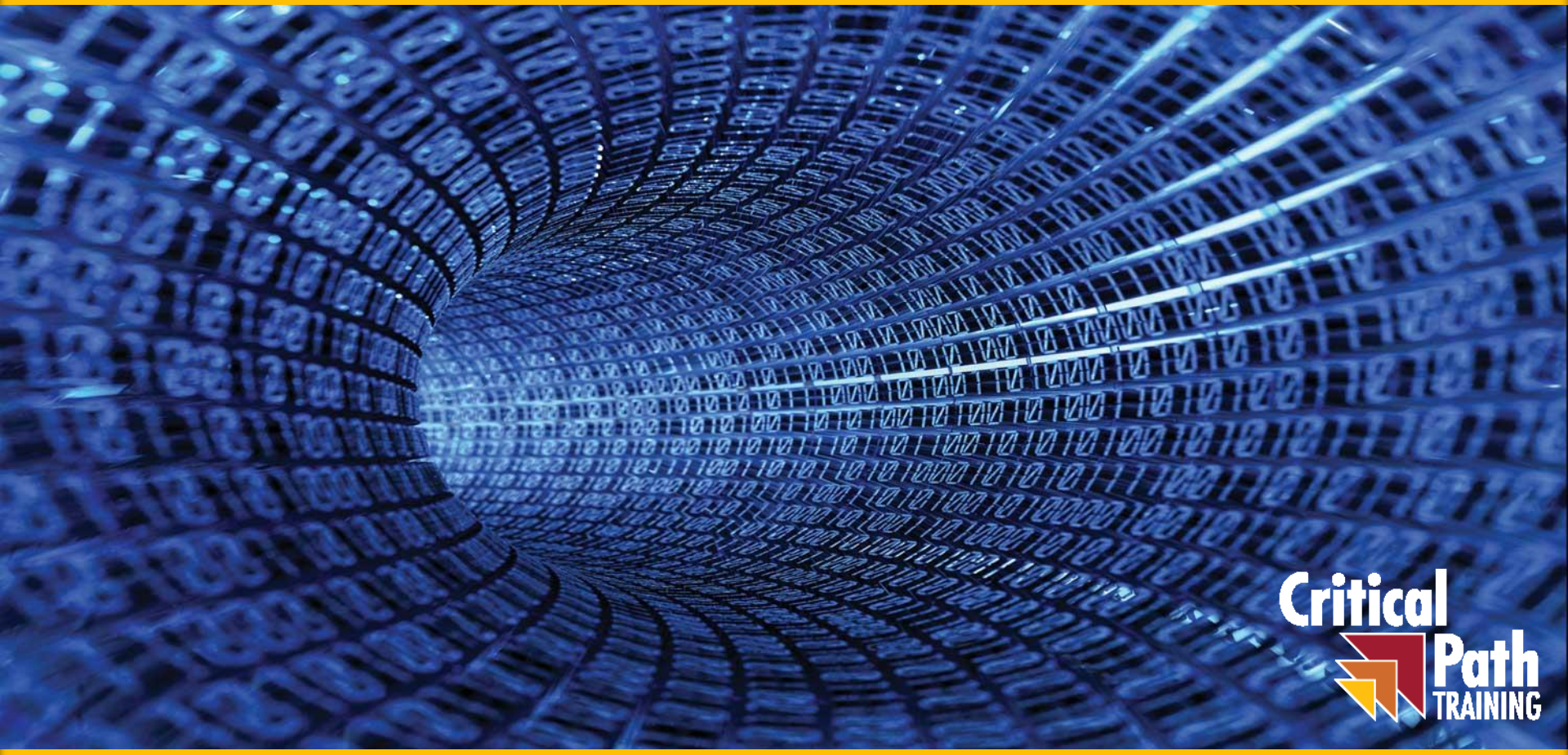
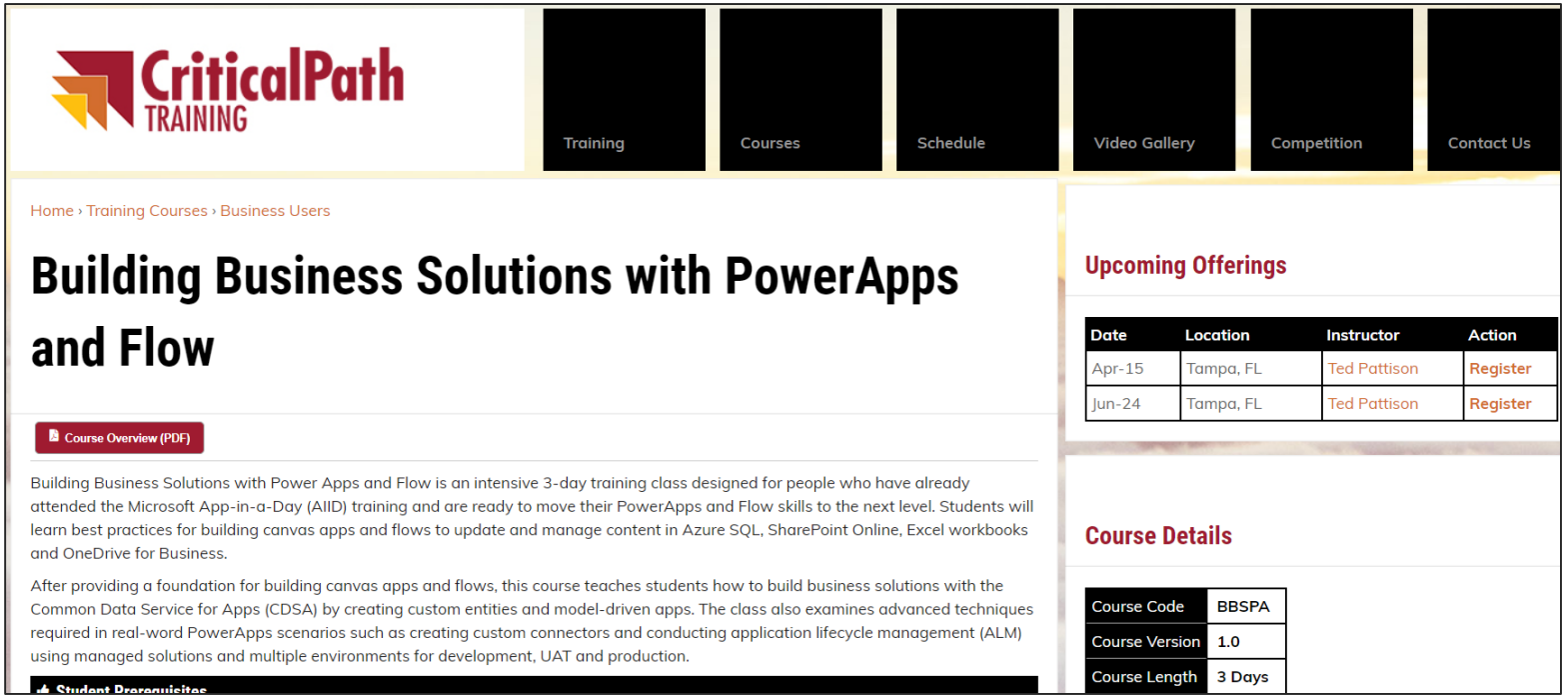
Professional Developer's Introduction to the PowerApps and Flow

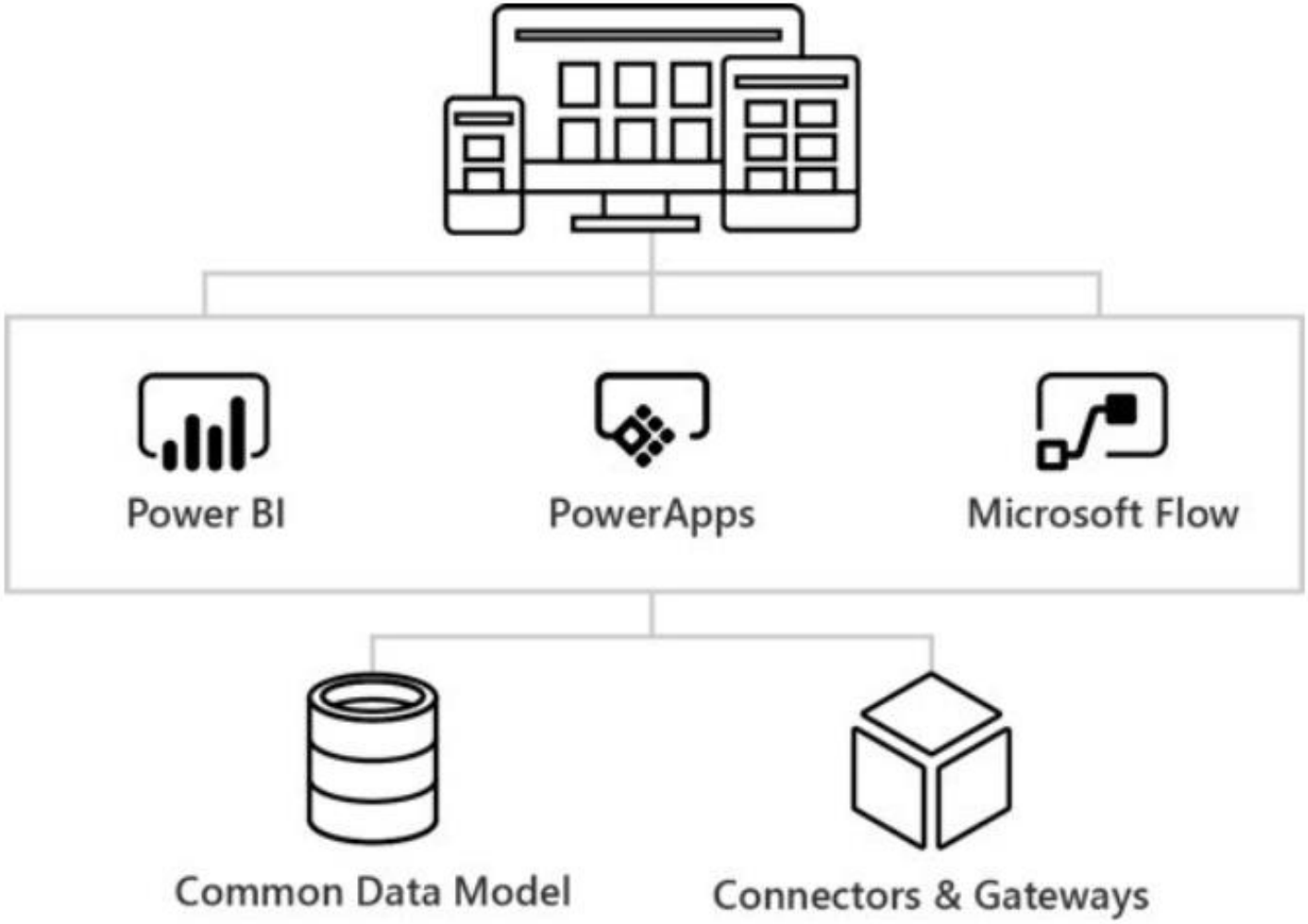


Our Newest PowerApps Training Course

- Advanced PowerApps and Flow training
  - Designed for students who have already attended Microsoft's App-in-a-day
  - Three days filled with plenty of hands-on lab exercises
  - Learn advanced builder skills & best practices
  - More info at https://CriticalPathTraining.com



Power Platform Overview



What Can You Build with PowerApps?

- Canvas Apps
- Connections
- Flows
- Custom Connectors
- PowerApps Components
- PowerApps Controls
- Common Data Service for Apps (CDS)
- Model-driven Apps

Examining the Customer Ordering Canvas App

Challenges with Building Canvas Apps

- Who is the typical PowerApps application maker?
  - A developer without a software development background
- What is the maker is responsible for?
  - designing, building, testing, deploying
- What are the challenges in deploying canvas apps?
  - Building projects that are easy to maintain and extend
  - Building consistency across team members
  - Building canvas apps with better performance

Read the Canvas Apps Whitepaper

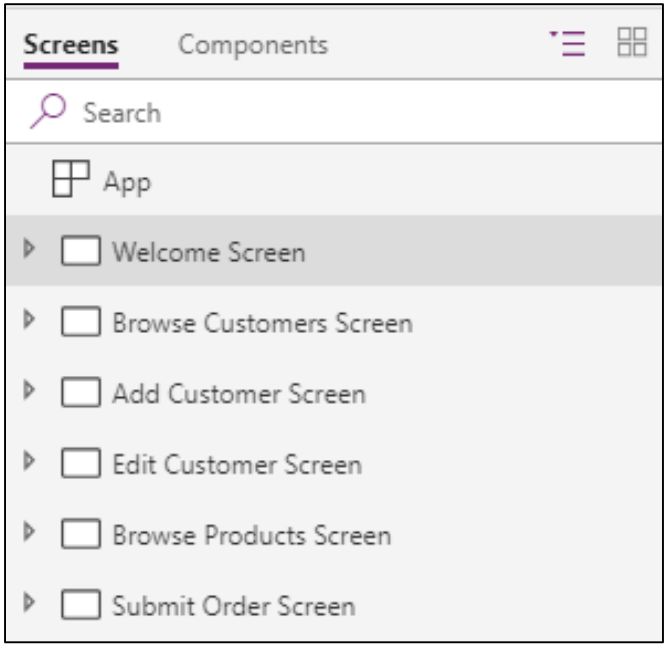
- PowerApps canvas app coding standards and guidelines
  https://powerapps.microsoft.com/en-us/blog/powerapps-canvas-app-coding-standards-and-guidelines/

Todd Baginski

Pat Dunn

Creating Screen Names

- It's import to create screen names correctly
  - Screen names read aloud by screen readers
  - Names should include spaces and avoid abbreviations
  - Screen name should end with the word "Screen"
  - Screen name should reflect purpose of screen



Control Naming Convention

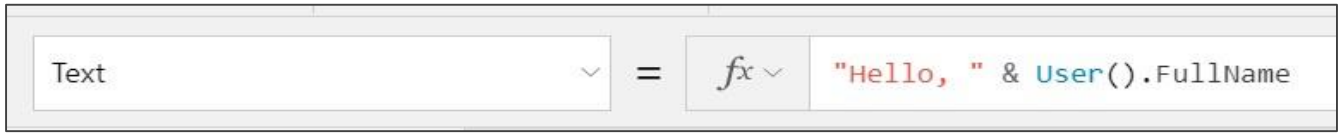
- Define and document a naming convention
  - Use this list as a starting point

button	btn
camera control	cam
canvas	can
card	crd
collection	col
combo box	cmb
date picker	dte
drop down	drp
radio button	rad
form	frm
gallery	gal

group	grp
header page shape	fdr
html text	html
icon	ico
image	img
label	lbl
page section shape	sec
shapes (rectangle, etc.)	shp
table data	tbl
text input	txt
timer	tim

PowerApps Formula Language

- PowerApps provides its own Formula Language
  - Designed to be as similar as possible to Excel Formula language
  - PowerApps Formula Language includes built-in set of functions
- You write formulas for specific properties
  - Set the Text property for a label



  - Set the Color property of the label text

  - Write an formula to filter the items shown in a gallery

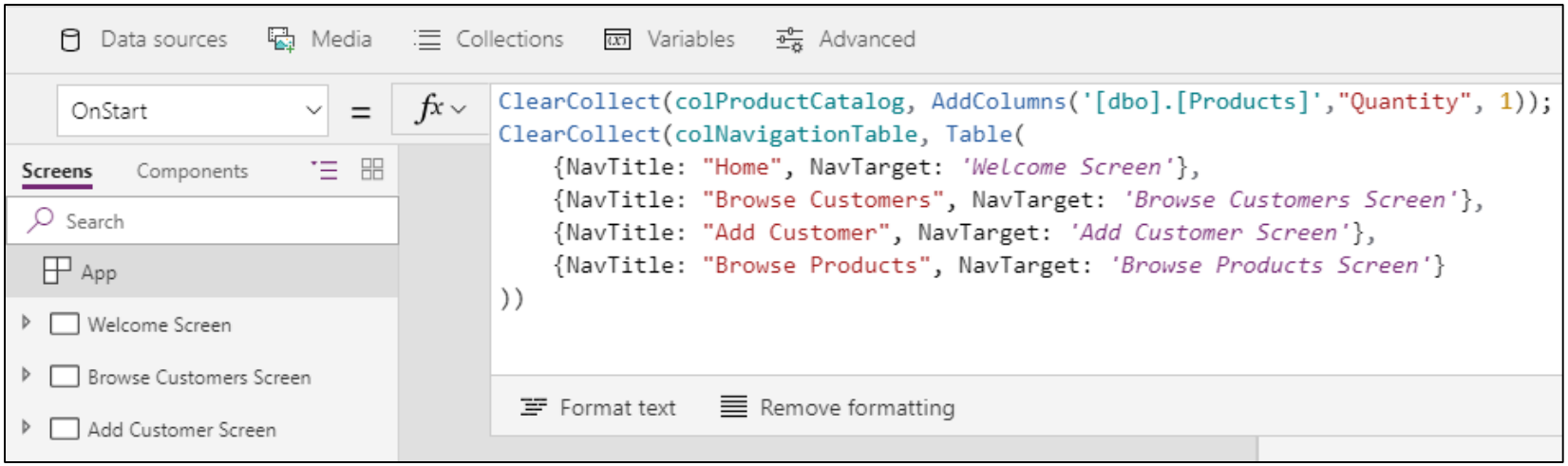
Events and State Changes

- Formulas for event properties can contain imperative logic
  - `OnSelect`, `OnVisible`, `OnStart`, etc.
- Imperative logic is used to take action
  - Set value of global variable or context variable
  - Add item to a collection
  - Navigate between screens
  - Submit data to server
- You can chain actions together with chaining operator (;)

```
Collect(Customers, {ID: NextCustomerId, FirstName: FirstNameTextInput.Text, LastName: LastNameTextInput.Text});
Reset(FirstNameTextInput);
Reset(LastNameTextInput);
Set(NextCustomerId, NextCustomerId+1);
```

App OnStart

- App **OnStart** property used to initialize state in app
  - Event provides support for initializing state in app at startup
  - Commonly used to initialize global variables and collections
  - Right-click **App** in left navigation to run **OnStart** while in editor



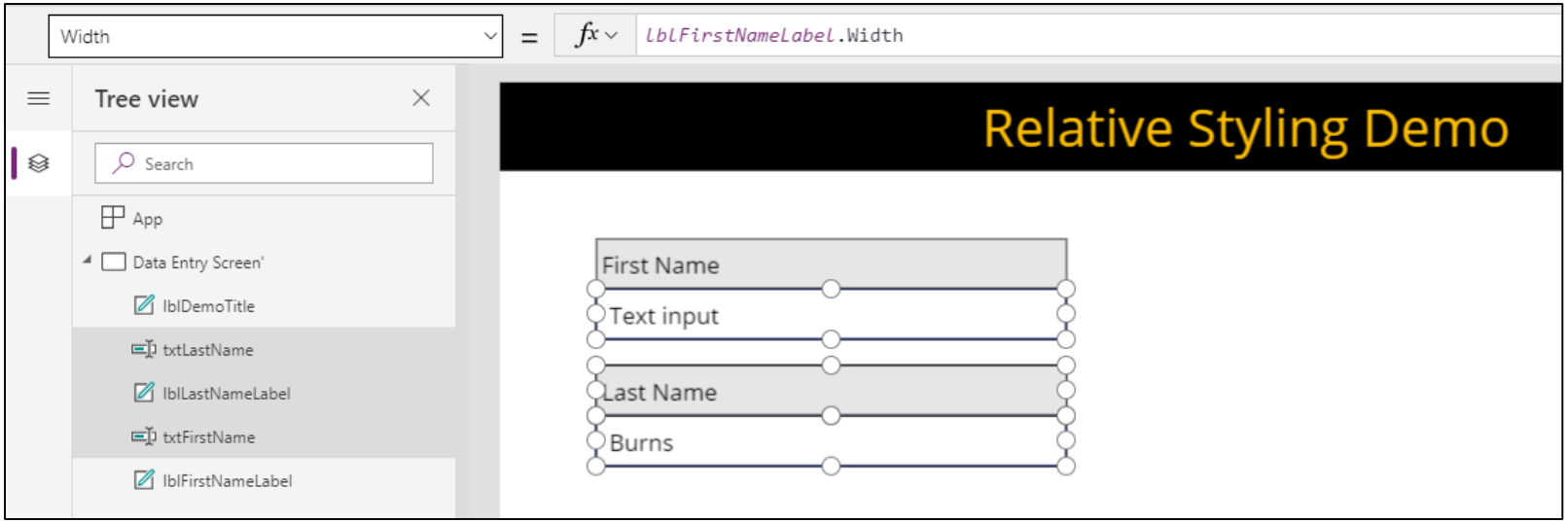
Declarative vs Imperative Functions

- Non-highlighted functions used to return values
  - These are declarative functions
- Highlighted functions used to perform actions
  - These are imperative functions

Abs	Collect	Day	HashTags	Max	Rand	Shuffle	TrimEnds
Acceleration	Color	Defaults	Hour	Mid	Refresh	Sin	Ungroup
Acos	ColorFade	Degrees	If	Min	Remove	Sort	Update
Acot	ColorValue	Disable	IfError	Minute	RemoveIf	SortByColumns	UpdateContext
AddColumns	Compass	Distinct	IsBlank	Mod	RenameColumns	Split	UpdateIf
And	Concat	Download	IsEmpty	Month	Replace	Sqrt	Upper
App	Concatenate	DropColumns	IsMatch	Navigate	Reset	StartsWith	User
Asin	Connection	EditForm	IsNumeric	NewForm	ResetForm	StdevP	Validate
Atan	Count	Enable	IsToday	Not	Revert	Substitute	Value
Atan2	Cos	EndsWith	Language	Notify	RGBA	SubmitForm	VarP
Average	Cot	Errors	Last	Now	Right	Sum	ViewForm
Back	CountA	EncodeUrl	LastN	Operators	Round	Switch	Weekday
Blank	CountIf	Exit	Launch	Or	RoundDown	Table	Year
Calendar	CountRows	Exp	Left	Param	RoundUp	Tan	
Char	DataSourceInfo	Filter	Len	Patch	SaveData	Text	
Choices	Date	Find	Ln	Pi	Search	Time	
Clear	DateAdd	First	LoadData	PlainText	Second	TimeValue	
ClearCollect	DateDiff	FirstN	Location	Power	Select	TimeZoneOffset	
Clock	DateTimeValue	ForAll	LookUp	Proper	Set	Today	
Coalesce	DateValue	GroupBy	Lower	Radians	ShowColumns	Trim	

Smart Designing Screen

- Use relative styling
  - Calculate control property values from other controls
  - Common to use properties like X, Y, Width, Height, Size, Fill, etc.



Working with Edit Forms and Data Cards

- Form acts as a container for data cards
  - Each form binds to a single record
  - Within a form, each data card binds to an underlying field
  - Each data card contains an encapsulated set of child controls

Using Patch Instead of an Edit Form

- Sometimes edit forms are not the best option
  - **Patch** function used to create and update records in data source

```
Patch(
    '[dbo].[Orders]',
    Defaults('[dbo].[Orders]'),
    {
        CustomerId: galCustomers.Selected.CustomerId,
        OrderAmount: Sum(
            colShoppingCart,
            Total
        ),
        OrderDate: Today()
    }
)
```

  - **Patch** function can be used with **ForAll** to insert multiple records

```
ForAll(
    colShoppingCart,
    Patch(
        '[dbo].[OrderDetails]',
        Defaults('[dbo].[OrderDetails]'),
        {
            OrderId: First(colLastOrder).OrderId,
            ProductId: ProductId,
            Quantity: Quantity,
            Total: Total
        }
    )
)
```

State Variables

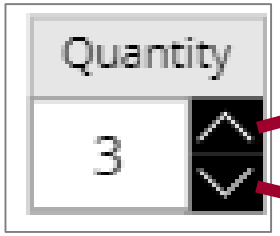
- Collections
  - Created as tables at app scope
  - Managed using Collect, Clear and ClearCollect
  - Can be stored to local device using SaveData & LoadData
- Context variables
  - Created as primitive, record or table at screen scope
  - Managed using UpdateContext and Navigate
- Global variables
  - Created as primitive, record or table at app scope
  - Created and managed using Set function

Updatable Collection Columns

- Often helpful to add updatable column to collections

```
ClearCollect(colProductCatalog, AddColumns('[dbo].[Products]',"Quantity", 1));
```

- Columns updated using calls to Patch




```
Patch(colProductCatalog, galProducts.Selected, { Quantity: (ThisItem.Quantity + 1) } )
```

```
Patch(colProductCatalog, galProducts.Selected, { Quantity: (ThisItem.Quantity - 1) } )
```

Capturing Return Value from Patch

```
ClearCollect(
    colLastOrder,
    Patch(
        '[dbo].[Orders]',
        Defaults('[dbo].[Orders]'),
        {
            CustomerId: galCustomers.Selected.CustomerId,
            OrderAmount: Sum(
                colShoppingCart,
                Total
            ),
            OrderDate: Today()
        }
    )
);
ClearCollect(
    colLastOrderDetails,
    ForAll(
        colShoppingCart,
        Patch(
            '[dbo].[OrderDetails]',
            Defaults('[dbo].[OrderDetails]'),
            {
                OrderId: First(colLastOrder).OrderId,
                ProductId: ProductId,
                Quantity: Quantity,
                Total: Total
            }
        )
    )
);
```

CustomerId	OrderAmount	OrderDate	OrderId
10	44.85	4/3/2019	14

colLastOrder

Id	OrderId	ProductId	Quantity	Total
36	14	1	1	14.95
37	14	4	2	19.9
38	14	6	10	10

colLastOrderDetails

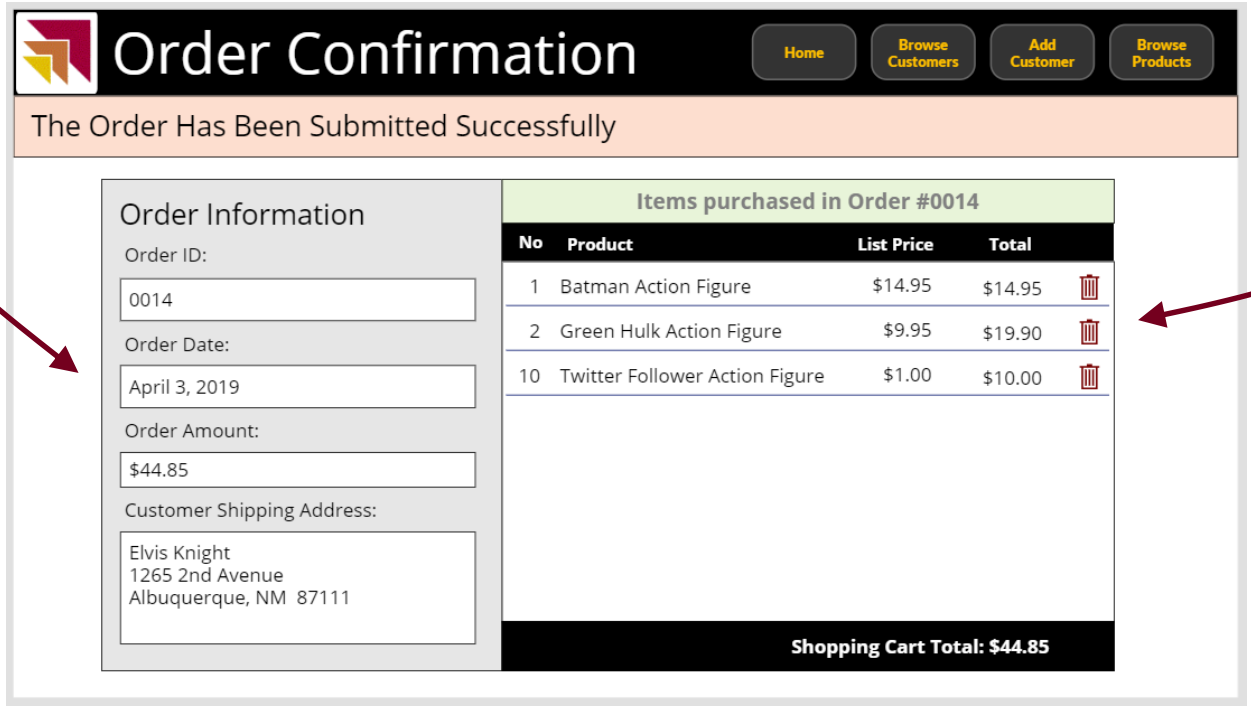
Populating UI from Cached State

CustomerId	OrderAmount	OrderDate	OrderId
10	44.85	4/3/2019	14

colLastOrder

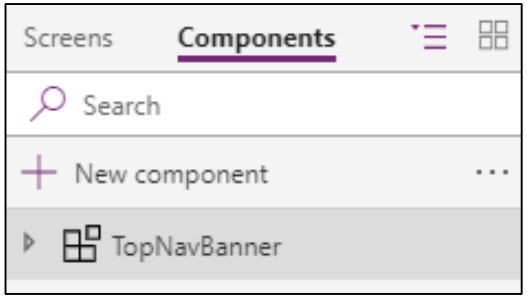
Id	OrderId	ProductId	Quantity	Total
36	14	1	1	14.95
37	14	4	2	19.9
38	14	6	10	10

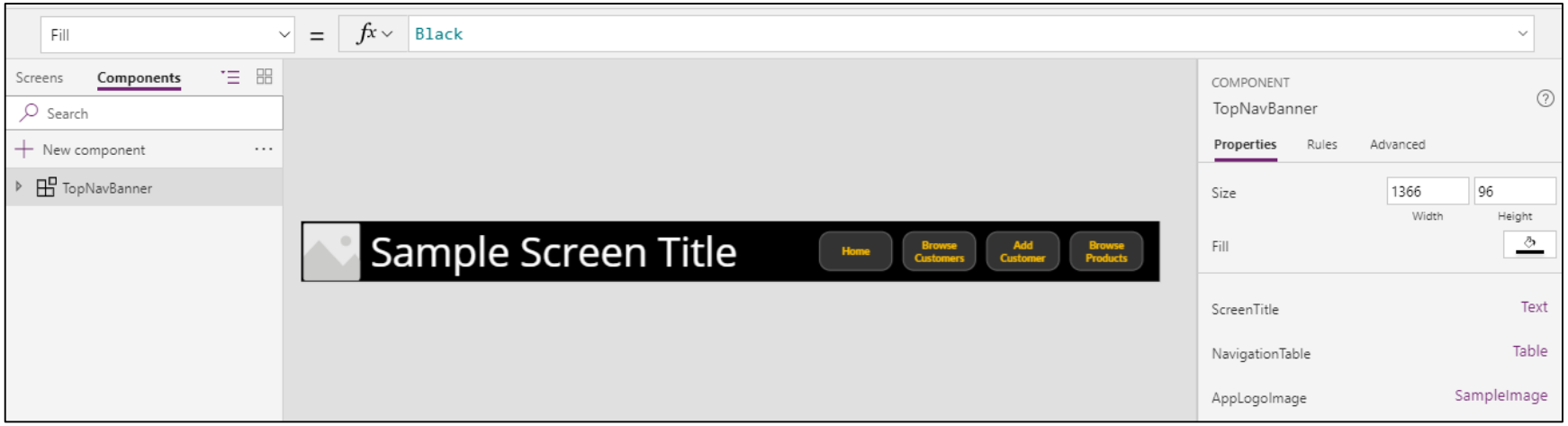
colLastOrderDetails



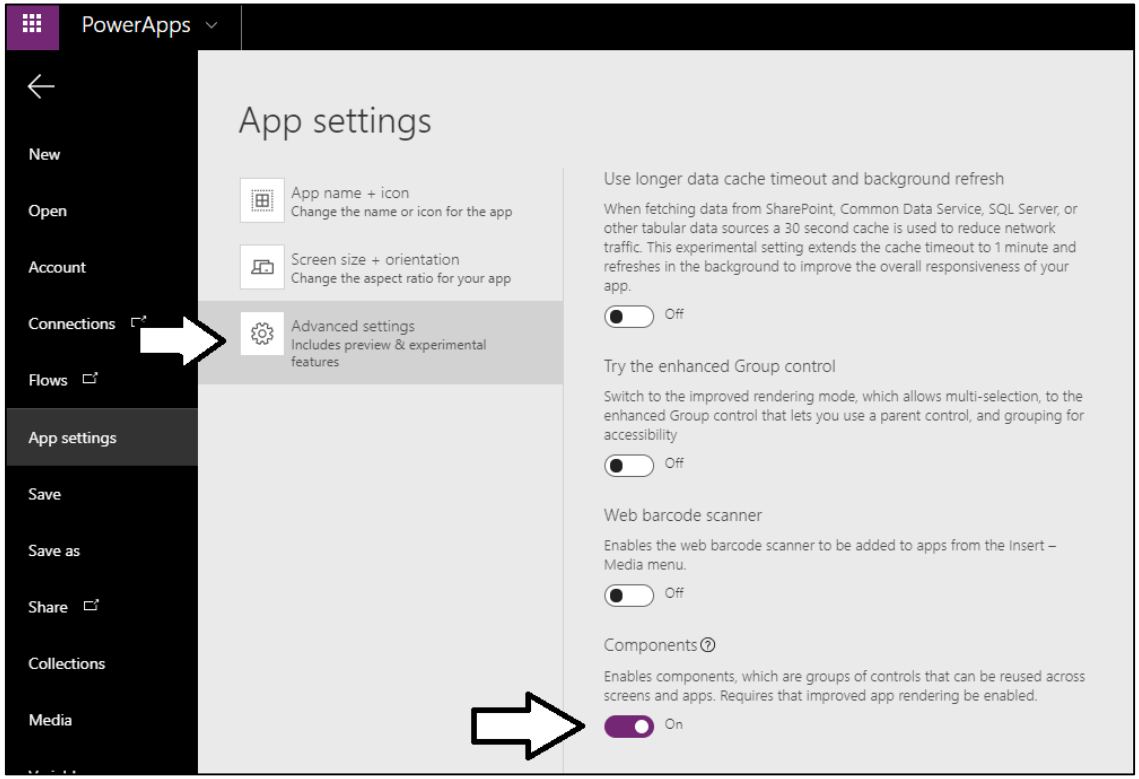
Designing Components

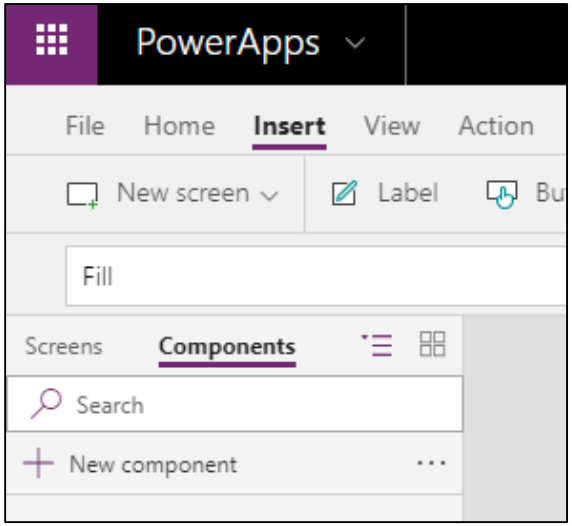
- Steps to using components
  - Create new component
  - Add component properties
  - Implement component UI and behavior
  - Add component to screens in your canvas apps





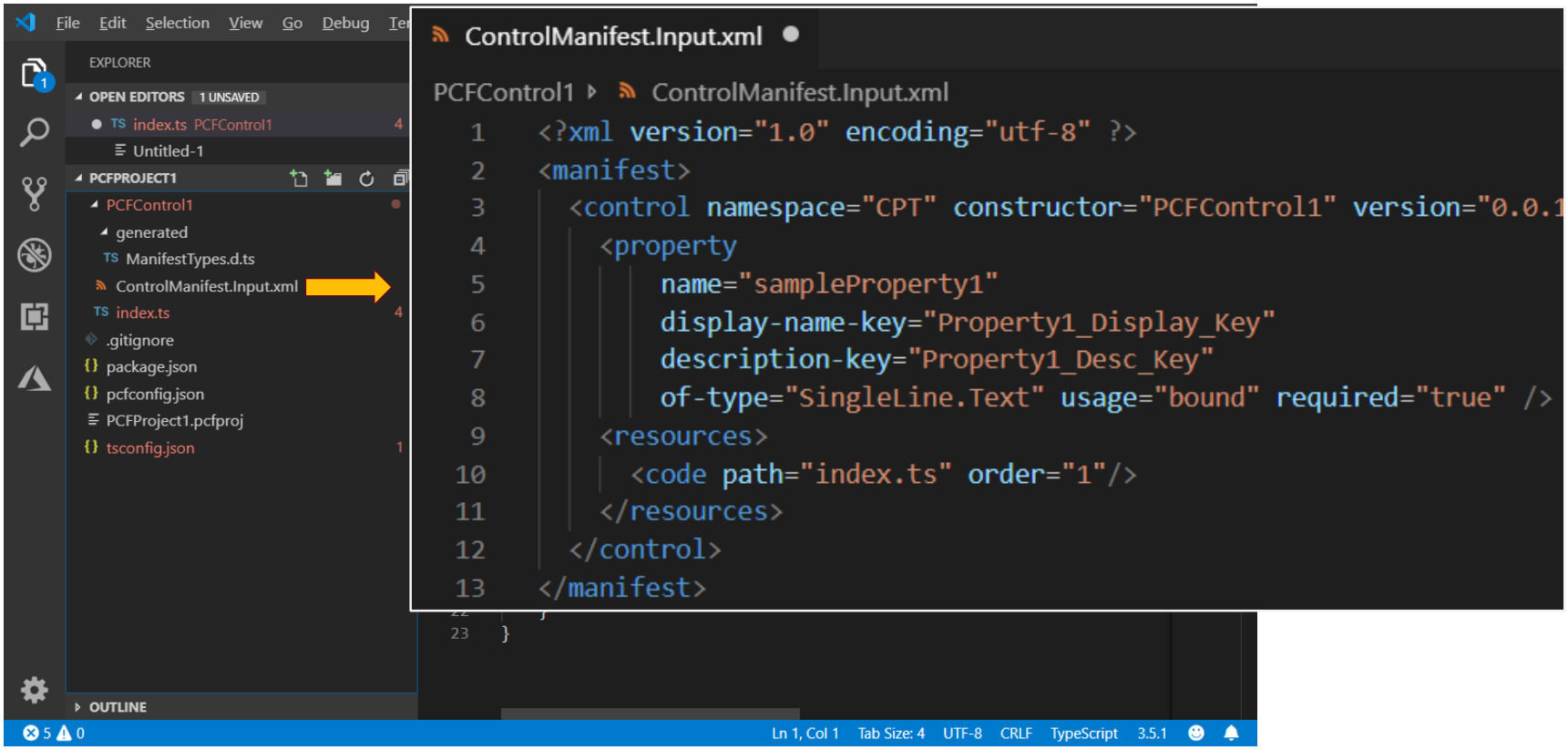
Enabling Components in a Canvas App





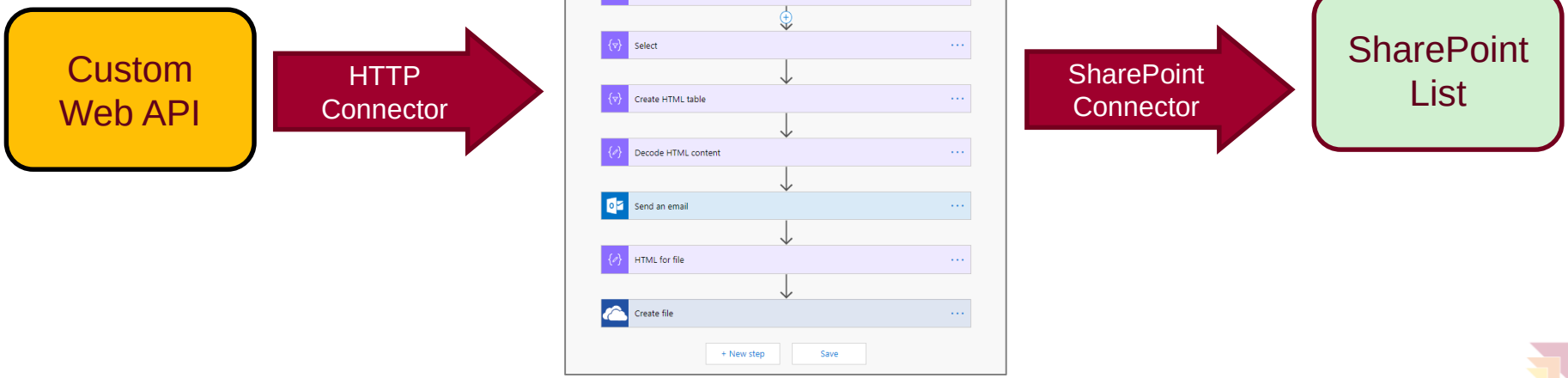
PowerApps Control Framework

- Controls developed using Node.js environment
  - You implement behavior using TypeScript



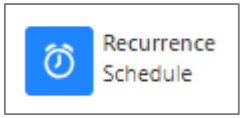
Building Blocks of Flow

- **Triggers** - events that start a flow
- **Actions** - tasks and operation executed by flow
- **Services** - sources and destinations for data
- **Connectors** - wrappers to communicate with service APIs

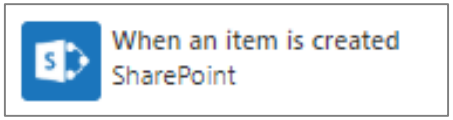


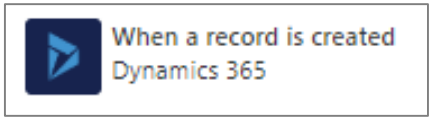
Flow Trigger Types

- Scheduled Flow Triggers
  - Runs periodically based on an interval

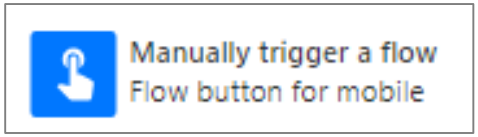


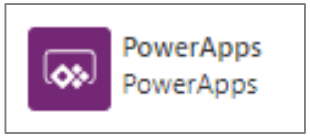
- Automated Flow Triggers
  - Runs when something happens





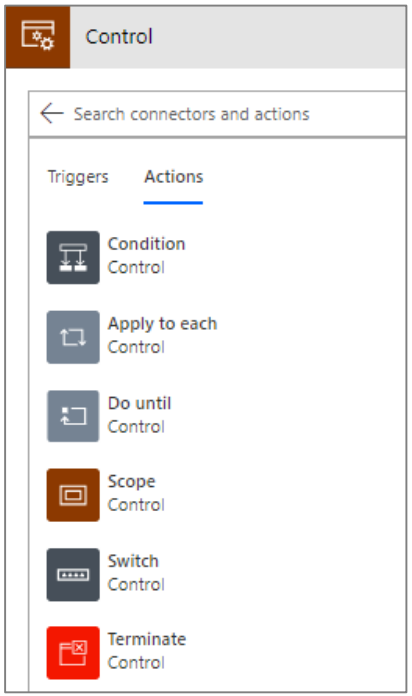
- On-demand Flow Triggers
  - Runs when a user clicks a button

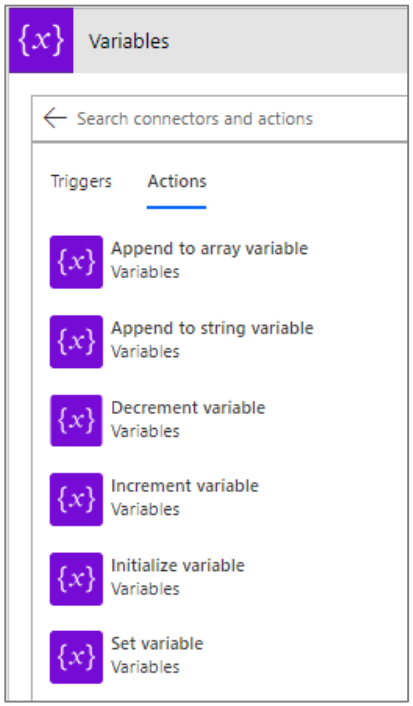


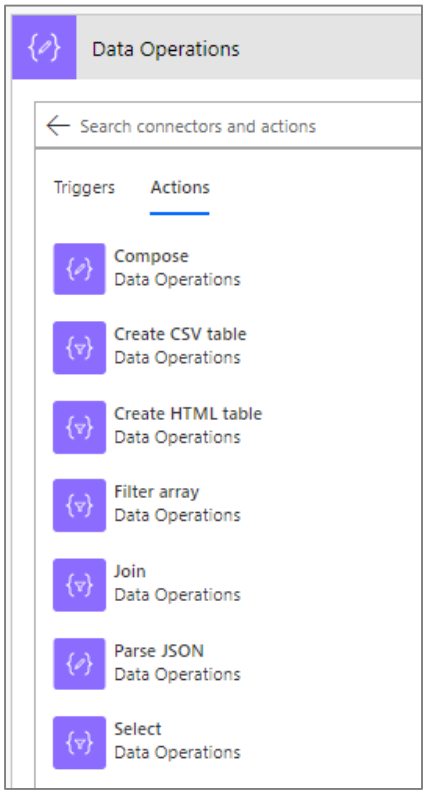


Core Action Categories

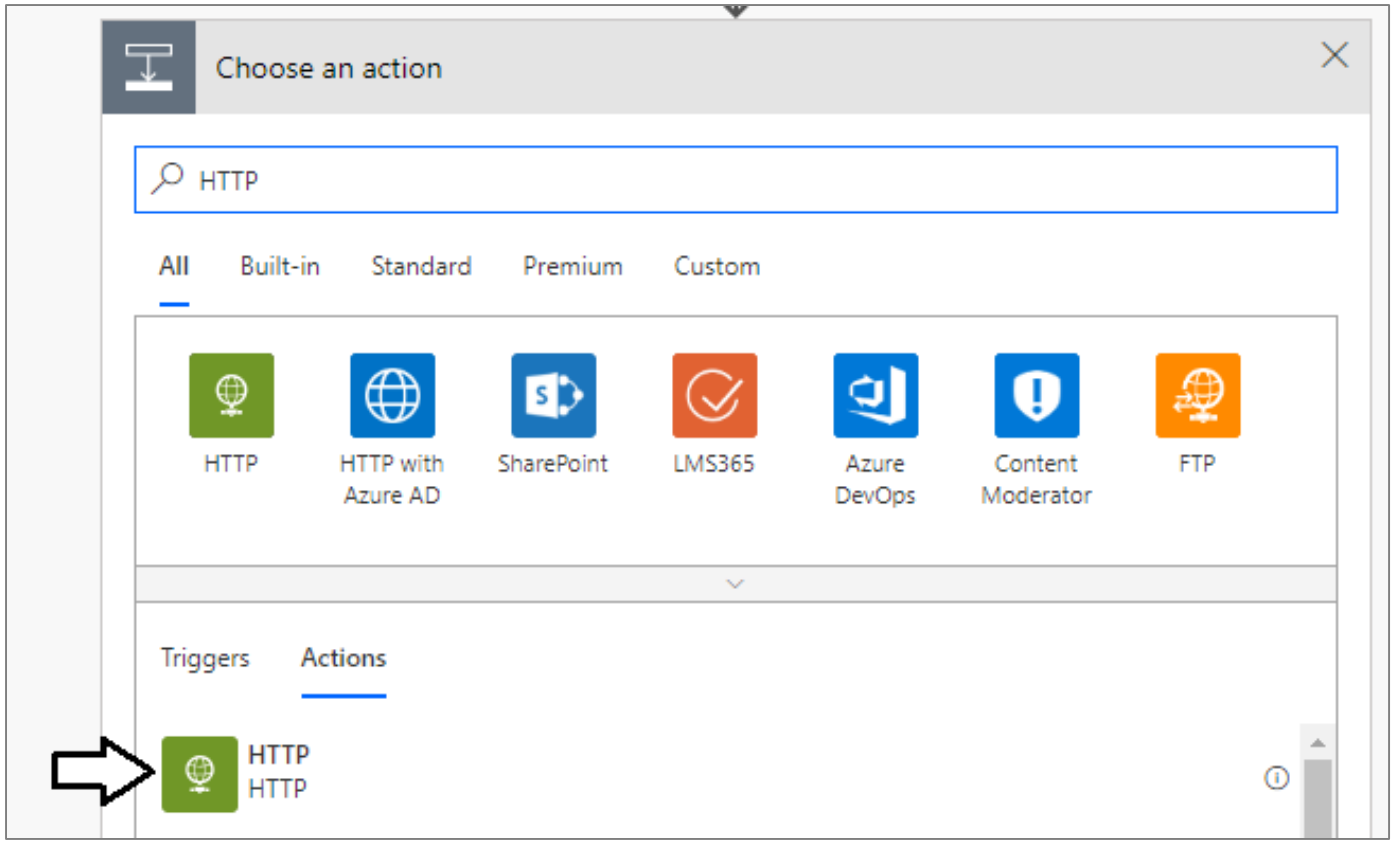
- **Control**: actions to provide control-of-flow
- **Variables**: actions to manage state within flow lifetime
- **Data operations**: action to process data & prepare content



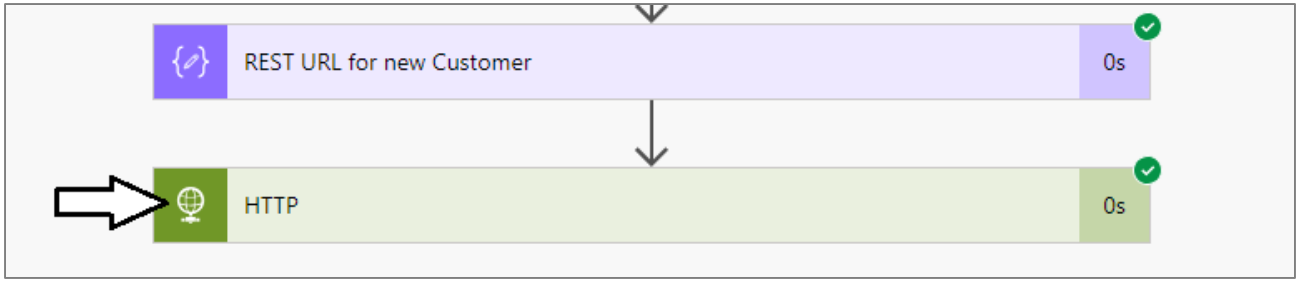


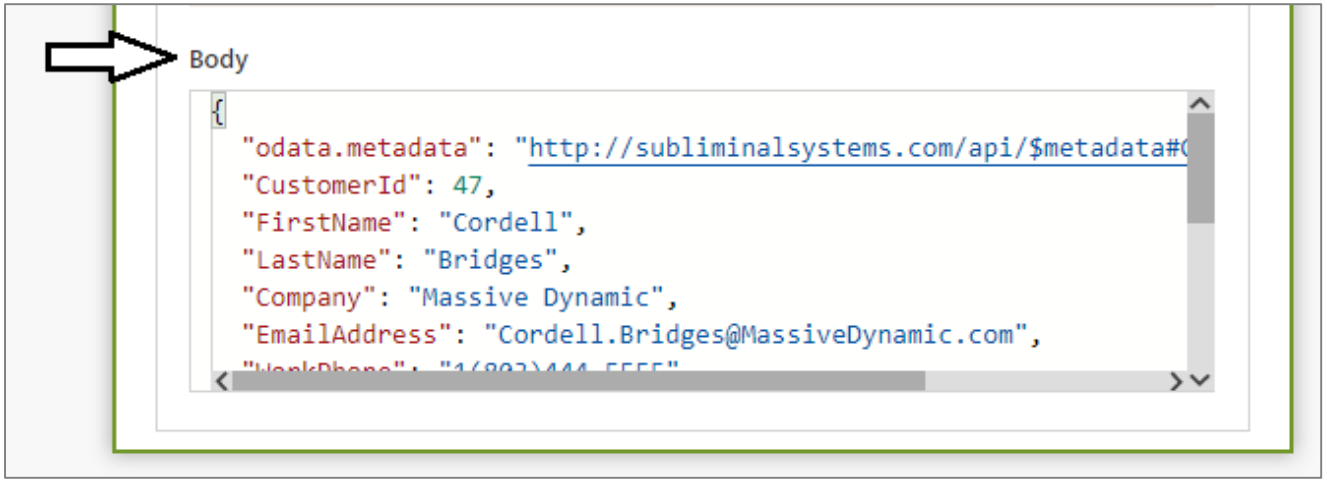


Using the HTTP Action

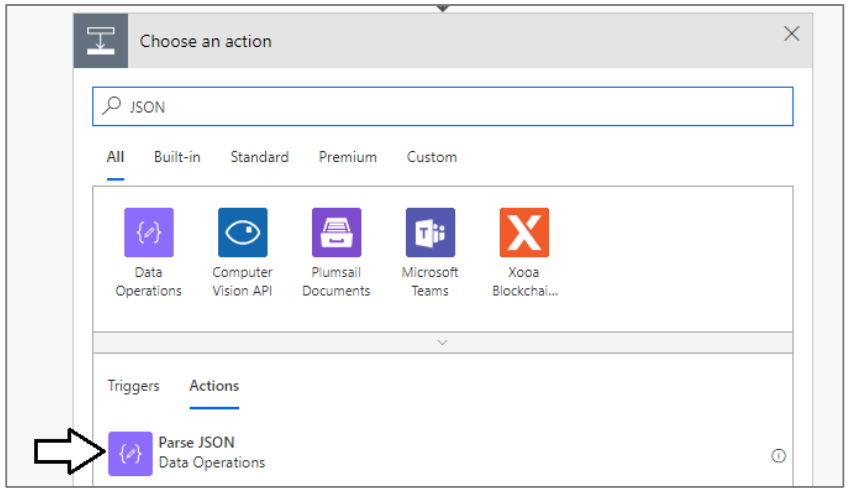


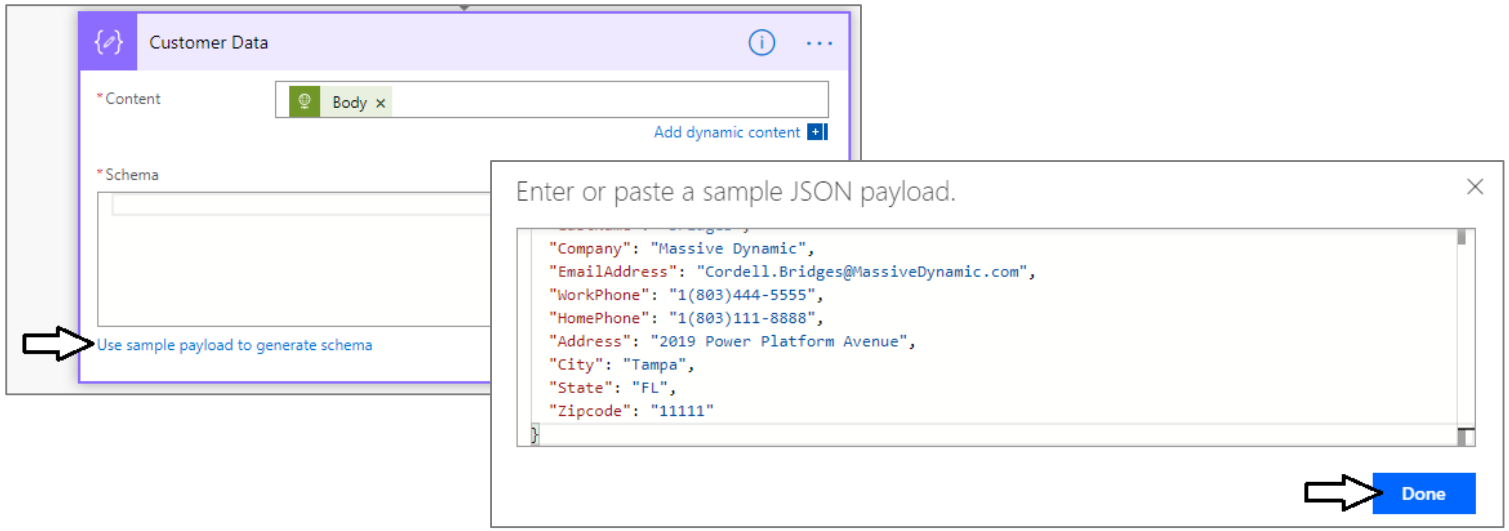
Inspecting the body of the HTTP Response



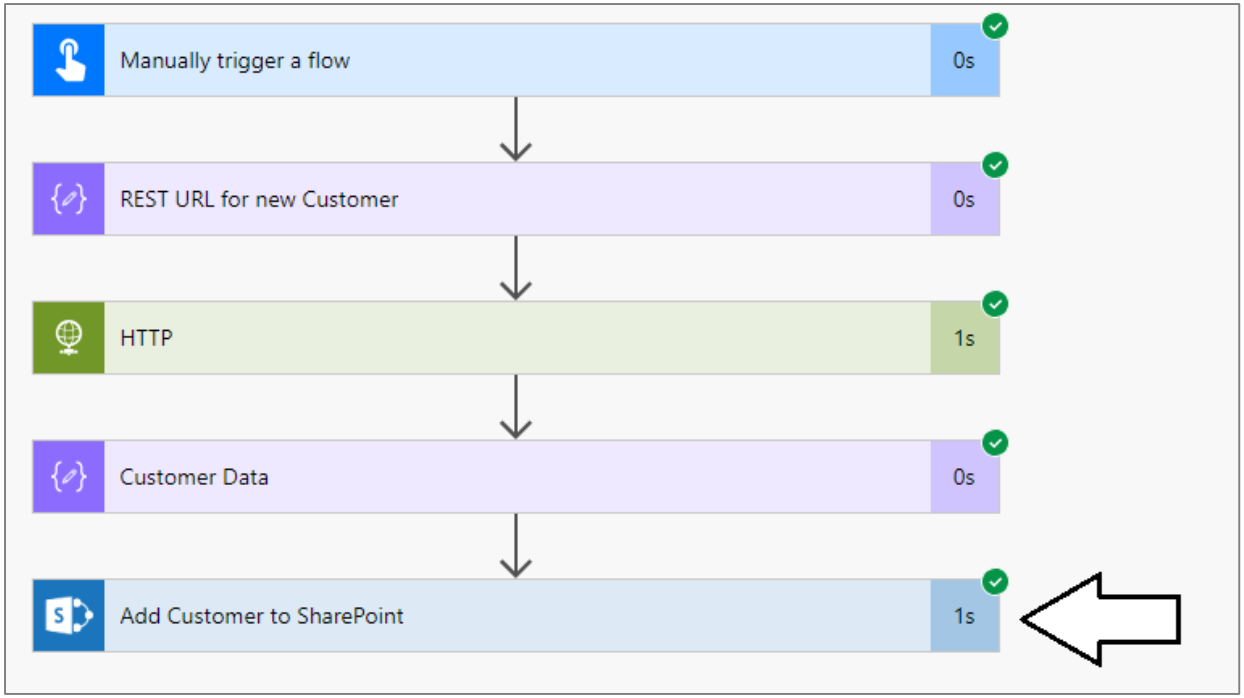


Parsing JSON



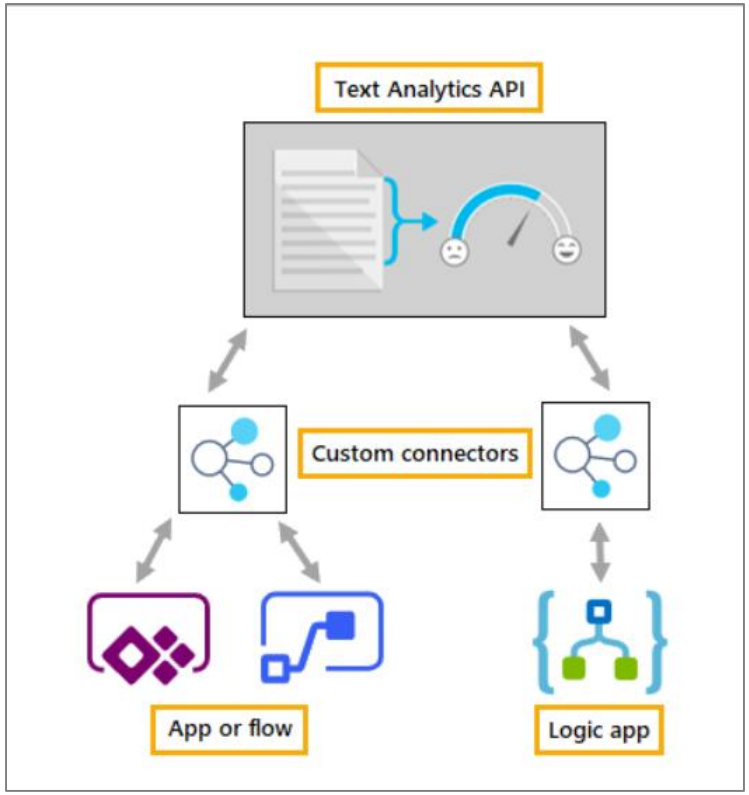


Adding a New Item to a SharePoint List

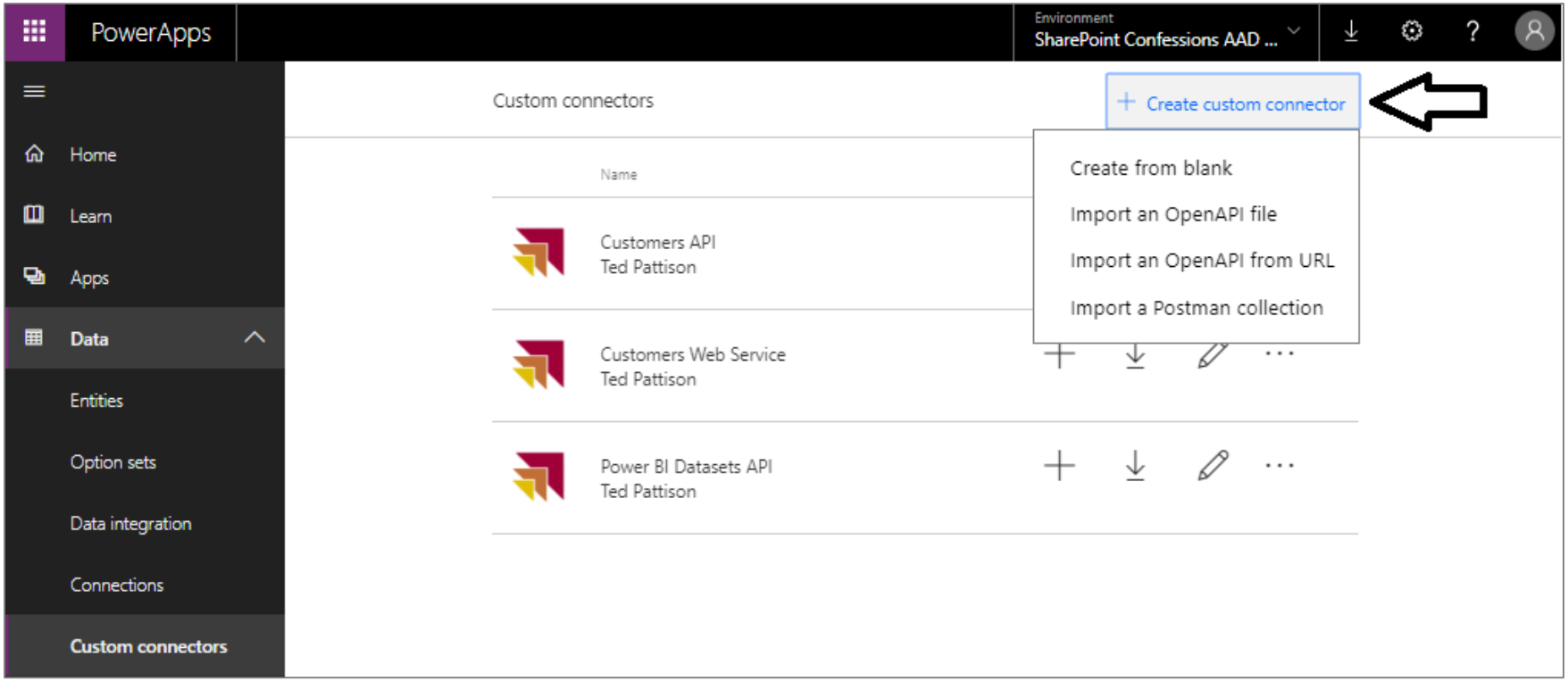


Standard Connectors vs Custom Connectors

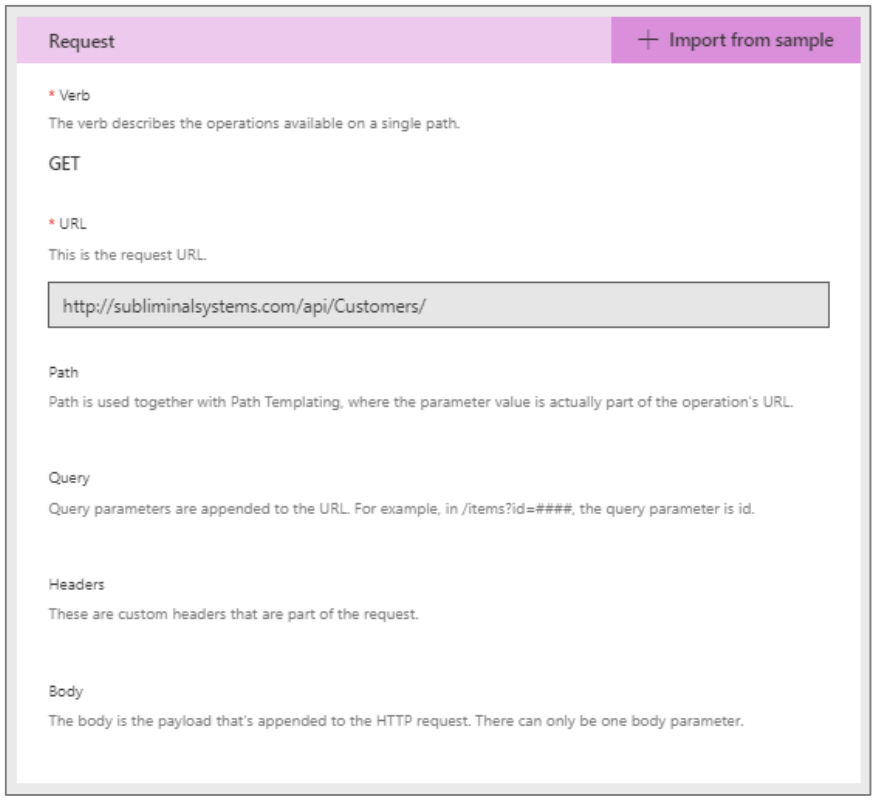
- PowerApps Supports two types of connectors
  - Standard connectors supplied out-of-box and vetted by Microsoft
  - Custom connectors created by organizations for their own use

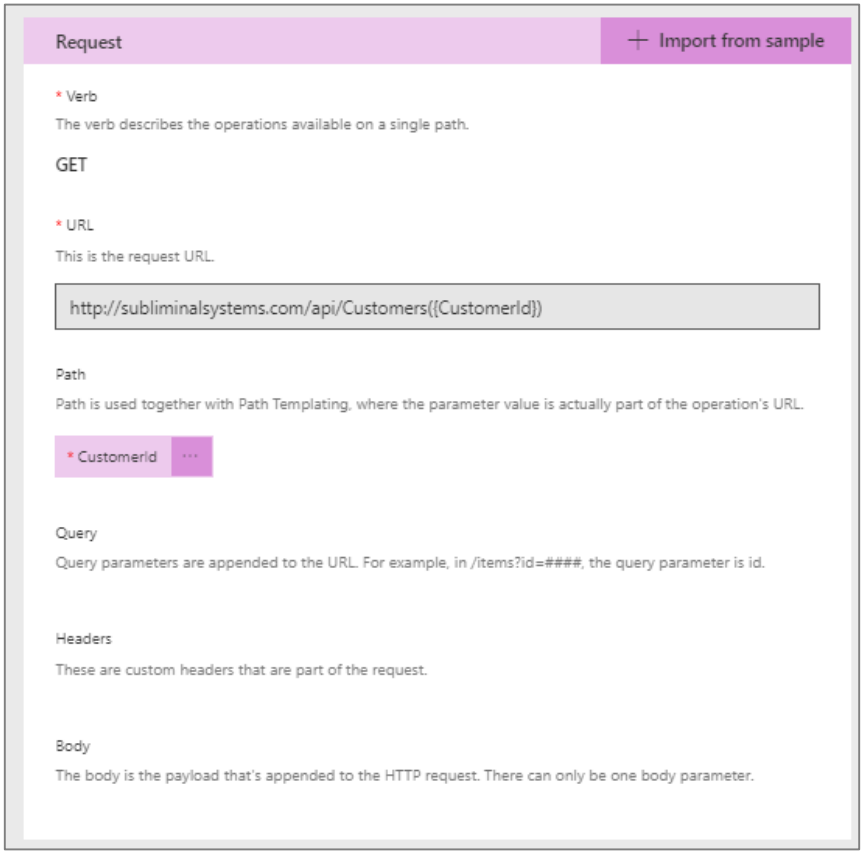


Creating a New Custom Connector

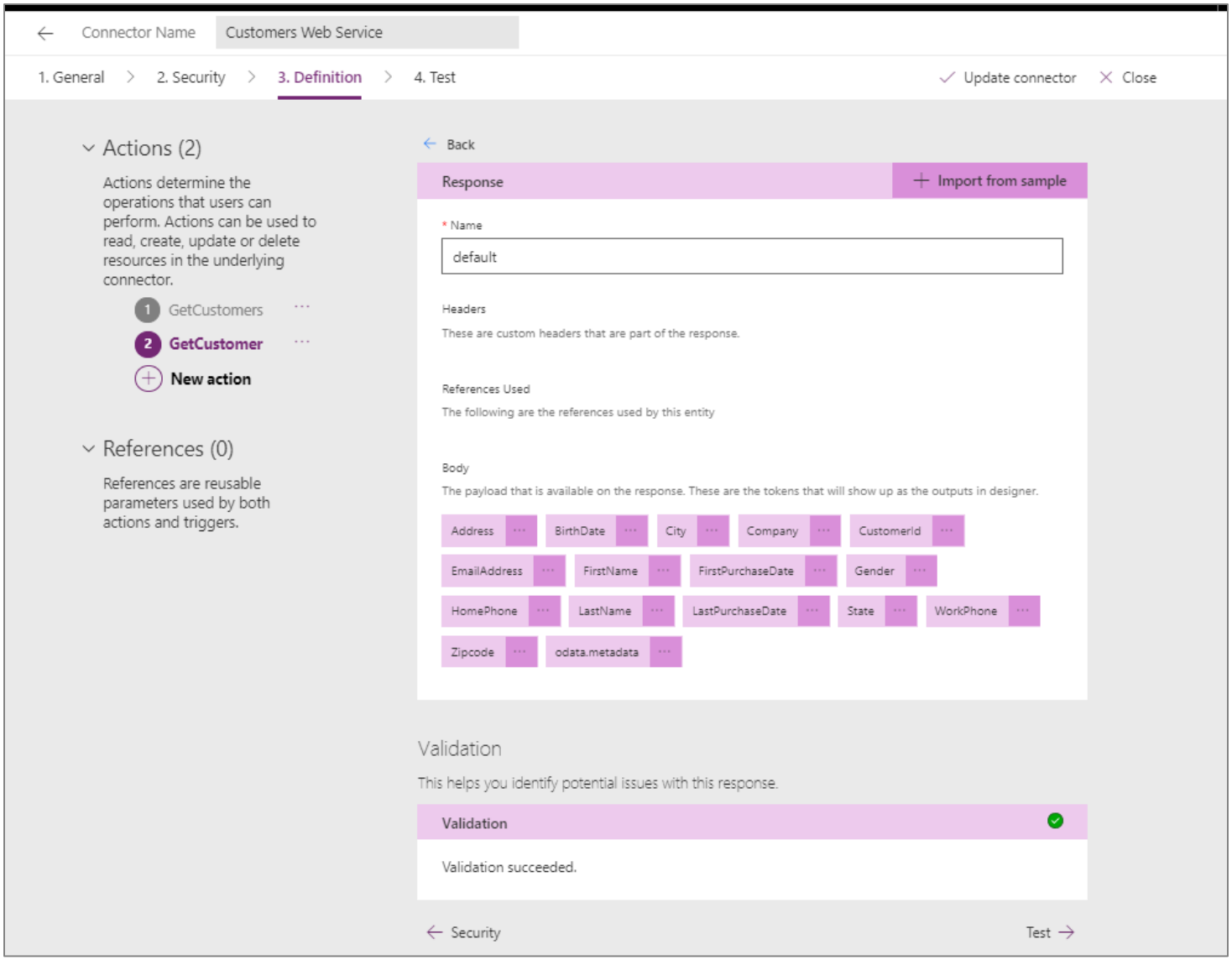


Defining Requests





Defining the Response



Our Newest PowerApps Training Course

- Advanced PowerApps and Flow training
  - Designed for students who have already attended Microsoft's App-in-a-day
  - Three days filled with plenty of hands-on lab exercises
  - Learn advanced builder skills & best practices
  - More info at https://CriticalPathTraining.com