



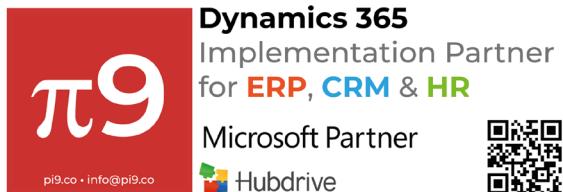
Agents for change//

A.I. POWER PLATFOR M POWER LONDO

MAY 24TH 2025

- A.I. AGENTS BOOTCAMP
- POWER PLATFORM DEVELOPER BOOTCAMP





Sponsors

THANK YOU!

The POWER COMMUNITY logo is set against a red background with a yellow border. It features the word "POWER" in large, white, blocky letters. To the left of "POWER" is a circular icon containing a rocket ship and the text "POWER PLATFORM" and "POWER COMMUNITY". Below "POWER" is the word "COMMUNITY" in a smaller, white sans-serif font.



Microsoft

Microsoft Power Platform
Power Platform London 2023

Senior Power Platform Lead @ Microsoft

Cathrine Bruvold

**Optimizing Canvas Apps for
maintainability, reusability,
and performance**



24th May



Microsoft

AI POWER PLATFORM
POWER UP LONDON 2025

Senior Cloud Architect @ Microsoft

Daniel Laskewitz

Optimizing Canvas Apps for
maintainability, reusability,
and performance



24th May



Agenda

✨ Introduction

✨ Building Maintainable Software book

- ✓ 10 guidelines

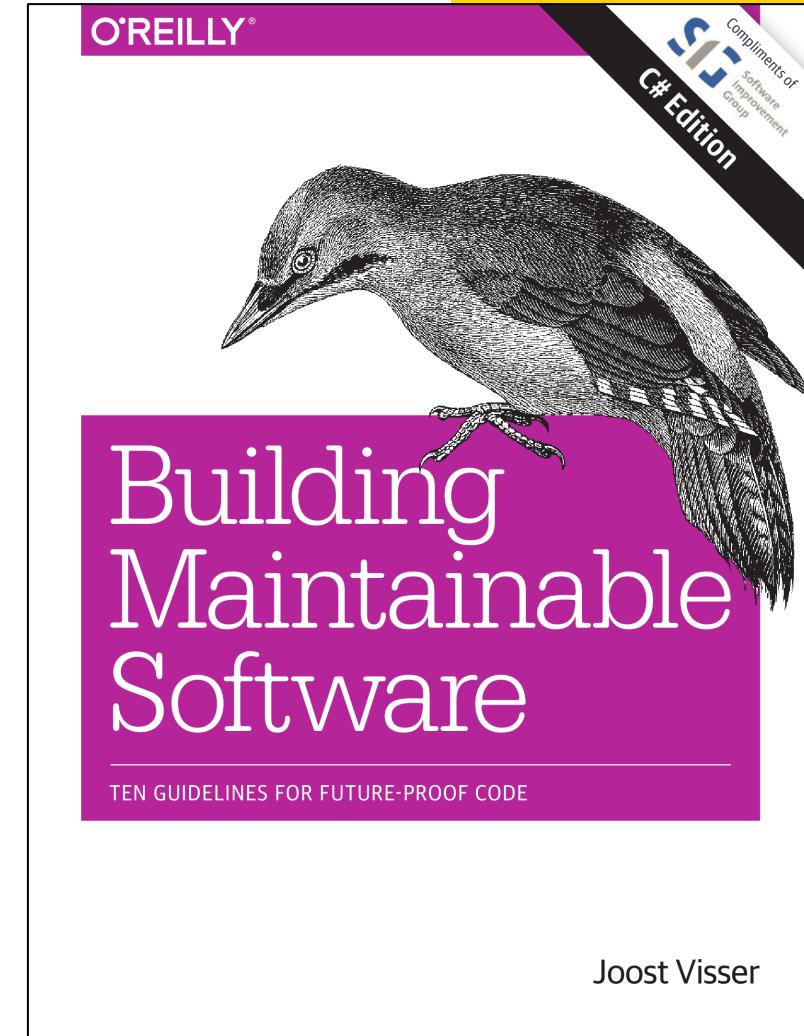
✨ Canvas Apps Best Practices

✨ Resources

Building Maintainable Software

Guidelines for future-proof code

1. Write short units of code
2. Write simple units of code
3. Write code once
4. Keep unit interfaces small
5. Separate concerns in modules
6. Couple architecture components loosely
7. Keep architecture components balanced
8. Keep your codebase small
9. Automate development pipeline and tests
10. Write clean code



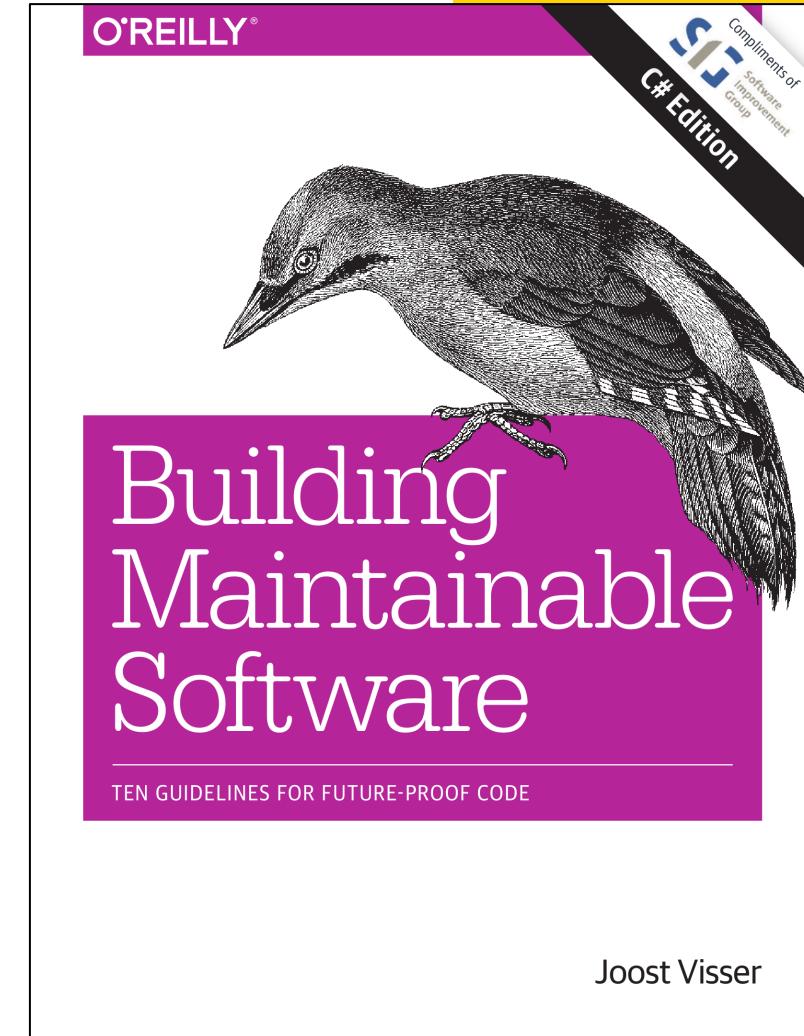
1

Write short units of code



Guideline:

- Limit the length of code units to 15 lines of code.
- Do this by not writing units that are longer than 15 lines of code in the first place, or by splitting long units into multiple smaller units until each unit has at most 15 lines of code.
- This improves maintainability because small units are easy to understand, easy to test, and easy to reuse.



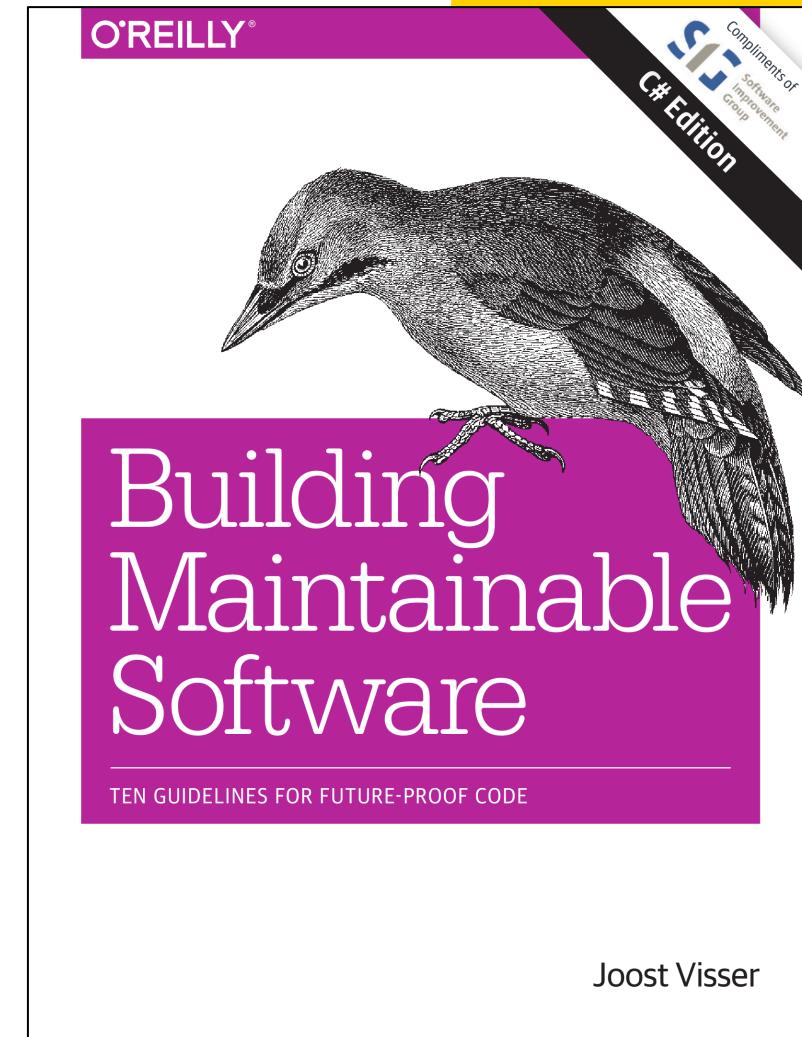
2

Write simple units of code



Guideline:

- Limit the number of branch points per unit to 4.
- Do this by splitting complex units into simpler ones and avoiding complex units altogether.
- This improves maintainability because keeping the number of branch points low makes units easier to modify and test.



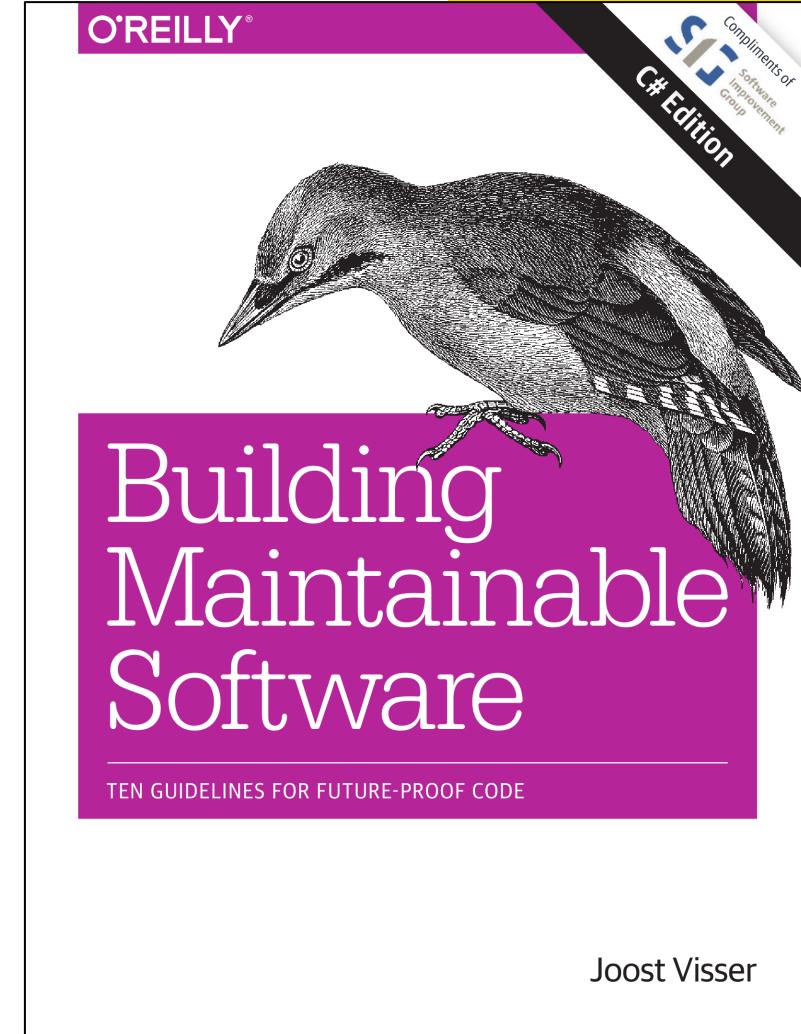
3

Write code once



Guideline:

- Do not copy code.
- Do this by writing reusable, generic code and/or calling existing methods instead.
- This improves maintainability because **when code is copied, bugs need to be fixed at multiple places**, which is inefficient and error-prone.



The App of Horrors

Built **before** best practices existed...



DEMO

Lack of maintainability

Hard coded values, repetition of formulas, poor naming conventions

Bad performance

Unnecessary loading of data, hitting delegation limitations, nested lookups and galleries

Little to no reusability

Code duplication, repetition and spaghetti logic

User experience

Confusing navigation, information overload and static layouts

Delegation

Maintainability, performance and data accuracy

Data processing

Offload tasks for data transformation to the server-side

Large datasets – 500+ records

Delegable data sources and syntax.

Build for performance

Avoid loading unnecessary data – lazy loading



Delegable

Use Filter(), Sort(), StartsWith()

Locally processed – Collect(), ClearCollect(), Search()

Power CAT Tools

1 Write short units of code

Code review

AI solution documentation

Risk assessment

...and more



DEMO

The screenshot shows a Microsoft Power Platform blog post titled "Elevate your Microsoft Power Platform development experience with Power CAT Tools" by Denise Moran, published on March 4, 2025. The post discusses the launch of Power CAT Tools, which provide automated tools for code review, AI documentation generation, and risk assessment. The page includes a sidebar with featured kits like Copilot Studio kit and Creator kit, and links for documentation, frequently asked questions, and support.

Reduce the amount of code

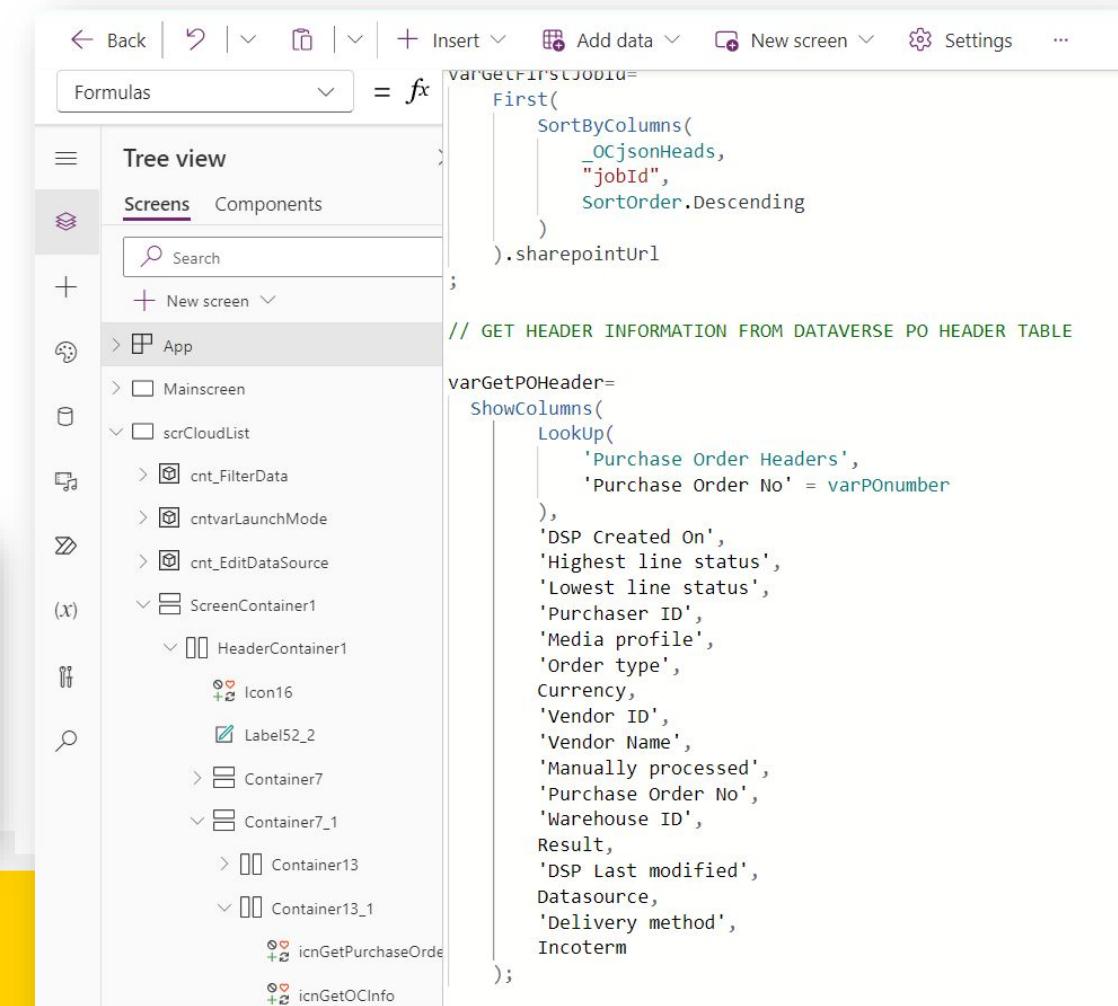
1 Write short units of code, 3 Write code once

Named Formulas

Set on **App.Formulas** containing the code you want to execute and reference the defined name

Enables developers to reference a short formula name rather than the entire syntax

```
// get data from dataverse Purchase order header table
clearCollect(
    colCurrentHeadCollection,
    varGetPOHeader // Set by named formulas
),
```



Reduce the amount of code

- ② Write simple units of code,
- ④ Keep units small

With()

Simplify complex formulas with temporary variables that only functions within the context of With()



Old syntax

$(\text{RateSlider}/8/100/12)^*(\text{AmountSlider} * 10000) / (1 - (1 + (\text{RateSlider}/8/100/12))^{-(\text{YearsSlider} * 12)})$

Improved syntax

```
With( { r: RateSlider/8/100/12, // interest rate
        P: AmountSlider*10000, // loan amount
        n: YearsSlider*12 }, // number of payments
        r*P / (1 - (1+r)^-n) ) // standard interest calculation
```

User Defined Functions/Types

- 1 Write short units of code,
- 3 Write code once &
- 5 Separate concerns in modules



Blocks of logic

Reusable functions that can be called from anywhere in the canvas apps

Separation of concerns

Clean UI logic

The screenshot shows the 'Updates' section of the Microsoft Power Apps Settings. It includes a search bar, a list of categories (New, Preview, Experimental, Retired), and a toggle switch for 'smart email address validation' which is currently off. Below this is a section for 'User-defined functions' with a toggle switch that is currently on. At the bottom, there is a section for 'Faster offline synchronization' with a toggle switch that is currently off.

The screenshot shows a blog post titled 'User defined functions, user defined types, and enhanced component properties move forward' by Greg Lindhorst, published on January 16, 2025. The post discusses updates to Power Fx formula reuse and maintenance, including the ability to include behavior functions with side effects in UDFs. It also introduces User-defined types (UDTs) for passing tables and records between UDFs and UDTs, and enhanced component properties (ECPs) for sharing logic across apps through a component library. The post includes social sharing icons (Facebook, Twitter, LinkedIn) and a 'Close' button at the bottom right.

User Defined Functions/Types



- 1 Write short units of code,
- 3 Write code once &
- 5 Separate concerns in modules

```
Multiply(x: Number, y: Number): Number = x * y;
```

Formulas

Define name and logic, and return values

Canvas components

3 Write code once

Reusable components

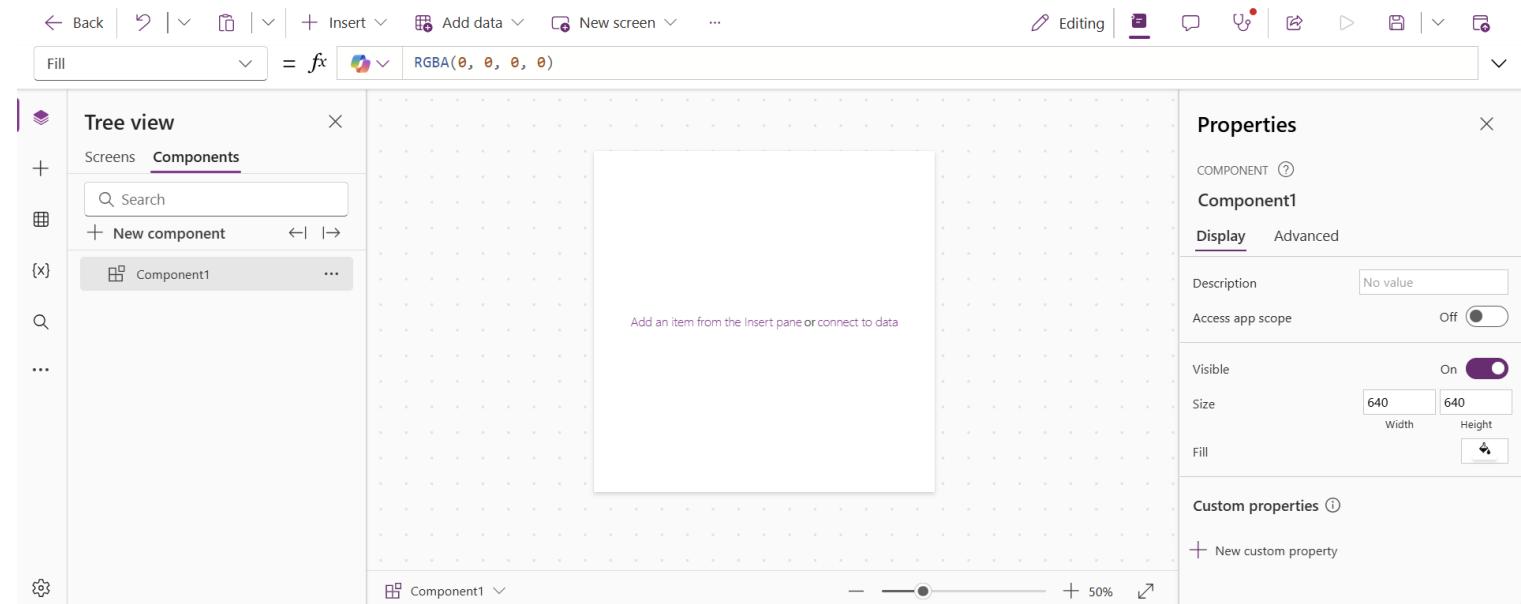
Building blocks for similar patterns

Implement advanced features

Custom properties – standardizing UI

Maintainability

Change one component – update everywhere its being used



Code components

3 Write code once

Fusing code and low-code

HTML, CSS and TypeScript

Power Apps Component Framework

Extend standard functionality

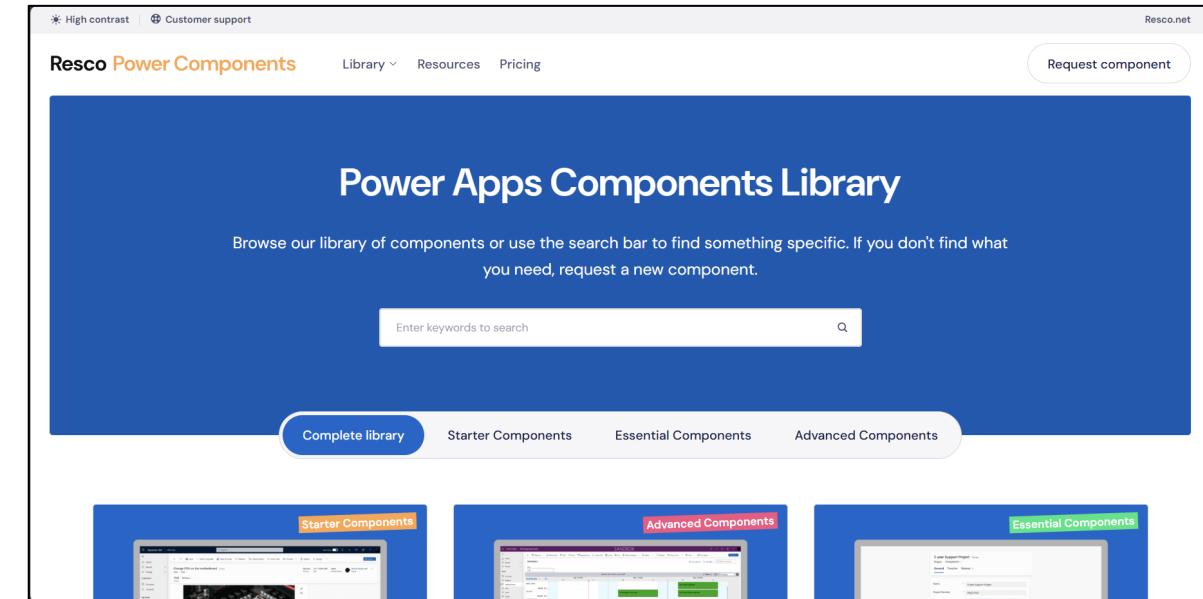
Rendered UI component that is fully customizable

Examples:

- Resco Power Components (ISV)
- Creator Kit (Microsoft Power CAT)
- PCF Gallery (Community)

Power Platform CLI

Increased complexity and can't be generated in a canvas app



Custom Connectors

3 Write code once, 5 Separate concerns in modules



Reduce redundancy

Expose only what the app needs from APIs

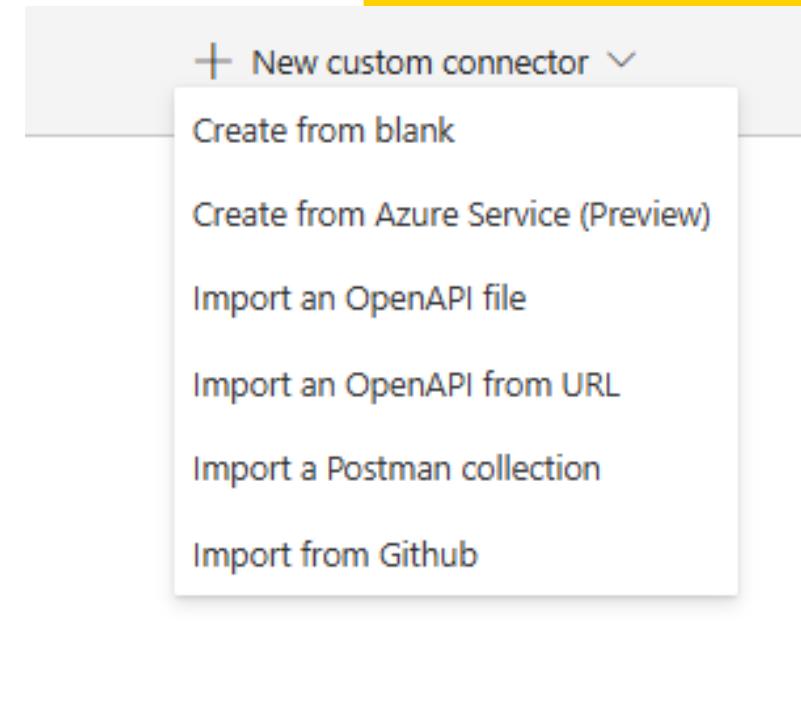
Maintainability

If changes to APIs, only the custom connector needs updating.

Replaces messy HTTP requests by named actions

Security

Keep hard coded values and credentials out of apps and flows



Canvas YAML

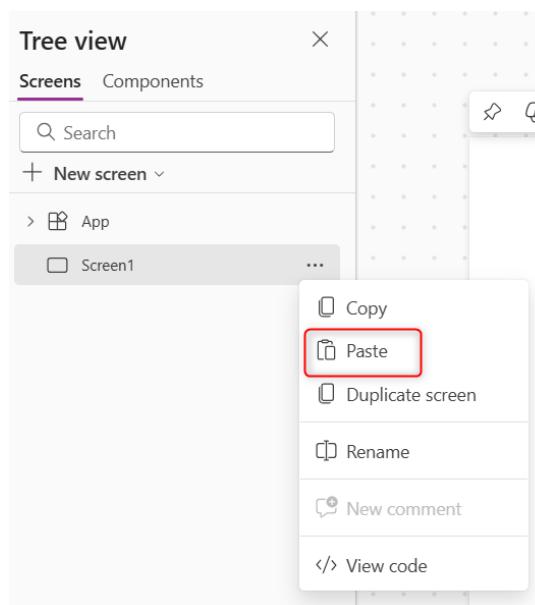
3 Write code once



DEMO

Snippets of code

Copy and paste code intelligence in canvas apps



Code Issues Pull requests Actions Projects Security Insights

Files

main

Go to file

.github assets copilot-studio power-apps

power-apps

animated-progress-circle-chart animated-svg-loading calendar glass-effect-template gradient-background grouped-vertical-feed

Laskewitz Added sample.json animated-svg-loading

Name	Last commit
..	
animated-progress-circle-chart	Updated sample
animated-svg-loading	Added sample
calendar	Updated sample
glass-effect-template	Update README
gradient-background	Added sample
grouped-vertical-feed	Bug fixes

Responsive Layout Management

2 Write simple units of code

Containers

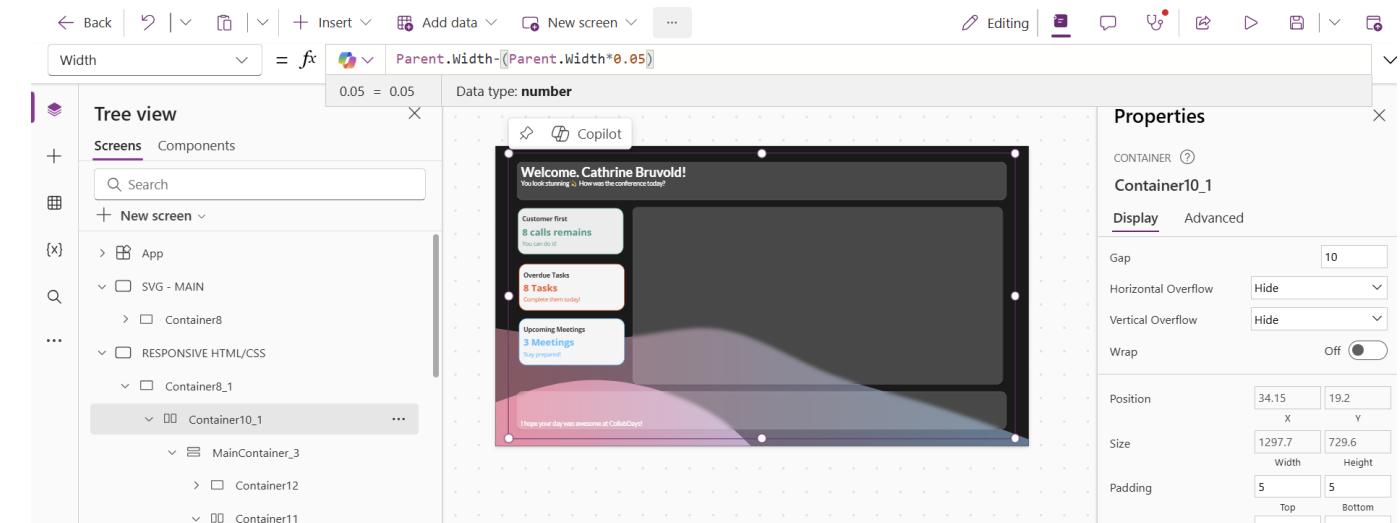
Horizontal and vertical – dynamic resizing

Align and center screen elements

`Parent.Height & Parent.Width`

Build efficiently

Reuse YAML



DEMO

Test Engine

9 Automate development pipeline and tests



Power Apps Test Engine – code first

Automation testing for UI for canvas apps

Integration

CI/DC - Azure DevOps and GitHub Action

Write tests – Power Platform CLI

TypeScript

Benefits

Faster feedback loop – no manual testing needed

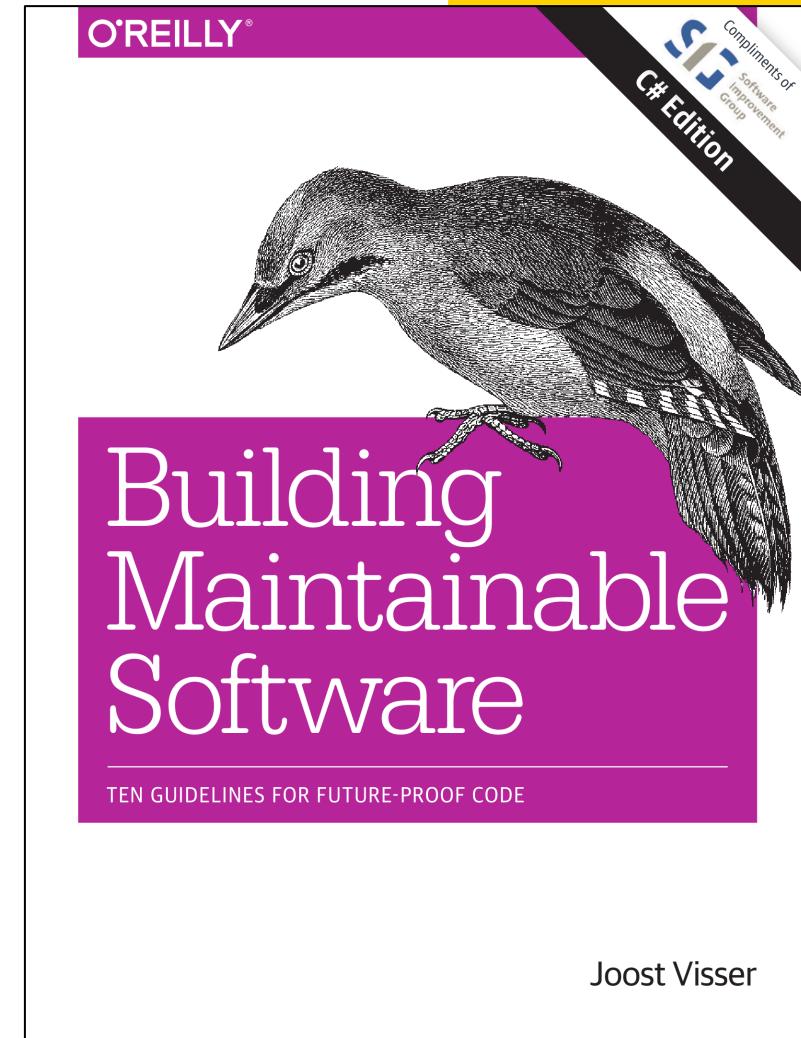
```
import test = '@microsoft/powerapps-test';

test('Submit button triggers success' as {
  async asyn.open()
  app.selectScreen("MainScreen");
  app.enterText("TextInput_Name", Cathrine);
  app.click("Button_Submit");
  app.expectText("Label_Result", Submission
    successful');
})
```

Building Maintainable Software

Guidelines for future-proof code

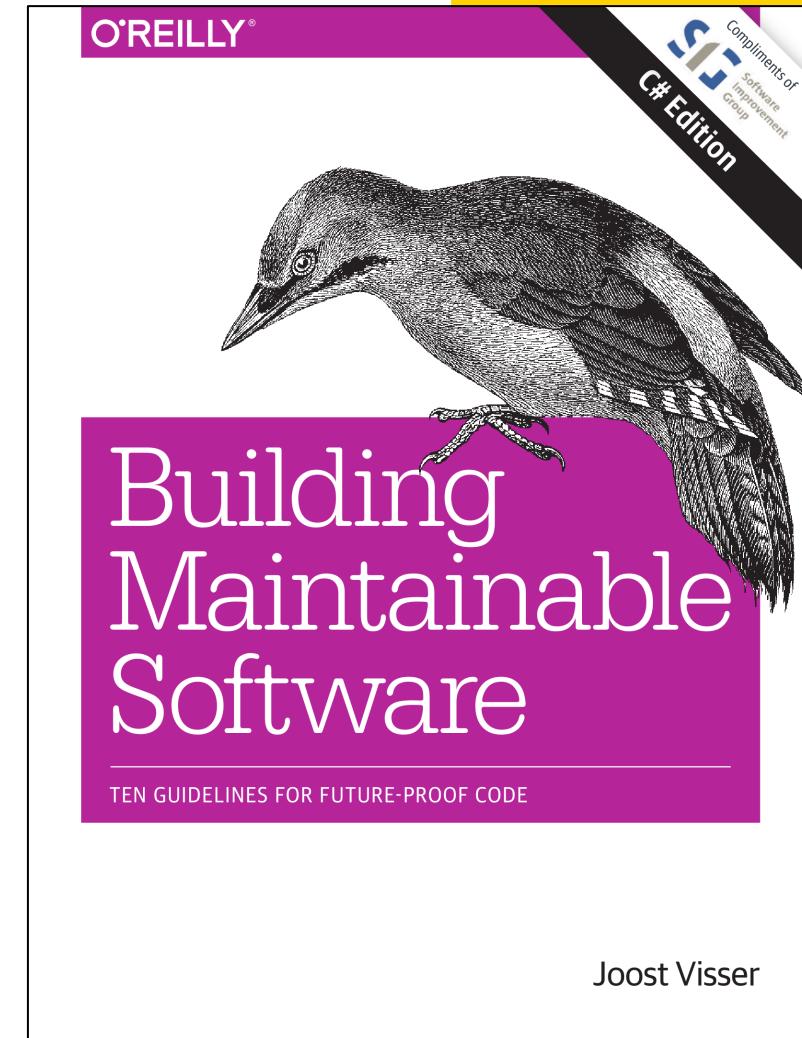
1. Write short units of code
2. Write simple units of code
3. Write code once
4. Keep unit interfaces small
5. Separate concerns in modules
6. Couple architecture components loosely
7. Keep architecture components balanced
8. Keep your codebase small
9. Automate development pipeline and tests
10. Write clean code



Building Maintainable Software

Guidelines for future-proof code

1. **Write short units of code**
2. **Write simple units of code**
3. **Write code once**
4. **Keep unit interfaces small**
5. **Separate concerns in modules**
6. Couple architecture components loosely
7. Keep architecture components balanced
8. Keep your codebase small
9. **Automate development pipeline and tests**
10. Write clean code



Resources

