**SCRIPTING IN SAP ADOBE FORM**

The main advantage of using scripting in Adobe forms is that we can improve functionality and usability of forms.

With scripting we can get full control over forms at run time.

**Benefits of Scripting:**

* Display/Hide form objects based on Conditions while generating the PDF Form

        (Examples: Hide /Display Table or Page in generated PDF Form, Adding Back ground colours based on conditions)

* Automatic calculation in Forms.

        (Examples: Sum, Avg, Percentage, etc.. )

* Data validations.

        (Check existence of Data using flag conditions..).

        All the Actions written in Script editor will be executed at the run time.

**Where should we have to write our Script in Adobe Forms:**

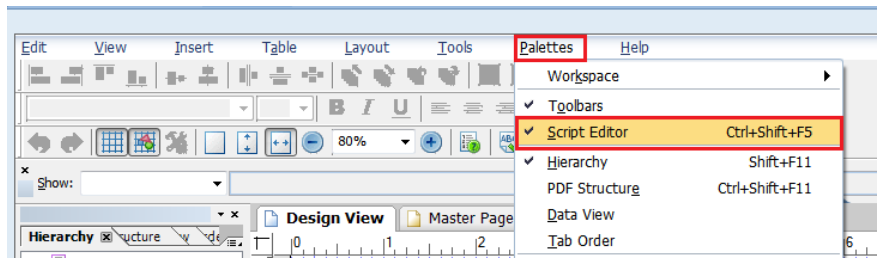
        We can create our forms using Transaction Code **SFP (Form Builder).**

        Go to Transaction Code SFP.

        Enter the Form Name

        Click on **Layout**.

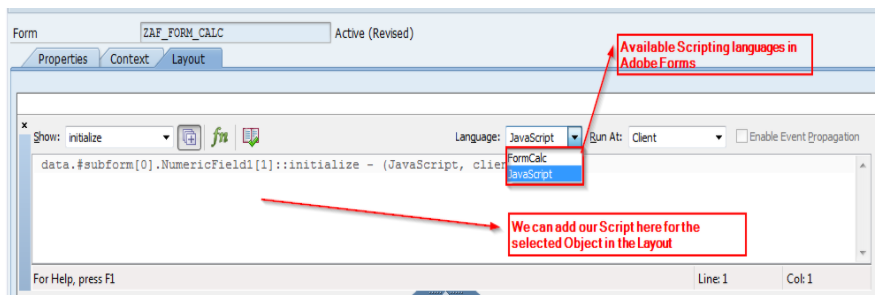
**Menu Bar –>** **Pallets –>** **Script Editor:**



Select the **Script Editor**.

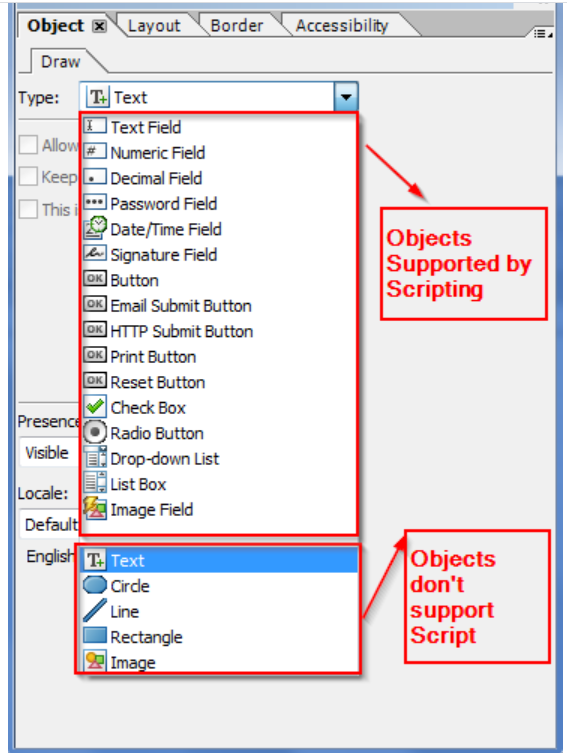
Select an **Object** in the Layout.

The Script editor will be displayed as shown below.



We can divide the objects into two categories.

* **Objects supported by Scripting Languages.**
* **Objects not Supported by Scripting languages.**



SAP Adobe forms Supports two scripting Languages.

* **FormCalc.**
* **JavaScript.**

A Form can use both languages in the form at the same time but we can’t use both scripting languages for single Object.

**FormCalc:**

         FormCalc is the default scripting language in Designer. It is an easy-to-use calculation language.

         FormCalc is the best choice to work on functions like (date and time, financial, arithmetic, logical etc.)

**Advantages:**

* **Built-in-Functions**

        FormCalc has many built in Functions that covers a range of Areas including Finance, logic, dates/times and mathematics.

* **Performance**

        FormCalc usually provides better performance than JavaScript because it executes calculations and validations more quickly.

        FormCalc can be the best choice in many situations, regardless of programming experience.

* **Easier to maintain**

        This scripting language is very useful for Non-Programmers who need to add calculations for their forms.

        It is very similar to the Language that we use in Microsoft Office like Excel, Most FormCalc scripts are one-line long.

**Disadvantages:**

* FormCalc is not as powerful or as ubiquitous as Java Script.
* It will not work for HTML Forms ..
* Not very useful for creating sophisticated interactive and dynamic forms.

**Java Script:**

Although FormCalc is default scripting language in designer, Java script is more ideal for creating sophisticated interactive and dynamic forms.

**Advantages of Java script:**

* **JavaScript is ubiquitous.**

        JavaScript is a scripting standard that many programmers and designers already know.

* **Object -oriented.**

         We can create JavaScript objects and custom functions, and use them through out Form.

         This function is not available in FormCalc.It has been scripting standard we can find sample scripts for almost we need to do.

* **Works for HTML and PDF Forms:**

        We can use JavaScript for both PDF Forms and HTML Forms.

**Disadvantages:**

* **Performance**

        Performance is lower than Formcalc.

**Events**

Every calculation or script you attach to a form object is associated with a specific event. An event is defined as a particular occurrence or action that can change the state of a form and, when the change of state occurs, automatically invoke a calculation or script associated with the event. Events occur at various times, from the beginning of the form rendering process when merging data with a form design, all the way through to a form filler interacting with objects on a form in a client application. By applying calculations and scripts to specific events, you can control every aspect of how you present form objects, as well as form data, and how the objects and data respond to form filler interaction.

A single change of state or form filler action may trigger multiple events. For example, tabbing from the current field to the next field triggers both the exit event for the current field and the enter event for the next field. If the current and next fields are in different subforms, a total of four events are triggered; namely, exit events for the current field and subform, and enter events for the next field and subform. In general, each of the different categories of form events follow a predictable ordering.

**Types of events**

Form events fall into one of the following categories:

**Process events** This type of event initiates automatically as the result of an internal process or action related to objects on a form. For example, if a form filler clicks a button that adds a new page to the form, the initialize, calculate, validate, and layout: ready process events initiate automatically for the new page.

**Interactive events** This type of event initiates as a direct result of form filler actions. For example, if a form filler moves the pointer over a field on a form, the mouse Enter event initiates in response to the action.

**Application events** This type of event initiates as a result of the actions that either a client application or a server application performs. For example, you can create a calculation or script to perform a task immediately after the form is saved by using the post Print event.

**Process events**

Process events initiate automatically as the result of an internal process or action related to a form or objects on a form. These events initiate immediately following significant form changes; for example, after a form design is merged with data or after the form pagination process finishes. Process events also initiate immediately after interactive events initiate. For example, immediately after any interactive event initiates, the calculate event initiates followed by the validate event.

The following list contains the process events, which are available from the Show list in the Script Editor:

* calculate
* form:ready
* indexChange
* initialize
* layout:ready
* validate

Process events can initiate many times as a result of dependencies; that is, actions associated with a single event that ultimately initiates one or more additional events. Using an example of a form filler clicking a button to reveal a previously hidden portion of the form, after the form filler clicks the button, not only does a series of interactive and processing events initiate for the button itself, but a number of process events for the new subform initiates as well.

**Interactive events**

Interactive events initiate as a direct result of form filler actions, which makes these events useful for a variety of calculation and scripting tasks. For example, you can add a script to the mouseEnter event for a text field that changes the border color of the field to blue and a script to the mouseExit event that changes the border color back to the original color. This action creates a highlighting effect when form fillers move the pointer over the field to visually assist them while filling the form. Interactive events are also useful for changing form data in response to a form filler selection. For example, you can add a script to the change event for a drop-down list that updates the data values in multiple fields in response to the value the form filler selects in the drop-down list.

The following list contains the interactive events, which are available from the Show list in the Script Editor:

* change
* click
* enter
* exit
* mouseDown
* mouseEnter
* mouseExit
* mouseUp
* postOpen
* postSign
* preOpen
* preSign

**Application events**

Application events initiate as a result of the actions that a client application or a server application perform, either due to a form filler action or an automated process. Application events do not exist as part of a general flow of events. They are single events that correspond to actions that the client or server application performs.

The following list contains the processing events, which are available from the Show list in the Script Editor.

* docClose
* docReady
* postPrint
* postSave
* postSubmit
* prePrint
* preSave
* preSubmit