

Welcome to Cybage



Document Name	Standard and Guideline - General
Version No.	V.1
Release Date	

This document of Cybage Software Pvt. Ltd. is for restricted circulation. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means – recording, photocopying, electronic and mechanical, without prior written permission of Cybage Software Pvt. Ltd.

Document Template History

Version No.	Authored / Modified by	Reviewed by, Date	Approved by, Date	Date Remark / Change History
V0.1	Awesh, Yogesh G			
V0.2				

This document contains guidelines for web applications. This document's primary motive is to implement code consistency and best practices to all UI projects. By maintaining consistency in coding styles and conventions, we can ease the burden of code maintenance, and mitigate risk of breakage in the future.

By adhering to best practices, we ensure optimized page loading, performance and maintainable code.

Table of Contents

1. Objective	4
2. Application Structure	4
3. HTML Coding Standard	5
DOCTYPE	5
Tags must be correctly nested	5
Capitalisation of attributes	5
Title	6
Quote marks	6
Always close your tags	6
Use lower case markup	6
4. CSS Coding Standard	6
General Guidelines	6
Reset CSS	6
CSS file structure	7
Naming convention of Classes and IDs	7
CSS Formatting	7
CSS Shorthand	8
Zero values	9
Font sizing	10
Downlevel browsers	10
5. References	10

1. Objective

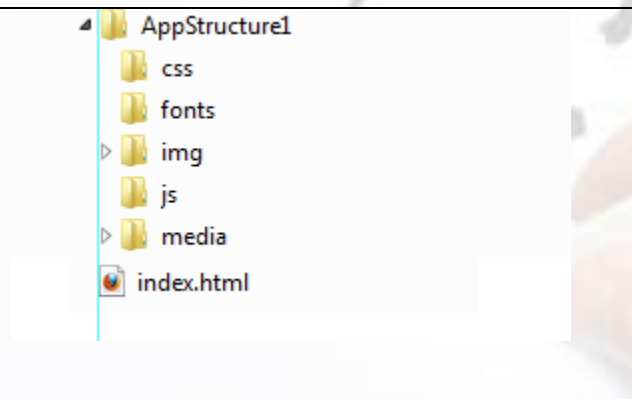
This document contains guidelines for web applications. This document's primary motive is to implement code consistency and best practices to all UI projects. By maintaining consistency in coding styles and conventions, we can ease the burden of code maintenance, and mitigate risk of breakage in the future.

2. Application Structure

Organizing web directories is as important as optimizing code. Creating a clean structure to work within will be beneficial when we or our team have to add or edit files.

Here are some recommended folder structures to be followed for Our UI Projects.

Directory Structure 1:

	<p>This structure is suitable for small scale projects. This structure will have separate folders for varying filetypes:</p> <ul style="list-style-type: none">a. styleb. fontsc. imagesd. Scriptse. media <p>HTML pages will be placed in the root folder.</p>
--	--

Directory Structure 2:

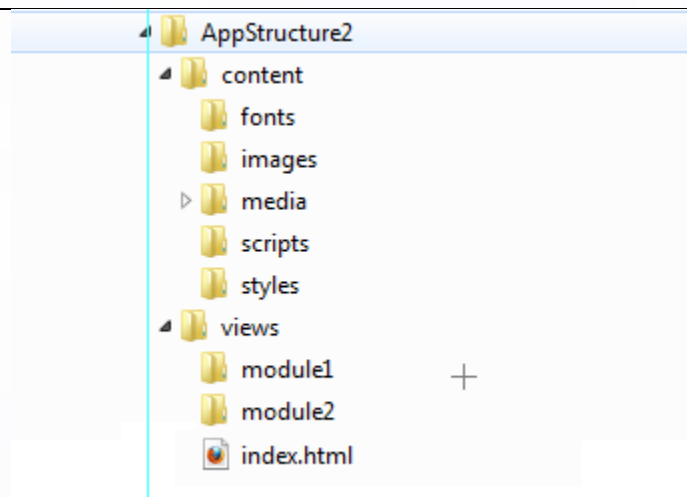
	<p>This structure consists of two main folders for varying filetypes:</p> <ul style="list-style-type: none">a. Content This folder will have sub folder for various assets for the project. Images, Scripts, Styles, Media, Fontsb. Views This folder will have all the HTML pages, sub folder can be created for different modules
---	--

Image Folder	Will have all final cropped .jpg, .png, .gif files
Styles Folder	Will have all CSS files such as main stylesheet, stylesheets for reset and/or print etc. in case of LESS files - base.partial.less, global.partial.less, desktop.less, phone.less, tablet.less should be seperated
Script Folder	Will have all Script files
Media Folder	Will have all video or flash files
Font Folder	Custom fonts folder will have .eot, .woff, .svg, .ttf files

3. HTML Coding Standard

Semantic HTML is processed by regular web browsers as well as by many other user agents. CSS is used to suggest its presentation to human users. Creating valid pages is essential to form a solid foundation for the CSS and JavaScript to build upon.

There are a number of tasks that need to be completed before a page will validate against the rules as set out in the XHTML1 Transitional DOCTYPE.

DOCTYPE

The DOCTYPE tag is used to declare the DTD (Document Type Definition) for an (X)HTML document. This tag is mandatory and must appear at the top (on the first line) of all HTML code. If the DOCTYPE tag is not present, then it is not HTML code. The document is validated against the rules within the DTD.

DTD declaration:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

For HTML 5: HTML5 (HTML syntax) is preferred for all HTML documents:

```
<!DOCTYPE html>.
```

Tags must be correctly nested

When placing tags inside other tags, they must be closed in the correct order.

Wrong: `<p>This is My Domain</p>`

Right: `<p>This is My Domain</p>`

Capitalisation of attributes

Contents of tags except for values must be in lowercase.

Wrong: `My Domain`

Right: `My Domain`

Title

The `<title>` tag helps make a web page more meaningful and search-engine friendly.

Wrong: `<title>Home Page</title>`

Right: `< title> Outsourced Product Development | IT Services | Enterprise Business Solutions – Cybage</ title>`

Quote marks

All attribute values must be enclosed in quote marks.

Wrong: `Home`

Wrong: `Home`

Right: `Home`

Always close your tags

Closing all your tags is a W3C specification. Some browsers may still render your pages correctly (under Quirks mode), but not closing your tags is invalid under standards

Use lower case mark-up

It is an industry-standard practice to keep your mark-up lower-cased. Capitalizing your mark-up will work and will probably not affect how your web pages are rendered, but it does affect code readability.

4. CSS Coding Standard

General Guidelines

- Add CSS through external files. It should always be added in the HEAD of the document
- Don't write inline styles in the HTML document
- Elements that occur only once inside a document should use IDs, otherwise, use classes. Always try to avoid using IDs in CSS file. We can leave IDs for Java Script
- Refer following links to know how to write CSS:
 - https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Writing_efficient_CSS?redirectlocale=en-US&redirectslug=CSS%2FWriting_Efficient_CSS
 - http://www.stuffandnonsense.co.uk/archives/css_specificity_wars.html

Reset CSS

The goal of a reset style sheet is to reduce browser inconsistencies in things like default line heights, margins and font sizes of headings etc. The Reset CSS file removes and neutralizes the inconsistent

default styling of HTML elements, creating a level playing field across browsers and providing a sound foundation upon which you can explicitly declare your intentions.

The rules contained in the reset CSS carry out the following functions amongst others:

- Remove all default margins, padding and borders from all HTML elements.
- Remove numbering and bullets from lists.

CSS file structure

CSS file structure can be defined as follows:

- A Global stylesheet that will include:
 - Site wide Reset styles.
- A View specific stylesheet. This will include styles for:
 - A Homepage only stylesheet.
 - Layout rules for defined layout templates for this specific view.
 - Rules specific to the branding of a particular view.

A print stylesheet containing print only rules will be defined using the @media rule at the bottom of the Global stylesheet. This structure creates a clear hierarchy of styles that will promote future maintainability of code.

Separate CSS can be created for ie7/8/9.

Naming convention of Classes and IDs

When creating new styles, developers must be careful to create class names that describe the content they relate to and not what the visual appearance of that content is. For example '**larger_blue_left_text**' may well be a perfectly valid name for a particular type of heading now, but if the designs were to change in the future and this same content now becomes right-aligned and red the semantic value for this element is lost.

To create consistency all class and id names will always be in lowercase with an underscore being used to denote word separation where necessary.

Classes and ids **must not begin** with a digit; only with a lowercase alpha character

Wrong: `.11lineText_blue`

Right: `.note_text` or `.noteText`

CSS Formatting

At minimum, format CSS with selectors on one line and each property on its own line. The declarations are indented.

As an enhancement to that style, related or child styles and additional 2 or 4 spaces. That allows for hierarchical scanning and organization and makes an easier-to-read style sheet.

Also, if you specify multiple selectors, it's a good idea to start each on new line.

```
.post-list li a{
    color:#A8A8A8;
}
.post-list li a:hover{
    color:#000;
    text-decoration:none;
}
.post-list li .author a,
.post-list li .author a:hover{
    color:#F30;
    text-transform:uppercase;
}
```

CSS Shorthand

The following CSS declaration is perfectly valid code. However, this is a very long-winded example of how this simple set of style declarations can be set.

```
.classname {
    color           : #333333;
    margin-top      : 3px;
    margin-right     : 0px;
    margin-bottom    : 3px;
    margin-left      : 0px;
    font-weight      : bold;
    padding-top      : 0px;
    padding-right     : 0px;
    padding-bottom    : 0px;
    padding-left      : 0px;
    font-size        : 1.6em;
}
```

Following is the shorthand version of the above example CSS.

```
.classname {
    color           : #333333;
    margin          : 3px 0px 3px 0px;
    font-weight      : bold;
    padding          : 0px 0px 0px 0px;
    font-size        : 1.6em;
}
```

However, this can be further trimmed down to the following.


```
.classname {  
    color          : #333;  
    margin         : 3px 0;  
    font-weight    : bold;  
    padding        : 0;  
    font-size      : 1.6em;  
}
```

1. The color declaration was #333333 and is now #333 as pairs of values can be compressed further. A clearer example of this is #336699 being compressed to #3369. However #123456 cannot be compressed further as the RGB values are not pairs. As a general rule of thumb, web safe colours can be compressed in this way.
2. The margin values were 3px 0px 3px 0px and are now 3px 0 as CSS values are read as top, right, bottom and then left. When only two values are given, the first value is given to the top and bottom and the second value to the left and right.
3. The padding values were 0px 0px 0px 0px and is now just 0. When only one value is declared, this value is given to all 4 rules as described in 2.
4. If three values are declared the first value is applied to top, the second value to left and right and the third value applied to the bottom.

This shorthand technique can be applied to the following rules:

- Margin
- Padding
- Border

Shorthand can also be applied to the background property.

```
.classname {  
    background-image : url(/images/en_UK/btn_arrow_blue.gif);  
    background-color : #fff;  
    background-repeat : no-repeat;  
    background-position : 0px;  
}
```

Can be shortened to:

```
.classname {  
    background : #fff url(/images/en_UK/btn_arrow_blue.gif) no-repeat 0 0;  
}
```

Using the format background-color, background-image, background-repeat then background-position.

Zero values

When declaring values that are measurements and have a value of 0, a unit is not needed alongside this.

For example **margin: 0px only needs to be margin: 0.**

Font sizing

All font size declarations will be defined in ems. Using the em unit allows the user to resize the font size to their own requirements.

As part of the reset rules, the font size for body is defined as 62.5% for all elements. This brings the default font size down to the equivalent of 10px for the majority of modern browsers. From this baseline, developers can easily, simply and predictably make an element any font size.

For example if a font size of 14px is required for a heading, the declaration for that particular element will be defined as 1.4em multiplying the base font size of 10px by a factor of 1.4 resulting in a font size of the equivalent of 14px.

Downlevel browsers

In general: functionality will be universal (given that completely outmoded browsers aren't in use, e.g. IE5). Look-and-feel, however, will gracefully degrade. We should not put an inappropriate amount of effort into trying to apply a universal look-and-feel across browsers, especially as the rate of browser releases is increasing.

For IE9 and earlier, it is possible to target them individually using Conditional Comments, e.g.:

```
<!--[if IE 6]>  
<link rel="stylesheet" type="text/css" href="c/ie6.css">  
<![endif]-->  
  
<!--[if IE 7]>  
<link rel="stylesheet" type="text/css" href="c/ie7.css">  
<![endif]-->  
  
<!--[if IE 8]>  
<link rel="stylesheet" type="text/css" href="c/ie8.css">  
<![endif]-->
```

Testing and support requirements should take this on board; we should test for functionality before concerning ourselves with exact look-and-feel matching.

5. References

http://www.digital-web.com/articles/keep_it_simple_stupid/

http://en.wikipedia.org/wiki/Separation_of_concerns

http://en.wikipedia.org/wiki/Semantic_HTML#Semantic_HTML

<http://www.w3.org/TR/xhtml1/>

<http://www.cs.tut.fi/~jkorpela/quirks-mode.html>

<http://meyerweb.com/eric/tools/css/reset/>

<http://prem.ghin.de/2008-06/creating-sites-with-flexible-editable-layout-templates/>

[http://en.wikipedia.org/wiki/Namespaces_\(computer_science\)](http://en.wikipedia.org/wiki/Namespaces_(computer_science))

<http://www.thefutureoftheweb.com/blog/writing-semantic-html>

<http://taitems.github.com/Front-End-Development-Guidelines/>

<http://meyerweb.com/eric/tools/css/reset/>

[http://www.epochdev.com/blog/tutorials/html_development/get_organized-create a clean file and folder structure](http://www.epochdev.com/blog/tutorials/html_development/get_organized-create_a_clean_file_and_folder_structure)

