

# ABAP Part III

## Lesson 03: Adobe Forms

# Lesson Objectives



After completing this lesson, participants will be able to -

- Know the Adobe Forms Architecture
- Understand Interface, context and Layout
- Know how to integrate Adobe Form in an ABAP application program





Forms are used for mass printing in SAP systems. Besides using the printer for standard output you can also select the Internet (by using a generated HTML output), a fax, or e-mail as the output medium.

## Tools Delivered by SAP for Form Designing

- SE71 – Sapscripts
- SmartForms – Smart Forms (introduced in SAP Basis Release 4.6C)
- SFP – Adobe Form (As of SAP NetWeaver '04 )

As of SAP NetWeaver '04 (in SAP Web Application Server), you can use a new solution to create interactive forms and print forms for the optimization of your form-based business processes. This solution uses Portable Document Format (PDF) and software from Adobe Systems Inc. that has been integrated into the SAP environment



# Overview - Features

Create form templates for the layout that include logos or pictures

Edit forms online or offline

Forms can be filled in advance automatically with specific data from SAP applications and then sent to the correct recipients using secure methods

Automatic consistency checks for forms

Activate enhanced functions such as comments

Digital signatures and form certification

User-friendly tools reduce the time and costs associated with creating form layouts.

The usage of the PDF format means that forms retain their appearance regardless of the environment they are used in.

# Overview - Advantages Over Smart Forms/SAPscript



- Adobe Lifecycle Designer is an easy to use, flexible tool for designing forms
- Full integration into the SAP development environments for Java and ABAP
- Graphics (BMP, JPEG, GIF, PNG, EXIF) can be included into forms directly no conversion required
- Objects (including texts) can be rotated
- Different page orientations (landscape, portrait) are possible within one form
- Graphical elements can be included in forms
- Existing PDF or Word documents can be imported
- Barcodes can be printed on all printers of types Postscript, PCL, PDF, or Zebra
- Mailing and faxing is easier
- Scenarios and integration into browser-based applications are possible (Web Dynpro for Java or ABAP)

# Overview - Adobe Designer: - Technical Pre-requisites



SAP 6.40 GUI patches are installed



## **SAP GUI for Windows**

- Make sure that Designer is installed and check on your hard drive. Default location for Windows: C:\Program Files\Adobe\Designer 7.1 (depending on version)

Adobe Reader 7.0 (incl. update to 7.0.8) - The most current version should always be used, in particular for interactive features. Check SAP Note 834573 for details.



## **Adobe Reader**

- used to be called Acrobat Reader



## **Microsoft Windows**



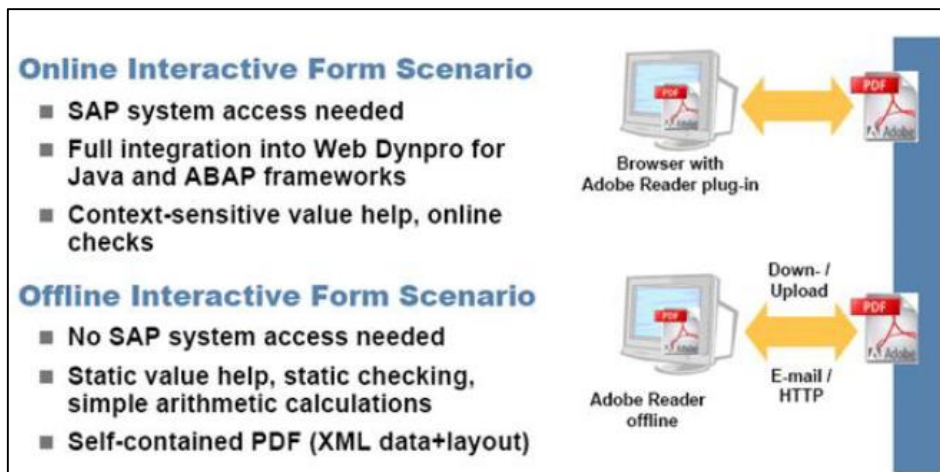
Microsoft Windows 2000 or higher

# Overview - Adobe Form – Types



There are two basic types of forms:-

- 1. Interactive Forms ( Basic scenarios )
  - Online
    - The user is logged on to the SAP system when he or she fills out the form.
  - Offline
    - The user is not logged on to the SAP system when he or she fills out the form. Once the form has been filled out, the user sends it to the issuer of the form, for example by e-mail. The SAP system of the issuer then extracts the data from the form.





## Print Forms - Used normally for Printing/Fax/Email



- Forms for Printing, E-Mail, or Fax
- You can use the Form Builder (integrated into ABAP Workbench) to create
- PDF-based print forms that you can then print or send by e-mail or fax. When you create these print forms, you can rely on the tried and trusted principle of separate data retrieval and form layout processes.
- This enables you to make changes to either one of the processes, without affecting the other.
- PDF-based print forms can be used for the following: Order confirmations  
Invoices , Account statements , Checks etc



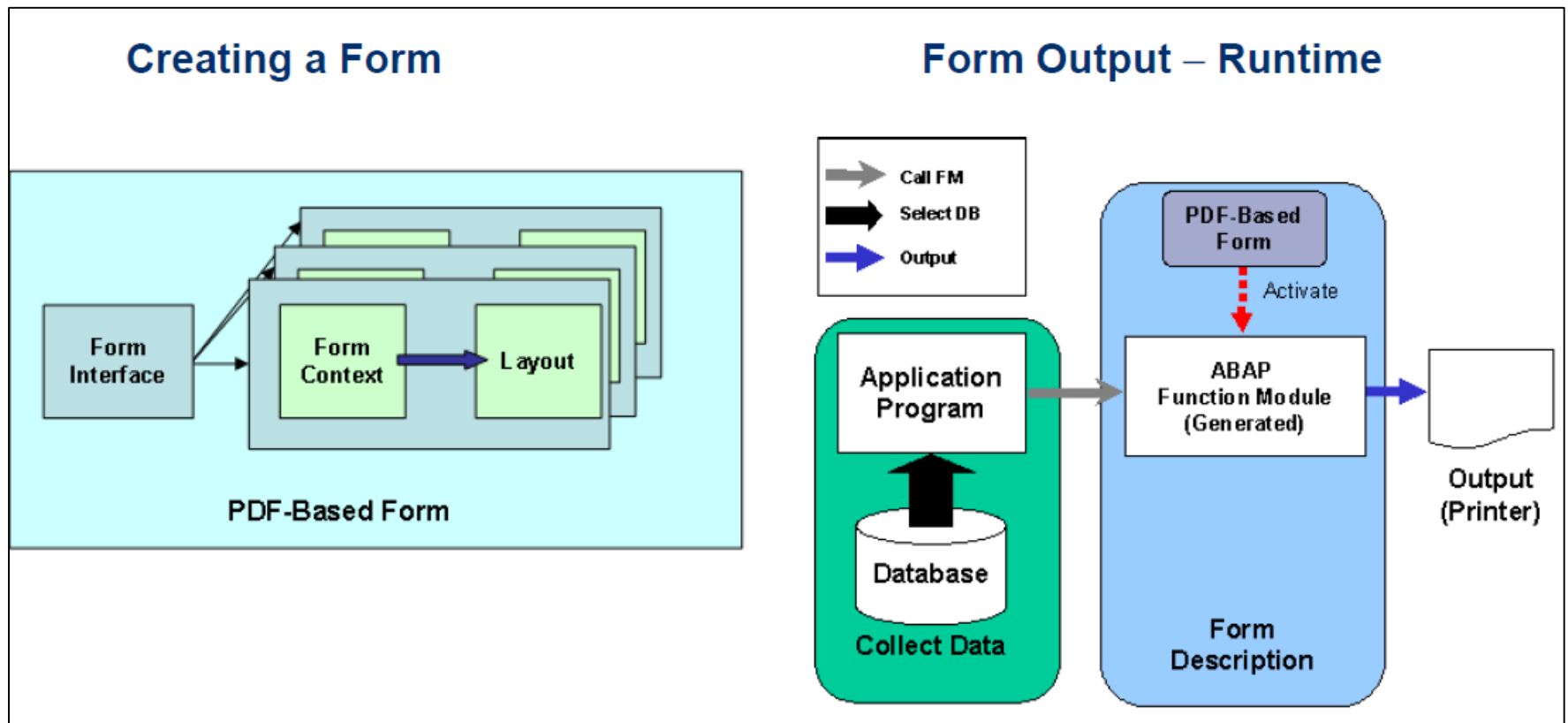


A PDF-base print form has the following attributes:

- A form interface that sends the application data to the form.
- A form context that contains the form logic. This logic controls the dynamic formatting of the form. For example, it enables variable fields to be displayed; it specifies that certain texts appear only under certain conditions (one text for a first warning and a different text for a second warning); and it can specify that invoice items can be processed repeatedly in a table.
- A layout. In the layout, you define how the output data is positioned, its appearance in graphics, and the design of the pages.



The following graphics show you the architecture that is implemented when you create and print a PDF-based form.



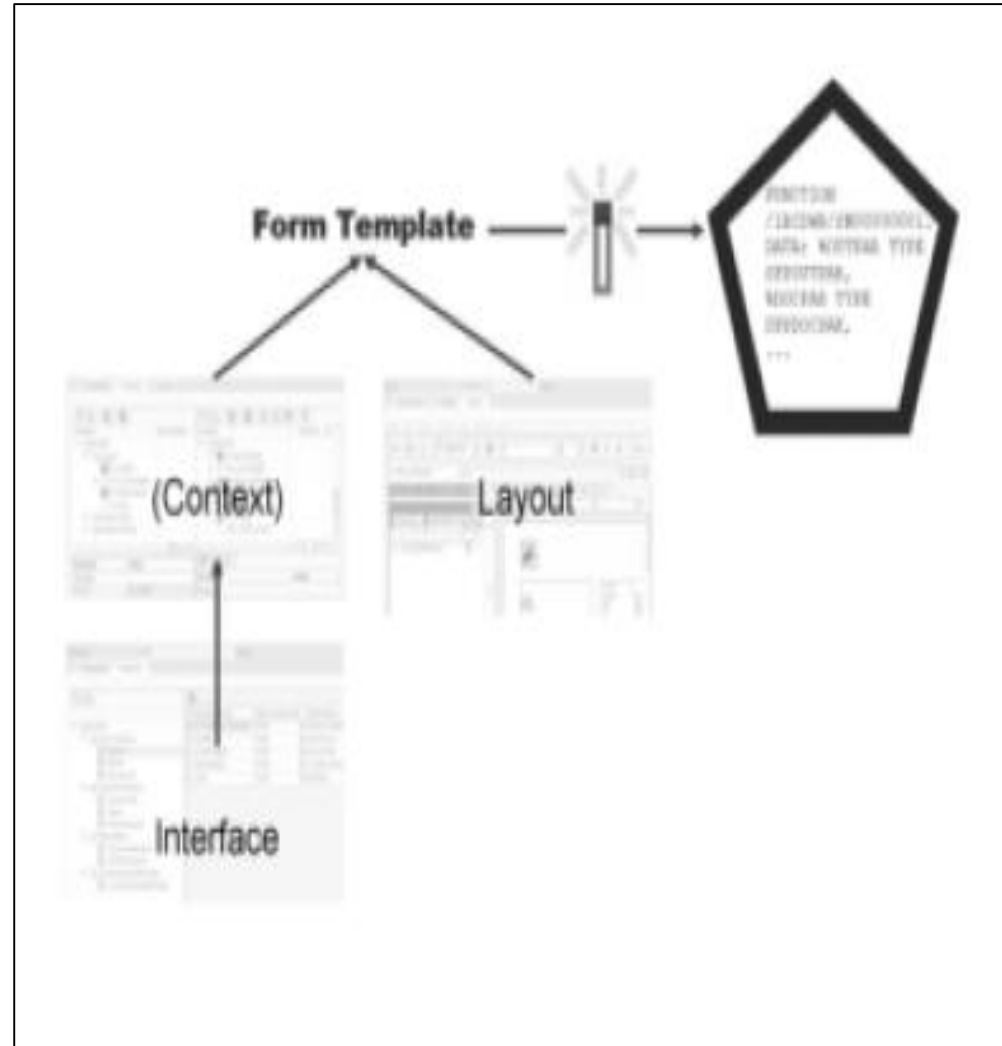
# Architecture - The Tools Involved (Design Time)

## 1. Interface (transaction SE80 or SFP)

- Reusable
- The interface defines which data a
  - Program can possibly pass on to a form.

## 2. Form template (transaction SE80 or SFP)

- Consists of a context and the layout.



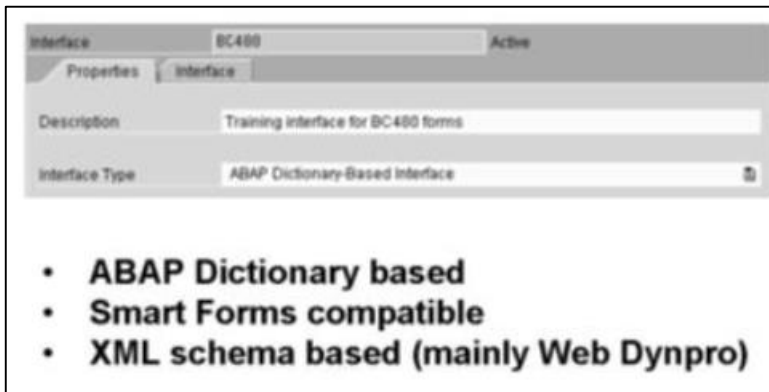


Every PDF-based form needs to have an interface; it is the link between the ABAP program and the form.

The program can pass data to the form only if it is defined in the interface, and (most of) the dynamic data used in the form layout will be defined in the interface.

To access interface maintenance, use transaction SFP. Alternatively, use transaction SE80 and choose Other object.

## Types of an Interface



The screenshot shows the SAP SFP (Smart Forms Properties) transaction for interface BC400. The 'Interface' tab is selected, showing the following details:

- Interface: BC400
- Status: Active
- Description: Training interface for BC400 forms
- Interface Type: ABAP Dictionary-Based Interface

Below the screenshot, the following types of interfaces are listed:

- **ABAP Dictionary based**
- **Smart Forms compatible**
- **XML schema based (mainly Web Dynpro)**



The types “ABAP Dictionary based” and “Smart Forms compatible” are used for print scenarios.

“XML schema based” (which was introduced in SAP NetWeaver 2004s) is primarily used for Web Dynpro scenarios.



A form interface of the type ABAP Dictionary has only one default import parameter (/1bcdwb/docparams of type sfpdocparams). It is used to determine a form's locale (language and country) and whether the form will allow interactive features.

Export parameters can be added only for those interfaces that are compatible with Smart Forms.

Exceptions that you declare in an interface can be raised in ABAP coding of a form. They are based on the traditional exception concept (not the class-based concept that was introduced in SAP Web Application Server 6.10).

You raise an exception as such: `RAISE <exception>.` (Alternative: `MESSAGE <message_type> <message>(<message_class>) RAISING <exception>`)



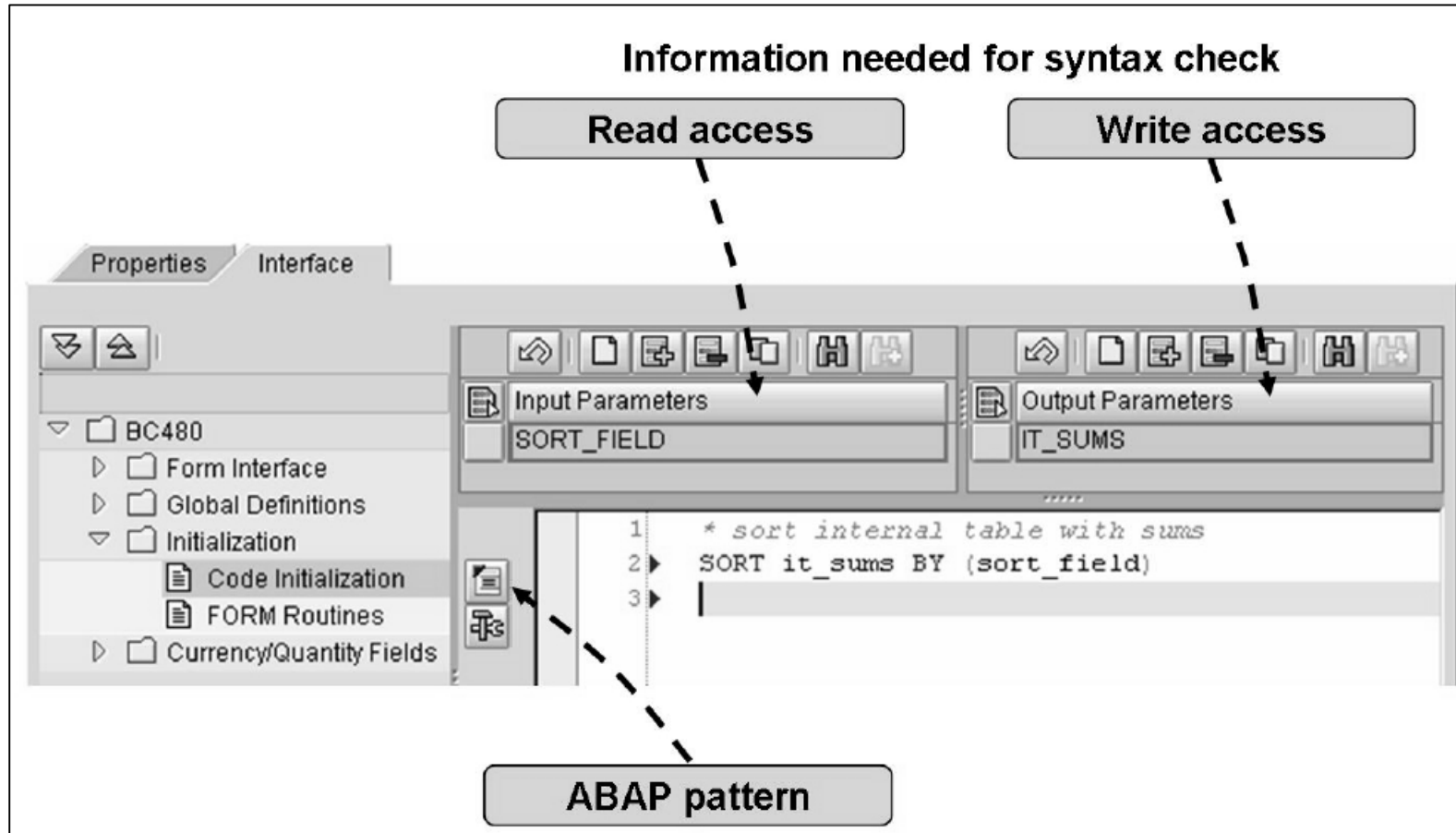
**Global Definitions** - There are also parts to an interface that are actually invisible from outside, that is, they cannot be accessed from the application program. Among them are the global definitions, initialization coding, and currency/unit fields.

**Global fields:** Global fields can be integrated in the form layout.

**Field symbols:** Field symbols might act as placeholders for variables. They are useful in dynamic programming and for speeding up the processing of internal tables.

**Global types:** If your global fields (or any fields that you might declare within ABAP coding) need types other than ABAP types (i, n, f, c, p, and so on) or Dictionary types, you can create local types in the editor that opens when you choose *Types*.

# Interface - Initialization





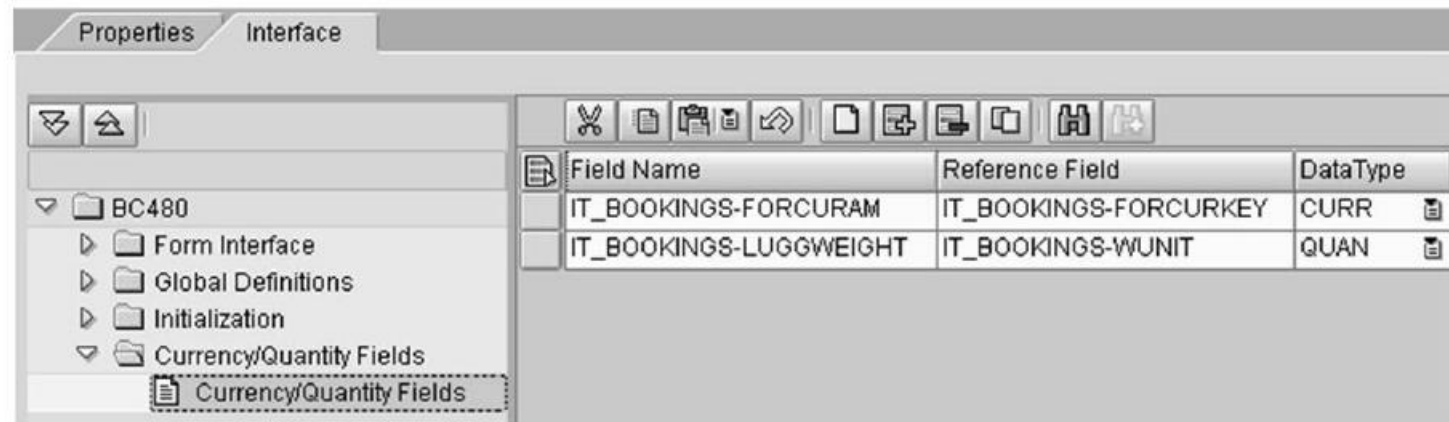
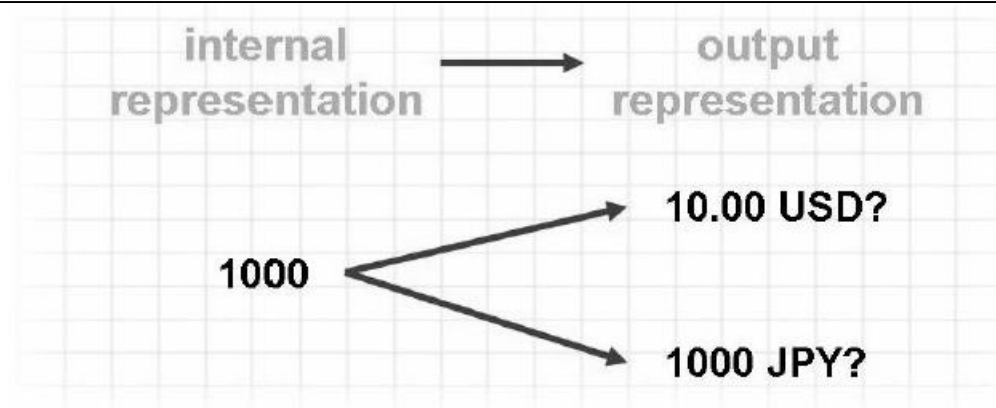


During initialization, data coming from the program can be changed before it is sent to the form. In forms that have not been migrated from Smart Forms, initialization is the only time when ABAP coding can be executed.

Even if the initialization coding makes use of form interface fields or global fields, you still have to make them known to the initialization coding. Enter those fields that you read from under Input Parameters, and those that you set under Output

In the initialization coding, you can call form routines that you have created in the interface: `PERFORM ...` Form routines make sense for coding that needs to be executed several times. You define form routines by using the ordinary ABAP syntax `FORM xyz... ENDFORM`.

# Interface – Currency/Quantity Fields

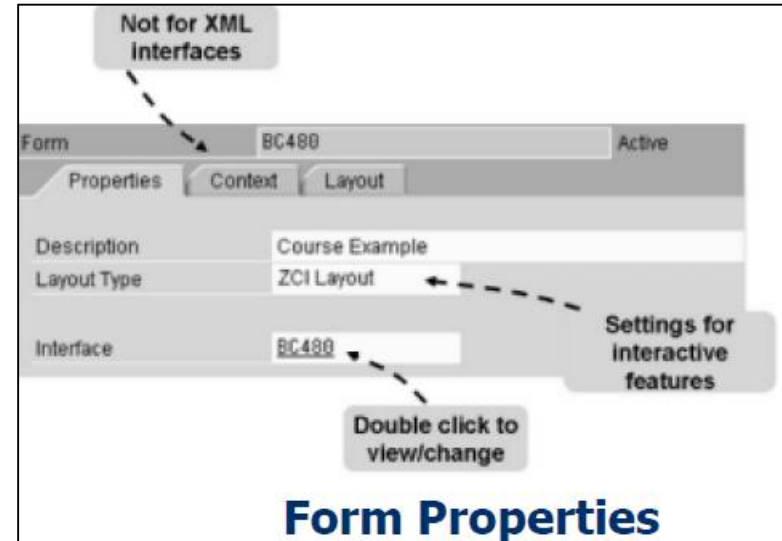


# Context



The context is essential, as it provides the source for data for a form. Apart from static elements, only those texts, fields, images, and so on can be included in the layout of a form that have been integrated into the form's context. However, the context should not be overloaded, as this will have a negative impact on printing performance.

It can be seen as a subset of the interface enriched with some form specific information.



# Context - General Handling of the Context



**Interface that is used**

**Context**

Properties Context Layout

Interface

| Interface     | Description               |
|---------------|---------------------------|
| BC480         |                           |
| Import        |                           |
| IT_SUMS       |                           |
| IS_CUSTOMER   |                           |
| IT_BOOKINGS   | Table for Flight Bookings |
| IV_URL        |                           |
| Global Data   |                           |
| System Fields |                           |

Property Value

| Property         | Value          |
|------------------|----------------|
| General          |                |
| Name             | IT_SUMS        |
| Description      |                |
| Type Information |                |
| Type Category    | Internal Table |
| Type Name        | FLPRICE_T      |

**Properties of interface field**

Context

| Context         | Inactive | Generated | Description                    |
|-----------------|----------|-----------|--------------------------------|
| BC480           |          |           |                                |
| IT_BOOKINGS     |          |           | Table for Flight Bookings      |
| IS_CUSTOMER     |          |           | Flight customers               |
| WHICH_CLERK     |          |           | Choose clerk based on custom   |
| TRUE            |          |           |                                |
| FALSE           |          |           |                                |
| IT_SUMS         |          |           | Flight Price and Currency (Tra |
| DATE            |          |           | Application Server Date        |
| LOGO            |          |           | Company logo                   |
| COMPANY_ADDRESS |          |           | Company address for envelope   |

Properties

Property Value

| Property    | Value |
|-------------|-------|
| General     |       |
| Name        | TRUE  |
| Description |       |

**Properties/details/conditions of context element**

# Context - Using the Interface



The screenshot shows the SAP Context menu for the 'BC480' interface. The menu is divided into several sections, with annotations highlighting key features:

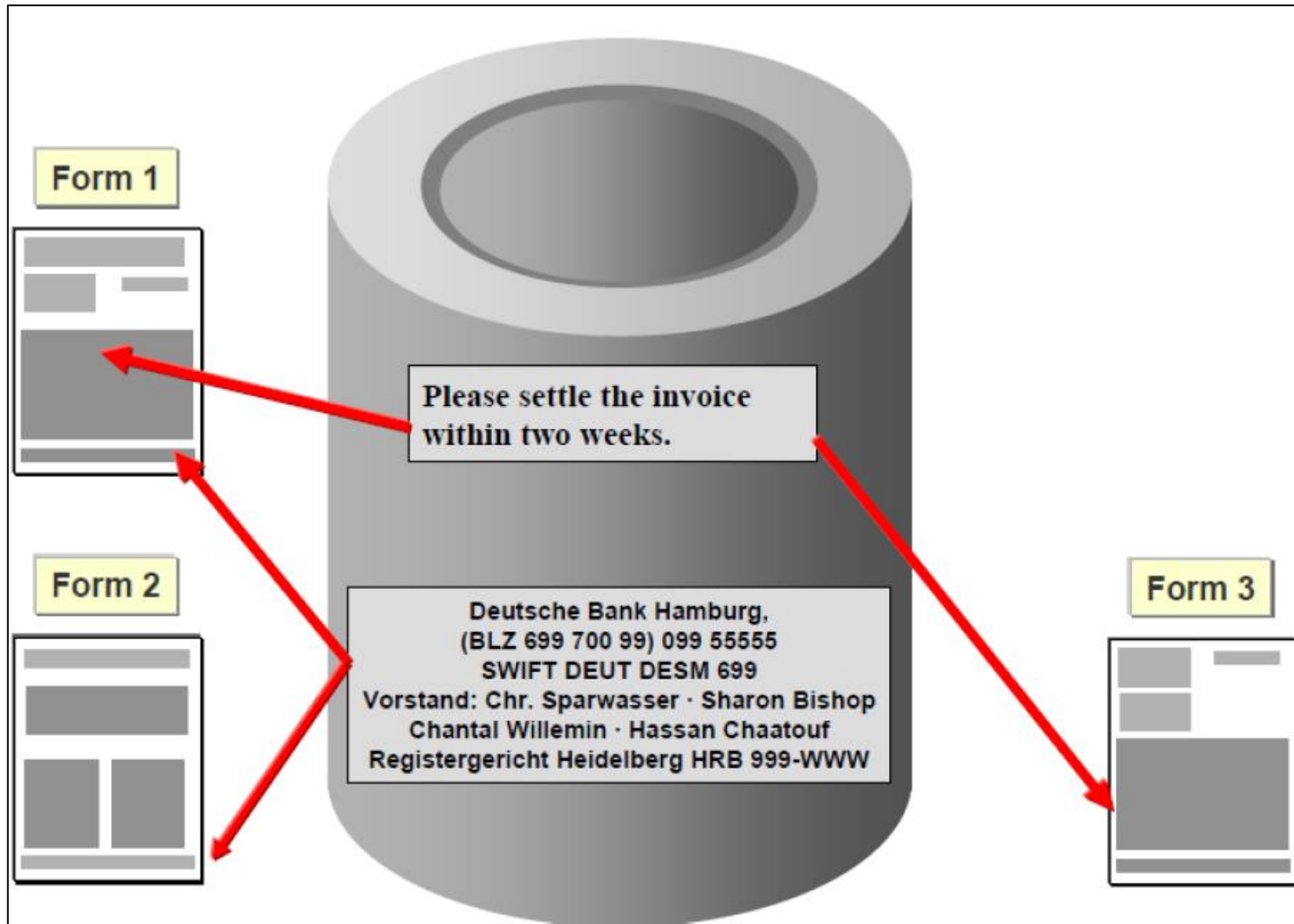
- For interface structure fields:** Points to the 'Generated' column header.
- Create new nodes:** Points to the 'Create' menu item, which includes options like 'Graphic', 'Address', 'Text', 'Alternative', 'Folder', 'Data...', 'Structure...', 'Loop...', and 'Single Record'.
- Integrate interface fields:** Points to the 'Integrate' menu item, which includes options like 'Field', 'Structure', 'Loop', and 'Single Record'.
- Performance tuning:** Points to the 'Performance' menu item, which includes options like 'Flight Price and Currency (Training Purpose)', 'Current Date of Application Server', and 'Company logo'.

The main menu structure is as follows:

- Interface
  - BC480
    - Import
      - IT\_SUMS
      - IS\_CUSTOMER
        - IT\_BOOKINGS
          - IV\_IMAGE\_OVR
    - Global Data
    - System Fields
      - SFPSY
        - DATE
        - TIME
        - USERNAME
        - SUBRC

The 'Context' menu is also visible, showing a list of fields and their status (Inactive or Generated).

# Context - Text Modules and SAP script Texts



# Context - Text Modules and SAP script Texts



You need to create the texts only once and can then reuse them as required.

You make changes centrally only once without having to modify the actual forms.

Typical examples include headers (company address), footers (company information like board members and so on), and whole pages containing introductions or terms of trade.

Use text modules of Smart Forms and also include texts (that is, SAPscript texts).

Run transaction SMARTFORMS (for text modules) or SO10 (for include texts

# Context - Creating Text Modules



The screenshot illustrates the SAP SmartForms interface. The top window, titled "SMARTFORMS", shows the "SAP Smart Forms: Initial Screen". It includes a menu bar (Text Module, Edit, Goto, Utilities, System, Help) and a toolbar. On the left, there are radio buttons for "Form", "Style", and "Text module", with "Text module" selected. Below these are buttons for "Display", "Change", and "Create". A text field contains "ZADDR\_FOOTER". An arrow points from the "Change" button to a second window below.

The second window, titled "Text Management", has tabs for "Text" and "Management". It features a toolbar with icons for editing and formatting. Below the toolbar, there are sections for "Paragraph formats" (set to "\* left-aligned") and "Character formats". The main text area contains the following content:

4 Truckee Way  
New York, NY 12345-678  
Telephone (212) 123-4567  
Fax (212) 123-4568\*

A callout box on the right lists the following items:

- Package
- Translation attributes
- Style



# Context - Including Text Modules



| Context | Inactive | Generated | Description    |
|---------|----------|-----------|----------------|
| BC480   |          |           |                |
| TEXT    |          |           | Text Node TEXT |

| Property                       | Value                               |
|--------------------------------|-------------------------------------|
| <u>General</u>                 |                                     |
| Name                           | TEXT                                |
| Description                    | Text Node TEXT                      |
| Status                         | Active                              |
| <u>Text</u>                    |                                     |
| Text Type                      | Text Module                         |
| <u>Text Module</u>             |                                     |
| Text Name                      | 'ZADDR_FOOTER'                      |
| Text Language                  | LAN                                 |
| No error if text not available | <input type="checkbox"/>            |
| Copy Style From Text Module    | <input checked="" type="checkbox"/> |
| Style                          |                                     |

**Determined at design time** (points to 'Text Module' and ''ZADDR\_FOOTER')

**Determined at runtime** (points to 'LAN')

# Context - Including Dynamic Texts



| Context          | Inactive | Generated | Description      |
|------------------|----------|-----------|------------------|
| BC480            |          |           |                  |
| CUSTOMER_ADDRESS |          |           | Customer address |

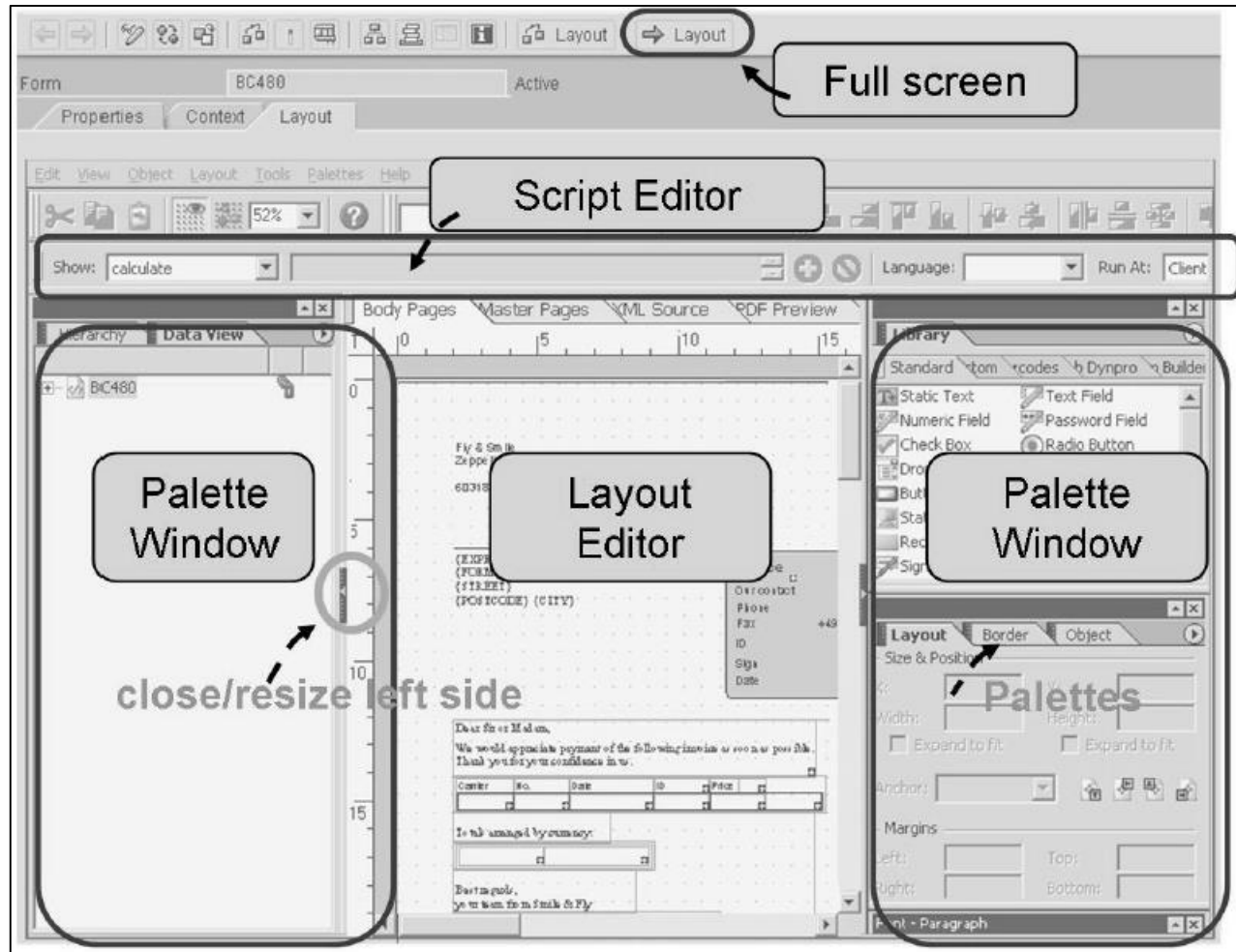
Properties Conditions

| Property            | Value            |
|---------------------|------------------|
| <u>General</u>      |                  |
| Name                | CUSTOMER_ADDRESS |
| Description         | Customer address |
| Status              | Active           |
| <u>Text</u>         |                  |
| Text Type           | Dynamic Text     |
| <u>Dynamic Text</u> |                  |
| Field               | GT_ADDRESS       |
| Text Language       |                  |
| Style               | BC480            |

Smart Style

Internal table of type TLINE

# Designer - Adobe Lifecycle Designer: Overview



# Designer - Adobe Lifecycle Designer: Overview




1. In SAP NetWeaver 2004s, a pushbutton Layout was added to transaction SFP, which displays Designer in a full screen
  2. The Designer workspace consists of four main areas. All but the central one (the Layout Editor) can be closed by choosing Palettes → Workspace (Palettes → Manage Palettes in some versions).
  3. In the top area, the Script Editor can be displayed. It allows you to enter scripts for calculations. You can choose between JavaScript and Adobe's FormCalc.
  4. The subdivisions of the left and right areas are called palette windows with further subdivisions of palettes
- Note: You can always return to the standard by choosing Palettes → Workspace → Reset Palette Locations.

# Designer - Toolbars

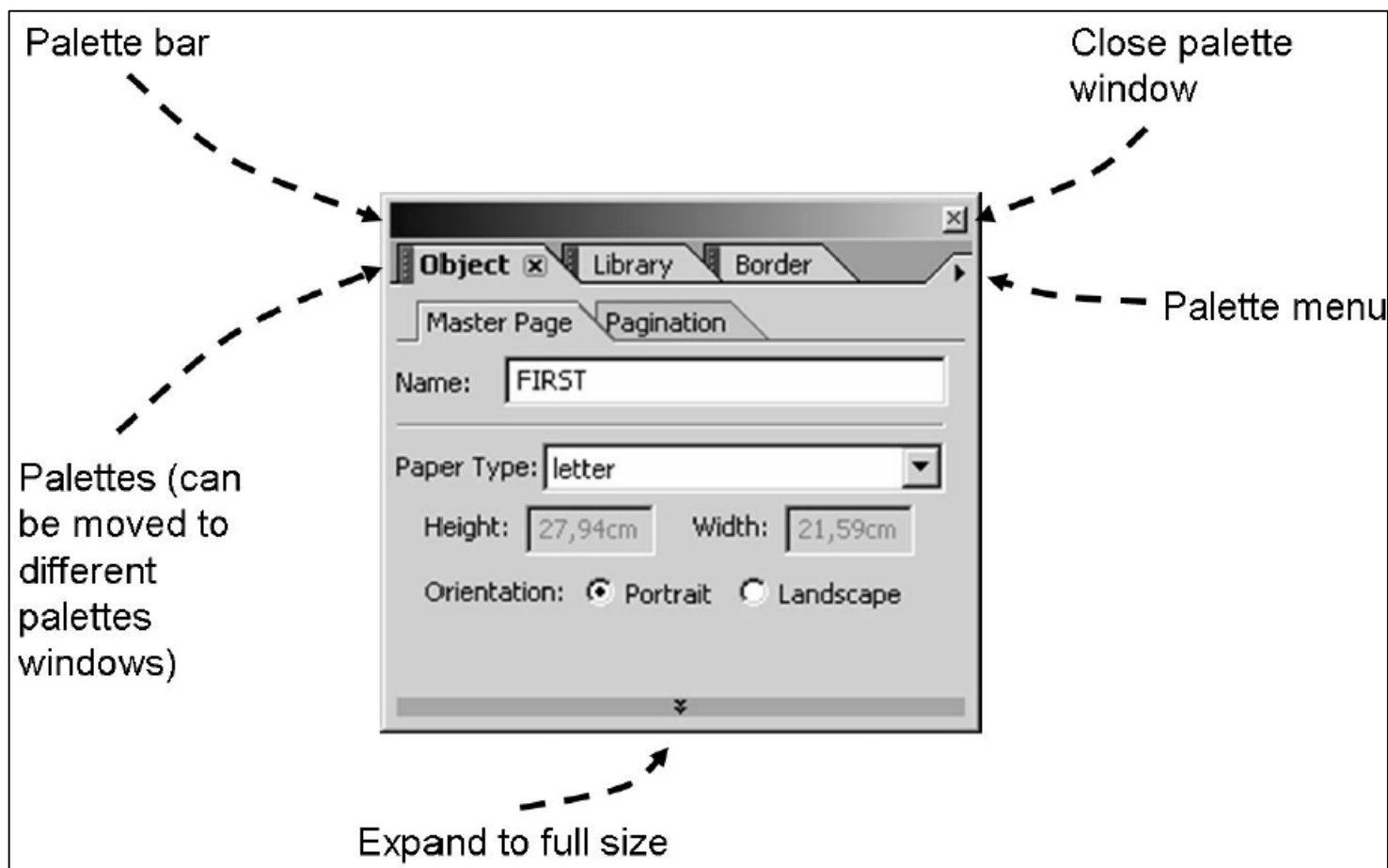


The screenshot shows the Adobe Designer toolbar with four main sections: Standard, Font, Paragraph, and Layout. The Layout section is circled, and an arrow points to a separate floating window titled 'Layout'. Below the screenshot, text instructions explain how to move and lock toolbars.

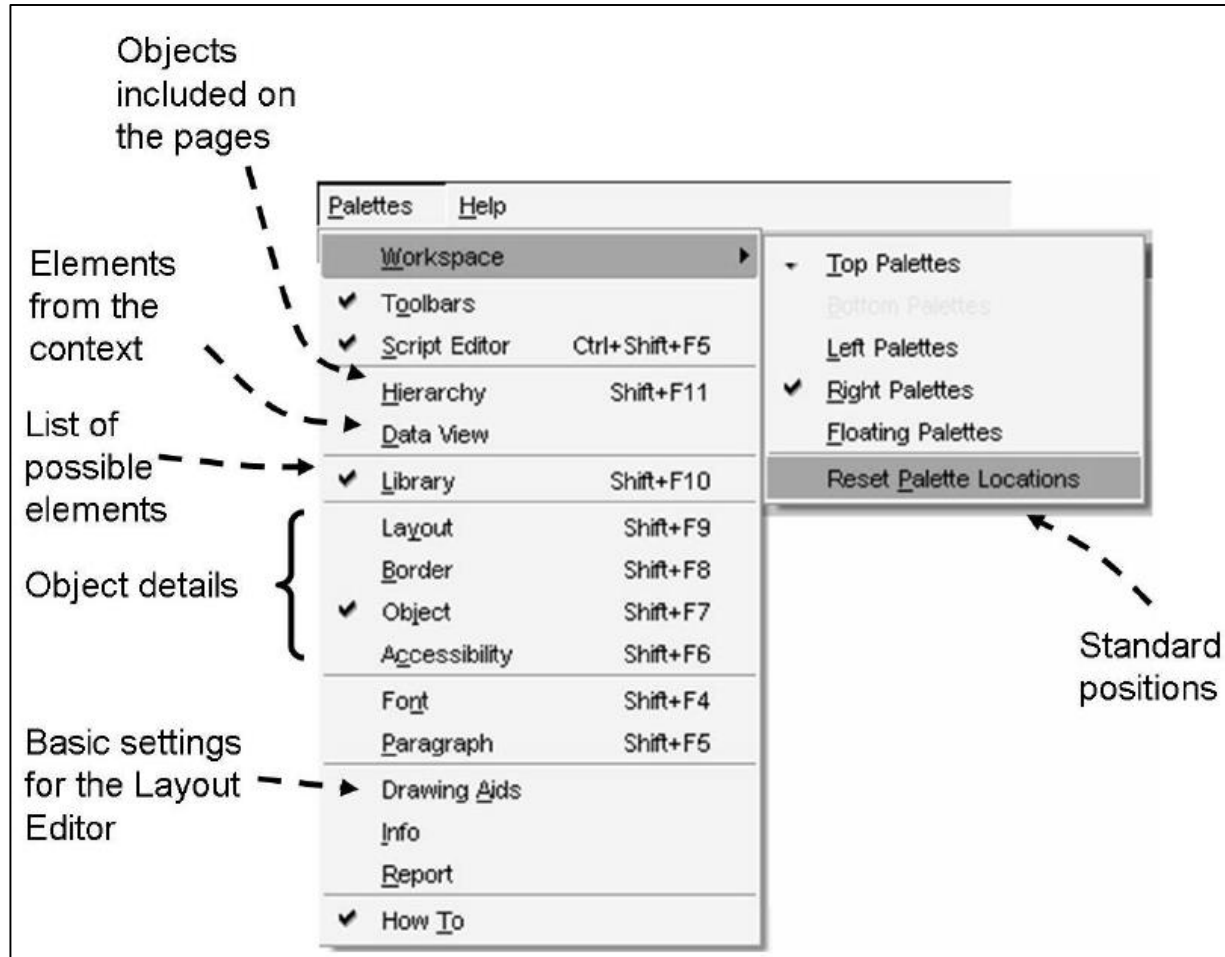
Click here to move toolbar  
Double-click to unlock toolbar

CTRL+click to move toolbar freely   
Double-click to let it snap back to its former position

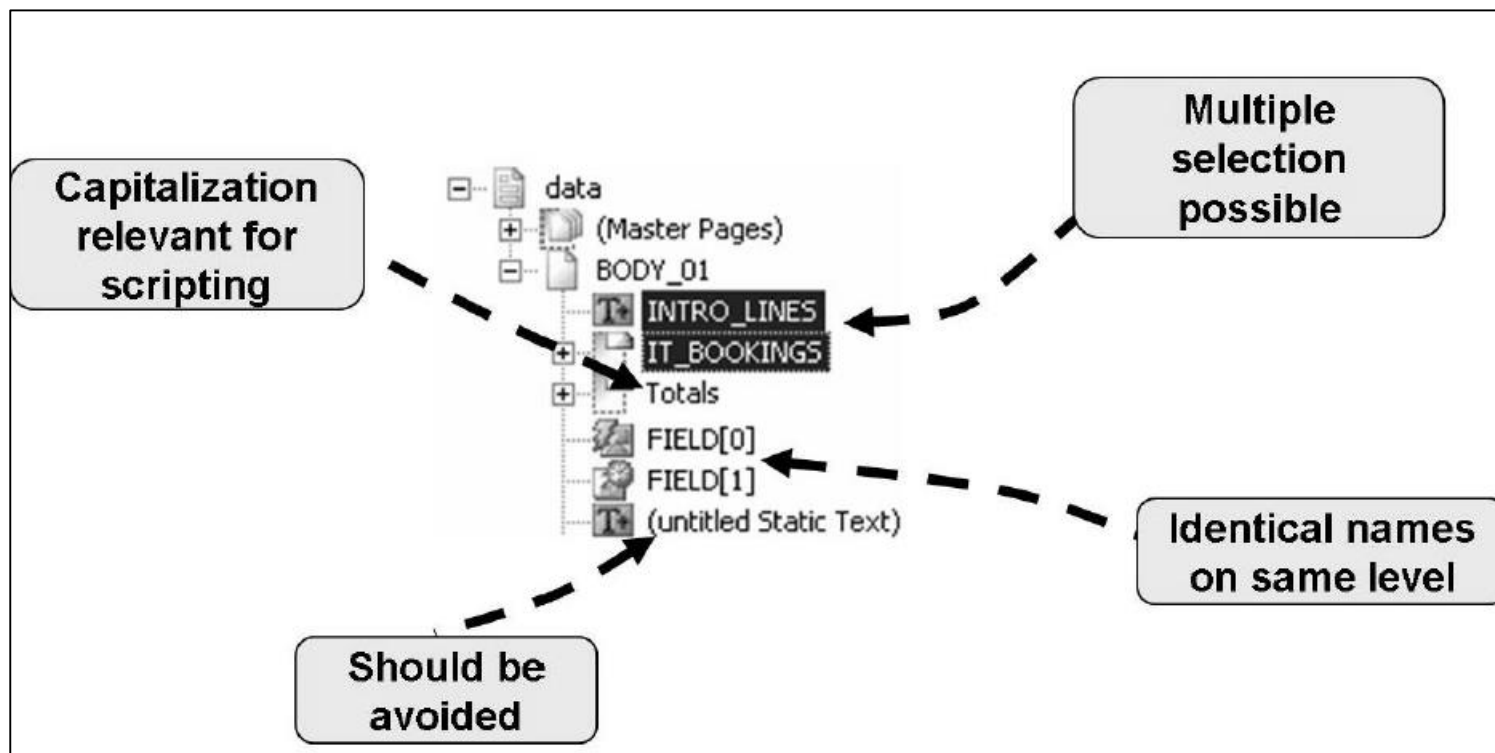
# Designer - Palette Windows: Handling



# Designer - Palettes: Overview

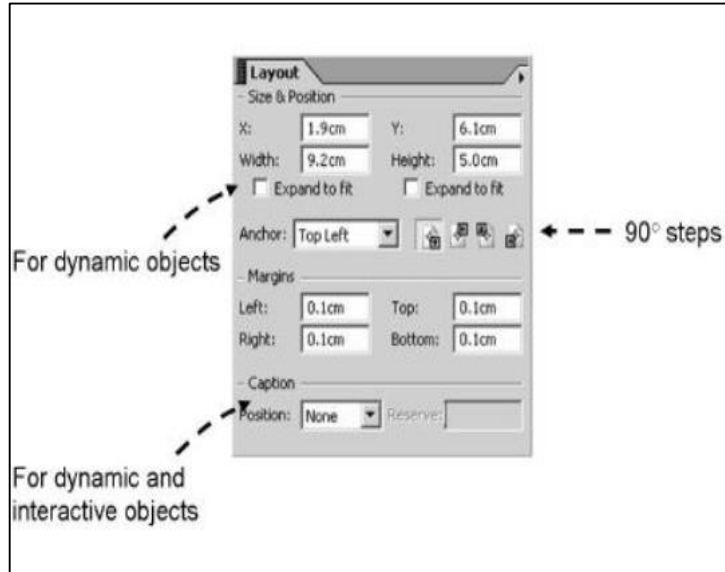


# Designer - The Hierarchy Palette





# Designer - The Layout Palette



1. You can position and size an object by clicking the resize handles of the Layout Editor and moving them. You can achieve the same by typing in the coordinates and the width/height in the Layout palette.

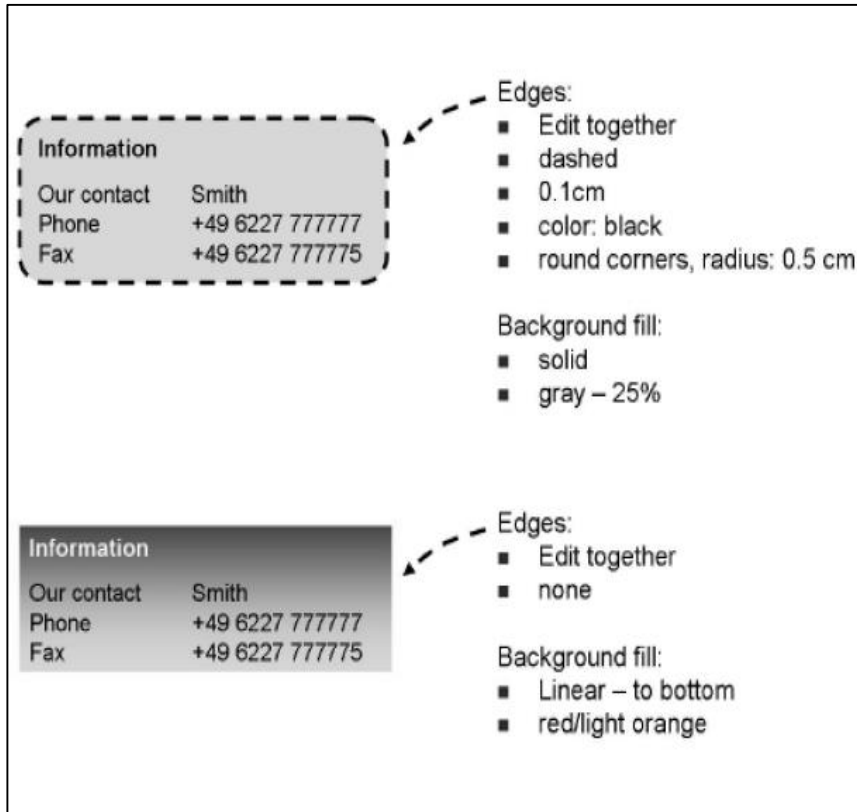
2. For dynamic elements (like dynamic texts that come via the application program) you can select Expand to fit for the width and/or height to avoid the disappearance of lines.

3. You can set the margins, for instance, the space between text and the borders of the text object.

4. Dynamic and interactive objects (like text fields or checkboxes) will normally need to have a caption. You can determine its size and its position with regards to the object itself.

5. Objects can be rotated in 90° steps. You must specify around which anchor point the object should be rotated

# Designer - Borders and Background Colors



1. On the Border palette, you can determine edges and/or background fills.

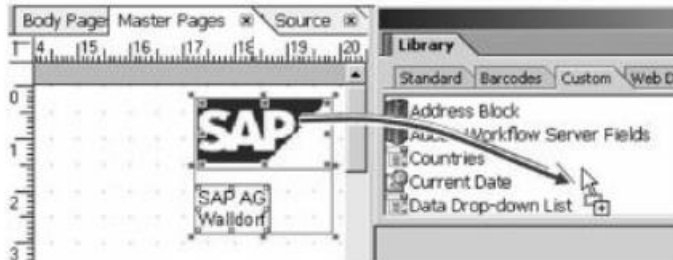
2. Edges can be edited together or individually.

3. For a background fill, you can choose between none, solid (one color) and various patterns for two colors.

4. For objects that are non-static (like a text field), you can also specify the border properties of the fillable areas. For example, you might choose to have a background color for a text field that differs from its caption color. To achieve this, select the object. In the Object palette, choose the Field tab. From the Appearance list, select Custom.



1. Mark element(s)
2. Drag and drop to desired *Library* tab

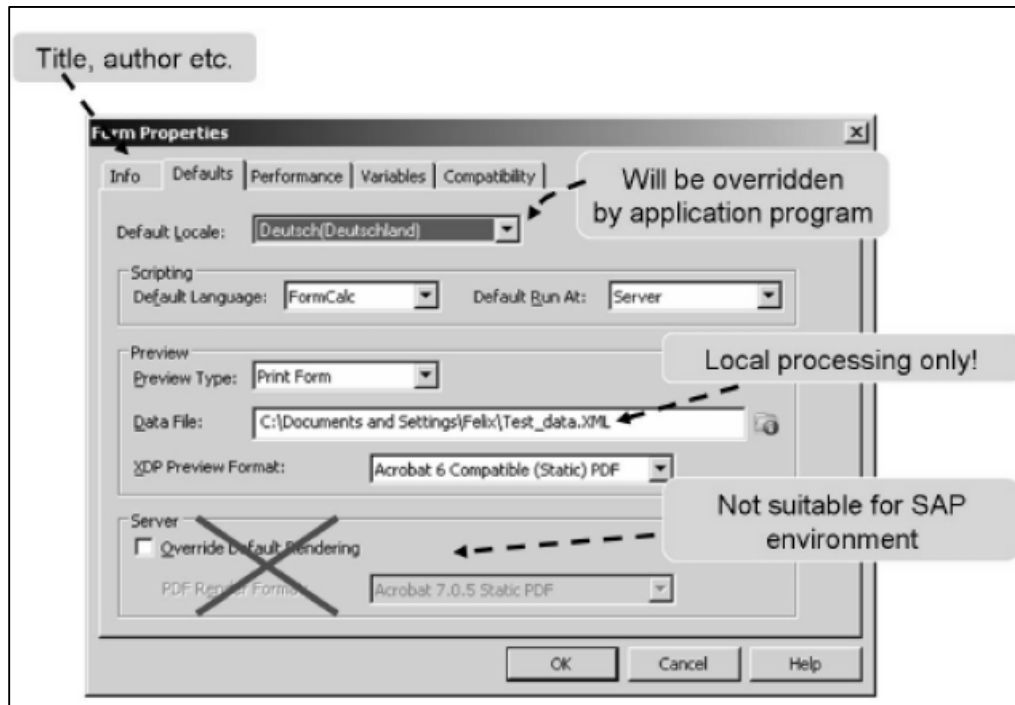


Enable object  
preview

Contains several  
individual objects

If you create an element and need to use it several times in your layout, it can be added to a tab page of the *Library palette*. You can then drag and drop your element from the *Library*, just like all predefined elements. All standard objects that come with Designer can be restored to any *Library tab* by selecting *Restore Standard Objects from the palette menu*. A library with its objects can also be published on a server

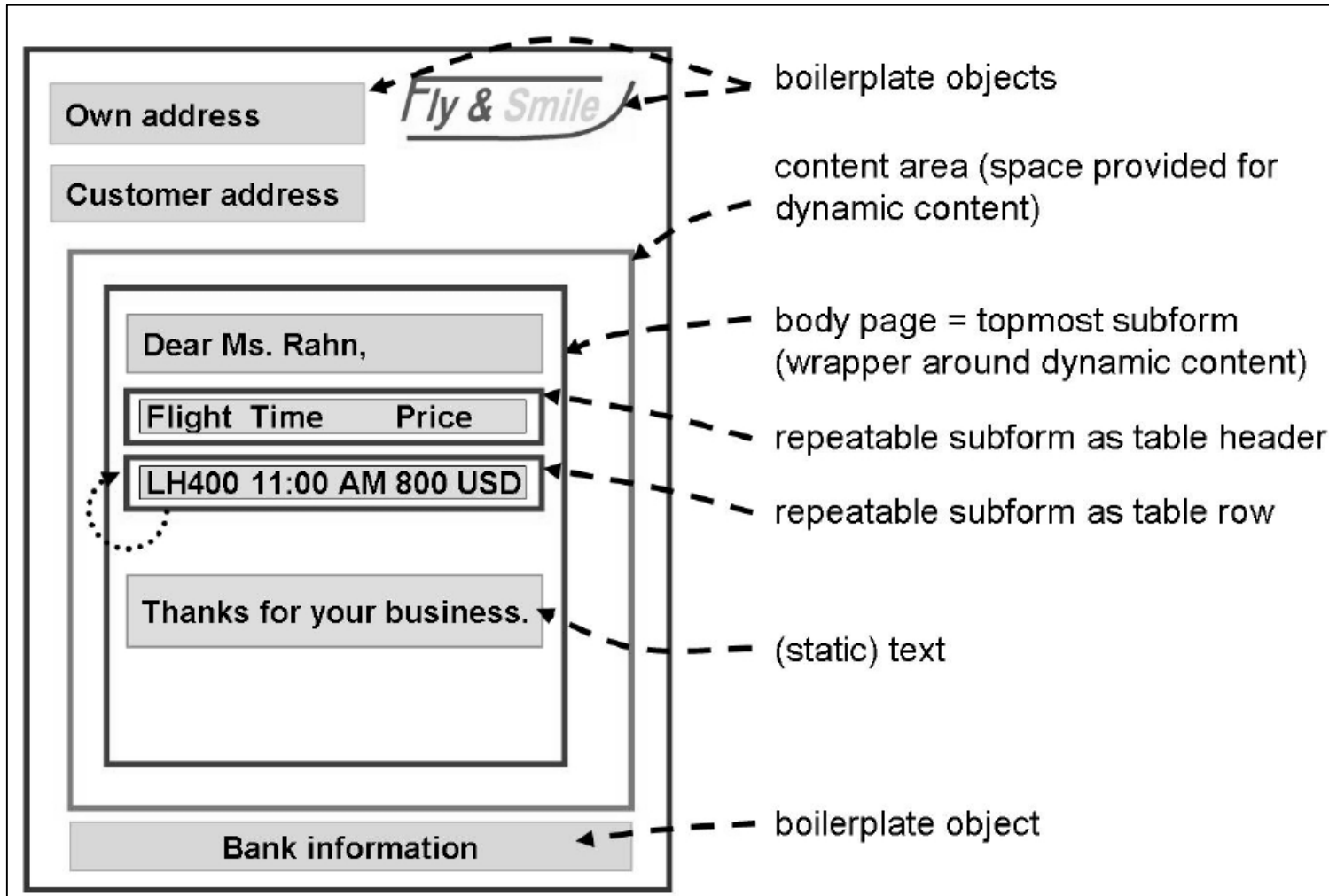
# Designer - Form Properties



# Designer



- 1.Master Page
- 2.Body Page
- 3.Content Area
- 4.Sub forms





Every form design contains at least one master page that Adobe Lifecycle Designer creates automatically.

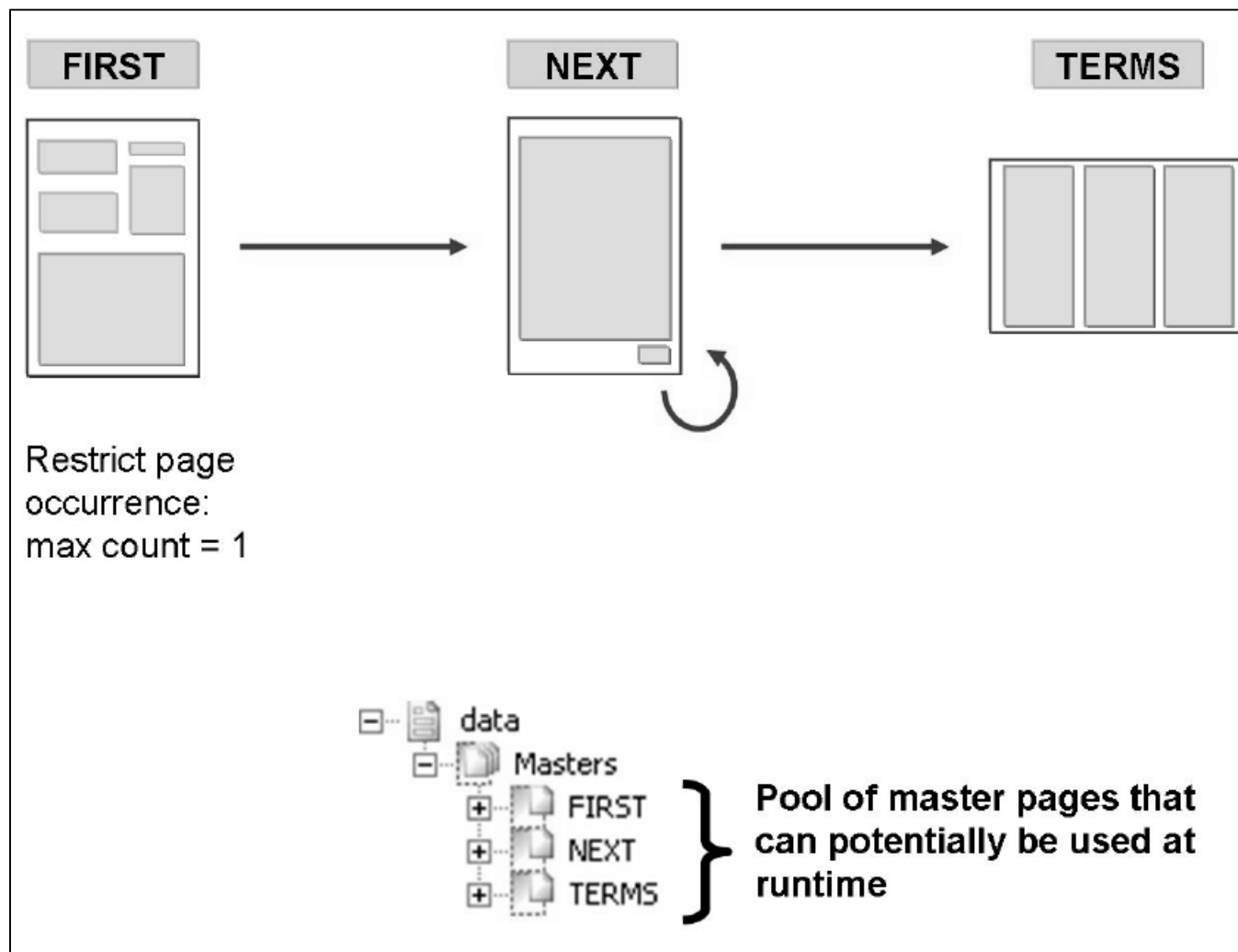
You can put objects on a master page that will appear on any resulting page at runtime, for example, your company logo. Such objects that never change are called boilerplate objects.

To some extent, the boilerplate objects of a master page could be compared to secondary windows in SAPscript or Smart Forms.

On a master page, you must include at least one content area. This defines the size to be used for dynamic output Content

Content areas can be included only on master pages.

# Designer - Inserting Several Master Pages





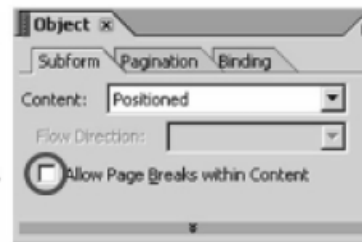
Subforms can be thought of as folders containing several objects. They can be used for the simple reason of keeping order in the Hierarchy, as it is possible to expand and compress subforms.

Subforms also help to rearrange several objects at a different place in the form.

## When to Include Subforms

### Placing objects in a subform makes sense:

- If you want to visually group objects
- If you want to keep objects together (protect them against page breaks)
- If you want to output the elements repeatedly (as table row or table header)
- If you want to hide several elements at once
- If you want to influence the screen reader order



# Designer - Types of Subforms



1.If of type *Positioned*, *objects of subforms can be laid down at their exact position at runtime, relative to the subform. For example, if a text field has been positioned at the top left corner of a subform of type Positioned, it will always be positioned at the top left corner of the subform, independent where on a page this subform is included. (The Hierarchy position of an object within a subform of type Positioned is irrelevant for its layout position.)*

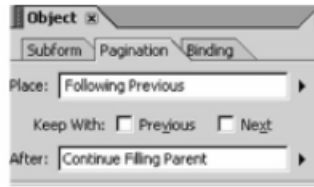
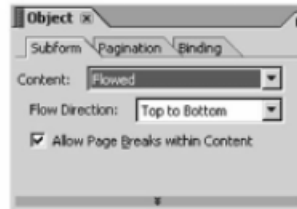
2.If of type *Flowed*, the objects will follow each other, depending on the space they require at runtime. A body page (as the topmost subform) is typically of this type

# Designer - Types of Subforms



## Subforms: containers for grouping several objects

- If of type *Positioned*, objects of subforms can be laid down at their exact position at design time. (Hierarchy position of objects is not relevant for layout position.)
- If of type *Flowed*, the objects will follow each other, depending on which space they require at runtime. Body pages (being top level subforms) are typically of type *Flowed*.





1. Insert static elements into a form: images, texts, and graphical objects
2. Set object properties for static form elements
3. Insert dynamic elements into a form: text fields, image fields, date/time fields, floating fields
4. Set the data binding (the connection between the layout fields and the business data)
5. Apply patterns (picture clauses) to influence field output
6. Insert tables into a form
7. Format tables
8. Set a header for a table

# Integration into ABAP Programs



Designing the layout of a form is the most time-consuming part of administering a printing scenario. However, a form itself cannot be run; it can only be previewed with test data. An ABAP program is required to process it.

```
* (1) Data retrieval and processing
SELECT ... FROM ...
...

* (2) Find out name of generated function module
CALL FUNCTION 'FP_FUNCTION_MODULE_NAME'...

* (3) Start form processing
CALL FUNCTION 'FP_JOB_OPEN'...

LOOP AT ...
* (4) Call function module dynamically
    CALL FUNCTION <generated function module> ...
ENDLOOP.

* (5) End form processing
CALL FUNCTION 'FP_JOB_CLOSE'...
```

## Sections of the ABAP Application Program

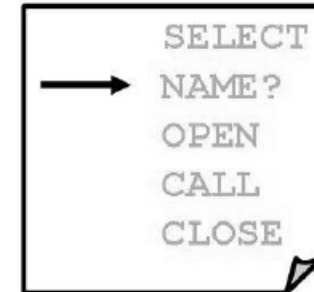
# Integration into ABAP Programs



## FP\_FUNCTION\_MODULE\_NAME

DATA:

```
form          TYPE fpwbformname,  
fm_name       TYPE rs381_fnam,  
interface_type TYPE fpinterfacetype.
```



\* (2) Find out name of generated function module

```
CALL FUNCTION 'FP_FUNCTION_MODULE_NAME'
```

```
EXPORTING
```

```
  i_name          = form
```

```
IMPORTING
```

```
  e_funcname      = fm_name
```

```
  e_interface_type = interface_type.
```



## Starting and Ending Form Processing

DATA:

```
    fp_outputparams    TYPE sfpoutputparams.
```

...

\* (3) Start form processing

```
CALL FUNCTION 'FP_JOB_OPEN'...
```

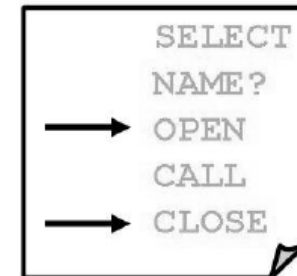
\* set output parameters like printer, preview...

```
    CHANGING ie_outputparams = fp_outputparams ...
```

...

\* (5) End form processing

```
CALL FUNCTION 'FP_JOB_CLOSE'...
```

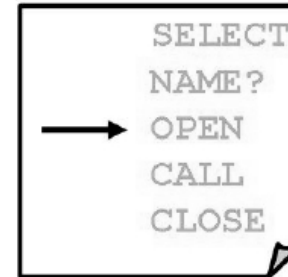




## Parameters of FP\_JOB\_OPEN

**Type SFPOUTPUTPARAMS**  
**of changing parameter IE\_OUTPUTPARAMS**

|                 |                                |
|-----------------|--------------------------------|
| <b>nodialog</b> | no printer dialog popup        |
| <b>noprint</b>  | no backend printing            |
| <b>nopdf</b>    | no PDF document, only PDL      |
| <b>getpdf</b>   | PDF as return parameter        |
| <b>dest</b>     | output device                  |
| <b>copies</b>   | number of copies to be printed |
| <b>reqnew</b>   | start a new spool job          |
| <b>reqfinal</b> | spool request completed        |







## Calling the Generated Function Module

```
DATA:
    fp_docparams TYPE sfpdocparams,
    fm_name      TYPE rs381_fnam,
    fp_result    TYPE TYPE fpformoutput, ...

LOOP AT ...
    fp_docparams-langu = customer_language.

    * (4) Call function module dynamically
    CALL FUNCTION fm_name
        EXPORTING
            /lbcdwb/docparams = fp_docparams
            it_bookings       = gt_bookings
        IMPORTING
            /lbcdwb/formoutput = fp_result
        EXCEPTIONS
            OTHERS              = 1.
    ENDLOOP.
```

# Summary



In this lesson, you have learnt:

- The Adobe Forms Architecture
- The Interface, context and Layout
- How to integrate Adobe Form in an ABAP application program



# Review Question



Question 1: The master page is created automatically.

- True/False

Question 2: \_\_\_\_\_ help to rearrange several objects at a different place in the form.

- True/False

