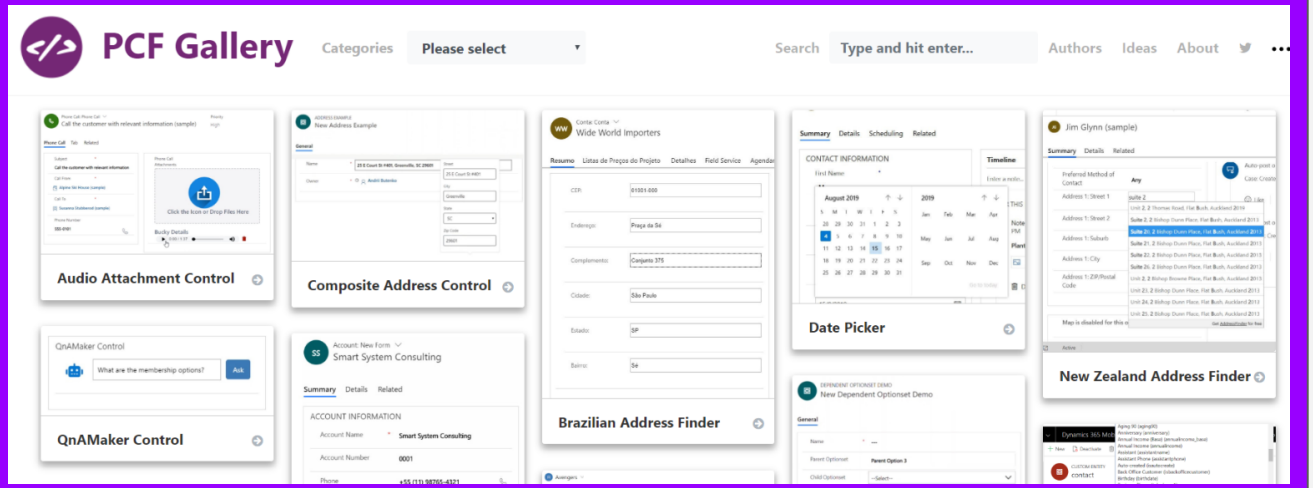
PowerApps component framework

- Developers were getting bored
- Makers were getting bored
- Everyone was asking for Microsoft to come up with something cool
- And...

PowerApps component framework has come!



About myself

Name: Alex Shlega

Title: Dynamics 365 Consultant/Developer/Solution Architect

Blog: https://www.itaintboring.com

Linkedin: https://www.linkedin.com/in/alexandershlega/

Twitter: https://twitter.com/ashlega

Blog > Admin Features > Announcements > Dynamics 365 > General > Model Driven Apps > New Features > PowerApps

Announcing the general availability of the PowerApps component framework for model-driven applications and PowerApps CLI

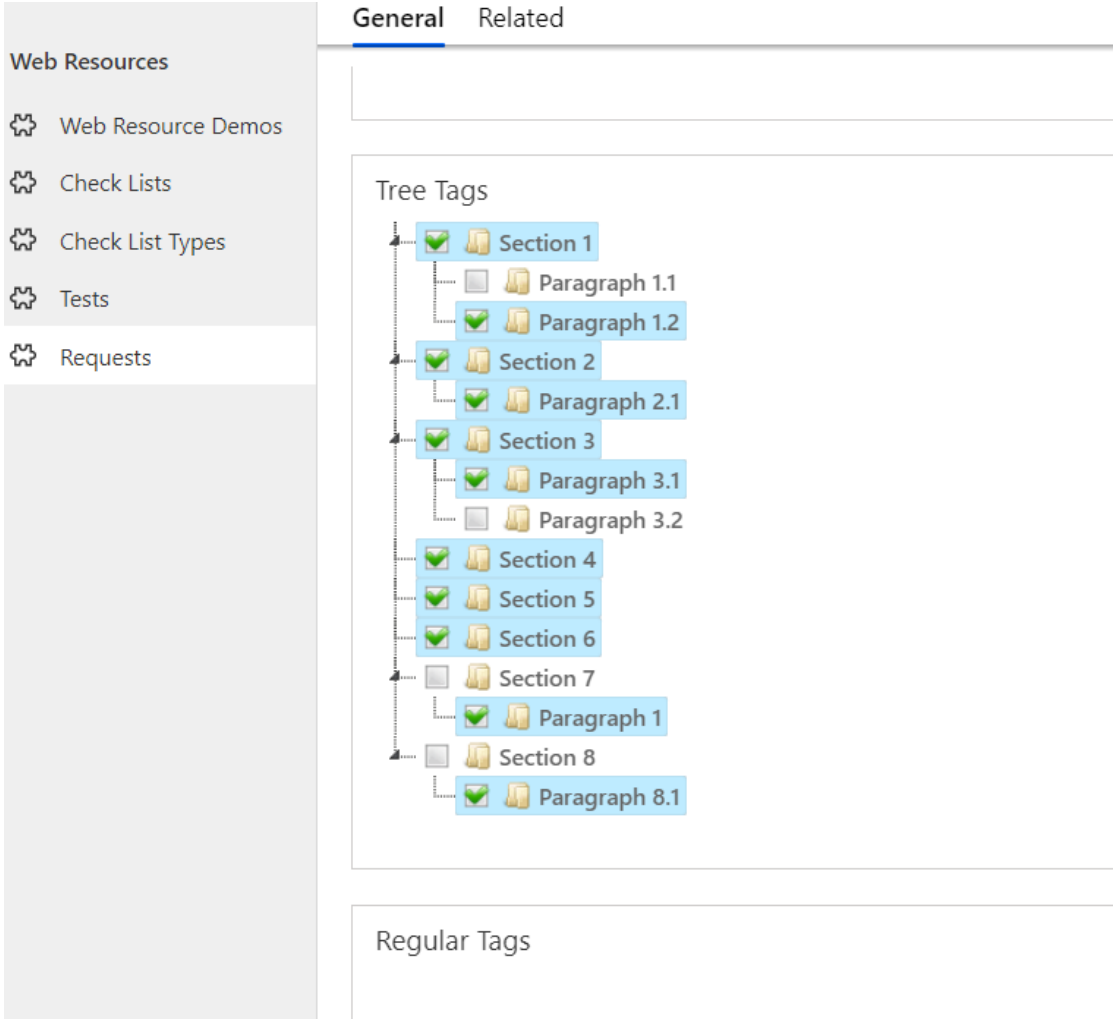
Hemant Gaur, , Tuesday, October 1, 2019

Today we are making the **PowerApps component framework** for model-driven apps and the **PowerApps CLI** generally available. Just a short time ago we made the PowerApps component framework available for public preview and in less than six months our pro-developer community has grown and created over 800 third party code components with over 14k monthly active users. These truly innovative components now enable new scenarios that

https://powerapps.microsoft.com/en-us/blog/announcing-the-general-availability-of-the-powerapps-component-framework-for-model-driven-applications-and-powerapps-cli/

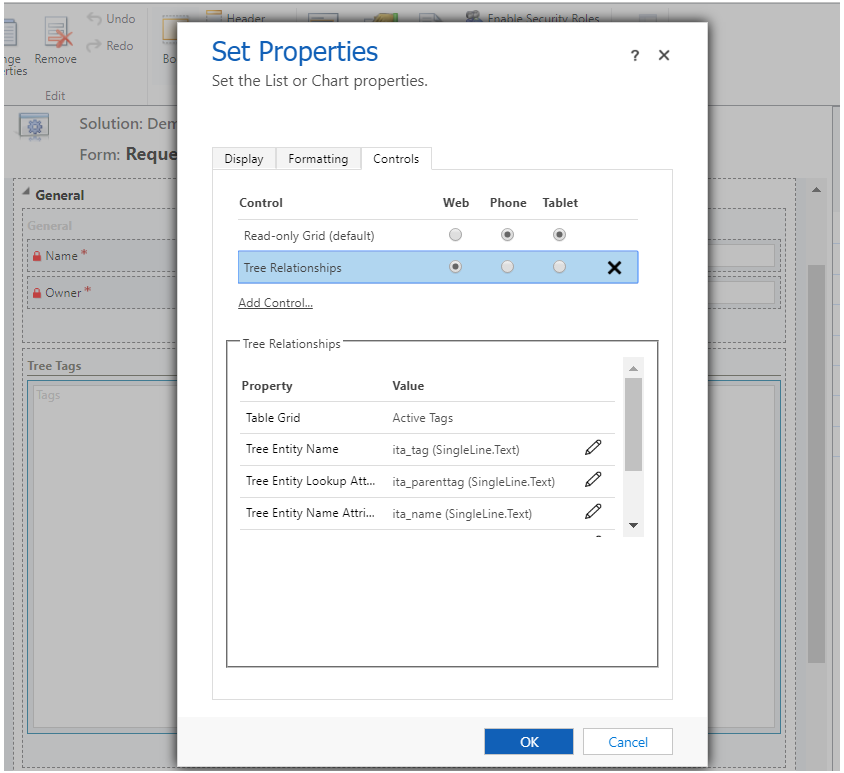
Example:
N:N in a TreeView

- Regular N:N
- Mapped to a TreeView
- All related records are still traceable through the N:N (Advanced Find, reporting, etc)



Why PCF?

- Reusable & configurable (properties/values/configuration)
- Solution-Aware
- No iframes anymore, no additional form scripts to control the iframe, much less related “plumbing”
- Can be developed by developers and easily re-used by makers



What’s involved

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine

npm makes it easy for JavaScript developers to share and reuse code, and makes it easy to update the code that you’re sharing, so you can build amazing things.

It is a strict syntactical superset of JavaScript that adds optional static typing to the language

- JavaScript
- JQuery
- Solutions
- React
- Angular
- PowerShell

Limitations

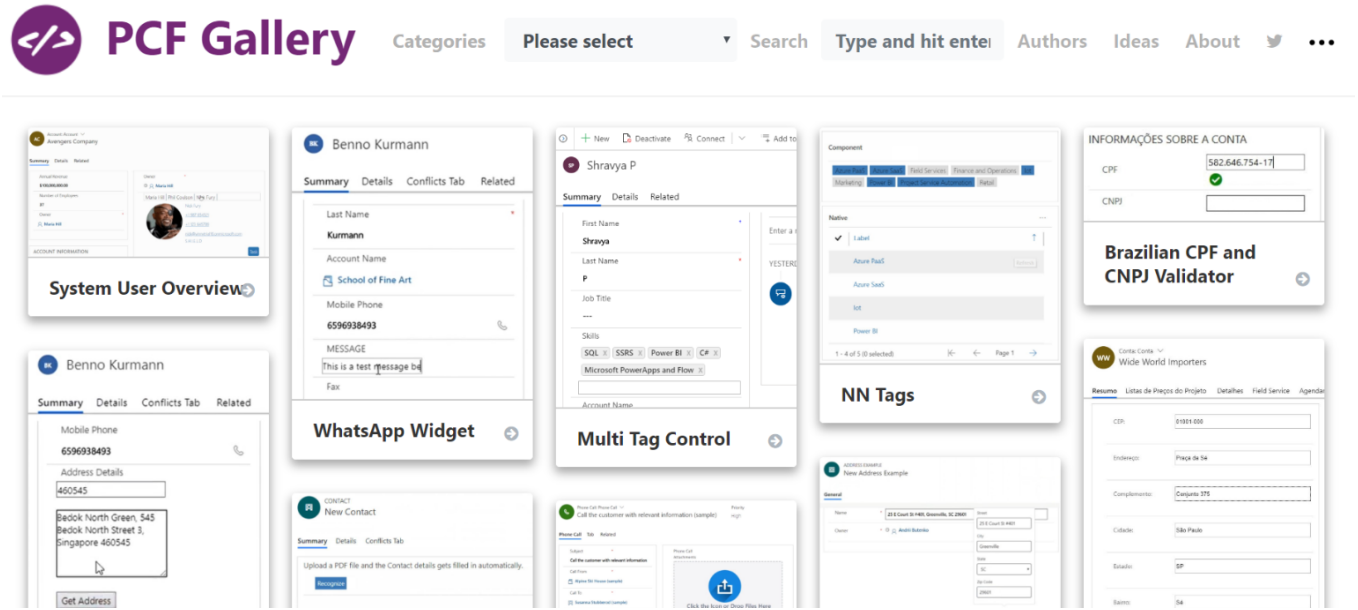
- Components should bundle all code including external library content into the primary code bundle
- Sharing libraries across components using library nodes in component manifest is not supported for public preview
- Preview feature for now - proceed with caution if planning to use in production

*Throughout the preview, there were a couple of releases which were not quite backward compatible

From WebResources to PCF

- TypeScript instead of JavaScript
- CLI tools instead of in-app web resource editor
- HTML elements to be added to the container programmatically instead of using plain HTML
- Out of the box event plumbing (init, updateView, getOutputs, destroy)
- Configuration through parameters
- Eventually, the same component framework for both types of PowerApps
- New opportunities for the ISV-s

PCF Gallery (78 components as of September 17, 2019)



Mostly free components with the usual pros and cons:

you can find awesome open-source components there, but the level of support behind them will vary depending on the willingness and availability of their creators.

https://pcf.gallery/

Developed and maintained by Guido Preite, MVP

Who can build PCF components?

- As long as you know the basics of html & javascript, you should be able to create PCF components

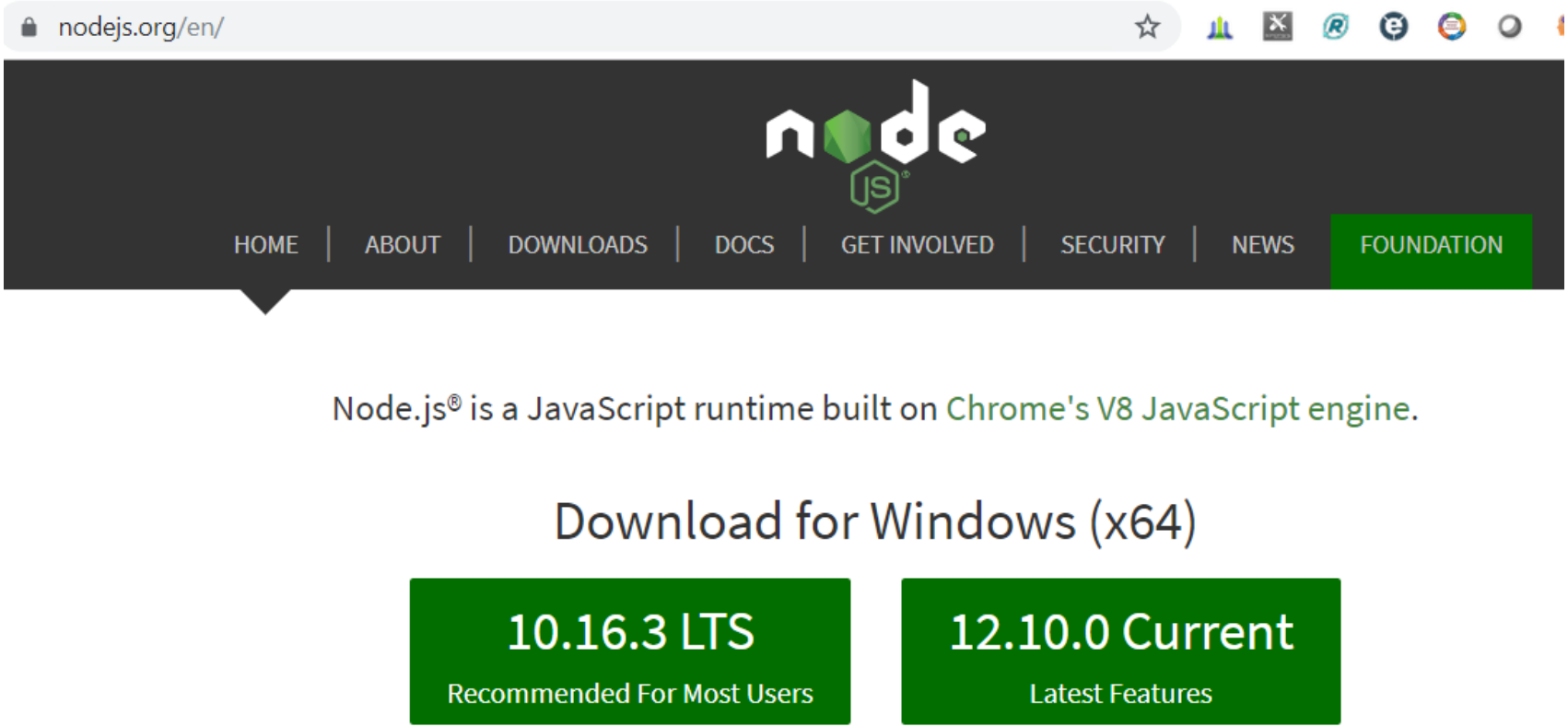
  because

- Essentially, it’s front-end development

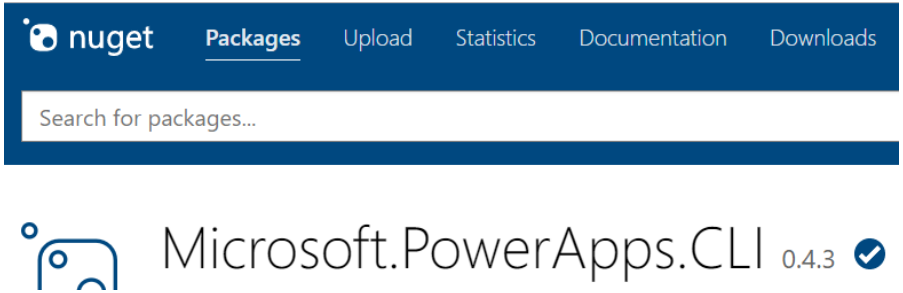
  however

- Just as it used to be with the the web-resources, PCF development can vary in complexity from being relatively simple to being extremely advanced

Step 1: Get Node.js and NPM



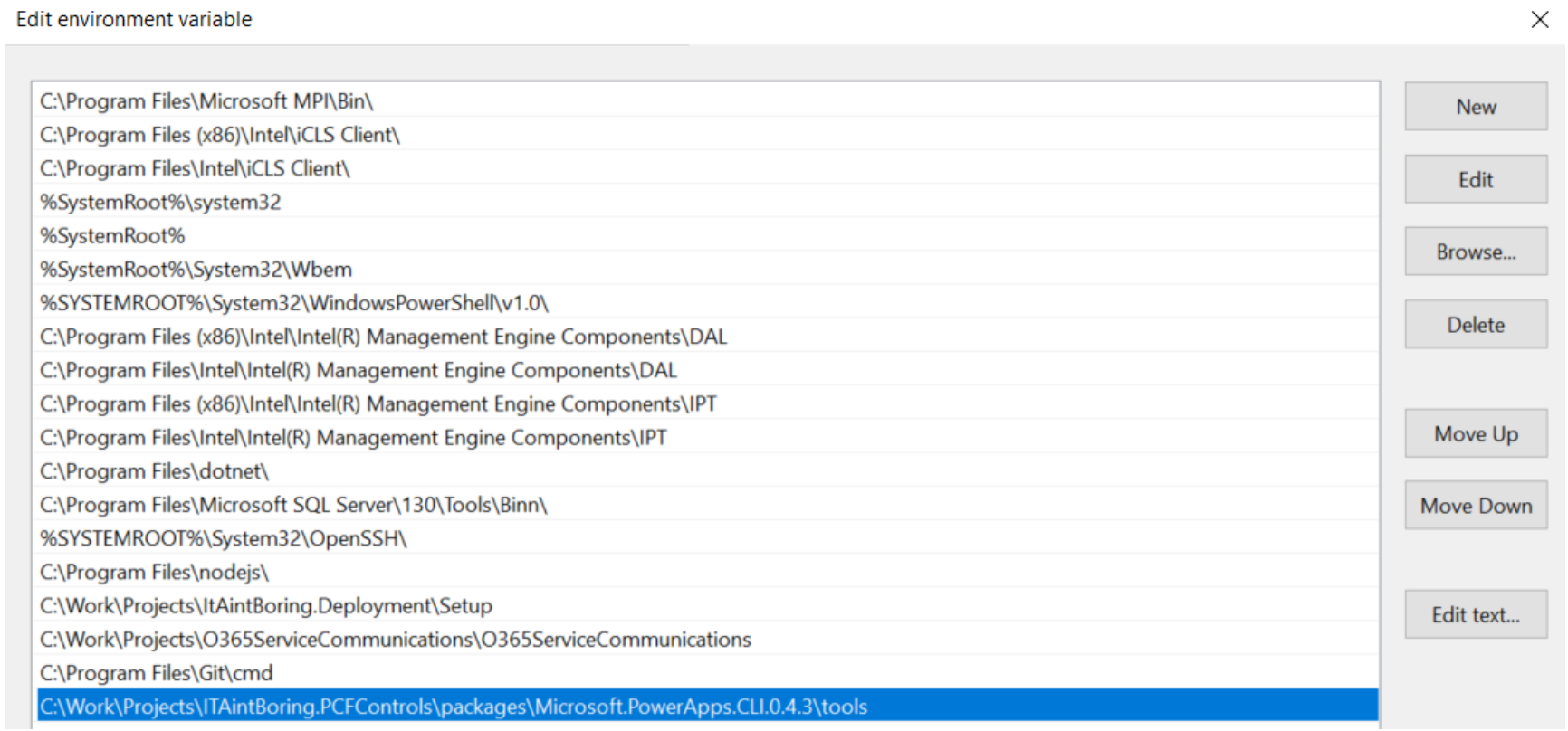
Step 2: Get the tools




```
$sourceNugetExe = "https://dist.nuget.org/win-x86-commandline/latest/nuget.exe"
$targetNugetExe = ".\nuget.exe"
Remove-Item .\Tools -Force -Recurse -ErrorAction Ignore
Invoke-WebRequest $sourceNugetExe -OutFile $targetNugetExe
Set-Alias nuget $targetNugetExe -Scope Global -Verbose

./nuget install  Microsoft.PowerApps.CLI -O .\packages
```

Step 3: Add tools folder to the path



Step 4: Create a component

- Create a folder for your component
- Initialize the component

pac.exe pcf init --namespace ItAintBoring.PCFControls --name DemoRegexValidation --template field
npm install

Step 5: Build and test the component

npm run build
npm start (or you can use **start npm start** to open a new window**)**

Creating a New Component
Adding Regex Validation Code
&
Testing

Step 6: Create a solution for the component

- Create a folder for the solution project
- Create solution files

cd <solution folder>
pac solution init --publisher-name <enter your publisher name> --publisher-prefix <enter your publisher prefix>

Step 7: Add component to the solution

pac solution add-reference --path <path of your PowerApps component framework project on disk>

Step 7: Build the soluion

msbuild /t:build /restore (msbuild must be in the path)

**If you get an error stating that the control directory already exists, delete “obj” and “bin” folders first*

*If you want to build a managed package, add add /p:configuration=Release

msbuild /t:build /restore **/p:configuration=Release**

- Importing the solution
- Model-Driven App Example
- Canvas App Example

A few things to consider

If you are a developer:

- It’s a great new framework for developers
- There are no iframes
- Typescript is enforced
- It’s easy to build redistributable components

If you are an organization:

- You can start building reusable component to utilize them across different solutions. You will need somebody with a little more advanced dev skills to do that
- If you are an ISV, there are apparent opportunities there
- It’s easy with PCF to utilize third-party components (but think about their supportability, too)

If you are a maker:

- You can easily reuse PCF components developed by professional developers
- PCF components are configurable
- May need to pay attention to the future compatibility/maintenance

Good to know

- Upgrading from pre 1.0.* versions

  If a component is created using the CLI tooling version lower than 1.0.*, you may need to rebuild with the updated version of the tools

- Pay attention to the feature-usage element in the manifest file

```
<feature-usage>
    <uses-feature name="Device.captureAudio" required="true" />
    <uses-feature name="Device.captureImage" required="true" />
    <uses-feature name="Device.captureVideo" required="true" />
    <uses-feature name="Device.getBarcodeValue" required="true" />
    <uses-feature name="Device.getCurrentPosition" required="true" />
    <uses-feature name="Device.pickFile" required="true" />
    <uses-feature name="Utility" required="true" />
    <uses-feature name="WebAPI" required="true" />
 </feature-usage>
```

Good to know

- Updating the component

  If you notice that the changes you are making to the component code are not showing up in the environment somehow, try increasing component version in the manifest file:

```
constructor="DemoRegexValidation" version="0.0.2" display-name-
```

Good to know

- Can we use React?

> ⓘ **Important**
>
> Although the PowerApps host applications work on top of React, the version of React you bundle will not communicate with the host version, nor is it dependent on that version. A new copy of React (or any third-party library you bundle with your component) will be loaded into the host page for every instance of that control, so be mindful of how large you are making your page(s) as you add components. We will have a solution to this issue in a future release.

WebAPI vs Xrm

- There is no Xrm in the Canvas Applications
- There is no WebAPI in the Canvas Applications
- Xrm is unlikely to ever show up in the Canvas Applications
- WebAPI might still show up there

What if you still wanted to use Xrm in your PCF component while in the model-driven app space?

You could simply take advantage of the fact that PCF component is running within the same form where Xrm is available, so you could declare Xrm variable:

```
declare var Xrm: any;
```

And use it in your typescript code:

```
var url: string = (<any>Xrm).Utility.getGlobalContext().getClientUrl();
```

PCF vs Embedded Canvas Apps

- PCF:

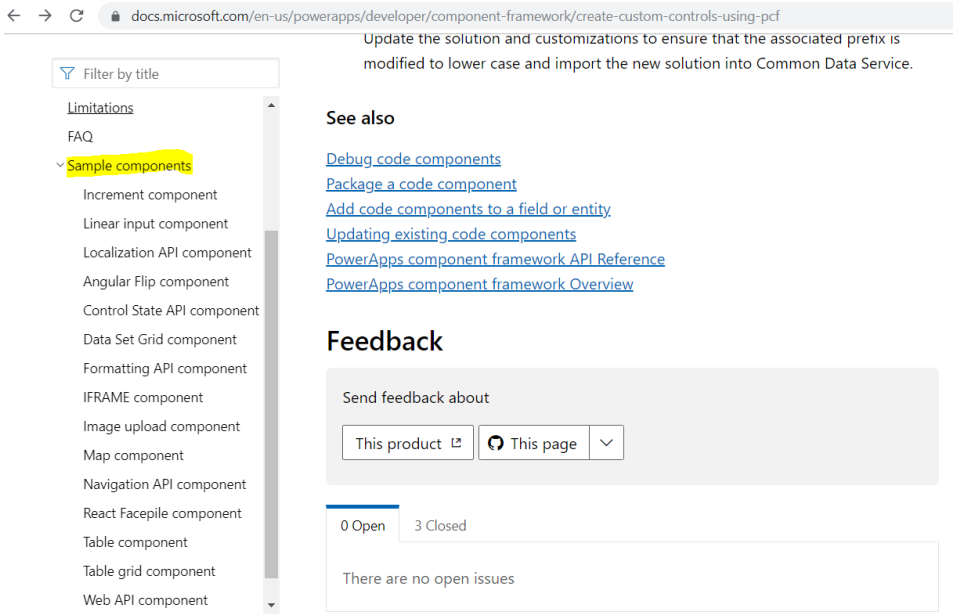
  HTML & Scripting, Web API, configurable property bindings, professional developers expertise

- Embedded Canvas Apps:

  Various connectors, canvas app look and feel, citizen development expertise

Ready to start building your own component?

- Go here: https://docs.microsoft.com/en-us/powerapps/developer/component-framework/create-custom-controls-using-pcf
- Pick a sample and start tweaking it to your needs:



Q & A