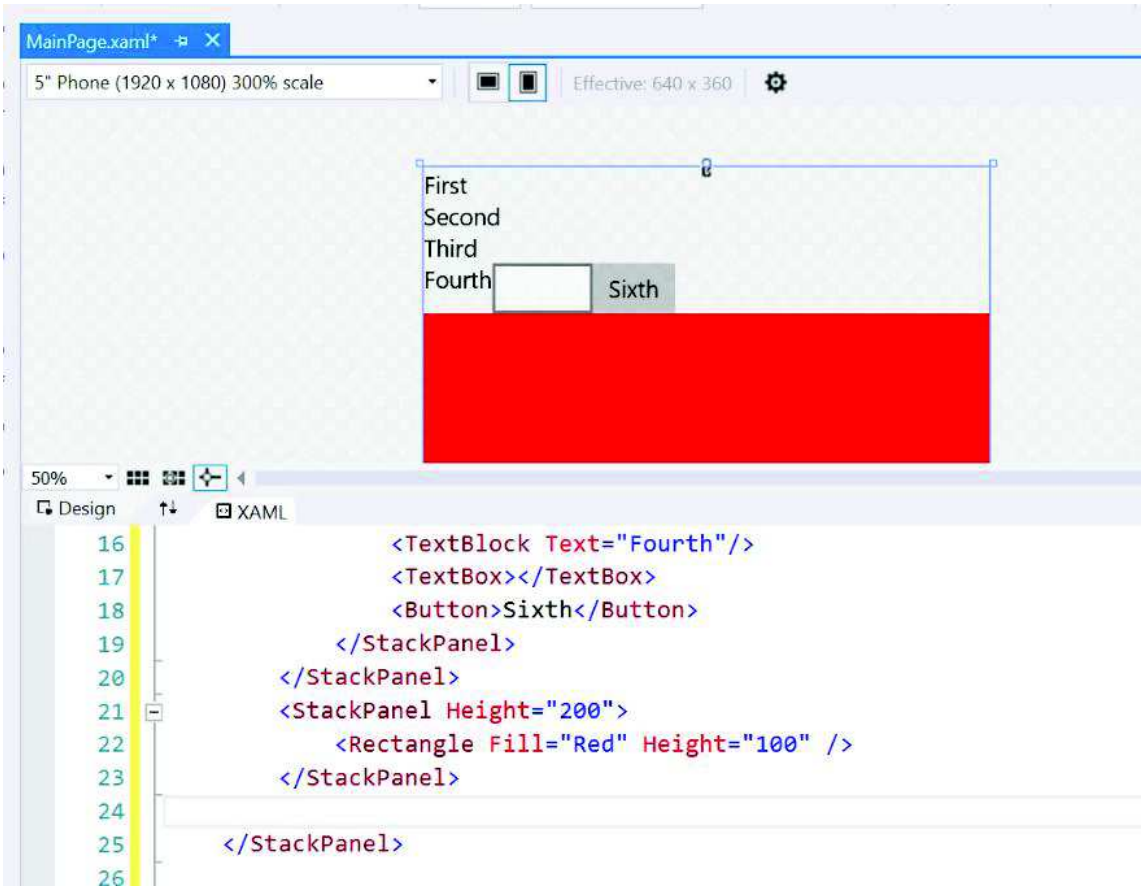
# UWP-009 - XAML Layout with StackPanel

The StackPanel layout control allows you to arrange your XAML Controls in a flow from top to bottom by default.  Or we can change the Orientation property to flow from left to right in a horizontal fashion.  We can even change the flow to go from left-to-right to right-to-left.

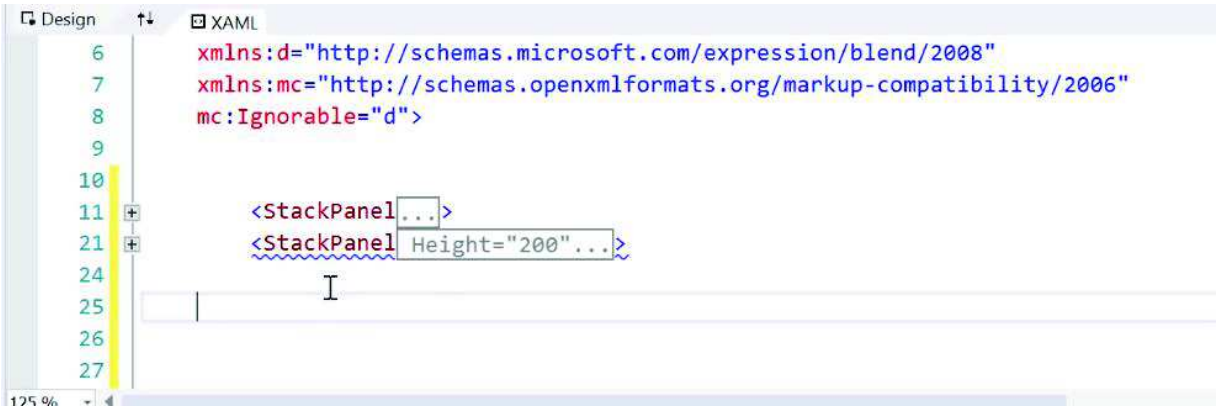
Creaing a StackPanel is simple.  It's simply a panel (almost like a div tag, if you're familiar with HTML development), and inside of the opening and closing tags you can add XAML controls which will be stacked on top of each other or side by side.

I created a very simple application named SimpleStackPanel.




As a side note: notice that I use a little plus and minus symbols in the left-most column to roll up my code so we can see the overall structure of this application.


And you might be wondering, well why do we have StackPanels inside of StackPanels? Why can't we just do something like this?




If I remove the outer-most StackPanel I get the blue squiggly line with the familiar exception that the property "Content" is set more than once. A Page only has a Content property and it does not have a Children property collection. So that's why we need an outermost StackPanel and then inside of that we can put as many StackPanels or other XAML controls as needed.

Inside of the first StackPanel, notice that I have three TextBlocks that are stacked on top of each other.

```
Design  XAML
10      <StackPanel>
11          <StackPanel>
12              <TextBlock>First</TextBlock>
13              <TextBlock>Second</TextBlock>
14              <TextBlock>Third</TextBlock>
15              <StackPanel Orientation="Horizontal">
16                  <TextBlock Text="Fourth"/>
17                  <TextBox></TextBox>
18                  <Button>Sixth</Button>
19              </StackPanel>
20          </StackPanel>
```

Notice they take up the full width because we haven't specified a width and by default, and that they're arranged in a vertical fashion. The item at the top will be the first item stacked, the item at the bottom will be the last item stacked. So the stacking is performed in order.

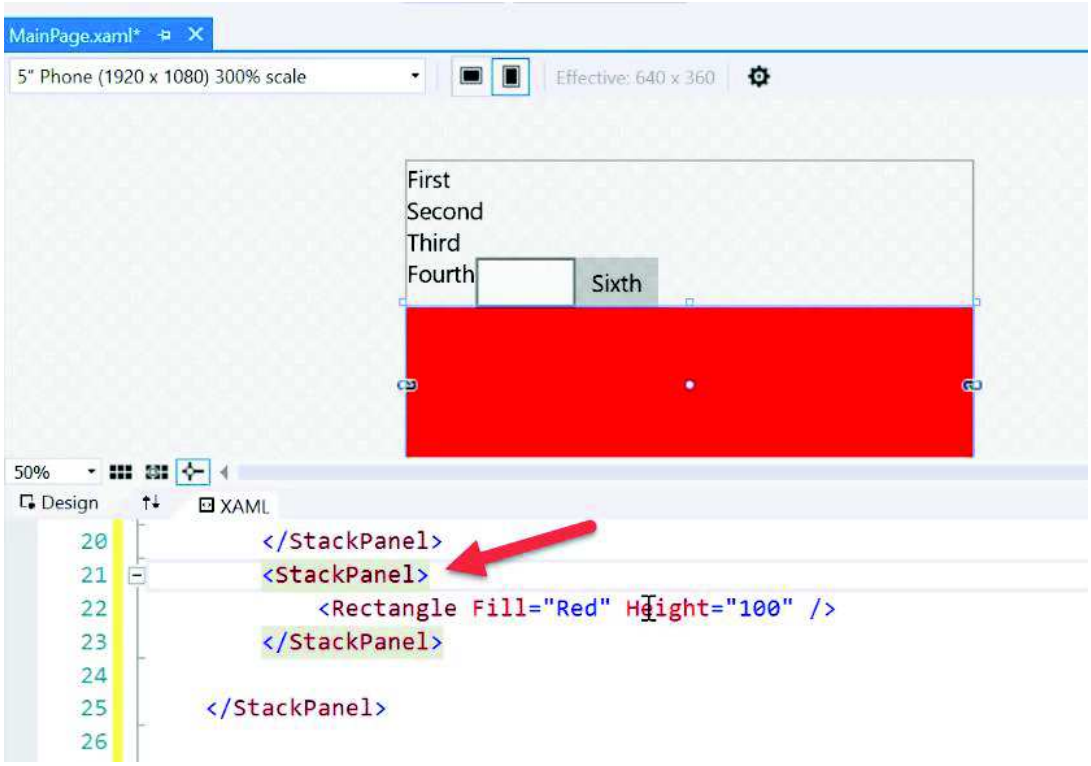
Furthermore, underneath the third TextBlock, I've added a second StackPanel. I set its Orientation to Horizontal. I'm using the StackPanel here as another "row" that's stacked beneath the TextBlocks. Inside of the inner-most StackPanel, I stacked several XAML Controls in a horizontal fashion. The leftmost stacked item is a TextBlock that has the word "Fourth" in it, the next item that is stacked next to it is a TextBox, then the next item is a button with the content, "Sixth", in it.

We can achieve almost anything by simply using this technique of stacking StackPanels inside of StackPanels. I tend to use StackPanel actually more than I use the Grid for layout. Personally, I think that it allows me to get the design that I want and it gives me the flow that I want regardless of the size of the device that the app is actually running on.

You can also see that I've got a StackPanel defined beneath it with a simple rectangle.

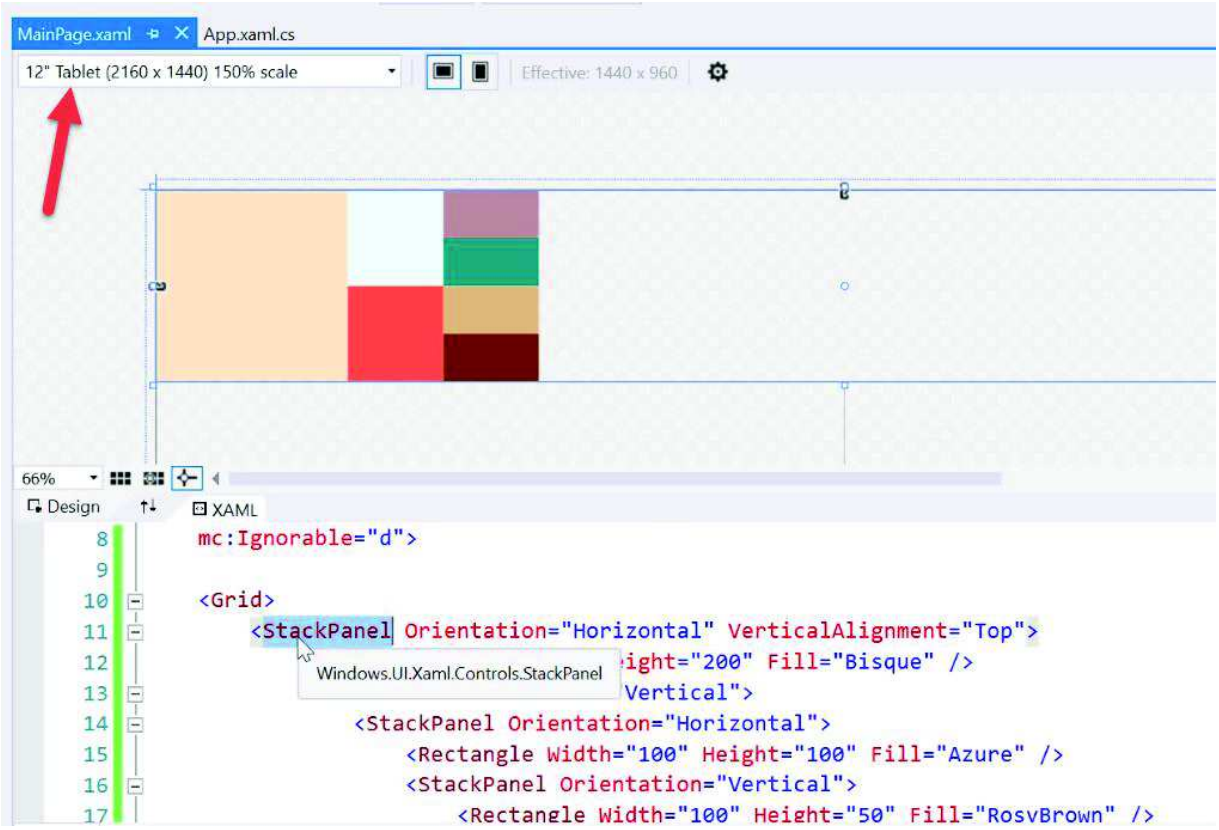
```
Design  XAML
20          </StackPanel>
21          <StackPanel Height="200">
22              <Rectangle Fill="Red" Height="100" />
23          </StackPanel>
24
25      </StackPanel>
```

The only thing I want to illustrate here is that the height of the StackPanel can be set, and yet items inside of the StackPanel can have their own height independent of the parent. If I were to remove the height, notice what happens:




Essentially, the Height of the StackPanel is set to Auto. If we were try to set it to star (*), it wouldn't work but we can set it to a numerical pixel value or we can just leave it at the default, which is Auto.

I created another example called ComplexStackPanel. Note that I have changed from the default viewer, that would be a five-inch phone, to a 12" Tablet to accommodate the larger width of this design that I created.



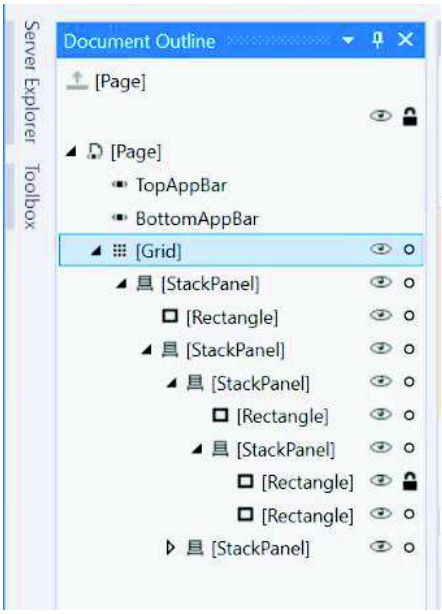

This example takes the concept of adding StackPanels inside of StackPanels to an extreme. Admittedly, it can become confusing to read the code and understand what code is producing a given StackPanel, however, hopefully, between the color descriptions and holding your mouse cursor on it and seeing the little boundary box that is selected you can make heads or tails of how this works.

The topmost StackPanel defines a “row” of sorts. The height of the top-most StackPanel is actually dictated by the items that are inside of it … specifically rectangle with the Fill="Bisque". Its height is set to 200 so that sets the height for the entire StackPanel since nothing else inside of it has a greater height.

Also, you'll notice that I set the VerticalAlignment of this top-most StackPanel, and the reason I did that, because if you change that, it will now get moved to the vertical middle of the grid cell, and that's just the difference in how grids work and StackPanels work.

So the result of StackPanels inside of StackPanels produces the desired intricacy until I have the series of rectangles that resemble something like an ornate pane of Frank Lloyd Wright designed glass.

One of the things that can help you out whenever you are working through an intricate amount of XAML and it's difficult to find your way around is this little Document Outline. If you do not see it by default on this left-most next to the Toolbox, you can go to the View menu > Other Windows > Document Outline.




You can expand each node in the Document Outline to reveal the hierarchy of items in the Designer. This is a representation of a “visual tree”. By selecting a given leaf in the tree you can see the little boundary box selection around the associated XAML Control in the Designer.

Furthermore, the Document Outline is very convenient whenever you want to make changes in the Properties window and it's difficult to find the exact item that you are looking for just by clicking around the Designer and the XAML is too intricate.

The Document Outline also can be used to hide items by clicking the left-column on the right-hand side to remove certain items from view. We can also lock items which means that they can't be selected and therefore they cannot easily be changed in the Properties window. You can see that it adds this IsLocked="True" XAML to the design time experience.

The final example project named GridAndStackPanel demonstrates a “gotcha” when using StackPanels. Furthermore it should help us to have a better understanding of the difference between Grids and StackPanels, and something that you need to watch out for and completely understand.

```
9
10      <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
11          <StackPanel...>
21          <StackPanel Height="200"...>
24      </Grid>
25  </Page>
```

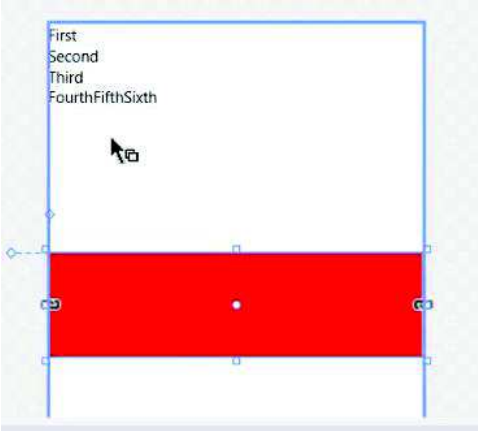
This is almost identical to the first example that we used earlier in this lesson. This time, we have a Grid that surrounds two StackPanels. The top-most StackPanel contains three TextBlocks, first, second and third, and then another embedded StackPanel whose Orientation property is set to Horizontal. This time I just have three TextBlocks that will be stacked horizontally, so we got fourth, fifth and sixth.

```
11          <StackPanel>
12              <TextBlock>First</TextBlock>
13              <TextBlock>Second</TextBlock>
14              <TextBlock>Third</TextBlock>
15              <StackPanel Orientation="Horizontal">
16                  <TextBlock>Fourth</TextBlock>
17                  <TextBlock>Fifth</TextBlock>
18                  <TextBlock>Sixth</TextBlock>
```

Below that, we have another StackPanel with a rectangle inside of it.

```
20          </StackPanel>
21          <StackPanel Height="200">
22              <Rectangle Fill="Red" Height="100" />
23          </StackPanel>
24      </Grid>
```

Now in this case, you'll look at it and say, why is there so much space in between this Rectangle and the bottom of this StackPanel?




And you might attempt to resolve this by setting the VerticalAlignment="Top". It would appear as though you lost the top-most StackPanel. You didn't lose it, actually. It is sitting behind the second StackPanel (the one with the red Rectangle).

There are three things going on here at the same time, and this will hopefully help you to understand the difference between Grids and StackPanels.

First, some controls like image and rectangle controls that set themselves inside of a grid cell will be set to 100 width and 100 height by default. So that is also true of StackPanels. When you put a StackPanel inside of a grid cell, in this case the default grid cell, cell zero or row zero, column zero, then the two StackPanels are essentially, well, the first StackPanel has set itself to 100 height and 100 width.

Now the second StackPanel has set itself to 200 height, but the second thing that you need to understand that by default the content in a cell of a grid will be vertically and horizontally centered, so

we have the second StackPanel, and let's remove the VerticalAlignment. The default here would be center. And so you can see that it is actually sitting, the entire StackPanel is in the center of that cell.

Third, you can easily overlap items in a grid cell. So if two or more controls are set to reside in the exact same grid cell, and their HorizontalAlignment equals top and their VerticalAlignment equals top, then they will literally sit on top of each other. We're actually looking at two StackPanels. This first StackPanel takes up the entire length, and the items inside of it, however, are aligned to the top of that StackPanel, but the StackPanel itself takes up the whole frame. And then the second StackPanel is set into the middle of that cell, but it's only 200 tall. However, whenever we change the VerticalAlignment and now we move that up to the very top of that single cell inside the grid, that's when we overlapped.

So I just want you to understand that about grids and about StackPanels, that StackPanels will never ever allow you to put two of them on top of each other where you cannot see the one that is underneath it. Whereas grid cells will absolutely allow you to do that.

There are a couple of different ways that we could rectify this. The easiest one is just to use StackPanels as the outermost container here in this particular layout. However, you could also create two rows instead of just one row and put this first StackPanel in the top row, put the second StackPanel in the bottom row, and That will ensure that they don't overlap. You could also set the StackPanels, you can set its VerticalAlignment to the top but then use a margin to push it down. That seems a little more fragile, I don't know if I like that idea.

There are a couple of different ways to still make this work even though you are working with a grid. But ideally, you'll probably switch to just using StackPanels. In fact, like I said earlier, I typically use the StackPanel just about for everything. I don't use Grids as much as I used to, except to give the page an overall structure. And with everything else, I try to use StackPanels.

There is this special technique, however, that I learned when we talk about adaptive triggers and adaptive layout that utilizes Grids to adapt from a desktop to a mobile layout, that wouldn't be possible with a StackPanel. The Grid still does have its uses. I just prefer, in most cases, to stick with the StackPanel.