Layouts are used to maintain a consistent look and feel across multiple views within ASP.NET MVC application. As compared to Web Forms, layouts serve the same purpose as master pages, but offer a simple syntax and greater flexibility.

**Basic structure of Layout Page**

1. **<!DOCTYPE html>**
2. **<html>**
3. **<head>**
4. **<meta charset="utf-8" />**
5. **<meta name="viewport" content="width=device-width" />**
6. **<title>@ViewBag.Title</title>**
7. **@Styles.Render("~/Content/css")**
8. **@Scripts.Render("~/bundles/modernizr")**
9. **</head>**
10. **<body>**
11. **@RenderBody()**
13. **@Scripts.Render("~/bundles/jquery")**
14. **@RenderSection("scripts", required: false)**
15. **</body>**
16. **</html>**

In Asp.Net MVC, at application level we have \_ViewStart file with in Views folder for defining the default Layout page for your ASP.NET MVC application. For rendering layout page refer this article [Different ways of rendering layouts in Asp.Net MVC](http://www.dotnet-tricks.com/Tutorial/mvc/3WDE140613-Different-ways-of-rendering-layouts-in-Asp.Net-MVC.html).

**Styles.Render and Scripts.Render**

Style.Render is used to render a bundle of CSS files defined within BundleConfig.cs files. Styles.Render create style tag(s) for the CSS bundle. Like Style.Render, Scripts.Render is also used to render a bundle of Script files by rendering script tag(s) for the Script bundle.

1. **public class BundleConfig**
2. **{**
3. **public static void RegisterBundles(BundleCollection bundles)**
4. **{**
5. **bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(**
6. **"~/Scripts/jquery.unobtrusive\*",**
7. **"~/Scripts/jquery.validate\*"));**
9. **bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(**
10. **"~/Content/themes/base/jquery.ui.core.css",**
11. **"~/Content/themes/base/jquery.ui.resizable.css",**
12. **"~/Content/themes/base/jquery.ui.selectable.css",**
13. **"~/Content/themes/base/jquery.ui.button.css",**
14. **"~/Content/themes/base/jquery.ui.dialog.css",**
15. **"~/Content/themes/base/jquery.ui.theme.css"));**
16. **}**
17. **}**

Note

1. Styles.Render and Scripts.Render generate multiple style and script tags for each item in the CSS bundle and Script bundle when optimizations are disabled.
2. When optimizations are enabled, Styles.Render and Scripts.Render generate a single style and script tag to a version-stamped URL which represents the entire bundle for CSS and Scripts.

You can enable and disable optimizations by setting EnableOptimizations property of BundleTable class to true or false with in Global.asax.cs file as shown below.

1. **protected void Application\_Start()**
2. **{**
3. ***//Other code has been removed for clarity***
4. **System.Web.Optimization.BundleTable.EnableOptimizations = false;**
5. **}**

**Sections**

A section allow you to specify a region of content within a layout. It expects one parameter which is the name of the section. If you don’t provide that, an exception will be thrown. A section in a layout page can be defined by using the following code.

1. **@section header{**
2. **<h1>Header Content</h1>**
3. **}**

You can render above defined section header on the content page as given below:

1. **@RenderSection("header")**

By default, sections are mandatory. To make sections optional, just provides the second parameter value as false, which is a Boolean value.

1. **@RenderSection("header",false)**

Note

A view can define only those sections that are referred to in the layout page otherwise an exception will be thrown.

**RenderBody**

RenderBody method exists in the Layout page to render child page/view. It is just like the ContentPlaceHolder in master page. A layout page can have only one RenderBody method.

1. **@RenderBody()**

**RenderPage**

RenderPage method also exists in the Layout page to render other page exists in your application. A layout page can have multiple RenderPage method.

1. **@RenderPage("~/Views/Shared/\_Header.cshtml")**