Working with CSS

- Styles define how HTML tags must be formatted in addition to what they are meant for.
- There are three ways of inserting a style sheet:
    - Inline style
    - Internal style sheet
    - External style sheet
- Inline style inherits the formatting properties from Internal styles defined in <head> <style> section of the same page and can overwrite and extend properties as needed.
- Internal styles section inherits the properties from CSS file and can overwrite and extend properties as needed.
- If link to external style is placed after internal styles in HTML, then the external style sheet will override the internal style sheet.

CSS Syntax

Tag

{ ... }

Tag1, Tag2 -- Common Styles of Tag1 and Tag2

{ ... }

Tag1 Tag2 -- Apply styles for Tag2 nested in Tag1

{ ... }

Tag1.ClassName1 --Class specific for a tag

{ ... }

.ClassName2 -- Generic class

{ . . .}

#id1

{ . . . }

In Body of and HTML page we can link the styles as below

<Tag **Style**="..."></Tag> --Example of Inline style

<Tag1 **class**="ClassName1"></Tag1>

<Tag2 **class**="CalssName2"></Tag2>

<Tag3 **class**="CalssName2"></Tag3>

<Tag4 **id**="id1"></Tag4>

External Style Sheet: One common CSS file can be linked to all the Web Pages in the application:

<link href="DemoStyleSheet.css**" type="**text/css**" rel="**stylesheet**" />**

More about CSS: http://www.w3schools.com/css/default.asp

Working with Themes

Step 1: To an aspx page add all the required controls (Label) and set their style properties.

Step 2: Right click on Project → Add ASP.NET Folder → Theme

This adds to the project a folder by name **App_Themes** and a subfolder "Theme1"

Note: Any number of Themes can be added to "App_Themes".

Step 3: Add another Theme by name “Theme2” to the “App_Theme” folder

Under every Theme, **Skin** and **StyleSheet** files can be added. Skin files contain style properties based on server tag. StyleSheet file contains style properties based on client tag.

Step 4: To the Theme folders (Theme1 and Theme2) add “Label.skin” and copy the formatted control from aspx file into the skin file. Remove "Id" and other attributes which have nothing to do with formatting.

Step 5: To Apply theme to the controls of a given page

```
<%@ Page Theme="Themename"....%>
```

Step 6: Run the application and see output rendered for every control with style attribute

Note: In a Theme folder we can have multiple Skin files and their filenames can be anything but must have an extension as ".skin". Its not compulsory but it is recommended to name skin files based on the control names eg: Label.Skin, Calendar.Skin, Button.Skin etc..

Step 7: A skin file can have default skin and named skin for every control.

- Default Skin: <asp:Label ... />
- Named Skin: <asp:Label **SkinId**="lblDemoSkin" .../>

Step 8: To link named skin to control in aspx page:

```
<asp:Label SkinId="lblDemoSkin" .../>
```

Step 9: To disable theme for a control

```
<asp:Label EnableTheming="False" .../>
```

Notes:

1. If a theme is associated with the webform, when the webform is executing on the server…the skin files of the specified theme are all applied together to various server controls which are in webform.

2. Style Properties of Theme overwrites the Local Style Properties of the control.
3. Themes are not reflected in the design view of the webform in Studio.
4. If the same theme has to be used for all the webforms in the web application, the following line can be added to web.config.

   <pages theme="ThemeName"/>
5. The Theme can be assigned to the page also using <%@ Page **StylesheetTheme**="Theme1" ... but this takes lower precedence over local style properties of the control.
6. The Theme folder can also contain ".css" files and these files should contain style properties based on client tag name.
7. These css files are rendered as <link> tag in the head section of the webform and thus these style properties takes lower precedence than skin properties because skin properties are rendered as inline style properties.
8. If a class as initialized in css file has to be assigned to a server control, "**CssClass"** property of the server control must be initialized.

Setting Theme Programmatically:

The 'Theme' property of the Page can be set in or before the **'PreInit'** event of Page.

```
protected void btnTheme1_Click(object sender, EventArgs e)
{
   Session["Theme"] = "Theme1";
   Response.Redirect(Request.RawUrl);
}
protected void btnTheme2_Click(object sender, EventArgs e)
{
   Session["Theme"] = "Theme2";
   Response.Redirect(Request.RawUrl);
}
protected override void OnPreInit(EventArgs e)
{
   if (Session["Theme"] != null)
      Page.Theme = Session["Theme"].ToString();
}
```