Arbitrary attributes and parameters in Blazor

In this article we will discuss **how a component can accept arbitrary attributes**. This is continuation to our previous article [Attribute Splatting](https://www.pragimtech.com/blog/blazor/blazor-attribute-splatting/).

Child Component (ChildComponent.razor)

Dictionary parameter InputAttributes is applied to the <input> element using @attributes razor directive. This InputAttributes parameter allows a parent component to be able to pass arbitrary attributes.

<input id="firstName" @attributes="InputAttributes" />

@code {

[Parameter]

public Dictionary<string, object> InputAttributes { get; set; } =

new Dictionary<string, object>()

{

{ "placeholder", "Child Component Placeholder" }

};

}

Copy

Parent Component (ParentComponent.razor)

The parent component is passing 2 attributes (required and placeholder) to the child component using the dictionary parameter InputAttributes. Using this technique, you can pass any number of arbitrary attributes from the parent component to child component.

<ChildComponent InputAttributes="attributesFromParent">

</ChildComponent>

@code {

public Dictionary<string, object> attributesFromParent { get; set; } =

new Dictionary<string, object>()

{

{ "required", "required" },

{ "placeholder", "Parent Component Placeholder" }

};

}

Copy

However, what you cannot do is specify the arbitrary attributes in the HTML. In this example maxlength attribute is included in the HTML. This causes InvalidOperationException: Object of type 'EmployeeManagement.Web.Pages.ChildComponent' does not have a property matching the name 'maxlength'. One way to fix this is to include a corresponding parameter in the child component. It's tedious to include a parameter for every attribute especially if we want to support large number of attributes.

<ChildComponent InputAttributes="attributesFromParent" maxlength="15">

</ChildComponent>

@code {

public Dictionary<string, object> attributesFromParent { get; set; } =

new Dictionary<string, object>()

{

{ "required", "required" },

{ "placeholder", "Parent Component Placeholder" }

};

}

Copy

CaptureUnmatchedValues Property

To be able to capture attribute values that do not have matching parameters, simply set CaptureUnmatchedValues property of the dictionary parameter to true.

Child Component (ChildComponent.razor)

<input id="firstName" @attributes="InputAttributes" />

@code {

[Parameter(CaptureUnmatchedValues = true)]

public Dictionary<string, object> InputAttributes { get; set; } =

new Dictionary<string, object>()

{

{ "placeholder", "Child Component Placeholder" }

};

}

Copy

Parent Component (ParentComponent.razor)

The parent component is passing 3 arbitrary attributes - maxlength, placeholder and required.

<ChildComponent maxlength="15" placeholder="Parent Component Placeholder" required>

</ChildComponent>

Copy

When CaptureUnmatchedValues property is set to true, we cannot explicitly set the value for the dictionary property in the parent component. If we try to do that, we get the following exception.

InvalidOperationException: The property 'InputAttributes' on component type 'EmployeeManagement.Web.Pages.ChildComponent' cannot be set explicitly when also used to capture unmatched values.

Default attribute values

If required, the child component can specify default values. However, the order in which the default values and @attributes directive are applied matters.

Child Component (ChildComponent.razor)

In this example, a default value for the placeholder attribute is specified by the child component. This default value is present on the left hand side of the @attributes directive.

<input id="firstName" placeholder="Default Placeholder" @attributes="InputAttributes" />

@code {

[Parameter(CaptureUnmatchedValues = true)]

public Dictionary<string, object> InputAttributes { get; set; } =

new Dictionary<string, object>();

}

Copy

Parent Component (ParentComponent.razor)

Parent component did not explicitly pass a value for the placeholder attribute, so the child component default placeholder value will be used.

<ChildComponent></ChildComponent>

Copy

In the example below, parent component is passing a value for the placeholder attribute, so this will override the child component default placeholder value as expected.

<ChildComponent placeholder="Parent Component Placeholder">

</ChildComponent>

Copy

Now, in the child component, reverse the @attributes directive and the placeholder default value.

Child Component (ChildComponent.razor)

The attributes are always processed from right to left, so the default placeholder attribute value specified in the child component is always used, even if the parent component passes an explicit value for the placeholder attribute.

<input id="firstName" @attributes="InputAttributes" placeholder="Default Placeholder"/>

@code {

[Parameter(CaptureUnmatchedValues = true)]

public Dictionary<string, object> InputAttributes { get; set; } =

new Dictionary<string, object>();

}

Copy

Parent Component (ParentComponent.razor)

Parent Component has specified an explicit value for the placeholder attribute, but this will be overriden by the child component.

<ChildComponent placeholder="Parent Component Placeholder">

</ChildComponent>