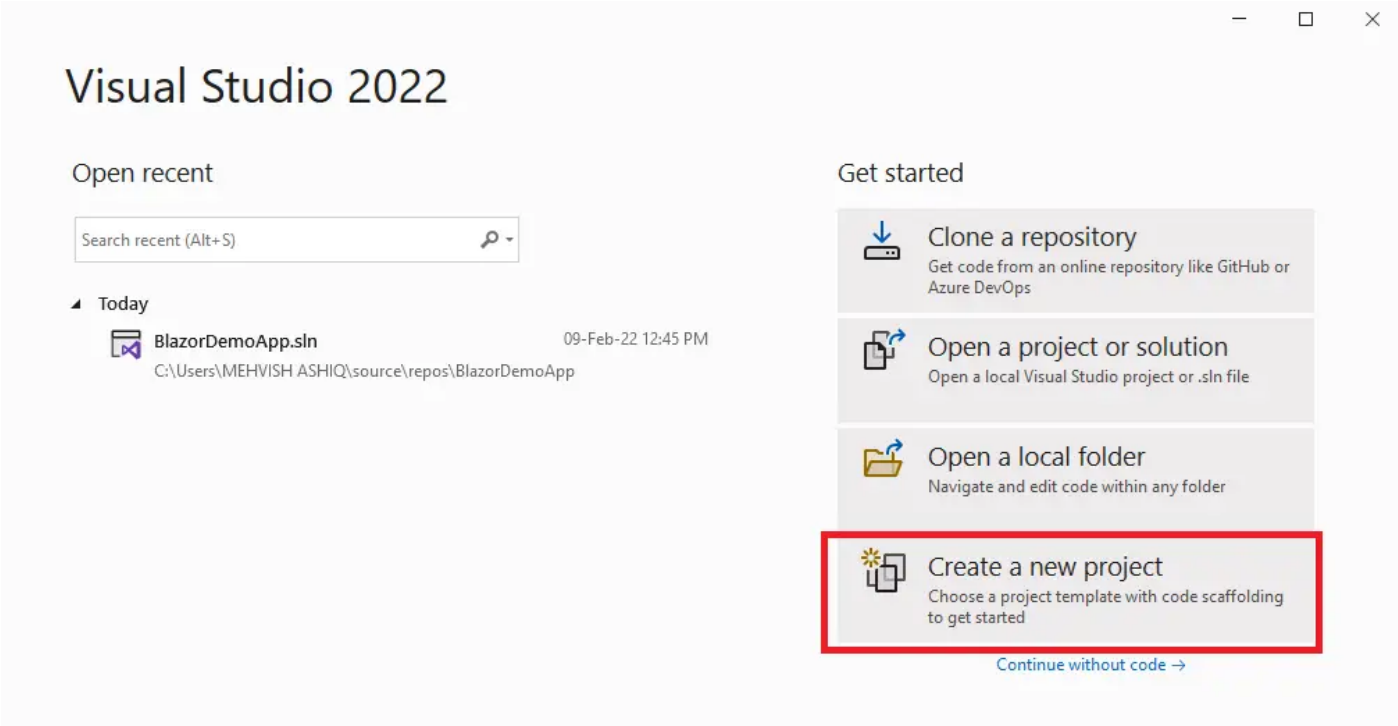
从 JavaScript 调用 C# 函数

Mehvish Ashiq



本教程重点介绍如何使用 Blazor 从 JavaScript 代码调用 C# 函数。

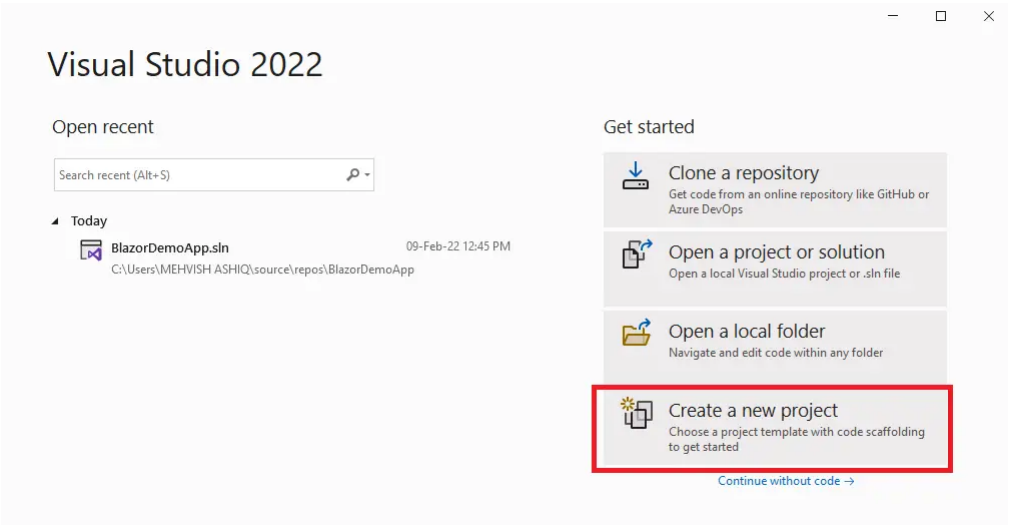
Blazor 是 Browser & Razor 的变种，它是一个开源和免费的 Web 框架，允许我们使用 C# 开发交互式 Web 用户界面 (UI)。

我们使用 Microsoft Visual Studio 制作 Blazor 应用程序；你也可以从这里下载。

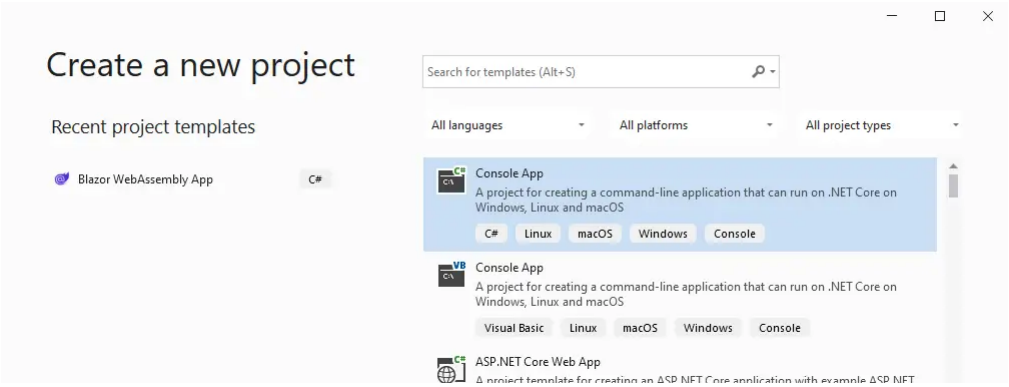
使用 Blazor 从 JavaScript 代码调用 C# 函数

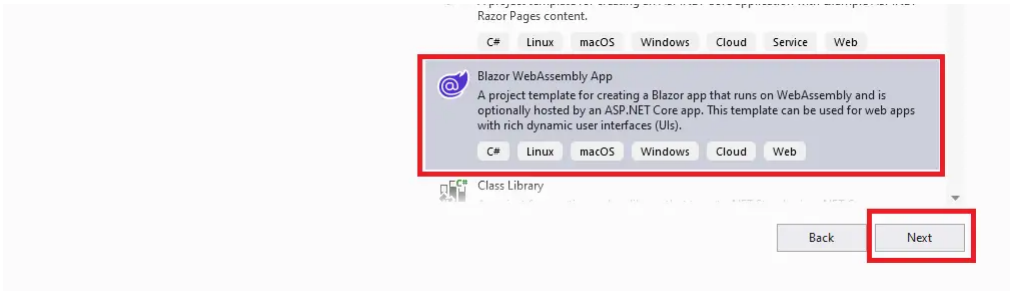
让我们创建一个 Blazor 应用程序来逐步开始编码。

打开 Microsoft Visual Studio 并选择创建新项目。

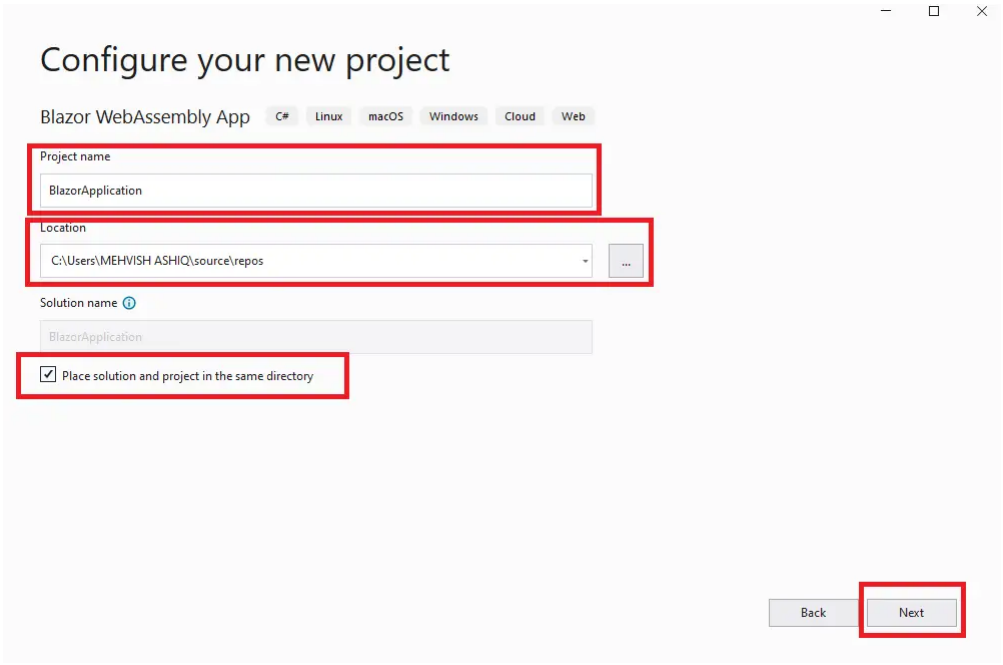


选择 `Blazor WebAssembly` 应用，然后单击 下一步。



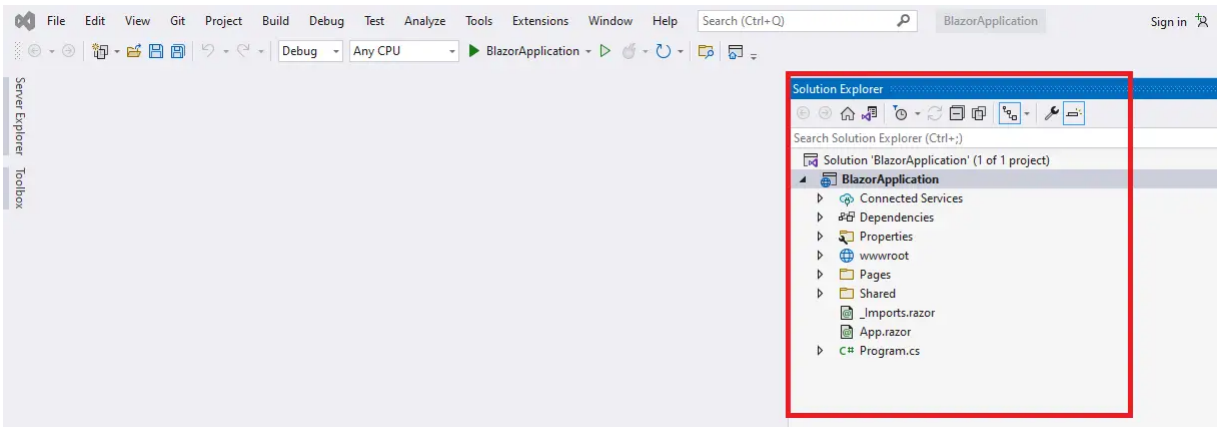


写下你的项目名称，选择要保存的位置，选中复选框以将解决方案保留在同一目录中，然后单击下一步。



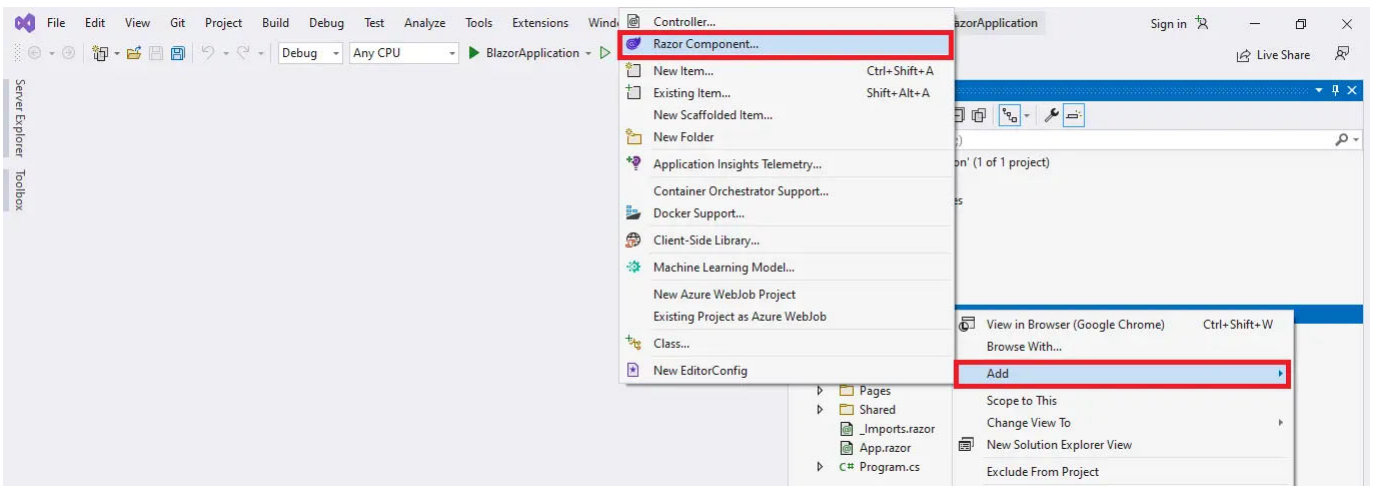
在下一个屏幕上保持默认设置，然后单击创建。它将为我们创建默认文件和文件夹。

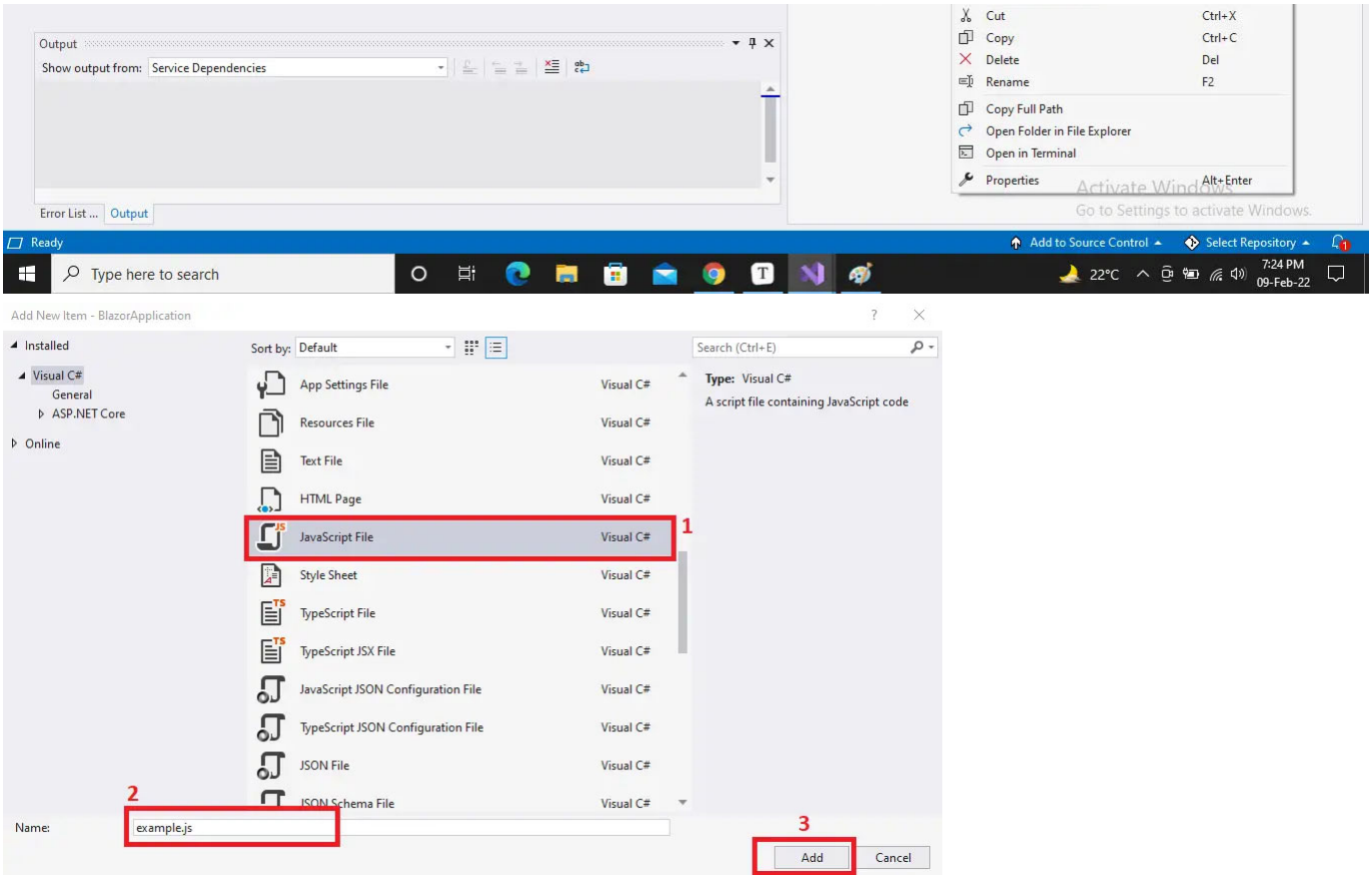
请参阅下面给出的屏幕截图。



在 `wwwroot` 中创建一个名为 `scripts` 的新文件夹。右键单击 `scripts` 文件夹并转到 `Add->Razor Component` 创建一个 JavaScript 文件。

它打开一个新窗口，选择 `JavaScript File`，将其名称写为 `example.js`，然后单击 `Add`。

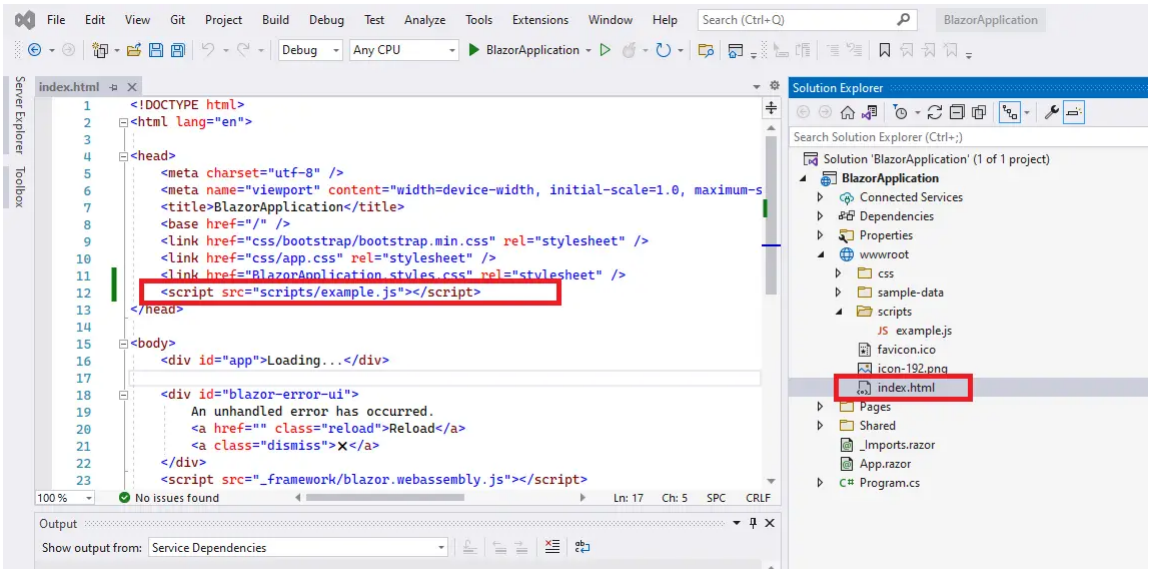




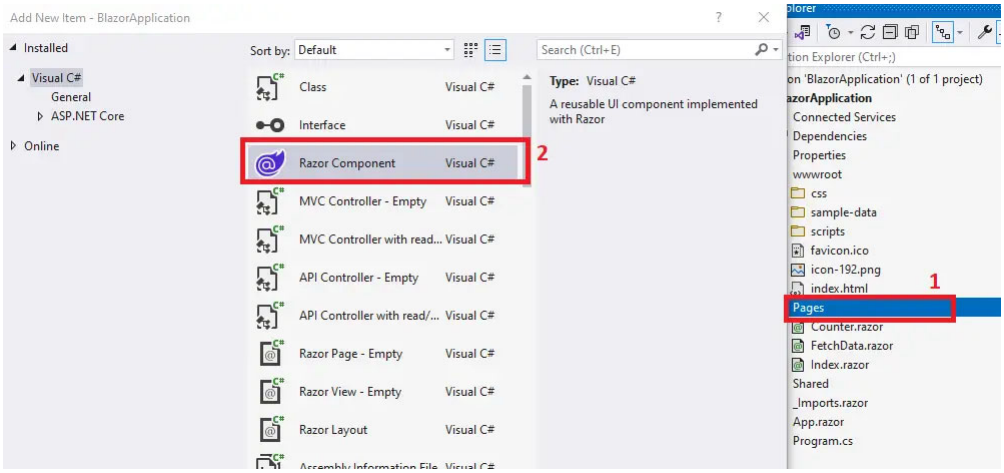
使用以下代码行将 JavaScript 文件附加到 `index.html`（它位于 `wwwroot`）。

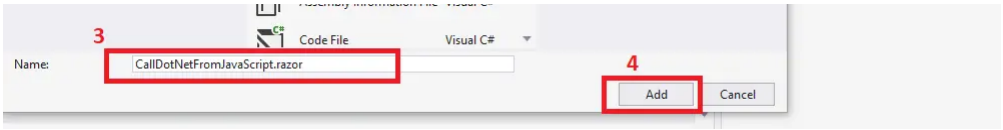
javascriptCopy

```
<script src="scripts/example.js"></script>
```

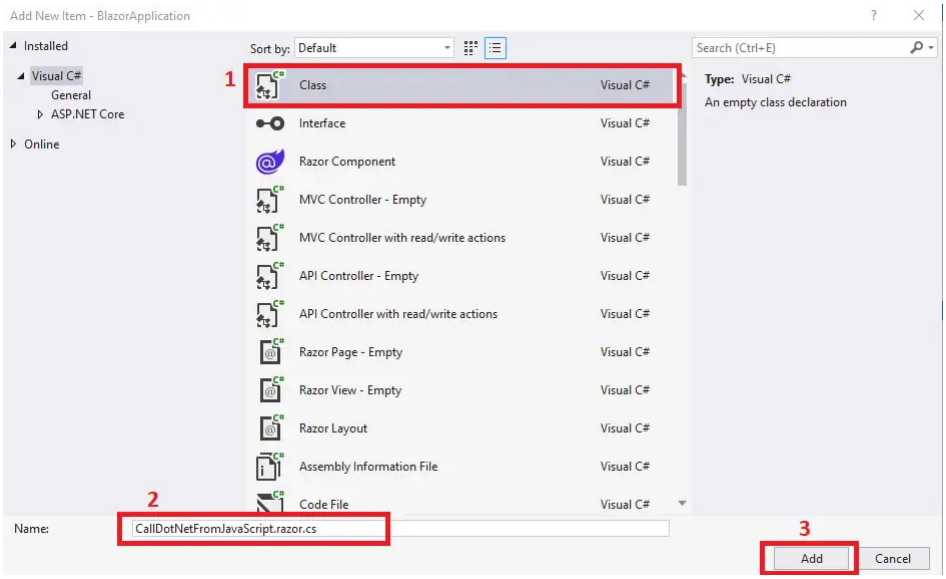


右键单击 `Pages` 文件夹并转到 `Add->Razor` 组件。从弹出的窗口中选择 `Razor Component`，将其命名为 `CallDotNetFromJavaScript.razor`，然后单击 `Add`。

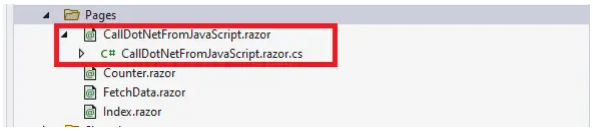




右键单击 Pages 文件夹并再次转到 Add->Razor Component，但这次选择 Class，将其命名为 CallDotNetFromJavaScript.razor.cs，然后单击 Add。



现在，你可以在 Pages 文件夹中看到两个文件 CallDotNetFromJavaScript.razor 和 CallDotNetFromJavaScript.razor.cs，如下所示。



现在，通过添加以下代码来修改 CallDotNetFromJavaScript.razor 文件。

javascriptCopy

```
@page "/dotnetinjs"
<h1> Learn How to Call DotNet From JavaScript Code</h1>
<br />

<div class="row">
        <div class="col-md-4">
                <h4>Call static method from JS</h4>
        </div>
        <div class="col-md-2">
                <button type="button" class="btn btn-success" onclick="jsMethods.checkNumber()">
           Check Number
                </button>
        </div>
        <div class="col-md-4">
                <span id="string-result" class="form-text"></span>
        </div>
</div>
<hr />
```

通过添加以下代码修改 Index.razor 文件。

javascriptCopy

```
@page "/"
<ul>
        <li>
                <a href="/dotnetinjs">Call from JavaScript</a>
        </li>
</ul>
```

在 CallDotNetFromJavaScript.razor.cs 文件中创建一个静态方法（请记住，我们可以通过这种技术调用静态方法和那些作为实例化类的方法）。

我们只是实现了一个函数 checkNumber(int number) 来检查给定的数字是偶数还是奇数。

关键是理解 [JSInvokable] 属性；它表示紧随其后的方法 ([JSInvokable]) 将从 JavaScript 代码调用/调用。

javascriptCopy

```
public partial class CallDotNetFromJavaScript{
    [JSInvokable]
        public static string checkNumber(int number){
                if(number%2 == 0)
                        return $"The Number {number} is Even";
                else
                        return $"The Number {number} is Odd";
    }
}
```

在 scripts 文件夹中的 example.js 文件中编写以下代码。

javascriptCopy

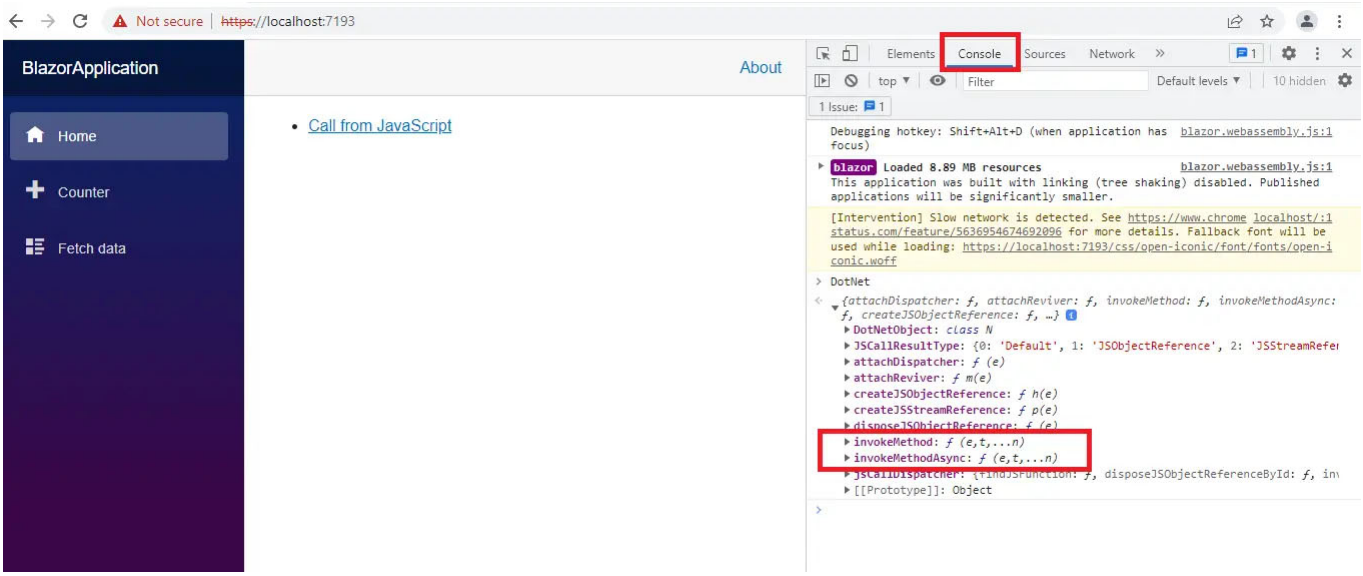
```
var jsMethods = {};

jsMethods.checkNumber = function () {
        const number = prompt("Enter your number");
        DotNet.invokeMethodAsync("BlazorApplication", "checkNumber", parseInt(number))
        .then(result => {
                        var el = document.getElementById("string-result");
                        el.innerHTML = result;
        });
}
```

你可能有一个问题，我们想从 JavaScript 调用 C# 方法，但使用 DotNet 对象。DotNet 对象用于从 JS 代码（JavaScript 代码）调用/调用 C# 方法。

还在迷茫吗？再次运行项目；当它显示界面时，按F12，转到 Console 标签页，然后输入 DotNet。

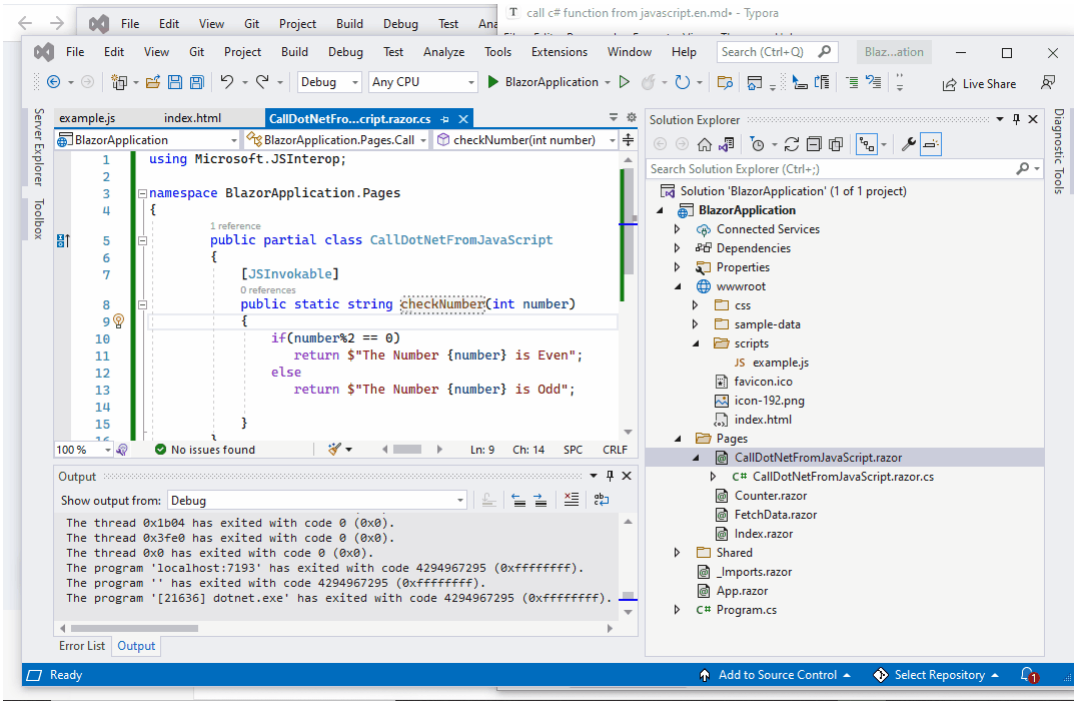
请参阅以下屏幕截图。



你可能已经观察到 DotNet 对象包含两个属性，invokeMethod 和 invokeMethodAsync，用于从 JavaScript 代码调用 C# 静态方法。我们在本教程中使用 invokeMethodAsync。

让我们再次运行 blazor 应用程序以从 JavaScript 调用 C# 函数并检查其行为。

输出：



很好! 我们从 C# 方法获得响应，以检查数字是偶数还是奇数。