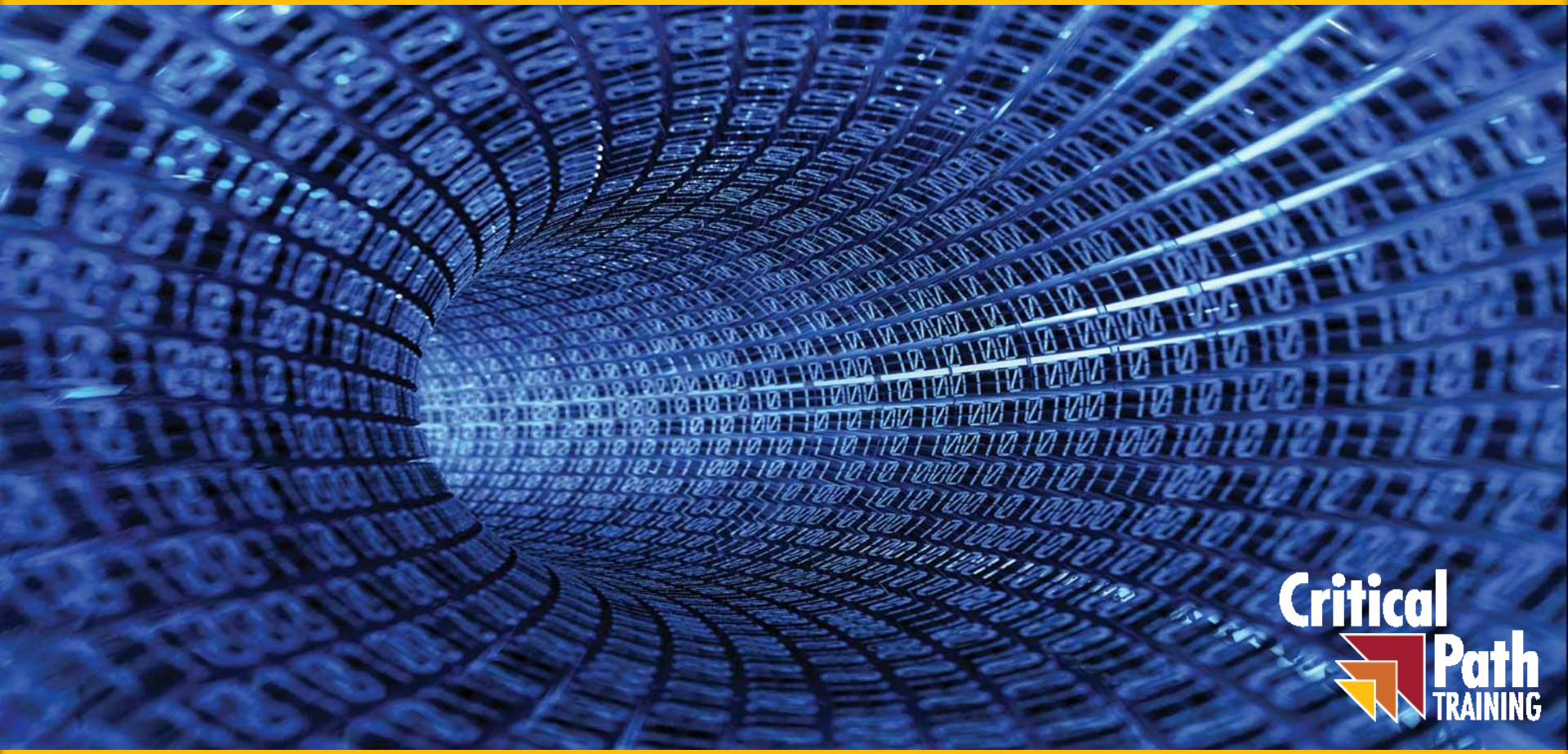
Developing and Distributing Custom Visuals



Agenda

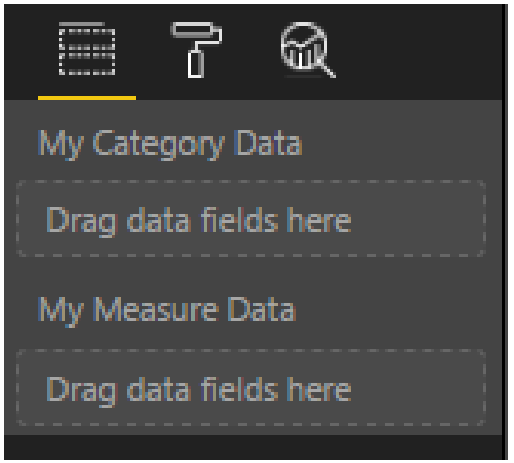
- Defining Visual Capabilities and Data Mappings
- Creating Visuals Bound to Categorical Data
- Extending a Visual with Custom Properties
- Packaging and Deploying Custom Visuals

Visual Capabilities

capabilities.json*

```
{
  "dataRoles": ...,
  "dataViewMappings": ...,
  "objects": ...
}
```

```
{
  "dataRoles": [
    {
      "displayName": "My Category Data",
      "name": "category",
      "kind": "Grouping"
    },
    {
      "displayName": "My Measure Data",
      "name": "measure",
      "kind": "Measure"
    }
  ],
  "dataViewMappings": ...,
  "objects": ...
}
```



Data Mappings

```
{
  "dataRoles": ...,
  "dataViewMappings": [
    {
      "conditions": [
        {
          "category": { "max": 1 },
          "measure": { "max": 1 }
        }
      ],
      "categorical": {
        "categories": {
          "for": { "in": "category" },
          "dataReductionAlgorithm": { "top": {} }
        },
        "values": {
          "select": [ { "bind": { "to": "measure" } } ] }
      }
    }
  ],
  "objects": ...
}
```

Binding Visuals to Categorical Data

```
public update(options: VisualUpdateOptions) {

  // ensure dataview contains categories and measurable values
  var categorical = options.dataViews[0].categorical;
  if (typeof categorical.categories === "undefined" ||
      typeof categorical.values === "undefined") {
    // remove all existing SVG elements
    this.svgGroupMain.empty();
    return;
  }

  // get categorical data from visual data view
  this.dataview = options.dataViews[0];

  // convert categorical data into specialized data structure for data binding
  var visualData: CategoryItem[] = Visual.converter(this.dataview.categorical);
```

Creating a Converter Function

```
public static converter(categoricalData: DataViewCategorical): CategoryItem[] {

  var visualData: CategoryItem[] = [];

  var categories: PrimitiveValue[] = categoricalData.categories[0].values;
  var categoryValues: PrimitiveValue[] = categoricalData.values[0].values;

  for (var i = 0; i < categoryValues.length; i++) {

    var category: string = <string>categories[i];
    var categoryValue: number = <number>categoryValues[i];

    visualData.push({
      Category: category,
      Value: categoryValue
    });
  }

  visualData.sort( (cat1, cat2) => { return cat2.Value - cat1.Value; })

  return visualData;
}
```

Extending Visuals with Custom Properties

capabilities.json

```
{
  "dataRoles": ...,
  "dataViewMappings": ...,
  "objects": {
    "colorSelector": {
      "displayName": "Bar Chart Colors",
      "properties": {
        "fill": {
          "displayName": "Color",
          "type": { "fill": { "solid": { "color": true } } }
        }
      }
    }
  }
}
```

Packaging and Deploying Custom Visuals