

Counter with Parameter

```
@code {  
    [Fact]  
    public void IncrementCounterByOneWhenButtonClicked()  
    {  
        // Render our component including all lifecycle events  
        var cut = Render(@<Counter CurrentCount="5" />);  
  
        // Find the button via CSS selector and click it  
        IElement button = cut.Find("button");  
        button.Click();  
  
        // Assert the markup  
        var pElement = cut.Find(".some-text");  
        pElement.TextContent.Should().Be("Current count: 6");  
        pElement.MarkupMatches(@<p style:ignore>Current count: 6</p>);  
    }  
}
```

```
<h1>Counter</h1>  
  
<p class="some-text">Current count: @CurrentCounter</p>  
  
<button id="increment-button"  
    class="btn btn-primary"  
    @onclick="IncrementCount">Click me</button>  
  
@code {  
    [Parameter]  
    public int CurrentCounter { get; set; }  
  
    private void IncrementCount()  
    {  
        CurrentCounter++;  
    }  
}
```

➤ Super convenient way of passing parameters, CascadingParameters, Events, ...

Conclusion

- Try to test semantic not structure.
- If your structure changes your test becomes invalid as well.
 - Well, that's bad and defeats the purpose.