

## UWP-031 - Stupendous Styles Challenge

This next challenge creates an app for a fictitious company I created called GoNuts, a drive-through donut shop. This app would allow you to create your order and schedule it for a future date and time, then drive up to the take out window where robot delivers it to your car.

We're only going to focus on the layout and styling for this challenge. In other words, there are no real e-commerce functionality.

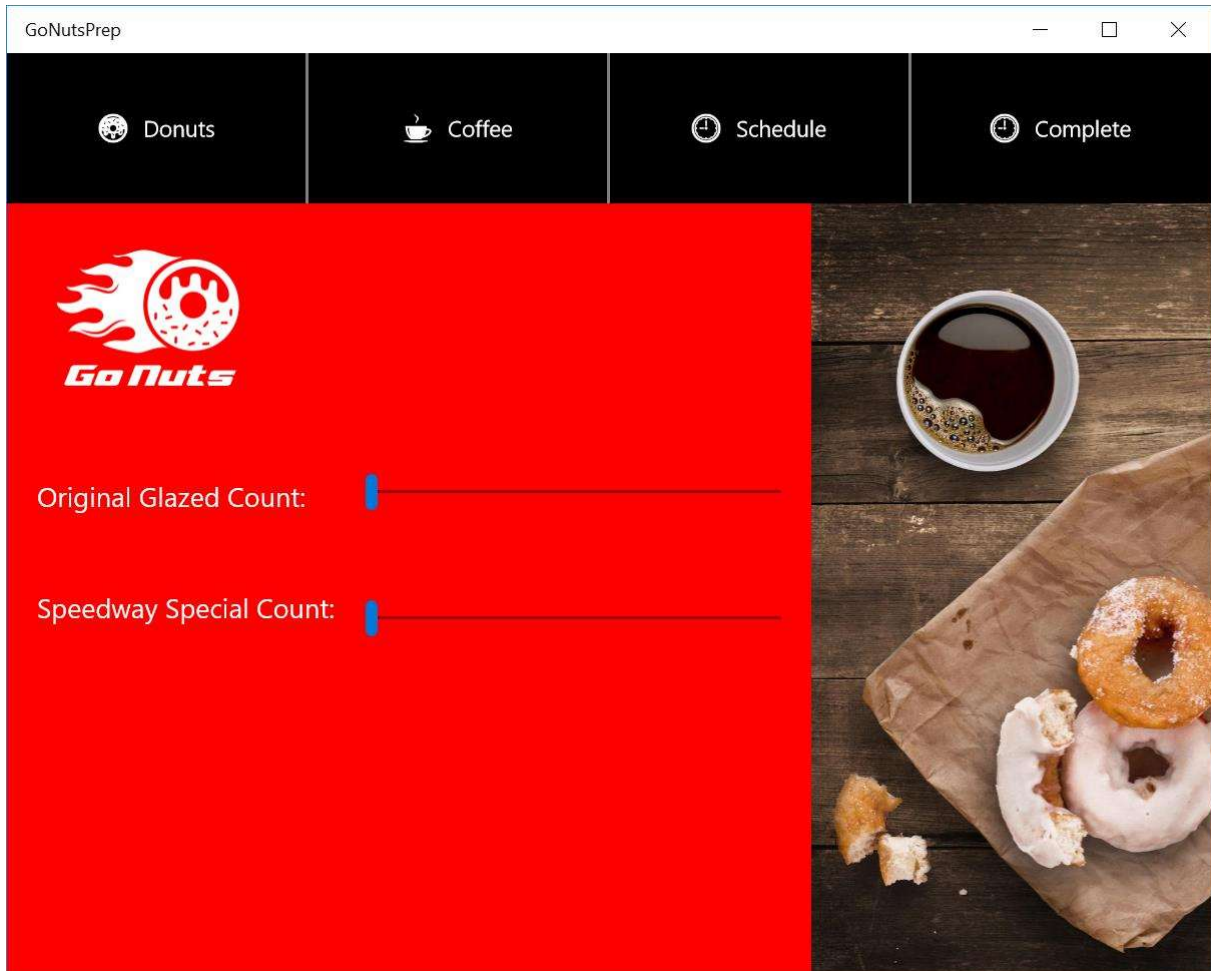
You'll use the resources contained in the zipped folder accompanying this lesson.

Refer to the instructions in the file called UWP-031-Instructions.txt

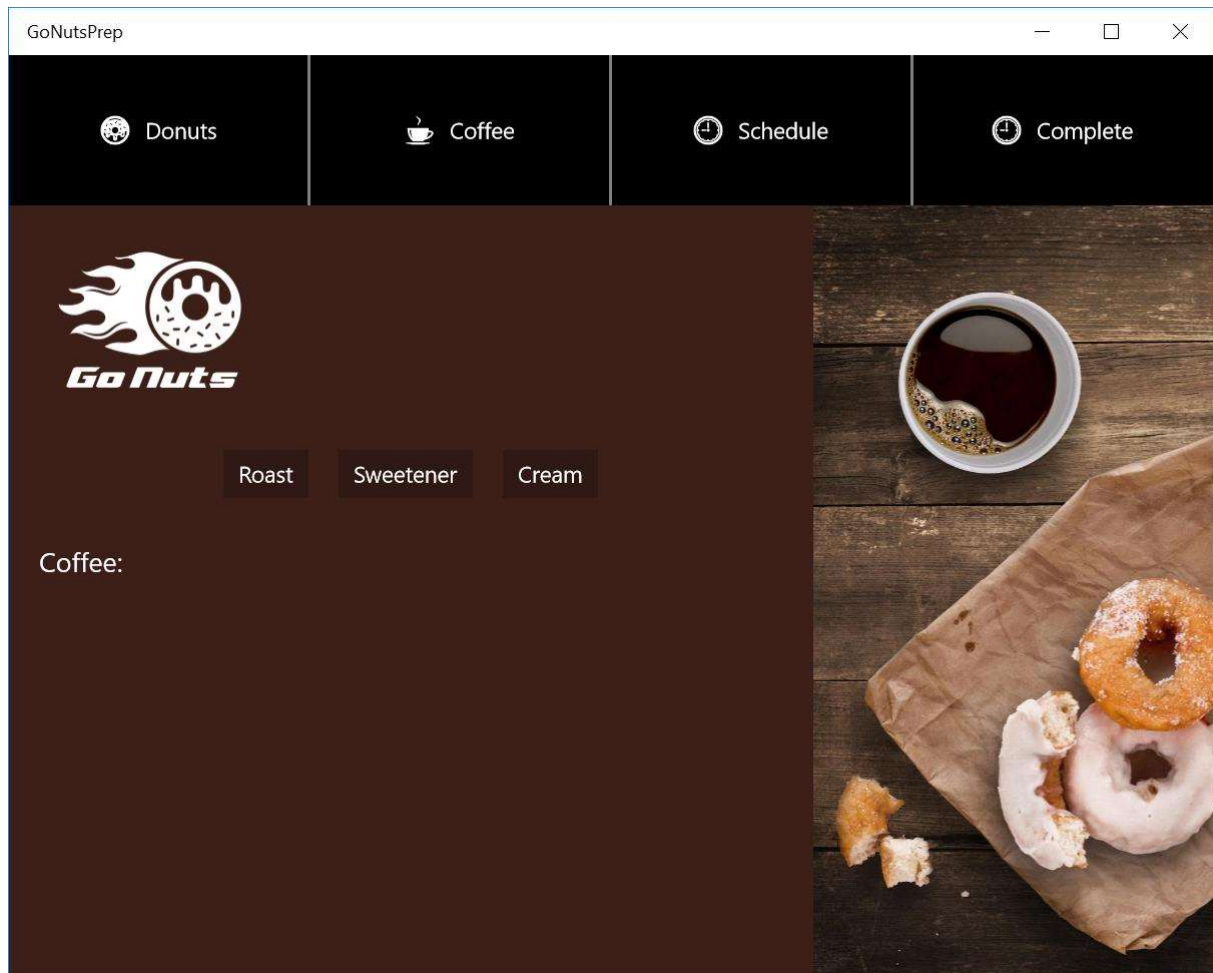
I'll briefly walk through each of the four pages you'll create.

There's a top navigation area with four clickable buttons.

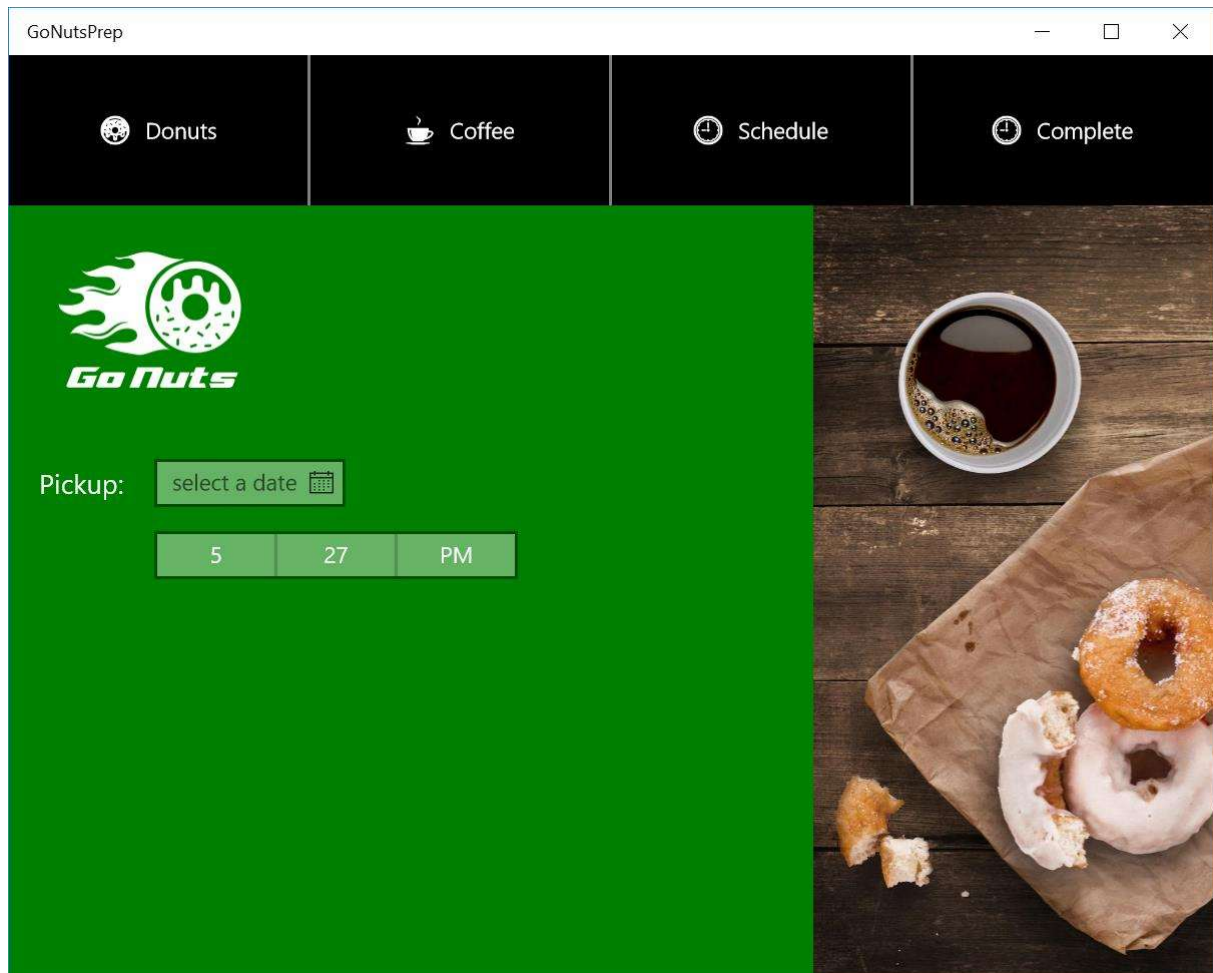
The Donuts button will allow the user to select the number of donuts for each type of donut the company makes:



The coffee button allows the user to add on a cup of coffee and customize it with cream and sweetener:

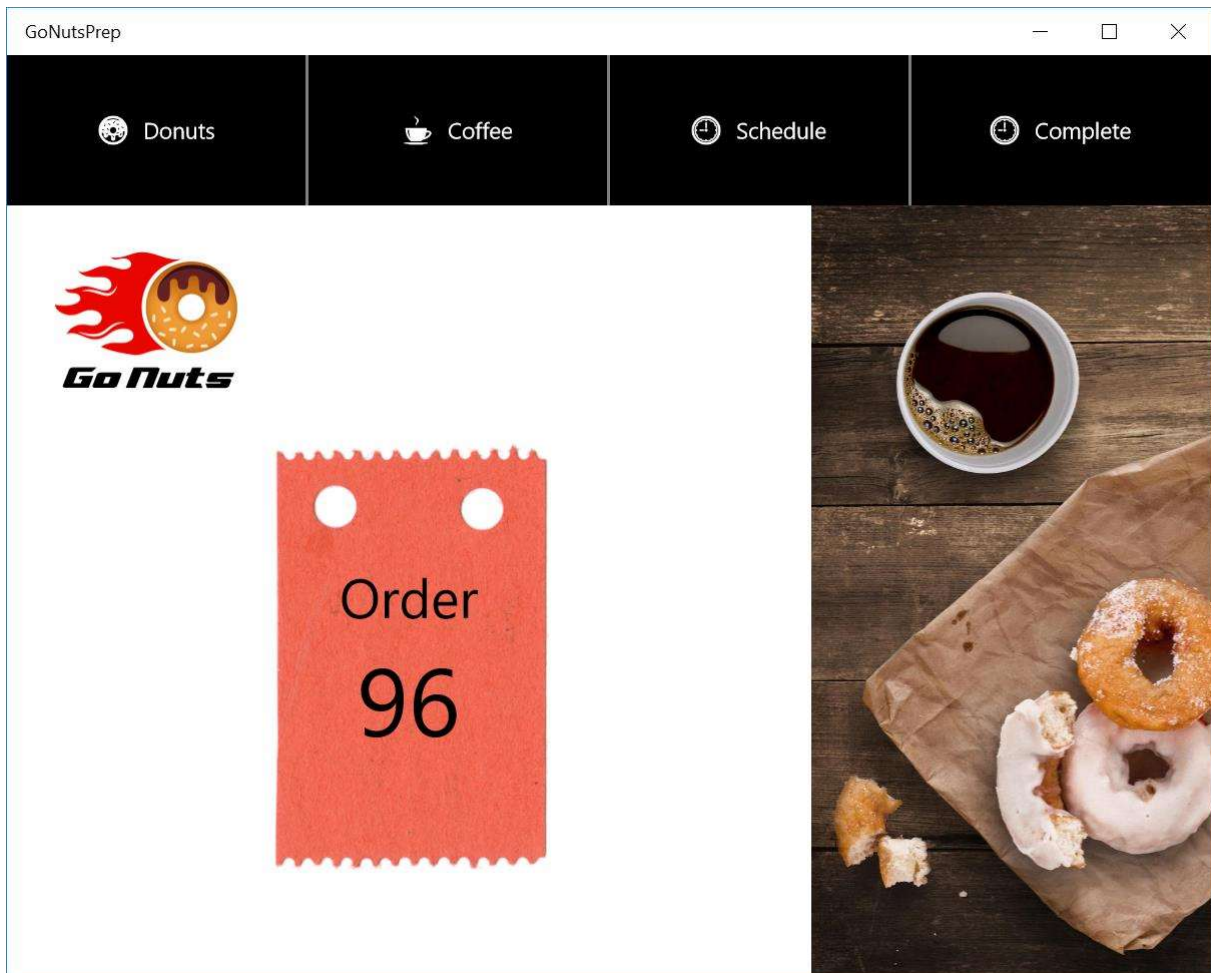


The schedule allows the user to specify a date and time to pick up the order:



The screenshot shows a web application window titled "GoNutsPrep". The interface has a dark navigation bar with four tabs: "Donuts" (with a donut icon), "Coffee" (with a coffee cup icon), "Schedule" (with a clock icon), and "Complete" (with a clock icon). The "Schedule" tab is currently active. The main content area is split into two panels. The left panel has a green background and features the "Go Nuts" logo (a donut with a flame) and a "Pickup:" section. This section includes a date selector with the text "select a date" and a calendar icon, and a time selector with three buttons: "5", "27", and "PM". The right panel displays a photograph of a cup of coffee and several donuts on a wooden surface.

The Complete button allows the user to get an order confirmation number styled as a ticket:



Create styles and resources as often as possible. Again, refer to the instructions document in the zipped folder, and utilize all of the graphical assets provided for you, along with the screenshots of the app you're attempting to build.

## UWP-032 - Stupendous Styles Challenge Solution - Part 1: MainPage

I'll split the solution to the challenge into several lessons focusing on just one part each time.

Also, the text version of these lessons are greatly abbreviated from the video format.

Step 1: Create a new Blank App Template project named GoNuts.

Step 2: Create an Assets folder and copy all of the graphic assets from the zipped folder associated with the previous lesson into that folder.

Step 3: Create the other pages named:

- DonutPage.xaml
- CoffeePage.xaml
- SchedulePage.xaml
- CompletePage.xaml

Step 4: On MainPage.xaml create an overall layout for the app. Add column and row definitions for each of the main sections of the app's user interface. In this case, two rows. Each row will contain it's own inner Grid.

```
23  <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
24      <Grid.RowDefinitions>
25          <RowDefinition Height="100" />
26          <RowDefinition Height="*" />
27      </Grid.RowDefinitions>
```

Step 6: In the first row, I'll create a Grid with four columns – one for each of the buttons.

```
29  <Grid>
30      <Grid.ColumnDefinitions>
31          <ColumnDefinition Width="*" />
32          <ColumnDefinition Width="*" />
33          <ColumnDefinition Width="*" />
34          <ColumnDefinition Width="*" />
35      </Grid.ColumnDefinitions>
```

Step 7: I'll create one of the four buttons in the first column of that grid, then copy / paste the other three modifying what's different about each one. Notice that during the course of building the first button I realize there are two opportunities to dramatically reduce the amount of code that will be required to define the button by creating two styles:



```

37 <Button Grid.Column="0" Style="{StaticResource NavigationButtonStyle}">
38     <StackPanel Orientation="Horizontal">
39         <Image Source="Assets/donut-icon.png"
40             Style="{StaticResource IconImageStyle}" />
41         <TextBlock Text="Donut" Foreground="White" />
42     </StackPanel>
43 </Button>

```

Here are the two styles as defined at the top of the Page:

```

9 <Page.Resources>
10     <Style TargetType="Button" x:Key="NavigationButtonStyle">
11         <Setter Property="Background" Value="Black" />
12         <Setter Property="HorizontalAlignment" Value="Stretch" />
13         <Setter Property="VerticalAlignment" Value="Stretch" />
14         <Setter Property="BorderBrush" Value="Gray" />
15         <Setter Property="BorderThickness" Value="0,0,2,0" />
16     </Style>
17     <Style TargetType="Image" x:Key="IconImageStyle">
18         <Setter Property="Height" Value="20" />
19         <Setter Property="Width" Value="20" />
20         <Setter Property="Margin" Value="0,0,10,0" />
21     </Style>
22 </Page.Resources>

```

Step 8: I copy and paste the first button three times, modifying the column, image and text for each:

```

45 <Button Grid.Column="1" Style="{StaticResource NavigationButtonStyle}">
46     <StackPanel Orientation="Horizontal">
47         <Image Source="Assets/coffee-icon.png"
48             Style="{StaticResource IconImageStyle}" />
49         <TextBlock Text="Coffee" Foreground="White" />
50     </StackPanel>
51 </Button>
52
53 <Button Grid.Column="2" Style="{StaticResource NavigationButtonStyle}">
54     <StackPanel Orientation="Horizontal">
55         <Image Source="Assets/schedule-icon.png"
56             Style="{StaticResource IconImageStyle}" />
57         <TextBlock Text="Schedule" Foreground="White" />
58     </StackPanel>
59 </Button>
60
61 <Button Grid.Column="3" Style="{StaticResource NavigationButtonStyle}">
62     <StackPanel Orientation="Horizontal">
63         <Image Source="Assets/complete-icon.png"
64             Style="{StaticResource IconImageStyle}" />
65         <TextBlock Text="Complete" Foreground="White" />
66     </StackPanel>
67 </Button>

```

Step 9: Remove the frame counter in the App.xaml.cs file.



## UWP-033 - Stupendous Styles Challenge Solution - Part 2: Navigation and DonutPage

Next I tackle navigation up in the Stupendous Styles Challenge solution and then work on the Donut Page.

Step 1: In the MainPage.xaml add a Grid that will be a child to the top-most grid, filling the second row. Create two column definitions. In the first column add a Frame named "MyFrame". In the second column, add an Image control whose source is the background.jpg in the Source folder. Set it to fill the entire area without stretching.

```
81 <Grid Grid.Row="1">
82   <Grid.ColumnDefinitions>
83     <ColumnDefinition Width="2*" />
84     <ColumnDefinition Width="1*" />
85   </Grid.ColumnDefinitions>
86   <Frame Name="MyFrame"></Frame>
87   <Image Source="Assets/background.jpg" Grid.Column="1" Stretch="UniformToFill" />
88 </Grid>
```

Step 2: Give each button a name corresponding to its function then add an event handler to each of the buttons.

```

37 <Button Name="DonutButton"
38     Click="DonutButton_Click" ←
39     Grid.Column="0"
40     Style="{StaticResource NavigationButtonStyle}">
41     <StackPanel Orientation="Horizontal">
42         <Image Source="Assets/donut-icon.png"
43             Style="{StaticResource IconImageStyle}" />
44         <TextBlock Text="Donut"
45             Foreground="White" />
46     </StackPanel>
47 </Button>
48
49 <Button Name="CoffeeButton"
50     Click="CoffeeButton_Click" ←
51     Grid.Column="1"
52     Style="{StaticResource NavigationButtonStyle}">
53     <StackPanel Orientation="Horizontal">
54         <Image Source="Assets/coffee-icon.png"
55             Style="{StaticResource IconImageStyle}" />
56         <TextBlock Text="Coffee" Foreground="White" />
57     </StackPanel>
58 </Button>
59
60 <Button Name="ScheduleButton"
61     Click="ScheduleButton_Click" ←
62     Grid.Column="2"
63     Style="{StaticResource NavigationButtonStyle}">
64     <StackPanel Orientation="Horizontal">
65         <Image Source="Assets/schedule-icon.png"
66             Style="{StaticResource IconImageStyle}" />
67         <TextBlock Text="Schedule" Foreground="White" />
68     </StackPanel>
69 </Button>
70
71 <Button Name="CompleteButton"
72     Click="CompleteButton_Click" ←
73     Grid.Column="3"
74     Style="{StaticResource NavigationButtonStyle}">
75     <StackPanel Orientation="Horizontal">
76         <Image Source="Assets/complete-icon.png"
77             Style="{StaticResource IconImageStyle}" />
78         <TextBlock Text="Complete" Foreground="White" />
79     </StackPanel>
80 </Button>

```

Put your mouse cursor on each of the click event handler names and press F12 to create the method stub for each.

Step 3: In the MainPage.xaml.cs, add navigation code to each button so that the Frame navigates to each of the XAML pages created in the previous lesson. Also, navigate to the DonutPage when the app first opens:

```
23 public sealed partial class MainPage : Page
24 {
25     public MainPage()
26     {
27         this.InitializeComponent();
28         MyFrame.Navigate(typeof(DonutPage));
29     }
30
31     private void DonutButton_Click(object sender, RoutedEventArgs e)
32     {
33         MyFrame.Navigate(typeof(DonutPage));
34     }
35
36     private void CoffeeButton_Click(object sender, RoutedEventArgs e)
37     {
38         MyFrame.Navigate(typeof(CoffeePage));
39     }
40
41     private void ScheduleButton_Click(object sender, RoutedEventArgs e)
42     {
43         MyFrame.Navigate(typeof(SchedulePage));
44     }
45
46     private void CompleteButton_Click(object sender, RoutedEventArgs e)
47     {
48         MyFrame.Navigate(typeof(CompletePage));
49     }
50 }
```

Step 4: In the DonutPage.xaml, create the layout for the page adding four row and two column definitions. Set the Grid's background to red.

```

10 <Grid Background="Red">
11     <Grid.RowDefinitions>
12         <RowDefinition Height="Auto" />
13         <RowDefinition Height="Auto" />
14         <RowDefinition Height="Auto" />
15         <RowDefinition Height="*" />
16     </Grid.RowDefinitions>
17     <Grid.ColumnDefinitions>
18         <ColumnDefinition Width="Auto" />
19         <ColumnDefinition Width="*" />
20     </Grid.ColumnDefinitions>
21

```

Step 5: Add the white logo into row 1, column 1. Two TextBlocks as labels and two slider controls to allow the user to select the donut count for both types of donuts. Set both slider's Maximum property to 24. Add margins as needed.

```

22     <Image Source="Assets/white-logo.png" Width="150" Margin="20,20,0,0" />
23
24     <TextBlock Grid.Row="1"
25         FontSize="18"
26         Text="Original Glazed Count:"
27         Foreground="White"
28         Margin="20,20,20,0" />
29
30     <Slider Grid.Row="1"
31         Grid.Column="1"
32         Maximum="24"
33         Margin="20,20,20,0" />
34
35     <TextBlock Grid.Row="2"
36         FontSize="18"
37         Text="Speedway Special Count:"
38         Foreground="White"
39         Margin="20,20,20,0" />
40
41     <Slider Grid.Row="2"
42         Grid.Column="1"
43         Maximum="24"
44         Margin="20,20,20,0" />
45

```

## UWP-034 - Stupendous Styles Challenge Solution - Part 3: CoffeePage

Step 1: In the CoffeePage.xaml create the layout using four row definitions. Set the background to the required brown color:

```
16 <Grid Background="#3C1F19">
17     <Grid.RowDefinitions>
18         <RowDefinition Height="Auto" />
19         <RowDefinition Height="Auto" />
20         <RowDefinition Height="Auto" />
21         <RowDefinition Height="*" />
22     </Grid.RowDefinitions>
23
```

Step 2: Now that we see we'll use the white logo on multiple pages, it's time to extract it's property settings and create a style. In App.xaml, create a new Style targeted at the image control named "WhiteLogoStyle".

```
1 <Application
2     x:Class="GoNuts.App"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:GoNuts"
6     RequestedTheme="Light">
7     <Application.Resources>
8         <Style TargetType="Image" x:Key="WhiteLogoStyle">
9             <Setter Property="Source" Value="Assets/white-logo.png" />
10            <Setter Property="Width" Value="150" />
11            <Setter Property="Margin" Value="20,20,20,0" />
12            <Setter Property="HorizontalAlignment" Value="Left" />
13        </Style>
14
```

Back on the CoffeePage.xaml, add the image using the "WhiteLogoStyle":

```
23
24     <Image Style="{StaticResource WhiteLogoStyle}" />
25
```

Back on the donut page, do the same.

Step 2: Add a StackPanel for row 2, set the orientation to horizontal and horizontal alignment to center:



```

25
26 <StackPanel Orientation="Horizontal"
27           Grid.Row="1"
28           HorizontalAlignment="Center">
29

```

Step 3: Create a Button with a MenuFlyout, and once styled correctly, copy and paste two times replacing the content of each. The button's properties can be extracted to a local style to reduce the amount of XAML required so create a new style named FlyoutButtonStyle at the top of the page:

```

30 <Button Content="Roast"
31       Style="{StaticResource FlyoutButtonStyle}">
32   <Button.Flyout>
33       <MenuFlyout>
34           <MenuFlyoutItem Text="None" Click="Roast_Click" />
35           <MenuFlyoutItem Text="Dark" Click="Roast_Click" />
36           <MenuFlyoutItem Text="Medium" Click="Roast_Click" />
37       </MenuFlyout>
38   </Button.Flyout>
39 </Button>
40

```

Notice the MenuFlyoutItems' Click event handler names are all the same.

The FlyoutButtonStyle should be defined like this on the CoffeePage.xaml:

```

1 <Page
2   x:Class="GoNuts.CoffeePage"
3   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5   xmlns:local="using:GoNuts"
6   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8   mc:Ignorable="d">
9   <Page.Resources>
10       <Style TargetType="Button" x:Key="FlyoutButtonStyle">
11           <Setter Property="Margin" Value="10" />
12           <Setter Property="Foreground" Value="White" />
13       </Style>
14   </Page.Resources>
15

```

Copy, paste and change the button twice to add the other two buttons, changing the appropriate values:



```

41         <Button Content="Sweetener"
42             Style="{StaticResource FlyoutButtonStyle}">
43             <Button.Flyout>
44                 <MenuFlyout>
45                     <MenuFlyoutItem Text="None" Click="Sweetener_Click" />
46                     <MenuFlyoutItem Text="Sugar" Click="Sweetener_Click" />
47                 </MenuFlyout>
48             </Button.Flyout>
49         </Button>
50
51         <Button Content="Cream"
52             Style="{StaticResource FlyoutButtonStyle}">
53             <Button.Flyout>
54                 <MenuFlyout>
55                     <MenuFlyoutItem Text="None" Click="Cream_Click" />
56                     <MenuFlyoutItem Text="2% Milk" Click="Cream_Click" />
57                     <MenuFlyoutItem Text="Whole Milk" Click="Cream_Click" />
58                 </MenuFlyout>
59             </Button.Flyout>
60         </Button>
61     </StackPanel>

```

Again, take note of the Click event handler for each of the MenuFlyoutItems.

Step 4: In the third row of the layout Grid, add a StackPanel and two TextBlocks, one for the label and the other to be used to display the current selections from the Button's FlyoutMenus.

```

63     <StackPanel Orientation="Horizontal" Grid.Row="2">
64         <TextBlock Text="Coffee:" Style="{StaticResource LabelTextBlockStyle}" />
65         <TextBlock Name="ResultTextBlock" Style="{StaticResource LabelTextBlockStyle}" />
66     </StackPanel>
67

```

When styling the TextBlocks, it seems like we have a good candidate for an app level Style. Create a new style in App.xaml named: LabelTextBlockStyle:

```

14     <Style TargetType="TextBlock" x:Key="LabelTextBlockStyle">
15         <Setter Property="FontSize" Value="18" />
16         <Setter Property="Foreground" Value="White" />
17         <Setter Property="Margin" Value="20,20,20,0" />
18     </Style>
19 </Application.Resources>
20 </Application>
21

```

Go to the DonutPage.xaml and utilize the new style for the two TextBlocks.

Step 5: Place your mouse cursor on each of the three Click event handler names and press F12 to create method stubs.

Step 6: On CoffeePage.xaml.cs, create three private fields. Inside of each of the Click events, grab the selected MenuFlyoutItem, grab its text and put it into the respective private field.

```

34     private void Roast_Click(object sender, RoutedEventArgs e)
35     {
36         var selected = (MenuFlyoutItem)sender;
37         _roast = selected.Text;
38         displayResult();
39     }
40
41     private void Sweetener_Click(object sender, RoutedEventArgs e)
42     {
43         var selected = (MenuFlyoutItem)sender;
44         _sweetener = selected.Text;
45         displayResult();
46     }
47
48     private void Cream_Click(object sender, RoutedEventArgs e)
49     {
50         var selected = (MenuFlyoutItem)sender;
51         _cream = selected.Text;
52         displayResult();
53     }
54
55     }

```

Notice that I'm calling a private helper method `displayResult()` in each click event handler. This will be delegated presentation logic for what will be displayed in the result label.

Step 7: Create the `displayResult()` helper method. The cream and sweetener should only be displayed when a roast has been selected. Style the entire string of text appropriately:

```

57     private void displayResult()
58     {
59         if (_roast == "None" || String.IsNullOrEmpty(_roast))
60         {
61             ResultTextBlock.Text = "None";
62             return;
63         }
64
65         ResultTextBlock.Text = _roast;
66
67         if (_cream != "None" && !String.IsNullOrEmpty(_cream))
68             ResultTextBlock.Text += " + " + _cream;
69
70         if (_sweetener != "None" && !String.IsNullOrEmpty(_sweetener))
71             ResultTextBlock.Text += " + " + _sweetener;
72
73     }

```

## UWP-035 - Stupendous Styles Challenge Solution - Part 4: SchedulePage

Next we'll focus on the Schedule Page. It is pretty straight forward now that we have several app level styles defined.

Step 1: Add four row and two column definitions, set the background of the grid to Green:

```
10 <Grid Background="Green">
11   <Grid.RowDefinitions>
12     <RowDefinition Height="Auto" />
13     <RowDefinition Height="Auto" />
14     <RowDefinition Height="Auto" />
15     <RowDefinition Height="*" />
16   </Grid.RowDefinitions>
17   <Grid.ColumnDefinitions>
18     <ColumnDefinition Width="Auto" />
19     <ColumnDefinition Width="*" />
20   </Grid.ColumnDefinitions>
```

Step 2: Add the logo:

```
21
22   <Image Style="{StaticResource WhiteLogoStyle}" />
23
```

Step 3: Create the label and use the LabelTextBlockStyle:

```
24   <TextBlock Grid.Row="1"
25     Style="{StaticResource LabelTextBlockStyle}"
26     Text="Pickup:" />
27
```

Step 4: Add a CalendarDatePicker

```
28   <CalendarDatePicker Grid.Row="1" Grid.Column="1" Foreground="White" />
29
```

Step 5: Add a TimePicker

```
30   <TimePicker Grid.Row="2" Grid.Column="1" Foreground="White" />
31
```

## UWP-036 - Stupendous Styles Challenge Solution - Part 5: CompletePage

This is the last of the solution lesson for the Stupendous Styles Challenge and will focus on the CompletePage.

Step 1: Create two row definitions.

```
10 <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
11     <Grid.RowDefinitions>
12         <RowDefinition Height="Auto" />
13         <RowDefinition Height="Auto" />
14     </Grid.RowDefinitions>
```

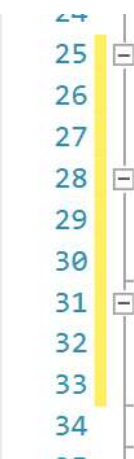
Step 2: The logo on this final page uses a different image file, otherwise it shares all of the same settings as the previous Page's logos. Use the WhiteLogoStyle, but add a Source property to override the white logo with the color logo.

```
16 <Image Style="{StaticResource WhiteLogoStyle}"
17       Source="Assets/color-logo.png" />
```

Step 3: In the second row, add the ticket.jpg and set its width to 300. Center it both horizontally and vertically.

```
19 <Image Grid.Row="1"
20       Source="Assets/ticket.jpg"
21       Width="300"
22       VerticalAlignment="Center"
23       HorizontalAlignment="Center" />
```

Step 4: We'll take advantage of how the Grid works by overlaying two TextBlocks containing the text "Order" and "96", respectively. To ensure they are centered, use a StackPanel and set its vertical and horizontal alignment to center.



```
<StackPanel Grid.Row="1"
    VerticalAlignment="Center"
    HorizontalAlignment="Center">
    <TextBlock Text="Order"
        FontSize="36"
        HorizontalAlignment="Center" />
    <TextBlock Text="96"
        FontSize="64"
        VerticalAlignment="Center" />
</StackPanel>
```