

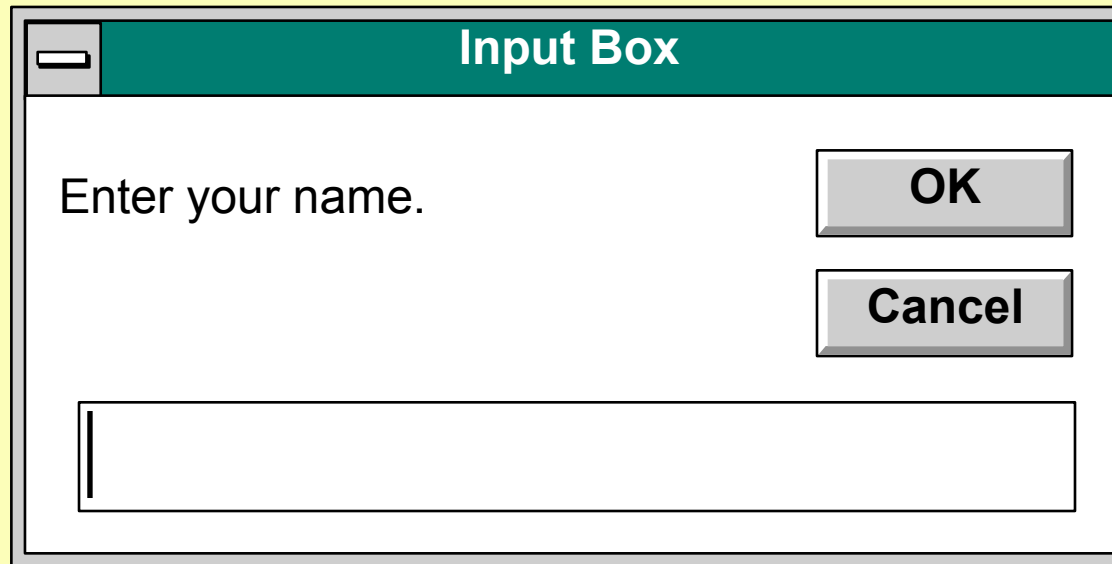
Module 1: Input Validation

1

Overview

Getting the Input
Validating the Input

Using the InputBox\$ Function

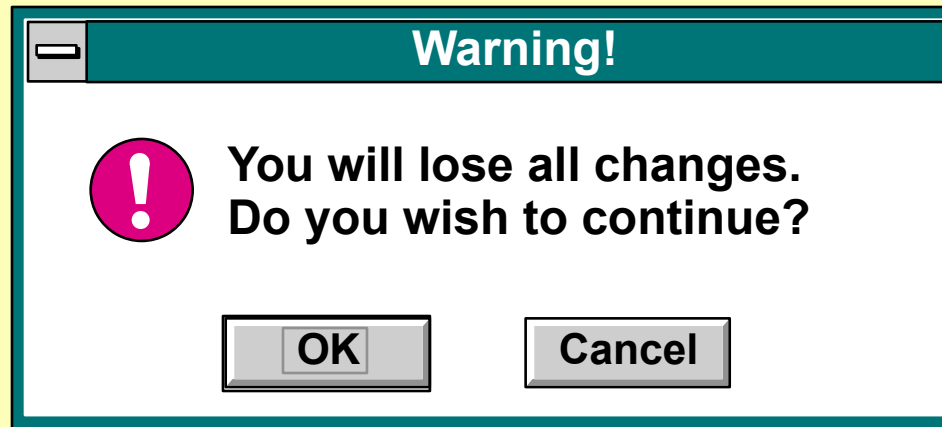


```
UserTypeIn = InputBox$ (MsgText, MsgTitle, Default)
```

Displaying Text with a Label or Message Box

Hello

```
Label1.Caption = "Hello"
```



```
Response = MsgBox (MsgText, MsgType, MsgTitle)
```

Controlling the Tab Order

TabIndex Values

The diagram illustrates a form titled "Form1" with a tab order sequence of 8 controls. The controls and their assigned TabIndex values are as follows:

TabIndex	Control
0	Name: (text label)
1	(text input field for Name)
2	(form border)
3	Number: (text label)
4	(text input field for Number)
5	City: (text label)
6	OK (button)
7	Cancel (button)

Assigning Access Keys

Form1

Name:

Number:

City:

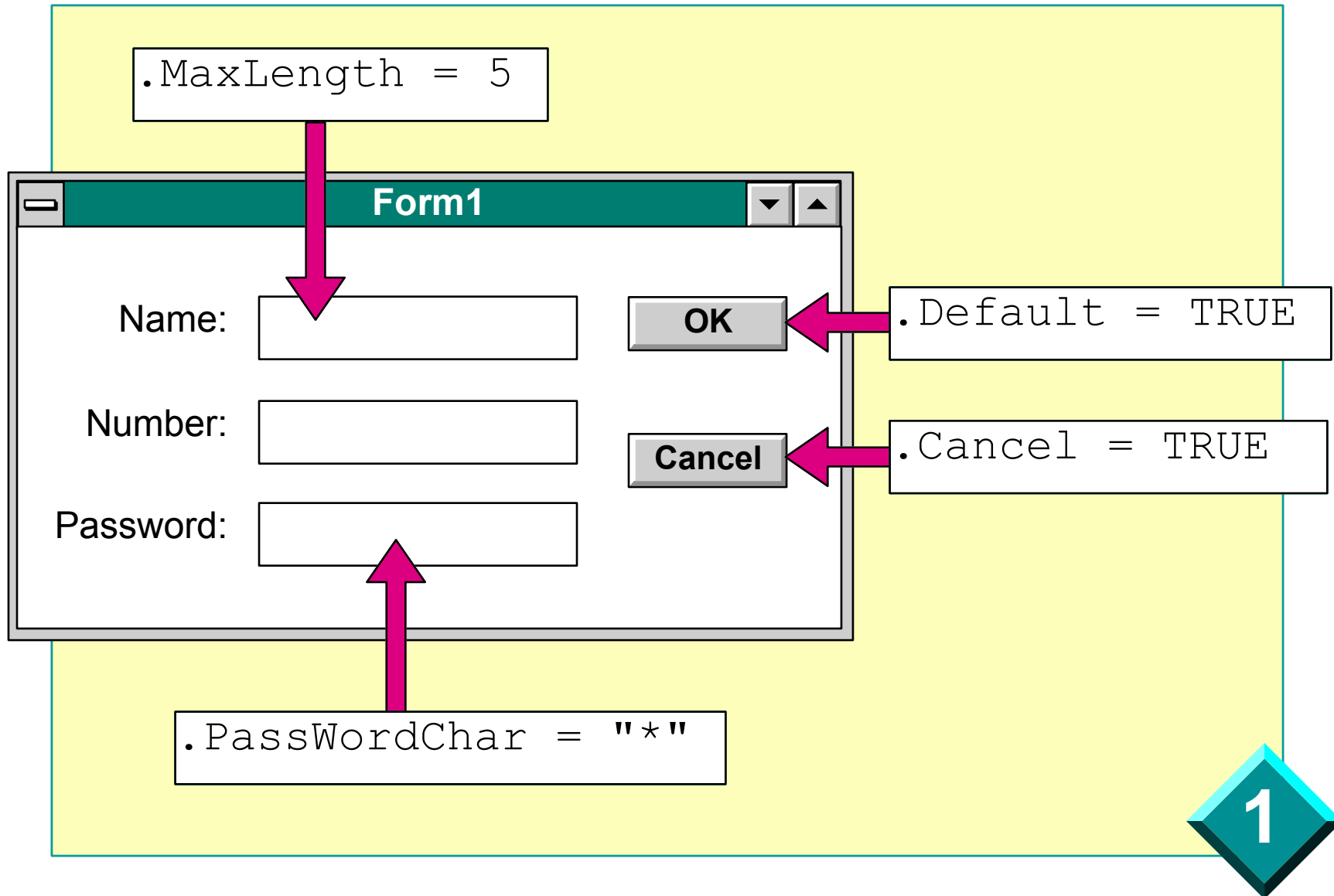
OK

Cancel

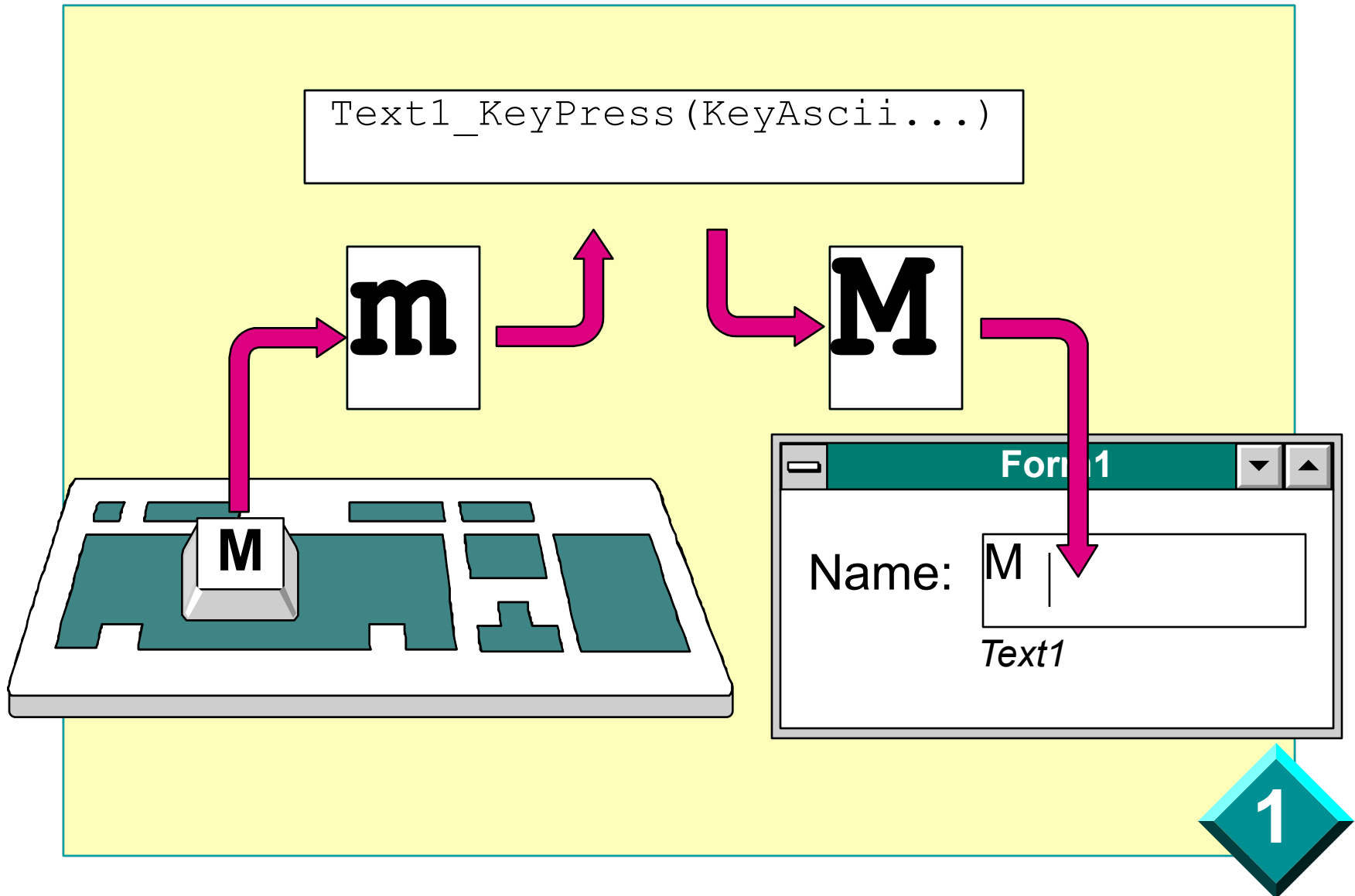
`Label1.TabIndex = 4`

`Text3.TabIndex = 5`

Using Properties to Restrict Data Entry

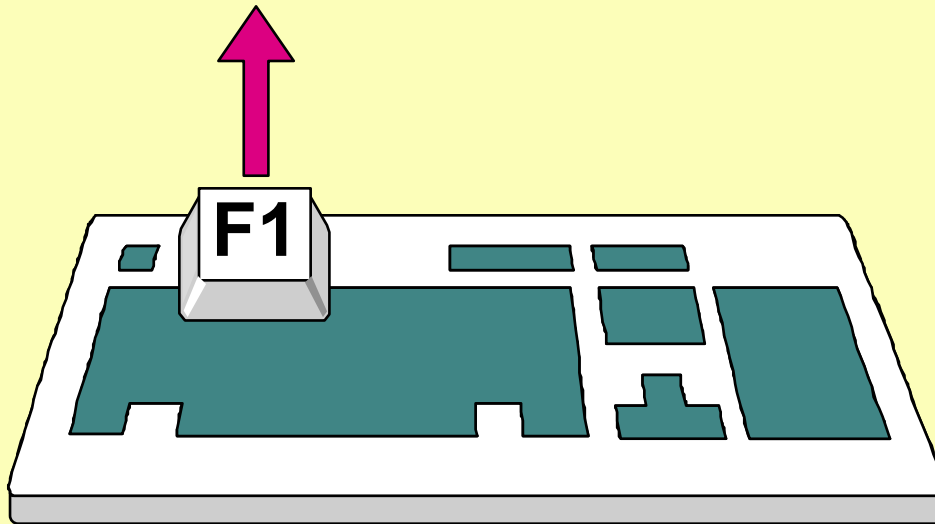


Using KeyPress to Restrict Keystrokes



Using KeyUp and KeyDown to Trap Keystrokes

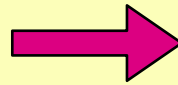
```
Text1_KeyDown(KeyCode...) 'Check Key
```



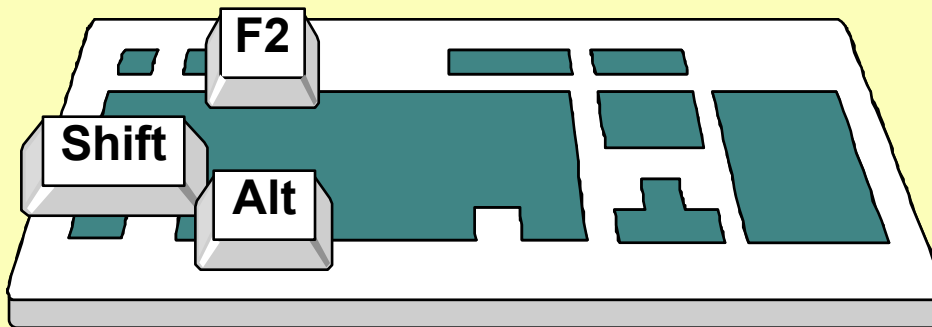
A Form-Level Keyboard Handler

```
Form1.KeyPreview = True
```

```
Form_KeyDown
```



```
Text1_KeyDown
```



Using ENTER Key to Tab Between Controls

Form1

0 Name: 1

2 Number: 3

```
Form_KeyPress  
'If ENTER key pressed  
'Check TabIndex  
'Set new TabIndex  
'Set focus
```

The Masked Edit Control

The diagram illustrates a form titled "Form1" with five input fields and two buttons. Each input field is connected by a pink arrow to a corresponding code snippet in a box on the right. The fields and their associated code are:

- Name:** `.Mask = &&&&&`
- State:** `.Mask = ??`
- Number:** `.Mask = ##`
- Soc-Sec-No:** `.Mask = ###-##-####`
- Empty field:** `lblError.Caption
 ↪= "Error Message"`

At the bottom of the form are two buttons: **OK** and **Cancel**.

The ValidationError Event

ValidationError Event

MaskedEdit1_ValidationError

↳ (InvalidText as String, StartPosition as Integer)

Analyze InvalidText

Analyze StartPosition

String Functions

Function	Purpose
LTrim\$	Deletes left trailing spaces
RTrim\$	Deletes right trailing spaces
Trim\$	Deletes left and right trailing spaces
Mid\$	Returns a string that is part of another string
Left\$	Returns a string consisting of the <i>n</i> leftmost characters of a string
Right\$	Returns a string consisting of the <i>n</i> rightmost characters of a string

String Functions *(continued)*

Function	Purpose
Chr\$	Converts ASCII to single character
Val	Converts String to Numeric
Format\$	Converts Numeric to Formatted String as specified
Str\$	Converts Numeric to String
Instr	Searches Target String for matching Search String: Returns Position
Hex\$/Oct\$	Returns a string consisting of the converted numerical value

Enabling the OK Button

```
Sub ValidateOKButton  
    'Verify Controls
```

Form1

Name:

Number:

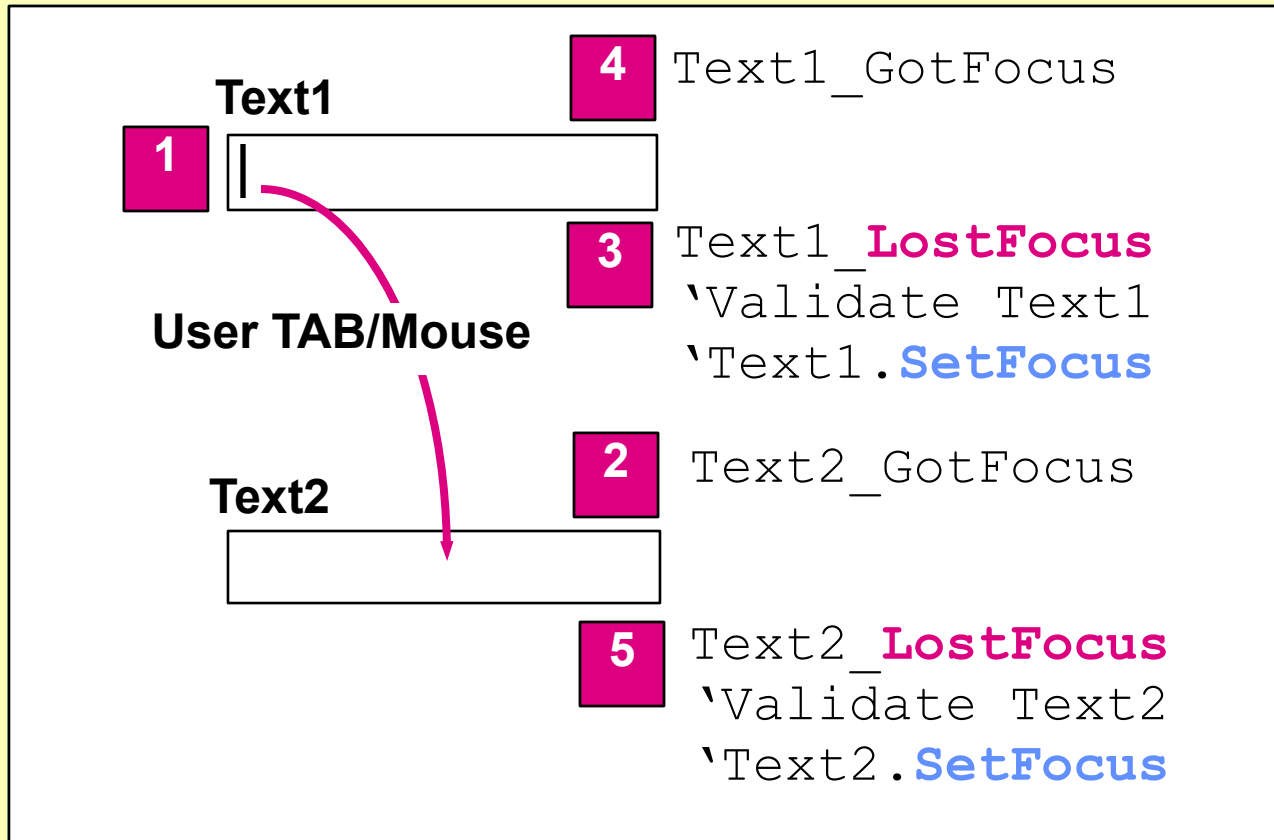
```
mskName_Change  
    ValidateOKButton mskName
```

```
mskNumber_Change  
    ValidateOKButton mskNumber
```

Disable at Design Time

Competing for Focus

Do Not Use SetFocus in LostFocus Event



Validate Controls

Form1

Name:

Number:

`txtName_LostFocus`
`'Set flag if valid`

`txtNumber_LostFocus`
`'Set flag if valid`

`cmdOK_Click`
`'Check Flags`

Detecting Form Focus

**A Form Receives a LostFocus Event Only If
There Are No Controls on the Form**

A Form Receives an Activate or Deactivate Event

Input Validation Hierarchy

Keystroke Validation

Masked Edit Control - Mask Setting;
ValidationError

KeyPress, KeyUp, KeyDown, Change Events

Input Field Validation

MaxLength Property

ValidationError Event; LostFocus Event

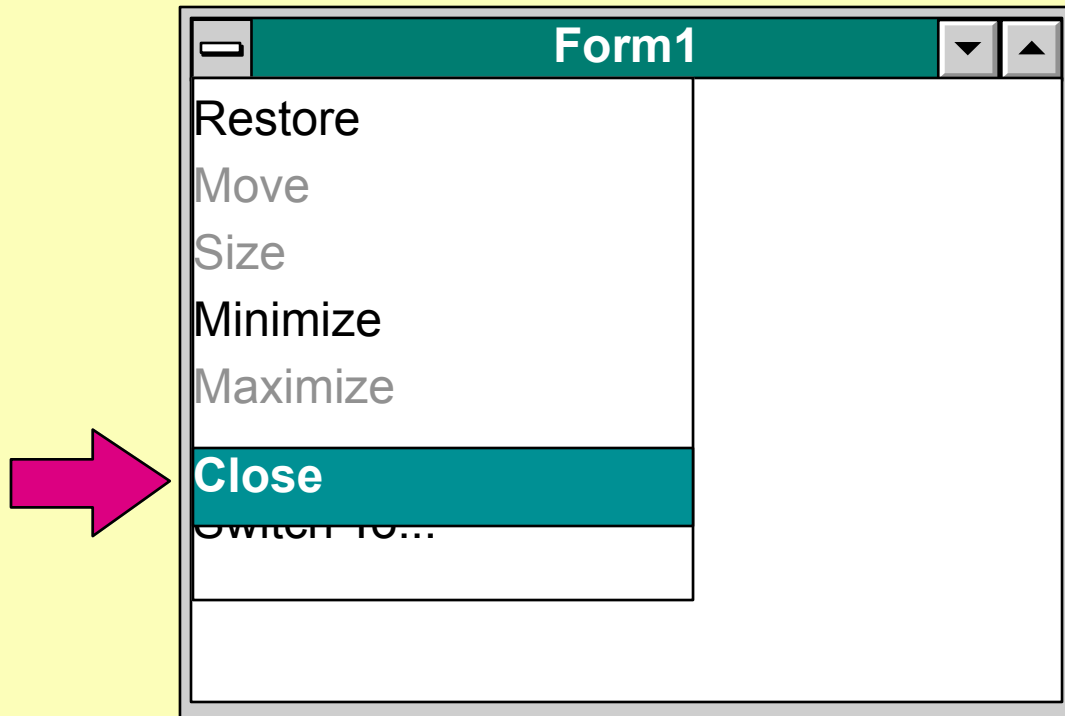
String Analysis

Ensemble Validation

Flags Array

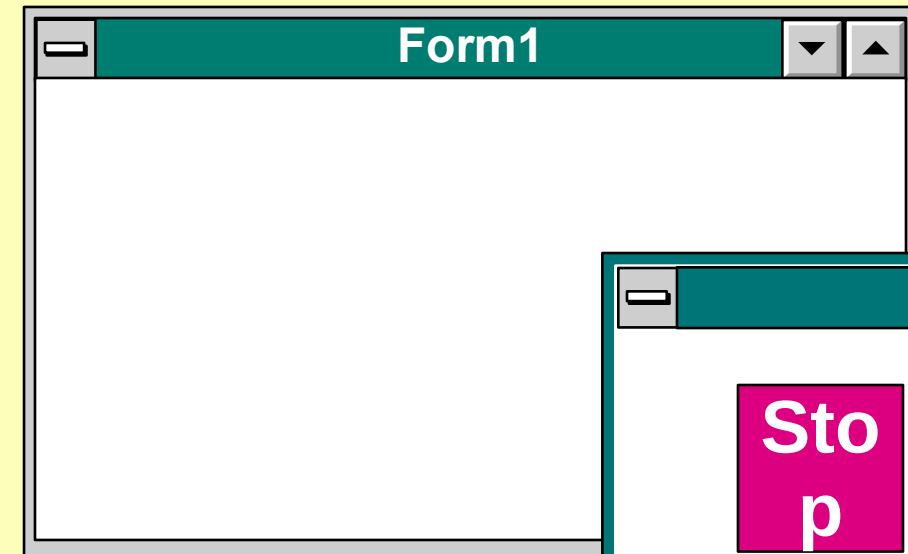
Using QueryUnload

```
Sub Form_QueryUnload (Cancel As Integer,  
    ↪ UnloadMode As Integer)  
    If UnloadMode = FORM_CONTROLMENU Then...
```

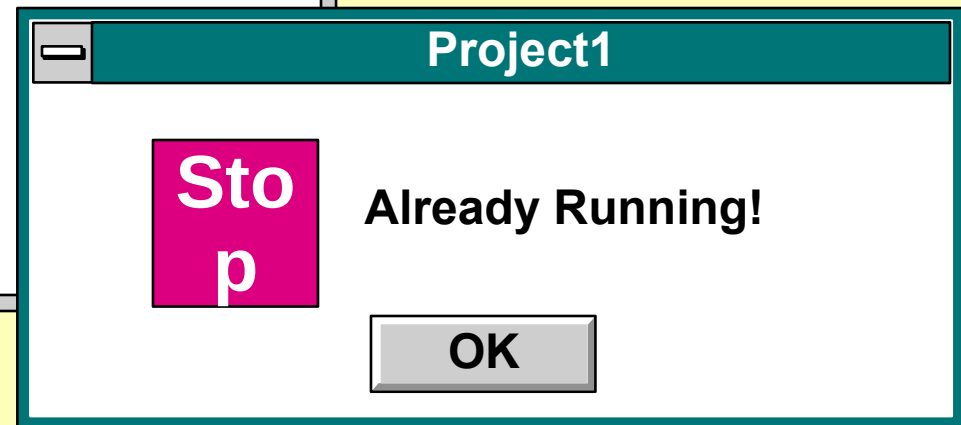


Checking the App Object

```
Sub Form_Load ()  
    If App.PrevInstance...
```



MYPROG.EXE



MYPROG.EXE

1

Summary

Getting the Input
Validating the Input