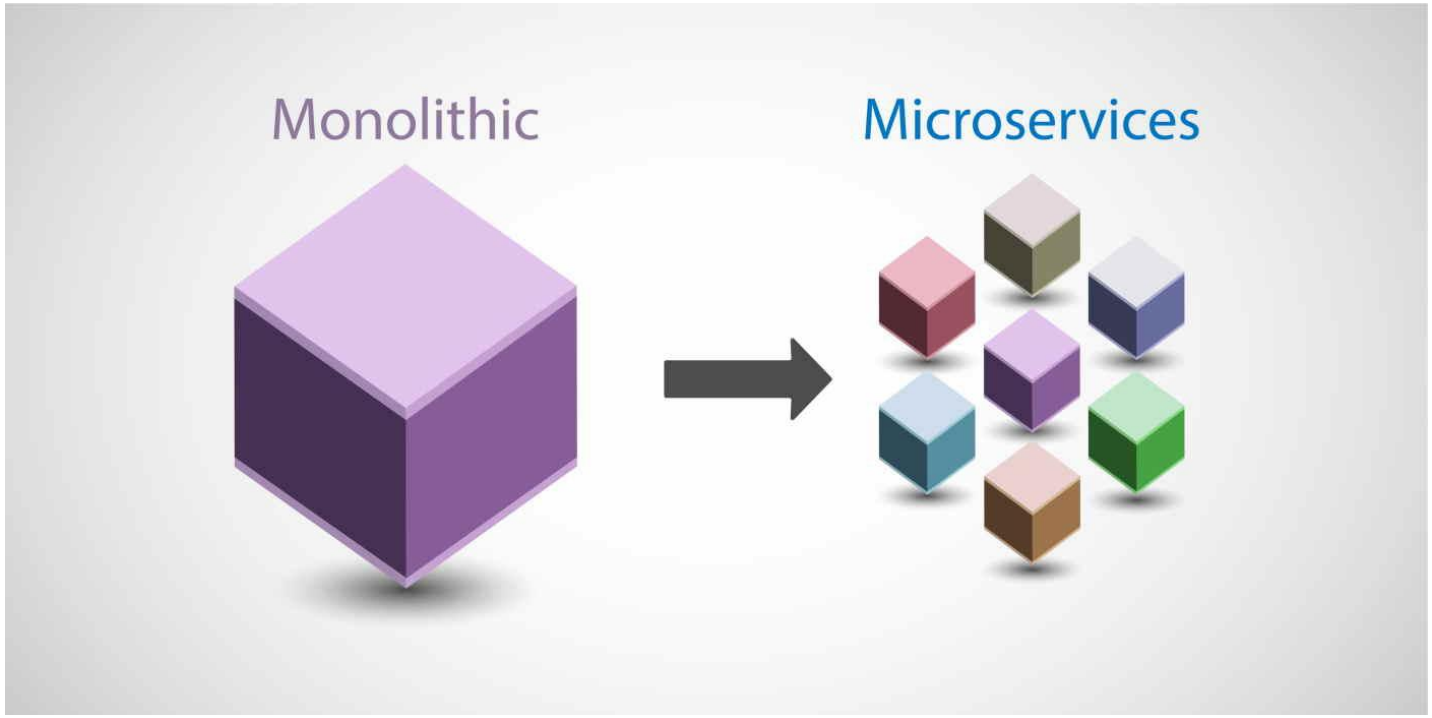
# Power Apps Code Review Tool

- Reviews a canvas app against a checklist of best practices and provides a score, e.g. 90% / A
- App Checker results are also provided
- App Analysis checking important app setting flags are on, number of controls per screen, etc
- Code Viewer lists all screens, controls and properties with a search and filter function.



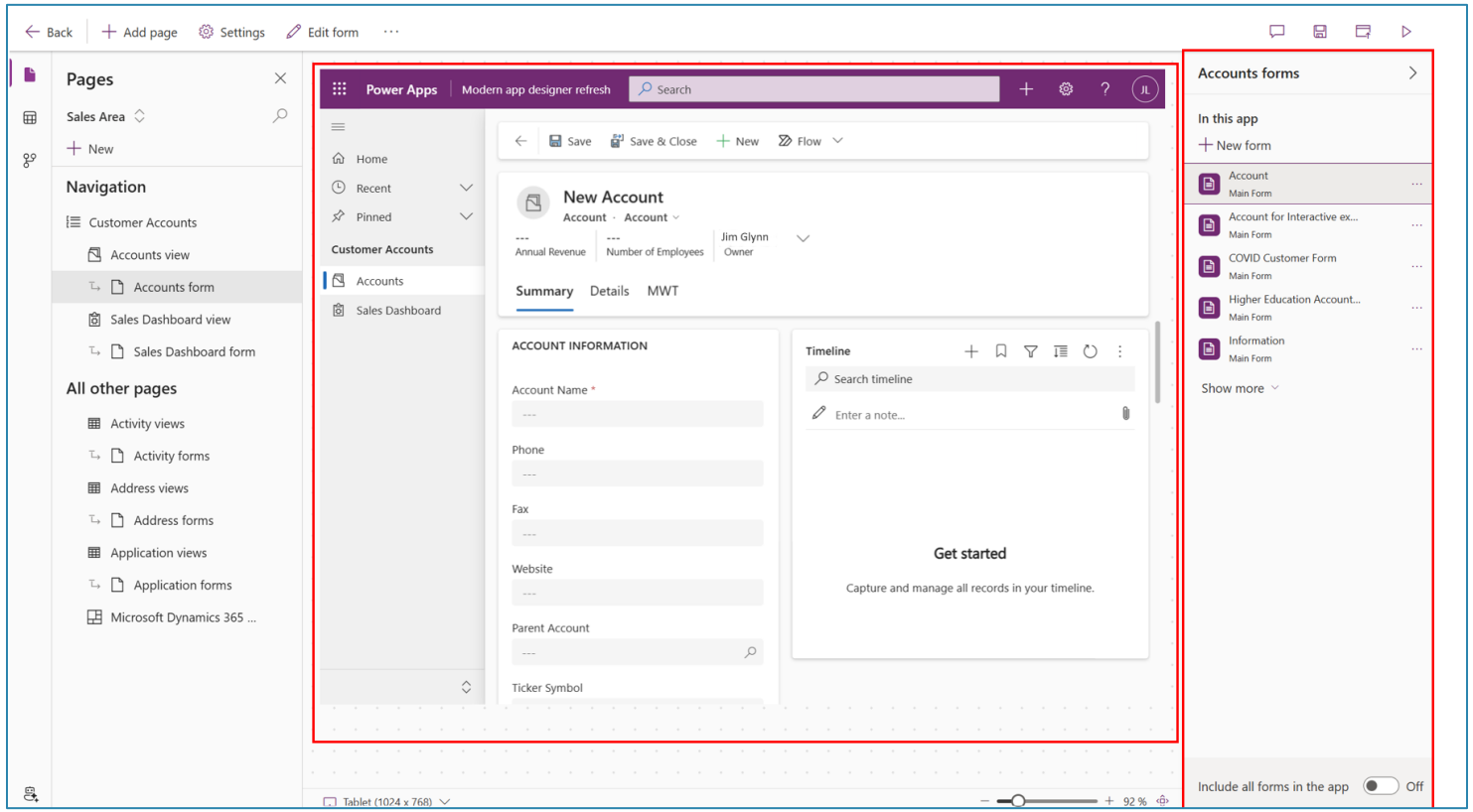
# Monolithic design

- 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 navigate to other apps using the *Launch* function and passing parameters if required.



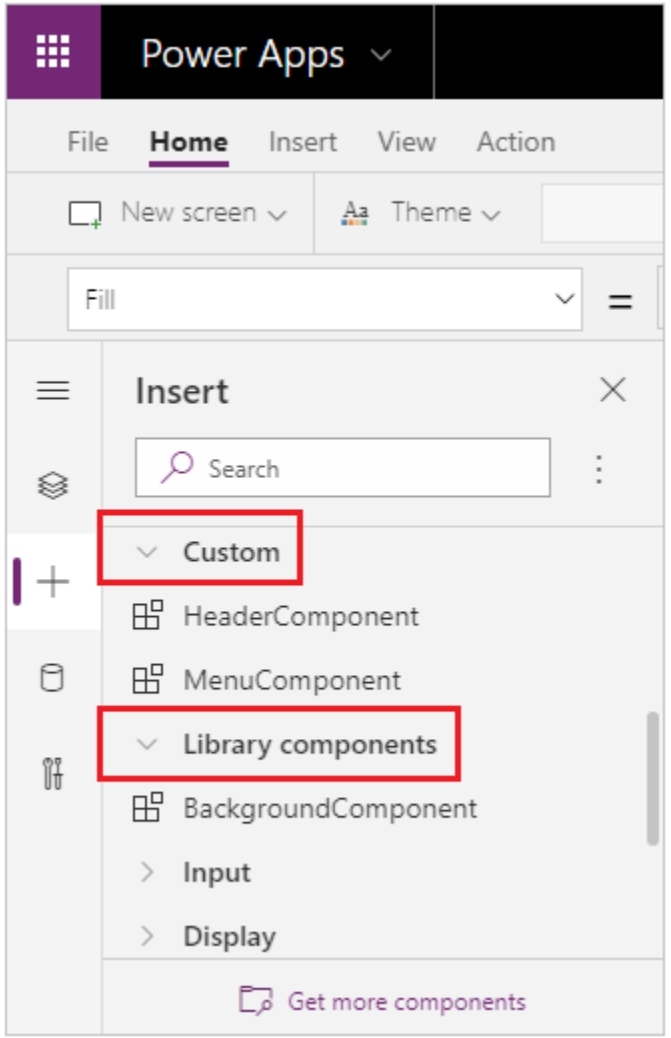
# Model-driven app considerations

- 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.

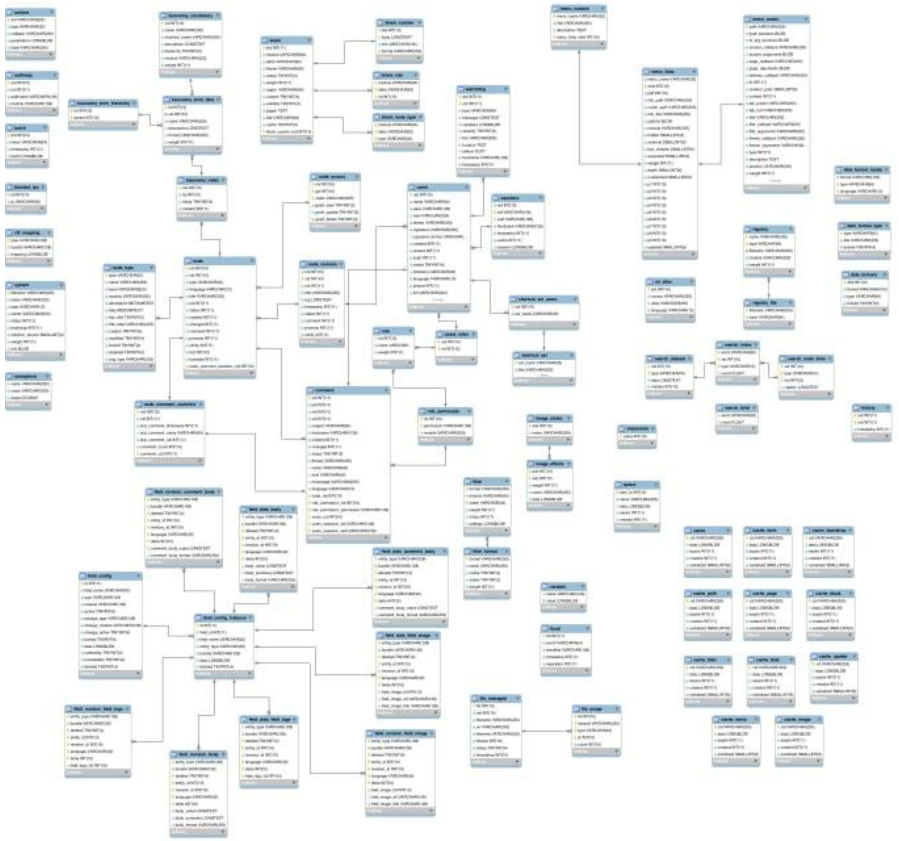


# *Development* optimisation

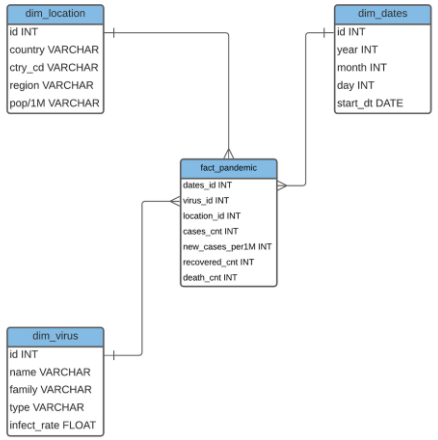
- DRY = *Don't Repeat Yourself*
- Consider reusable items:
  - Named formulas
  - User defined functions
  - Components 👉
  - Component libraries
  - Dataverse Functions (formerly known as *low-code plug-ins*, preview).



# Beware of Dataverse data model over-normalisation



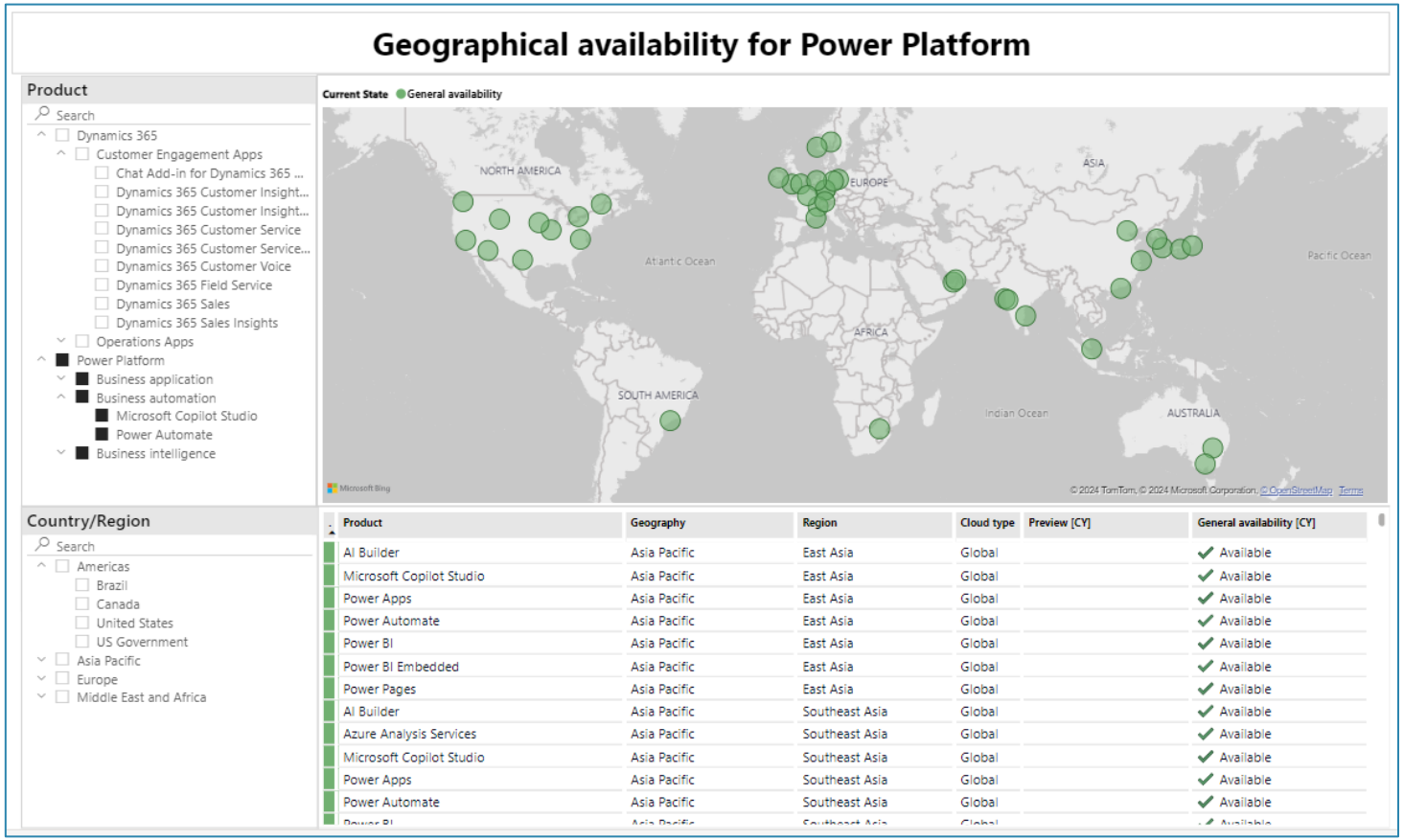
What I thought I needed…




What I *actually* needed 🙄

# Are your users close to your data?

- Consider the region for new Power Platform 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?

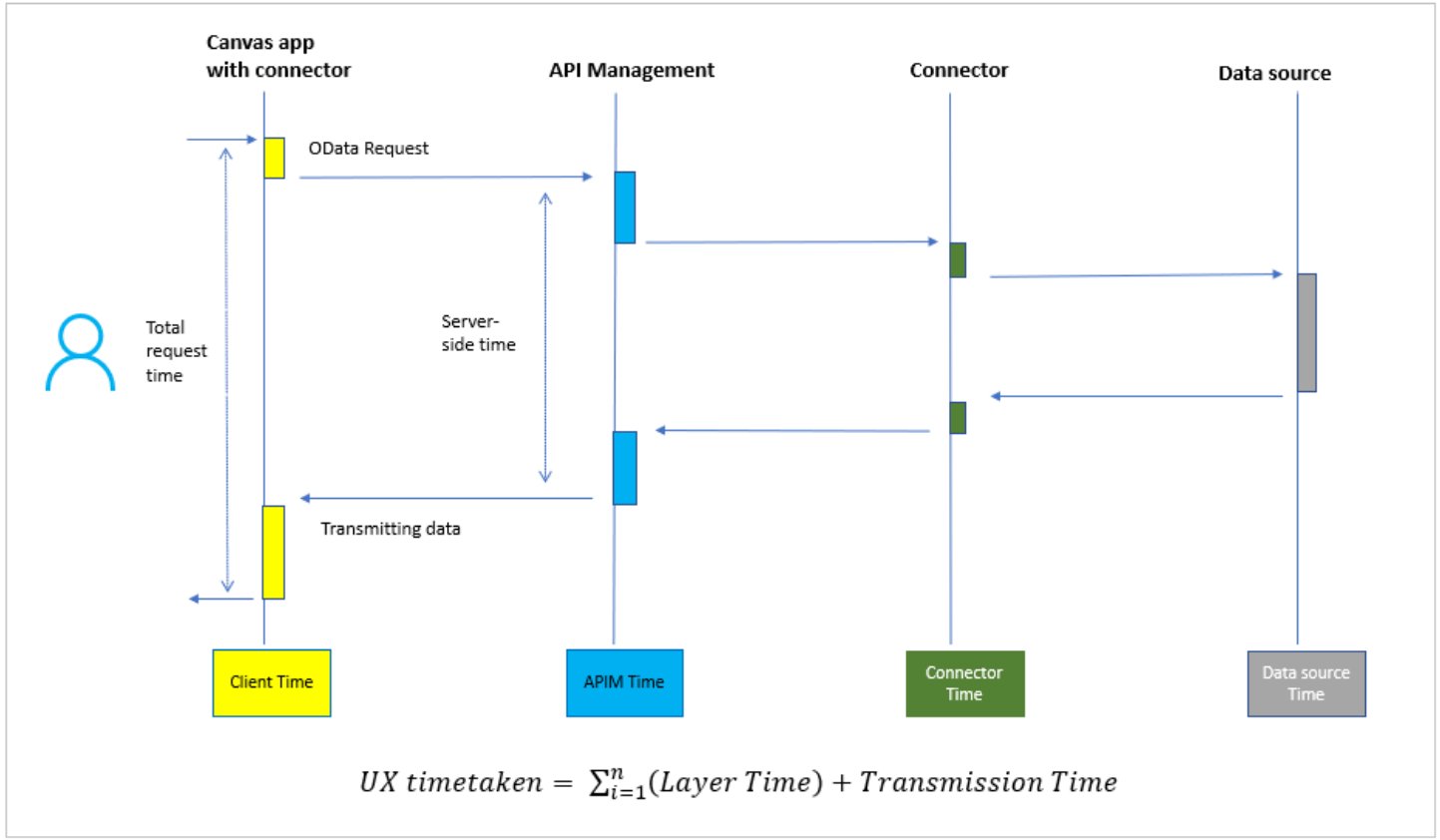

Online data source
e.g. Azure SQL Database


On-premises data source
e.g. SQL Server

Dataverse

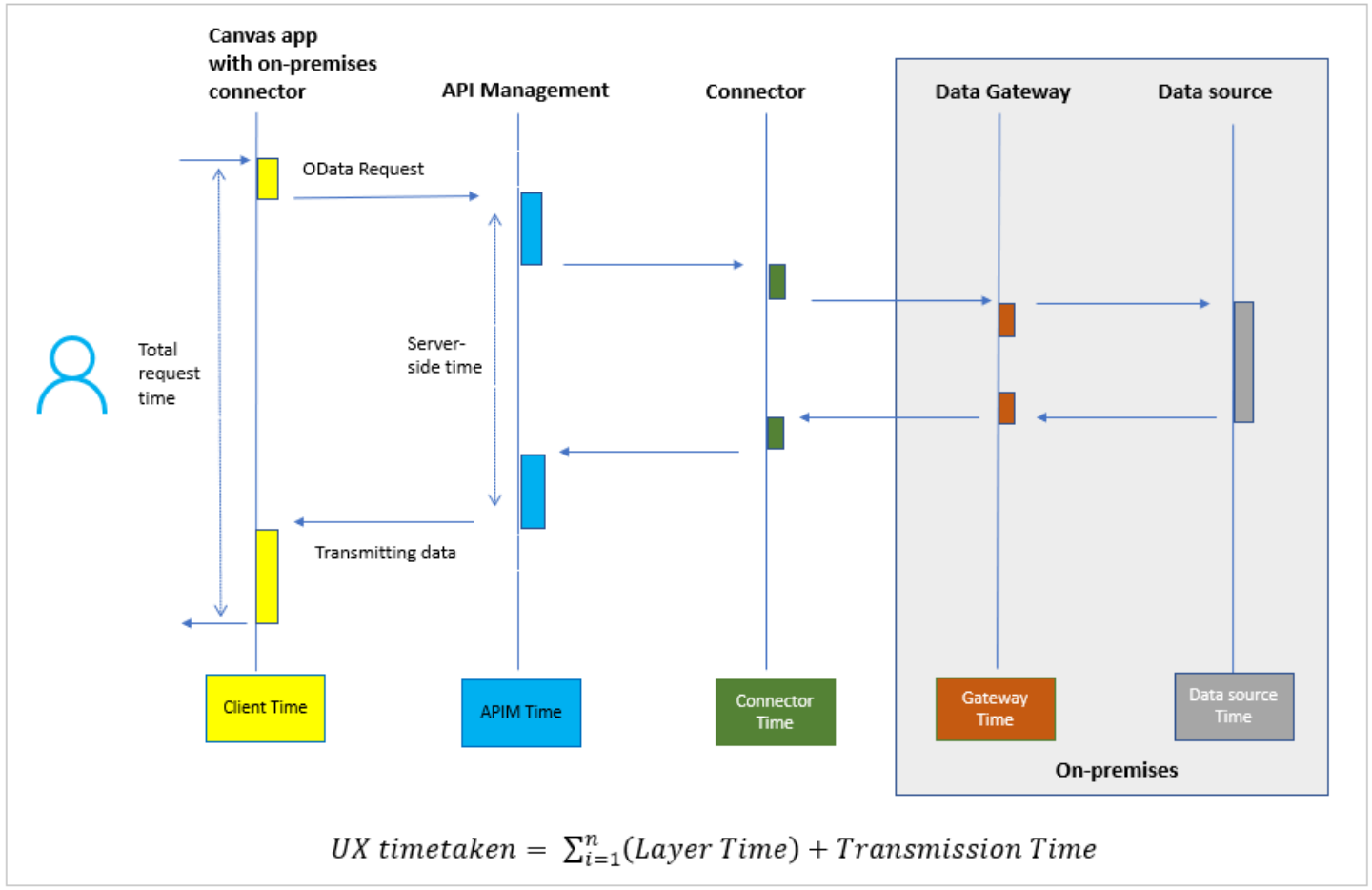
# Online data source (e.g. Azure SQL Database) data call flow

- 2 layers to the data source



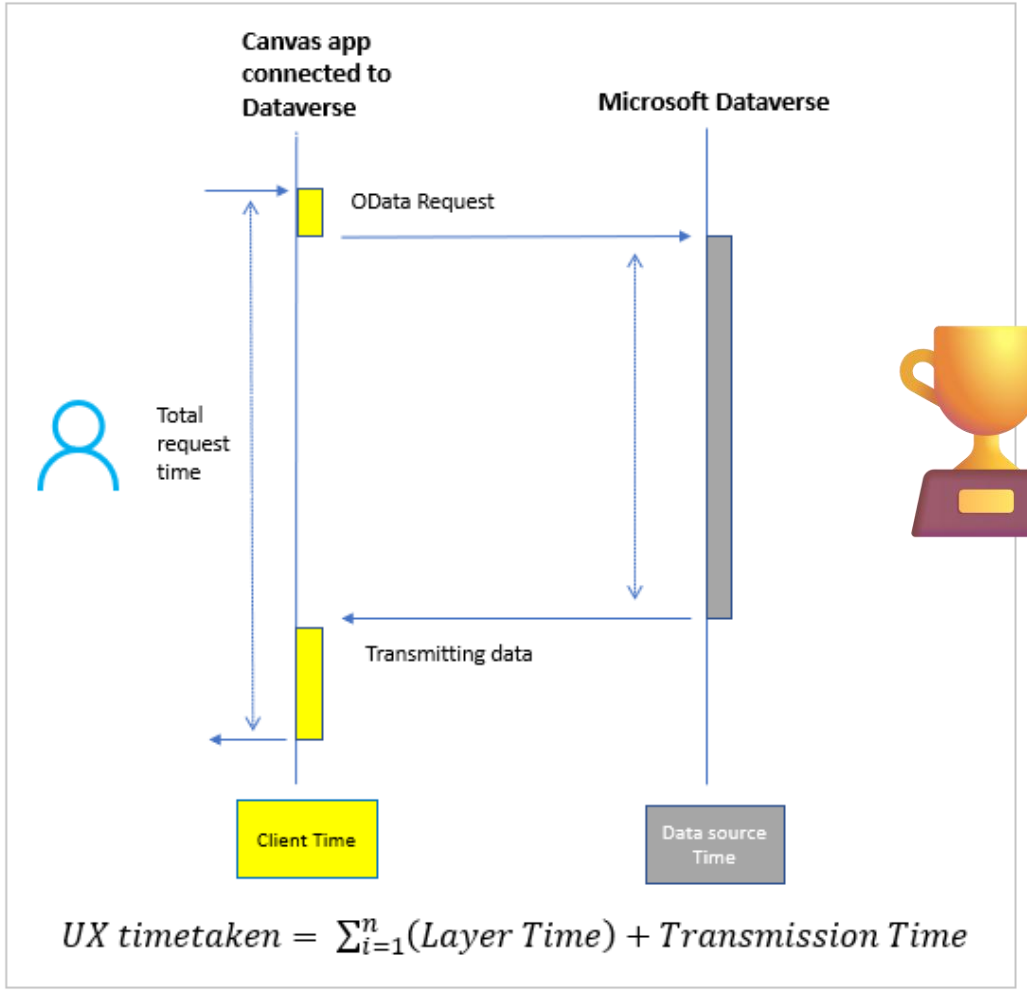
# On-premises data source data call flow

- 3 layers to the data source!
- You need to have another layer called an *on-premises data gateway*



# Dataverse data call flow

- Data requests go to the environment instance directly – without passing through Azure API Management (APIM) 😎






?

# 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: Canvas 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).

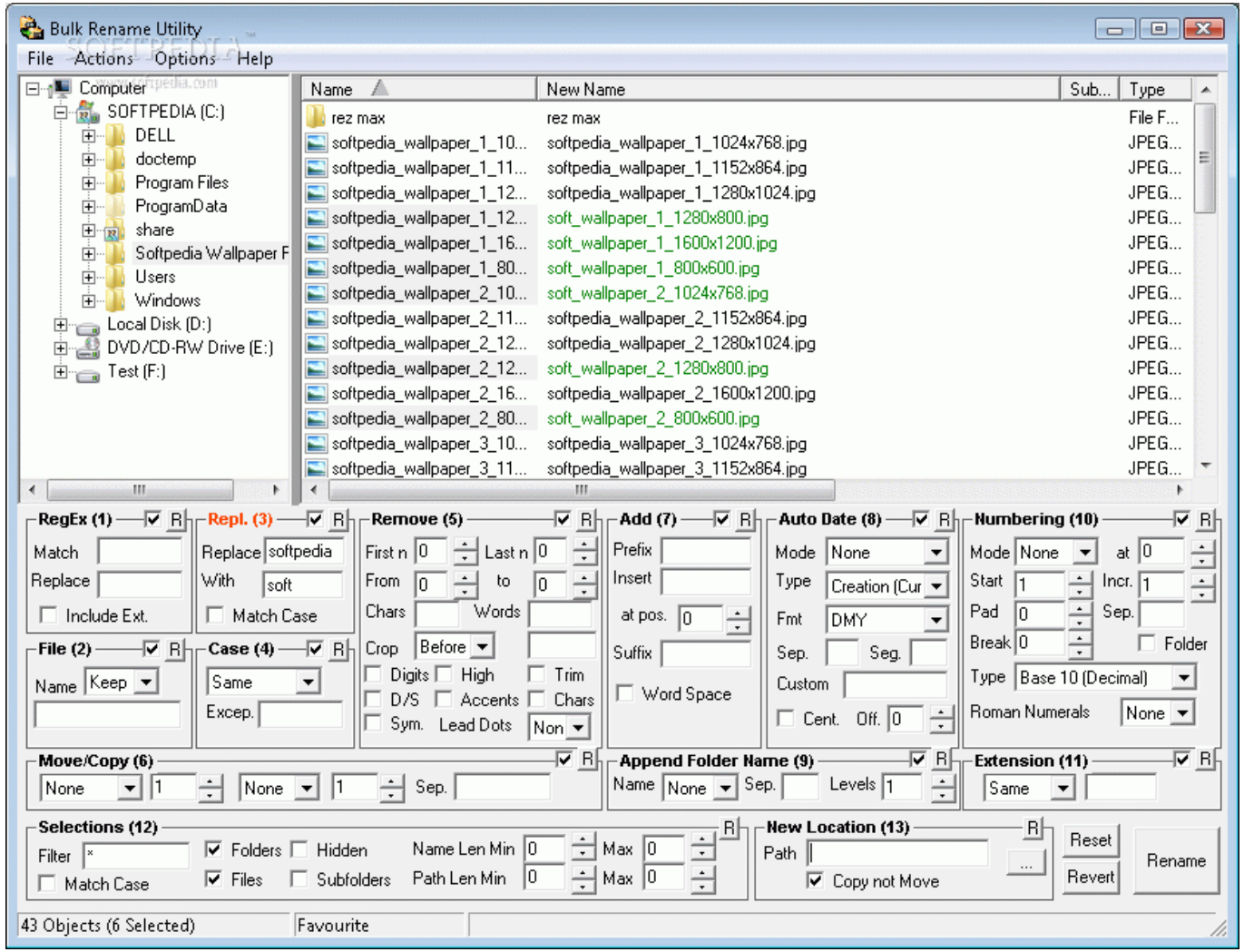
Expand table

| 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)?

- Controls require memory, therefore *lots of* controls
- C
- C



# Providing user feedback (demo #3)

# Text Input *DelayOutput* property (demo #4)

# Monitoring and troubleshooting

# Power Apps Monitor (demo #5)

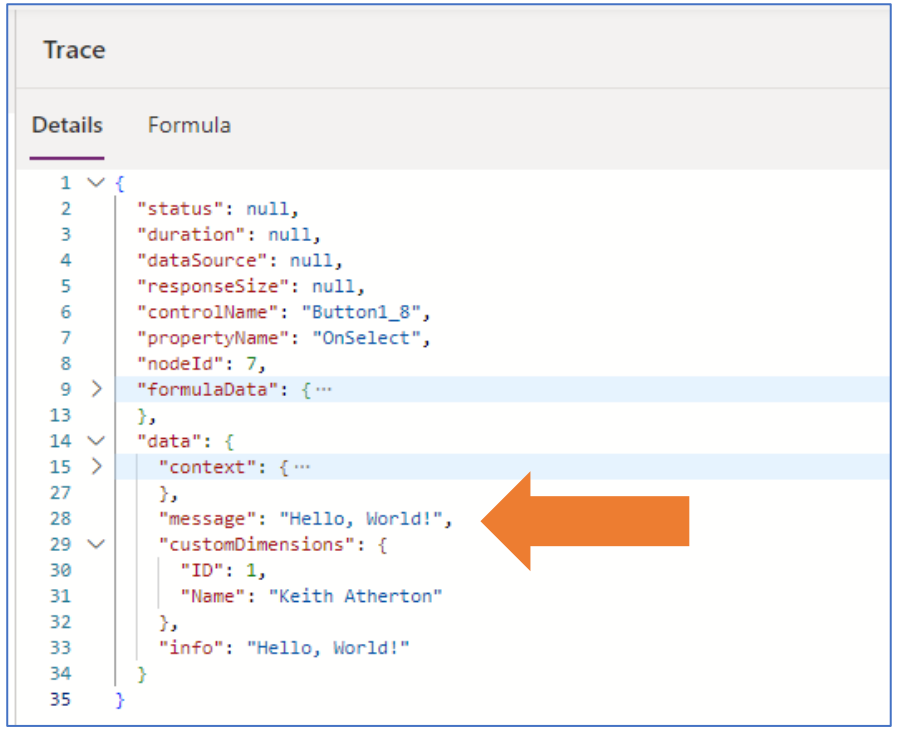
# Trace function

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

**Syntax:**

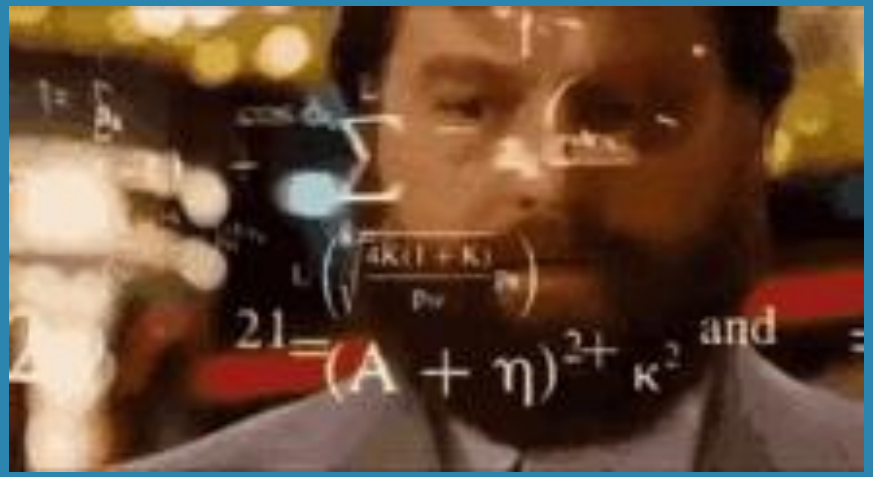
Trace( *Message* [, *TraceSeverity* [, *CustomRecord* [, *TraceOptions* ] ] ]
)

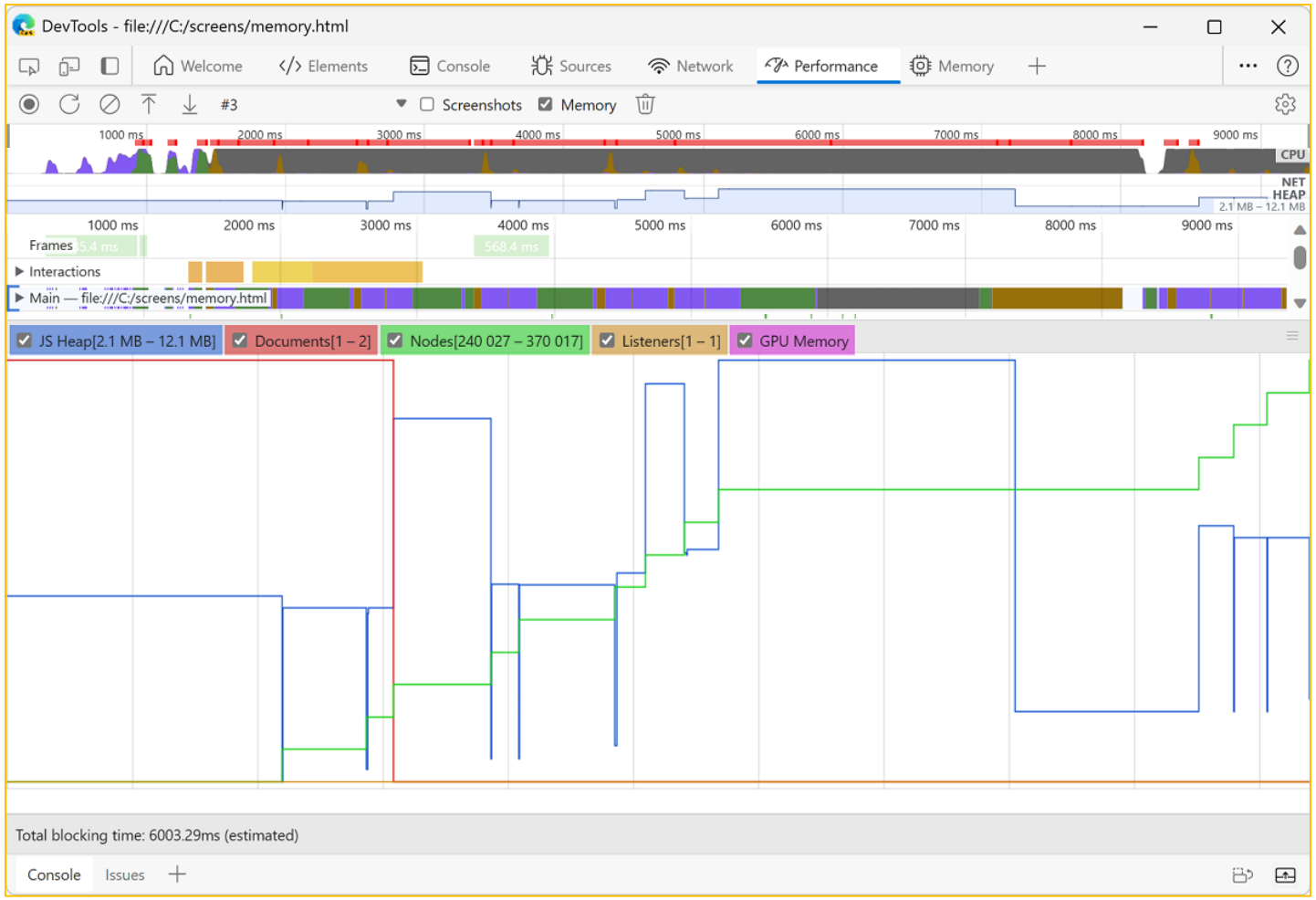
```
Trace("Hello, World!",
    TraceSeverity.Information,
    {
        ID:1,
        Name:"Keith Atherton"
    }
);
```



# Measuring memory consumption graphically

- You can use browser developer tools for your browser to profile memory consumption graphically.



# Agenda

- Code optimisation
- Architectural optimisation
- User experience (UX)
- Monitoring and troubleshooting.

# Thanks! Keith Atherton – Connect with me on LinkedIn

- Power Platform Architect
- Professional developer
- Over 25 years experience
- Microsoft MVP
- Microsoft Certified Trainer
- LinkedIn Learning Instructor
- 11x Microsoft certified
- *Power Platform Community High Five* user group leader
- *Women in Power (Platform)* mentoring program mentor
- *On Air in the Cloud* podcast host