MODULE 14

STYLES

MODULE TOPICS

Style Basics
Creating a Style
Attaching Event Handlers
Style Inheritance
Typed Styles
Property-Based Triggers
Event-Based Triggers

STYLE BASICS

- One of the most common uses of resources is for styles
- A style is a collection of property values that can be applied to an element
- Plays a similar role to CSS in HTML
  - As with CSS, WPF styles can work automatically, target specific element types, and cascade through the element tree
- Supports triggers to change the style of a control when another property is changed

CREATING A STYLE

```
<Style x:Key="TimesNewSunriseStyle">
    <Setter Property="Control.FontFamily" Value="Times New Roman"/>
    <Setter Property="Control.FontSize" Value="25"/>
    <Setter Property="Control.FontWeight" Value="Bold"/>
    <Setter Property="Control.Background" Value="{StaticResource SunriseBrush}"
</Style>
```

APPLYING A STYLE

```
Style="{StaticResource TimesNewSunriseStyle}"
```

ATTACHING EVENT HANDLERS

- You can also create a collection of EventSetter objects that wire up events to specific handlers

```
<Style x:Key="MouseOverHighlightStyle">
    <EventSetter Event="TextBlock.MouseEnter" Handler="element_MouseEnter"/>
    <EventSetter Event="TextBlock.MouseLeave" Handler="element_MouseLeave"/>
    <Setter Property="TextBlock.Padding" Value="5"/>
</Style>
```

STYLE INHERITANCE

- You might want to create a style that builds on another style
- Can be done using the BasedOn attribute

```
<Style x:Key="BigStyle">
    <Setter Property="Control.FontFamily" Value="Times New Roman" />
    <Setter Property="Control.FontSize" Value="18" />
    <Setter Property="Control.FontWeight" Value="Bold" />
</Style>

<Style x:Key="CoolBigStyle" BasedOn="{StaticResource BigStyle}">
    <Setter Property="Control.Foreground" Value="White" />
    <Setter Property="Control.Background" Value="DarkBlue" />
</Style>
```

APPLYING STYLES BY TYPE

- You can apply a style automatically to elements of a certain type
- Set the TargetType property to indicate the appropriate type

```
<Style TargetType="Button">
    <Setter Property="Control.FontFamily" Value="Times New Roman" />
    <Setter Property="Control.FontSize" Value="18" />
    <Setter Property="Control.FontWeight" Value="Bold" />
</Style>
```

- An element can opt-out by setting a null style

TRIGGERS

- Using triggers, you can automate simple style changes that would normally require event handling logic
- Triggers are linked to styles through the Style.Triggers collection
- Every style can have an unlimited number of triggers
- Each trigger is an instance of a class that derives from System.Windows.TriggerBase

TRIGGERS

```
<Style x:Key="BigFontButton">
    <Style.Setters>
        <Setter Property="Control.FontFamily" Value="Times New Roman" />
        <Setter Property="Control.FontSize" Value="18" />
    </Style.Setters>
    <Style.Triggers>
        <Trigger Property="Control.IsFocused" Value="True">
            <Setter Property="Control.Foreground" Value="DarkRed" />
        </Trigger>
    </Style.Triggers>
</Style>
```

MULTITRIGGER

- For a trigger that switches on only if several criteria are true, you can use a MultiTrigger

```
<Style x:Key="BigFontButton">
    <Style.Triggers>
        <MultiTrigger>
            <MultiTrigger.Conditions>
                <Condition Property="Control.IsFocused" Value="True">
                <Condition Property="Control.IsMouseOver" Value="True">
            </MultiTrigger.Conditions>
            <MultiTrigger.Setters>
                <Setter Property="Control.Foreground" Value="DarkRed"/>
            </MultiTrigger.Setters>
        </MultiTrigger>
    </Style.Triggers>
</Style>
```

WALKTHRU - STYLES

ANY QUESTIONS?