Clipboard Access Code

Copy and paste text from Display CodeBehind and client scripts.

Use **TClipboard** to read and write the system clipboard from Display CodeBehind and client scripts.

TClipboard compiles and runs identically on the desktop (RichClient / SmartClient, WPF) and the web (HTML5) client, so the same CodeBehind works on both targets.

> TClipboard is available in the FrameworX 10.1.5 line. On earlier builds, use the platform clipboard as described in **Migrating from Clipboard** below.

API Surface
Examples
Copy text to the clipboard
Paste clipboard text into a Tag
Web Client Behavior
Permission on read
Denied access
Migrating from Clipboard

API Surface

TClipboard resolves directly in client scripts. No *using* directive and no qualification are needed.

Method	Returns	Description
`TClipboard.SetTextAsync(string text)`	`Task`	Places *text* on the clipboard.
`TClipboard.GetTextAsync()`	`Task<string>`	Returns the text currently on the clipboard.
`TClipboard.ContainsTextAsync()`	`Task<bool>`	Returns true when the clipboard holds text.

> **The API is asynchronous by design**
>
> Web browsers expose only an asynchronous clipboard, and no synchronous read is possible on the web client. TClipboard therefore offers async methods only, so that every method behaves the same on both clients. On the desktop client the calls complete immediately.

Examples

Copy text to the clipboard

Call the method from a Display object dynamic, for example a button Action.

CodeBehind: copy to the clipboard

```
public async Task CopyAsync(object sender, System.Windows.Input.InputEventArgs e)
{
    await TClipboard.SetTextAsync("payload");
}
```

Paste clipboard text into a Tag

Check for text before reading, so the Tag is left untouched when the clipboard holds an image or is empty.

CodeBehind: paste into a Tag

```
public async Task PasteAsync(object sender, System.Windows.Input.InputEventArgs e)
{
    if (await TClipboard.ContainsTextAsync())
        @Tag.ClipText = await TClipboard.GetTextAsync();
}
```

Declare the handler as `async Task` and *await* every call. Calling a TClipboard method without awaiting it raises compiler warning CS4014 and the operation may not have completed when the handler returns.

Web Client Behavior

On the HTML5 client the browser controls access to the clipboard, which adds two behaviors that do not exist on the desktop client.

Permission on read

The first read triggers the browser permission prompt, or a paste confirmation box depending on the browser. TClipboard waits for that grant and completes the read within the same user gesture, so the operation succeeds on the click that started it.

Denied access

When the user denies the prompt, or a browser policy blocks clipboard access, the read throws an `InvalidOperationException` with a message pointing at the permission. Handle it to give the operator a clear message.

CodeBehind: handling a blocked read

```
public async Task PasteAsync(object sender, System.Windows.Input.InputEventArgs e)
{
    try
    {
        if (await TClipboard.ContainsTextAsync())
            @Tag.ClipText = await TClipboard.GetTextAsync();
    }
    catch (InvalidOperationException ex)
    {
        @Tag.StatusMessage = ex.Message;
    }
}
```

Read the clipboard from a user action such as a button click. Browsers reject clipboard reads that are not tied to a user gesture, so a read started from a timer or from a Display lifecycle event fails.

Migrating from Clipboard

CodeBehind is compiled for both the desktop and the web target, and the platform `Clipboard` class differs between them. Scripts using `Clipboard.SetText(...)` keep working, but the web pass reports:

```
[HTML5] warning CS0618: 'Clipboard.SetText(string)' is obsolete: 'Use SetTextAsync(string) instead.'
```

Do not apply that suggestion to the `Clipboard` class. The desktop pass has no `Clipboard.SetTextAsync`, so the build fails with error CS0117. Replace the class with **TClipboard** instead, as shown below.

Replace each call with its TClipboard equivalent:

Replace	With
`Clipboard.SetText(text)`	`await TClipboard.SetTextAsync(text)`
`Clipboard.GetText()`	`await TClipboard.GetTextAsync()`
`Clipboard.ContainsText()`	`await TClipboard.ContainsTextAsync()`

The synchronous form does not exist on TClipboard. This code does **not** compile:

Not supported: synchronous clipboard access

```
// error CS0117: 'TClipboard' does not contain a definition for 'SetText'
public async Task CopySync(object sender, System.Windows.Input.InputEventArgs e)
{
    TClipboard.SetText("payload");
}

// error CS0117: 'TClipboard' does not contain a definition for 'ContainsText' / 'GetText'
public async Task PasteSync(object sender, System.Windows.Input.InputEventArgs e)
{
    if (TClipboard.ContainsText())
        @Tag.ClipText = TClipboard.GetText();
}
```

Migrating also future proofs the script: if a future web client engine drops the obsolete synchronous methods, TClipboard absorbs the change and the solution keeps running.

For the wider set of differences between the desktop and web compile targets, see Display Web Expressions Code. For guidance on async methods in scripts, see Asynchronous Programming Code.

In this section...
