# Tag Helpers

Tag Helpers enable server-side code to participate in creating and rendering HTML elements in Razor files. There are many built-in Tag Helpers for common tasks - such as creating forms, links, loading assets and more - and even more available in public GitHub repositories and as NuGet packages. Tag Helpers are authored in C#, and they target HTML elements based on element name, attribute name, or parent tag. For example, the built-in LabelTagHelper can target the HTML <label> element when the LabelTagHelper attributes are applied. If you're familiar with HTML Helpers, Tag Helpers reduce the explicit transitions between HTML and C# in Razor views. In many cases, HTML Helpers provide an alternative approach to a specific Tag Helper, but it's important to recognize that Tag Helpers don't replace HTML Helpers and there's not a Tag Helper for each HTML Helper.

Tag Helpers scope is controlled by a combination of @addTagHelper, @removeTagHelper, and the "!" opt-out character. The @addTagHelper directive makes Tag Helpers available to the view. In this case, the view file is Pages/\_ViewImports.cshtml, which by default is inherited by all files in the Pages folder and sub-folders; making Tag Helpers available. The code above uses the wildcard syntax ("\*") to specify that all Tag Helpers in the specified assembly (Microsoft.AspNetCore.Mvc.TagHelpers) will be available to every view file in the Views directory or sub-directory. The first parameter after @addTagHelper specifies the Tag Helpers to load (we are using "\*" for all Tag Helpers), and the second parameter "Microsoft.AspNetCore.Mvc.TagHelpers" specifies the assembly containing the Tag Helpers. Microsoft.AspNetCore.Mvc.TagHelpers is the assembly for the built-in ASP.NET Core Tag Helpers.

@using AuthoringTagHelpers

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

@addTagHelper \*, AuthoringTagHelpers

@addTagHelper AuthoringTagHelpers.TagHelpers.E\*, AuthoringTagHelpers

@addTagHelper AuthoringTagHelpers.TagHelpers.Email\*, AuthoringTagHelpers

The @removeTagHelper has the same two parameters as @addTagHelper, and it removes a Tag Helper that was previously added. For example, @removeTagHelper applied to a specific view removes the specified Tag Helper from the view. Using @removeTagHelper in a Views/Folder/\_ViewImports.cshtml file removes the specified Tag Helper from all of the views in Folder.

You can disable a Tag Helper at the element level with the Tag Helper opt-out character ("!"). For example, Email validation is disabled in the <span> with the Tag Helper opt-out character:

<!span asp-validation-for="Email" class="text-danger"></!span>

You must apply the Tag Helper opt-out character to the opening and closing tag. (The Visual Studio editor automatically adds the opt-out character to the closing tag when you add one to the opening tag). After you add the opt-out character, the element and Tag Helper attributes are no longer displayed in a distinctive font.

### Using @tagHelperPrefix to make Tag Helper usage explicit

The @tagHelperPrefix directive allows you to specify a tag prefix string to enable Tag Helper support and to make Tag Helper usage explicit. For example, you could add the following markup to the Views/\_ViewImports.cshtml file:

CSHTMLCopy

@tagHelperPrefix th:

In the code image below, the Tag Helper prefix is set to th:, so only those elements using the prefix th: support Tag Helpers (Tag Helper-enabled elements have a distinctive font). The <label> and <input> elements have the Tag Helper prefix and are Tag Helper-enabled, while the <span> element doesn't.

**Custom Tag Helper**

public class NoFollowTagHelper : TagHelper

{

// Public properties becomes available on our custom tag as an attribute.

public string Href { get; set; }

public override void Process(TagHelperContext context, TagHelperOutput output)

{

output.TagName = "a"; // Specify our tag output name

output.TagMode = TagMode.StartTagAndEndTag; // The type of tag we wish to create

output.Attributes["href"] = Href;

if (!output.Attributes["href"].Value.ToString().Contains("josephwoodward.co.uk"))

{

output.Attributes["rel"] = "nofollow";

}

base.Process(context, output);

}

}