Blenderbox Guidelines for HTML Templates

last modified: 06.30.2011
version: 1.3.2
created by: Caleb Brown

contact: Caleb Brown

Technology Lead

cbrown@blenderbox.com / technology@blenderbox.com

tel 718 963 4594 x106
fax 718 218 7804

www.blenderbox.com

Blenderbox Inc.
26 Dobbin Street, 3rd Floor, Brooklyn NY 11222
tel 718 963 4594 | www.blenderbox.com

INTRODUCTION

This document is a list of guidelines, tips, and examples for creating HTML, JavaScript and CSS for Blenderbox projects. Where applicable, please follow these guidelines to create a clean, SEO friendly website that adheres to Blenderbox's best practices.

1.0 - HTML

- Use the default package for directory structure and the base styles.
- Use XHTML for both coding practices and Doctype. This means that all image and break tags must have a close tag. Follow the index.html file for an example.
- **Do not use in-line JavaScript** and **never use in-line CSS**.
- Keep scripts at the bottom of the page and CSS links in the head of the page to help with page load performance.
- Do not use the @import tag in your CSS but instead use the link attribute. For reasons against the import tag, please review http://www.stevesouders.com/blog/2009/04/09/dont-use-import/ (available at the time of this writing).
- Do not use tables when you can avoid it, and most likely, you can avoid it.
- Always use a list when listing something, i.e. a list of news items in a promo chip, a list of subsections in the left navigation, a list of search results, or a list of links to pages on the site in the footer. Following this guideline helps search engines understand the content better and focus on what content is important.
- Only use ids when there is only one element on the page that needs style — otherwise use a class.
- Do not put all images in the /images folder, but rather, logically separate them into sub-directories. Our most commonly used image folders are already a part of the default package.
- Make sure all images have an alt, height, and width attribute.

1.1 - CSS NAMING CONVENTIONS

- CSS files should always be named after the media type they are associated with. For example, a CSS file associated with the 'all' media type should be named 'all.css.'
  A list of all W3C types can be found here http://www.w3.org/TR/CSS2/media.html.
- We have a tendency to design for the 960 grid system. Please check to see if the designs fit into that system and use it accordingly. There are two example files which should be copied into the all.css file if we are using the grid system. We have the 12 and 16 column grids stylesheets in the '/stylesheets/' folder. Please add that style to the all.css file if you are using the grid system. For more on the grid system, please review http://www.960.gs
- Do not make the CSS overly complicated, but rather let the styles cascade where reasonable and applicable while using as little mark up as necessary.
- When starting a project, begin by styling the basic HTML elements. If no design is provided for that, please confirm that it is not needed.
- Do not edit the RESET block in the package's '/stylesheets/all.css' file. This style comes from Eric Meyer's reset project and helps to make pages cross-browser compliant. If you notice that there has been an update to the reset CSS version, please notify technology@blenderbox.com.
- Organize the CSS into logical sections such as HTML, classes (site wide only), layout, promos, etc. as seen in the '/stylesheets/all.css' file.
- Make sure **all** links and buttons have a rollover state. If you cannot find a rollover or on state in a PSD, please contact project management and let them know and they will provide you with one.
- Use lowercase and for ids and dashes for classes.

  Example: #layout-container { } and .search-results { }

- The syntax format we use is bracket, space, property, colon, value, space, bracket—all on one line.

  Example: p { color:red; text-align:center; }

2.0 - TIPS TO FOLLOW

Some of the common issues with browser compliance as well as the Blenderbox standards can be found below.

- Use the line-height attribute when vertically aligning a single line of text or input field. This is often helpful when styling footers etc.
- Add padding to input fields so the text does not run flush against the field on the top, bottom, or left side. When working with this, make sure the descenders of the text are not cut off by the bottom of the input.
- Write the CSS with a hierarchical approach. For instance, if the header element appears on the top of the page, put the #header element at the top of the CSS file.
- All label tags attributes should be set to the value of the input field id so a user can click the label to select the field. All fields should have a label if there is design for it.
- Do not add padding to block elements with set height or width without adjusting the height and width to account for the padding. Under most circumstances, please only add padding to elements with no set height and width.

  Example: If you have an input field with the following style

  ```
  input[type='text'] , input.input-field { height:30px; width:150px; }
  ```

  and you would like to add a padding of 2px to the field, the style must become

  ```
  input[type='text'], input.input-field { height:26px; width:146px; padding:2px; }
  ```

- **CLEARING**
  - We have two options for clearing, first the clearfix classes and second the clear classes. You can read more about the clearfix here http://www.webtoolkit.info/css-clearfix.html

    The basic clearing classes are .clearfix, .clearfix-left, and .clearfix-right. These classes use the ':after' pseudo-selector to add clearing after the tag you are floating. To clear after a floated element, you can use clearfix.

    Examples of clearing after:

    ```
    <div class="float clearfix">Content that is floating</div>

    <div class="float-left clearfix-left">Content that is floating left</div>

    <div class="float-right clearfix-right">Content that is floating right</div>
    ```

    If you need to use a block element to clear a float, please use a div with a non-breaking space.

    Examples:

    ```
    <div class="float-left"></div>
    <div class="clear"> </div>
    ```

```
<div class="float-left"></div>
<div class="clear-left"> </div>
<div class="float-right"></div>
<div class="clear-right"> </div>
```

- **BUTTONS / IMAGE ROLLOVERS / IMAGES WITH TEXT**
  - Do not use JavaScript to handle rollovers of buttons, navigation elements, etc. because it means browsers with JavaScript disabled will not see the correct styles for the site. Instead, use CSS sprites, the technique outlined here - http://blog.blenderbox.com/2008/03/11/css-and-rollovers/. The package already has an example of how to handle navigation buttons that use images. Two things to note that are not included in the post are one, you should always put the text from the image in the link, and two, use 'text-indent:-2000px;' to make the text is invisible to the user but visible to search engines and screen readers. This is a best for SEO and 508 compliance and is included in the default package.

    For images with text like logos, follow the same principle to make the page more SEO Friendly.
- **IMAGES**
  - Cut out as much white-space from an image as you can.
  - Save .png files instead of .gif files when you can. If you do use .png files, make sure to account for the fact that IE 6 cannot natively display .png files. Look into pngfix.js and make sure to apply it to the CSS as well as inline images.
  - Save .gif files when there are few colors or the image is a vector to optimize for performance and quality.
  - Save .jpg files at 'High' in the Save for Web settings when using Photoshop.
  - All images should have alt tags describing the image and their height and width attributes should be set to aid with browser performance.

3.0 - DELIVERING TEMPLATES

- Please make sure the CSS and HTML on all pages validates. You can use the Firefox web developer plug-in to test validation locally (Tools > Validate Local HTML and Tools > Validate Local CSS).
- Test the files in IE6, IE7, IE8, Firefox, and Safari. Our requirements for IE6 are that it displays, but it does not have to be pixel perfect.
- Make sure there are no 404s before sending the final package. This can most easily be accomplished by viewing the Net/Network tab in the Firebug extension or Webkit Developer Tools.

4.0 - THE DEFAULT PACKAGE FILE DIRECTORY

/images

/backgrounds - All backgrounds go here.

/buttons - All button images go here.

/icons - All icons go here.

/promos - All images for promos are here.

bullet.gif - This is the image for use with unordered lists.

logo.gif – The logo for the project

/javascripts

/app - All page specific Javascripts go here.

application.js - **All** application wide JavaScript goes here. This is where you start the JavaScript development. There is a function for logging JavaScript that can be called with the following:

log('inside coolFunc', this, arguments);

css3.js – add css3 transitions here if they're not supported by the browser

/libs – A folder with libraries written by someone else. JQuery, belatedpng, modernizer, should be here.

If you add new libraries, place them here.

jquery-1.6.1.min.js – the jQuery library.

modernizr.min.js – A JavaScript library for HTML5 tests. http://www.modernizr.com/ This should be included in the Head of your page to add HTML5 elements to IE. Review the file to see what tests are available and visit the website if you need to add more tests.

plugin-template.jquery.js – A template for creating a jQuery plugin.

simple-plugin-template.jquery.js – A simpler jQuery plugin template.

/stylesheets

960-12.css – If we're using the 12 column 960 grid, copy the file into all.css

960-16.css – If we're using the 16 column 960 grid, copy the file into all.css

all.css - Styles for all media types.

print.css - Styles for printing a page. This should mostly just give elements on the page the display:none attribute.

handheld.css – Styles for handheld devices.

index.html - A basic file for starting the site with most relevant HTML elements for style.

humans.txt – Read more: http://goo.gl/IjffW

robots.txt – Read more: http://goo.gl/JzIqS

5.0 - IE SPECIFIC STYLES

If you want to style for IE specifically, you can use the following classes that get added to the HTML node via conditional statements and Modernizr.

IE6 - ie6 oldie Example: .ie6 {}

IE7 - ie7 oldie Example: .ie7 {}

IE8 - ie8 oldie Example: .ie8 {}

For browsers that do not have JavaScript support, you can use the no-js class.