PlatformForge LABS

POWER FX CHEAT SHEET

Power Fx expressions to know first

Common patterns for Canvas Apps, SharePoint and Dataverse

Build formulas around four jobs

Read
Filter, LookUp, Search

Write
SubmitForm, Patch

State
Set, UpdateContext

Control
Navigate, If, With

Cleaned up from the original list

- Use LookUp, not Lookup.
- Use TimeUnit.Months and TimeUnit.Days in date formulas.
- For SharePoint Choice columns, patch {Value: "Complete"}.
- Power Fx filtering is not security. Enforce access at the data source.

The safe starter set

`User()` `Filter()` `LookUp()` `Patch()` `With()`

`IfError()` `Coalesce()` `Param()` `App.Formulas`

Start with delegable reads, simple writes, explicit error handling and named reusable logic. Avoid turning every problem into a collection, especially when your data source will grow.

PlatformForge LABS

CORE PATTERNS

User, variables, navigation and forms

The expressions you will use in nearly every canvas app.

1 Current logged-in user

```
User().Email
User().FullName
User().Image
```

Use Lower(Trim(User().Email)) when matching email values.

2 Global variable

```
Set(varUserEmail, Lower(User().Email))
Set(varMode, "Browse")
```

Available across screens. Do not use for security.

3 Screen variable

```
UpdateContext({
    locSpinner: true,
    locMode: "Edit"
})
```

Screen-scoped. Good for UI state like spinners, modes and flags.

4 Navigate with context

```
Navigate(
    scrDetail,
    ScreenTransition.Fade,
    { locItem: ThisItem }
)
```

Cleaner than loading globals before every screen.

5 Submit a form

```
SubmitForm(frmAdvance)
```

Use when the form structure matches the data. Put success logic in OnSuccess and error logic in OnFailure.

6 Deep link parameter

```
Set(varAdvanceId, Value(Param("advanceID")));
If(!IsBlank(varAdvanceId),
    Navigate(scrDetails))
```

Useful for approval links, admin actions and context-specific entry.

PlatformForge LABS

READ DATA

Filter, LookUp, Search and delegation

Find the right rows without breaking performance at scale

Filter rows

```
Filter(
    Accounts,
    Dept = varDept &&
    Status.Value = "Live"
)
```

Returns a table. Keep conditions delegable where possible.

Look up one value

```
LookUp(
    Colleagues,
    EmployeeID = varEmployeeID,
    FullName
)
```

Returns the first matching record or selected value.

Search text safely

```
Filter(
    Colleagues,
    StartsWith(Title, txtSearch.Text)
)
```

For large SharePoint lists, StartsWith is usually safer than contains-style search.

Sort and shape

```
SortByColumns(
    Filter(Assets, Dept = varDept),
    "Title",
    SortOrder.Ascending
)
```

Filter first, shape later. Do not hide delegation issues.

Enterprise rule: design for delegation first

Do	Filter by indexed columns, equality, StartsWith, date ranges and simple And/Or patterns.
Avoid	Pulling data into ClearCollect just to filter locally when the source can grow.
Check	Blue delegation warnings. A formula that works on 200 rows can silently fail at 20,000.

PlatformForge LABS

WRITE DATA

Patch, SubmitForm and error handling

Use Patch declaratively and wrap writes that matter

1 Create a record

```
Patch(
    Assets,
    Defaults(Assets),
    {
        Title: txtAsset.Text,
        Depot: varDepot
    }
)
```

2 Update selected record

```
Patch(
    Assets,
    galAssets.Selected,
    { Status: "Available" }
)
```

ⓘ The base record should point back to the same data source.

3 Patch with result

```
With({ saved: Patch(Assets,
     Defaults(Assets),
     { Title: txtAsset.Text }) },
    Notify("Saved ID: " & saved.ID,
        NotificationType.Success)
)
```

4 Error-handled save

```
IfError(
   Patch(Assets, ThisItem, { Status: "Checked" }),
   Notify("Save failed: " & FirstError.Message,
      NotificationType.Error),
   Notify("Saved", NotificationType.Success)
)
```

5 Bulk update pattern

```
ForAll(colUpdates As row,
    Patch(Assets, LookUp(Assets,
        ID = row.ID),
        { Status: row.Status })
)
```

ⓘ Use carefully. Consider flows or Dataverse actions for bigger workloads.

6 Reset UI after save

```
ResetForm(frmAsset);
Reset(txtSearch);
Refresh(Assets);
UpdateContext({ locSpinner: false })
```

ⓘ Refresh only when needed. It can be expensive on large screens.

PlatformForge LABS

PACKED FIELDS

Choice, lookup and combo-box patterns

Use the field shape the connector expects, not just display text

1. SharePoint Choice patch

```
Patch(
   Requests,
   ThisItem,
   { Status: { Value: "Complete" } }
)
```

Choice columns are records. Patch the Value.

2. SharePoint Choice filter

```
Filter(
   Requests,
   Status.Value = ddStatus.Selected.Value
)
```

For Dataverse Choices the shape differs.
Check the column type.

3. Choices in controls

```
Choices(Requests.Status)

ThisItem.Status
```

Choices() is useful for lookup and choice-backed controls.

4. Dataverse lookup patch

```
Patch(
   Orders,
   ThisItem,
   { Customer: cmbCustomer.Selected }
)
```

Patch the selected related record,
not just the display text.

5. Create or update screen

```
Patch(
   Assets,
   If(FormMode.New, Defaults(Assets), galAssets.Selected),
   { Title: txtTitle.Text }
)
```

6. Form updates pattern

```
Patch(
   Assets,
   galAssets.Selected,
   frmAsset.Updates
)
```

Useful when a form captures changes
but you still need custom save logic.

PlatformForge LABS

SHAREPOINT PERSON

Person picker patterns

The field is a record, so patch Claims, DisplayName and Email

Patch current user into a person field

```
Patch(
    Requests,
    Defaults(Requests),
    {
        Title: txtTitle.Text,
        Owner: {
            Claims: "i:0#.f|membership|" & Lower(User().Email),
            DisplayName: User().FullName,
            Email: Lower(User().Email)
        }
    }
)
```

Design for SharePoint person columns. Multi-person fields need a table of person records.

Default current user in a person picker

```
If(
    Form1.Mode = FormMode.New,
    [ {
        Claims: "i:0#.f|membership|" & Lower(User().Email),
        DisplayName: User().FullName,
        Email: Lower(User().Email)
    } ],
    Parent.Default
)
```

Set this on DefaultSelectedItems. Wrap the record in square brackets when the control expects a table.

PlatformForge LABS

MODERN MUST-KNOW

Newer patterns worth learning early

Cleaner apps, faster load times and fewer support tickets

Named formulas

```
// App.Formulas
PreferredLabel = Lower(User().Email);
IsToday = Today()
```

- Prefer reusable formulas over too much App.OnStart logic.

Concurrent loading

```
Concurrent(
    ClearCollect(colReports, Reports),
    ClearCollect(colShifts, Shifts)
)
```

- Only use for independent calls. Do not use where order or dependency matters.

Coalesce fallback

```
Coalesce(
    ThisItem.Comments,
    "No comments captured"
)
```

- Cleaner than nested If(IsBlank(...)) patterns.

Record scope clarity

```
ForAll(
    colPeople As person,
    Collect(colEmails, person.Email)
)
```

- Use As when ThisRecord or ThisItem becomes hard to read.

Parse JSON safely

```
With({ parsed: ParseJSON(jsonText) },
    Text(parsed.value.phoneNumber)
)
```

- Dynamic values need converting before real use: Text(), Value(), Boolean() and DateValue().

Avoid stale collections

```
Filter(Assets, Dept = varDept)
// Use ClearCollect only for small
reference data.
```

- Collections are snapshots. They do not auto-refresh.

PlatformForge LABS

DATES AND GUARDRAILS

Date helpers and production rules

Useful in absence, rota, audit, payroll and warehouse apps

1. First day of month

```
Date(
    Year(DatePicker1.SelectedDate),
    Month(DatePicker1.SelectedDate),
    1
)
```

Note: Swap DatePicker1.SelectedDate with Today() if you always want the current month.

2. Last day of month

```
With({ firstOfMonth:
  Date(Year(DatePicker1.SelectedDate),
    Month(DatePicker1.SelectedDate), 1) },

  DateAdd(DateAdd(firstOfMonth, 1,
    TimeUnit.Months), -1, TimeUnit.Days)
)
```

3. Start of week

```
DateAdd(
    DatePicker1.SelectedDate,
    1 - Weekday(DatePicker1.SelectedDate,
        StartOfWeek.Monday),
    TimeUnit.Days
)
```

4. Days between dates

```
DateDiff(
    StartDatePicker.SelectedDate,
    EndDatePicker.SelectedDate,
    TimeUnit.Days
)
```

Note: Useful for absence duration, audit ageing and SLA checks.

Production guardrails

Security	Do not use Power Fx as the only security boundary. Enforce permissions in Dataverse, SharePoint or the API layer.
Scale	Design delegable filters before adding controls. Index SharePoint columns used in common filters.
ALM	Keep formulas reusable. Use naming conventions, comments and reusable formulas for team-owned apps.
Support	Document formulas, flows and dependencies. Include patch, delegation, code optimisation, App.Formulas, ParseJSON and error handling.