XAML for WPF Cheat Sheet

Data Binding

{Binding}	Binds the current DataContext
{Binding *propertyName*}	Binds the property of the current DataContext
{Binding Source={StaticResource *resName*}}"	Binds to a staticresource such as a string
{Binding ElementName= *elementName*, Path= *propertyName*}	Binds to the property of the given element
{Binding RelativeSource={RelativeSource AncestorType ={x:Type *typeName*}}, Path= *propertyName*}	Looks up the visual tree to find an ancestor of the given type. Specify AncestorLevel as an integer to set the level
{Binding RelativeSource={x:Static RelativeSource.Self}, Path=*propertyName*}	Binds to the current elements given property.
{TemplateBinding *propertyName}*	Binds to the template parents given property inside a ControlTemplate
<XmlDataProvider x:Key="*name*" Source=" *filePath*" />	Added to resources to allow xml data binding
{Binding Source={StaticResource *resName*}, XPath=x*pathValue*}	Binds to a an XML node using XPath
IsSynchronizedWithCurrentItem ="True"	Synchronizes elements bound to same data source

Resources

<*element Name*.Resources>	Holds resources accessible to anything under the element.
<system:String x:Key=" *resName*">*stringValue* </system:String>	Creates a string resource
<SolidColorBrush x:Key=" *resName*" Color="*colourValue*" />	Placed inside Resources tag, creates a resource with the value
StaticResource	Applied once, when needed
DynamicResource	Applied as the resources changes
Fill="{StaticResource *resName*}"	Sets fill to the resource
Fill="{DynamicResource {x:Static System.WindowBrush}}"	Sets the fill to the system resource WindowBrush

Brushes

SoldColorBrush	A solid colour brush
Linear Gradient Brush	Gradient brush using GradientStops
RadialGradientBrush	Gradient radiating out from a point
ImageBrush	Brush from an ImageSource
DrawingBrush	based on a given GeometryDrawing
VisualBrush	Brush using a visual elements image

Styles and Triggers

<Button Style="{StaticResource *styleName*}" />	Sets style on current button element
<Style Target="{x:Type *typeName*}">	Style applied to all elements of this type
<Style x:Key=" *styleName*">	Style applied to elements that set this style
<Style x:Key=" *styleName*" Target="{x:Type *typeName*}">	Style applied to elements that set this style and are type button
<Setter Property="*propertyName*" Value="*value*"/>	Setter sets a property of the current target type
<Style.Triggers>	Holds Triggers to set a style dependant on event
<Trigger Property=" *propertyName*" Value="*value*">	Triggered when dependency property is set
<DataTrigger Binding="{Binding Path=*propertyName*}" Value="*theValue*">	Triggered when bound value is set
<EventTrigger RoutedEvent="*eventName*">	Triggered when the given routed event is fired

Transforms

LayoutTransform	Holds transform or group, executed before layout, moves surrounding elements
RenderTransform	Holds transform or group executed before layout, create overlapping elements
TransformGroup	Used to hold multiple transforms
RotateTransform	Rotates the given element by setting Angle, CenterX and CenterY
TranslateTransform	Moves elements based on X and Y
SkewTransform	Skews the given element using AngleX AngleY, CenterX and CenterY
MatrixTransform	Complex transform based on matrix

Layout

Window	Primary container for windows applications
Page	Container with navigation support.
StackPanel	Horizontal or Vertical stacking of elements
Grid	Grid layout via Grid.Row and Grid.Column
Canvas	Arrange elements using X and Y co-ordinates
WrapPanel	Horizontal or Vertical wrapping of visual elements as space is available
DockPanel	Dock elements via DockPannel.Dock