TOOL DEVELOPMENT

WPF & XAML – RESOURCES & TEMPLATES (INTRO)

STATIC RESOURCES

INTRODUCTION TO RESOURCES & MARKUP EXTENTIONS

STATIC RESOURCES – WHY?

- The same set of colors is being reused for various items
- All titles should have the same style

STATIC RESOURCES: HOW?

```
<Grid>
    <Grid.Resources>
        <SolidColorBrush x:Key="BrightBrush" Color="#fdb900"/>
        <SolidColorBrush x:Key="MediumBrush" Color="#9b7007"/>
        <SolidColorBrush x:Key="DarkBrush" Color="#3f320f"/>
    </Grid.Resources>
```

- Add **resources** to a container
  - Resources will become available to all elements inside this container
  - Container = any element type that can hold other elements:
    Grid, Window, ListBox, StackPanel, Button,...
- Add a **key** to each resource
  - items in xaml refer to this key using StaticResource
  - `<Button Background="{StaticResource MediumBrush}" ... />`
  - markup extensions: { }

STATIC RESOURCES: TYPES

```
<Grid.Resources>
    <SolidColorBrush x:Key="BrightBrush" Color="#fdb900"/>
    <SolidColorBrush x:Key="MediumBrush" Color="#9b7007"/>
    <SolidColorBrush x:Key="DarkBrush" Color="#3f320f"/>

    <Int32 x:Key="TitleSize">21</Int32>
</Grid.Re
<!--<Soli                                                #1d1d1d"/>
```

The type 'Int32' was not found. Verify that you are not missing an assembly reference and that all referenced assemblies have been built.

Show potential fixes (Alt+Enter or Ctrl+.)

- SolidColorBrush is in the WPF namespace
- Int32 is in the mscorlib namespace
  - This namespace is not added by default

ADDING A NAMESPACE IN XAML

- Namespaces in code: `using namespace;`
- Namespaces in xaml:

Grid | Grid

```
<Window x:Class="L02_FirstUI.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Apex Legends" Height="700" Width="1400" MinWidth="1220" MinHeight="505">
    <Grid>
        <Grid.Resources>
            <SolidColorBrush x:Key="BrightBrush" Color="#fdb900"/>
            <SolidColorBrush x:Key="MediumBrush" Color="#9b7007"/>
            <SolidColorBrush x:Key="DarkBrush" Color="#3f320f"/>


        </Grid.Resources>

```

FYI: STYLES

- Styling an entire control at once

```
<Grid>
    <Grid.Resources>
        <SolidColorBrush x:Key="BrightBrush" Color="#fdb900"/>
        <SolidColorBrush x:Key="MediumBrush" Color="#9b7007"/>
        <SolidColorBrush x:Key="DarkBrush" Color="#3f320f"/>

        <Style TargetType="Button" x:Key="DetailButtonStyle">
            <Setter Property="FontSize" Value="21"/>
            <Setter Property="HorizontalContentAlignment" Value="Center"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Background" Value="{StaticResource MediumBrush}"/>
        </Style>

    </Grid.Resources>
```

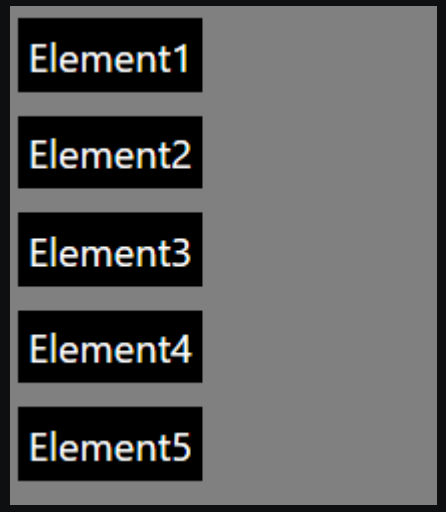
ITEM TEMPLATES

INTRODUCTION TO WORKING WITH TEMPLATES

custom listbox template

WORKING WITH DYNAMIC CONTENT

- What if you don't know in advance how many items you will need to display?




```
<ListBox Grid.Row="2" Background="Gray">
    <Label Content="Element1" Foreground="White" FontSize="20" Margin="4" Background="Black"></Label
    <Label Content="Element2" Foreground="White" FontSize="20" Margin="4" Background="Black"></Label
    <Label Content="Element3" Foreground="White" FontSize="20" Margin="4" Background="Black"></Label
    <Label Content="Element4" Foreground="White" FontSize="20" Margin="4" Background="Black"></Label
    <Label Content="Element5" Foreground="White" FontSize="20" Margin="4" Background="Black"></Label
</ListBox>
```

custom listbox template

FIRST THINGS FIRST: **LIST<T>** TYPE

- Collection type
  - does not have a fixed number of elements
  - useful to add / insert / remove elements at runtime
  - Equivalent to c++ vector

```
//Declare + initialize
List<string> myList = new List<string>();

//add elements
myList.Add("Hello");
myList.Add("!");

//insert 'world' between 'Hello' and '!'
myList.Insert(1, "world");
```

```
//fill right away
List<string> testList = new List<string>
            { "this", "is", "not", "a", "test" };

//remove item
testList.Remove("not");
//remove by Index
testList.RemoveAt(2);

//iterate over items in List<string>
foreach (string item in testList)
{
    //..
}
```

custom listbox template

FILL A LISTBOX: **ITEMSSOURCE**

List<string>

- MainWindow.xaml.cs:

```
List<string> nameList = new List<string>
            { "Aquaman", "Wonder Woman", "Cyborg", "Flash" };

lstWords.ItemsSource = nameList;
```

- MainWindow.xaml:

```
<ListBox x:Name="lstWords" Background="Black" Foreground="White"/>
```

- Result:

Aquaman
Wonder Woman
Cyborg
Flash

custom listbox template

FILL A LISTBOX: ITEMSSOURCE

List<Hero>

- MainWindow.xaml.cs:

```
List<Hero> heroList = new List<Hero>
{
    new Hero(){Character="Aquaman", RealName="Arthur Curry"},
    new Hero(){Character="Wonder Woman", RealName="Diana Prince"},
    new Hero(){Character="Cyborg", RealName="Victor Stone"},
    new Hero(){Character="Flash", RealName="Barry Allen"},
};

lstHeroes.ItemsSource = heroList;
```

- MainWindow.xaml:

```
<ListBox x:Name="lstHeroes" Background="Black" Foreground="White"/>
```

- Result:

~ AQUAMAN ~ Arthur Curry
~ WONDER WOMAN ~ Diana Prince
~ CYBORG ~ Victor Stone
~ FLASH ~ Barry Allen

```
public override string ToString()
{
    return $"~ {Character.ToUpper()} ~ \t{RealName}";
}
```

custom listbox template

LISTBOX: ITEMTEPLATE

```
0 references
public class Hero
{
    1 reference
    public string Character { get; set; }
    0 references
    public string RealName { get; set; }

    0 references
    public string ImageUrl
    {
        get
        {
            return $"/Resources/Heroes/{Character}.png";
        }
    }
}
```

custom listbox template

LISTBOX: ITEMTEPLATE

```
<ListBox x:Name="lstHeroes" Background="Black" Foreground="White">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Border CornerRadius="0,8,0,13" Padding="4"
                    BorderBrush="#FF6CB3ED" BorderThickness="1">
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition/>
                        <RowDefinition/>
                    </Grid.RowDefinitions>
                    <Image Source="{Binding ImageUrl}"
                           Height="70" Width="120" Stretch="UniformToFill"/>
                    <TextBlock Text="{Binding Character}" Grid.Row="1"
                               HorizontalAlignment="Center" Foreground="White"
                               FontFamily="Bahnschrift Condensed" FontSize="14"/>
                </Grid>
            </Border>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

custom listbox template

LISTBOX: ITEMTEMPLATE - **BINDING**

- Inside an ItemTemplate of a ListBox:
  - Use markup extensions: { } to bind to a property
  - ```
    <Image Source="{Binding ImageUrl}"
           Height="70" Width="120" Stretch="UniformToFill"/>
    ```
  - ItemsSource of the listbox set to a `List<T>`:
    - Binding will look for this property inside the Type T

- DataBinding in WPF / XAML:
  - We have only just begun.. ☺
  - To be continued in MVVM (in 2 weeks)
    ```