Custom Style

CustomStyle is a global function that can be called on the form onload event to apply custom CSS styling to the form. The styling allows the form to have a header section, a row of tabs, body sections that display based on which tab is selected, and a footer that stays in place regardless of which tab is selected. This styling results in a better user experience on long forms that have many sections. Rather than scrolling through information top-to-bottom, the user can select the tab they wish to view.

Contents

[Visual: In the Form Viewer 2](#_Toc5894871)

[Visual: In the Form Designer 3](#_Toc5894872)

[Form Designer 4](#_Toc5894873)

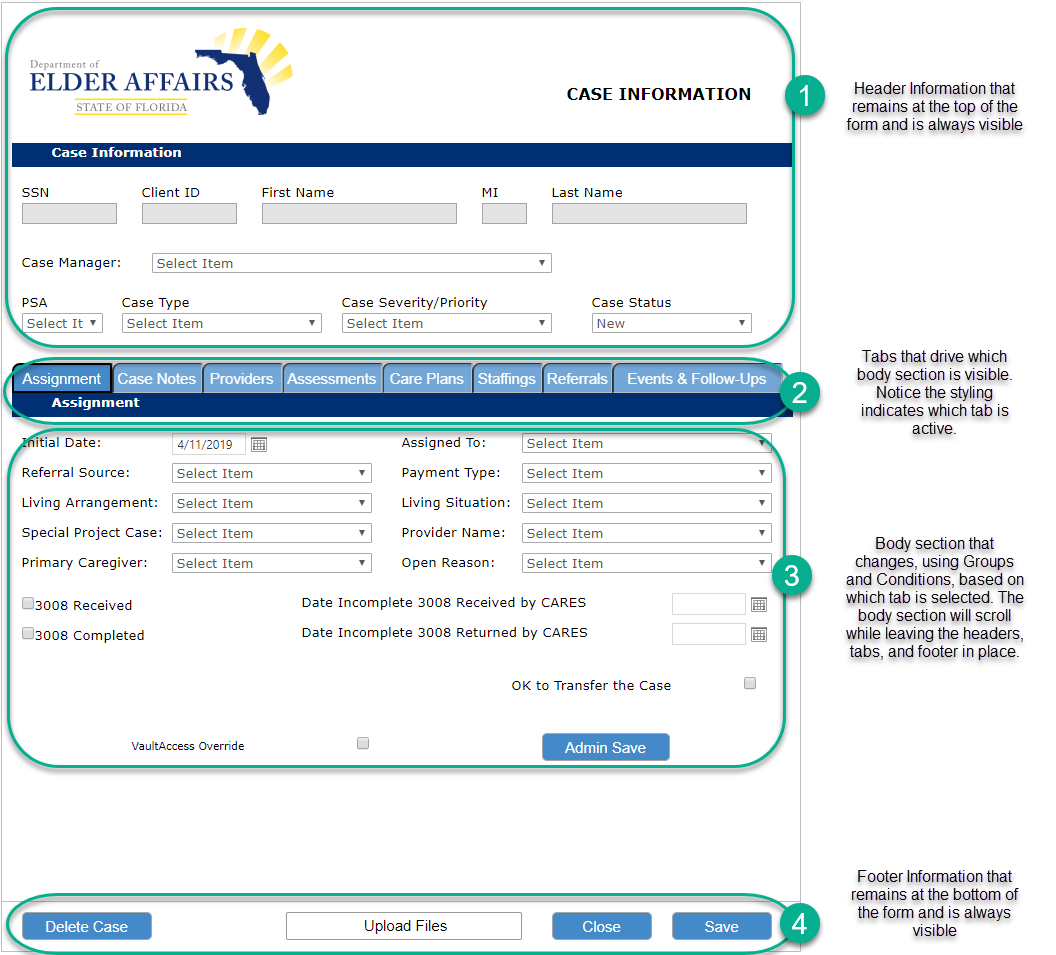
[Containers 4](#_Toc5894874)

[Form Fields 4](#_Toc5894875)

[Script Editor 4](#_Toc5894876)

[Groups and Conditions 5](#_Toc5894877)

# Visual: In the Form Viewer



# Visual: In the Form Designer

In the screenshot below, notice that the containers are not in the order in which they appear in the form viewer.



1. Headers. You may have up to two header containers.
2. Tabs. Notice that a solid rectangle is located directly below the tab buttons, with many labels stacked on top of each other. Each tab is associated with a label and one or more body containers. Only one tab container may exist.
3. Body containers continue down the form. You may have as many body containers as you wish. One or more than one can appear for each tab button selected.
4. Footer information that will be moved to the bottom of the screen when CustomStyle() is called. At least a Close button should be located in this container. Typically a Save button will be appropriate as well. Any other fields or buttons that should always be visible at the bottom of the form should be included in this footer container. Only one footer container may exist.

# Form Designer

Custom Style works best when used in the new form designer, as it requires the form EventsEnd event.

# Containers

CustomStyle requires that specific containers exist on your form. Ensure that your form includes containers named the following:

* \_Header\_1
* \_Header\_2 (optional)
* \_Tab\_1
* \_Footer\_1

All other containers will be considered “body” containers. Body containers are not required to be specially named.

# Form Fields

In the Hidden fields of your form, include a text field called “Tab Control.” Setting this field value programmatically from the tab buttons will drive the groups and conditions on the form, which creates the overall experience of clicking through tabs.

Tab buttons should be placed directly next to each other, with no space between them. The height of tab buttons should be updated to 30 in order to effectively place the solid-colored rectangle below without any space between. Tab buttons should be named with a prefix “’btnTab.”

Stacked labels in the \_Tab\_1 container should be named controls. This will enable you to put the each label into the correct group later. Prefix all of your tab label names with something consistent, like “LabelTab\_.”

# Script Editor

The CustomStyle.js script that is found on GitHub in VisualVault-Implementation-Library/Client Side Global Functions/CustomStyle/ should be added as a Global Function in the Script Editor.

VV.Form.Global.CustomStyle(); should be called in the form Load event and in the EventsEnd event.

The form Load event should also set the Tab Control field back to a default value. For example: VV.Form.SetFieldValue('Tab Control', 'Assignment');

Each tab button control (named with prefix “btnTab”) should include the following code in its onClick event: VV.Form.SetFieldValue('Tab Control', control.value); This code changes the Tab Control text box in the hidden area of your form to the button Text.

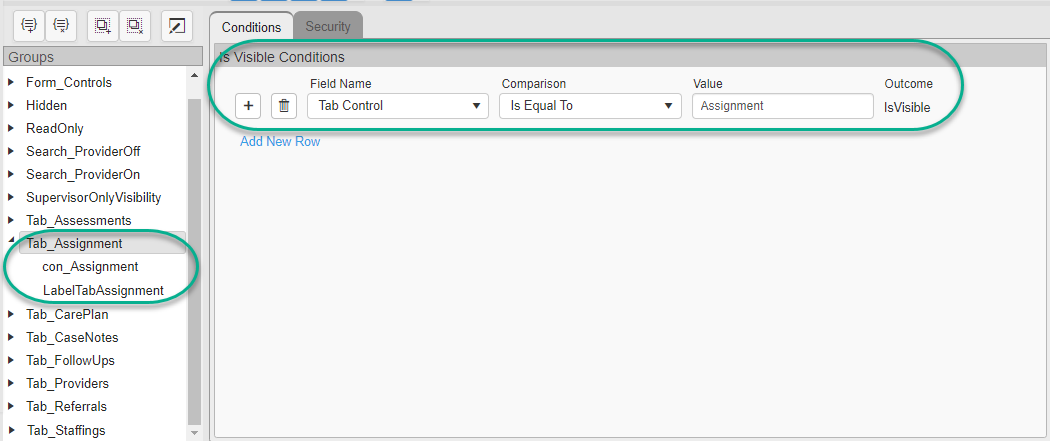
# Groups and Conditions

While the styling of the form and the tabs relies on CustomStyle(), groups and conditions are an important part of creating an app-like experience on a form.

Each tab button should have a corresponding group. The group names should be consistent, such as with a prefix of “Tab\_”, e.g. Tab\_Assessments, Tab\_CarePlan, Tab\_CaseNotes, etc.

Each tab group should include at least one body container and the label that is located in the \_Tab\_1 container.

The group should include IsVisible Conditions. The conditions should be: Tab Control is Equal To [button text].



Other groups may be used as needed. Simply remember to incorporate the current value of your Tab Control as appropriate.