Setting Up and Debugging a Blazor Project

Objective: by the end of this activity, you will be able to run and debug Blazor applications using Visual Studio Code. You will set up a project, utilize debugging tools like breakpoints, the Watch window, and Hot Reload, and identify issues in your code efficiently.

You'll create a Blazor WebAssembly application using Visual Studio Code. Ensure you have the required tools and extensions installed.

Step 1: Prepare for the Application

Instructions:

1. Verify Prerequisites: Ensure you have the following installed:
   - .NET SDK (minimum version 6.0)
   - Visual Studio Code
2. Install the C# Dev Kit Extension (if not already installed): The C# Dev Kit extension is required to debug Blazor applications. Follow these steps:
   - Open Visual Studio Code.
   - Go to the Extensions view by clicking the Extensions icon in the Activity Bar on the side of the window.
   - Search for C# Dev Kit and click Install.
3. Restart Visual Studio Code: Restart Visual Studio Code after installing the extension to ensure it's activated correctly.
4. Create a New Project Directory: In the terminal, create a new directory for your project and navigate to it: mkdir MyBlazorApp cd MyBlazorApp
5. Initialize a Blazor WebAssembly Application: Run the following command to create a new Blazor WebAssembly application: dotnet new blazorwasm -o MyBlazorApp
6. Open the Project in Visual Studio Code: Use the following command to open the project in Visual Studio Code:

```
code .
```

Here’s how you can update your activity to include the steps for setting up a launch configuration for debugging Blazor applications in Visual Studio Code. This addition will ensure you can configure your environment and run the application seamlessly.

Step 2: Using Print Statements to Debug

Scenario:

A for loop in your Blazor application isn’t iterating as expected. Use print statements to debug the issue.

Instructions:

1. Open the Counter.razor file in the Blazor project. You can find this file in the Pages folder of your application.

2. Replace the body of the IncrementCount method with a for loop that increments the currentCount variable by one, 5 times.

3. Add Console.WriteLine statements to your code that prints variables to the console.

4. When the Blazor app launches, select the Counter page from the user interface.

5. Select the Click Me! button.

6. Inspect the variables from the Console.WriteLine statement within the VS Code terminal window.

Step 3: Using Hot Reload for lnstant Updates

Scenario:

You want to modify the UI text and see changes reflected immediately without restarting the application.

Instructions:

1. Start the application using: dotnet watch run
2. Modify the text displayed in the Counter.razor file. For example, change the button text or the title.
3. Save the changes and observe the updated UI in the browser instantly.

Counter.razor:

```
@page "/counter"

<PageTitle>Counter</PageTitle>

<h1>Counter</h1>

<p role="status">Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click
me</button>

@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        for (int i = 0; i < 5; i++)
        {
            currentCount++;
            Console.WriteLine($"Loop iteration {i + 1}, currentCount =
{currentCount}");
        }
    }
}
```