# Introduction

As I approached the task of customizing the SSA pages accessible to our web site’s members & guests, I had a number of assumptions in mind.

1. The SSA page code would be HTML standards-compliant.
2. All coding would adhere to best practices (ok, so at least the HTML, CSS and JS code would pass common validation services).
3. All layout and styling would be defined solely in CSS, and would generally be lightweight & non-intrusive, so that customizations would be easily applied with minimal concern for CSS Specificity rules.

*I was wrong on this — big time. The global CSS file used on SSA pages contains over 4,400 lines encompassing 610 CSS rules. And most, if not all, of these rules include an ID selector at the top level. Furthermore this ID selector (#aaSuperResponsiveContainer) is unnecessary as it is effectively specified at the top level of the HTML document, which by definition means that it does not need to be specified at all as a constraint on the other CSS rules.*

1. Fixed-width specifications would be minimal (if existing at all) as differing ACGI clients will have differing requirements vis-à-vis page size based on their individual web site’s design.
2. Common display *‘widgets*’, such as presentation of Customer ID & Name, would exist and could be styled across the set of SSA pages with a single CSS rule set.
3. Each page would contain a determinable (ideally through CSS) characteristic so that the Client’s custom CSS could adapt styling for specific SSA pages.
4. Font specifications would exist for a minimal number of elements and would not be repeated throughout the CSS. From your client’s perspective font usage, style guidelines and even colour usage is an important part of their branding, and the SSA pages need to easily adapt to a client’s requirements.
5. Coding would adhere to, and employ, HTML5 elements. With shim (polyfill) support where necessary for older browser support.

*Two related questions. 1.) Has ACGI published a roadmap/schedule for when their product offerings will be fully migrated to HTML5? 2.) Are analytics as to actual usage (for AA and SSA pages) at a client-specific or broader level available?*

1. And in my ideal world, a common public framework (think Bootstrap) for layout & design would be employed. Not only would this reduce ACGI development & maintenance time, but also the availability of resources knowledgeable and able to provide support in this situation can only benefit your clients.

*I am not suggesting the use of Bootstrap specifically as I think it or others, like Foundation, may be to heavy for usage in this context. That said there are a number of more lightweight frameworks that may be suitable here. As an FYI, for the customizations of SSA pages for BOMA/Chicago I’m evaluating Yahoo’s PureCSS framework.*

1. Finally – and this is more of an infrastructure concern rather than a Code practice, I would like to see the existing SSA Header document replaced.
   1. One approach might be 3 standard documents that would be included as separate references in each SSA page.
      1. SSA\_CLIENT\_CODE – for items other than CSS or Javascript. For example in our situation we include links to external resources (such as web fonts).
      2. SSA\_CLIENT\_CSS
      3. SSA\_CLIENT\_JS (or better this should be included at the bottom of the generated SSA document, as that is the recommended best-practice).
      4. To extend this concept even further, if the SSA\_CLIENT\_CSS and SSA\_CLIENT\_JS pointers could reference the FTP site then clients could load those files there. Which would benefit those clients with a development workflow, but also permit browser caching of these common files.

# Observations

This is only a representative list of situations where my going-in assumptions were invalidated.

## HTML standards-compliant

1. Invalid HTML element usage – specifically the same ID used multiple times on a page

This can be seen on any of the profile pages that include attributes that may be multi-valued such as a phone number or address. As an example, look at the censsaindprofile page and the phone number fields.

*Recommendation:* retain this selector (as it is needed for styling) but as a Class name — eliminating the multiple usage conflict, and simplifying CSS override concerns.

1. Misuse use of HTML element <label>

On the censsacustbio page the aaBioInfo ID is coded as:

<div id="aaBioInfo">

<ul>

<li><label>Name:</label> Mark Stephen Niemczyk</li>

<li><label>Bio Received Date:</label> 04/11/2014</li>

</ul>

</div>

This is not an appropriate use of the HTML <label> element, as it should *represent a caption for an item in a user interface*. Here it is solely used for styling via CSS; as such a better choice would be to use a <span> element instead.

1. Invalid construction of HTML element <ul>

On the customer profile page, there is a pattern that is used for each attribute (.censsaprofile class) coded as:

<div id="aaCenssaprofileDispProfDtlUsername" class="aaCenssaprofile">

<ul class="aaFormFields">

<form action="CENSSAINDPROFILE.section\_update" method="GET">

<input name="p\_cust\_id" value="1182972" type="hidden">

<input name="p\_profile\_ty" value="INDIVIDUAL\_PROFILE" type="hidden">

<input name="p\_section\_nm" value="Username" type="hidden">

<input name="p\_format" value="110" type="hidden">

<input name="p\_referrer" value="https://www.bomachicago.org/users/ssa/eNrLKCkpKLbS189NzU1KLdJLys9NTM7ITE5Mz9fLL0rXh\_KLixP1k1PzgFRmXkpBUX5aZk6qXkpmcUFOYmU8lG9fAGPFl1Taevq5eIZ5uoQ6-sQHBPm7efq4qhXEJ5cWl8RnptgaGloYWZobAQCJ3y1W" type="hidden">

<input name="p\_skip\_confirm\_fl" value="N" type="hidden">

<div class="aaHeaderwrapper">

<div id="aaCenssaprofileDispProfHdrUsername" class="aaHeading">

<h3>Username</h3>

<ul class="aaSubmission aaCenssaprofileSubmit">

<li>

<label></label>

<input value="Edit" class="aaSubmitButton" id="aaCenssaprofileDispProfHdrUsernameEdit" type="submit">

</li>

</ul>

</div> <!-- end aaCenssaprofileDispProfHdrUsername -->

</div> <!-- End Header wrapper -->

</form>

<li id="aaProfileLoginId">

<label>User Name::<font color="RED">\*</font></label>

<div class="aaProfileDataWrapper"><p>314159</p></div>

<div class="aaInlineValidationWrapper">

<div class="aaValidationWrapper-Inner">

<span class="aaInlineValidationIcon"></span>

<span class="aaValidationTxt"></span>

</div>

</div>

</li><!-- close li id="aaProfileLoginId" -->

</ul><!-- close ul class="aaFormFields" -->

</div>

The use of the<form> element as a direct descendent of the<ul> element is non-conforming. <ul> elements may only contain zero or more <li> elements.

## Layout and styling would be defined solely in CSS

## Minimal use of fixed-width specifications

## Common display *‘widgets*’

## Font specifications

2.

3. jQuery Performance

Within the shared.js.1.12 code base, the *checkWindowSize* function accounts for approximately 1/3rd of the code – just looking at lines of code.

This function consists of a series of if/else statements testing the browser window width, and then adding / removing CSS class the elements within the DOM.

The first such if/then block (lines 827 – 893) is:

if ( $(window).width() > 2500 ) {

$('body#wwwstdhdrbody').addClass('wwwstdhdrbody2500');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody2210');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1920');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1840');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1760');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1600');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1440');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1360');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1280');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1230');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1170');

$('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1120');

$('#innerbody').removeClass('innerbody1120');

$('#innerbody').removeClass('innerbody1170');

$('#innerbody').removeClass('innerbody1090');

$('#innerbody').removeClass('innerbody1050');

$('#innerbody').removeClass('innerbody1024');

$('#innerbody').removeClass('innerbody990');

$('.success-outer3').addClass('success-outer32500');

$('.success-outer3').removeClass('success-outer32210');

$('.success-outer3').removeClass('success-outer31920');

$('.success-outer3').removeClass('success-outer31840');

$('.success-outer3').removeClass('success-outer31760');

$('.success-outer3').removeClass('success-outer31600');

$('.success-outer3').removeClass('success-outer31440');

$('.success-outer3').removeClass('success-outer31360');

$('.success-outer3').removeClass('success-outer31280');

$('.success-outer3').removeClass('success-outer31230');

$('.success-outer3').removeClass('success-outer31170');

$('.success-outer3').removeClass('success-outer31120');

$('.success-outer3').removeClass('success-outer31090');

$('.success-outer3').removeClass('success-outer31050');

$('.success-outer3').removeClass('success-outer31024');

$('.success-outer3').removeClass('success-outer3990');

$('.warning-outer3').addClass('warning-outer32500');

$('.warning-outer3').removeClass('warning-outer32210');

$('.warning-outer3').removeClass('warning-outer31920');

$('.warning-outer3').removeClass('warning-outer31840');

$('.warning-outer3').removeClass('warning-outer31760');

$('.warning-outer3').removeClass('warning-outer31600');

$('.warning-outer3').removeClass('warning-outer31440');

$('.warning-outer3').removeClass('warning-outer31360');

$('.warning-outer3').removeClass('warning-outer31280');

$('.warning-outer3').removeClass('warning-outer31230');

$('.warning-outer3').removeClass('warning-outer31170');

$('.warning-outer3').removeClass('warning-outer31120');

$('.warning-outer3').removeClass('warning-outer31090');

$('.warning-outer3').removeClass('warning-outer31050');

$('.warning-outer3').removeClass('warning-outer31024');

$('.warning-outer3').removeClass('warning-outer3990');

$('.error-outer3').addClass('error-outer32500');

$('.error-outer3').removeClass('error-outer32210');

$('.error-outer3').removeClass('error-outer31920');

$('.error-outer3').removeClass('error-outer31840');

$('.error-outer3').removeClass('error-outer31760');

$('.error-outer3').removeClass('error-outer31600');

$('.error-outer3').removeClass('error-outer31440');

$('.error-outer3').removeClass('error-outer31360');

$('.error-outer3').removeClass('error-outer31280');

$('.error-outer3').removeClass('error-outer31230');

$('.error-outer3').removeClass('error-outer31170');

$('.error-outer3').removeClass('error-outer31120');

$('.error-outer3').removeClass('error-outer31090');

$('.error-outer3').removeClass('error-outer31050');

$('.error-outer3').removeClass('error-outer31024');

$('.error-outer3').removeClass('error-outer3990');

So looking at this coding style from the perspective of performance (although I suspect there is an over-arching CSS design issue to be considered), there are a number of less than preferred coding conventions.

1. The determination of $(window).width() need only be determined once, with the result retained for subsequent evaluation in later if/else blocks. A minor point, I concede.
2. Within this single if statement there is a sequence of jQuery class manipulations against a DOM element selectors, such as 'body#wwwstdhdrbody'. So the original code is:

828 $('body#wwwstdhdrbody').addClass('wwwstdhdrbody2500');

829 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody2210');

830 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1920');

831 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1840');

832 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1760');

833 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1600');

834 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1440');

835 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1360');

836 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1280');

837 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1230');

838 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1170');

839 $('body#wwwstdhdrbody').removeClass('wwwstdhdrbody1120');

1. The recommended convention, as DOM element selection is costly, is that if a selector is to be reused across multiple jQuery statements is that the result of the selector should be retained in a variable, and that variable reference subsequently. For this code block, that would eliminate 11 jQuery selections.

new var $sel = $('body#wwwstdhdrbody');

828 $sel.addClass('wwwstdhdrbody2500');

829 $sel.removeClass('wwwstdhdrbody2210');

830 $sel.removeClass('wwwstdhdrbody1920');

831 $sel.removeClass('wwwstdhdrbody1840');

832 $sel.removeClass('wwwstdhdrbody1760');

833 $sel.removeClass('wwwstdhdrbody1600');

834 $sel.removeClass('wwwstdhdrbody1440');

835 $sel.removeClass('wwwstdhdrbody1360');

836 $sel.removeClass('wwwstdhdrbody1280');

837 $sel.removeClass('wwwstdhdrbody1230');

838 $sel.removeClass('wwwstdhdrbody1170');

839 $sel.removeClass('wwwstdhdrbody1120');

1. However the best practice convention is to taking advantage of the ability to chain jQuery operations.

new var $sel = $('body#wwwstdhdrbody');

828 $sel.addClass('wwwstdhdrbody2500')

829 .removeClass('wwwstdhdrbody2210')

830 .removeClass('wwwstdhdrbody1920')

831 .removeClass('wwwstdhdrbody1840')

832 .removeClass('wwwstdhdrbody1760')

833 .removeClass('wwwstdhdrbody1600')

834 .removeClass('wwwstdhdrbody1440')

835 .removeClass('wwwstdhdrbody1360')

836 .removeClass('wwwstdhdrbody1280')

837 .removeClass('wwwstdhdrbody1230')

838 .removeClass('wwwstdhdrbody1170')

839 .removeClass('wwwstdhdrbody1120');

1. The code can be improved further as the removeClass operation does allow for multiple classes to be specified.

new var $sel = $('body#wwwstdhdrbody');

828 $sel.addClass('wwwstdhdrbody2500')

.removeClass('wwwstdhdrbody2210 wwwstdhdrbody1920 wwwstdhdrbody1840 wwwstdhdrbody1760 wwwstdhdrbody1600 wwwstdhdrbody1440 wwwstdhdrbody1360 wwwstdhdrbody1280 wwwstdhdrbody1230 wwwstdhdrbody1170 wwwstdhdrbody1120');

1. At this point (actually as a result of the use of chaining), the use of the $sel variable is no longer necessary.

828 $('body#wwwstdhdrbody').addClass('wwwstdhdrbody2500')

.removeClass('wwwstdhdrbody2210 wwwstdhdrbody1920 wwwstdhdrbody1840 wwwstdhdrbody1760 wwwstdhdrbody1600 wwwstdhdrbody1440 wwwstdhdrbody1360 wwwstdhdrbody1280 wwwstdhdrbody1230 wwwstdhdrbody1170 wwwstdhdrbody1120');

1. Finally, and this last change requires an understanding of all possible class names that could be placed on the <body> element (which may be used by 3rd parties and/orACGI clients) , the removeClass operation does provide for the removal of all classes attached to a selector when it is invoked without any class names provided.

828 $('body#wwwstdhdrbody').removeClass()

.addClass('wwwstdhdrbody2500');

And now we have a single jQuery statement that is easily understood, for which maintenance is much less problematic and performs better.

Look for custom attribute names