Questions:

1. What are ASP.NET Custom controls?
2. What are the three types of custom controls in ASP.NET?
3. What are the limitations of user controls in ASP.NET?
4. What are the steps to follow for creating and using a user control in a Web application?
5. How do you identify user controls?
6. What is the base class from which user controls derive?
7. What are the steps to follow to create properties and methods for the user control that you can use from a Web form?
8. What happens when you drag a user control from solution explorer and drop it on a web form?
9. What is Page.ClientScript.RegisterClientScriptBlock?

Questions and Answers:

1. What are ASP.NET Custom controls?

Answer: Custom controls extend the tools available to Web developers. Using custom controls, you can encapsulate key aspects of the visual interface and program logic that you want to reuse throughout your application, or throughout your organization.

2. What are the three types of custom controls in ASP.NET?

Answer: Microsoft Visual Studio .NET provides three types of custom control for use on Web forms.

a) Web user controls

   These combine existing server and HTML controls by using the Visual Studio .NET Designer to create functional units that encapsulate some aspect of the user interface. User controls reside in content files, which must be included in the project in which the controls are used.

b) Composite custom controls

   These create new controls from existing server and HTML controls. Although similar to user controls, composite controls are created in code rather than visually, and therefore they can be compiled into an assembly (.dll), which can be shared between multiple applications and used from the Toolbox in Visual Studio .NET.

c) Rendered custom controls

   These create entirely new controls by rendering HTML directly rather than using composition. These controls are compiled and can be used from the Toolbox, just like composite controls, but you must write extra code to handle tasks that are performed automatically in composite controls.

3. What are the limitations of user controls in ASP.NET?

Answer: As the user controls are not compiled into assemblies, they have the following limitations:

a) A copy of the control must exist in each Web application project in which the control is used.
b) User controls can't be loaded in the Visual Studio .NET Toolbox; instead, you must create them by dragging the control from Solution Explorer to the Web form.
c) User control code is initialized after the Web form loads, which means that user control property values are not updated until after the Web form's Load event.

4. What are the steps to follow for creating and using a user control in a Web application?

Answer:

a) Add a Web user control page (.ascx) to your project.
b) Draw the visual interface of the control in the designer.
c) Write code to create the control's properties, methods, and events.
d) Use the control on a Web form by dragging it from Solution Explorer to the Web form on which you want to include it.
e) Use the control from a Web form's code by declaring the control at the module level and then using the control's methods, properties, and events as needed within the Web form.

5. How do you identify user controls?

Answer: User controls are identified by their .ascx file extensions.

6. What is the base class from which user controls derive?

Answer: User controls derive from System.Web.UI.UserControl base class. This base class provides the base set of properties and methods you use to create the control.

7. What are the steps to follow to create properties and methods for the user control that you can use from a Web form?

Answer: To create properties and methods for the user control that you can use from a Web form, follow these steps:

a) Create the public property or method that you want to make available on the containing Web form.

b) Write code to respond to events that occur for the controls contained within the user control. These event procedures do the bulk of the work for the user control.

c) If the property or method needs to retain a setting between page displays, write code to save and restore settings from the control's ViewState.

8. What happens when you drag a user control from solution explorer and drop it on a web form?

Answer: When you drag a user control from solution explorer and drop it on a web form, Visual Studio .NET generates a @Register directive and HTML tags to create the control on the Web form.

9. What is Page.ClientScript.RegisterClientScriptBlock?

Answer: Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "k1", "alert('demo')", true);

a) The above function renders the Javascript code (including the script tag) only once irrespective of number of times the above line is executed.

b) This should be written in Page_Load event of Usercontrol in situations where multiple instances of Usercontrol will be included in the WebForm.