Predefined Script Tasks - Client Lifecycle Matrix

Per-flavor execution contract for the four predefined script tasks.

Lifecycle Matrix

FrameworX has four predefined script tasks that fire on platform lifecycle events. Each task has a fixed execution surface depending on which client flavor is connecting:

Predefined Task	WPF RichClient	HTML5 / WebClient	Notes
ServerStartup	n/a	n/a	Runs on the server (TServer.exe) at solution start, before any client connects.
ServerShutdown	n/a	n/a	Runs on the server at solution stop. Reliable — server shutdown is graceful.
ClientStartup	Yes	No	Runs once per client at session connect, on the client process. The HTML5 / WebAssembly client does not run client script tasks. For start-up logic that runs on every client, use the startup display's DisplayOpening (see below).
ClientShutdown	Yes	No (by design)	Runs at client close. Browser tab close gives the HTML5 WebClient no reliable shutdown window — tab kill, browser kill, navigation away, OS eviction on mobile, and device sleep all skip the unload phase. Wiring an unreliable hook would lie to script authors about determinism, so the platform deliberately does not invoke ClientShutdown on the WebClient.

Designing for Web Compatibility

If you need start-up logic to run on every client

Put it in the **DisplayOpening** method of the startup display's code behind (the first page the startup layout opens). DisplayOpening runs on the Rich Client, the Smart Client and the web client:

```
public void DisplayOpening()
{
    @Client.Theme = "Dark";
}
```

DisplayOpening runs each time that display opens. If the startup display can be opened again by navigation, keep one-time logic idempotent (setting a value is safe to repeat).

To choose only the theme a client starts with, no script is needed: set **Theme** in Displays Client Settings.

If you need teardown logic to fire reliably on web

Use the **ServerShutdown** task or a Server-domain script class instead. The server detects client disconnects and can perform any must-finish work centrally — saving state, releasing resources, persisting a session record.

If a ClientShutdown task body is non-empty in a web-targeted solution

The Designer's Build & Publish dialog emits a one-line info note in the build log (10.1.5 and later) so authors who copy a WPF solution to web don't silently lose teardown logic. Read it before you ship the web build.

Background

The decision to make ClientShutdown RichClient-only on the web reflects a deliberate platform stance: contracts that can fire unreliably are worse than contracts that don't fire at all. A "best-effort" ClientShutdown would silently skip in the most common shutdown paths a real user creates (closing the browser, switching tabs aggressively, OS evicting a backgrounded tab on mobile), and any code written under the assumption that it runs would be subtly incorrect.

By contrast, the WPF RichClient closes through a graceful `MainWindow.Closing` path that the platform can hook synchronously, so ClientShutdown fires deterministically there.

In this section...