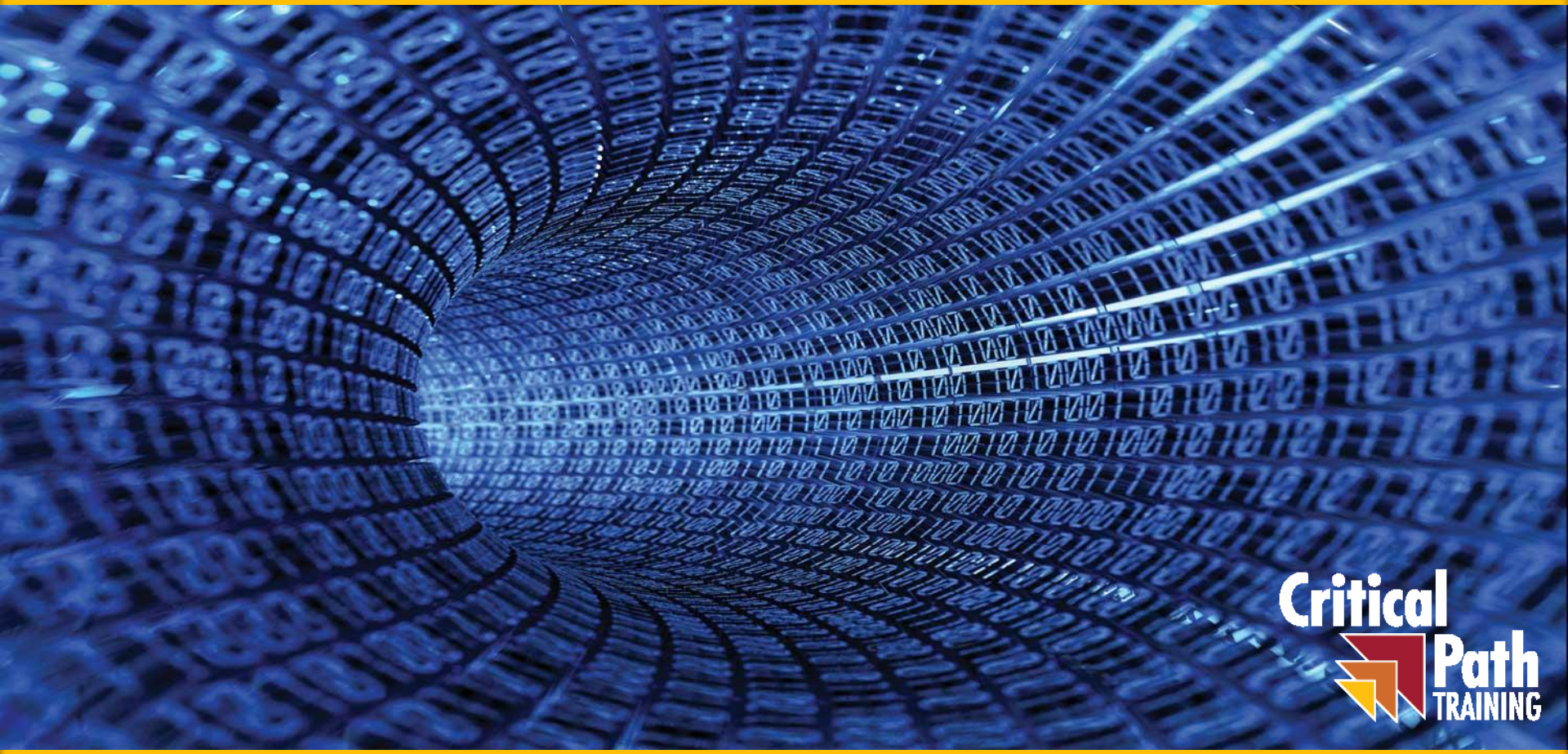
Building a Canvas App with a Shopping Cart



Agenda

- Caching State using Variables and Collections
- Using a Collection to Track Shopping Cart Data
- Using Patch Instead of an Edit Form
- Writing Shopping Cart Data to Back to SharePoint
- Designing Reusable Components

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

Code Naming Conventions

- Consistent naming conventions for variables & collections
  - Names should indicate type, purpose and scope
  - Collection names start with **col** such as **colProductCatalog**
  - Global variable names start with **gbl** such as **gblCustomApiUrl**
  - Context variables start with **loc** such as **locCustomerFilter**

Designing Canvas Apps using Collections

- Browse Products Screen allows user to build collection

Browse Products Screen

colShoppingCart (local collection)

No	Product	ListPrice	Total
1	Product 1	2.95	2.95
2	Product 2	4.95	9.90

- Submit Order Screen allows user to save to SharePoint

colShoppingCart (local collection)

No	Product	ListPrice	Total
1	Product 1	2.95	2.95
2	Product 2	4.95	9.90

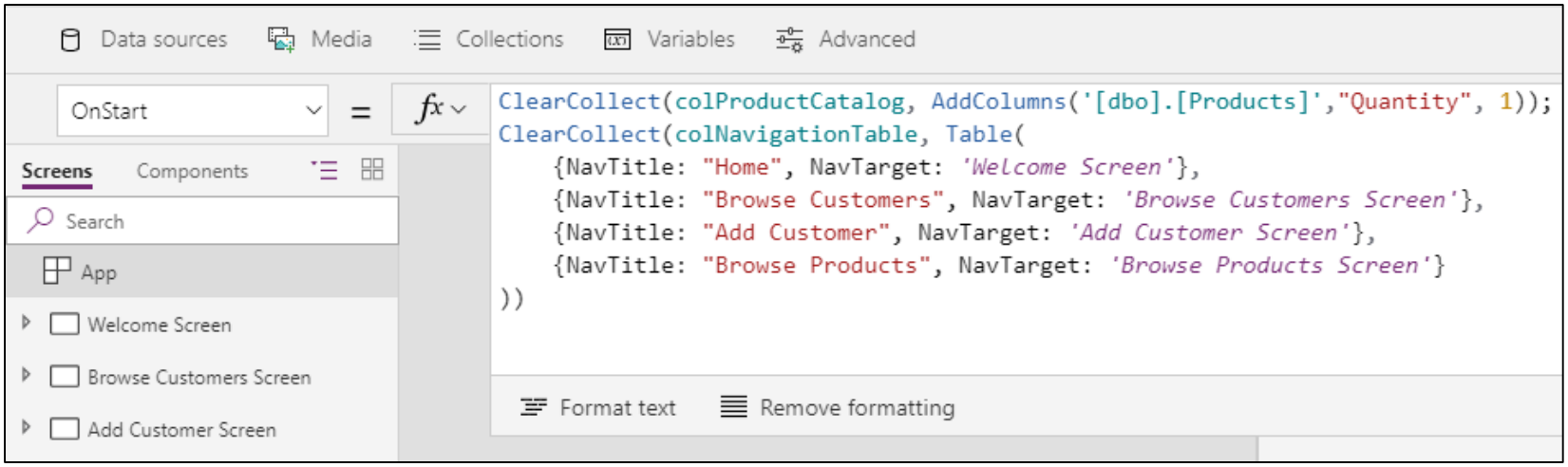
Submit Order Screen

OrderDetails (SharePoint List)

No	Product	ListPrice	Total
1	Product 1	2.95	2.95
2	Product 2	4.95	9.90

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



DEMO

Using PowerApps to create an App from a Modern SharePoint List

Agenda

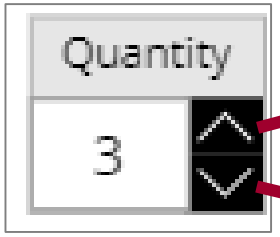
- ✓ Caching State using Variables and Collections
- ➢ Using a Collection to Track Shopping Cart Data
- Using Patch Instead of an Edit Form
- Writing Shopping Cart Data to Back to SharePoint
- Designing Reusable Components

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) } )
```

Agenda

- ✓ Caching State using Variables and Collections
- ✓ Using a Collection to Track Shopping Cart Data
- ➢ Using Patch Instead of an Edit Form
- Writing Shopping Cart Data to Back to SharePoint
- Designing Reusable Components

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
        }
    )
)
```

Agenda

- ✓ Caching State using Variables and Collections
- ✓ Using a Collection to Track Shopping Cart Data
- ✓ Using Patch Instead of an Edit Form
- ➢ Writing Shopping Cart Data to Back to SharePoint
- • Designing Reusable Components

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

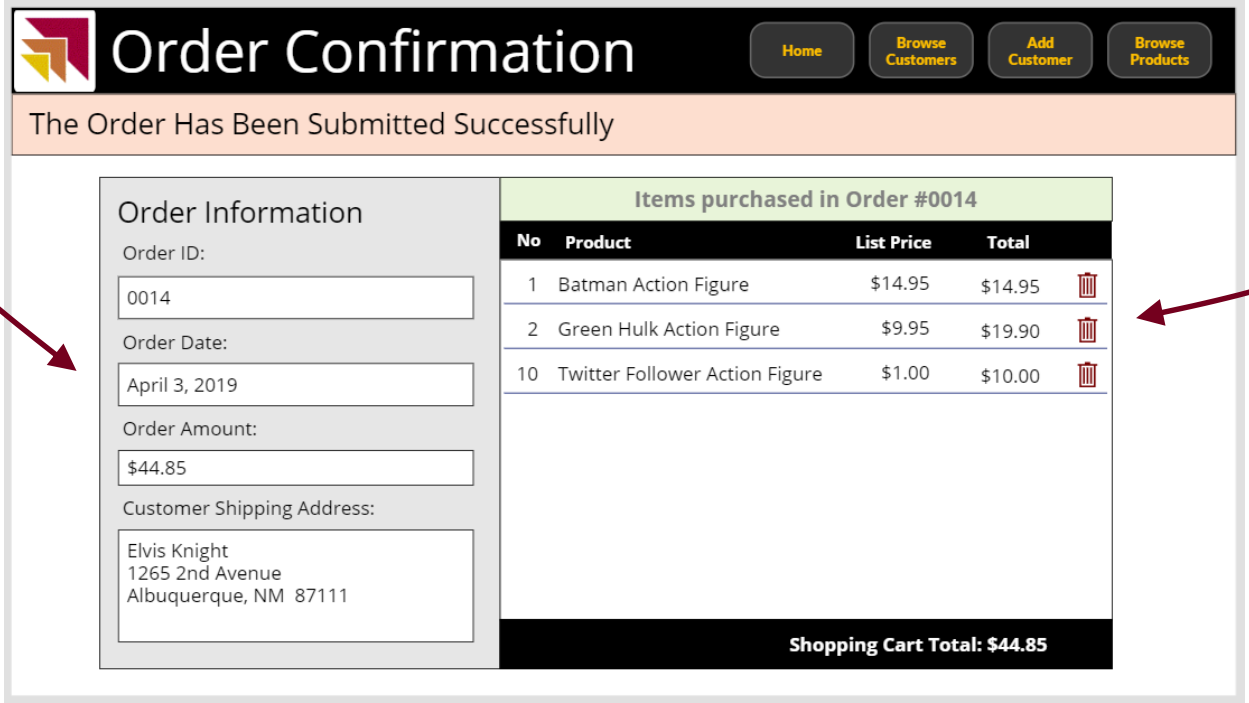
Populating UI from Cached State

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

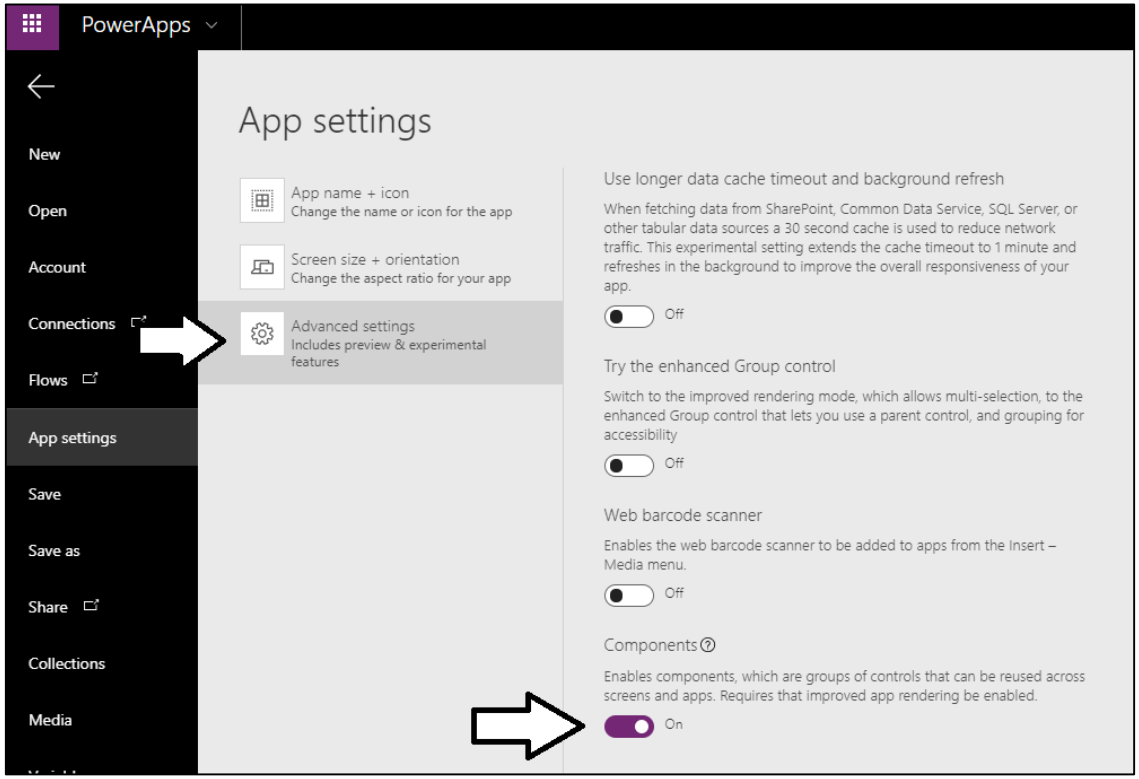
colLastOrderDetails

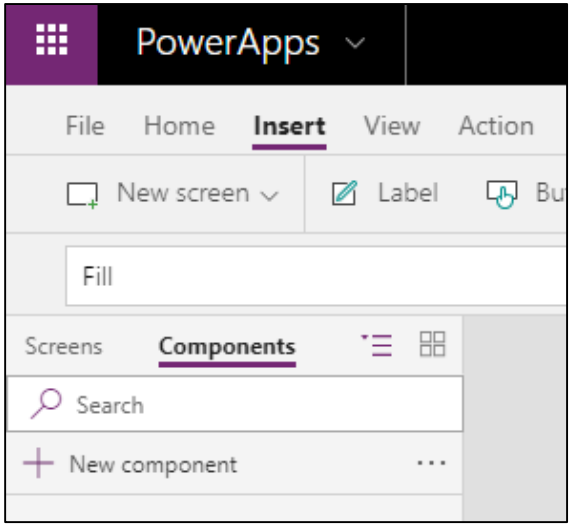


Agenda

- ✓ Caching State using Variables and Collections
- ✓ Using a Collection to Track Shopping Cart Data
- ✓ Writing Shopping Cart Data to Back to SharePoint
- ➢ Designing Reusable Components

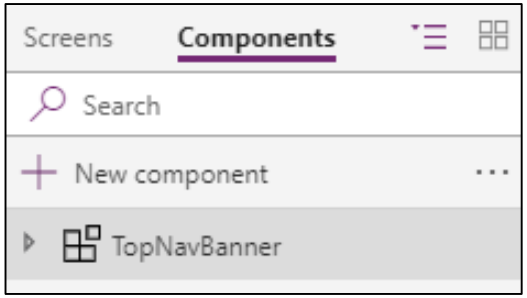
Enabling Components in a Canvas App

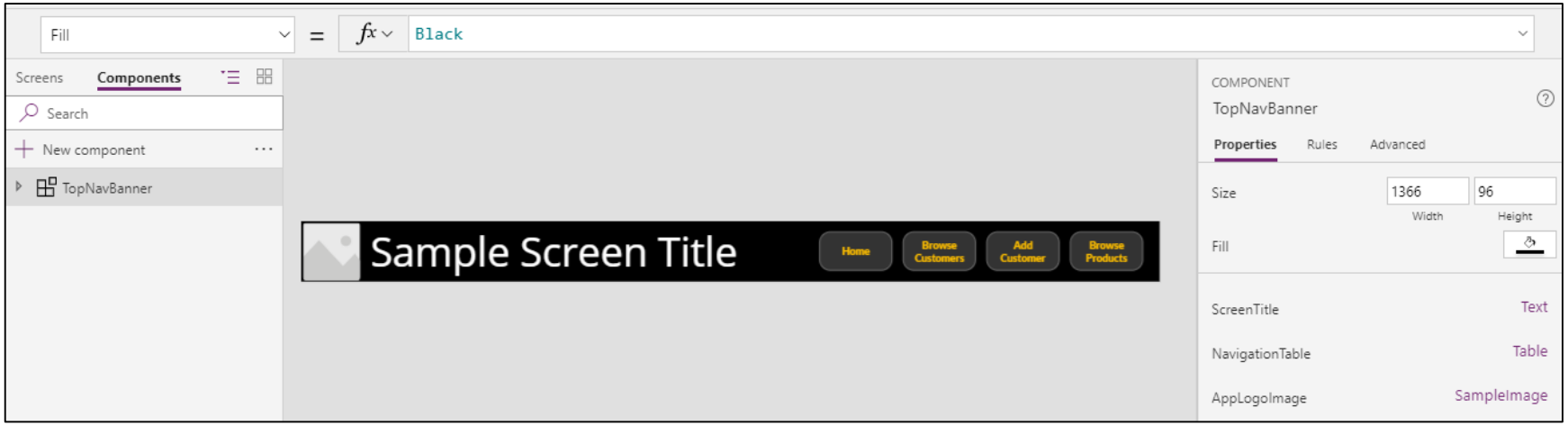




Designing Components

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





Summary

- ✓ Caching State using Variables and Collections
- ✓ Using a Collection to Track Shopping Cart Data
- ✓ Writing Shopping Cart Data to Back to SharePoint
- ✓ Designing Reusable Components