



Microsoft Power Platform

# Dev in a day

Lab 02 Build a code component/ December 2022

## Table of Contents

Lab Scenario .....	1
Exercise 1 – Build Code Component .....	1
Task 1: Create the code component .....	1
Task 2: Implement the component logic .....	6
Exercise 2 – Use the Code Component .....	14
Task 1: Allow Power Apps component framework .....	14
Task 2: Edit canvas app .....	15
Exercise 3 – Add Code Component to Solution .....	22
Task 1: Add component to solution .....	22

### Lab Scenario

Working as part of the PrioritZ fusion team you have been asked to create a Power Apps code component to allow drag and drop priority ranking of items in the PrioritZ Ask Power App. You will build a code component using the React JavaScript framework. A code component approach is used to address the requirement because there isn't a similar control already built-in.

You have collaborated with the app makers to identify the following properties to allow them to configure the code component in the app:

- BackgroundColor
- DragBackgroundColor
- ItemHeight
- FontSize
- FontColor

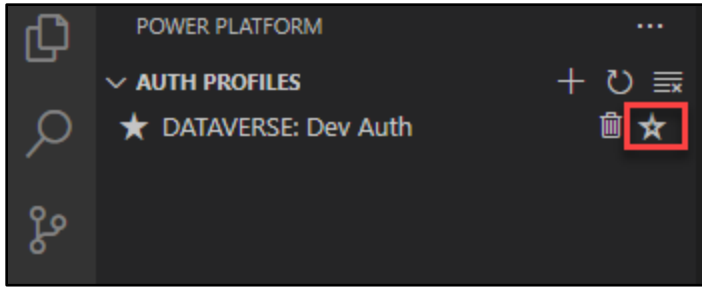
The PrioritZ Ask app will prepare a collection of the items to rank that will be bound as the dataset for the code component. When an item is dragged and dropped the code component will raise an OnSelect event that will be handled by the hosting app. The hosting app will update the collection items with their new rank. The code component will be stateless.

### Exercise 1 – Build Code Component

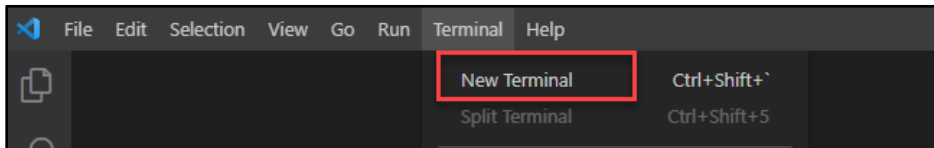
In this exercise, you will build the code component.

#### Task 1: Create the code component

1. Start Visual Studio Code.
2. Select the Power Platform tab and make sure your Dev Auth profile is selected. **NOTE:** The Power Platform tab is only available if you installed the Power Platform extension as explained in lab 0 and configured it in lab 1.

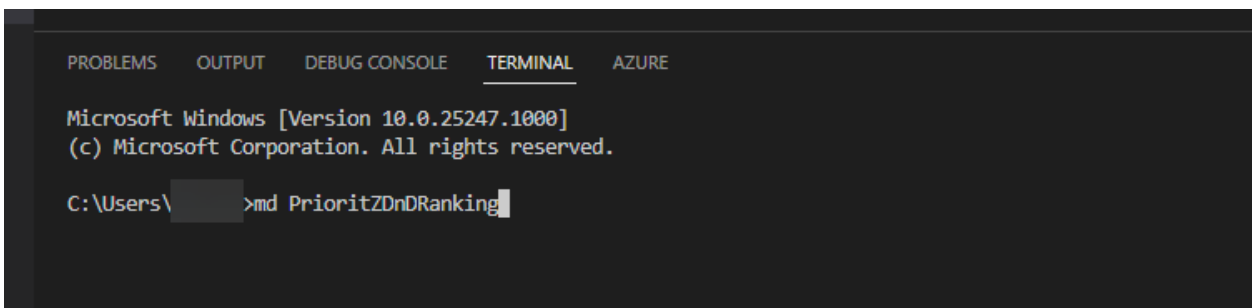


3. Click **Terminal** and select **New Terminal**.



4. In the Terminal window, make a new directory by running the command below.

```
md PrioritZDnDRanking
```

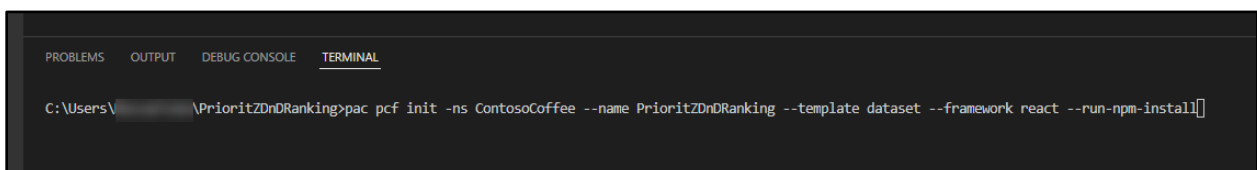


5. Run the command below to switch to the PrioritZDnDRanking directory you just created.

```
cd PrioritZDnDRanking
```

6. You should now be in the directory you created. Create a new component project and install dependencies by running the commands below. This will create the initial files for your code component.

```
pac pcf init -ns ContosoCoffee --name PrioritZDnDRanking --template dataset --framework react --run-npm-install
```



7. The component framework project should be created successfully.

```
C:\Users\...PrioritZDnDRanking>pac pcf init -ns ContosoCoffee --name PrioritZDnDRanking --template dataset --framework react --run-npm-install
The PowerApps component framework project was successfully created in 'C:\Users\...\PrioritZDnDRanking'.

Running 'npm install' for you...

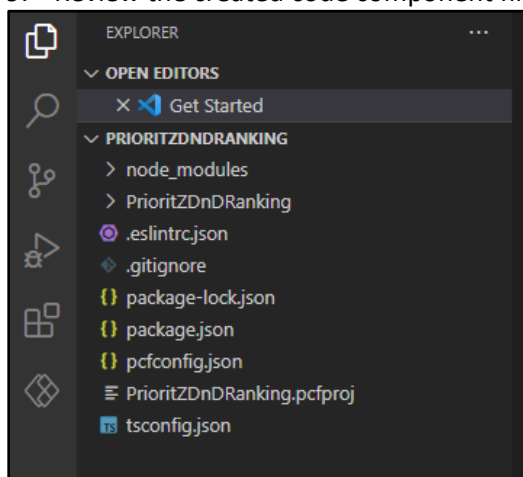
added 687 packages, and audited 688 packages in 58s
86 packages are looking for funding
run 'npm fund' for details
found 0 vulnerabilities

C:\Users\...PrioritZDnDRanking>
```

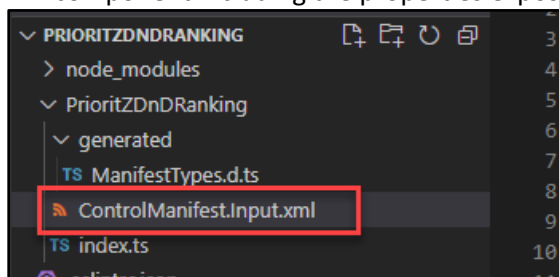
8. In the terminal, run the command below to open the project in your current Visual Studio Code session.

```
code -a .
```

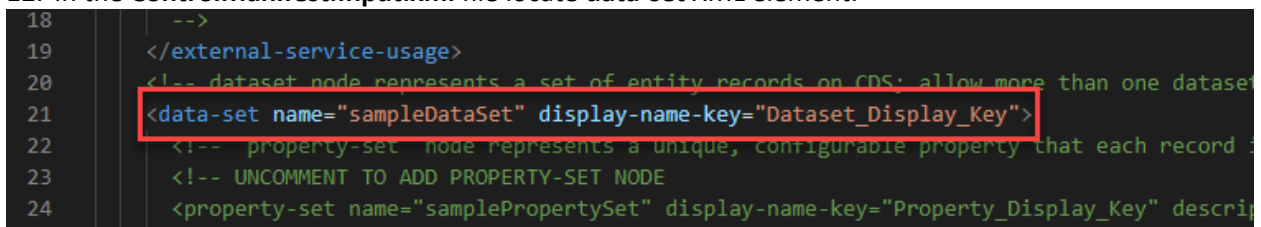
9. Review the created code component files.



10. Expand the **PrioritZDnDRanking** folder and then expand the component folder.
11. Open the **ControlManifest.Input.xml** file. The manifest is the metadata file that defines the component including the properties exposed to the hosting app.



12. In the **ControlManifest.Input.xml** file locate **data-set** XML element.



13. Change the **name** to **items** and the **display-name-key** to **items**. This defines the property the app will bind to a collection of items.

```
-->
</external-service-usage>
<!-- dataset node represents a set of entity records on CDS; allow more than one d
<data-set name="items" display-name-key="items">
  <!-- 'property-set' node represents a unique, configurable property that each re
  <!-- UNCOMMENT TO ADD PROPERTY-SET NODE
  <property-set name="samplePropertySet" display-name-key="Property_Display_Key" d
-->
```

14. Add the following properties after the closing tag of the data-set element **</data-set>**.

```
<property name="BackgroundColor" display-name-key="Background color"
usage="input" of-type="SingleLine.Text" default-value="#F3F2F1"/>
  <property name="DragBackgroundColor" display-name-key="Drag background
color" usage="input" of-type="SingleLine.Text" default-value="lightgreen"/>
  <property name="ItemHeight" display-name-key="Item height" usage="input"
of-type="Whole.None" default-value="32"/>
  <property name="FontSize" display-name-key="Font size" usage="input" of-
type="Whole.None" default-value="12"/>
  <property name="FontColor" display-name-key="Font color" usage="input" of-
type="SingleLine.Text" default-value="#333333"/>
```

```
<!-- dataset node represents a set of entity records on CDS; allow more than one datasets -->
<data-set name="items" display-name-key="items">
  <!-- 'property-set' node represents a unique, configurable property that each record in the dataset must provide. -->
  <!-- UNCOMMENT TO ADD PROPERTY-SET NODE
  <property-set name="samplePropertySet" display-name-key="Property_Display_Key" description-key="Property_Desc_Key" of-type="SingleLine.Text" usage
  -->
</data-set>

<property name="BackgroundColor" display-name-key="Background color" usage="input" of-type="SingleLine.Text" default-value="#F3F2F1"/>
<property name="DragBackgroundColor" display-name-key="Drag background color" usage="input" of-type="SingleLine.Text" default-value="lightgreen"/>
<property name="ItemHeight" display-name-key="Item height" usage="input" of-type="Whole.None" default-value="32"/>
<property name="FontSize" display-name-key="Font size" usage="input" of-type="Whole.None" default-value="12"/>
<property name="FontColor" display-name-key="Font color" usage="input" of-type="SingleLine.Text" default-value="#333333"/>

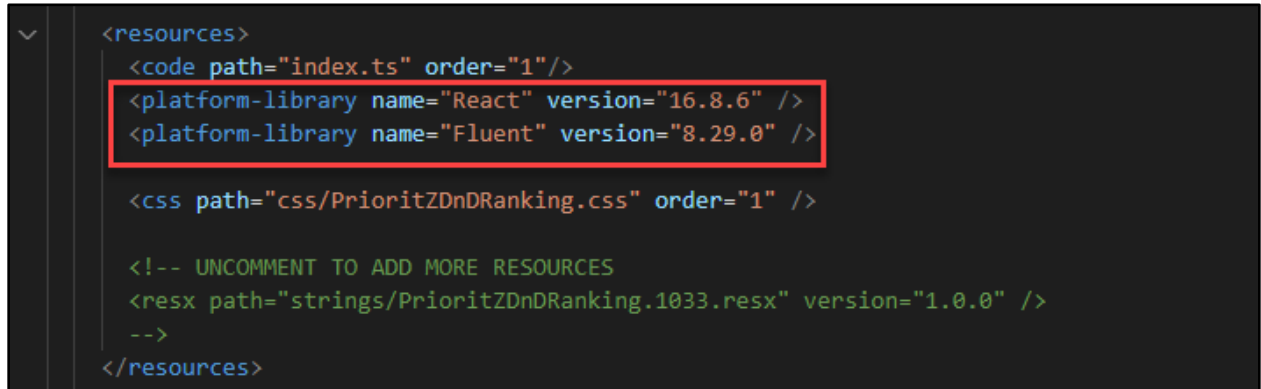
<resources>
  <code path="index.ts" order="1"/>
  <!-- UNCOMMENT TO ADD MORE RESOURCES
```

15. Locate **<resources>** and uncomment **css** resource. This will ensure that our styles will be bundled with the code component when it is deployed.

```
<resources>
  <code path="index.ts" order="1"/>
  <platform-library name="React" version="16.8.6" />
  <platform-library name="Fluent" version="8.29.0" />
  <css path="css/PrioritZDnDRanking.css" order="1" />
  <!-- UNCOMMENT TO ADD MORE RESOURCES
  <resx path="strings/PrioritZDnDRanking.1033.resx" version="1.0.0" />
  -->
</resources>
```

16. Notice the following two resources. This declares the component's dependency on these two libraries. This is a result of specifying `-framework React` on initialization.

```
<platform-library name="React" version="16.8.6" />
<platform-library name="Fluent" version="8.29.0" />
```



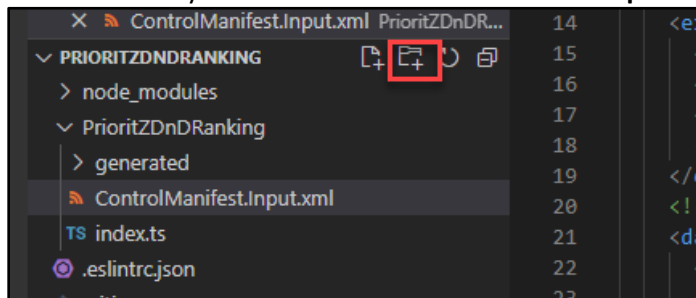
```
<resources>
  <code path="index.ts" order="1"/>
  <platform-library name="React" version="16.8.6" />
  <platform-library name="Fluent" version="8.29.0" />

  <css path="css/PrioritZDnDRanking.css" order="1" />

  <!-- UNCOMMENT TO ADD MORE RESOURCES
  <resx path="strings/PrioritZDnDRanking.1033.resx" version="1.0.0" />
  -->
</resources>
```

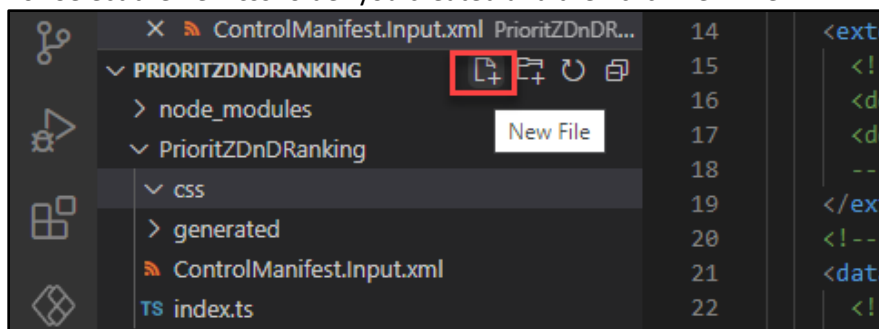
17. Click **File** and select **Save All**.

18. Make sure you still have the **ControlManifest.Input.xml** file selected and then click **New Folder**.



19. Name the new folder **css**.

20. Select the new **css** folder you created and then click **New File**



21. Name the new file **PrioritZDnDRanking.css**.

22. Paste the following css into the **PrioritZDnDRanking.css** file.

```
.prioritydnd-scroll-container {
    box-sizing: border-box;
    padding: 2px;
    overflow-y: auto;
```

```

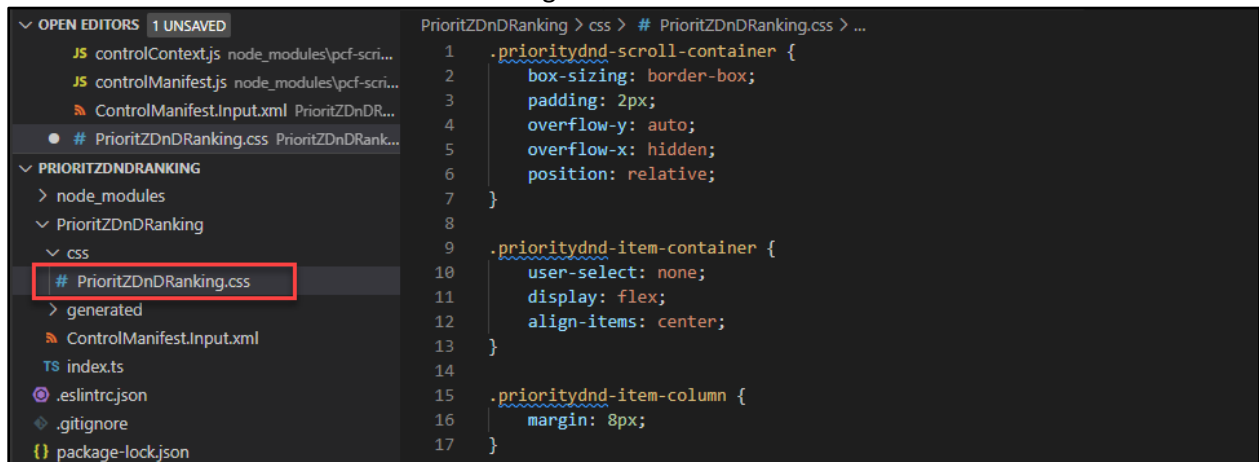
        overflow-x: hidden;
        position: relative;
    }

    .prioritydnd-item-container {
        user-select: none;
        display: flex;
        align-items: center;
    }

    .prioritydnd-item-column {
        margin: 8px;
    }
}

```

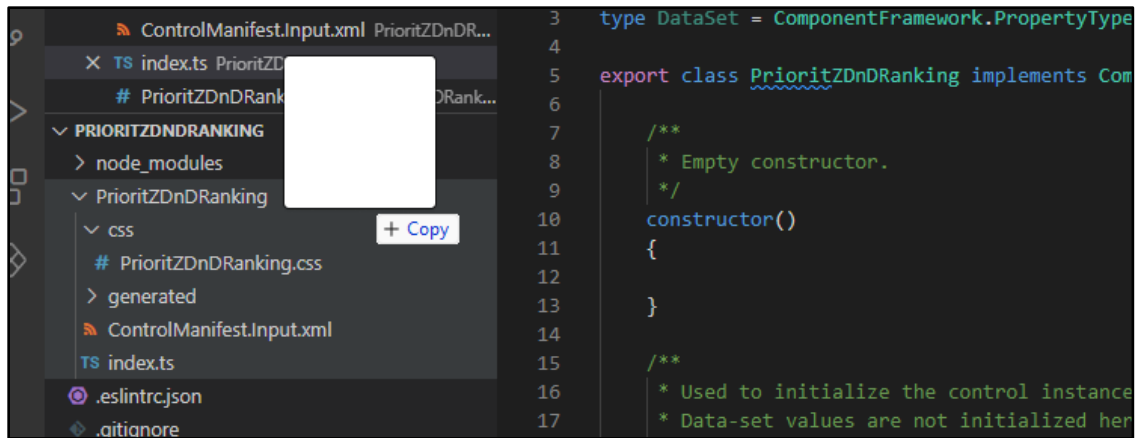
23. The file should now look like the following.



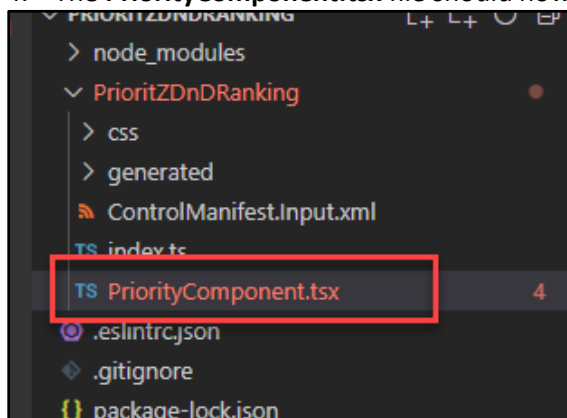
24. Click **File** and save your changes.

## Task 2: Implement the component logic

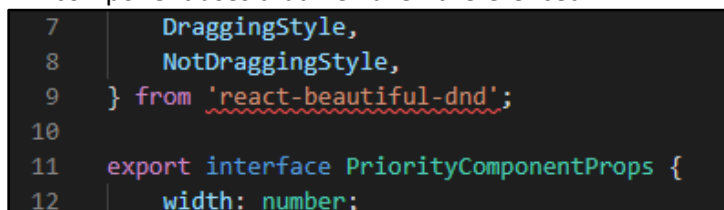
1. Delete the **HelloWorld.tsx** component file that is automatically created as we won't be using it.
2. Go to the lab resources folder.
3. Drag the **PriorityComponent.tsx** file and drop it in the **PrioriZDnDRanking** folder.



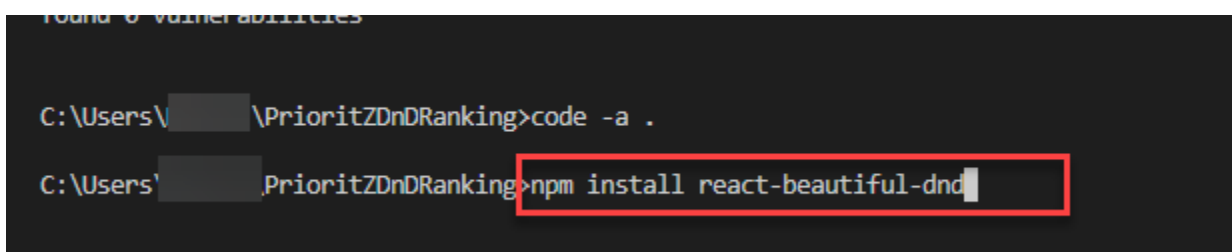
4. The **PriorityComponent.tsx** file should now be in the **PrioritZDnDRanking** folder.



5. Click **File** and save your changes.
6. Open the **PriorityComponent.tsx** and review the contents. This implements the React component that will be rendered to represent our draggable items.
7. Notice line 9 from 'react-beautiful-dnd' has a red underline. This is a npm package the component uses that we haven't referenced.



8. Run the following command in a terminal window to add a reference to react-beautiful-dnd





9. Add the following command for the type definitions.

```
npm install --save-dev @types/react-beautiful-dnd
```

10. Notice the red underline in line 9 has been resolved.

```
1 import * as React from 'react';
2 import {
3   DragDropContext,
4   Droppable,
5   Draggable,
6   DropResult,
7   DraggingStyle,
8   NotDraggingStyle,
9 } from 'react-beautiful-dnd';
10
11 export interface PriorityComponentProps {
```

11. Open the **index.ts** file.

12. Remove line 2 as we are no longer using HelloWorld.

```
import { HelloWorld, IHelloWorldProps } from './HelloWorld';
```

```
1 import { IInputs, IOutputs } from './generated/ManifestTypes';
2 import { HelloWorld, IHelloWorldProps } from './HelloWorld';
3 import * as React from 'react';
4
```

13. Add the import below to the **index.ts** file. This will reference the PriorityComponent.

```
import { PriorityComponent, PriorityComponentProps } from
'./PriorityComponent';
```

```
1 import { IInputs, IOutputs } from './generated/ManifestTypes';
2 import { PriorityComponent, PriorityComponentProps } from './PriorityComponent';
3 import * as React from 'react';
4
```

14. Locate the **export** class line in index.ts.

```
4
5 export class PriorityZDnRanking implements ComponentFramework.ReactControl<IInputs, IOutputs> {
6   private theComponent: ComponentFramework.ReactControl<IInputs, IOutputs>;
7   private notifyOutputChanged: () => void;
8
9   /**
10    * Empty constructor.
```

15. Add the following code below inside the **export** class. This defines some working variables you will be using in the class logic.

```
private context: ComponentFramework.Context<IInputs>;
private items: ComponentFramework.PropertyType.DataSet;
```

```

4
5 export class PriorityZDnDRanking implements ComponentFramework.ReactControl<IInputs, IOutputs> {
6     private context: ComponentFramework.Context<IInputs>;
7     private items: ComponentFramework.PropertyType.DataSet;
8
9     private theComponent: ComponentFramework.ReactControl<IInputs, IOutputs>;
10    private notifyOutputChanged: () => void;
11
12    /**

```

16. Locate the **init** function.

```

23
24    public init(context: ComponentFramework.Context<IInputs>, notifyOutputChanged: () => void, state: ComponentFramework.Dictionary): void {
25        this.notifyOutputChanged = notifyOutputChanged;
26    }
27
28    /**

```

17. Paste the code below inside the **init** function. This logic initializes our class variables from the runtime values and enables resize notification.

```

this.context = context;
context.mode.trackContainerResize(true);

```

```

24    public init(context: ComponentFramework.Context<IInputs>, notifyOutputC
25        this.context = context;
26        context.mode.trackContainerResize(true);
27
28        this.notifyOutputChanged = notifyOutputChanged;
29    }
30

```

18. Locate the **updateView** function.

```

35    /**
36    public updateView(context: ComponentFramework.Context<IInputs>): React.ReactElement {
37        const props: IHelloWorldProps = { name: 'Hello, World!' };
38        return React.createElement(
39            HelloWorld, props
40        );
41    }

```

19. Replace the **updateView** function with the function below. This logic creates the React Element from the PriorityComponent and adds it to the virtual DOM.

```

public updateView(context: ComponentFramework.Context<IInputs>):
React.ReactElement {
    const dataset = context.parameters.items;
    return React.createElement(PriorityComponent, {
        width: context.mode.allocatedWidth,
        height: context.mode.allocatedHeight,
        itemHeight: context.parameters.ItemHeight.raw,
        fontSize: context.parameters.FontSize.raw,
        fontColor: context.parameters.FontColor.raw,

```

```

        dataset: dataset,
        onReorder: this.onReorder,
        backgroundColor: this.context.parameters.BackgroundColor.raw,
        dragBackgroundColor:
this.context.parameters.DragBackgroundColor.raw,
    } as PriorityComponentProps);
}

```

```

/**
 * Called when any value in the property bag has changed. This includes field values, data-sets, global values such as
 * @param context The entire property bag available to control via Context Object; It contains values as set up by the
 */
public updateView(context: ComponentFramework.Context<IInputs>): React.ReactElement {
    const dataset = context.parameters.items;
    return React.createElement(PriorityComponent, {
        width: context.mode.allocatedWidth,
        height: context.mode.allocatedHeight,
        itemHeight: context.parameters.ItemHeight.raw,
        fontSize: context.parameters.FontSize.raw,
        fontColor: context.parameters.FontColor.raw,
        dataset: dataset,
        onReorder: this.onReorder,
        backgroundColor: this.context.parameters.BackgroundColor.raw,
        dragBackgroundColor: this.context.parameters.DragBackgroundColor.raw,
    } as PriorityComponentProps);
}

```

20. Add the function below after the **destroy** function. This logic handles the onReorder event from the PriorityComponent and identifies the involved items to the hosting app as selected items.

```

onReorder = (sourceIndex: number, destinationIndex: number): void => {
    const dataset = this.context.parameters.items;
    const sourceId = dataset.sortedRecordIds[sourceIndex];
    const destinationId = dataset.sortedRecordIds[destinationIndex];
    // raise the OnSelect event

this.context.parameters.items.openDatasetItem(dataset.records[sourceId].getName
dReference());
    // set the SelectedItems property
    this.context.parameters.items.setSelectedRecordIds([sourceId,
destinationId]);
};

```

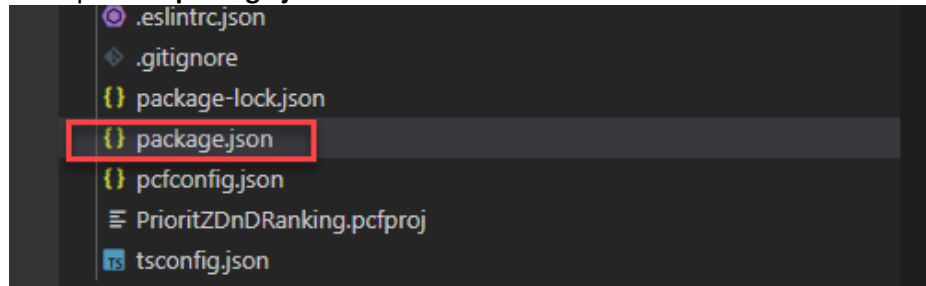
```

public destroy(): void {
    // Add code to cleanup control if necessary
}

onReorder = (sourceIndex: number, destinationIndex: number): void => {
    const dataset = this.context.parameters.items;
    const sourceId = dataset.sortedRecordIds[sourceIndex];
    const destinationId = dataset.sortedRecordIds[destinationIndex];
    // raise the OnSelect event
    this.context.parameters.items.openDatasetItem(dataset.records[sourceId].getNamedReference());
    // set the SelectedItems property
    this.context.parameters.items.setSelectedRecordIds([sourceId, destinationId]);
};

```

21. Open the **package.json** file.



22. Locate the **dependencies** JSON object.

```

10     "refreshTypes": "pcf-scripts refreshTypes"
11   },
12   "dependencies": {
13     "@fluentui/react": "8.29.0",
14     "react": "16.8.6",
15     "react-beautiful-dnd": "^13.1.0",
16     "react-dom": "16.8.6"
17   },
18   "devDependencies": {

```

23. Replace **dependencies** with the JSON below.

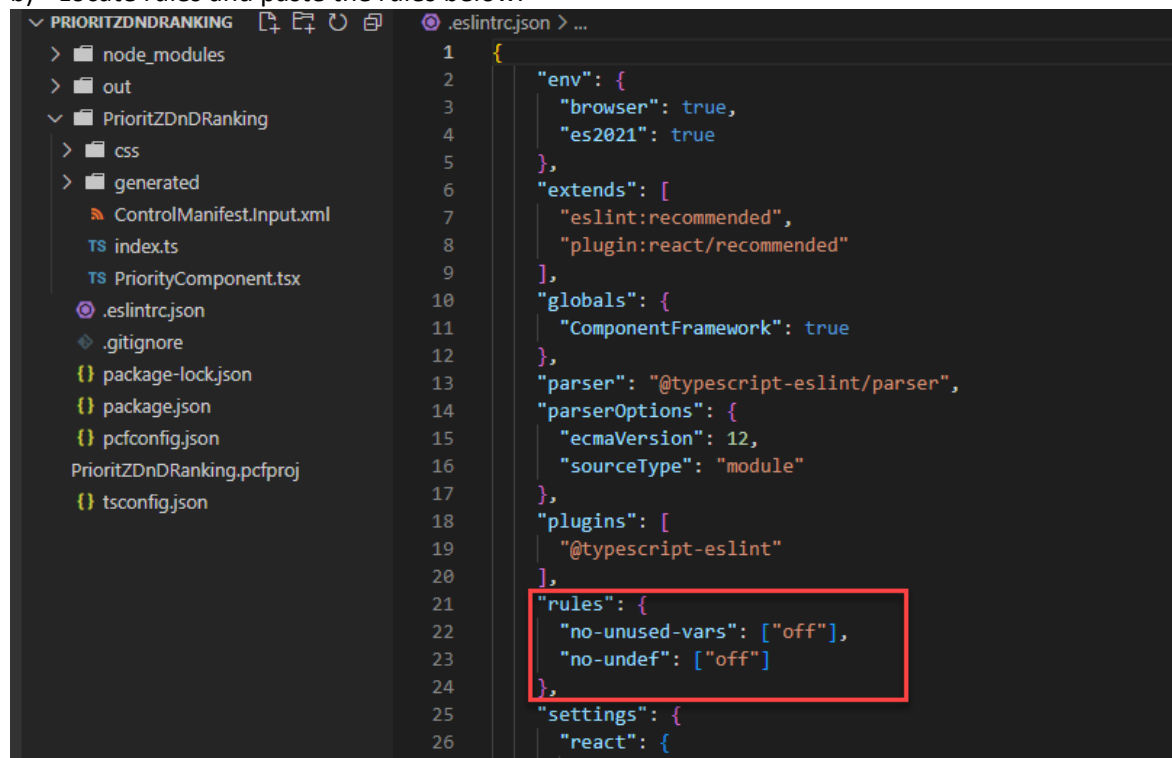
```

"dependencies": {
  "@fluentui/react": "8.29.0",
  "eslint-config-prettier": "^8.5.0",
  "eslint-plugin-prettier": "^4.0.0",
  "eslint-plugin-react": "^7.29.4",
  "eslint-plugin-react-hooks": "^4.4.0",
  "eslint-plugin-sonarjs": "^0.13.0",
  "prettier": "^2.6.1",
  "react": "16.8.6",
  "react-beautiful-dnd": "^13.1.0",
  "react-dom": "16.8.6"
},

```

24. Make the following modification to the file .eslintrc.json to work around a new lint rule that was too strict.

- a) Open the .eslintrc.json file.
- b) Locate rules and paste the rules below.



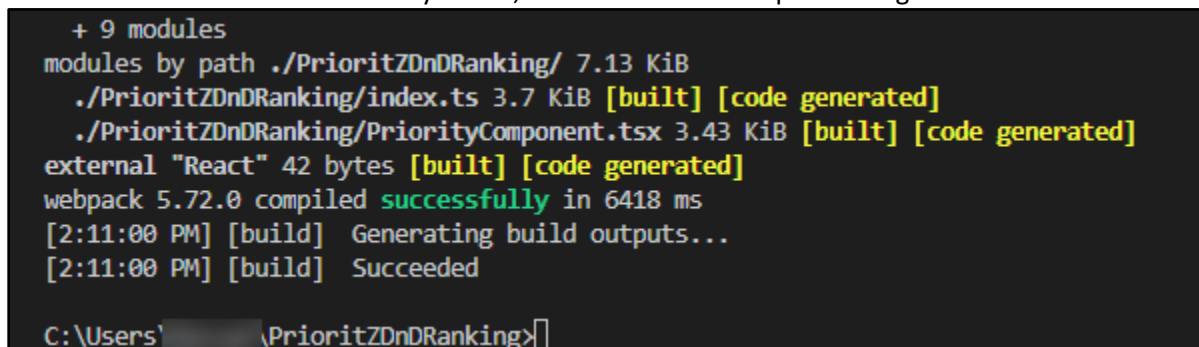
```
1  {
2    "env": {
3      "browser": true,
4      "es2021": true
5    },
6    "extends": [
7      "eslint:recommended",
8      "plugin:react/recommended"
9    ],
10   "globals": {
11     "ComponentFramework": true
12   },
13   "parser": "@typescript-eslint/parser",
14   "parserOptions": {
15     "ecmaVersion": 12,
16     "sourceType": "module"
17   },
18   "plugins": [
19     "@typescript-eslint"
20   ],
21   "rules": {
22     "no-unused-vars": ["off"],
23     "no-undef": ["off"]
24   },
25   "settings": {
26     "react": {
```

25. Click **File** and save all your changes.

26. Go to the terminal and run the command below. This will build your component and identify any problems.

```
npm run-script build
```

27. The build should succeed. If any errors, resolve them before proceeding.



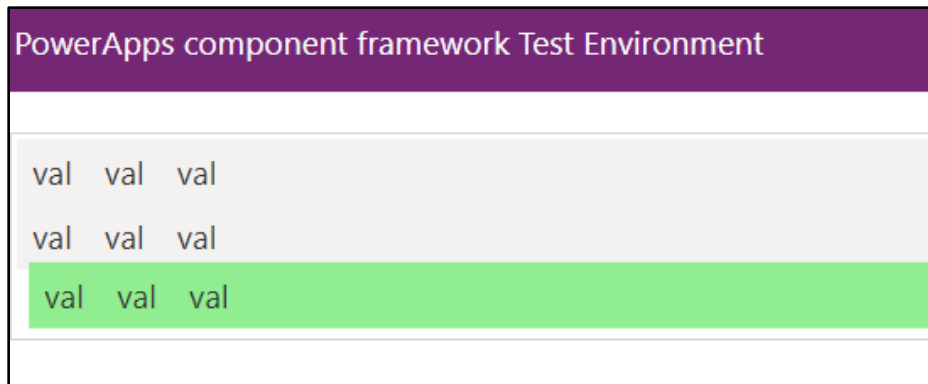
```
+ 9 modules
modules by path ./PrioritZDnDRanking/ 7.13 KiB
  ./PrioritZDnDRanking/index.ts 3.7 KiB [built] [code generated]
  ./PrioritZDnDRanking/PriorityComponent.tsx 3.43 KiB [built] [code generated]
external "React" 42 bytes [built] [code generated]
webpack 5.72.0 compiled successfully in 6418 ms
[2:11:00 PM] [build] Generating build outputs...
[2:11:00 PM] [build] Succeeded

C:\Users\...\PrioritZDnDRanking>
```

28. Run the command below to start the test harness.

```
npm start
```

29. The test harness should start. Try dragging the items and see if the behavior functions as expected. The data shown will be the default test harness data.



30. Close the test harness.
31. Stop the run by holding the **[CONTROL]** key + **C**.
32. Type **Y** and **[ENTER]**.

```
[Browsersync] Serving files from: C:\Users\
[Browsersync] Watching files...
Terminate batch job (Y/N)? Y
```

25. Run the command below to see your currently selected environment.

```
pac org who
```

26. You should have the dev environment you created selected. If not, run the select command from the end of lab one again.

```
C:\Users\>pac org who
Connected to...Lab Admin16's Environment
Organization Information
Org ID: c6d006e8-a634-4ffc-9436-46c3aeddac63
Unique Name: unqc6d006e8a6344ffc943646c3aedda
Friendly Name: Lab Admin16's Environment
Org URL: https://org0554957d.crm.dynamics.com/
User Email: labadmin16@augustenv.onmicrosoft.com
User ID: 58604ef5-452b-ed11-9db0-000d3a9ce6da
Environment ID: b6bd82c6-4c8e-e867-b85f-605e4fa68312

C:\Users\
```

33. Run the command below to push the component to your environment.

```
pac pcf push --publisher-prefix contoso
```

34. Wait for the solution to be imported and published to your environment.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  AZURE

Dropping temporary solution wrapper zip file: done.
File at C:\Users\...\PrioritZDnRanking\obj\PowerAppsTools_contoso\bin\Debug\PowerAppsTools_contoso.zip.
Importing the temporary solution wrapper into the current org.

Solution Importing...

Publishing All Customizations...

Published All Customizations.
Importing the temporary solution wrapper into the current org: done.
Updating the control in the current org: done.
PS C:\Users\...PrioritZDnRanking> |
```

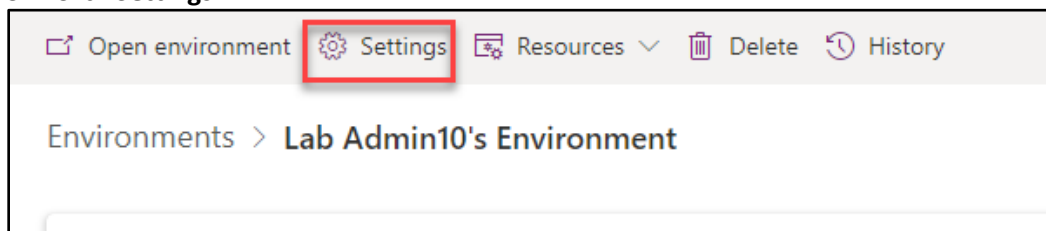
## Exercise 2 – Use the Code Component

In this exercise, you will use the code component you created in the PrioritZ Ask canvas application.

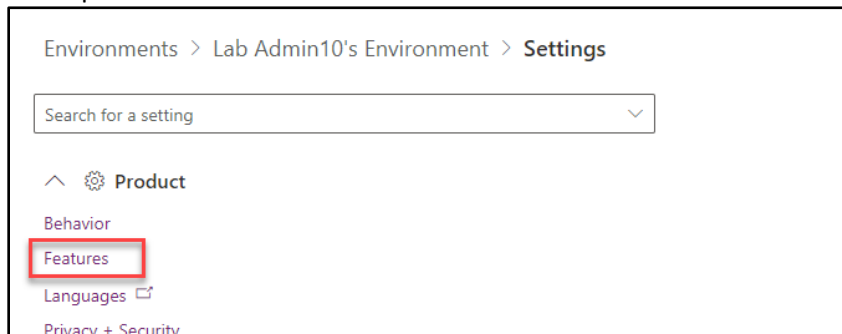
### Task 1: Allow Power Apps component framework

In this task, you will allow publishing of canvas apps with code components for your environment.

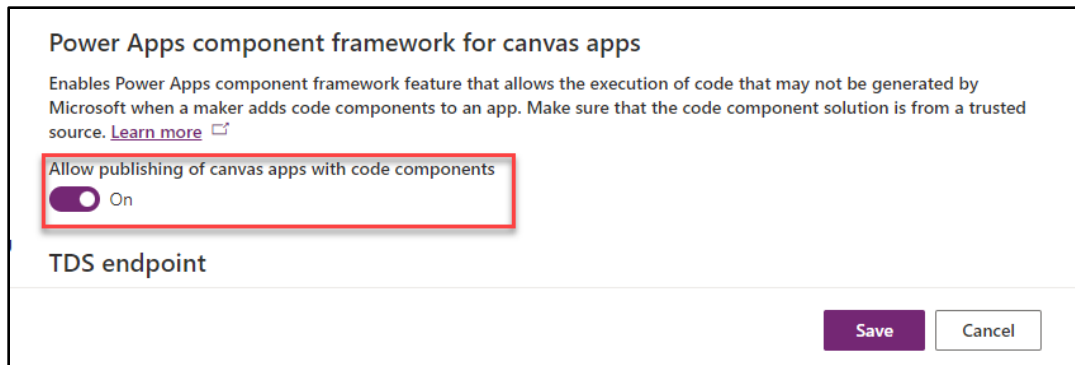
1. Navigate to [Power Platform admin center](#) and select environments.
2. Open the dev environment you are using for this lab.
3. Click **Settings**.



4. Expand **Products** and select **Features**.



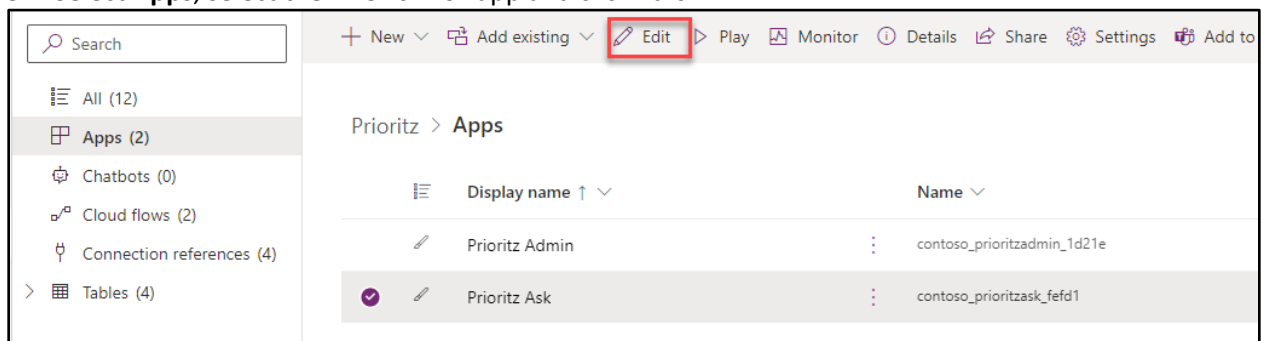
5. Turn on **Allow publishing of canvas apps with code components** and click **Save**.



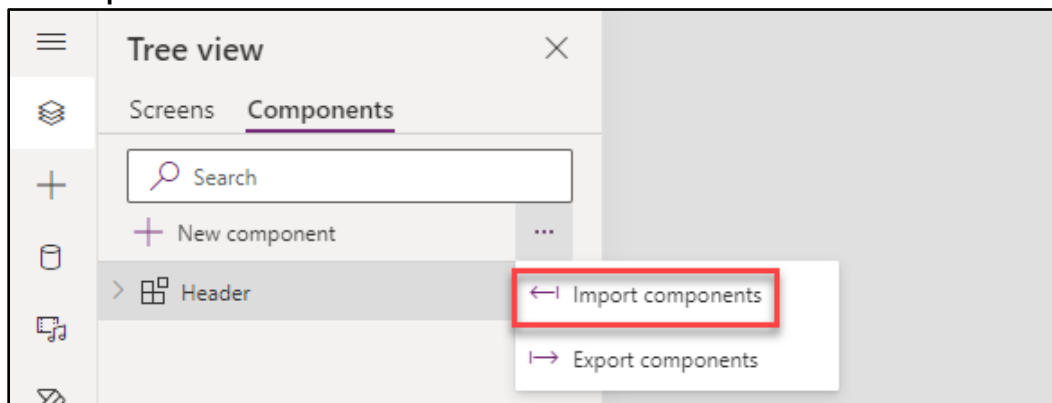
## Task 2: Edit canvas app

In this task, you will edit the PrioritZ Ask canvas application to use the code component you created.

1. Navigate to [Power Apps maker portal](#) and make sure you are in the correct dev environment.
2. Select **Solutions** and open the **PrioritZ** solution.
3. Select **Apps**, select the **PrioritZ Ask** app and click **Edit**.

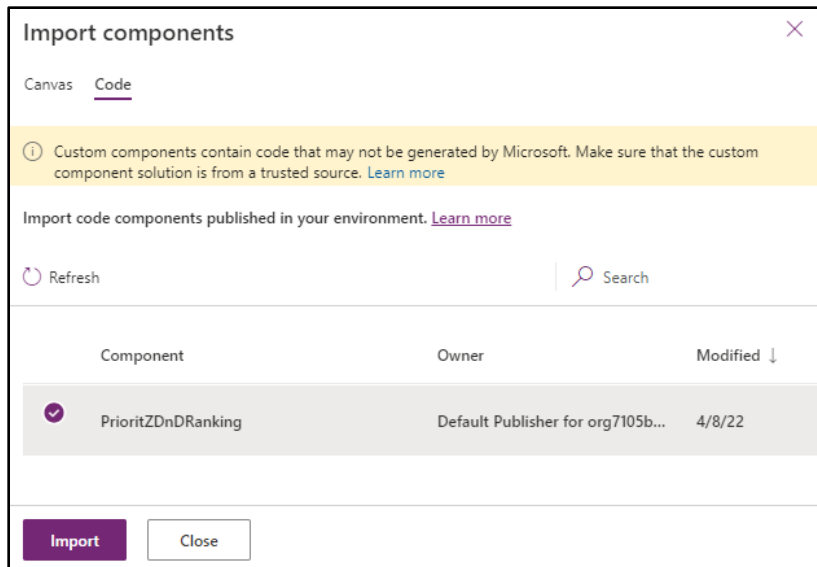


4. Select the **Components** tab, click on the ... **Components option button** and select **Import components**.

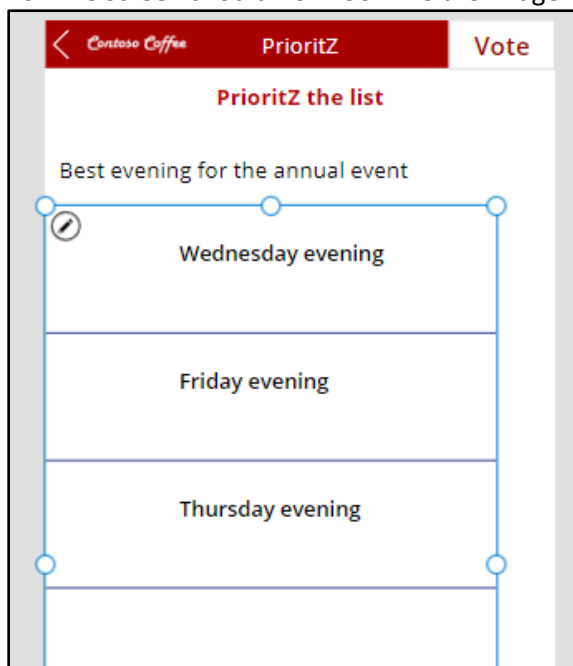


5. Select the **Code** tab.
6. Select the code component you created and click **Import**.

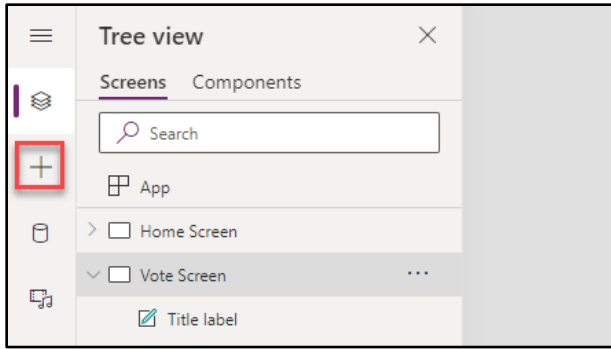




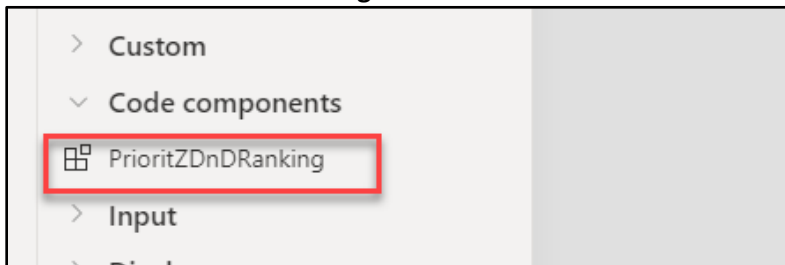
7. Select the **Screens** tab.
8. Expand the **Vote Screen** and select the **Votes gallery**.
9. Set the **Width** value of the Votes gallery to **570**.
10. The screen should now look like the image below.



11. Select the **Votes Screen** and click **+ Insert**.



12. Select **PrioritZDnDRanking**.



13. Go to the Tree view tab and select the **PrioritZDnDRanking** you just added.

14. Set the **Items** value of the **PrioritZDnDRanking** component to the formula below.

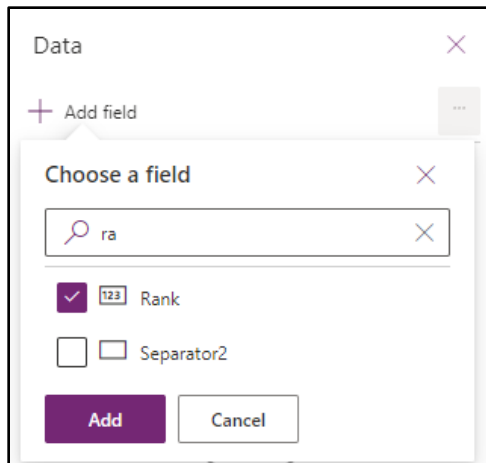
```
'Votes gallery'.AllItems
```

15. Select the **PrioritZDnDRanking** go to the **Properties** pane and click **Edit Fields**.



16. Click **+ Add field**.

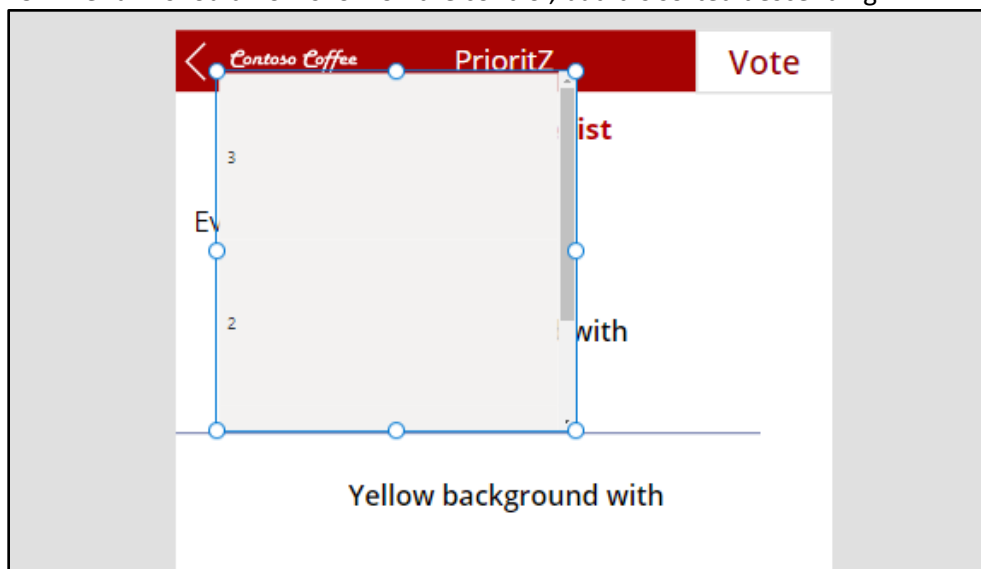
17. Select **Rank** and click **Add**.



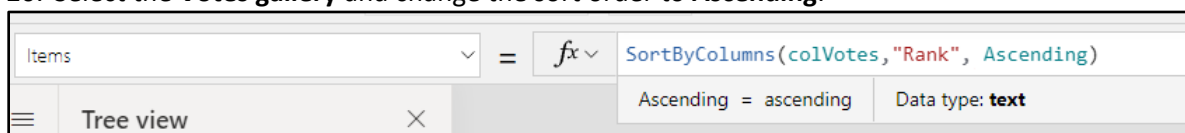
18. Set the **ItemHeight** of the **PrioritZDnDRanking** to the formula below.

`'Votes gallery'.TemplateHeight + 2`

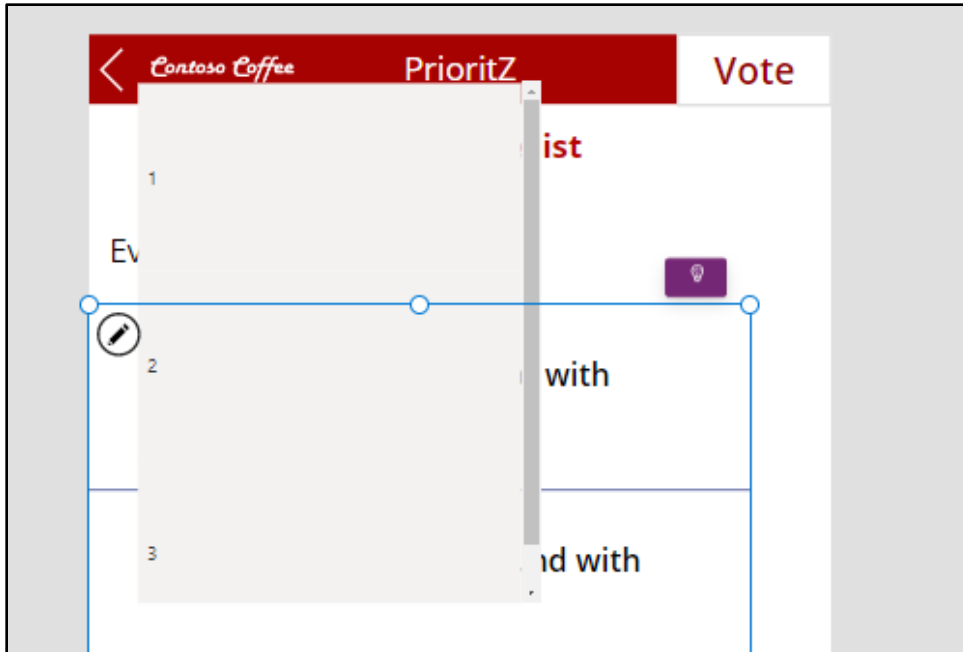
19. The rank should now show on the control, but it is sorted descending.



20. Select the **Votes gallery** and change the sort order to **Ascending**.



21. The rank should now get sorted ascending.



22. Select the **PrioritZDnDRanking** component.

23. Set the **X** value of the **PrioritZDnDRanking** component to the formula below.

```
'Votes gallery'.Width
```

24. Set the **Width** value of the **PrioritZDnDRanking** component to **60**.

25. Set the **Height** value of the **PrioritZDnDRanking** component to the formula below.

```
'Votes gallery'.Height
```

26. Set the **BackgroundColor** value of the **PrioritZDnDRanking** component to **"#99CCFF"**.

27. Set the **DragBackgroundColor** value of the **PrioritZDnDRanking** component to **"#A70202"**.

28. Set the **Y** value of the **PrioritZDnDRanking** component to the formula below.

```
'Votes gallery'.Y
```

29. Set the **OnSelect** value of the **PrioritZDnDRanking** component to the formula below.

```
With({
    sourceRank:First(Self.SelectedItems).Rank,
    destinationRank:Last(Self.SelectedItems).Rank
},
If(sourceRank<destinationRank,
    // Moving Up
    UpdateIf(colVotes,Rank>=sourceRank && Rank<=destinationRank,
    {
        Rank:If(Rank<>sourceRank,Rank-1,destinationRank)
    }
    );
```

```

);

If(sourceRank>destinationRank,
    // Moving Down
    UpdateIf(colVotes,Rank>=destinationRank && Rank<=sourceRank,
    {
        Rank:If(Rank<>sourceRank,Rank+1,destinationRank)
    }
);

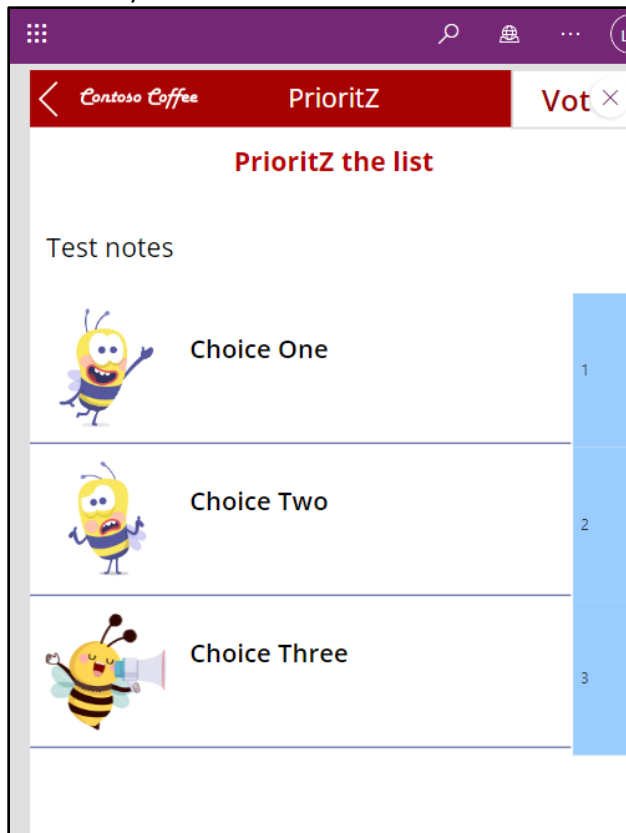
);
);

```

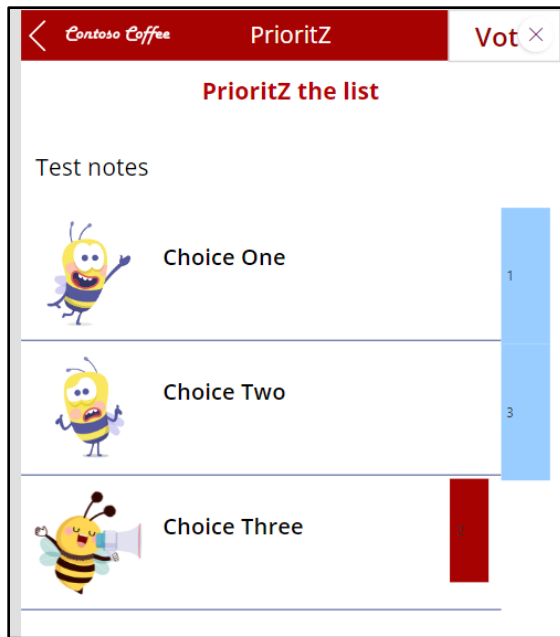
30. Select the **Home Screen** and click **Play**.

31. Select one of the topics.

32. Make your browser window smaller until it is the size of a phone screen.



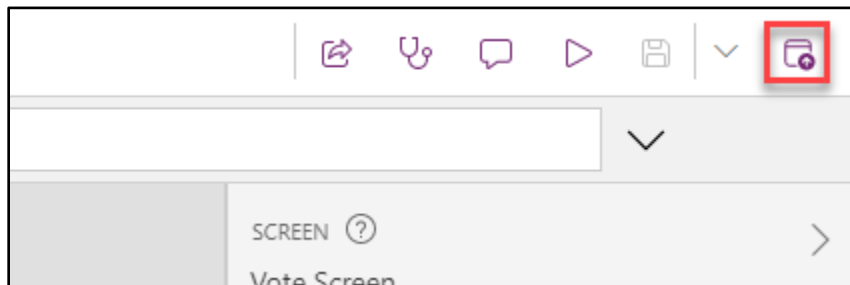
33. Drag one of the topic items and drop it in a different location.



34. The drag/drop should work as expected.

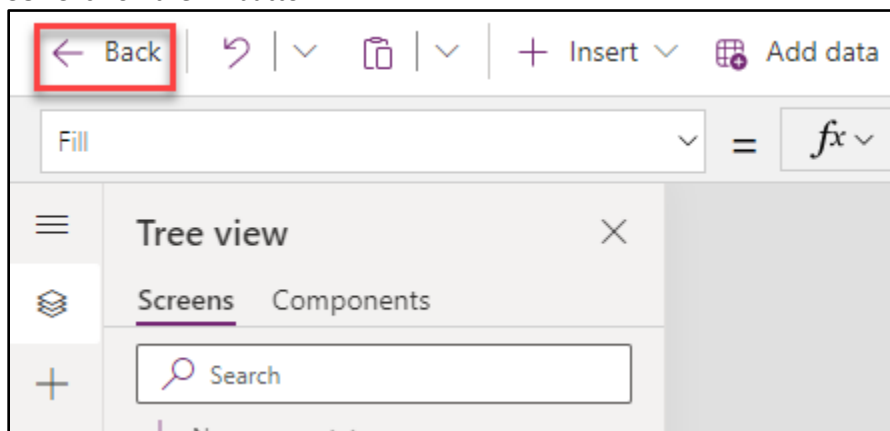
35. Close the preview.

36. Click **Publish**.



37. Select **Publish this version** and wait for the publish to be completed.

38. Click on the ← button.



39. Select **Leave** if prompted.

40. You should now be back on the solution.

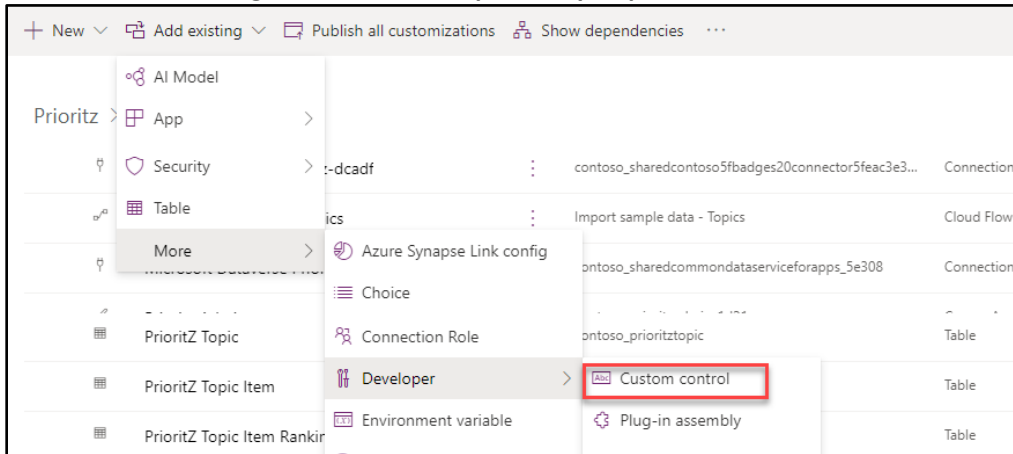
41. Do not navigate away from this page.

### Exercise 3 – Add Code Component to Solution

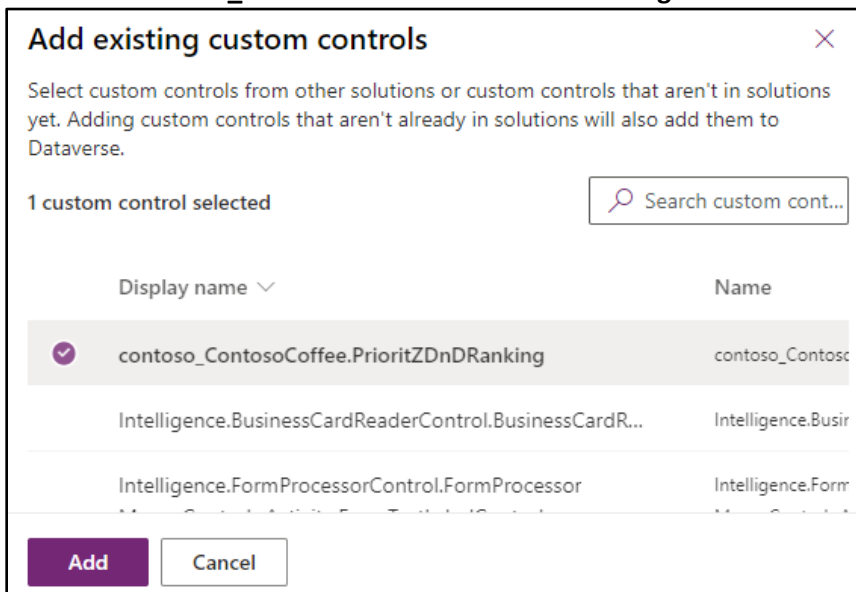
In this exercise, you will add the code component you created to the PrioritZ solution. This will ensure later when we move from dev to test our component is included.

#### Task 1: Add component to solution

1. Make sure you are still in the **PrioritZ** solution.
2. Click **Add existing** and select **More | Developer | Custom control**.



3. Select **contoso\_ContosoCoffee.PrioritZDnDRanking** and click **Add**.



4. The custom control should now be in your solution.

