

## UWP-004 - What Is XAML?

It's easy to figure out the absolute basics of XAML just by staring at the syntax. I imagine you were able to figure out the correlation between the tags, the properties, and what you saw in the visual designer in the previous lessons.

This lesson will discuss XAML's features and functions that may not be so obvious at first glance. Hope to take a couple of passes at this in subsequent lessons that, when combined together, will give us a pretty thorough understanding of XAML and how it all works.

First, we will talk about the purpose and the nature of XAML, especially in comparison to C#. Then in the next few lessons, we will talk about the special features of XAML; hidden features of the language that, again, may not be obvious when you first start looking at it.

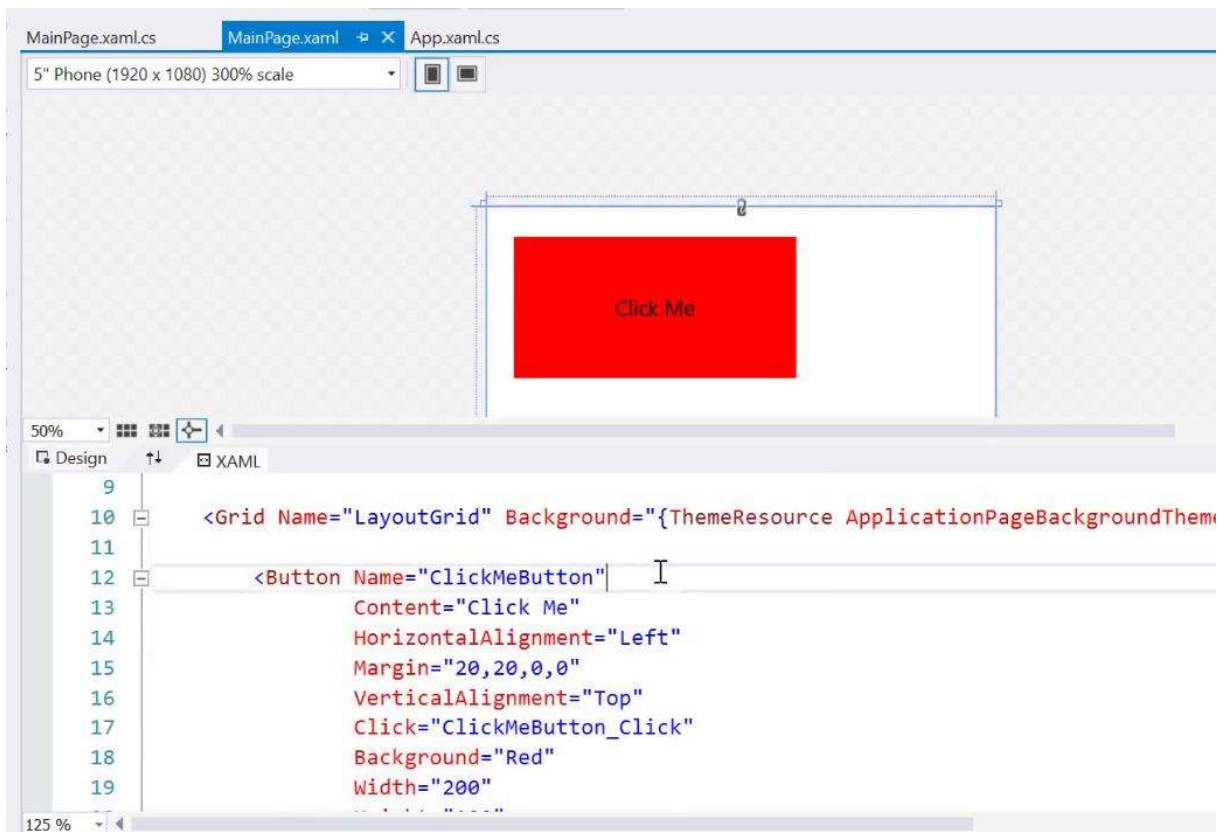
By the end of these first set of lessons my hope is that you'll have enough knowledge that you can look at the XAML that we write together in the remainder of this series and take a pretty good guess at what it's trying to do before I even explain what it does.

In the previous lesson, I made a passing remark about how XAML looks similar to HTML and that's no accident. XAML is really just XML, the eXtensible Markup Language. I'll explain that relationship in just a moment, but at a higher level, XML looks a lot like HTML inasmuch that they share a common ancestry. Whereas HTML is specific to structuring a web page document, XML is a little bit more generic in nature. And by generic I mean that you can use it for any purpose that you devise, and you can define the names and the elements and the attributes to suit your needs.

In the past, developers have used XML for things like storing application settings or using it as a means of transferring data between two systems that were never intended to work together. And to use XML, you start out by defining what's called a Schema which declares the proper names of the elements and their attributes. A schema is similar to a contract that two parties agree to. Everybody agrees, both the producer of the XML and the consumer of the XML, to write and read the XML, to conform to those rules set forth in the schema. And now that they both agree that they're working in the same rules and the same contract, they can communicate with each other. So a schema is a really important part of XML. And just keep that in mind, because we will come back to that in just a moment.

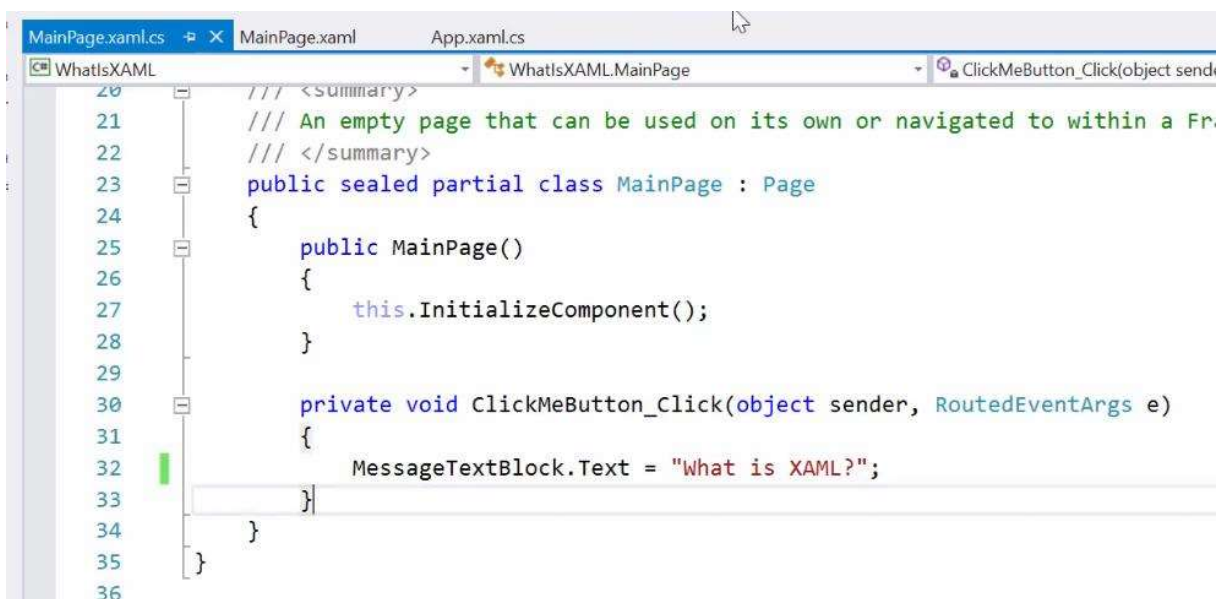
XAML is a special usage of XML. Obviously we see, at least in this case, XAML has something to do with defining a user interface for our application. So in that regard, it feels a lot like HTML, however there is a big difference. XAML is actually used to create instances of classes and set values of their properties.

For example, in the previous lesson, we created a little Hello World application. I expanded on that for this lesson, just to add a little more design. I added a button. Notice that I put all the attributes on separate lines so now we can see, hopefully, the definition a little bit better. There's a red button called "ClickMeButton", and there are also a MessageTextBlock that will appear below it.



It is very similar to the previous example, however I simply started over from scratch.

When a user clicks the button the Click event in the code behind will set the TextBlock's Text property equal to "What is XAML?"



To demonstrate how XAML works I will comment out the Button control in the XAML editor. To comment out XAML you use the following syntax:

```
<!--
```

Your XAML here

```
-->
```

Visual Studio's code editor displays comments with a green foreground by default.

Next, I want to give the Grid control a name to access it programmatically. I give it the name LayoutGrid because that is indicative of what it's actually doing for us.

```
7      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8      mc:Ignorable="d">
9
10     <Grid Name="LayoutGrid" Background="{ThemeResource ApplicationPageBackground}"
11
12         <!--
13         <Button Name="ClickMeButton"
14             Content="Click Me"
```

Next, I'll handle the Loaded event for the Page by typing:

Loaded="Page\_Loaded"



The screenshot shows the Visual Studio XAML editor with the following code:

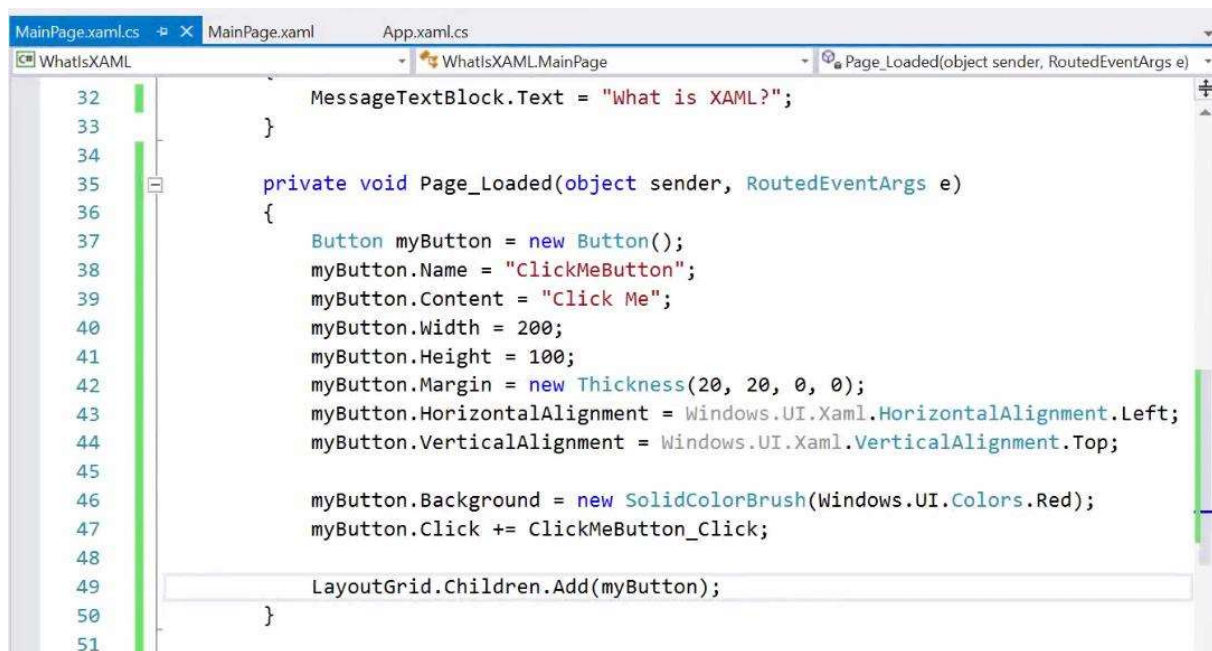
```
5      xmlns:local="using:WhatIsXAML"
6      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8      mc:Ignorable="d" Loaded="Page_Loaded">
9
```

A red arrow points to the `Loaded="Page_Loaded"` attribute in the `mc:Ignorable="d" Loaded="Page_Loaded">` tag.

... inside of the Page tag.

Next, with my mouse cursor on the term Page\_Loaded I press the F12 key on my keyboard which will open the Page's code behind to that event handler's method stub. This event handler will execute whenever the page has loaded into memory.

Next, I'll add C# code that will perform the same function as the XAML that I commented out previously.



```
32     MessageTextBlock.Text = "What is XAML?";
33 }
34
35 private void Page_Loaded(object sender, RoutedEventArgs e)
36 {
37     Button myButton = new Button();
38     myButton.Name = "ClickMeButton";
39     myButton.Content = "Click Me";
40     myButton.Width = 200;
41     myButton.Height = 100;
42     myButton.Margin = new Thickness(20, 20, 0, 0);
43     myButton.HorizontalAlignment = Windows.UI.Xaml.HorizontalAlignment.Left;
44     myButton.VerticalAlignment = Windows.UI.Xaml.VerticalAlignment.Top;
45
46     myButton.Background = new SolidColorBrush(Windows.UI.Colors.Red);
47     myButton.Click += ClickMeButton_Click;
48
49     LayoutGrid.Children.Add(myButton);
50 }
51
```

To verify that functionality has not changed, I run the application in Debug mode.

The point of this exercise is that it took about 12 lines of C# code to do what I was able to do in just one line of XAML code. Admittedly, I spaced the XAML out over several lines, but you can see that the C# version of this is much more verbose.

Second, this provides insight into what's going on as defining elements and attributes in XAML: we're creating new instances of classes in the Universal Windows Platform API and defining and setting their attributes, their properties, just like we're doing here in the C# code.

The important takeaway is this; XAML is simply a way to create instances of classes and set those objects' properties in a much more simplified, succinct syntax.

Furthermore, when using XAML, I get this automatic feedback here in the visual designer. In the preview pane. So I can see the impact of my changes instantly if I choose to work like that.

So in the case of the procedural C# that I wrote, I have to run the application each time that I wanted to see how my tweaks to the code actually worked.

To recap, we learned about the basics and purpose of XAML. First, XAML is just a specific flavor of XML. It follows all the rules of XML. Somebody defines a schema, a contract, and then both the producer and consumer of XML agree to the contract, and then they can begin to work together knowing that they're pretty much on the same page. Now in this case, the contract is XAML and it was defined by Microsoft. The producer of the XML is you, me, in Visual Studio, and then the consumer of the XML is the compiler, which will turn our code in an executable that will run in Windows 10.

Second, XAML is an easy way to create instances of classes and set their properties. And sure, you could do it all in C#, but it's much more verbose and you would lose the design-time tooling that we've become accustomed to in here in two or three lessons.