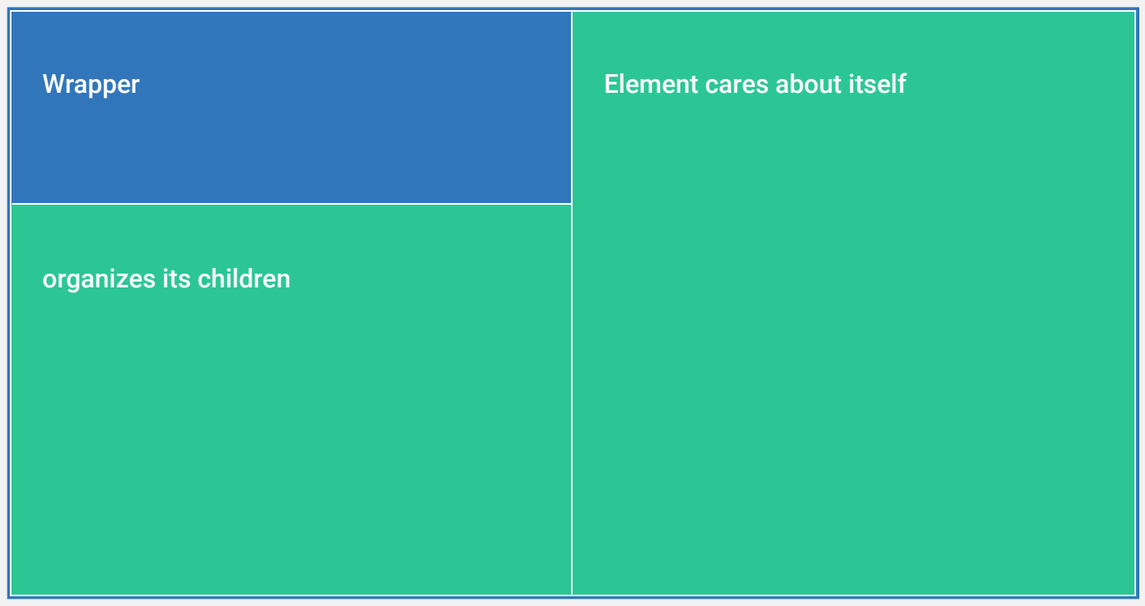
Wrapper / Element

FlexControls brings a piece of XAML and UWP design language features that help you to save time and de-couple your CSS code. You don't need to create an extra CSS file and classes for each page, just add new control instead of HTML element and set the desired properties. It's all done by combining Wrapper and Element controls, which differs in the number of properties.

While **Wrapper** control displays its children in CSS flexbox and contains more properties like *orientation*, *content alignment*, *spacing*, etc.,
Element control cares about its position and other settings in **Wrapper** control.

```
<flex:Wrapper Orientation="Row">

    <flex:Wrapper Orientation="Column"
                  Size="1">
        <flex:Element Size="1">
            <p>Wrapper</p>
        </flex:Element>
        <flex:Element Size="2">
            <p>organizes its children</p>
        </flex:Element>
    </flex:Wrapper>

    <flex:Element Size="1">
        <p>Element cares about itself</p>
    </flex:Element>

</flex:Wrapper>
```

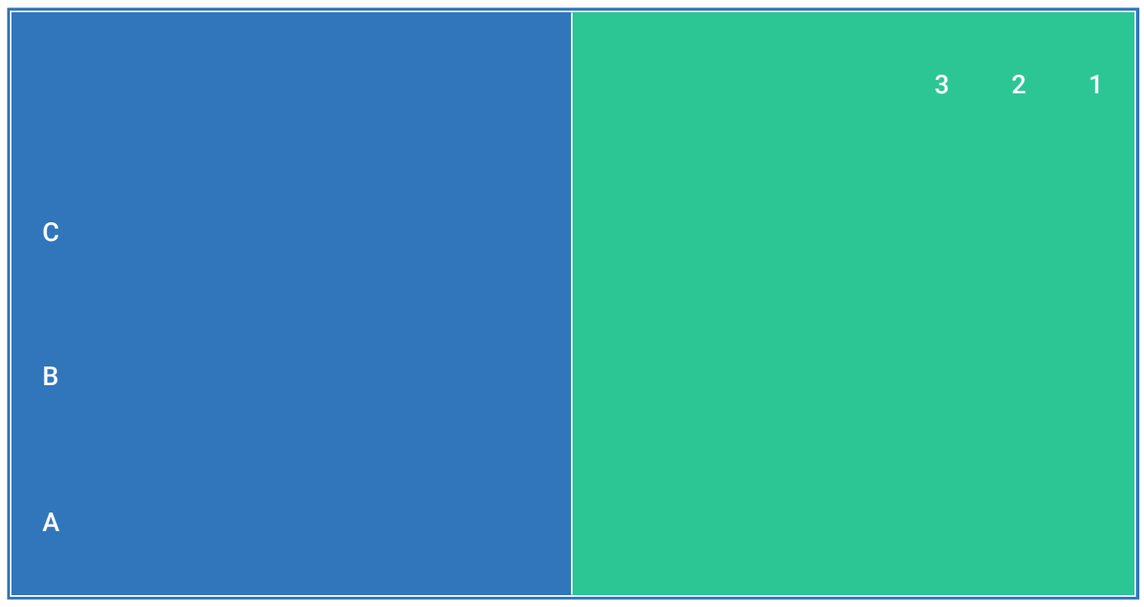


Orientation

To start, we recommend you to draw a layout of your page on paper or a whiteboard and divide it into horizontal and vertical sections.
If you're done, you're ready to start implementing your page layout.

The **Wrapper** is used to display its children in a column (vertically) or a row (horizontally). Add attribute **Orientation** to any Wrapper control and choose with four available values: **Column, ColumnReverse, Row, RowReverse;**

```
<flex:Wrapper Orientation="Row">

    <flex:Wrapper Orientation="ColumnReverse"
                  Size="1">
        <p>A</p>
        <p>B</p>
        <p>C</p>
    </flex:Wrapper>

    <flex:Wrapper Orientation="RowReverse"
                  Size="1">
        <p>1</p>
        <p>2</p>
        <p>3</p>
    </flex:Wrapper>

</flex:Wrapper>
```

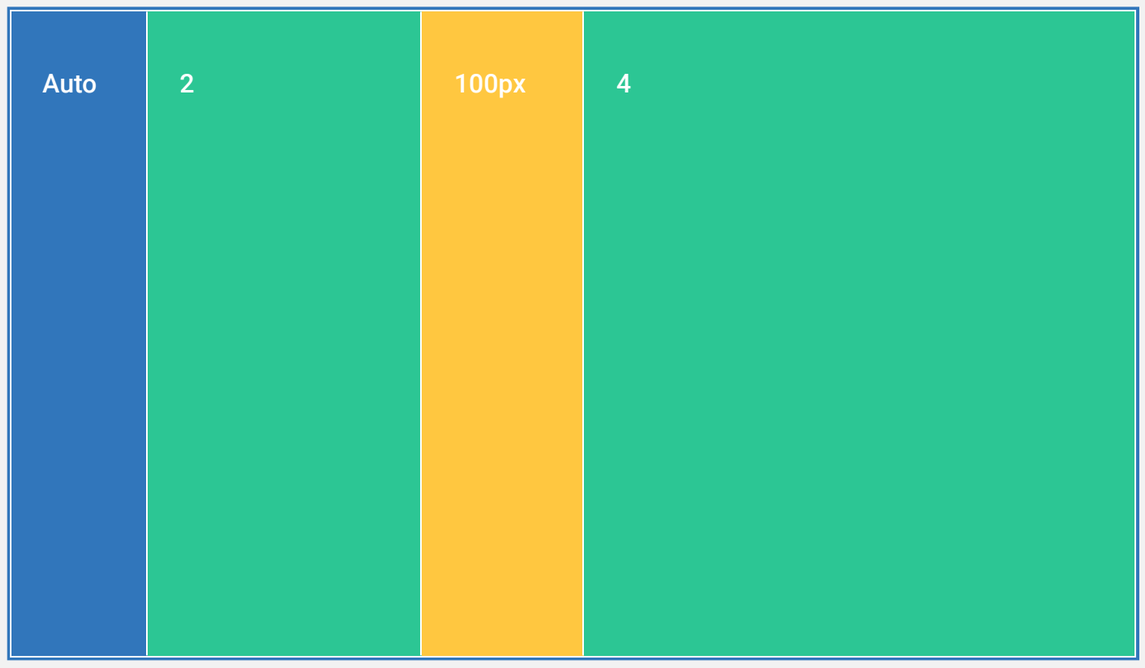


Size

The **Size** property can be applied to any **Element** or **Wrapper** placed inside **Wrapper** control content. You can choose from pre-generated proportional sizes **1-60**, **Auto** (which has the size of its content) or any other custom value like **100px, 20% or 10vh**.

Please note, that if you use any custom value, FlexControls generates additional CSS code into a page's style attribute.

```
<flex:Wrapper Orientation="Row">

    <flex:Element Size="1">
        <p>Auto</p>
    </flex:Element>

    <flex:Element Size="2">
        <p>2</p>
    </flex:Element>

    <flex:Element Size="100px">
        <p>100px</p>
    </flex:Element>

    <flex:Element Size="4">
        <p>4</p>
    </flex:Element>

</flex:Wrapper>
```

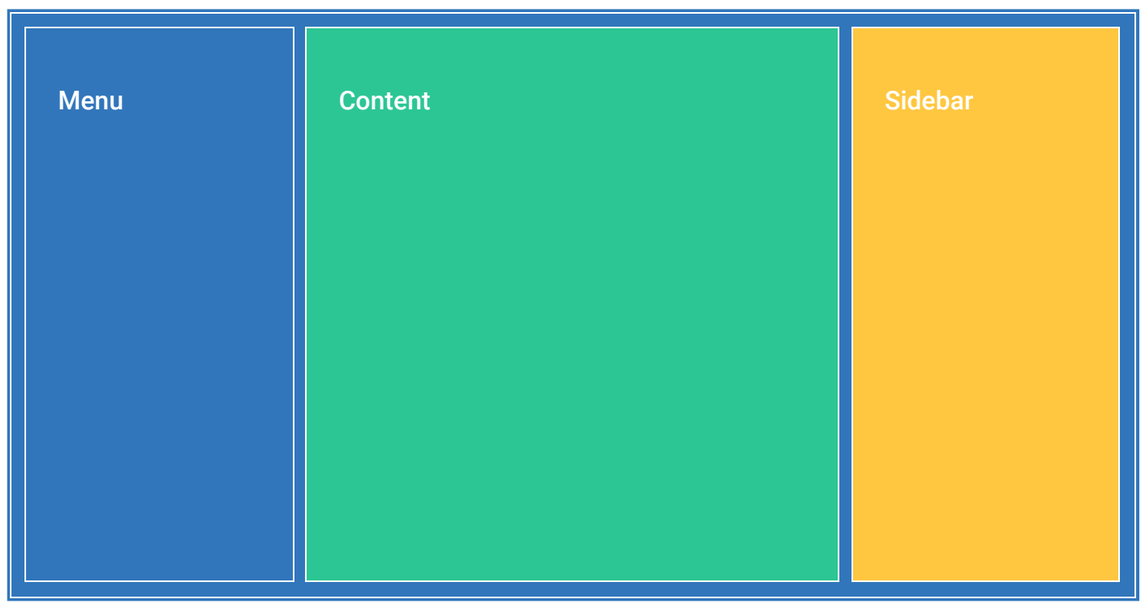


Horizontal & Vertical Spacing

Another cool feature of the **Wrapper** control is the possibility to specify **horizontal** and **vertical spacing** between its items.

You can choose from a range of values between 1-20.

```
<flex:Wrapper Orientation="Row"
              HorizontalSpacing="2"
              VerticalSpacing="2">

    <flex:Element Size="1">
        <p>Menu</p>
    </flex:Element>

    <flex:Element Size="2">
        <p>Content</p>
    </flex:Element>

    <flex:Element Size="1">
        <p>Sidebar</p>
    </flex:Element>

</flex:Wrapper>
```

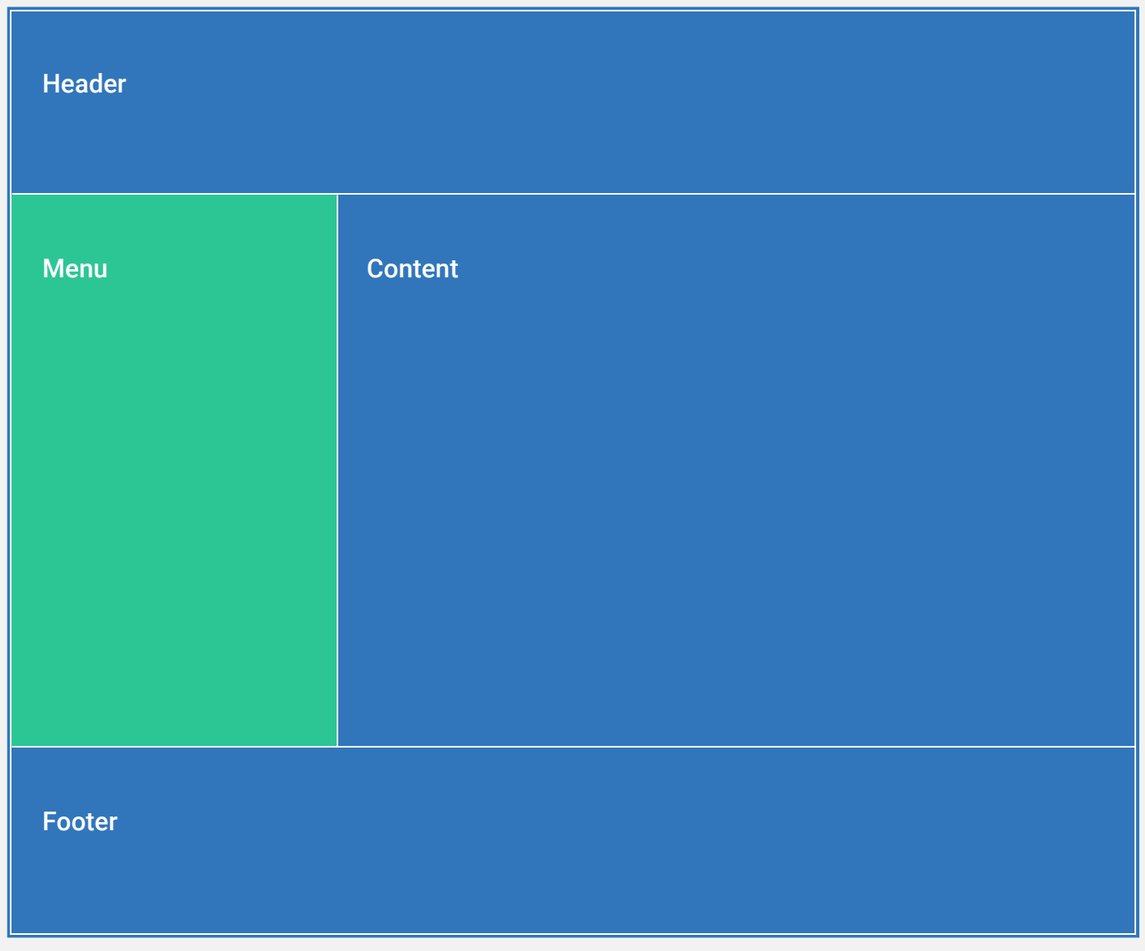


Minimalistic Layout

Let's see some real example. Imagine the basic layout of a web page, which consists of four parts.

With **FlexControls** you can achieve it in less than a minute!

```
<flex:Wrapper Orientation="Column"
              Size="1">

    <!-- Header -->
    <flex:Wrapper Orientation="Row"
                  Size="20%">
        <p>Header</p>
    </flex:Wrapper>

    <!-- Main -->
    <flex:Wrapper Orientation="RowReverse"
                  Size="1">
        <flex:Element Size="1">
            <p>Content</p>
        </flex:Element>

        <flex:Element Size="200px">
            <p>Menu</p>
        </flex:Element>
    </flex:Wrapper>

    <!-- Footer -->
    <flex:Wrapper Orientation="Row"
                  Size="20%">
        <p>Footer</p>
    </flex:Wrapper>

</flex:Wrapper>
```

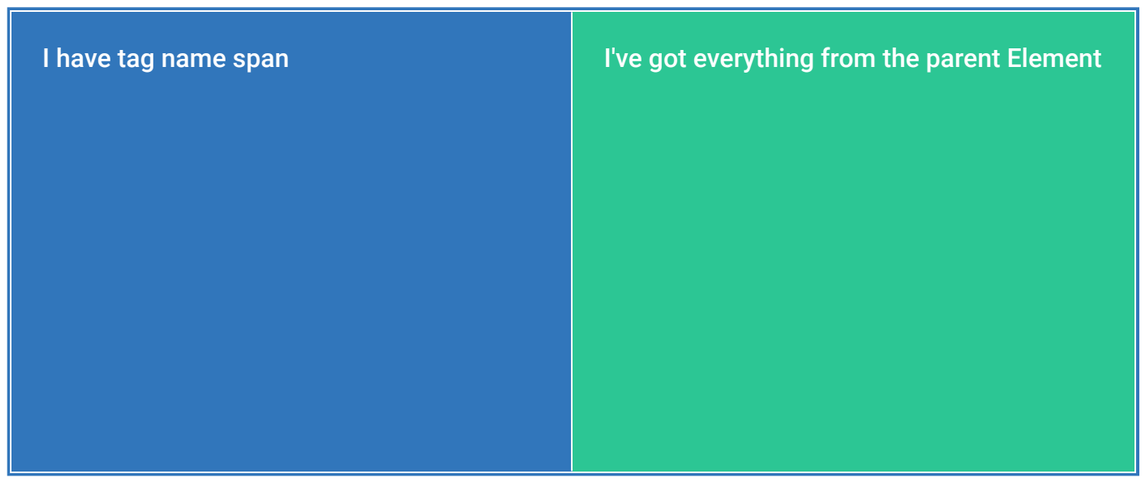


Tag Name

Of course, it would be hard and not reasonable to design the whole page with div elements. Therefore, you can change the element's tag name by setting the property **WrapperTagName**.

However, you still might need to set the flex properties to any DotVVM or custom controls. Because of that, our controls can work like **decorators**. If so, change the **RenderWrapperTag** property value to **false**.

```
<flex:Wrapper Orientation="Row">

    <flex:Element WrapperTagName="Span"
                  Size="1">
        I have tag name span
    </flex:Element>

    <flex:Element RenderWrapperTag="false"
                  Size="1">
        <dot:Literal Text="I've got everything from the parent Element"/>
    </flex:Element>

</flex:Wrapper>
```

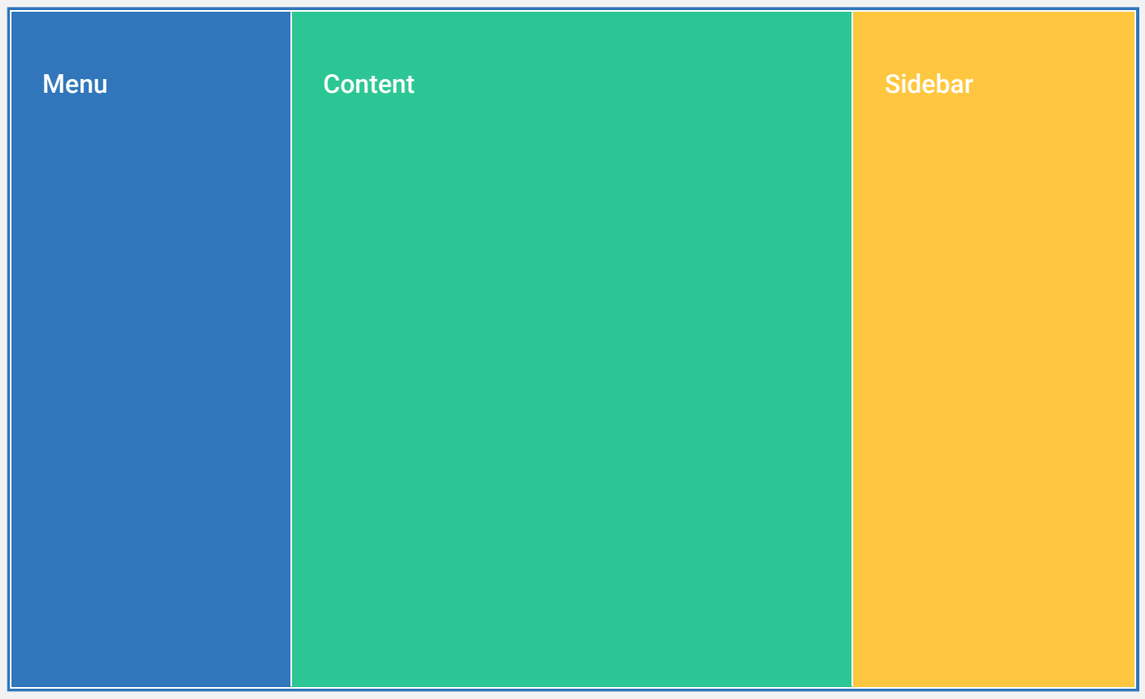


Responsivity

To ensure that your layouts look good on all devices, we've added an option to make those controls responsive.

In case of need, include a collection of **Triggers** for different screen sizes and change any property you need. Currently, both controls support for sizes: **Mobile, Tablet, DesktopSmall, and DesktopLarge**.

```
<flex:Wrapper Orientation="Row">

    <Triggers>
        <flex:MobileWrapper Orientation="Column" />
        <flex:TabletWrapper Orientation="Column" />
    </Triggers>

    <flex:Element Size="1">
        <p>Menu</p>
    </flex:Element>

    <flex:Element Size="2">
        <p>Content</p>
    </flex:Element>

    <flex:Element Size="1">
        <p>Sidebar</p>
    </flex:Element>

</flex:Wrapper>
```

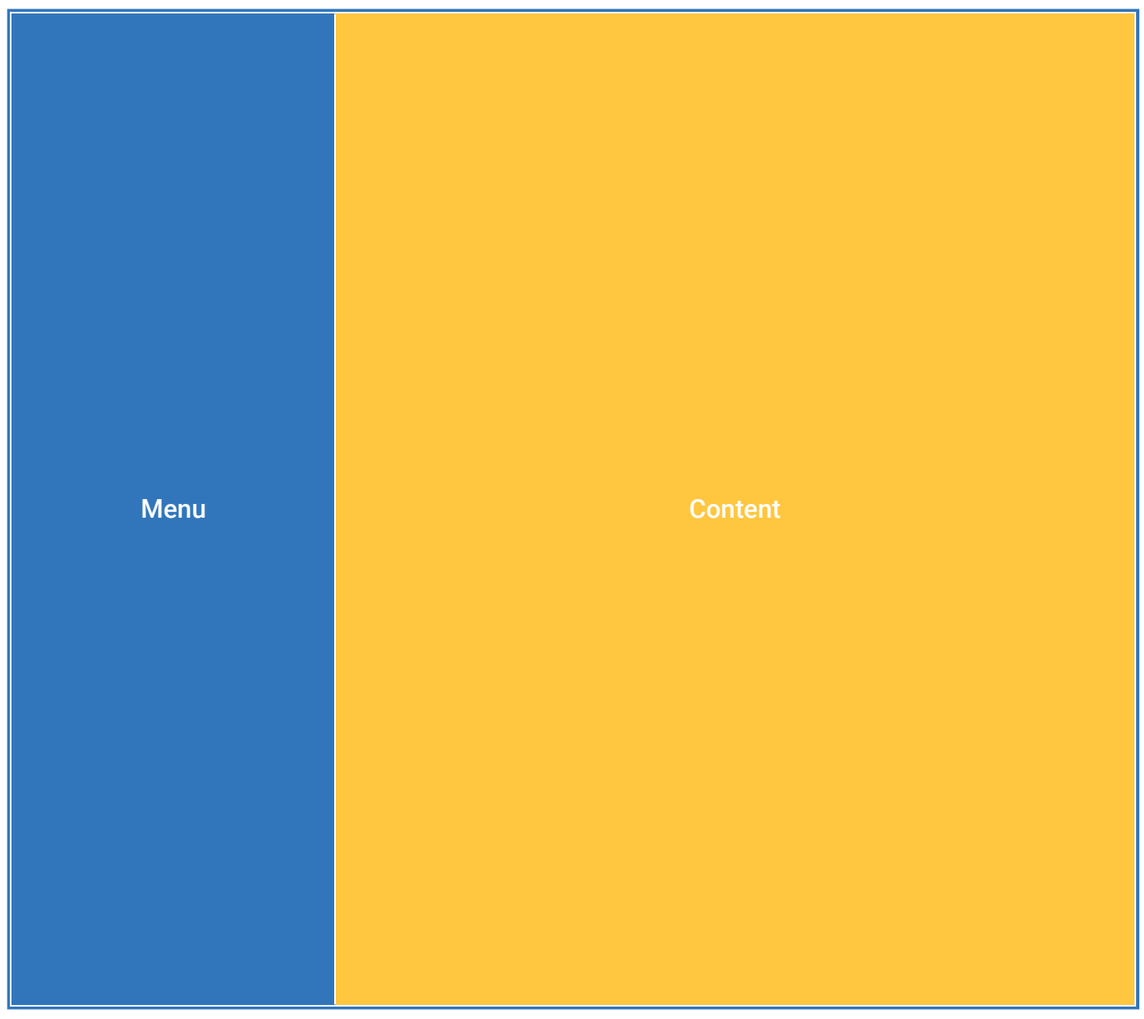


Resources

Another feature you might need is the possibility to **share values and control styles** among the views. Therefore we suggest you specify all your values and styles in static C# classes and bind them with **resource binding**.

```
public static class Variables
{
    public static readonly int SpacingDefault = 1;
    public static readonly string SizeMenu = "200px";
    public static readonly string SizeContent = "1";
}

public static readonly IWrapperStyle MyWrapperStyle = new WrapperStyle()
{
    Orientation = Orientation.Column,
    Size = "1",
    HorizontalContentAlignment = ContentAlignment.Center,
    VerticalContentAlignment = ContentAlignment.Center
};
```

```
<flex:Wrapper Orientation="Row">

    <flex:Wrapper BaseStyle="{resource: Styles.MyWrapperStyle}"
                  Size="{resource: Variables.SizeMenu}">
        <p>Menu</p>
    </flex:Wrapper>

    <flex:Wrapper BaseStyle="{resource: Styles.MyWrapperStyle}"
                  Size="{resource: Variables.SizeContent}">
        <p>Content</p>
    </flex:Wrapper>

</flex:Wrapper>
```

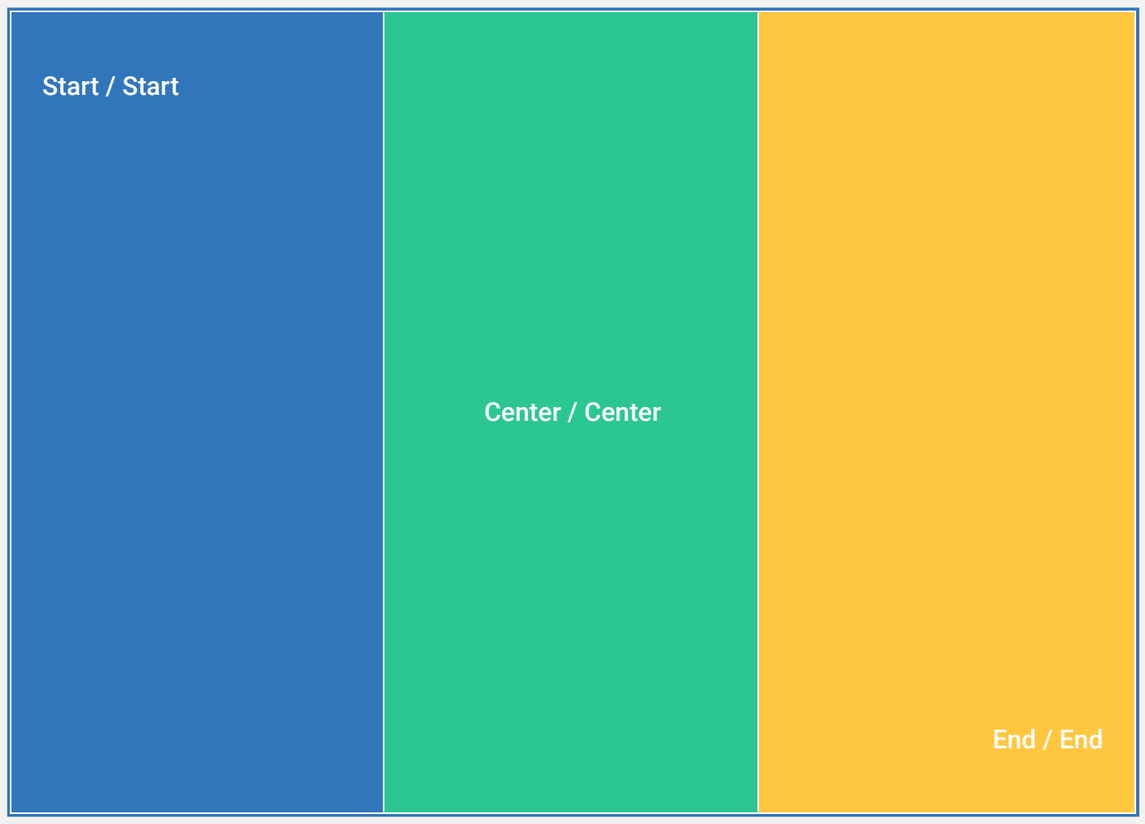


Horizontal & Vertical Content Alignment

The **Wrapper** control also offers the possibility to align its content horizontally and vertically.

To do so, use the properties **HorizontalContentAlignment** or **VerticalContentAlignment**.

```
<flex:Wrapper Orientation="Row">

    <flex:Wrapper Orientation="Column"
                  Size="1"
                  HorizontalContentAlignment="FlexStart"
                  VerticalContentAlignment="FlexStart">
        <p>Start / Start</p>
    </flex:Wrapper>

    <flex:Wrapper Orientation="Column"
                  Size="1"
                  HorizontalContentAlignment="Center"
                  VerticalContentAlignment="Center">
        <p>Center / Center</p>
    </flex:Wrapper>

    <flex:Wrapper Orientation="Column"
                  Size="1"
                  HorizontalContentAlignment="FlexEnd"
                  VerticalContentAlignment="FlexEnd">
        <p>End / End</p>
    </flex:Wrapper>

</flex:Wrapper>
```

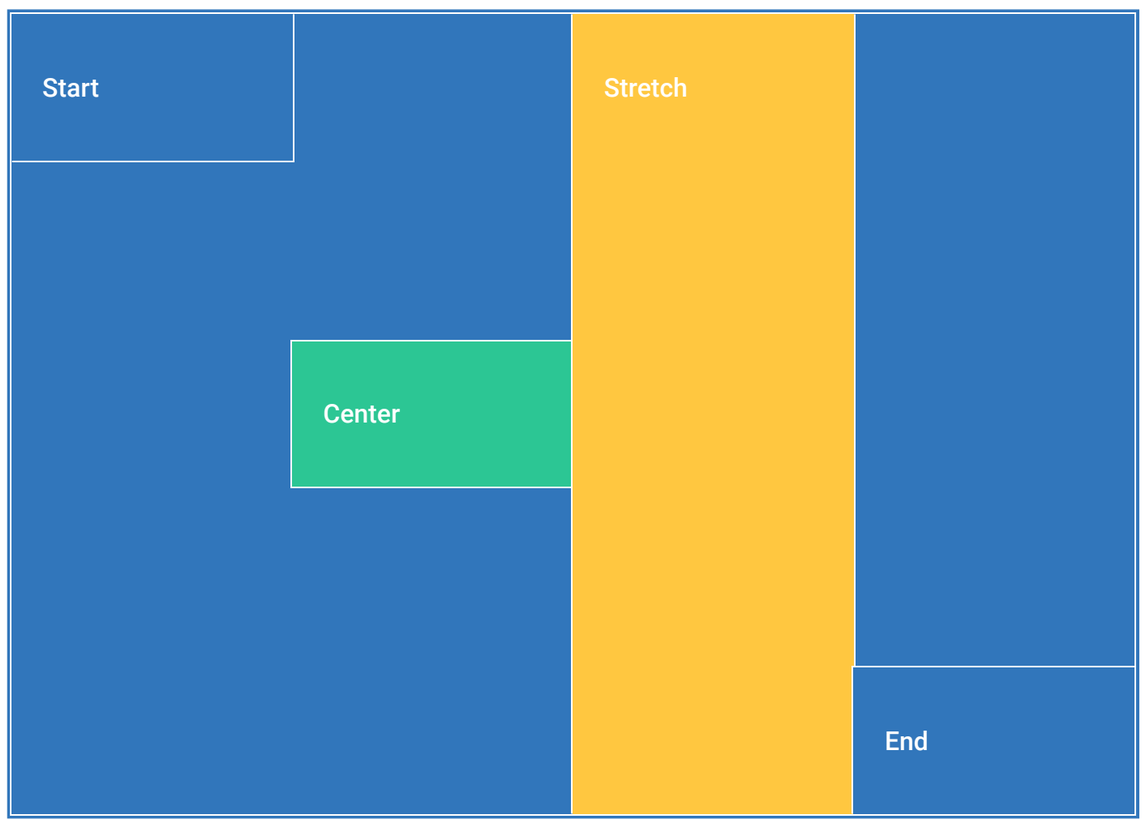


Alignment

The setting of content alignment is applied to all its children. However, if you wish, there is a possibility to change the cross-axis alignment of any children.

In that case, use the property **Alignment** and select from its values: **FlexStart, FlexEnd, Center, Baseline, Stretch.**

```
<flex:Wrapper Orientation="Row"
              HorizontalContentAlignment="Center">

    <flex:Element Size="1"
                  Alignment="FlexStart">
        <p>Start</p>
    </flex:Element>

    <flex:Element Size="1"
                  Alignment="Center">
        <p>Center</p>
    </flex:Element>

    <flex:Element Size="1"
                  Alignment="Stretch">
        <p>Stretch</p>
    </flex:Element>

    <flex:Element Size="1"
                  Alignment="FlexEnd">
        <p>End</p>
    </flex:Element>

</flex:Wrapper>
```

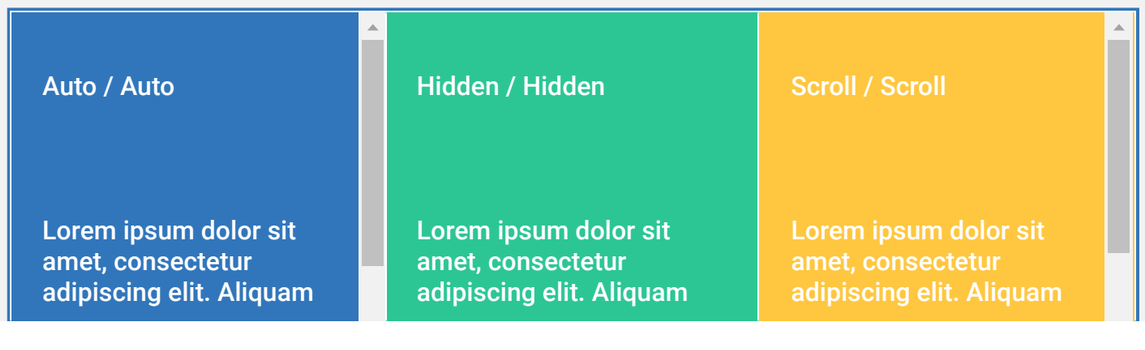


Horizontal & Vertical Overflow

When you divide your screen in any ratio, content of its parts can be larger than the currently visible area.

Use the **HorizontalOverflow** and **VerticalOverflow** properties to specify the behavior of horizontal and vertical overflow.

```
<%-- Row --%>
<flex:Wrapper Orientation="Row"
              VerticalOverflow="Hidden">


    <flex:Wrapper Orientation="Column"
                  Size="1"
                  HorizontalOverflow="Auto"
                  VerticalOverflow="Auto">
```



```
        <p>Auto / Auto</p>
        <p>{{controlProperty: LoremIpsum}}</p>
    </flex:Wrapper>

    <flex:Wrapper Orientation="Column"
                  Size="1"
                  HorizontalOverflow="Hidden"
                  VerticalOverflow="Hidden">
        <p>Hidden / Hidden</p>
        <p>{{controlProperty: LoremIpsum}}</p>
    </flex:Wrapper>

    <flex:Wrapper Orientation="Column"
                  Size="1"
                  HorizontalOverflow="Scroll"
                  VerticalOverflow="Scroll">
        <p>Scroll / Scroll</p>
        <p>{{controlProperty: LoremIpsum}}</p>
    </flex:Wrapper>

</flex:Wrapper>
```

vestibulum tempus nibh at pellentesque. Phasellus nec venenatis turpis, at laoreet velit. Aenean ex mauris, egestas ac tempus id, lacinia ultricies lectus. Cras congue erat mauris, euismod pellentesque dolor pretium ac. Integer volutpat vestibulum justo. Vestibulum molestie urna ut vulputate vehicula. Nunc accumsan erat vel elit commodo gravida. Ut dapibus tempor erat, non ornare mauris dictum nec. Duis vel tempus eros. Nunc a lacinia erat. Maecenas

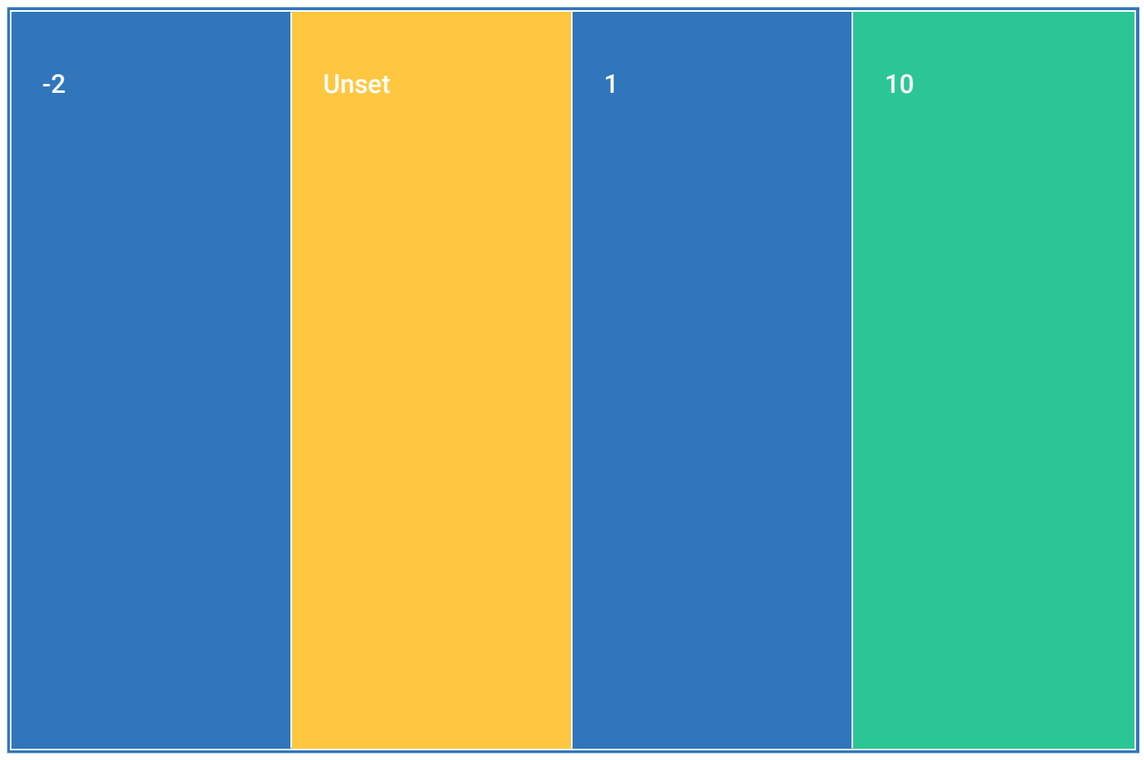
vestibulum tempus nibh at pellentesque. Phasellus nec venenatis turpis, at laoreet velit. Aenean ex mauris, egestas ac tempus id, lacinia ultricies lectus. Cras congue erat mauris, euismod pellentesque dolor pretium ac. Integer volutpat vestibulum justo. Vestibulum molestie urna ut vulputate vehicula. Nunc accumsan erat vel elit commodo gravida. Ut dapibus tempor erat, non ornare mauris dictum nec. Duis vel tempus eros. Nunc a lacinia erat. Maecenas sit amet volutpat leo. Suspendisse bibendum a diam in

vestibulum tempus nibh at pellentesque. Phasellus nec venenatis turpis, at laoreet velit. Aenean ex mauris, egestas ac tempus id, lacinia ultricies lectus. Cras congue erat mauris, euismod pellentesque dolor pretium ac. Integer volutpat vestibulum justo. Vestibulum molestie urna ut vulputate vehicula. Nunc accumsan erat vel elit commodo gravida. Ut dapibus tempor erat, non ornare mauris dictum nec. Duis vel tempus eros. Nunc a

Order

By default, children of **Wrapper** control are ordered by its orientation and default order. In case you need to reorder them, you can set the **Order** property to any pre-generated value **between -20 and 20**, or any other custom value.

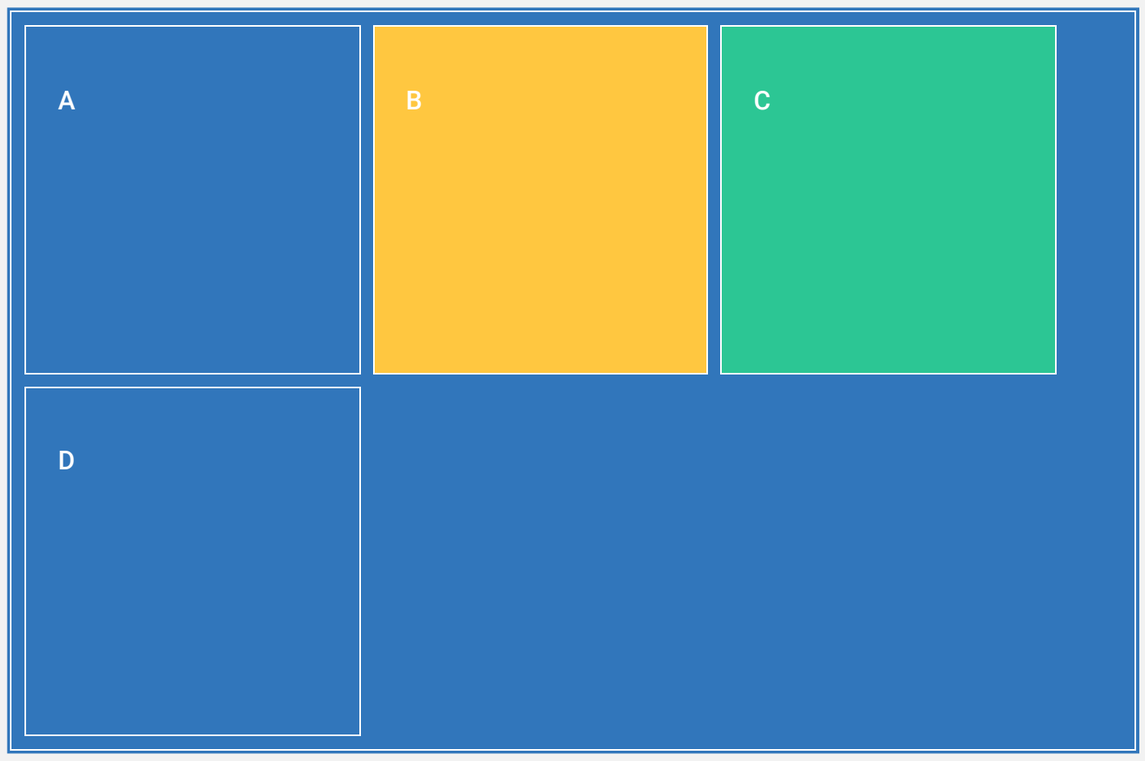
```
<flex:Wrapper Orientation="Row">

    <flex:Element Size="1"
                  Order="1">
        <p>1</p>
    </flex:Element>

    <flex:Element Size="1"
                  Order="10">
        <p>10</p>
    </flex:Element>

    <flex:Element Size="1">
        <p>Unset</p>
    </flex:Element>

    <flex:Element Size="1"
                  Order="-2">
        <p>-2</p>
    </flex:Element>

</flex:Wrapper>
```



Wrap

In order to handle content overflow, there is a possibility to specify the **Wrap** property to either force content display in a single row/column or allow to wrap its children and be shown on multiple rows/columns.

```
<flex:Wrapper Orientation="Row"
              Wrap="Wrap"
              HorizontalSpacing="2"
              VerticalSpacing="2">

    <flex:Element Size="30%">
        <p>A</p>
    </flex:Element>

    <flex:Element Size="30%">
        <p>B</p>
    </flex:Element>

    <flex:Element Size="30%">
        <p>C</p>
    </flex:Element>

    <flex:Element Size="30%">
        <p>D</p>
    </flex:Element>

</flex:Wrapper>
```

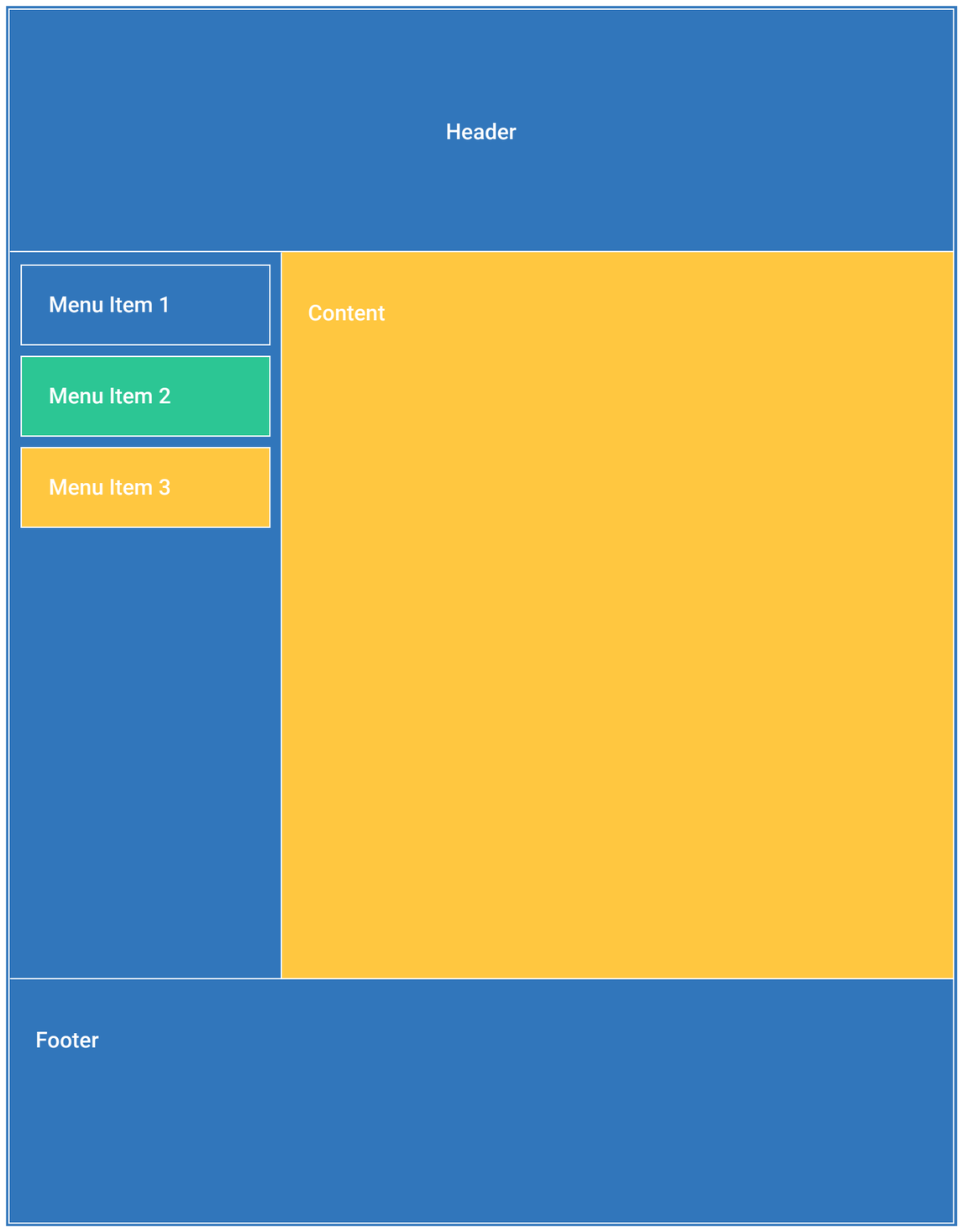


Basic Layout

Let's see some real example. Imagine the basic layout of a web page, which consists of four parts.

With **FlexControls** you can achieve it in less than a minute!

```
<flex:Wrapper Orientation="Column"
              Size="1">

    <!-- Header -->
    <flex:Wrapper Orientation="Row"
                  Size="20%"
                  WrapperTagName="Header"
                  HorizontalContentAlignment="Center" VerticalContentAlignmen
        <p>Header</p>
    </flex:Wrapper>

    <!-- Main -->
    <flex:Wrapper Orientation="Row"
                  Size="1"
                  WrapperTagName="Main">

        <!-- Menu -->
        <flex:Wrapper Orientation="Column"
                      Size="200px"
                      HorizontalSpacing="2" VerticalSpacing="2"
                      RenderWrapperTag="false">
            <dot:Repeater DataSource="{controlProperty: MenuItems}"
                          WrapperTagName="Nav">
                <flex:Element RenderWrapperTag="false">
                    <p>Menu Item {{value: _this}}</p>
                </flex:Element>
            </dot:Repeater>
        </flex:Wrapper>

        <!-- Content -->
        <flex:Element Size="1">
            <p>Content</p>
        </flex:Element>

    </flex:Wrapper>

    <!-- Footer -->
    <flex:Wrapper Orientation="Row"
                  Size="20%"
                  WrapperTagName="Footer">
        <p>Footer</p>
    </flex:Wrapper>

</flex:Wrapper>
```



www.dotvvm.com

Documentation | **Tutorials** | **Samples**