COMMONWEALTH OF PENNSYLVANIA
DEPARTMENT'S OF Human Services, INSURANCE, AND AGING

INFORMATION TECHNOLOGY GUIDELINE

Name Of Guideline: **CSS Standards**	Number: **GDL-EASS017**
Domain: **Application**	Category: **Guidelines**
Date Issued: **03/18/2014**	Issued By: **DHS Bureau of Information Systems**
Date Revised: **02/17/2016**	

Table of Contents

Introduction

This document describes the World Wide Web Consortium (W3C) accessibility guidelines and the Internet accessibility design guidelines explained by the Office of Information Technology (OIT) in the Commonwealth of Pennsylvania. If all the standards and guidelines in this document are followed the web page/application should be ADA compliant. However, a large part of ADA compliance lies in user checks that have no strict guidelines to follow. The W3C handbook and a little legwork during the design phase is the best recipe to make pages visually appealing and equally accessible.

This standards document reflects our experience and lessons learned in developing web applications. Designing web applications has given us experience in document creation, graphic design, user interface design, information design, and the technical authoring skills required to optimize the CSS code, graphics, and text within web pages. This document shares our experiences and provides an instrument to embed this experience into your web project.

The web application will be constructed with Microsoft Active Server Pages (ASP.NET). This solution allows developers to construct business and presentation logic rapidly right along-side HTML for content and CSS for presentation. ASP.NET allows logic to be constructed with a variety of languages, the most common being C#. The presentation layer of the application can be enhanced by implementation of Cascading Style Sheets (CSS) and Javascript. In this document, we will introduce standards to leverage best practices consistently throughout the application.

The ultimate goal for the Web application is to facilitate the sharing, exchange and gathering information with a targeted audience. However, there is a possibility that many members of this audience may not use computers and software in the same manner. Following the standards and guidelines in this document is essential to achieve the goal of serving special needs of the disabled community and the principles of accessible Web design.

Purpose

The purpose of this document is to provide developers with DEA CSS development standards and guidelines that are compliant with the standards and guidelines set by WC3.

CSS Standards

Why CSS

Cascading Style Sheets (CSS) are now the standard for styling web page content. The usage of CSS allows content to be separated from styling, page layout, and design elements. Utilizing CSS not only aids in the creation of web applications by separating the code logic from the design elements but also allows for the web page to render faster since CSS parses faster than HTML in a web browser.

CSS also allows for quick styling adjustments that are uniform by allowing usually one change to be made in one code file and have the effect ripple across the entire site.

Important Rules for CSS

The three main rules regarding CSS are as follows:

1. Always use CSS for styling a page.
2. Utilize the cascade.
3. Never use inline styling.

Basic Webpage and Application Structure

Web pages and web applications need to utilize CSS. When creating a web page or web application you need to separate your project into at least two different components:

1. An HTML page.
2. A base CSS file.

The HTML page will contain all HTML required to display your content. The HTML should only be used to display basic things like headers (<h1>,<h2>,<h3> etc), tables (<td>, <tr>, etc), and any other tags necessary to put the content on the page **only**. No formatting, styling, background colors, or spacing should be done in the HTML file.

The base CSS file will need to contain at minimum styles for all of the tags used on your HTML page. For example if you have an HTML file that has the following HTML code to display your content:

```
<div>
        <h1>My Page</h1>
        <label>My Awesome Site</label>
```

You need to have a corresponding CSS page that looks something like this:

```
div{
        font-family:Helvetica, Arial;
}

h1{
        font-family:Helvetica, Arial;
        font-size:x-large;
}

label{
        font-family:Helvetica, Arial;
}
```

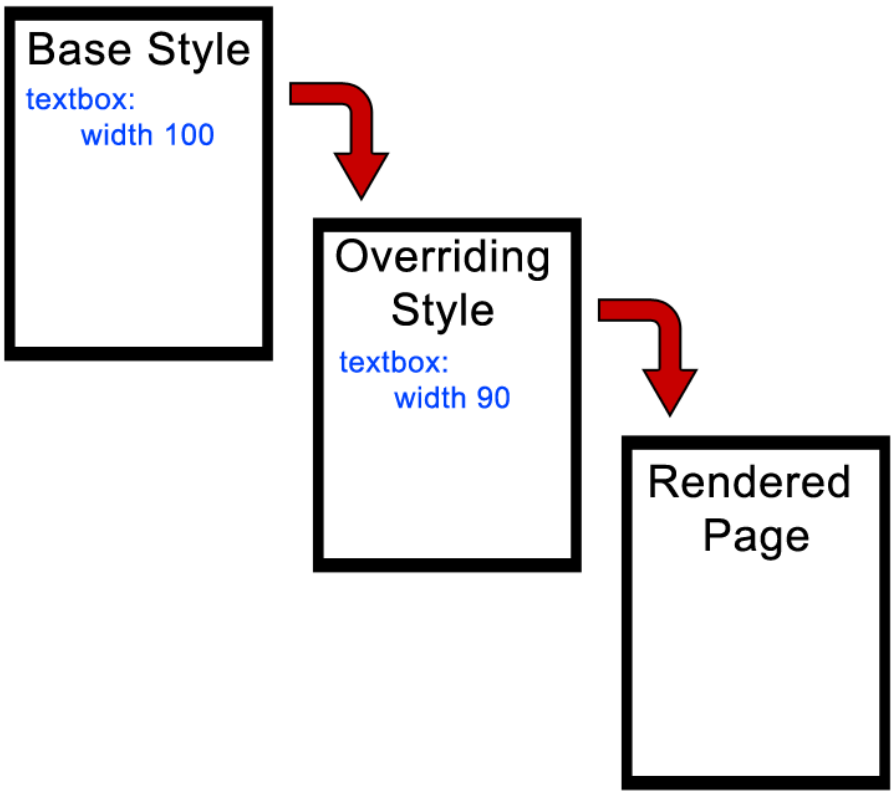
Having a default style sheet allows you to quickly generate a consistent look for all of the pages on your site or in your application. If you need to make a change it can be done much more quickly than with HTML styling.

Default style sheets, while fine for most cases will not cover all the necessary styles on larger applications where data may need to be presented in slightly different ways. Although it is possible to create custom classes for each element that needs to be styled separately, it is advised against since this creates giant CSS files that are hard to follow and maintain. A much better way is to utilize the “cascade”.

Utilizing the Cascade: Additional CSS Pages

The cascade (as it’s referred to) is CSS’s version of inheritance. Cascading is done by creating a separate CSS file for a page that will rely on special custom formatting. The page still derives its styles from the base CSS file but where the two pages have the same style for the same tag, the style with more specificity wins out.

For example, say I have in my CSS base file a style for an input.StandardTextBox that sets a textbox’s width at 100 pixels. The style works perfectly on all pages except for one in which it is just a bit too large (say it needs to be 90 pixels instead of 100). Instead of creating another style in the style sheet you would create a new style sheet with an overriding style:



Despite being a separate file, this actually keeps things cleaner and easier to maintain because now you do not have to prefix every HTML tag with the class="" parameter and can instead just code HTML as normal.

This may seem inconvenient at first when a simple class or an inline style would do but when dealing with more complex applications that display content such as different reports or tabular data utilizing the cascade is essential to creating easy to read and maintain style sheets.

The rest of the application will continue to use the base style sheet (unless similarly overridden).

NOTE: When using ASP MVC and CSS, you will need to create a content spot for your CSS in your Master page underneath your base style sheet, similar to this:

```
<link href="~/Content/MLSStyles.css" rel="stylesheet" />
<asp:ContentPlaceHolder ID="OverrideStyles" runat="server" />
```

A good rule to follow would be to try and design your application to utilize as much of the base style sheet as possible. Things like backgrounds, font, labels, textbox sizes, and headings

should all be a standard size and font and be able to be shared across all pages.

Content such as images, tabular data that changes in size from page to page, and non-standard form layouts are good candidates for requiring a page to have an overriding style sheet since that type of content often changes dynamically. It would still be a good idea to have all of these elements in your base style sheet since they may never need to be overridden, but in many cases you may need to override widths for objects like tables to get all of your data to fit.

CSS Resets

When working with CSS, it is important to realize that every browser renders it differently. This can cause issues in your page if you don't make custom CSS for every single default HTML tag. For example, if you are creating a page and you never overwrite a <label> tag even though you use it, it could show up differently in different browsers.

This is where a "Style Reset" comes into play. A style reset is a chunk of CSS that sets your default styles for you. It resides at the top of the base style sheet and should be used in every project. You can customize it to fit your needs.

Here is a CSS Style Reset that can be used:

```
/*--:::CSS RESET:::--*/

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
        margin: 0;
        padding: 0;
        border: 0;
        font-size: 100%;
        font: inherit;
        vertical-align: baseline;
}
/*HTML5 display-role reset for older browsers*/
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
        display: block;
}
body {
        line-height: 1;
}
ol, ul {
```

```
        list-style: none;
}
blockquote, q {
        quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
        content: '';
        content: none;
}
table {
        border-collapse:collapse;
        border-spacing: 0px;
}
/*--:::END CSS RESET:::--*/
```

CSS Formatting

Although CSS is case-insensitive, HTML selectors are not so CSS should be treated as case sensitive at all times.

CSS class names should be written started in lowercase letters. Using hyphens to separate words is acceptable.

```
label.small {
        width:80px;
}
```

Brackets should also be formatted with the left bracket next to the name of the CSS element. It doesn't look like much here but saving one line per element on a large style sheet can make the style sheet much easier to read and scroll through.

Although not necessary, it is suggested that CSS be blocked out with comments in order to make it easy to see which CSS sections belong to what element on the page.

For example, here is a blocked out section specifically for labels:

```
/--------*Labels*--------/
label.small{
}
label.medium{
}
label.large{
{
/------*End Labels*-----/
```

It is also a good idea to put the name of the author, the date, and a description of the style sheet at the top of the sheet, that way if a user needs to contact the author or needs to know what a particular style sheet does they are able to do it.

CSS Best Practices

- When setting up your CSS, you want to keep the “Box Model” in mind.

  This does not mean create divs for every piece but sections of the site such as the header, footer, and content would benefit from utilizing the Box Model since each piece could be moved around easily without ruining the rest of the formatting.

- When designing your interface keep your colors in mind. Make sure to use colors that aren’t overbearing but have enough contrast to make it easy to see for visually impaired users. Avoid using pure black and pure white colors as this can be tiresome on the eyes when users have to use the software for hours on end.

  Also be sure to use fonts that stick out and are clear and easy to read.

- Try to keep your style sheet generic. Have several sizes of labels, several sizes of text boxes, and whatever other generic elements you need for your page. Any element that will only be used on one or two pages should probably be created in an overriding style sheet.

- Utilize the cascade. If you create custom styles every time you need a new format adjustment your HTML will quickly become bloated and harder to maintain due to needing to specify a class for each tag.

CSS Browser Support

Ideally, the CSS you implement should display on the following browsers and versions listed below:

- Microsoft Internet Explorer (Version 9.0 or Higher)
- Mozilla Firefox (Version 30.0 or Higher)
- Google Chrome (Version 35.0 or Higher)

CSS Current Version

CSS3 is the latest standard for CSS. CSS3 is completely backwards-compatible with earlier versions of CSS.

Guideline Change Log

Change Date	Version	CR #	Change Description	Author and Organization
03/18/2014	1.0	N/A	Initial creation.	Scott Nelson, DEA
08/05/2014	1.1	N/A	Refinement after Deloitte review.	Scott Nelson, DEA
02/17/2016	1.2	NA	Updated to reflect browser support	Bradley Deetz, DEA