Microsoft
Power Platform

Agents for change//

A.I. POWER PLATFORM POWER LONDON

MAY 24TH 2025

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

dox42®

automate your documents

integrate your data

DOCENTRIC

LogiSam
IT Consultancy

π9
pi9.co • info@pi9.co

Dynamics 365
Implementation Partner
for ERP, CRM & HR

Microsoft Partner

Hubdrive

Sponsors

THANK YOU!

POWER COMMUNITY


Microsoft
A.I. POWER PLATFORM
POWER
LONDON
24th May
Senior Power Platform Lead@Microsoft
Cathrine Bruvold
Optimizing Canvas Apps for maintainability, reusability, and performance

Microsoft

A.I. POWER PLATFORM POWER LONDON 2025

24th May

Senior Cloud Architect @Microsoft

Daniel Laskewitz

Optimizing Canvas Apps for maintainability, reusability, and performance

Agenda

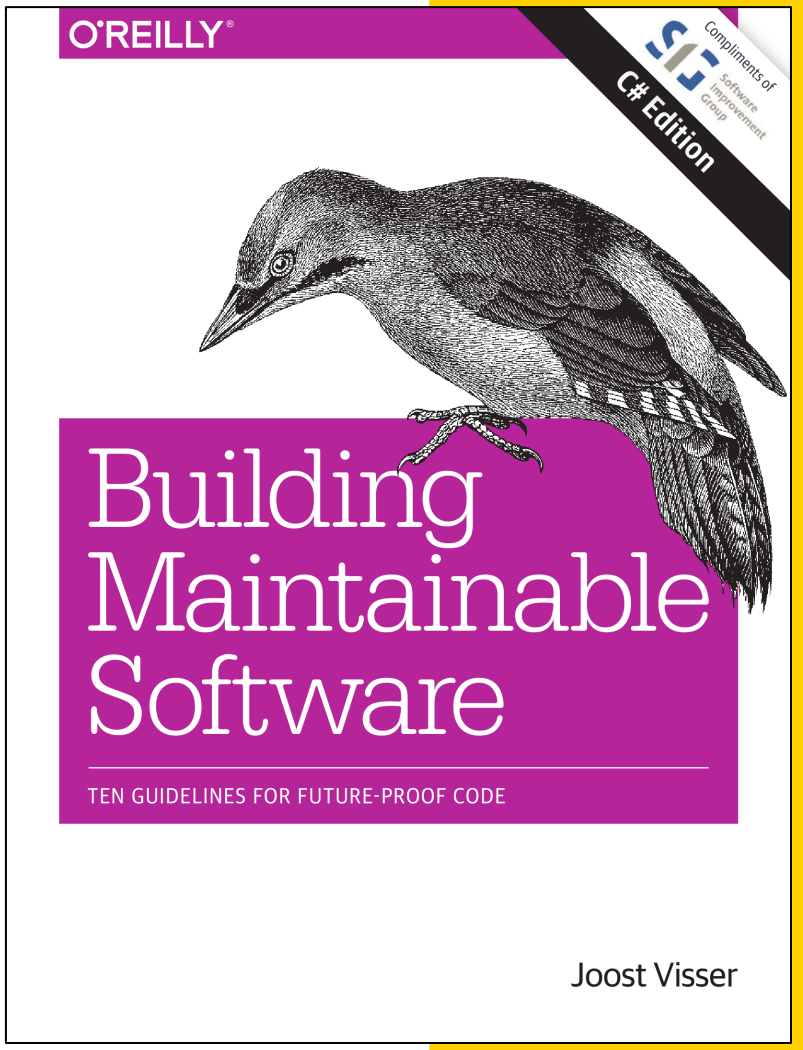
- ✨ Introduction
- ✨ Building Maintainable Software book
  - ✅ 10 guidelines
- ✨ Canvas Apps Best Practices
- ✨ Resources

POWER COMMUNITY

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

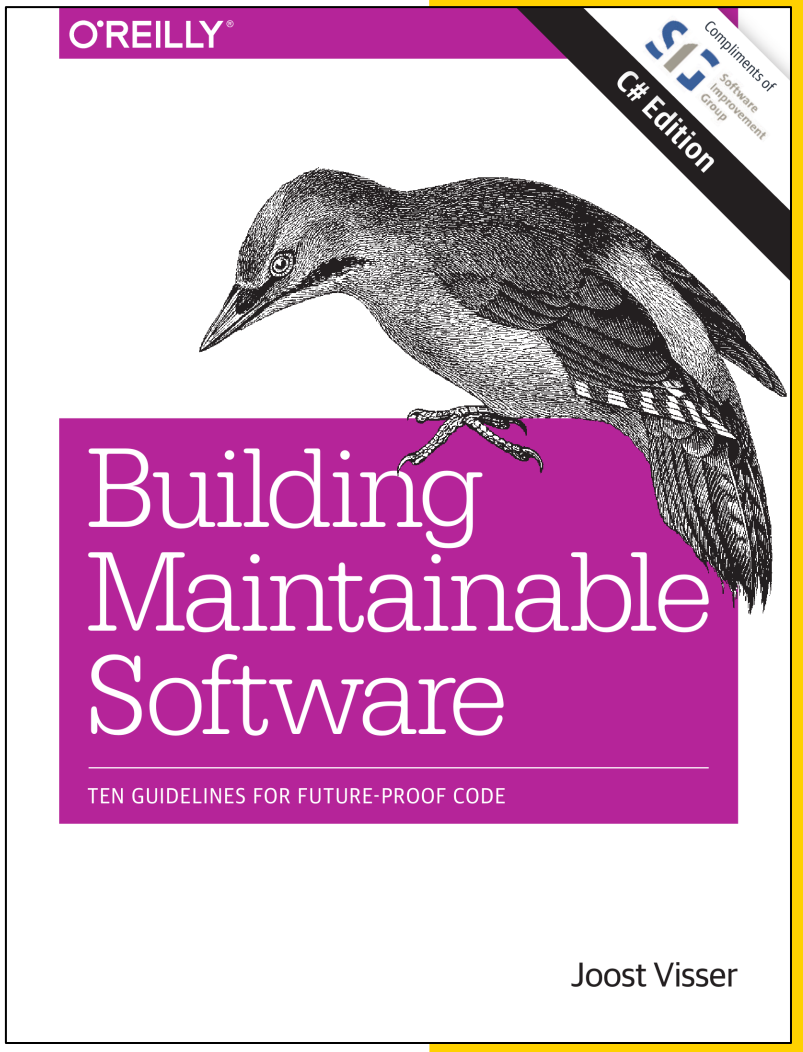


POWER COMMUNITY

1 Write short units of code

Guideline:

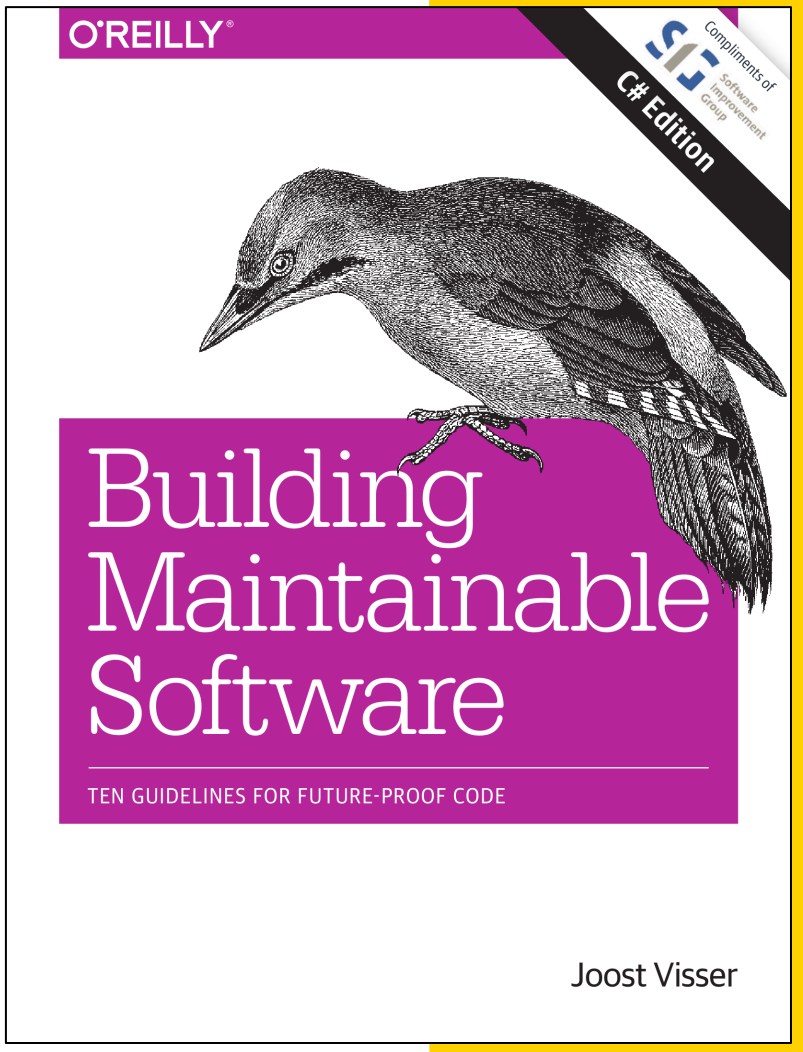
- **Limit the length of code units to 15 lines of code.**
- Do this by **not writing units that are longer than 15 lines of code** in the first place, or by **splitting long units into multiple smaller units** until each unit has at most 15 lines of code.
- This improves maintainability because **small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



2 Write simple units of code

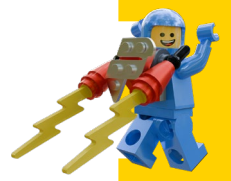
Guideline:

- **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.**



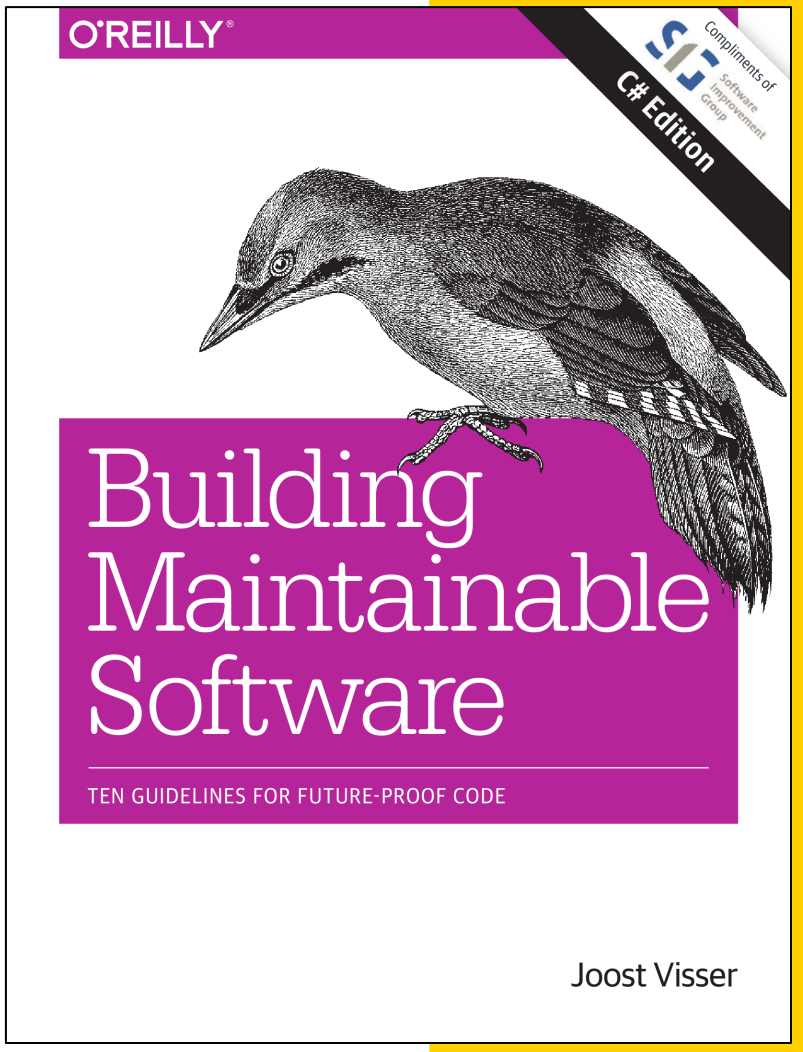
POWER COMMUNITY

3 Write code once



Guideline:

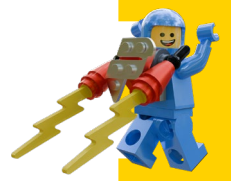
- **Do not copy code.**
- Do this by **writing reusable, generic code and/or calling existing methods instead.**
- This improves maintainability because **when code is copied, bugs need to be fixed at multiple places**, which is inefficient and error-prone.



POWER COMMUNITY

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

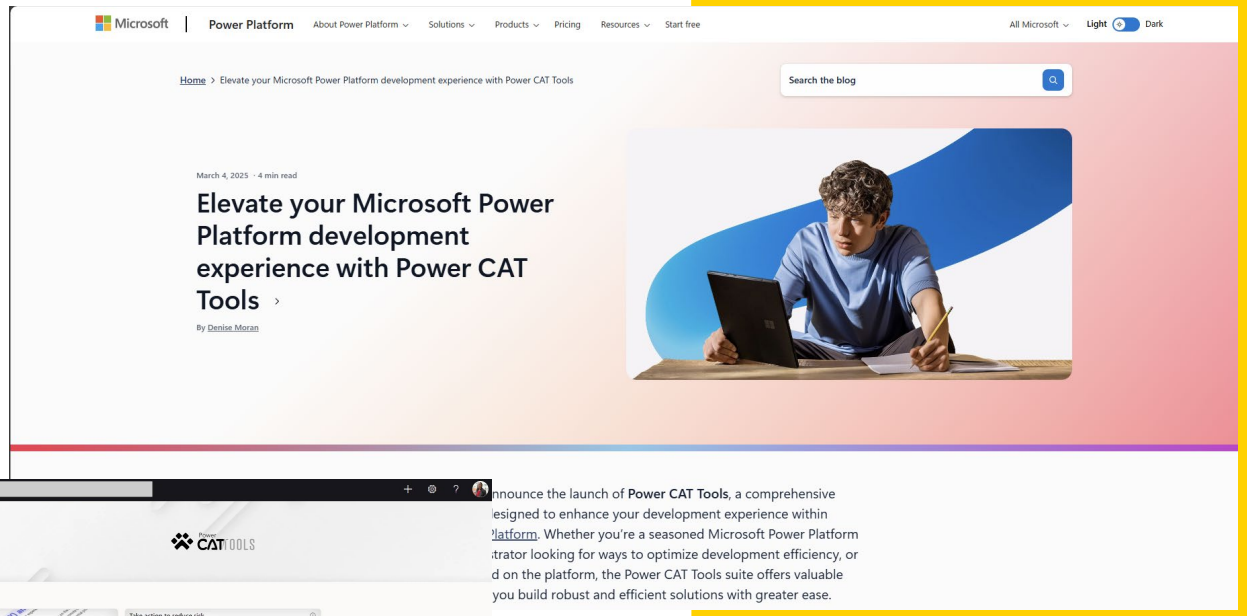
DEMO

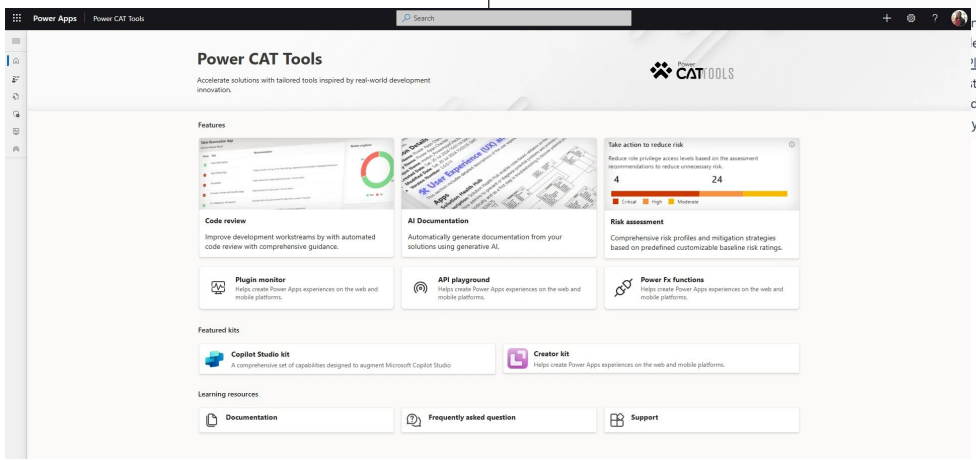
Code review

AI solution documentation

Risk assessment

...and more





Reduce the amount of code

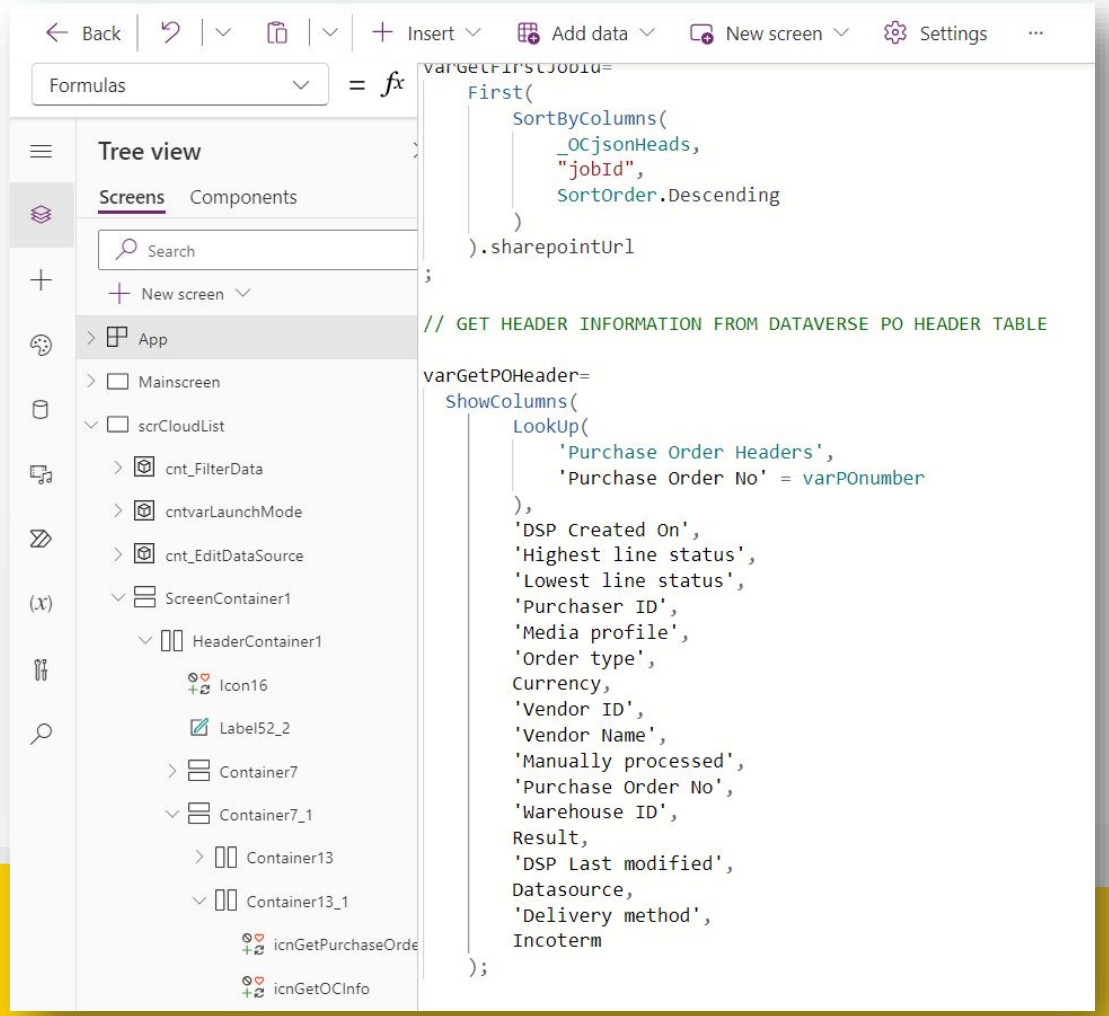
1 Write short units of code, 3 Write code once

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()

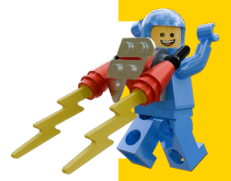
Old syntax

```
(RateSlider/8/100/12)*(AmountSlider*10000) / (1-(1+(RateSlider/8/100/12))^(-(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
```

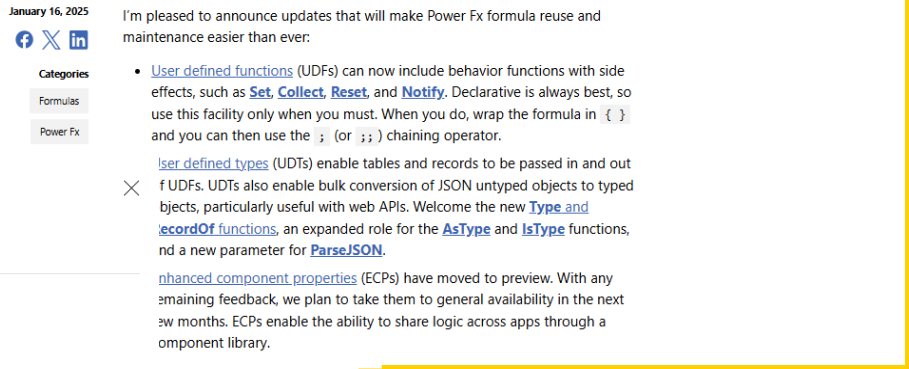
User Defined Functions/Types



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

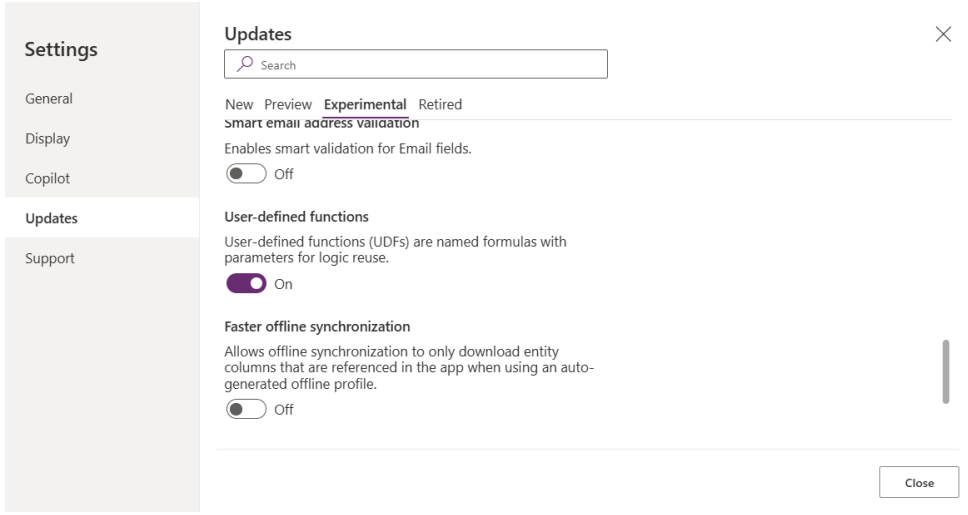
Blocks of logic

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



Separation of concerns

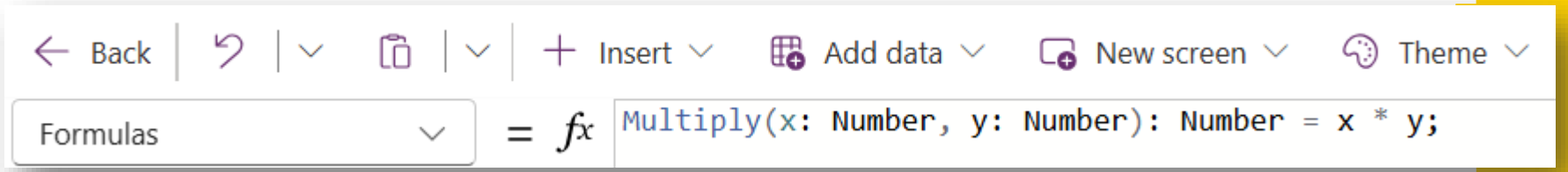
Clean UI logic



POWER COMMUNITY

User Defined Functions/Types

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



Formulas

Define name and logic, and return values

POWER COMMUNITY

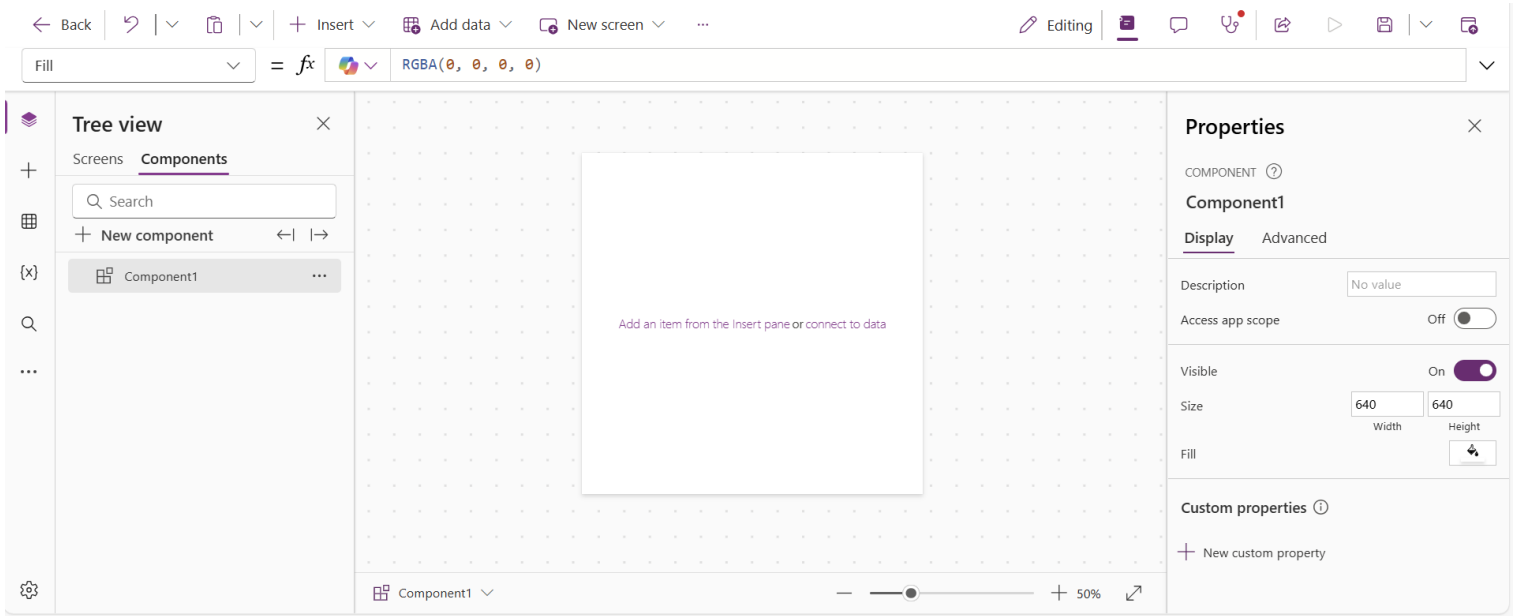
Canvas components

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

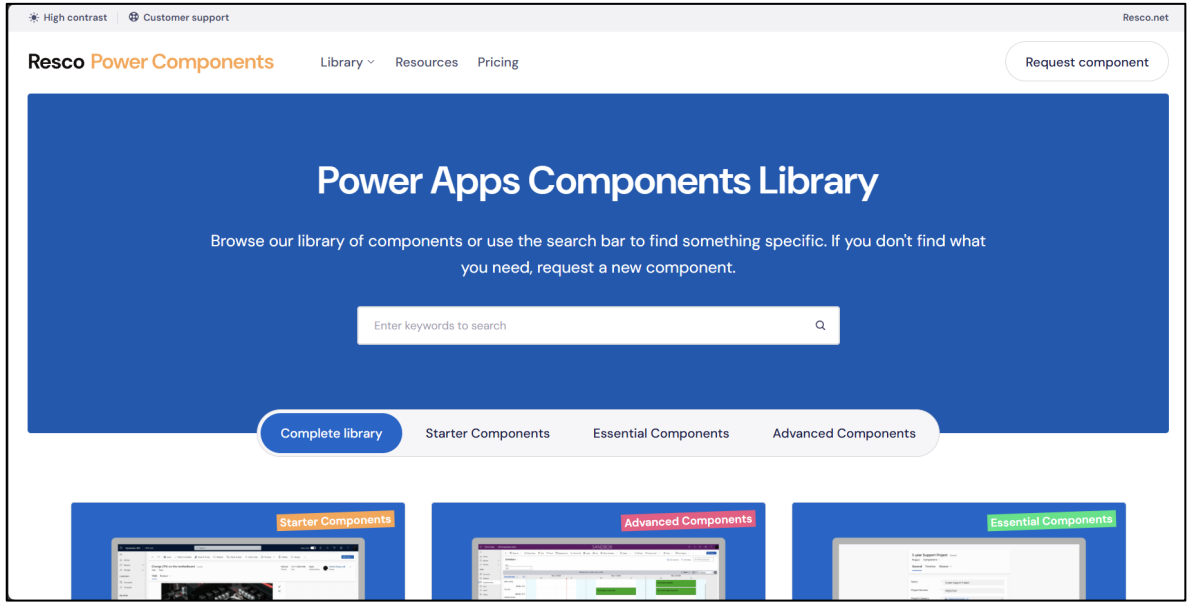
Rendered UI component that is fully customizable

Examples:

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

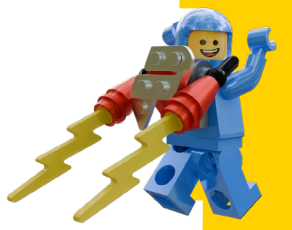
Power Platform CLI

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



Custom Connectors

3 Write code once, 5 Separate concerns in modules



Reduce redundancy

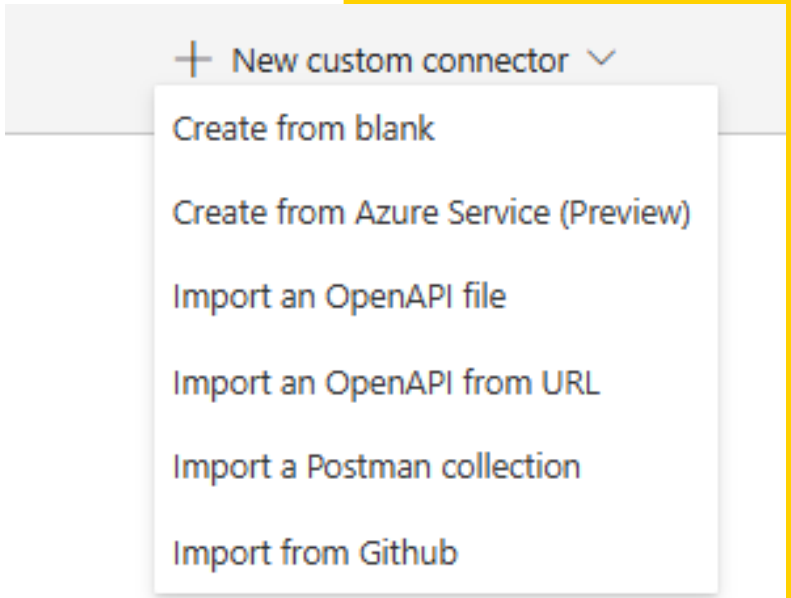
Expose only what the app needs from APIs

Maintainability

If changes to APIs, only the custom connector needs updating.
Replaces messy HTTP requests by named actions

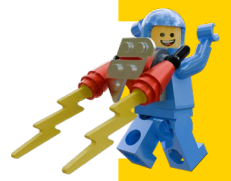
Security

Keep hard coded values and credentials out of apps and flows



Canvas YAML

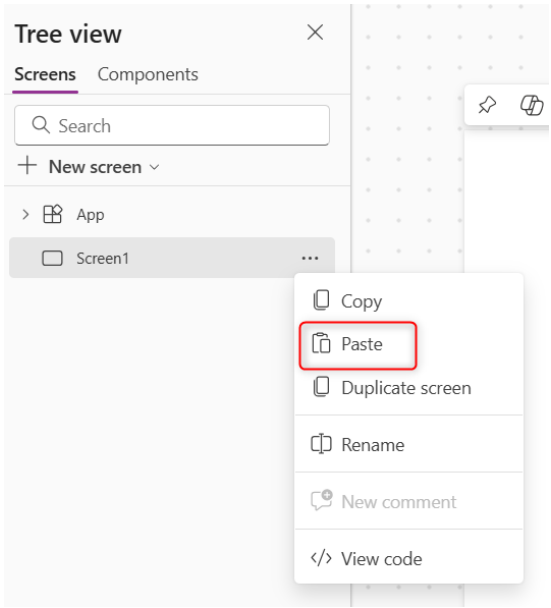
3 Write code once

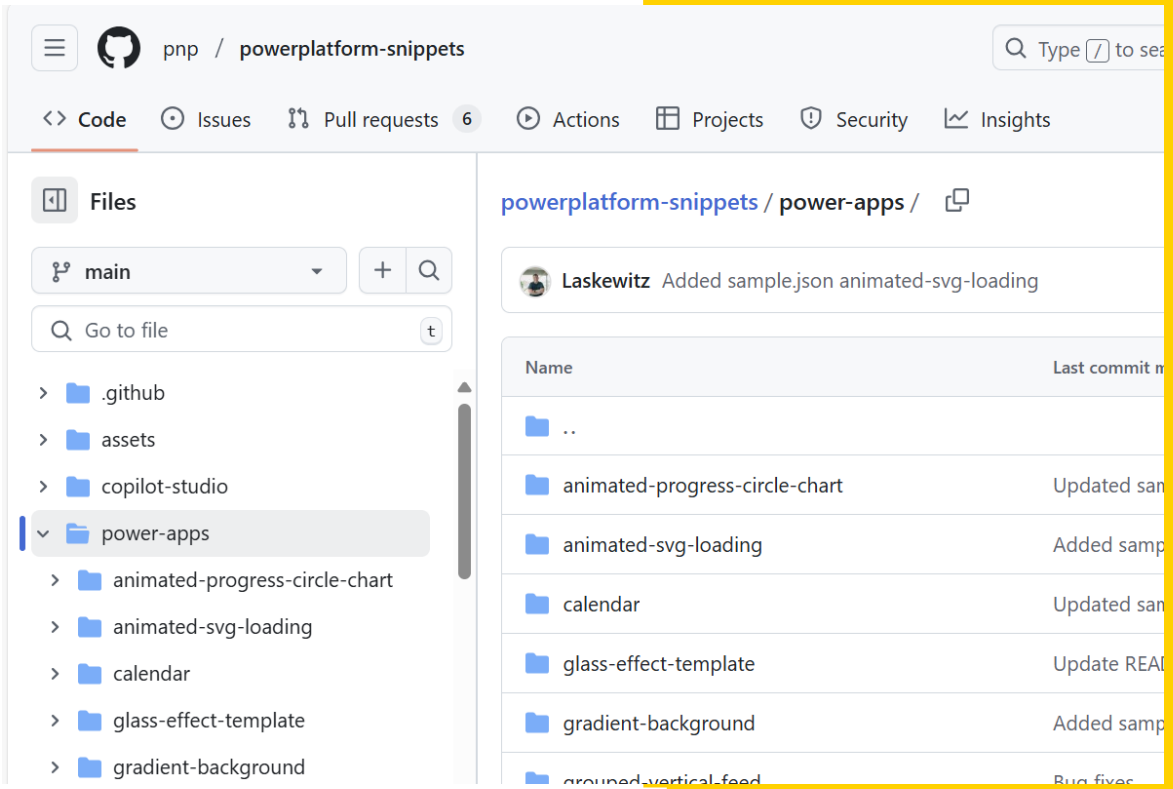


DEMO

Snippets of code

Copy and paste code intelligence in canvas apps





Responsive Layout Management

2 Write simple units of code

Containers

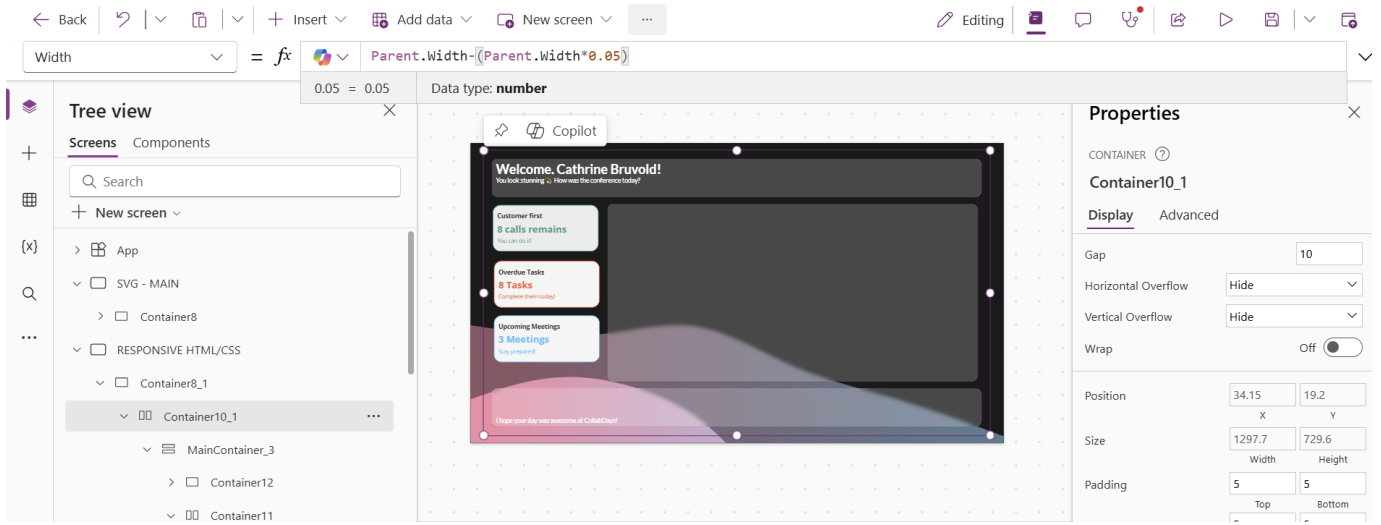
Horizontal and vertical – dynamic resizing

Align and center screen elements

Parent.Height & Parent.Width

Build efficiently

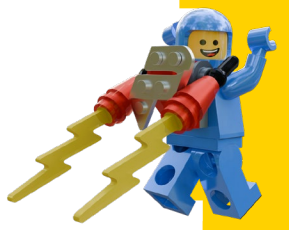
Reuse YAML



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

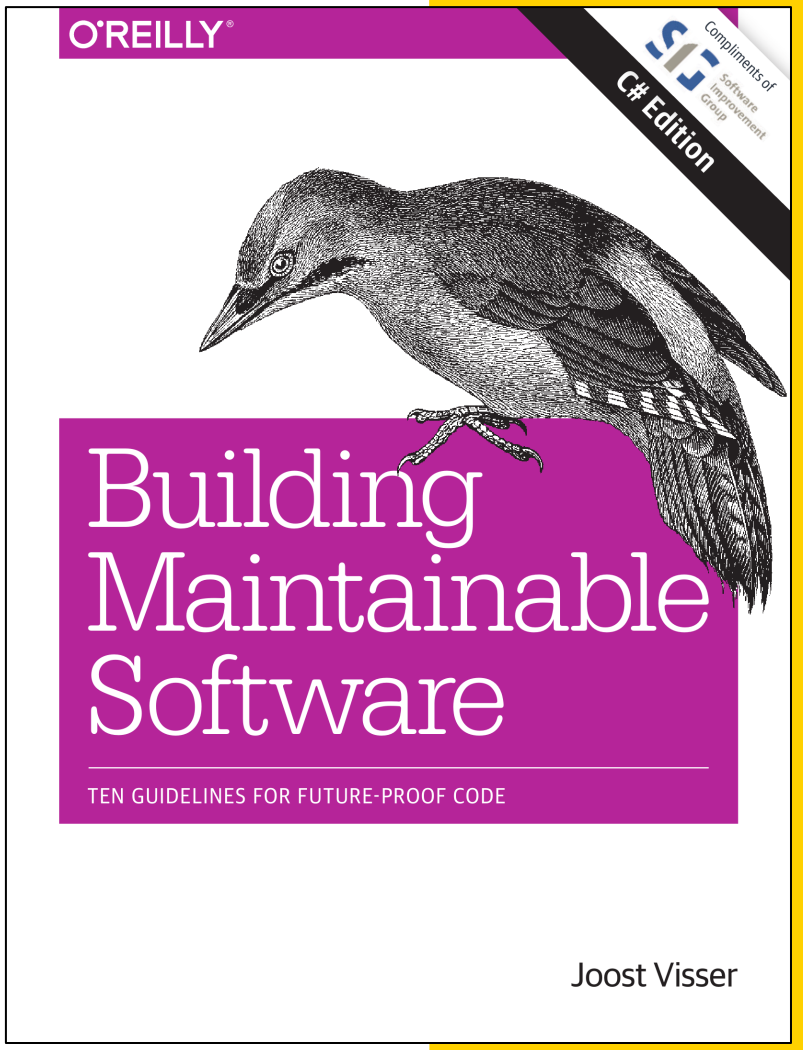
```
import test = '@microsoft/powerapps-test';

test('S"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');
}
```

POWER COMMUNITY

Building Maintainable Software

Guidelines for future-proof code

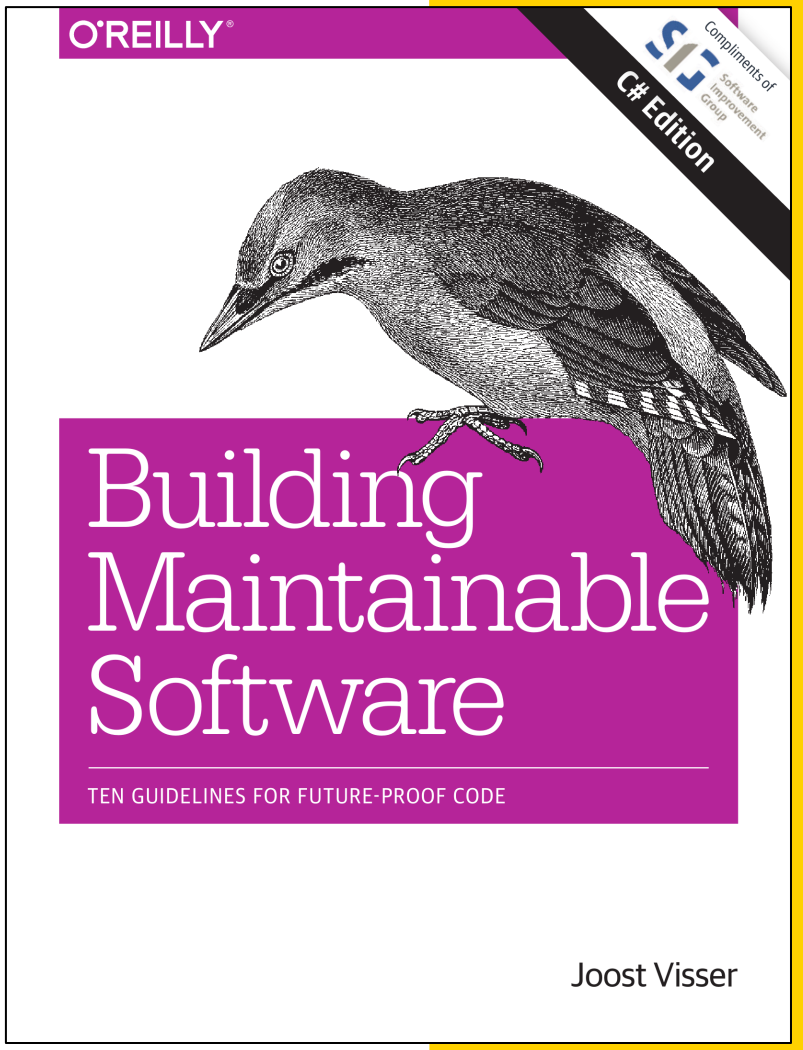
1. Write short units of code
2. Write simple units of code
3. Write code once
4. Keep unit interfaces small
5. Separate concerns in modules
6. Couple architecture components loosely
7. Keep architecture components balanced
8. Keep your codebase small
9. Automate development pipeline and tests
10. Write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



Building Maintainable Software

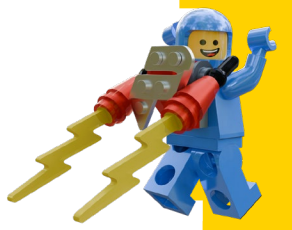
Guidelines for future-proof code

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



POWER COMMUNITY

Resources



POWER COMMUNITY