TOPIC 6

Asp.Net Advance Controls and JavaScript

LEARNING OUTCOME

By the end of this topic, students will be able to:

- Design Multi Views Web Pages
- Use Properties of View and MultiView Controls
- Create Panel Controls
- Work with the Panel and Ajax Controls
- Implement UpdatePanel Control
- Apply Properties and Methods of the UpdatePanel Control
- Describe UpdateProgress Control
- Use Properties and Methods of the UpdateProgress Control
- Use Timer Control in Web Project

Introduction

MultiView and View controls allow you to divide the content of a page into different groups, displaying only one group at a time. Each View control manages one group of content and all the View controls are held together in a MultiView control.

The MultiView control is responsible for displaying one View control at a time. The View displayed is called the active view.

The syntax of MultiView control is:

```
<asp:MultView ID= "MultiView1" runat= "server">

</asp:MultiView>
```

The syntax of View control is:

```
<asp:View ID= "View1" runat= "server">

</asp:View>
```

However, the View control cannot exist on its own. It would render error if you try to use it stand-alone. It is always used with a Multiview control as:

```
<asp:MultView ID= "MultiView1" runat= "server">

   <asp:View ID= "View1" runat= "server"> </asp:View>

</asp:MultiView>
```

Properties of View and MultiView Controls

Both View and MultiView controls are derived from Control class and inherit all its properties, methods, and events. The most important property of the View control is Visible property of type Boolean, which sets the visibility of a view.

The MultiView control has the following important properties:

Properties	Description
Views	Collection of View controls within the MultiView.
ActiveViewIndex	A zero based index that denotes the active view. If no view is active, then the index is -1.

The CommandName attribute of the button control associated with the navigation of the MultiView control are associated with some related field of the MultiView control.

For example, if a button control with CommandName value as NextView is associated with the navigation of the multiview, it automatically navigates to the next view when the button is clicked.

The following table shows the default command names of the above properties:

Properties	Description
NextViewCommandName	NextView
PreviousViewCommandName	PrevView
SwitchViewByIDCommandName	SwitchViewByID
SwitchViewByIndexCommandName	SwitchViewByIndex

The important methods of the multiview control are:

Methods	Description
SetActiveview	Sets the active view
GetActiveview	Retrieves the active view

Every time a view is changed, the page is posted back to the server and a number of events are raised. Some important events are:

Events	Description
ActiveViewChanged	Raised when a view is changed
Activate	Raised by the active view
Deactivate	Raised by the inactive view

Apart from the above mentioned properties, methods and events, multiview control inherits the members of the control and object class.

Example

This example illustrates how multiview controls are implemented. The web page has three views. Each view has two button for navigating through the views.

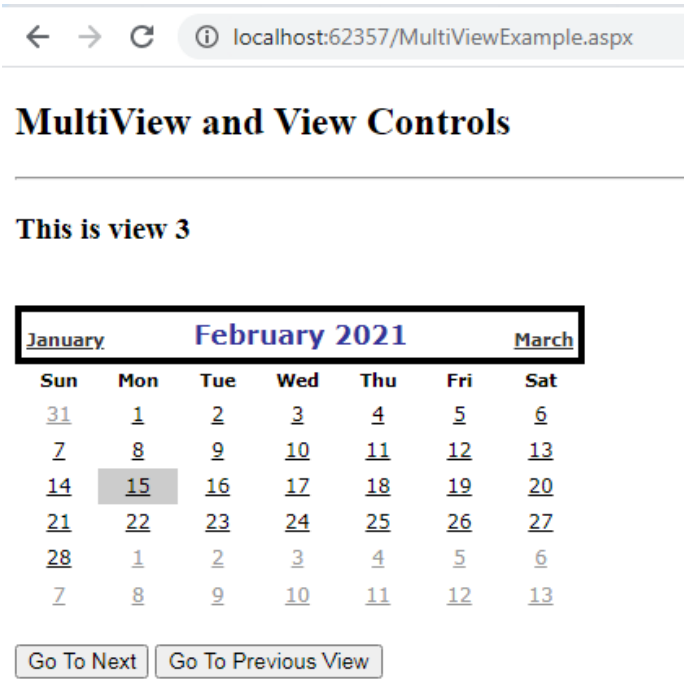
The content file code is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="MultiViewExample.aspx.cs"
Inherits="LUC_Web_Sample2.MultiViewExample" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <h2>MultiView and View Controls</h2>

            <asp:DropDownList ID="DropDownList1" runat="server"
onselectedindexchanged="DropDownList1_SelectedIndexChanged" Visible="False">
            </asp:DropDownList>

            <hr />

            <asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="2"
onactiveviewchanged="MultiView1_ActiveViewChanged" >
               <asp:View ID="View1" runat="server">
                  <h3>This is view 1</h3>
                  <br />
                  <asp:Button CommandName="NextView" ID="btnnext1"
runat="server" Text = "Go To Next" />
                  <asp:Button CommandArgument="View3"
CommandName="SwitchViewByID" ID="btnlast" runat="server" Text  ="Go To Last" />
               </asp:View>

               <asp:View ID="View2" runat="server">
                  <h3>This is view 2</h3>
                  <asp:Button CommandName="NextView" ID="btnnext2"
runat="server" Text = "Go To Next" />
                  <asp:Button CommandName="PrevView" ID="btnprevious2"
runat="server" Text = "Go To Previous View" />
               </asp:View>

               <asp:View ID="View3" runat="server">
                  <h3> This is view 3</h3>
                  <br />
                  <asp:Calendar ID="Calender1" runat="server"
BackColor="White" BorderColor="White" BorderWidth="1px" Font-Names="Verdana"
Font-Size="9pt" ForeColor="Black" Height="190px" NextPrevFormat="FullMonth"
Width="350px">
                     <DayHeaderStyle Font-Bold="True" Font-Size="8pt" />
```

```
                    <NextPrevStyle      Font-Bold="True"      Font-Size="8pt"
ForeColor="#333333" VerticalAlign="Bottom" />
                    <OtherMonthDayStyle ForeColor="#999999" />
                    <SelectedDayStyle  BackColor="#333399"  ForeColor="White"
/>
                    <TitleStyle     BackColor="White"     BorderColor="Black"
BorderWidth="4px" Font-Bold="True" Font-Size="12pt" ForeColor="#333399" />
                    <TodayDayStyle BackColor="#CCCCCC" />
                 </asp:Calendar>
                <br />
                <asp:Button                                  CommandArgument="0"
CommandName="SwitchViewByIndex" ID="btnfirst"    runat="server" Text = "Go To
Next" />
                <asp:Button      CommandName="PrevView"      ID="btnprevious"
runat="server" Text = "Go To Previous View" />
              </asp:View>

          </asp:MultiView>

    </div>
    </form>
</body>
</html>
```

Observe the following:

The MultiView.ActiveViewIndex determines which view will be shown. This is the only view rendered on the page. The default value for the ActiveViewIndex is -1, when no view is shown. Since the ActiveViewIndex is defined as 2 in the example, it shows the third view, when executed.



Panel Controls

The Panel control works as a container for other controls on the page. It controls the appearance and visibility of the controls it contains. It also allows generating controls programmatically.

The basic syntax of panel control is as follows:

```
<asp:Panel ID= "Panel1"  runat = "server">

</asp:Panel>
```

The Panel control is derived from the WebControl class. Hence it inherits all the properties, methods and events of the same. It does not have any method or event of its own. However it has the following properties of its own:

Properties	Description
BackImageUrl	URL of the background image of the panel.
DefaultButton	Gets or sets the identifier for the default button that is contained in the Panel control.
Direction	Text direction in the panel.
GroupingText	Allows grouping of text as a field.
HorizontalAlign	Horizontal alignment of the content in the panel.
ScrollBars	Specifies visibility and location of scrollbars within the panel.
Wrap	Allows text wrapping.

Working with the Panel Control

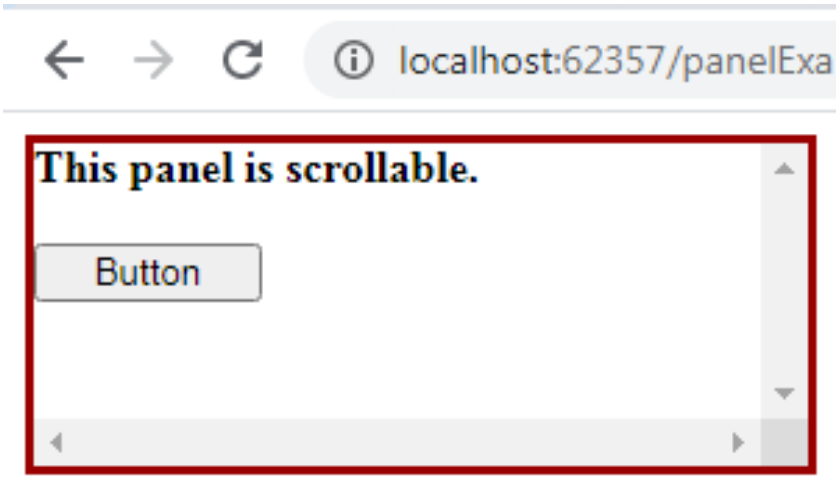
Let us start with a simple scrollable panel of specific height and width and a border style. The ScrollBars property is set to both the scrollbars, hence both the scrollbars are rendered.

The source file has the following code for the panel tag:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="panelExample1.aspx.cs" Inherits="LUC_Web_Sample2.panelExample1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
      <asp:Panel ID="Panel1" runat="server" BorderColor="#990000"
BorderStyle="Solid" width="1px" Height="116px" ScrollBars="Both"
style="width:278px">
          <b>This panel is scrollable.</b>

          <br />
   <br />
   <asp:Button ID="btnpanel" runat="server" Text="Button" style="width:82px"
/>
</asp:Panel>

    </form>
</body>
```

```
</html>
```

The panel is rendered as follows:



Example

The following example demonstrates dynamic content generation. The user provides the number of label controls and textboxes to be generated on the panel. The controls are generated programmatically.

Change the properties of the panel using the properties window. When you select a control on the design view, the properties window displays the properties of that particular control and allows you to make changes without typing.

The source file for the example is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="panelExample2.aspx.cs" Inherits="LUC_Web_Sample2.panelExample2" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Panel ID="pnldynamic" runat="server" BorderColor="#990000"
         BorderStyle="Solid"  width="1px"  Height="180px"    ScrollBars="Auto"
style="width:60%"         BackColor="White"                 Font-Names="Arial"
HorizontalAlign="Center">

         This panel shows dynamic control generation:
         <br />
         <br />
      </asp:Panel>
    </div>
          <table style="width: 51%;">
      <tr>
         <td class="style2">No of Labels:</td>
         <td class="style1">
            <asp:DropDownList ID="ddllabels" runat="server">
               <asp:ListItem>0</asp:ListItem>
```

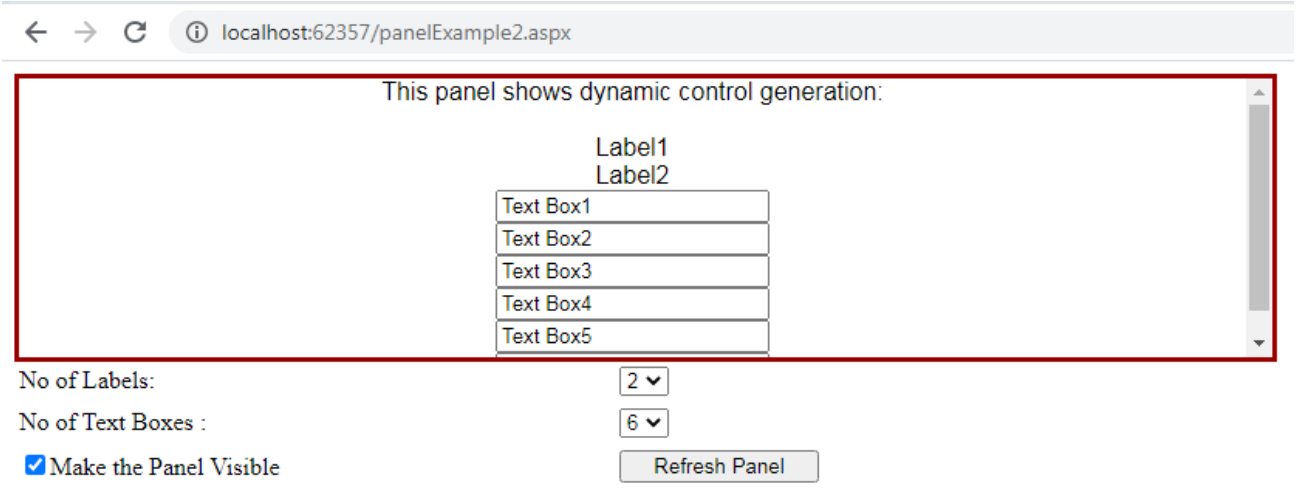
```
                <asp:ListItem>1</asp:ListItem>
                <asp:ListItem>2</asp:ListItem>
                <asp:ListItem>3</asp:ListItem>
                <asp:ListItem>4</asp:ListItem>
            </asp:DropDownList>
        </td>
    </tr>

    <tr>
        <td class="style2"> </td>
        <td class="style1"> </td>
    </tr>

    <tr>
        <td class="style2">No of Text Boxes :</td>
        <td class="style1">
            <asp:DropDownList ID="ddltextbox" runat="server">
                <asp:ListItem>0</asp:ListItem>
                <asp:ListItem Value="1"></asp:ListItem>
                <asp:ListItem>2</asp:ListItem>
                <asp:ListItem>3</asp:ListItem>
                <asp:ListItem Value="4"></asp:ListItem>
                <asp:ListItem>5</asp:ListItem>
                <asp:ListItem>6</asp:ListItem>
                <asp:ListItem Value="7"></asp:ListItem>
            </asp:DropDownList>
        </td>
    </tr>

    <tr>
        <td class="style2"> </td>
        <td class="style1"> </td>
    </tr>

    <tr>
        <td class="style2">
            <asp:CheckBox ID="chkvisible" runat="server"
                Text="Make the Panel Visible" />
        </td>

        <td class="style1">
            <asp:Button ID="btnrefresh" runat="server" Text="Refresh Panel"
                style="width:129px" />
        </td>
    </tr>
</table>

</form>
</body>
</html>
```

The code behind the Page_Load event is responsible for generating the controls dynamically:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace LUC_Web_Sample2
{
    public partial class panelExample2 : System.Web.UI.Page
```

```
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            //make the panel visible
            pnldynamic.Visible = chkvisible.Checked;

            //generating the lable controls:
            int n = Int32.Parse(ddllabels.SelectedItem.Value);
            for (int i = 1; i <= n; i++)
            {
                Label lbl = new Label();
                lbl.Text = "Label" + (i).ToString();
                pnldynamic.Controls.Add(lbl);
                pnldynamic.Controls.Add(new LiteralControl("<br />"));
            }

            //generating the text box controls:

            int m = Int32.Parse(ddltextbox.SelectedItem.Value);
            for (int i = 1; i <= m; i++)
            {
                TextBox txt = new TextBox();
                txt.Text = "Text Box" + (i).ToString();
                pnldynamic.Controls.Add(txt);
                pnldynamic.Controls.Add(new LiteralControl("<br />"));
            }

        }
    }
}
```

When executed, the panel is rendered as:



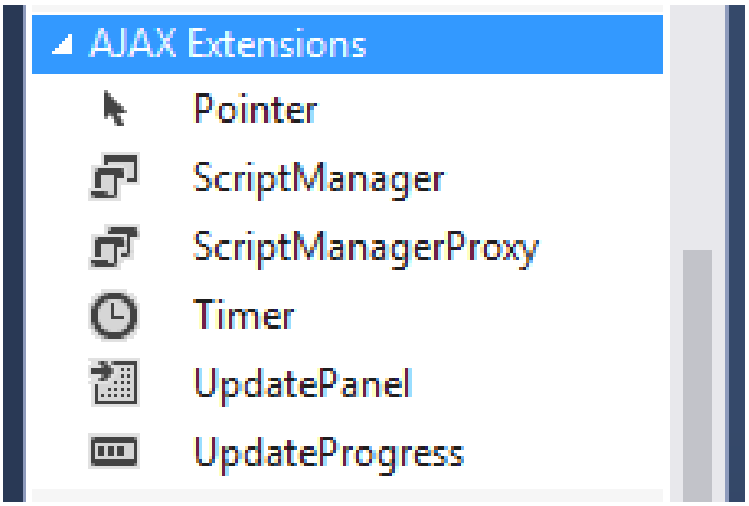
Ajax Control

AJAX stands for Asynchronous JavaScript and XML. This is a cross platform technology which speeds up response time. The AJAX server controls add script to the page which is executed and processed by the browser.

However like other ASP.NET server controls, these AJAX server controls also can have methods and event handlers associated with them, which are processed on the server side.

The control toolbox in the Visual Studio IDE contains a group of controls called the 'AJAX

Extensions'



The ScriptManager Control

The ScriptManager control is the most important control and must be present on the page for other controls to work.

It has the basic syntax:

<asp:ScriptManager ID="ScriptManager1" runat="server">

</asp:ScriptManager>

If you create an 'Ajax Enabled site' or add an 'AJAX Web Form' from the 'Add Item' dialog box, the web form automatically contains the script manager control. The ScriptManager control takes care of the client-side script for all the server side controls.

The UpdatePanel Control

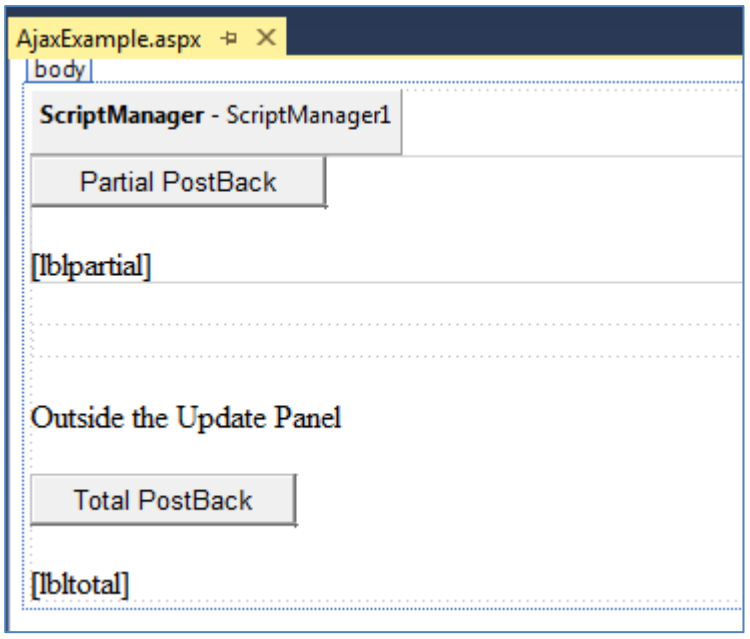
The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just that portion of the page.

For example, if a button control is inside the update panel and it is clicked, only the controls within the update panel will be affected, the controls on the other parts of the page will not be affected. This is called the partial post back or the asynchronous post back.

Example

This example illustrates how to work with AJAX control. Add an AJAX web form in your application. It contains the script manager control by default. Insert an update panel. Place a button control along with a label control within the update panel control. Place another set of button and label outside the panel.

The design view looks as follows:



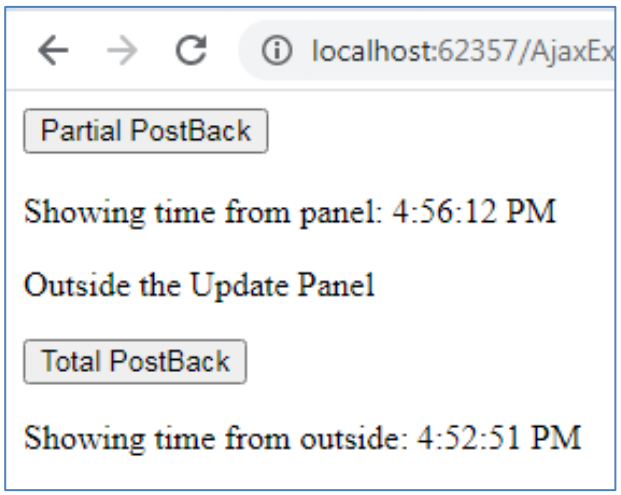
The source file is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="AjaxExample.aspx.cs"
Inherits="LUC_Web_Sample2.AjaxExample" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1"
runat="server"></asp:ScriptManager>
    </div>

            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
      <ContentTemplate>
         <asp:Button ID="btnpartial" runat="server" onclick="btnpartial_Click"
Text="Partial PostBack"/>
         <br />
         <br />
         <asp:Label ID="lblpartial" runat="server"></asp:Label>
      </ContentTemplate>
   </asp:UpdatePanel>

   <p> </p>
   <p>Outside the Update Panel</p>
   <p>
      <asp:Button ID="btntotal" runat="server" onclick="btntotal_Click"
Text="Total PostBack" />
   </p>

   <asp:Label ID="lbltotal" runat="server"></asp:Label>

    </form>
</body>
</html>
```

Both the button controls have same code for the event handler:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace LUC_Web_Sample2
{
    public partial class AjaxExample : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void btnpartial_Click(object sender, EventArgs e)
        {
            string time = DateTime.Now.ToLongTimeString();
            lblpartial.Text = "Showing time from panel: " + time;
            lbltotal.Text = "Showing time from outside: " + time;
        }

        protected void btntotal_Click(object sender, EventArgs e)
        {
            string time = DateTime.Now.ToLongTimeString();
            lblpartial.Text = "Showing time from panel: " + time;
            lbltotal.Text = "Showing time from outside: " + time;
        }
    }
}
```

Observe that when the page is executed, if the total post back button is clicked, it updates time in both the labels but if the partial post back button is clicked, it only updates the label within the update panel.



A page can contain multiple update panels with each panel containing other controls like a grid and displaying different part of data.

When a total post back occurs, the update panel content is updated by default. This default mode could be changed by changing the UpdateMode property of the control. Let us look at other properties of the update panel.

Properties of the UpdatePanel Control

The following table shows the properties of the update panel control:

Properties	Description
ChildrenAsTriggers	This property indicates whether the post backs are coming from the child controls, which cause the update panel to refresh.
ContentTemplate	It is the content template and defines what appears in the update panel when it is rendered.
ContentTemplateContainer	Retrieves the dynamically created template container object and used for adding child controls programmatically.
IsInPartialRendering	Indicates whether the panel is being updated as part of the partial post back.
RenderMode	Shows the render modes. The available modes are Block and Inline.
UpdateMode	Gets or sets the rendering mode by determining some conditions.
Triggers	Defines the collection trigger objects each corresponding to an event causing the panel to refresh automatically.

Methods of the UpdatePanel Control

The following table shows the methods of the update panel control:

Methods	Description
CreateContentTemplateContainer	Creates a Control object that acts as a container for child controls that define the UpdatePanel control's content.
CreateControlCollection	Returns the collection of all controls that are contained in the UpdatePanel control.
Initialize	Initializes the UpdatePanel control trigger collection if partial-page rendering is enabled.
Update	Causes an update of the content of an UpdatePanel control.

The behavior of the update panel depends upon the values of the UpdateMode property and ChildrenAsTriggers property.

UpdateMode	ChildrenAsTriggers	Effect

Always	False	Illegal parameters.
Always	True	UpdatePanel refreshes if whole page refreshes or a child control on it posts back.
Conditional	False	UpdatePanel refreshes if whole page refreshes or a triggering control outside it initiates a refresh.
Conditional	True	UpdatePanel refreshes if whole page refreshes or a child control on it posts back or a triggering control outside it initiates a refresh.

The UpdateProgress Control

The UpdateProgress control provides a sort of feedback on the browser while one or more update panel controls are being updated. For example, while a user logs in or waits for server response while performing some database oriented job.

It provides a visual acknowledgement like "Loading page...", indicating the work is in progress.

The syntax for the UpdateProgress control is:

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server" DynamicLayout="true"
AssociatedUpdatePanelID="UpdatePanel1" >

   <ProgressTemplate>

      Loading...

   </ProgressTemplate>

</asp:UpdateProgress>
```

The above snippet shows a simple message within the ProgressTemplate tag. However, it could be an image or other relevant controls. The UpdateProgress control displays for every asynchronous postback unless it is assigned to a single update panel using the AssociatedUpdatePanelID property.

Properties of the UpdateProgress Control

The following table shows the properties of the update progress control:

Properties	**Description**
AssociatedUpdatePanelID	Gets and sets the ID of the update panel with which this control is associated.
Attributes	Gets or sets the cascading style sheet (CSS) attributes of the UpdateProgress control.
DisplayAfter	Gets and sets the time in milliseconds after which the progress template is displayed. The default is 500.

DynamicLayout	Indicates whether the progress template is dynamically rendered.
ProgressTemplate	Indicates the template displayed during an asynchronous post back which takes more time than the DisplayAfter time.

Methods of the UpdateProgress Control

The following table shows the methods of the update progress control:

Methods	**Description**
GetScriptDescriptors	Returns a list of components, behaviors, and client controls that are required for the UpdateProgress control's client functionality.
GetScriptReferences	Returns a list of client script library dependencies for the UpdateProgress control.

The Timer Control

The timer control is used to initiate the post back automatically. This could be done in two ways:

1. Setting the Triggers property of the UpdatePanel control:

```
<Triggers>

   <asp:AsyncPostBackTrigger ControlID="btnpanel2" EventName="Click" />

</Triggers>
```

2. Placing a timer control directly inside the UpdatePanel to act as a child control trigger. A single timer can be the trigger for multiple UpdatePanels.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Always">

   <ContentTemplate>

      <asp:Timer ID="Timer1" runat="server" Interval="1000">

         </asp:Timer>

      <asp:Label ID="Label1" runat="server" Height="101px" style="width:304px"
>

         </asp:Label>

   </ContentTemplate>

</asp:UpdatePanel>
```

JavaScript

JavaScript is a lightweight, interpreted programming language with object-oriented capabilities

that allows you to build interactivity into otherwise static HTML pages.

JavaScript code is not compiled but translated by the translator. This translator is embedded into the browser and is responsible for translating JavaScript code.

Key Points

- It is Lightweight, interpreted programming language.
- It is designed for creating network-centric applications.
- It is complementary to and integrated with Java.
- It is complementary to and integrated with HTML
- It is an open and cross-platform

JavaScript Statements
JavaScript statements are the commands that tell the browser what action to perform. Statements are separated by semicolon (;).

JavaScript statement constitutes the JavaScript code which is translated by the browser line by line.
Example of JavaScript statement:

```
document.getElementById("demo").innerHTML = "Welcome";
```

Following table shows the various JavaScript Statements:

Sr.No.	Statement	Description
1.	switch case	A block of statements in which execution of code depends upon different cases. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.
2.	If else	The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.
3.	While	The purpose of a while loop is to execute a statement or code block repeatedly as long as expression is true. Once expression becomes false, the loop will be exited.
4.	do while	Block of statements that are executed at least once and continues to be executed while condition is true.
5.	for	Same as while but initialization, condition and increment/decrement is done in the same line.
6.	for in	This loop is used to loop through an object's properties.
7.	continue	The continue statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block.
8.	break	The break statement is used to exit a loop early, breaking out of the enclosing curly braces.
9.	function	A function is a group of reusable code which can be called anywhere in your programme. The keyword function is used to declare a function.
10.	return	Return statement is used to return a value from a function.
11.	var	Used to declare a variable.
12.	try	A block of statements on which error handling is implemented.

13.	catch	A block of statements that are executed when an error occur.
14.	throw	Used to throw an error.

JavaScript Comments
JavaScript supports both C-style and C++-style comments, thus:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.-->
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Example

```
<script language="javascript" type="text/javascript">
  <!--

    // this is a comment. It is similar to comments in C++

    /*
      * This is a multiline comment in JavaScript
      * It is very similar to comments in C Programming
    */
  //-->
<script>
```

JavaScript variable
Variables are referred as named containers for storing information. We can place data into these containers and then refer to the data simply by naming the container.

Rules to declare variable in JavaScript
Here are the important rules that must be followed while declaring a variable in JavaScript.

- In JavaScript variable names are case sensitive i.e. a is different from A.
- Variable name can only be started with a underscore (_) or a letter (from a to z or A to Z), or dollar ($) sign.
- Numbers (0 to 9) can only be used after a letter.
- No other special character is allowed in variable name.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword as follows:

```
<script type="text/javascript">
  <!--
    var money;
    var name, age;
  //-->
```

```
</script>
```
Variables can be initialized at time of declaration or after declaration as follows −

```
<script type="text/javascript">
   <!--
      var name = "Munir";
      var money;
      money = 2000.50;
   //-->
</script>
```

Javascript Data Type
There are two kinds of data types as mentioned below −

- Primitive Data Type
- Non Primitive Data Type

The following table describes Primitive Data Types available in javaScript

Data type	Description
String	Can contain groups of character as single value. It is represented in double quotes. E.g. var x= “tutorial”.
Numbers	Contains the numbers with or without decimal. E.g. var x=44, y=44.56;
Booleans	Contain only two values either true or false. E.g. var x=true, y= false.
Undefined	Variable with no value is called Undefined. E.g. var x;
Null	If we assign null to a variable, it becomes empty. E.g. var x=null;

The following table describes Non-Primitive Data Types in javaScript

Data type	Description
Array	Can contain groups of values of same type. E.g. var x={1,2,3,55};
Objects	Objects are stored in property and value pair. E.g. var rectangle = { length: 5, breadth: 3};

JavaScript Functions
Function is a group of reusable statements (Code) that can be called any where in a program. In javascript function keyword is used to declare or define a function.

Key Points

- To define a function use function keyword followed by functionname, followed by parentheses ().
- In parenthesis, we define parameters or attributes.
- The group of reusabe statements (code) is enclosed in curly braces {}. This code is executed whenever function is called.

Syntax

```
function functionname (p1, p2) {
   function coding…
}
```

JavaScript Operators
Operators are used to perform operation on one, two or more operands. Operator is represented by a symbol such as +, =, *, % etc. Following are the operators supported by javascript −

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Arithmetic Operators
Following table shows all the arithmetic operators supported by javascript −

Operator	Description	Example
+	Add two operands.	10 + 10 will give 20
-	Subtract second operand from the first.	10 – 10 will give 0
*	Multiply two operands.	10 * 30 will give 300
/	Divide numerator by denominator	10/10 will give 1
%	It is called modulus operator and gives remainder of the division.	10 % 10 will give 0
++	Increment operator, increases integer value by one	10 ++ will give 11
--	Decrement operator, decreases integer value by one	10 – will give 9

Comparison Operators
Following table shows all the comparison operators supported by javascript:

Operator	Description	Example
==	Checks if values of two operands are equal or not, If yes then condition becomes true.	10 == 10 will give true
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	10 !=10 will give false
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	20 > 10 will give true
<	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	10 < 20 will give true
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	10 >=20 will give false
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	10 <=20 will give true.

Logical Operators
Following table shows all the logical operators supported by javascript:

Operator	Description	Example
&&	Logical AND operator returns true if both operands are non zero.	10 && 10 will give true.
||	Logical OR operator returns true If any of the operand is non zero	10 || 0 will give true.
!	Logical NOT operator complements the logical state of its operand.	! (10 && 10) will give false.

Assignment Operators
Following table shows all the assignment operators supported by javascript:

Operator	Description	Example
=	Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Conditional Operator
It is also called ternary operator, since it has three operands.

Operator	Description	Example
?:	Conditional Expression	If Condition is true? Then value X : Otherwise value Y

Control Structure
Control structure actually controls the flow of execution of a program. Following are the several control structure supported by javascript.

- if … else
- switch case
- do while loop
- while loop
- for loop

If … else
The if statement is the fundamental control statement that allows JavaScript to make decisions

and execute statements conditionally.

Syntax

```
if (expression){
  Statement(s) to be executed if expression is true
}
```

Example

```
<script type="text/javascript">
    var age = 20;
    if( age > 18 ){
      document.write("<b>Qualifies for driving</b>");
    }
</script>
```

Switch case

The basic syntax of the switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Syntax

```
switch (expression) {
  case condition 1: statement(s)
            break;
  case condition 2: statement(s)
            break;
  ...
  case condition n: statement(s)
            break;
  default: statement(s)
}
```

Example

```
<script type="text/javascript">
      var grade='A';
      document.write("Entering switch block<br/>");
      switch (grade) {
         case 'A': document.write("Good job<br/>");
            break;
         case 'B': document.write("Pretty good<br/>");
            break;
         case 'C': document.write("Passed<br/>");
            break;
         case 'D': document.write("Not so good<br/>");
            break;
         case 'F': document.write("Failed<br/>");
            break;
         default:  document.write("Unknown grade<br/>")
      }
      document.write("Exiting switch block");
</script>
```

Do while Loop

The do...while loop is similar to the while loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is false.

Syntax

```
do{
   Statement(s) to be executed;
} while (expression);
```

Example

```
<script type="text/javascript">
      var count = 10;
      document.write("Starting Loop" + "<br/>");
      do{
         document.write("Current Count : " + count + "<br/>");
         count--;
      }while (count > 0);
      document.write("Loop stopped!");
</script>
```

While Loop

The purpose of a while loop is to execute a statement or code block repeatedly as long as expression is true. Once expression becomes false, the loop will be exited.

Syntax

```
while (expression){
   Statement(s) to be executed if expression is true
}
```

Example

```
<script type="text/javascript">
   <!--
      var count = 0;
      document.write("Starting Loop" + "<br/>");
      while (count < 10){
         document.write("Current Count : " + count + "<br/>");
         count++;
      }
      document.write("Loop stopped!");
   //-->
</script>
```

This will produce following result:

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

For Loop

The for loop is the most compact form of looping and includes the following three important parts:

- The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.

- The test statement which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.

- The iteration statement where you can increase or decrease your counter.

Syntax

```
for (initialization; test condition; iteration statement){
   Statement(s) to be executed if test condition is true
}
```

Example

```
<script type="text/javascript">
      var count;
      document.write("Starting Loop" + "<br/>");
      for(count = 0; count < 10; count++){
         document.write("Current Count : " + count );
         document.write("<br/>");
      }
      document.write("Loop stopped!");
</script>
```

This will produce following result which is similar to while loop:

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

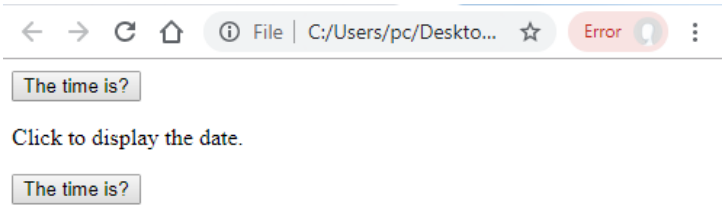
Creating Sample Program

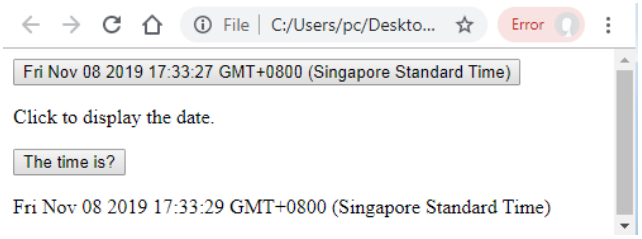
Following is the sample program that shows time, when we click in button.

```
<html>
   <body>
      <button onclick="this.innerHTML=Date()">The time is?</button>
      <p>Click to display the date.</p>
      <button onclick="displayDate()">The time is?</button>
      <script>
         function displayDate() {
            document.getElementById("demo").innerHTML = Date();
         }
       </script>
         <p id="demo"></p>
   </body>
```

</html>

Output





ACTIVITY

1. Describe UpdatePanel Control in ASP.NET
2. What is Multi View and how to implement it in an ASP.NET web project?
3. What is Panel Control and how to implement it in an ASP.NET web project?
4. Highlight any 5 properties of panel control in ASP.NET
5. What is Ajax Control and how to implement it in an ASP.NET web project?

KEYWORDS

- Panel
- Ajax
- Multi Views
- Timer
- UpdatePanel

SUMMARY

This chapter presents the following topics:

- Introduction to Multi Views
- Properties of View and MultiView Controls
- Panel Controls
- Working with the Panel Control
- Ajax Control
- The UpdatePanel Control
- Properties of the UpdatePanel Control
- Methods of the UpdatePanel Control
- The UpdateProgress Control
- Properties of the UpdateProgress Control
- Methods of the UpdateProgress Control
- The Timer Control