# ICE Styling

## Part 1 – Styling Controls

We are going to expand our paging application now and style the output. There are multiple ways to style the app at the control level, page level, and at the app level. We will be doing each of those in this exercise.

For this example, we will be doing the styling in the xaml pages themselves. The same things could be done in the code behind, but it is much more complex.

Please keep in mind that since this is based on the Paging exercise, things might look a little different depending on your code.

1. Copy the Paging project into a new folder called Styling. The project name, namespace and everything will still say Paging, but that is ok.
2. Update the project to the latest Xamarin Form (4.8+).
3. Go to the DetailPage.xaml file and look at the Labels that you have. It might look similar to this:

<Label Text="{Binding Data.Name}" />

<Label Text="{Binding Data.YearOfRelease}" />

<Label Text="{Binding Data.Rating}" />

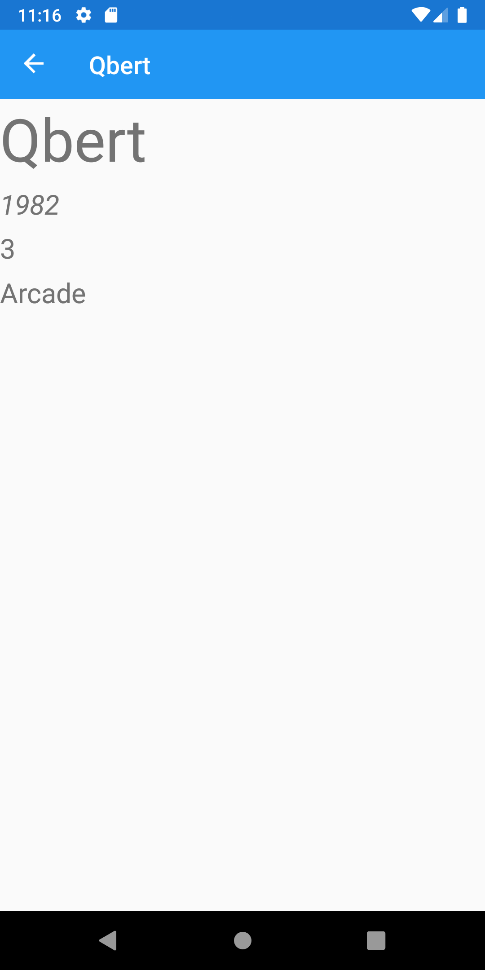
<Label Text="{Binding Data.Platform}" />

Each of these labels are showing parts of the display. Each part is the same size and font. Everything looks similar.

1. In the first label we are going to want that to be nice and large since it is the name of the game. In the label node (between the brackets) add a new property called **FontSize**. Notice that when you add the **FontSize** and then type the = it will give you a dropdown of possible values. Select the “Header” value and see what it will do.  
     
     
   Header Size  
     
   This is a little too big for what we want, so let us change to the “Large” value and see the output. This one is a little too small. The **FontSize** can also be given a pixel size as well, so change it to “48” and that will be what we use.

  
Large Size 48 Size  
  
HotLoading of the XAML page lets you edit the size, save the file, and then see the results while you are debugging. This lets you make changes pretty quickly and to see the output.

1. For the rest of the labels, add a **FontSize** with a value of “Large”.
2. Go back to the year display and add a **FontAttribute** with a value of “Italics”.

At this point, display should look like the following image, and the XAML code will look like the following:  
  
  
  
 <Label Text="{Binding Data.Name}" FontSize="48" />

<Label Text="{Binding Data.YearOfRelease}" FontSize="Large" FontAttributes="Italic" />

<Label Text="{Binding Data.Rating}" FontSize="Large" />

<Label Text="{Binding Data.Platform}" FontSize="Large" />

## Part 2 – Styling a Page

From this point we can see that when you have more and more controls on the page, setting the current font will be time consuming and slow. You can make a page level style for the labels to set the values that are needed.

1. At the top of the DetailPage.xaml, add a new **<ContentPage.Resources>** section. This is where we will be making the styles.
2. Create a **<Style>** in the Resources and set the **TargetType** to “Label”. This will tell the page that every label on the page will use these values.

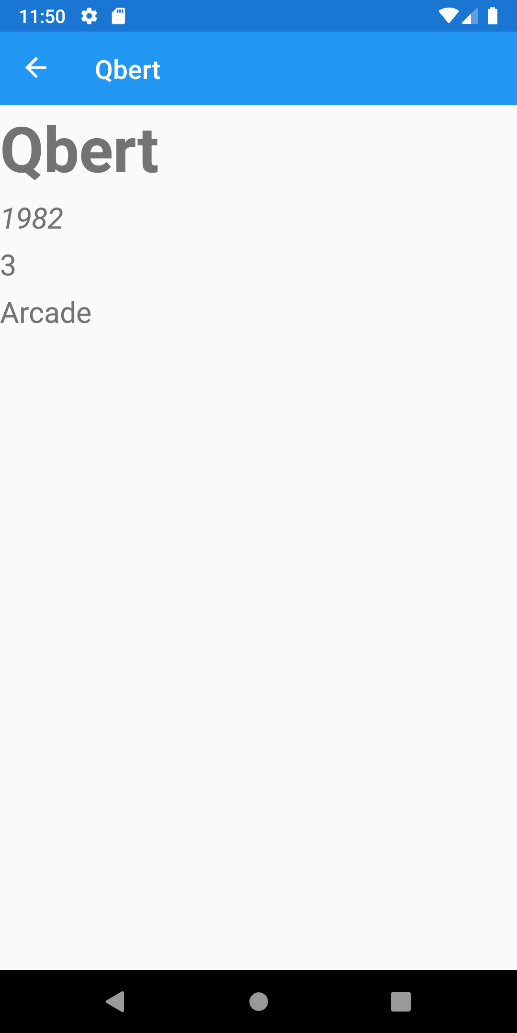
Inside the Style create a new **<Setter>** and then set the **Property** to “FontSize” and the **Value** to “Large”. The style will now look like this:  
  
  
 <ContentPage.Resources>

<Style TargetType="Label">

<Setter Property="FontSize" Value="Large" />

</Style>

</ContentPage.Resources>

1. Go to the labels on the page and then remove the **FontSize** that are set to “Large”. Run the app and you will see that is now looks the same. This one value can be changed and the sizes of each of these will change to the new size. Multiple <Setter> can be added to a single style.   
     
   This allows setting ANY property that is available for that **TargetType**. This means that you cannot set the **BorderColor** on a Label, but you can set that on a Button. Visual Studio will show an error and underline it with a red squiggly line if you add a property that does not belong to the **TargetType**.
2. Multiple styles can also be made for a single type so that you can have different styles for various parts of the display. Create a new **<Style>** and set the **TargetType** to “Label” again. Add a value to x:Key to the style (inside the brackets) and give it a value of “HeaderStyle”.  
     
   Add a Setter to set the **FontSize** to 48  
   Add a Setter to set the **FontAttribute** to Bold  
     
   Go down to the label for the game name and remove the **FontSize** property. Add a new property called **Style** and set it to "{StaticResource Key=HeaderStyle}". This is a similar format to how databinding is done but uses **StaticResource** instead of the word **Binding**.  
     
   Running the code now will allow you to see something similar to this:  
     
   

## Part 3 – Styling the App

Up until now, we have added styling to separate components as well as page level styles. Now there is one more thing that can you do that so that you can make the styles at the application level and it will use that for all pages and components. If you go from page to page at this point you will notice that the fonts are small on the main page and the about but on the details we see the larger font.

1. From the DetailPage.xaml copy the styles that were created and remove them from this file. Go to the App.xaml file and paste the styles into the **App.Resources** section.
2. Now the entire application will use a Large font size by default for every Label. You can override any other property such as TextColor, Padding, etc
3. Create a new named style for the other fields that you might have on the CollectionView or a special style for something on the AboutPage. Put this named style into the app.xaml and then use it in your MainPage or AboutPage.
4. Add a style for the Button to get to the AboutPage. Set the BackgroundColor, TextColor, and the CornerRadius

**Submission: ZIP and Post to the dropbox**