Blazor - Practice Sheet

10 Q&A on component lifecycle, data binding, JS interop, and state

Q1. What is Blazor and what are the differences between its hosting models?

A: Blazor Server runs C# on the server and uses a SignalR WebSocket to push UI diffs to the browser low initial load, requires persistent connection. Blazor WebAssembly runs the .NET runtime in the browser as a WASM binary offline-capable, larger initial download.

Q2. What is the Blazor component lifecycle?

A: OnInitialized(Async) runs once on mount. OnParametersSet(Async) runs when parameters change. ShouldRender controls re-render. OnAfterRender(Async)(firstRender) fires after the DOM is updated. StateHasChanged() triggers a re-render manually from async callbacks.

Q3. How does two-way data binding work with @bind in Blazor?

A: @bind='property' is syntactic sugar for value='@property' plus oninput/onchange='@(e => property = e.Value)'. @bind:event='oninput' changes the trigger event. @bind works on input, select, and textarea elements, and on child component parameters.

Q4. How does Blazor JavaScript interop work?

A: Inject IJSRuntime. Call await JS.InvokeVoidAsync('window.alert', 'hello') to call JS from C#. In JS, DotNet.invokeMethodAsync('Assembly', 'StaticMethod', args) calls a [JSInvokable] static C# method. For instance methods, use DotNetObjectReference.Create(this).

Q5. What are cascading parameters in Blazor?

A: <CascadingValue Value='theme'> in an ancestor propagates theme to all descendant components that declare [CascadingParameter] ThemeConfig Theme. This avoids prop-drilling for cross-cutting concerns like themes, auth state, and localization.

Q6. How does Blazor routing work?

A: @page '/users/{Id:int}' at the top of a component registers the component as a route. NavigationManager.NavigateTo('/home') navigates programmatically. The Router component in App.razor matches the URL to a component and renders it in the Found template.

Q7. How do you validate a form in Blazor?

A: Wrap inputs in <EditForm Model='person' OnValidSubmit='HandleSubmit'>. Add <DataAnnotationsValidator /> to wire DataAnnotations rules and <ValidationSummary /> to display all errors. Per-field errors appear with <ValidationMessage For='() => person.Email'/>.

Q8. How do you manage shared state in a Blazor application?

A: Register a Scoped service (per-circuit in Blazor Server, per-session in WASM) that holds state. Inject it into components. Call StateHasChanged() in components after the service mutates state. Use events or Action callbacks on the service to notify subscribing

Q9. What are RenderFragment and templated components in Blazor?

A: RenderFragment is a delegate representing a chunk of Blazor markup. Expose a [Parameter] RenderFragment ChildContent and render it with @ChildContent in the template. Templated components accept RenderFragment<T> to give the caller control over rendering each item.

Q10. How do you make HTTP requests in Blazor WebAssembly?

A: Register a typed HttpClient with builder.Services.AddHttpClient<WeatherClient>(c => c.BaseAddress = new Uri(builder.HostEnvironment.BaseAddress)). The browser's fetch API handles the request under the hood. Use GetFromJsonAsync<T>() and