Power Apps + Power Fx
Common functions and reusable patterns

A practical, copy-ready cheat sheet for canvas-app makers • Office Solutions Pro • November 2026

Use this cheat sheet to build a reliable app foundation, reuse user information across screens, display signed-in user photos, work with forms and data, and avoid common formula mistakes.

NOTE Examples use English/United States Power Fx punctuation (commas for arguments, semicolons for action sequencing). Some locales use different separators. Names such as frmRequest, txtSearch, and Project Requests are examples; replace them with the actual control, screen, and column names in your app.

Find the pattern you need

If you need to…	Start here
Open a specific screen from a link	1. App startup and deep linking
Display the user name, photo, or role	2. Signed-in user, profile photo, and reusable headers; 4. Admin identification and role-based display
Decide between global, local, and formula values	3. Variables: global, screen, temporary, and calculated
Find, search, sort, or count records	6. Galleries, queries, search, and sorting
Create or edit a request	7. Forms: new, edit, view, and save, and 8. Creating and updating data with Patch
Add or update attachments	9. Saving attachments: forms and Patch
Validate input and handle failures	10. Validation, errors, and user feedback
Improve load time and avoid missing records	12. Collections, refresh, and performance

The core mental model

- A formula property (for example, Text, Visible, Items, or StartScreen) calculates a value and recalculates when its dependencies change.
- A behavior property (for example, OnSelect, OnVisible, or OnStart) runs actions, such as Set, Notify, Navigate, Patch, or SubmitForm.
- Use named formulas for reusable read-only definitions and automatically derived values. Use Set() or UpdateContext() for values the app must explicitly change during execution.

1 App startup and deep linking

Place the first-screen decision in App.StartScreen. Use App.OnStart only for initialization that truly requires an action. A deep link can select a screen, but it must not bypass data access or authorization.

App.StartScreen • route a URL parameter

Example: a link containing &start=intake opens the intake screen; an unknown or missing value opens the main screen. The start parameter controls navigation only. It does not identify the user or determine their permissions.

```
With(
    { startPage: Lower(Param("start")) },
    Switch(
        startPage,
        "intake", 'Request Intake Form',
        "main",   'Main Screen',
        'Main Screen'
    )
)
```

App.OnStart • capture signed-in user details once

For a multi-screen app that repeats the signed-in user's name, email and photo, capture the User() record once and save the needed values in global variables. This is a session snapshot, independent of deep links and admin permissions. The screen can become visible before nonblocking OnStart completes, so allow for an initial blank value where necessary. In some contexts OnStart is turned off by default. If you don't see it, turn it on in the app's Advanced settings.

gblUserEmail holds a lowercase copy of what User().Email returns, which is the user principal name (UPN), for display. gblUserId holds the Microsoft Entra object ID, which stays the same if the user's UPN or email address changes, so the lookups in Section 4 use it to find the user.

```
With(
    { currentUser: User() },
    Set(gblUserName, currentUser.FullName);
    Set(gblUserEmail, Lower(currentUser.Email));
    Set(gblUserId, currentUser.EntraObjectId);
    Set(gblUserImage, currentUser.Image)
)
```

WATCH OUT App.StartScreen cannot read global variables or collections created by OnStart; it can read Param(), User(), data-source queries, and named formulas. OnStart may continue running after the first screen becomes interactive. Do not use Navigate() in OnStart for new apps.

App.Formulas • read-only definitions and calculated values

Named formulas are defined once and cannot be changed directly with Set() or UpdateContext(). They can hold constants and calculated results. Power Fx evaluates them when needed and updates results when dependencies change; referencing a named formula does not mean its expression is re-executed on every use. Choose a variable when the app must explicitly set a new value at runtime.

```
// App.Formulas — formulas use Name = Expression;
AppDisplayName = "Project Intake";
BrandColor = ColorValue("#243A5A");
CurrentYear = Year(Today());
```

URL parameters are case-sensitive by name and return text. Convert values explicitly when needed, for example Value(Param("page")) for a number. You do not need an extra variable to select the initial screen. If the same route is needed elsewhere, you can optionally save it as gblStartPage. Test links with and without parameters.

What goes where

Property	Typical responsibility
App.StartScreen	Return the initial screen object based on a parameter or directly available value.
App.Formulas	Reusable read-only named definitions. Results recalculate as needed if their dependencies change; they cannot be manually assigned.
App.OnStart	Capture user profile once or run necessary startup actions; do not rely on these globals in StartScreen.
Screen.OnVisible	Prepare screen-specific state when a screen is displayed.
Control.OnSelect	Respond to user actions, such as save, navigate, reset, and delete.

2 Signed-in user, profile photo, and reusable headers

User() returns the current Power Apps user, including the name, UPN, image and Entra object ID. For a multi-screen app with the same header on every page, the first example below captures this information once in App.OnStart and reuses the saved values, and the named-formula alternative after it does the same without OnStart. For a simple page, direct User() expressions remain valid and may be sufficiently efficient.

Property	Example	Use
User().FullName	"Alex Morgan"	Greeting or header name
User().Email	"alex@contoso.com"	User principal name (UPN); not always SMTP mail
User().Image	Blob image URL	Profile photo in Image or Avatar control
User().EntraObjectId	GUID	Stable identity key for matching to an Entra user

Direct User() formulas • simplest example, not the repeated-header default

A beginner can put these expressions directly on the relevant controls. They are valid, and Power Fx only recalculates when required by its dependency model. However, a repeated header with these expressions across many screens is less consistent than referencing one shared user record or a reusable component. Do not assume every reference makes a fresh expensive user-profile call.

Blocks written as Control.Property = formula show where each formula goes: enter only the part after = in that property. They aren't single formulas to paste.

```
lblHeaderName.Text    = User().FullName
lblHeaderEmail.Text   = User().Email
imgHeaderUser.Image   = User().Image
```

Repeated header • use the App.OnStart globals

```
// Already captured once in App.OnStart (Section 1)

// Header name label > Text
gblUserName

// Header email label > Text
gblUserEmail

// Header photo control > Image
gblUserImage
```

Alternative named-formula header • read-only and automatically derived

```
// App.Formulas (alternative, not a Set() variable)
CurrentUser = User();
CurrentUserImage = CurrentUser.Image;

// Header image control > Image
CurrentUserImage

// Header name label > Text
CurrentUser.FullName
```

Named formulas are not user-editable app state: the app cannot Set(CurrentUser, ...). This approach can avoid OnStart initialization and is not inherently slower; the formula engine evaluates and refreshes named formulas when needed. Use the App.OnStart snapshot if you specifically want one captured user record throughout the session.

Modern Avatar control (recommended when available)

Insert an Avatar control into the header. If you initialized the user in App.OnStart, set Name to gblUserName and Image to gblUserImage. The Avatar can display initials when no image is available, eliminating the need for a separate placeholder graphic.

Modern Avatar property values

```
// Modern Avatar control properties
avtHeader.Name  = gblUserName
avtHeader.Image = gblUserImage
```

Create one reusable header component

For a header that appears on every screen, create a canvas component (for example, cmpHeader). Put the title, avatar, signed-in user name, navigation icons, and any role indicator inside it. Add three custom input properties to the component:

Input property	Type	Use inside the component
ProfileImage	Image	Set the internal Image/Avatar Image property to this input.
DisplayName	Text	Set the name label or Avatar Name property to this input.
ShowAdmin	Boolean	Set an optional admin icon Visible property to this input.

On each screen, insert the header component and set its custom input properties to the shared values below. The component controls its own visual layout, while the app supplies one consistent user profile. The values might be briefly blank if App.OnStart has not yet completed.

Component instance custom property values

```
// On the header component instance
ProfileImage = gblUserImage
DisplayName  = gblUserName
ShowAdmin    = gblIsAdmin
```

NOTE Component definitions should not assume they can read the host app's globals. Input properties are clearer and more reusable. Consider building the component in a component library if you plan to reuse it across multiple apps.

Performance guidance: Using User() directly in one or two controls is valid and is not automatically a performance problem. For a shared profile on many screens, capturing the record once simplifies maintenance and avoids duplicate expressions. Use Monitor and real app measurements before attributing slowdowns to User().

User photo and identity cautions

- If no image is returned, use the modern Avatar control's initials or a deliberately designed placeholder. Test with users who have no profile image.

- User().Email is the UPN; it may differ from the person's SMTP address and from 'Primary Email' in the Dataverse Users table. Use a stable identifier such as EntraObjectId for identity mapping when the data model supports it. In Dataverse, it matches the Users table's 'Azure AD Object ID' column.
- Do not display an administrator badge as proof of data authorization. A visible control cannot enforce Dataverse or connector permissions.

3 Variables: global, screen, temporary, and calculated

Pattern	Scope / lifecycle	Best use
Set(gblValue, …)	Whole app, mutable until app closes	Selected record, toggles shared across screens, mutable state
UpdateContext({locValue: …})	Current screen, mutable until app closes	Screen-specific panel state or filters
With({x: …}, formula)	Single expression only	Readable intermediate values and repeated calculations
App.Formulas	Whole app; computed, read-only	Theme constants and calculated expressions that the app should recompute automatically; not state that must be changed with Set().
ClearCollect(colData, …)	Local in-memory table	Small working sets, temporary selections, offline-like UI needs

Global variable • button OnSelect

Read gblSelectedRequest on another screen.

```
Set(gblSelectedRequest, galRequests.Selected);
Navigate('Request Detail Screen', ScreenTransition.Fade)
```

Screen context variable • menu toggle

Set a panel's Visible property to locShowMenu. This is screen-local state.

```
UpdateContext({ locShowMenu: !locShowMenu })
```

Temporary value • With

Use With when a temporary name makes a formula easier to understand, without introducing persistent state. This example is for a label inside a gallery row, where ThisItem is that row.

```
With(
    { remaining: DateDiff(Today(), ThisItem.'Due Date', TimeUnit.Days) },
    If(remaining < 0, "Overdue", Text(remaining) & " days")
)
```

Local collection • small sample table

```
ClearCollect(
    colStatusOptions,
    Table(
        { Label: "Draft", Order: 1 },
        { Label: "Submitted", Order: 2 },
        { Label: "Completed", Order: 3 }
    )
)
```

Naming suggestion: gbl for global variables, loc for context variables, col for collections, and frm/gal/txt/cmb for controls. Names are conventions, not Power Fx requirements.

4 Admin identification and role-based display

A permission table can drive the app interface. In this example, Project Intake Permissions contains a User lookup to the Dataverse Users table and an Admin Yes/No column. Adapt the names to your actual Dataverse schema. A canvas app that uses Dataverse needs Power Apps premium use rights for each user.

App.OnStart • retrieve the user permission record

Three different values describe a user, and they aren't interchangeable:

- User().Email is the signed-in user's UPN. It often looks like an email address, but it isn't always their mailbox address.
- 'Primary Email' is a column in the Dataverse Users table that holds the user's email address. It can differ from the UPN.
- User().EntraObjectId is the user's Microsoft Entra object ID. In the Dataverse Users table, the same value is in the 'Azure AD Object ID' column.

This example finds the signed-in user's row in Users by comparing 'Azure AD Object ID' with gblUserId, which the profile initialization in Section 1 set from User().EntraObjectId. It then finds the permission record whose User lookup is that row. Neither step depends on the UPN matching the email address, and both keep working if either one changes. LookUp returns one record or Blank(). Confirm the schema and the delegation behavior for your tables.

Add the Dataverse Users table to the app as a data source. If you use the named-formula header instead of the App.OnStart snapshot, still set gblUserId in App.OnStart, or use User().EntraObjectId in its place.

```
// App.OnStart — place AFTER the profile initialization,
// which sets gblUserId from User().EntraObjectId.
Set(gblRolesLoaded, false);

// The signed-in user's row in the Dataverse Users table.
// A blank ID finds nothing, rather than a Users row with no ID.
Set(
    gblUserRecord,
    If(
        IsBlank(gblUserId),
        Blank(),
        IfError(
            LookUp(Users, 'Azure AD Object ID' = gblUserId),
            Blank()
        )
    )
);

// No Users row means no permission record, rather than
// matching a permission row whose User lookup is empty.
Set(
    gblRole,
    If(
        IsBlank(gblUserRecord),
        Blank(),
        IfError(
            LookUp(
                'Project Intake Permissions',
                User = gblUserRecord
            ),
            Blank()
        )
    )
);
Set(gblIsAdmin, Coalesce(gblRole.Admin, false));
Set(gblRolesLoaded, true)
```

If your app uses named formulas instead of OnStart, the same user lookup can be one. Power Fx can wait to calculate it until it's needed.

```
// App.Formulas
CurrentUserRecord =
    If(
        IsBlank(User().EntraObjectId),
        Blank(),
        LookUp(Users, 'Azure AD Object ID' = User().EntraObjectId)
    );
```

Admin-only menu control > Visible

```
gblRolesLoaded && gblIsAdmin
```

Admin-only button > DisplayMode

```
If(gblIsAdmin, DisplayMode.Edit, DisplayMode.Disabled)
```

When no role record is found

A missing record evaluates as false. For a large app, use a separate load-error flag so permission-data outages are not mistaken for a legitimate non-admin user.

```
Coalesce(gblRole.Admin, false)
```

SECURITY A canvas app permission lookup affects what users see. It is NOT a security boundary. Enforce actual record access in Dataverse security roles, ownership/sharing, or the back-end service. A hidden button does not stop someone from accessing data they already have permission to read or change.

More reliable identity matching

An email-based lookup is easy to start with, but an address or UPN can change, and the UPN in User().Email isn't always the address in 'Primary Email'. Comparing 'Primary Email' with User().Email works only while the two happen to be the same. Prefer the Entra object ID, as in the example above. If your permissions table stores the ID itself instead of a User lookup, compare that column with gblUserId. Use the UPN only when the column actually stores UPNs.

5 Navigation and record deep links

Open a screen • button OnSelect

```
Navigate('Request Intake Form', ScreenTransition.Fade)
```

Return to previous screen • button OnSelect

```
Back()
```

Pass a selected record without a global • gallery OnSelect

Put this in the gallery's OnSelect, or in a button inside the gallery template, where ThisItem is the row the user selected. From a button outside the gallery, use galRequests.Selected instead of ThisItem. On Request Detail Screen, locRequest is a screen context variable. Use locRequest in the form Item property.

```
Navigate(
    'Request Detail Screen',
    ScreenTransition.Fade,
    { locRequest: ThisItem }
)
```

Read a record ID from a deep link • App.Formulas

Use when your record ID is a GUID. An invalid or missing value becomes Blank().

```
DeepLinkRequestId =
    IfError(GUID(Param("id")), Blank());
```

Detail form > Item

Example assumes the primary key column is Project Request (a GUID). Replace with the real primary key and table.

```
If(
    IsBlank(DeepLinkRequestId),
    Blank(),
    LookUp(
        'Project Requests',
        'Project Request' = DeepLinkRequestId
    )
)
```

Deep-link test: verify the record exists, verify the viewer has access, then show a meaningful "not found or no access" state. Do not treat the URL parameter as a trusted authorization check.

6 Galleries, queries, search, and sorting

Gallery > Items • show matching records

Choice syntax depends on your source. Replace the example choice column and option set with the ones in your app.

```
Filter(
    'Project Requests',
    'Request Status' = 'Request Status Choice'.Submitted
)
```

Gallery > Items • search by title

StartsWith can delegate for supported text columns and data sources; check warnings for your connector.

```
Filter(
    'Project Requests',
    IsBlank(txtSearch.Text) ||
        StartsWith('Request Title', txtSearch.Text)
)
```

Gallery > Items • filter then sort

The sort string must match a valid column name supported by the data source. Dataverse createdon is an example logical name; verify your table and control version.

```
SortByColumns(
    Filter(
        'Project Requests',
        StartsWith('Request Title', txtSearch.Text)
    ),
    "createdon",
    SortOrder.Descending
)
```

LookUp • return a single record

Use LookUp for one record, not First(Filter(...)). Here gblRequestId stands for a request ID your app already holds, for example one read from a link.

```
LookUp('Project Requests', 'Request ID' = gblRequestId)
```

As • disambiguate record scope

Illustrative: related-column queries may not delegate in every connector. Explicit As naming can make nested formulas easier to follow.

This example finds the requests the signed-in user created, by comparing the creator's Entra object ID with gblUserId.

```
Filter(
    'Project Requests' As request,
    request.'Created By'.'Azure AD Object ID' = gblUserId
)
```

Core query distinctions

Function	Returns	Typical scenario
Filter(table, condition)	Table (zero or more rows)	Gallery of matching requests
LookUp(table, condition)	One record (or blank)	Get current user's permission row
Search(table, text, column)	Table	Search a text field (syntax and delegation vary)
SortByColumns(table, name, order)	Table	Sort a gallery by a known column
CountRows(table)	Number	Small/local table count; verify data-source counting support

WATCH OUT Avoid Lower(Column), Text(Column), and similar conversions applied to large data-source columns in Filter unless you have verified delegation. The yellow delegation warning means a formula may operate on only a limited local subset: the first records up to the app's Data row limit, which is 500 by default and can be raised to at most 10,000. Raising it doesn't make the query delegable. It does not mean "slightly slower."

7 Forms: new, edit, view, and save

Form controls provide validation and an integrated save workflow. For straightforward record editing, they are usually easier than writing Patch for every field.

Action	Where to put it	Power Fx
New request button	OnSelect	NewForm(frmRequest); Navigate('Request Intake Form')
Edit request button	OnSelect	EditForm(frmRequest); Navigate('Request Intake Form')
Read-only mode	OnSelect	ViewForm(frmRequest)
Save button	OnSelect	SubmitForm(frmRequest)
Cancel button	OnSelect	ResetForm(frmRequest); Back()
Edit form > Item	Item	gblSelectedRequest or locRequest
Save button enabled	DisplayMode	If(frmRequest.Valid, DisplayMode.Edit, DisplayMode.Disabled)

Form > OnSuccess • post-save actions

LastSubmit contains the last successfully submitted record. Put dependent actions here, not immediately after SubmitForm() on the button.

```
Set(gblSavedRequest, frmRequest.LastSubmit);
Notify("Request saved.", NotificationType.Success);
Navigate('Main Screen', ScreenTransition.Fade)
```

Form > OnFailure • show the failure

```
Notify(
    Coalesce(frmRequest.Error, "Unable to save the request."),
    NotificationType.Error
)
```

Hide admin section while creating a new record

Set the admin section container Visible property to this expression, adding && gblIsAdmin if appropriate.

```
frmRequest.Mode <> FormMode.New
```

WATCH OUT SubmitForm validates required fields and fires OnSuccess or OnFailure when the save completes. A button formula such as SubmitForm(frmRequest); Navigate(...) can navigate away before the save outcome is known.

8 Creating and updating data with Patch

Patch is useful for updating selected fields, writing one record without a form, or handling a specialized save. Use SubmitForm for ordinary editable forms when possible.

Create a record • Patch + Defaults

Defaults must refer to the same data source you are patching.

```
Patch(
    'Project Requests',
    Defaults('Project Requests'),
    {
        'Request Title': txtTitle.Text,
        'Requested Date': Today()
    }
)
```

Update an existing record

The base record should come from the same data source, for example a gallery or LookUp against Project Requests.

```
Patch(
    'Project Requests',
    galRequests.Selected,
    { 'Request Title': txtTitle.Text }
)
```

Handle a Patch error

IfError catches a failed Patch; it does not replace the need for back-end permissions.

```
IfError(
    Patch(
        'Project Requests',
        galRequests.Selected,
        { 'Request Title': txtTitle.Text }
    ),
    Notify(
        "Save failed: " & FirstError.Message,
        NotificationType.Error
    )
)
```

Delete with confirmation

For a consequential delete, use two steps. The Delete button remembers the chosen record and opens a confirmation panel, and only the panel's Confirm button removes it. Because the record is kept in a context variable, the delete applies to the record the person chose, even if the gallery selection changes while the panel is open.

```
// Delete button > OnSelect (outside the gallery; inside it, use ThisItem)
UpdateContext({ locDeleteTarget: galRequests.Selected, locConfirmDelete: true })

// Delete button > DisplayMode
If(IsBlank(galRequests.Selected), DisplayMode.Disabled, DisplayMode.Edit)

// Confirmation panel > Visible
locConfirmDelete

// Confirmation label > Text
"Delete """ & locDeleteTarget.'Request Title' & """?"
```

The Confirm button closes the panel and clears the target before it deletes, so a second click can't delete again.

```
// Confirm button > OnSelect
With(
    { target: locDeleteTarget },
    UpdateContext({ locConfirmDelete: false, locDeleteTarget: Blank() });
    If(
        IsBlank(target),
        Notify("Select a request to delete.", NotificationType.Warning),
        IfError(
            Remove('Project Requests', target);
            true,
            Notify("Delete failed: " & FirstError.Message, NotificationType.Error),
            Notify("Request deleted.", NotificationType.Success)
        )
    )
)

// Confirm button > DisplayMode
If(IsBlank(locDeleteTarget), DisplayMode.Disabled, DisplayMode.Edit)
```

Cancel resets the same two values. Context variables keep their values when the person leaves the screen, so reset them when the screen opens too.

```
// Cancel button > OnSelect, and Screen > OnVisible
UpdateContext({ locConfirmDelete: false, locDeleteTarget: Blank() })
```

If Remove fails, for example because the person doesn't have permission to delete the record or it can no longer be found, the message shows the reason the data source gave. Nothing is deleted twice, because the panel is already closed.

9 Saving attachments: forms and Patch

Uploading an attachment is not the same as patching a text or choice field. The supported approach depends on the data source and whether you use the classic or modern Attachments control. The formulas below use separate sample names for SharePoint and Dataverse to avoid mixing those two data models.

Choose the right attachment pattern

Scenario	Save method	Key limitation
Classic Attachments: SharePoint list or Dataverse notes	SubmitForm(frmRequest)	File adds/removals are saved through an Edit form.
Modern Attachments: SharePoint list (preview)	Patch(list, item, {Attachments: control.Attachments})	Can run outside a form; preview feature, not a production default.
Modern Attachments: Dataverse File column (preview)	Patch(table, row, {file column: {FileName, Value}})	One file per File column; uploading a new file replaces it.

Classic Attachments • save with an Edit form

Best choice for an established production app. Add the Attachments field to the Edit form, keep the generated data card and its Update property, and use the following formula on the Save button. This example assumes frmRequest is bound to a SharePoint list or a Dataverse table with attachments enabled.

```
SubmitForm(frmRequest)
```

Put success and failure feedback on the form itself, rather than notifying immediately after SubmitForm. These events run after the save succeeds or fails.

```
// frmRequest.OnSuccess
Notify("Request and attachments saved.", NotificationType.Success)

// frmRequest.OnFailure
Notify(frmRequest.Error, NotificationType.Error)
```

WATCH OUT Do not use Patch(list, item, {Attachments: ClassicControl.Attachments}) as a general replacement for SubmitForm. The classic control requires a form to save additions and removals.

Modern SharePoint Attachments • patch an existing item (preview)

This example requires the modern Attachments control and a SharePoint list named SP Requests with attachments enabled. Set galSPRequests.Items to the list. Set attSPFiles.Items to galSPRequests.Selected.Attachments. Place the formula below on a Save attachments button (OnSelect).

```
// Modern control: attSPFiles.Items
galSPRequests.Selected.Attachments
```

```
If(
    IsBlank(galSPRequests.Selected),
    Notify("Select a request first.", NotificationType.Warning),
    IfError(
        Patch(
            'SP Requests',
            galSPRequests.Selected,
            { Attachments: attSPFiles.Attachments }
        );
        true,
        Notify(
            "Attachments could not be saved: " & FirstError.Message,
            NotificationType.Error
        ),
        Notify("Attachments saved.", NotificationType.Success)
    )
)
```

Adding or removing files in the modern control stages changes; Patch actually saves them. Keep the selected item unchanged while edits are pending. This example updates an existing SharePoint item, not a new unsaved list item.

WATCH OUT The modern Attachments control remains a PREVIEW feature as of November 2026. Microsoft does not recommend preview features for production. Check availability and test behavior before using this approach.

Dataverse File column • patch one uploaded file (preview)

A Dataverse File column holds one file per record. The modern control supplies the selected file, but the Patch value must be a FileName/Value record rather than the full Attachments table. This sample uses the Dataverse table Project Documents, its Document File column, a gallery called galDocuments, and a modern control called attDocumentFile. Set its Items property to Blank() and MaxAttachments to 1.

```
// Modern control: attDocumentFile.Items
Blank()

// attDocumentFile.MaxAttachments
1
```

```
If(
    IsBlank(galDocuments.Selected),
    Notify("Select a document record first.", NotificationType.Warning),
    IsEmpty(attDocumentFile.Attachments),
    Notify("Choose a file first.", NotificationType.Warning),
    IfError(
        Patch(
            'Project Documents',
            galDocuments.Selected,
            {
                'Document File': {
                    FileName: First(attDocumentFile.Attachments).Name,
                    Value: First(attDocumentFile.Attachments).Value
                }
            }
        );
        true,
        Notify(
            "File could not be saved: " & FirstError.Message,
            NotificationType.Error
        ),
        Reset(attDocumentFile);
        Notify("File saved.", NotificationType.Success)
    )
)
```

Saving another file in the same Dataverse File column replaces the prior file. If the app needs multiple supporting documents per record, use Dataverse notes/attachments with a form or design a related document table instead. If you are storing documents in a SharePoint document library, that is a separate upload workflow.

Attachment checks before publishing

- Verify that attachments are enabled on the SharePoint list or Dataverse table and that the user has permission to update the record.
- Check each file against both control-level limits and data-source limits; accepted file types in the UI do not replace server-side policies.
- Test adding, removing, and replacing files; confirm persistence by reopening the record after saving.
- For the classic control, validate OnSuccess and OnFailure on the form. For the modern control, handle Patch errors and unsaved changes.

10 Validation, errors, and user feedback

Function / property	What it solves
Coalesce(a, b)	First value that isn't blank or an empty string; returns blank if every value is blank or empty, so Coalesce(x, "") doesn't turn a blank into ""
IsBlank(value)	Blank scalar/string check
IsEmpty(table)	No records in a table or collection
If(condition, yes, no)	Two-way conditional decision
Switch(value, option, result, …)	Clear branching over discrete values
IfError(value, replacement)	Handle formula/runtime errors
Notify(message, type)	Show brief user feedback
IsMatch(text, pattern)	Validate text against a pattern, for example email

Required-field validation • button OnSelect

```
If(
    IsBlank(TrimEnds(txtTitle.Text)),
    Notify("Enter a request title.", NotificationType.Warning),
    SubmitForm(frmRequest)
)
```

Switch • readable display text

Replace options with the choice values in your data source. This example is for a label inside a gallery row, where ThisItem is that row.

```
Switch(
    ThisItem.Priority,
    'Priority Choice'.High, "High priority",
    'Priority Choice'.Medium, "Medium priority",
    "Normal priority"
)
```

App.OnError • central diagnostics

This logs the error and rethrows it so the default error message can still appear. Validate your telemetry and avoid logging private data.

```
Trace(
    "Power Fx error: " & FirstError.Message
);
Error(FirstError)
```

11 Date, time, text, and common UI expressions

Need	Example formula
Today's date	Today()
Current date and time	Now()
Seven days from today	DateAdd(Today(), 7, TimeUnit.Days)
Days until due date, in a gallery row	DateDiff(Today(), ThisItem.'Due Date', TimeUnit.Days)
Format date/time	Text(Now(), "[$-en-US]yyyy-mm-dd hh:mm")
Normalize user-entered text	Lower(TrimEnds(txtEmail.Text))
Join text values	gblUserName & " | " & gblUserEmail
Convert text to number	IfError(Value(txtAmount.Text), Blank())
Disable the Save button	If(frmRequest.Valid, DisplayMode.Edit, DisplayMode.Disabled)
Show a menu for admins only	gblIsAdmin
Loading label visible	!gblRolesLoaded

Power Fx date/time values and connected data sources can have different time-zone behavior. Verify whether your Dataverse column is User Local, Time-Zone Independent, or Date Only before building time-sensitive conditions.

12 Collections, refresh, and performance

Refresh changed data • button OnSelect

Useful when another person or process might have changed records; unnecessary frequent refreshing can slow the app.

```
Refresh('Project Requests')
```

Small working set • ClearCollect

Use only when you know the resulting row count is small. Collection loading is not a general solution for large or growing tables. This example keeps the signed-in user's own requests, using gblUserRecord, their row in the Users table, from Section 4. Run it only after App.OnStart has set gblUserRecord, for example once gblRolesLoaded is true.

```
ClearCollect(
    colMyRequests,
    Filter('Project Requests', 'Created By' = gblUserRecord)
)
```

Parallel independent calls • Concurrent

Only when the two loads are independent. Do not put dependent assignments in parallel branches.

```
Concurrent(
    ClearCollect(colStatusOptions, 'Status Options'),
    ClearCollect(colPriorityOptions, 'Priority Options')
)
```

Delegation: the most common data bug

Delegation means a filter, search, sort, or aggregation runs in the data source rather than on a limited local sample. A formula can work perfectly in a tiny test table and omit records in production when it does not delegate.

- Check the delegation warning and Microsoft's connector-specific delegation table for Dataverse or SharePoint.
- Prefer direct comparisons and supported operations on columns. Apply formatting transformations to input values rather than wrapping the data-source column.

- Raising the Data row limit (Settings > General, 500 by default, at most 10,000) doesn't solve delegation. It only moves the cut-off: a table with more records than the limit still gives incomplete results, and a higher limit can slow the app. Change the query so it delegates instead.
- Do not use ClearCollect to bypass a delegation warning on a large list. Test with data volumes beyond the app's row limit. Setting the Data row limit to 1 while testing makes a formula that doesn't delegate obvious, because it returns at most one record.
- Keep startup data loading small. Named formulas let Power Fx defer and recalculate read-only values when needed; use OnStart variables where a captured session value or explicit mutable state is intentional.

13 Quick copy-and-paste formula index

Where / purpose	Formula
App.StartScreen	Switch(Lower(Param("start")), "intake", 'Request Intake Form', 'Main Screen')
Image.Image	gblUserImage
Avatar.Name	gblUserName
Avatar.Image	gblUserImage
Header label.Text	gblUserName
Admin icon.Visible	gblIsAdmin
Menu toggle.OnSelect	UpdateContext({ locShowMenu: !locShowMenu })
Button.OnSelect	Navigate('Main Screen', ScreenTransition.Fade)
Cancel.OnSelect	ResetForm(frmRequest); Back()
Save.OnSelect	SubmitForm(frmRequest)
Edit section.Visible	frmRequest.Mode <> FormMode.New
Gallery.Items	Filter('Project Requests', StartsWith('Request Title', txtSearch.Text))
Gallery.OnSelect	Set(gblSelectedRequest, ThisItem)
Error message	Notify("Unable to save.", NotificationType.Error)
Format due-date label in a gallery row	Text(ThisItem.'Due Date', "[$-en-US]mmm d, yyyy")
Classic attachments (Save button)	SubmitForm(frmRequest)
Modern SharePoint attachments (preview)	Patch('SP Requests', galSPRequests.Selected, { Attachments: attSPFiles.Attachments })
Dataverse File column (preview)	Patch('Project Documents', galDocuments.Selected, { 'Document File': { FileName: First(attDocumentFile.Attachments).Name, Value: First(attDocumentFile.Attachments).Value } })

14 Common mistakes to avoid

Mistake	Better pattern
Using Navigate in App.OnStart	Use App.StartScreen for the initial screen selection.
Using OnStart globals in StartScreen	Use Param(), User(), or App.Formulas; global variables are not available there.
Assuming User().Email is SMTP mail	Treat it as the UPN. To find a Dataverse user, prefer 'Azure AD Object ID' = User().EntraObjectId. Matching 'Primary Email' with User().Email works only while the two are the same.
Showing admin controls as authorization	Enforce Dataverse or service-side security as well.
Using First(Filter()) for one record	Use LookUp() for an explicit single-record request.
Navigating immediately after SubmitForm	Navigate from Form.OnSuccess after the save succeeds.
Ignoring delegation warnings	Change the query or design; test with large record counts.
Applying Lower() to source columns in Filter	Normalize input once; keep source-column operations delegable.

Mistake	Better pattern
Creating many global variables by habit	Use a single User() snapshot for repeated profile controls; use named formulas for read-only calculated values.
Copying formulas without adjusting schema	Confirm field types, option names, data sources, control types, and locales.
Patching classic form attachments directly	Use SubmitForm; direct SharePoint attachment Patch requires the modern preview control.

Sources

The formulas are examples for canvas apps and must be adapted to actual controls, data-source types, and security requirements. All formulas use the English (United States) Power Fx locale.

- App object, StartScreen, OnStart and App.Formulas — Microsoft Learn
- User() profile data and image — Microsoft Learn
- Modern Avatar control — Microsoft Learn
- Canvas components — Microsoft Learn
- Variables and state — Microsoft Learn
- Large canvas apps, named formulas and when to use Set() — Microsoft Learn
- Filter, Search and LookUp — Microsoft Learn
- Forms and SubmitForm — Microsoft Learn
- Patch — Microsoft Learn
- Power Apps delegation overview — Microsoft Learn
- Launch and Param — Microsoft Learn
- Error handling and IfError — Microsoft Learn
- Classic Attachments control — Microsoft Learn
- Modern Attachments control (preview) — Microsoft Learn
- Dataverse File columns — Microsoft Learn

Office Solutions Pro | Updated November 2026. Formula behavior and control properties may change. Validate performance with Monitor.