Validation Controls

Chapter 7

What Are You Going To Learn?

- At the end of this lesson, you will be able to:
  - Define validation controls.
  - Differentiate different types of validation controls.
  - Apply six types of validation controls.
  - Set other available properties.

What is validation control?

- Validation server control is used to validate the data of an input control.
- If the data does not pass validation, it will display an error message to the user.
- Each validator is associated with a single input control, but you can associate two or more validators with the same input control.

Process validation

- Validation tests are typically done on the client before the page is posted to the server. **Why?**

- Validation is always done on the server too. **Why?**

Six types of validation controls

RequiredFieldValidator	• Ensures that a field is not left empty
CompareValidator	• Compares the entered text with a value or with another control • Check data type
RangeValidator	• Ensures the entered text is within a specified range

Six types of validation controls

RegularExpression Validator	• Matches the entered text against a specified pattern
CustomValidator	• Runs custom code to validate the entered text
ValidationSummary	• Provides a place to summarize error messages

Common validator properties

ControlToValidate	• The ID of the control to be validated.
Display	• Determine how the error message is to be displayed. • Value = none, static or dynamic.
ErrorMessage	• The message that's displayed in the validator and/or the ValidationSummary control when the validation fails.
Text	• The message that's displayed in the validator.

Common validator properties

Enabled	• Indicates whether the validation control is enabled.
EnableClientScript	• Indicates whether the validation will be done on the client.
ValidationGroup	• Indicates which group the validation is part of.
IsValid	• Indicates whether the control passed the validation.

Properties of ValidationSummary

DisplayMode	• Specifies how the error messages from the validation controls are to be displayed.
HeaderText	• The text that's displayed before the list of error messages.
ShowSummary	• A Boolean value that determines whether the validation summary should be displayed on the web page.

Properties of ValidationSummary

ShowMessageBox	• A Boolean value that determines whether the validation summary should be displayed in a message box.

Validation Group

- Validation controls can be grouped together, allowing you to perform validation against a selection of controls.
- All controls with the same ValidationGroup are checked at the same time, which means that controls that are not part of that group are not checked.

Page.IsValid

- Use the IsValid property to determine whether all the input passed the validation.

CausesValidation property

- Determine whether validation should be done when a particular control is being clicked.
- Default is true.

SetFocusOnError property

- Determines whether client-side script gives the focus to the first control that generated an error.
- Default if False.

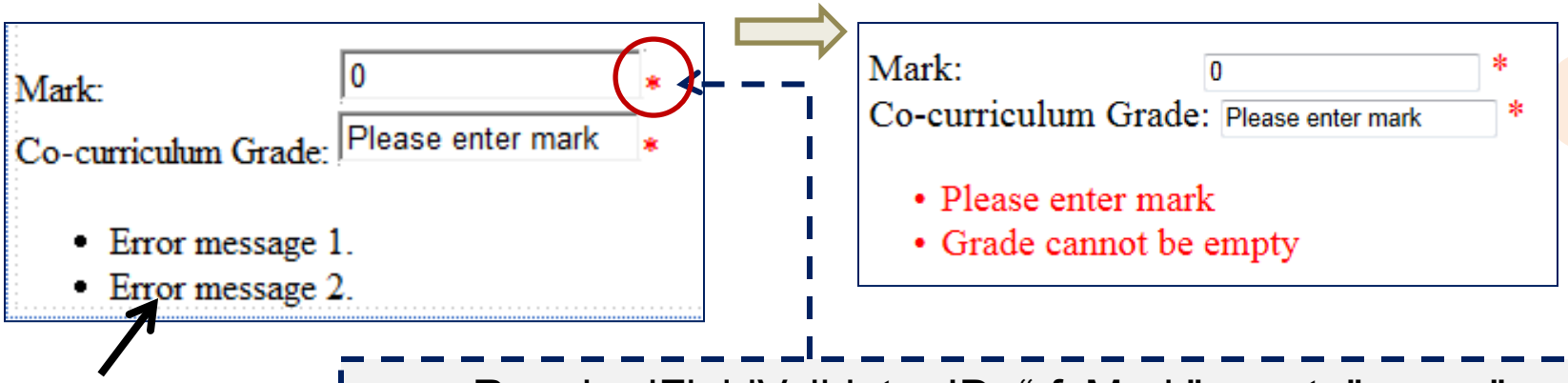
EnableClientScript property

- Gets or sets a setting that determines whether the control provides validation at the client.
- The default is True.

property of the RequiredFieldValidator

InitialValue

- The initial value of the control that's validated. If this value isn't changed, the validation fails.
- The default is an empty string.



ValidationSummary

```
<asp:RequiredFieldValidator ID="rfvMark" runat="server"
ErrorMessage="Please enter mark" InitialValue="0"
ControlToValidate="txtMark">*</asp:RequiredFieldValidator>
```

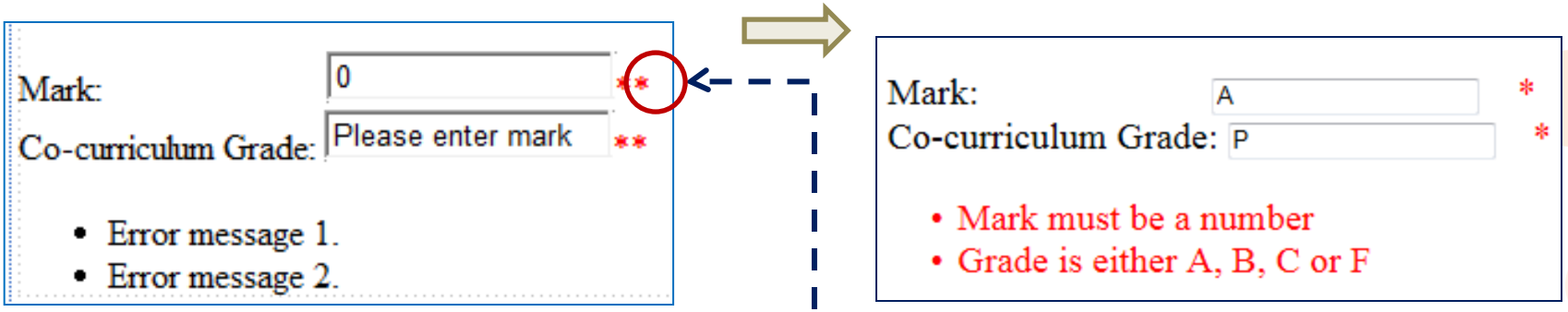
properties of the CompareValidator

ValueToCompare	• The value that the control specified in the ControlToValidate property should be compared to.
Operator	• The type of comparison to perform. • value = Equal, NotEqual, LessThan, GreaterThan, LessThanEqual,GreaterThanEqual or DataTypeCheck
Type	• The data type to use for the comparison. • Value = String, Double, Integer, Date or Currency

properties of the CompareValidator

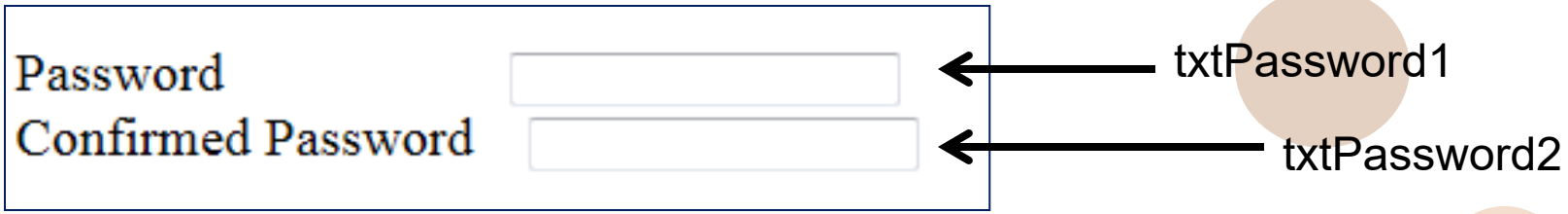
ControlToCompare

- The ID of the control that the value of the control specified in the ControlToValidate property should be compared to.




```
<asp:CompareValidator ID="cvMark" runat="server"
ErrorMessage="Mark must be a number" Operator="DataTypeCheck"
Type = "Double"
ControlToValidate="txtMark">*</asp:RequiredFieldValidator>
```

Questions



Assume that the above is part of a Change Password page. Suggest:

1. TWO different validations that should be done.
2. Demonstrate the code on how to validate whether 2 passwords are matched using the appropriate validation control in ASP.NET.

properties of the RangeValidator

MinimumValue	• The minimum value allowed for the control.
MaximumValue	• The maximum value allowed for the control.
Type	• The data type to use for the comparison.

Example

Assume that the above is part of an Assessment page. Suggest:

1. TWO different validations that should be done for Mark field.
2. Demonstrate the code on how to validate the above using the appropriate validation control(s) in ASP.NET, so that the mark and grade must be within the given ranges.

Common regular expression elements

.	• Ordinary character – Matches any character/string
\	• Matches the character that follows.
\d	• Matches any decimal digit (0-9).
\D	• Matches any character other than a decimal digit.

Common regular expression elements

\w	• Matches any word character (a-z, A-Z, and 0-9).
\W	• Matches any character other than a word character.
\s	• Matches any white space character (space, tab, new line, etc.).
\S	• Matches any character other than a whitespace character.

Common regular expression elements

[abcd]	• Matches any character included between the brackets.
[^abcd]	• Matches any character that is not included between the brackets.
[a-z]	• Matches any characters in the indicated range.
{n}	• Matches exactly *n* occurrences of the preceding element or group.

Common regular expression elements

{n,}	• Matches at least *n* occurrences of the preceding element or group.
{n,m}	• Matches at least *n* but no more than *m* occurrences of the preceding element or group.
*	• Matches zero or more occurrences of the preceding element.
?	• Matches zero or one occurrences of the preceding element.

Common regular expression elements

+	• Matches one or more occurrences of the preceding element.
|	• Matches any of the elements separated by the vertical bar.
( )	• Groups the elements that appear between the parentheses.

Metacharacter-+?*|

- **Examples:**
- [hc]+at matches "*hat*", "*cat*", "*hhat*", "*chat*", "*hcat*", "*ccchat*", and so on, but not "*at*".
- [hc]?at matches "*hat*", "*cat*", and "*at*".
- [hc]*at matches "*hat*", "*cat*", "*hhat*", "*chat*", "*hcat*", "*ccchat*", "*at*", and so on.
- cat|dog matches "*cat*" or "*dog*".

Examples of regular expressions

```
(AB|SB)-\d{1,5}
```

- The letters AB or SB, followed by hyphen and a one- to five-digit number. (e.g. SB-3276)

```
\d{5}(-\d{4})?
```

- A five-digit number, optionally followed by a hyphen and a four-digit number. (93711-2765)

Examples of regular expressions

```
\w*\d\w*
```

- A text entry that contains at least one numeral. (arm01)

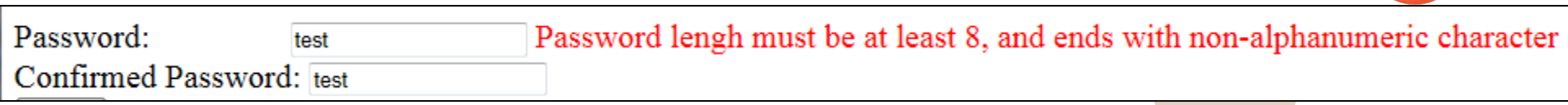
```
[xyz]\d{3}
```

- The letter x, y, or z, followed by a three-digit number. (x023)

Interpret the regular expressions below and provide examples of VALID input

- \d{5}
- \w{8, }
- \d{3}-\d{7,8}
- \w{1,8}\.[a-z]{3,4}
- (WAD|WAB)\d{8}

Example




```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server“  ControlToValidate=“txtPassword1"
ErrorMessage="Password length must be at least 7, with 1 non-
alphanumeric character"
ValidationExpression=“                                          ">
</asp:RegularExpressionValidator>
```

Given the code for the RegularExpressionValidator above, complete the ValidationExpression value so that the password length must be at least 7, and one non-alphanumeric character at the end

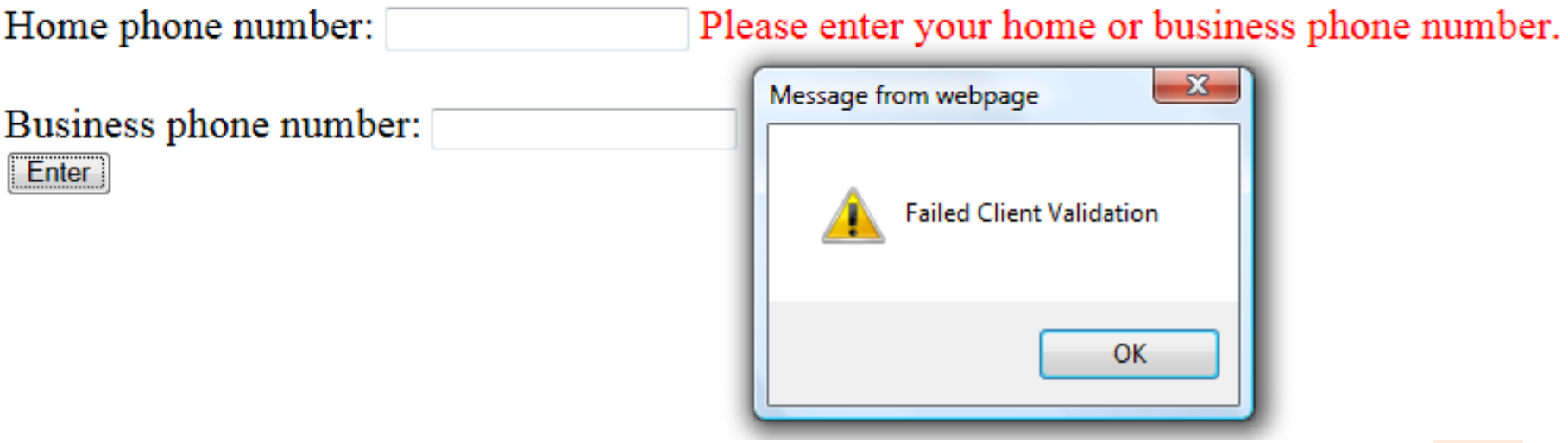
properties of the CustomValidator

ClientValidation Function	• To specify the client-side validation function that should be called to perform the validation at client.
OnServerValidate	• To specify the server side event handler that should be called when server side validation is performed.

Properties of the ServerValidateEventArgs Class

Value	• the text string to be validated.
IsValid	• A Boolean property that you set to True if the value passes the validation test or to False otherwise.

Example



Write a custom validation if you would like to customize your own validation. For example, require the user to enter at least home phone number or business phone number.

Sample Code (Sample.aspx)

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
    ErrorMessage= " enter home or business phone number."
    ClientValidationFunction=“ClientValidatePhoneNo"
    onservervalidate=“ServerValidatePhoneNo">
    EnableClientScript="true"
</asp:CustomValidator>
```

Setting the value to “true” would invoke client script (JavaScript); value “false” for server code

Server code function

JavaScript function

Client Script Validation (Sample.aspx)

```
<script type="text/javascript">
function ClientValidatePhoneNo(source, args)
{
    var txtHome = document.getElementById('<%= txtHome.ClientID %>');
    var txtBusiness = document.getElementById('<%= txtBusiness.ClientID %>');
    if (txtHome.value != " " || txtBusiness.value != " ")
    {       window.alert("Passed client validation");
            args.isValid = true; }
     else
    {       window.alert("Failed client validation");
            args.IsValid = false;}
}
</script>
```

Server Validation (Sample.aspx.cs)

```
void ServerValidatePhoneNo(object source, ServerValidateEventArgs args)
   {
      if (txtHome.Text != string.Empty || txtBusiness.Text != string.Empty)
      {
         Response.Write(“Passed server validation");
         args.IsValid = true;
      }
      else
      {
         Response.Write(“Failed server validatioin");
         args.IsValid = false;
      }
   }
```