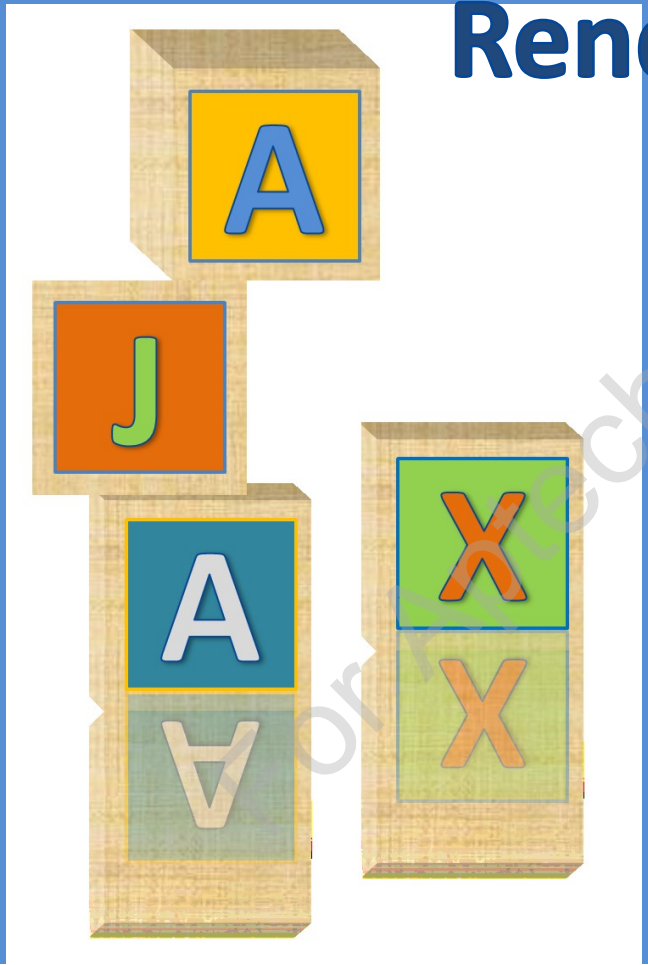
# Programming ASP.NET AJAX

## Session: 4

## Rendering Partial Pages



# Objectives



- Explain Partial Page Rendering
- Explain implementing Partial Page Updates
- Explain traditional Web Page Rendering
- Identify the need for Partial page Rendering
- Outline the features of Partial Page Rendering
- Explain the `Sys.WebForms.PageRequestManager` class
- Explain the `ScriptManager` control
- Explain the `UpdatePanel` control
- Explain the `Timer` control

# Partial Page Rendering

AJAX

- Partial-page rendering uses server controls found in ASP.NET AJAX and client functions available in Microsoft AJAX Library.
- The client library APIs can be utilized for further AJAX functionality.

Consider the scenario wherein, a retail company has outlets in many parts of the country and operates from a central location. The company wants to gather information, about people's shopping pattern, which will be used for market analysis and strategy formulation.

The company wants to develop a poll application that can run on the home page of their Web site without any difficulty, as the users are already familiar with the existing functionalities.

# Traditional Approach

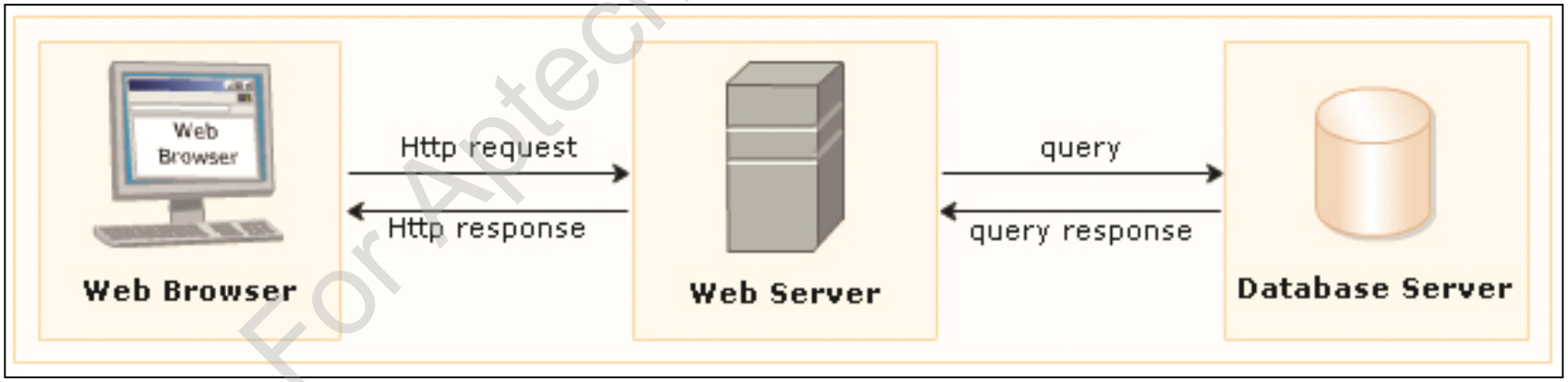
AJAX

- A developer might build the poll feature by using traditional HTML elements.

Develop a Web page that contains a **Submit** button and a series of check boxes with the values 10, 20, 30, and so on.

When a user selects a check box and clicks **Submit**, the user's poll response is registered in a database with the user's `UserID`.

The page refreshes because it completes a round trip to the server.

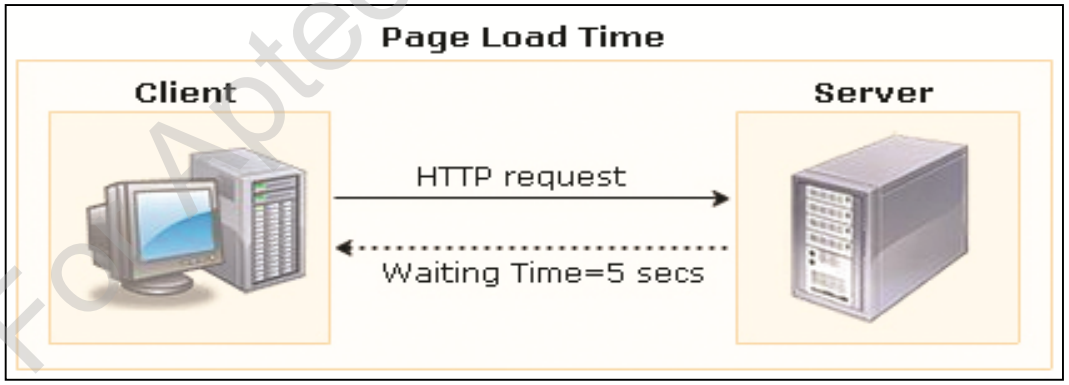



**Traditional Approach**

# Drawbacks of the Traditional Approach

AJAX

- Synchronous calls to the server
- Refreshing and reloading of the entire page
- Slow latency
- Long page load time
- Long page transfer time
- Flickering because of multiple reloads due to presence of large images

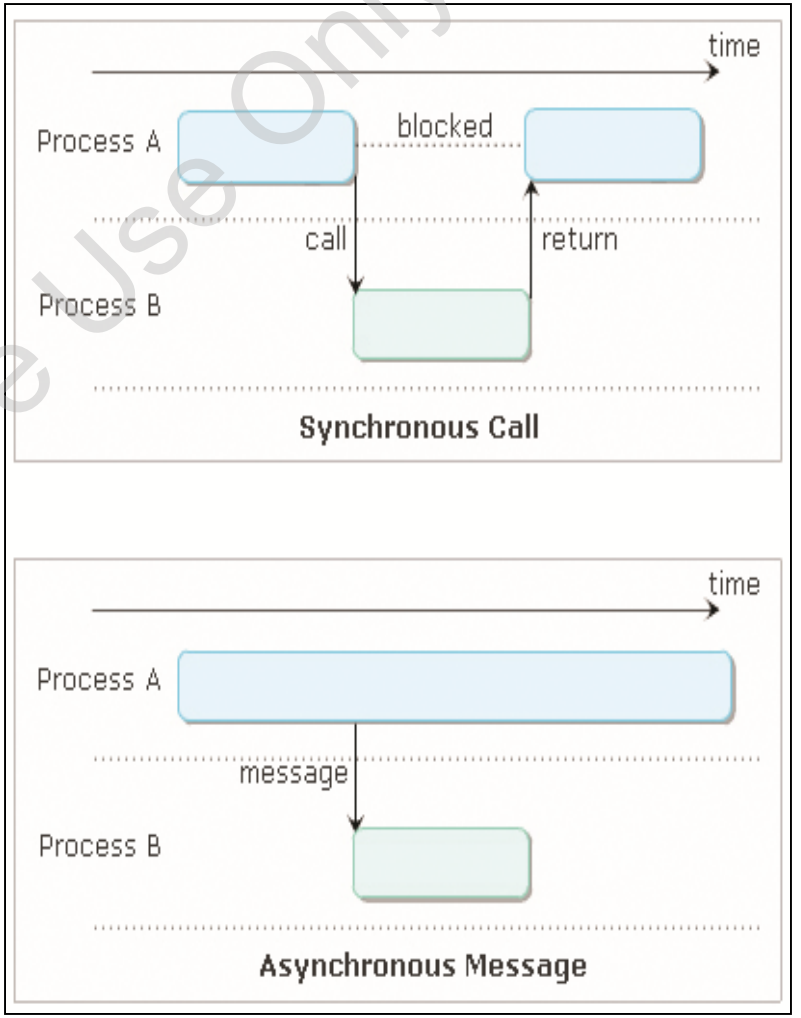



**Scenario Showing Waiting Time of Five Seconds**

# Solution – Partial Page Rendering with AJAX

AJAX

- For better server response time:
  - The application must send only the poll data
  - The application requests should be sent in an asynchronous manner
- With **partial page rendering** provided by AJAX, specific portions of a page can be sent asynchronously.
- This greatly nullifies the flickering effect and reduces the load on the server with better bandwidth utilization.
- In ASP.NET AJAX, partial page rendering can be achieved by using server extensions controls in the **Toolbox** of Visual Studio 2013 IDE.




**Asynchronous vs. Synchronous Call**

# Features of Partial Page Rendering

AJAX

## Declarative Coding

- Requires declaring markup tags.
- Enables partial page rendering in a Web site, using these markup tags with server extension controls.

## Integration

- Takes place between Microsoft server extension controls and Microsoft AJAX library to achieve common tasks.
- Users are allowed to cancel a postback.

## Flicker Reduction

- Is possible due to use of partial page updates, thus providing better user experience.

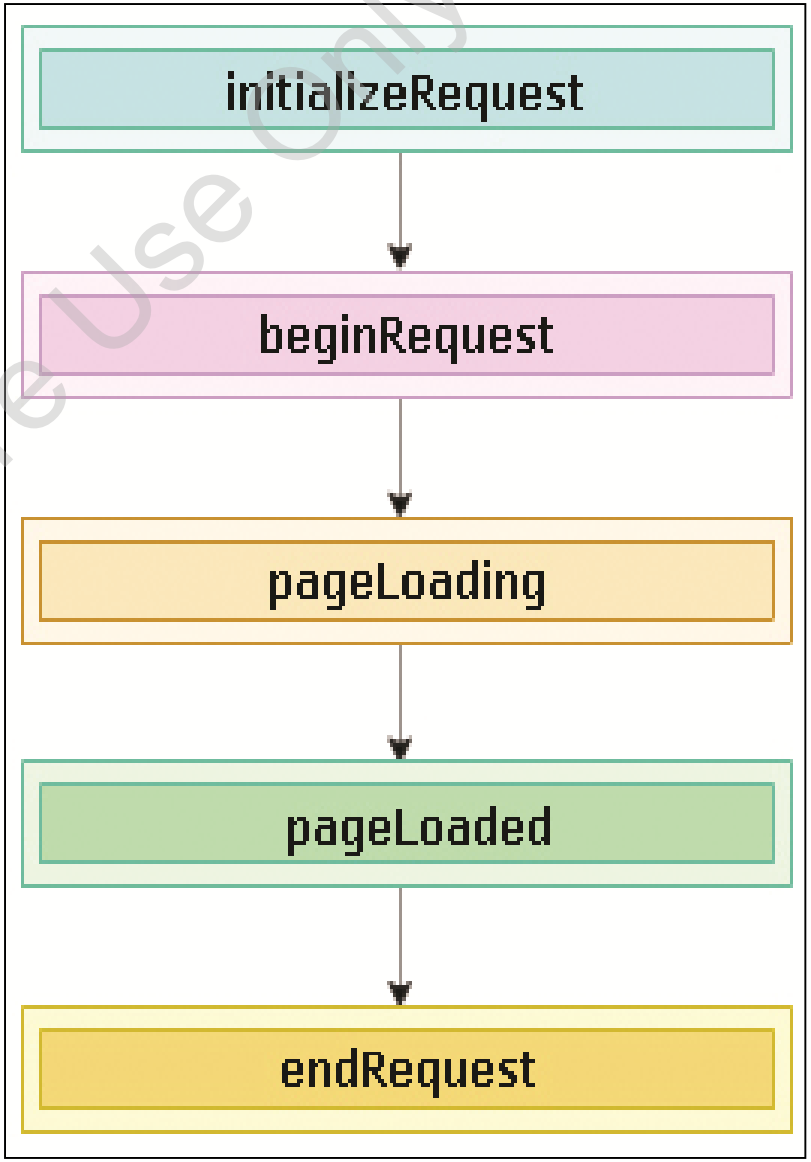
## Error Handling

- Enables customization of the way errors are displayed in the browser.

# Implementing Partial Page Updates 1-6

AJAX

- `Sys.WebForms.PageRequestManager` class is responsible for partial postback.
- An object of this class can be accessed in a client application by using the `ScriptManager` control and/or the `UpdatePanel` control.
- The `PageRequestManager` class provides properties, methods, and events to enable partial page rendering.
- Some of the events of `PageRequestManager` class are as follows:
  - `initializeRequest`
  - `beginRequest`
  - `pageLoading`
  - `pageLoaded`
  - `endRequest`




**PageRequestManager Class**

# Implementing Partial Page Updates 2-6

## initializeRequest

- This event is triggered during initialization of an asynchronous postback, but before the asynchronous request begins.
- Handlers which are called after the `initializeRequest` event is invoked, are added to this event similar to any other event. The following code snippet shows the `initializeRequest`.

**Code Snippet**

```
//This statement should be inside pageInit
// or pageLoad event.
//adds the handler Initialize to
// the initializeRequest event
Sys.WebForms.PageRequestManager.getInstance().add_initializeRequest(Initialize);

//if you want to remove the handler Initialize from
// the initializeRequest event, uncomment these lines
//Sys.WebForms.PageRequestManager.
// getInstance().remove_initializeRequest(Initialize);

function Initialize(sender,args)
{
  alert("The Request that was sent is getting initialized");
}
```

# Implementing Partial Page Updates 3-6

## beginRequest

- This event is triggered after the `initializeRequest` event.
- Handlers are added to this event similar to any normal event. The code snippet is as follows:

**Code Snippet**

```
//This statement should be inside pageInit
// or pageLoad event.
//adds the handler BeginRequest1 to
// the beginRequest event
Sys.WebForms.PageRequestManager.getInstance().add_beginRequest
                                    (BeginRequest1);

//To remove the handler BeginRequest1 from
// the beginRequest event, uncomment these lines
//Sys.WebForms.PageRequestManager.
// getInstance().remove_beginRequest(BeginRequest1);

function BeginRequest1(sender,args)
{
  alert("Request is ready to be sent to server");
}
```

# Implementing Partial Page Updates 4-6

## pageLoading

- This event is triggered after the server processes the request, but before loading the page.
- This event can be used if any animation or transition is required, before the updated data is placed in their respective positions. The code snippet for page load is as follows:

**Code Snippet**

```
//This statement should be inside pageInit
// or pageLoad event.
//adds the handler PageLoading1 to
// the pageLoading event
Sys.WebForms.PageRequestManager.getInstance().add_pageLoading(PageLoading1);

// To remove the handler PageLoading1 from
// the pageLoading event, uncomment these lines
//Sys.WebForms.PageRequestManager.
// getInstance().remove_pageLoading(PageLoading1);

function PageLoading1(sender, args)
{
  alert("My page is started loading");
}
```

## pageLoaded

- This event is triggered when all the contents of the page is updated.
- In this event, animation or effects initiated in the `beginRequest`, can be cleared. The following code snippet shows the `pageLoaded`.

**Code Snippet**

```
//This statement should be inside pageInit
// or pageLoad event.
//adds the handler PageLoaded1 to the
// pageLoaded event
Sys.WebForms.PageRequestManager.getInstance().add_pageLoaded
                                        (PageLoaded1);
// To remove the handler PageLoaded1 from the pageLoaded event, uncomment
// these lines
//Sys.WebForms.PageRequestManager.
// getInstance().remove_pageLoaded(PageLoaded1);

function PageLoaded1(sender, args)
{
  alert("My page is loaded");
}
```

# Implementing Partial Page Updates 6-6

## endRequest

- This event is the last event which is triggered in the client postback cycle.
- This event fires irrespective of the postback being a success or failure same as `finally` in a `try...catch...finally` block. The following code snippet shows the `endRequest`.

**Code Snippet**

```
//This statement should be inside pageInit
// or pageLoad event.
//adds the handler EndRequest to
// the endRequest event
Sys.WebForms.PageRequestManager.getInstance().add_endRequest(EndRequest1);

// To remove the handler EndRequest1
//from the endRequest event,
// uncomment these lines
// Sys.WebForms.PageRequestManager.
// getInstance().remove_endRequest(EndRequest1);

function EndRequest1(sender, args)
{
  alert("My Request has ended");
}
```

# ScriptManager

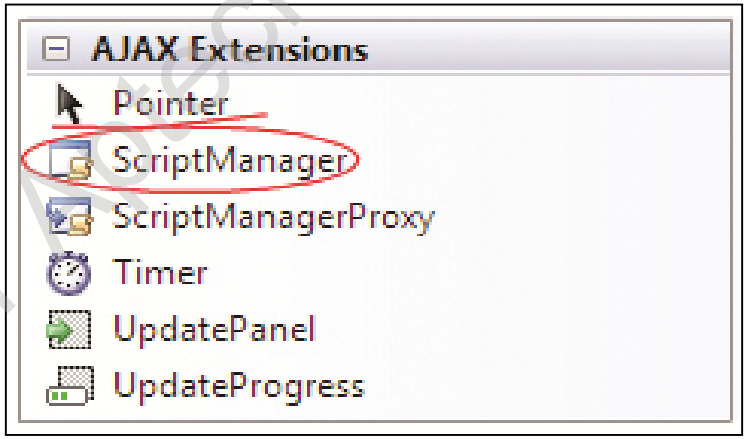
- Is responsible for delivering the client script to the browser asynchronously and also enables partial page update.
- Communicates with the events in the page life cycle.
- Control's `EnablePartialRendering` property's default value is set to true.
- The syntax and code snippet for `ScriptManager` control are as follows:

**Code Syntax**

```
<asp:ScriptManager ID=<id> runat="server"> </asp:ScriptManager>
```

**Code Snippet**

```
<asp:ScriptManager ID="scriptmgrEmp" runat="server"> </asp:ScriptManager>
```




**ScriptManager Control**

# Mark-up Enabled Properties of ScriptManager Control 1-2

AJAX

| Property Name | Type | Description |
|---|---|---|
| **AllowCustomErrorsRedirect** | bool | Enables you to handle the custom error section of the Web.config file. |
| **AsyncPostBackErrorMessage** | String | Enables you to retrieve or assign error messages that will be sent to the client if an error is raised. |
| **AsyncPostBackTimeout** | Int32 | Retrieves or sets the amount of time (in seconds), a client has to wait for an asynchronous request to complete. |
| **ScriptLoadTimeout** | Int32 | Determines the amount of time required for loading scripts into the client. |
| **ScriptMode** | Enum | Retrieves or sets the release version of scripts. |
| **ScriptPath** | String | Retrieves or sets the root path of script files to be sent to the client. |
| **EnableScriptGlobalization** | Bool | Gets or sets a value that displays if the ScriptManager control has rendered script in which parsing and formatting of culture-specific information is supported. |
| **EnableScriptLocalization** | Bool | Gets or sets a value that displays if the ScriptManager control has rendered the local versions of script. |

# Mark-up Enabled Properties of ScriptManager Control 2-2

- The following code snippet demonstrates some of the properties of a `ScriptManager` control.

**Code Snippet**

```
<asp:ScriptManager
ID="scriptmgrEmp" runat="server"
AsyncPostBackErrorMessage="There is an error"
AsyncPostBackTimeout="1000"
ScriptPath="validation.js">
</asp:ScriptManager>
```

# Code-only Properties of ScriptManager Control

AJAX

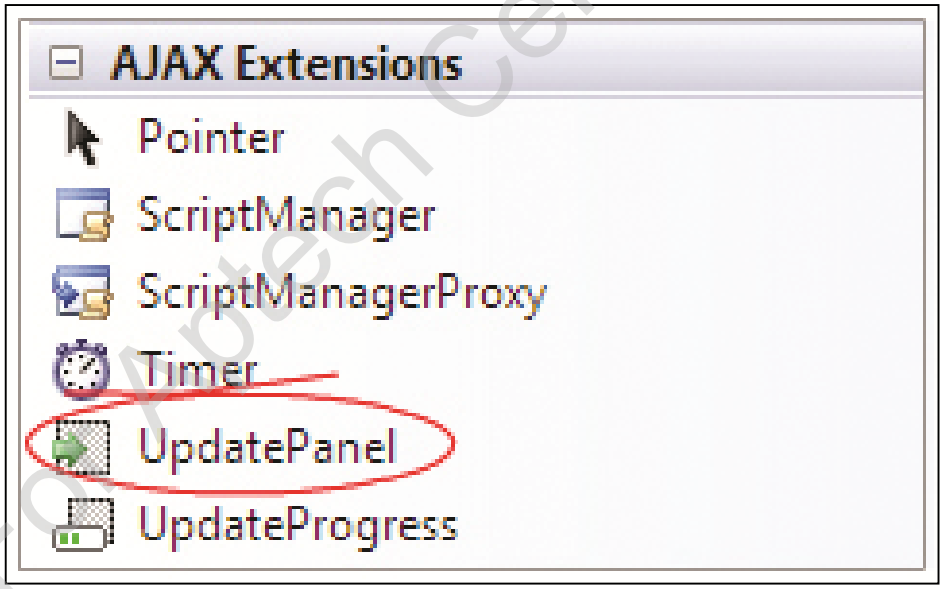
| Member Name | Type | Description |
|---|---|---|
| **IsDebuggingEnabled Property** | bool | Determines whether debugging is enabled. |
| **IsInAsyncPostback Property** | bool | Determines whether the page is requested using an asynchronous postback mode. |
| **Scripts Property** | Collection Base <Script Reference> | Retrieves a collection of script references which are sent to the client. |
| **SupportsPartialRendering Property** | bool | Retrieves or sets a value to indicate that all requests made by the client are partial updates (if set to true) or standard postbacks (if set to false). |
| **SetFocus(string) Method** | void | This method sets the focus to a particular control after a request has completed. |

**Code Snippet demonstrating code-only properties of a ScriptManager control**

```
protected void Page_Load(object sender, EventArgs e)
{
   scriptmgrEmp.IsInAsyncPostBack == true;
   scriptmgrEmp.SupportsPartialRendering == true;
}
```

# UpdatePanel 1-2

- Defines an area on the page in order to contain controls taking part in partial page rendering.
- Allows partial page rendering without writing any client script.
- Can be used when a `ScriptManager` control is included in the page.
- A page can have any number of `UpdatePanel` controls which can also be nested.
- The `UpdatePanel` control consists of control triggers.




**`UpdatePanel` Control**

# UpdatePanel 2-2

**Code Snippet illustrating the `UpdatePanel` control**

```
<asp:UpdatePanel ID="updpnlClock" runat="server">
<ContentTemplate>
<asp:Label ID="lblTime" runat="server" Text="">
</asp:Label>
<hr />
<asp:Button ID="btnSubmit" runat="server"
Text="Submit"Height="24px" Width="59px" />
</ContentTemplate>
</asp:UpdatePanel>
```

# Mark-up Enabled Properties of UpdatePanel Control



| Property Name | Type | Description |
|---|---|---|
| **ChildrenAsTriggers** | bool | This property, if set to true, enables the child controls to trigger a refresh on postback. |
| **RenderMode** | Enum (Block, Inline) | This property describes the way the content will be displayed. |
| **UpdateMode** | Enum (Always, conditional) | This property specifies whether the UpdatePanel refreshes during a partial page update or refreshes only when a specific trigger is hit. |

**Code Snippet illustrating the mark-up enabled properties of UpdatePanel control**

```
<asp:UpdatePanel ID="updpnlClock" runat="server"
RenderMode="Block"
ChildrenAsTriggers="true"
UpdateMode="Conditional">
```

# Code-only Properties of UpdatePanel Control

AJAX

| Property Name | Type | Description |
|---|---|---|
| **IsInPartialRendering** | bool | This property checks whether the UpdatePanel control supports partial rendering for the current request. |
| **ContentTemplate** | ITemplate | This property gets or sets the content inside the UpdatePanel control. |
| **ContentTemplate-Container** | Control | This property represents the template container for updating request. |
| **Triggers** | UpdatePanel – TriggerCollection | This property will get the list of triggers associated with the current UpdatePanel control. |

**Code Snippet illustrating use of the IsInPartialRendering property**

```
if (UpdatePanel.IsInPartialRendering)
{
    //do something
}
```

# Timer Control

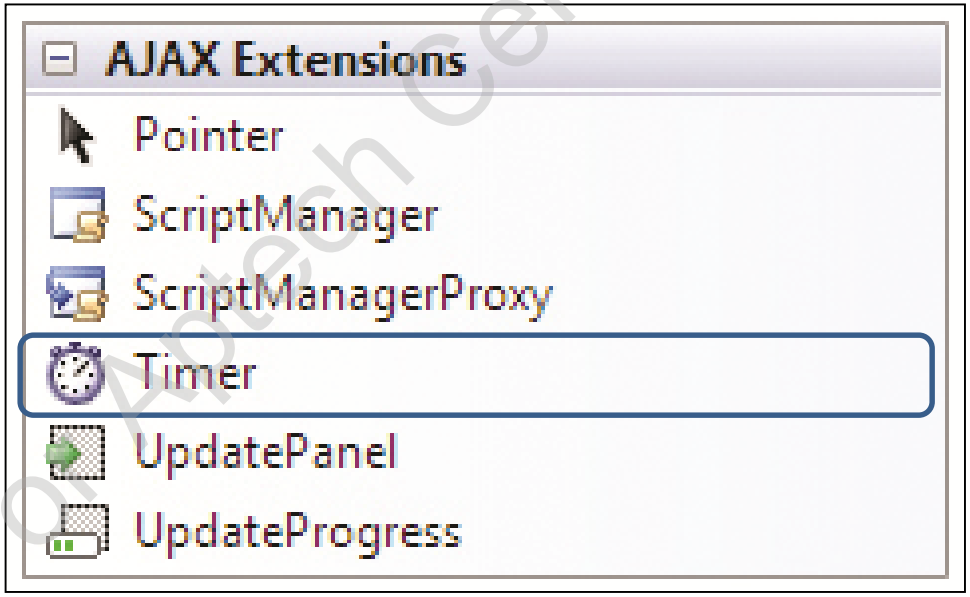
The ASP.NET AJAX `Timer` control performs a postback at specific intervals of time.

It is usually used with an `UpdatePanel` control in order to enable partial rendering of the contents in the panel at specific intervals of time.

The `Timer` control can also be used for synchronous postbacks of the complete page in definite intervals of time.

# Timer Control – Working

- The `Timer` control embeds a JavaScript component, which is responsible for initiating a postback, on the Web page.
- The `Timer` control requires an instance of `ScriptManager` control to exist in the page.
- The `Tick` event is triggered when the `Timer` control initiates a postback on the server.
- A page can contain multiple `Timer` controls.



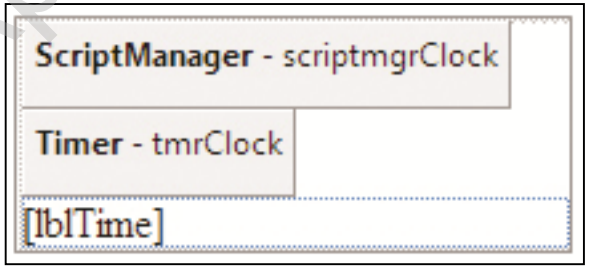

**`Timer` in the Toolbox**

# Timer Control – Example Code and Output

AJAX

**Code Snippet demonstrating a Timer control**

```
<asp:ScriptManager runat="server" id="scriptmgrClock" />
<asp:Timer ID="tmrClock" runat="server" Interval="120000"
  OnTick="tmrClock_Tick">
</asp:Timer>
<asp:UpdatePanel ID="updpnlClock" runat="server">
  <Triggers>
   <asp:AsyncPostBackTrigger ControlID="tmrClock"
    EventName="Tick" />
  </Triggers>
  <ContentTemplate>
   <asp:Label ID="lblTime" runat="server" ></asp:Label>
 </ContentTemplate>
</asp:UpdatePanel>
```




**Output - Timer in the Toolbox**

# Summary



- Partial-page rendering enables the user to refresh portion of a Web page, thus reducing the response time taken by conventional Web applications.
- Server extension controls in the Toolbox of Visual Studio enable partial page rendering.
- Asynchronous delivery of the client script to the browser and partial page updates are performed by the ScriptManager control.
- Defining the page area, which contains the controls that carry out partial page rendering, is performed by the UpdatePanel control.
- Enabling partial page rendering at specific time intervals is performed by the Timer control, used in addition to the UpdatePanel control.
- Partial page rendering or updates greatly nullifies the flickering effect and enables you to reduce the load on the server with better bandwidth utilization.
- Display of errors on the browser, during partial page rendering, can be customized using the error handling options.