General Page Life-cycle Stages

Stage	Description
`Page request`	The page request occurs before the page life cycle begins. When the page is requested by a user, ASP.NET determines whether the page needs to be parsed and compiled or whether a cached version of the page can be sent in response without running the page.
`Start`	In the start step, page properties such as Request and Response are set. At this stage, the page also determines whether the request is a postback or a new request and sets the IsPostBack property. Additionally, during the start step, the page's UICulture property is set.
`Page initialization`	During page initialization, controls on the page are available and each control's UniqueID property is set. Any themes are also applied to the page. If the current request is a postback, the postback data has not yet been loaded and control property values have not been restored to the values from view state.
`Load`	During load, if the current request is a postback, control properties are loaded with information recovered from view state and control state.
`Validation`	During validation, the Validate method of all validator controls is called, which sets the IsValid property of individual validator controls and of the page.
`Postback event handling`	If the request is a postback, any event handlers are called.
`Rendering`	Before rendering, view state is saved for the page and all controls. During the rendering phase, the page calls the Render method for each control, providing a text writer that writes its output to the OutputStream of the page's Response property.
`Unload`	Unload is called after the page has been fully rendered, sent to the client, and is ready to be discarded. At this point, page properties such as Response and Request are unloaded and any cleanup is performed.

Data Binding Events for Data-Bound Controls

Control Event	Typical Use
`DataBinding`	This event is raised by data-bound controls before the PreRender event of the containing control (or of the Page object) and marks the beginning of binding the control to the data.

`RowCreated (GridView)`
`ItemCreated (DataList, DetailsView, SiteMapPath, DataGrid, FormView, Repeater)`
	Use this event to manipulate content that is not dependent on data binding. For example, at run time, you might programmatically add formatting to a header or footer row in a GridView control.

`RowDataBound (GridView)`
`ItemDataBound (DataList, SiteMapPath, DataGrid, Repeater)`
	When this event occurs, data is available in the row or item, so you can format data or set the FilterExpression property on child data source controls for displaying related data within the row or item.

`DataBound`	This event marks the end of data-binding operations in a data-bound control. In a GridView control, data binding is complete for all rows and any child controls. Use this event to format data bound content or to initiate data binding in other controls that depend on values from the current control's content.

More .NET Cheat Sheats available at http://john-sheehan.com/blog/
More info at http://msdn2.microsoft.com/en-us/library/7949d756-1a79-464e-891f-904b1cfc7991.aspx

Common Life-cycle Events

Page Event	Typical Use
`PreInit`	Use this event for the following: • Check the IsPostBack property to determine whether this is the first time the page is being processed. • Create or re-create dynamic controls. • Set a master page dynamically. • Set the Theme property dynamically. • Read or set profile property values. Note: If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at this stage, its value might be overwritten in the next event.
`Init`	Raised after all controls have been initialized and any skin settings have been applied. Use this event to read or initialize control properties.
`InitComplete`	Raised by the Page object. Use this event for processing tasks that require all initialization be complete.
`PreLoad`	Use this event if you need to perform processing on your page or control before the Load event. After the Page raises this event, it loads view state for itself and all controls, and then processes any postback data included with the Request instance.
`Load`	The Page calls the OnLoad event method on the Page, then recursively does the same for each child control, which does the same for each of its child controls until the page and all controls are loaded.
`Control events`	Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event. In a postback request, if the page contains validator controls, check the IsValid property of the Page and of individual validation controls before performing any processing.
`LoadComplete`	Use this event for tasks that require that all other controls on the page be loaded.
`PreRender`	Before this event occurs: • The Page object calls EnsureChildControls for each control and for the page. • Each data bound control whose DataSourceID property is set calls its DataBind method. • The PreRender event occurs for each control on the page. Use the event to make final changes to the contents of the page or its controls.
`SaveStateComplete`	Before this event occurs, ViewState has been saved for the page and for all controls. Any changes to the page or controls at this point will be ignored. Use this event perform tasks that require view state to be saved, but that do not make any changes to controls.
`Render`	This is not an event; instead, at this stage of processing, the Page object calls this method on each control. All ASP.NET Web server controls have a Render method that writes out the control's markup that is sent to the browser. If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the Render method. A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code.
`Unload`	This event occurs for each control and then for the page. In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections. For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks. Note: During the unload stage, the page and its controls have been rendered, so you cannot make further changes to the response stream. If you attempt to call a method such as the Response.Write method, the page will throw an exception.