Best Software Engineering Practices & Coding Standards

This document gives a high level overview of Front-end Software Engineering Best Practices, Coding Standards and Technology Framework at Shopping.

If in any doubt on any aspects of the documentation, please speak to your Development Lead or Line Manager.

If this copy of the document is hard copy, it is likely to be an obsolete version. Please check with the Development Lead or Line Manager, to ensure that you are referring to the current version.

Best Practices

1. Have a clear understanding of your **task**. If not, follow up with your Line Manager or peer who will help or point you in the right direction.
2. Do you have all the necessary **software** and development environment to do your task? See http://62.60.21.47/display/MS/2.6.12+GIT
3. If your task includes **functional changes** that are design driven, have you checked you have latest update of the design(s)?
4. Use **software and frameworks** recommended by Business, and are open source.
5. Do you have all the necessary **tools for testing**? Use W3C and other recommended validators.
6. Before you commit any changes to SVN or GIT, first request a **technical peer review and visual check** (from Test Engineer or Creative Design) to first check the work and provide feedback. Note you own your work and responsible for it working and failing.
7. All related **documentation** for the integration of your work at different stages MUST be completed. These include providing summary of your solution in some cases, updating JIRA.
8. All **blockers** must be assigned to Line Manager for accessing.
9. Use **white spaces, tabs and indentation** to make your code more human readable. This will allow quick access and to discern to any part of your code if you need help or during code review. Use consistent vertical line spacing in your code.
10. **Don't chain multiple lines of code** on the same line, and each line of complete statement MUST be terminated by a semi-column. Even if it is the last line of code.
11. Use **shorter syntaxes** and notations to write less lines of code where possible.
12. **Avoid in-line scripting** with Javascript and CSS
13. Use techniques that will enable you to **write less** lines of code, and remove foot prints during execution. Including Delegation, Closures, Self References, Caching etc.
14. Do not execute any function call using **in-line event handler** in your html or PHP pages. Trigger using classes and IDs from external scripts. Avoid calling lengthy hierarchical DOM elements in any event call.
15. Avoid downloading software that can **potentially harm business operations**. If you are not sure please check. Note, you are responsible for your actions.
16. Add **useful comments** in your code. First, it helps to explain why you possibly choose an implementation over another, and also when you need help. When you are away from work, it will provide useful documentation to the business.
17. **Remove all debugging scripts** and placeholder content before you commit code.
18. Use **CSS reset.**
19. Note by placing **JavaScript files at the bottom** of your documents, you'll ensure that JS files will be loaded only when the content has been properly displayed.
20. Use **HTML semantically**. Structure your documents by denoting structural semantics for text such as headings, paragraphs, lists, and others.
21. Use **HTML5 Doctype** and HTML5 features when appropriate.

22. All markup should be delivered as **UTF-8 for character encoding**, as its the most friendly for internationalisation. It should be designated in both the HTTP header and the head of the document.

23. Do not use the **size attribute** on your input fields. The size attribute is relative to the font-size of the text inside the input. Instead use CSS width.

24. Place an **html comment** on some closing div tags to indicate what element you're closing. It will help when there is lots of nesting and indentation.

25. Add CSS through **external files**, minimising the number of files, if possible. It should always be in the HEAD of the document. Use the LINK tag to include, never the @import. The @import is a slow directive.

26. Elements that occur only once inside a document should use IDs, **otherwise use classes**.

27. Understand **cascading and selector specificity** so you can write very terse and effective code.

28. Intimate knowledge and understanding of the CSS and browser-based **box model** is necessary for conquering fundamentals of CSS layouts.

29. **Test while you build** to avoid cross-browser issues.

30. Use **pseudo-classes** to enable you dynamically style content.

31. Use **combinators & attribute selectors** to provide shortcuts for selecting elements that are a descendant element, a child element, or an element's sibling.

32. Use the **px unit of measurement to define font size**, because it offers absolute control over text. Using the em unit for font sizing used to be popular, to accommodate for Internet Explorer 6 not re-sizing pixel based text. However, all major browsers (including IE7 and IE8) now support text re-sizing of pixel units and/or full-page zooming. Since IE6 is largely considered deprecated, pixels sizing is preferred. Additionally, unit-less line-height is preferred because it does not inherit a percentage value of its parent element, but instead is based on a multiplier of the font-size.

33. You should be aware of the TRBL acronym, denoting the order in which the sides of an element are defined, in a clock-wise manner: **Top , Right, Bottom, Left**. If bottom is undefined, it inherits its value from top. Likewise, if left is undefined, it inherits its value from right. If only the top value is defined, all sides inherit from that one declaration.

34. Verify that your **colour management** settings in your Photoshop is consistent with colour management settings in the team.

35. When using **@font-face** it's recommended to declare the source for each font format. This is important if you want it to work in the most number of browsers (woff: WOFF (Web Open Font Format)ttf: TrueType ttf, otf: OpenType eot: Embedded OpenType svg, svgz: SVG Font)- Cross-Browser Compatibility: Safari, IE 6-9, IE 9 Compatibility Modes, Firefox, Chrome, iOS, Android and Opera.

36. Ensure you run all your changes and code fixes on versions of your local build before you commit back to repository.

37. Lunch times are between the hours of 12:00 GMT to 14:00 GMT, and for an hour.

38. Attend all meetings and on time.

Coding Standards

JavaScript

JavaScript should be used entirely unobtrusively and only for the purpose of behaviour not presentation.

All JavaScript should be applied using a progressive enhancement strategy. Fundamentally, this means that the site must work without the use of JavaScript in the first place and when the JavaScript layer is applied it simply enhances the original behaviour.

- **JavaScript - Best Practices**
- **Javascript - FEAR**
- **ECMAScript6 - ECMAScript 2015**
- **jQuery**

HTML

Each document must be written in HTML5. In order to facilitate this please use the HTML5 Boilerplate as a starting point. It can be found here:

http://html5boilerplate.com/

Depending on specific project requirements, all of the features of the boilerplate may not be required. You can customise the boilerplate with what you need

- **Character encoding**

All files must be saved as utf-8 encoded and the specified charset for any html file must also be utf-8.

CSS

All style sheets should be placed within the head of the HTML document. No style elements or inline styles are permitted within the HTML document. Separate Internet Explorer style sheets are NOT permitted and browser hacks are also discouraged. IE HTML tag classes are used within the HTML5 boilerplate to allow for the targeting of specific IE versions. The number of style sheets should be kept to an absolute minimum.

- **FEAR** CSS
- CSS3
- **Sprites & background images**

When using sprites to display background images (icons, etc...), use the defined best practice as follow:

- normal resolution sprite image should be used by all devices by default
- never use background-size for this default sprite (no support for IE8) - it should be displayed with original dimensions
- use media queries to provide high resolution sprite for Retina and other high definition devices
  - @media only screen and (-webkit-min-device-pixel-ratio: 2) { ... }
  - @media only screen and (-webkit-min-device-pixel-ratio: 1.5) { ... }
- resize the high resolution sprites using background-size property
  - resize to half the high resolution image size
  - ex: sprite image is 200px x 1000px
  - use: background-size: 100px 500px;
- never use the shorthand with background-size (no support for iOS 6.1)
  - background: transparent url(../image.png) 10px 30px / 50% no-repeat;
  - this breaks the whole background property in iOS 6.1
- for each project, use a consistent naming convention for images
  - image.png
  - image-hd.png
  - or
  - /sprite/image.png
  - /sprite-hd/image.png
- when saving sprite images, always use even dimensions to avoid half-pixel issues
  - good: 36 x 822 (background-size: 18px 411px;)
  - bad: 35 x 822 (background-size: 17.5px 411px;) not reliable, may cause sprite to be off by 1 pixel left or right
- **Code example (sprite dimensions are 80 x 1000):**

```
.icon {
    width: 10px;
    height: 10px;
    background: transparent url('../icons.png') -20px -500px no-repeat;
}
@media only screen and (-webkit-min-device-pixel-ratio: 1.5) {
    .icon {
        background-image: url('../icons-hd.png');
        background-size: 40px 500px;
    }
}
```

- **Coding Styles**

Developers have many different coding styles. Whiles no style is "correct", we ask that consistent visual appearance of source code is applied throughout any given project. Moreover if you inherit a project, the previous coding and indent styles should be maintained for the sake of consistency and readability.

For instance, if a previous developer has coded their CSS as such:

```
div{margin:0;padding:0;}
```

You would be expected to maintain the above style rather than do this:

```
div{
   margin:0;
   padding:0;
}
```

Additionally, please do not indent tabs. Instead use space for consistency of code layout across editors. In some editors, the TAB key can be configured to indent by a given number of spaces rather than the default behaviour.

Java & WCS Support

- JAVA & WCS Support

Accessibility

All applications should take into consideration Web Content Accessibility Guidelines and Search Engine Optimization. This means documents should be written semantically correct, to aid assistive tecnologies and provide support to all users.

Guidelines can be found below:

http://www.w3.org/WAI/intro/wcag.php

http://www.w3.org/WAI/intro/aria

Performance

Performance is assessed against YSLOW guidelines and other certified debugging tools. All developers must install the YSLOW extension for Firefox and aim at grade 'A' against each performance rule. Documentation can be found here;

http://developer.yahoo.com/yslow/help/

- **Minification:**

  Both combined and minified versions of style sheets and JavaScript files should be maintained alongside the human readable versions with the former being deployed and the lattter being used for development.

  HTML5` Boilerplate comes bundled with a build script that performs this task. Alternatively, tools such as the Google Closure compiler - http://closure-compiler.appspot.com/home may be used for JS and CSSmin - http;//tools.w3clubs.com/cssmin/ for CSS.

Browser Support

M&S Shopping supports the latest version of the following browser, plus the previous release. Always check with the Tech Lead on any given project to see what browsers are supported according to the Statement of Work. Otherwise, the supported list is as follows:

- **PC**

  Google Chrome
  Safari
  Firefox
  Opera
  Internet Explorer (IE7 min), and check for graceful degradation.

- **MAC**

  Google Chrome
  Safari
  Firefox
  Opera

Folder Structures

Folder structures may differ from project to project depending on what technologies are used. Where the Front-end Developerhas a degree of control, however, please ensure that the following conventions are applied:

Images should be saved in **/img**

Style sheets should be saved in **/css**

CSS sprite PSDs should be saved in **/css/sprites**

JavaScript should be saved in **/js**

JavaScript libraries should be saved in **/js/libs**

Subfolders for internal html pages and images may be saved where necessary.

Version Control

M&S Shopping uses GIT for version control on projects. Please see the Tech Lead or Development Manager on the project in order to obtain and account.

Information on GIT can be found here:

Accessing M&S Shopping GIT

GIT absolutely must be used for all deployments from development to staging. No file transfer outside GIT is permitted.

References

- JavaScript best practices
- https://github.com/johnpapa/angularjs-styleguide
- http://campus.codeschool.com/courses/shaping-up-with-angular-js/contents
- http://cssguidelin.es/