Scandiweb

Magento development guidelines

1 Code standards & formatting

- Use the coding standards from Zend Framework to improve readability en structure of coding http://framework.zend.com/manual/en/coding-standard.overview.html
  See also here how Magento 2 team hass adapted Zend standarts ➜ https://wiki.magento.com/display/MAGE2DOC/PHP+Coding+Standards+and+Practices#PHPCodingStandardsandPractices-4.CodeSize
- We use Magento standard for indentation, which consists of 4 spaces and NO TABs if you see any code file with TABS, please convert those files to use 4 spaces instead.
- Always commit files where you change from Tabs to spaces separately from actual changes you intend to do to these files to ensure readability of the changeset (mixing tab➜space conversions with code changes makes the “real” changes very difficult to see in the changeset)
- Line endings LF (no CR and no CRLF)
- Encoding UTF-8
- Maximum line length 120char, target 80
- There should be no trailing spaces on any line (configure your text editor to remove trailing spaces on save)
- PHP shorttags must not be used in Magento
- Do not use sql querys but always create models for fetching data from (custom) database tables.
- Create/use (custom) cache for resource intensive code parts.
- Place js and css in file instead of templates and load them trough the xml files.
- Don’t leave commented code in source unless there is a reason for that. We use SVN/GIT for most of the projects to keep the history of changes. No need to leave old/unused code in there.

2 Do not ever modify core files

- When working in Magento you should work only in your theme folder or your extensions folder. At no point (other then to test something quickly) you should be thinking about modifying magento core files.
- By core files we mean;

  app/code/core
  app/design/frontend/base
  skin/frontend/base

3 Theming specs

- Use Magento fall back functionality to as great extent as possible. When building new theme, only copy the modified files from fall back layer into your theme. Never make changes directly into fall back layer.
- Wrap the hard-coded text in template file into the translate method. Normally like this $this->__('hard coded text'). Basically, this would be text that the user sees on the page or text that appears inside the "alt" and "title" attributes. Look at the file app/code/core/Mage/Core/Helper/Abstract.php to see what it does.
- When reffering to local directories always use Magento built in functionas. e.g. $dir = Mage::getBaseDir('var') instead of directly (e.g. $dir = 'var/');
- Can't use img tag like this.

  You should do
  <img class="fb" src="<?php echo $this->getSkinUrl('images/fb.png') ?>"/>
- All layout changes are to be done in local.xml only. How to make changes only in local.xml is described here -
- Better way to modify Magento layout - http://goo.gl/DZ3qJ
- Working with Magento layout files
- http://magebase.com/magento-tutorials/demystifying-magentos-layout-xml-part-1/
- http://magebase.com/magento-tutorials/digging-deeper-into-magentos-layout-xml-part-2/
- http://magebase.com/magento-tutorials/5-useful-tricks-for-your-magento-local-xml/
- Always make sure that all the blocks that we won't use in this website are removed (for example some webshops will not feature compare or wishlist functionality so the sidebar blocks and customer account mentu items need to be removed from the site in that case).
- When working on templates **Don`t copy all theme files, only the absolutely necessary ones**. If it is absolutely necessary to change a file copy it from the 'app/design/frontend/base/dafault/' folder (or for enterprise versions -> enterprise/default) in to your theme folder and make the adjustments. You should use as few template files as possible in the theme folder. That way it's easy to upgrade to a new version of Magento. :)
- Create a translate.csv file but don't use it too much. More than 50 lines should ring a bell that there's something wrong with the method your using to translate the theme.
- For all changes in magento admin area as well as for all changes to database use all Magento's database install/upgrade scripts. You can create a module named [projectname]_migrations which will do jsut that - update magento with latest database and configuration changes - all changes to database should be written in code and added as extension upgrade scripts.

4 Templating blocks

We want as few non text HTML tags as possible in our static blocks. That's why we use templates to load in a block. You can use a template form more then one block. So you can make a template for sidebar, footer, header, maincontent, etc.

LOCAL.XML or PAGE LAYOUT

```
<block type="core/template" name="…" template="…">
    <action method="setIdentifier">
        <identifier>…</identifier>
    </action>
</block>
```

TEMPLATE FILE

```
<?php
// Get the block indentifier
$block_identifier = $this->getIdentifier();
// Load the block
$block = Mage::getModel('cms/block')->load($block_identifier);
?>
<div class="block block-cms">
    <div class="block-title">
        <h3><?php echo $this->__($block->getTitle()); ?></h3>
    </div>
    …
</div>
```

5 Deploy process

- Insert client's name and e-mail address for Magento admin: System ➜ My Account
- Insert client's e-mail address for Store e-mails: System ➜ Configuration ➜ Store Emails
- Set robots indexing to Index, Follow: System ➜ Configuration ➜ Design, Default Robots
- Please check product and CMS page editing functionality
- Go through website and perform a Checkout
- Check IE for custom fonts, pop-ups and overlays
- JS & CSS merge must be tested and working when site has been finished and si going to be deployed.

Scandiweb

Magento page styling guide

General

- For elements like paragraphs, headings (h1,h2,h3 etc), buttons, tables, lists and other elements that repeat all over the website always let them inherit styles from base template if not specified otherwise.
- If page has very specific elements that differ from those on other pages please always use as specific css style declaration as possible in order not to break the other pages.
- For pages which include forms where users can enter data always test on all browsers how the error message looks and if it is properly styled.
- All buttons need to have cusors:pointer; on hover.

1 Base template

- If working in a team divide css in multiple css files so each page has its own styles in it. If working alone please use single css file
- Go through the all designs and check which design elements are repeating accross the pages and style them so they are included in base. And when-newly created in the template they would appear styled already.
- Add width and other general css rules to col-main, col-right & col-left.
- All the buttons need to be created using sprite technique in single file. Sprite example - if do`t know how to create - consult Raivis
- Style general look of sidebar blocks using class name .block (not .block related,.block-viewed etc)
- Style the alert messages according the design
- Check all texts in designs and all links and add to css as global rule the onestyle that is repeated the most.
- Disable modules that you see are not included in the designs so they dont appear on pages (ask PM to be sure)

- Add the site logo for e-mail templates & pdf invoices
- Enable gzip compression to speed up magento http://www.xpcdesign.com/2008/09/speed-up-magento-by-235/
- Think globally not only locally(meaning styling only for single page)
- If there is @font-face used in the project and client has sent-in the font - please make sure that it is the correct font - the same one which is used in the psd file. If it is not - ask PM for the correct font. Otherwise it is impossible to style the page exactly according the design as each font has different line heights & sizes and it can create very big problems afterwards.

2 Contact form

- For input fields always add some minor padding so the text gets does not collide with borders
- Error messages need to be tested in all browsers so when they appear they don`t overflow the fields but fields gracefully slide down giving a space for error messages to appear.
- Example of well styled contact form http://screencast.com/t/PQxgXQdo
- Example of well styled contact form error messages http://screencast.com/t/GBWHxEJ6ix

3 CMS pages

- Always style such elements for the CMS page even if they are not included in the design:
  - Heading 1, 2, 3, 4, 5, 6 - if not specified otherwise in the design h4 should be same size like the paragraph text in the website + bold. Each level up or down then is styled adding or decreasing font size by 2px.
  - Paragraph text always should have bottom padding. If not specified in the design it should be set to 25px. Line height must be set according to design. If not specified, general rule for line height to be 1.6. - example: p {line-height:1.6;} Rules for paragraph text must be added to the base css if they are abscent.
  - Table must always be styled properly, that includes - table has 1px borders, cellpadding, cellspacing.
  - Always make sure the styles you implemented on 2 column left cms page template work also on 3 column, 1 column & 2 column right cms page templates.
  - In the sidebar for 2 column left, 2 column right & 3 column cms page templates leave same items/static blocks which are on the product view sidebar. If in doubt - ask project manager.
  - Pictures in the cms pages if not designed otherwise must have margin 20px, 3px padding & 1px light grey border around them
  - Links in the cms pages must be have same style which do have other links on the page. Need to make sure that they use the style from the base css. If not - include the style in the base css before that consulting project manager.
- Examples of good styled cms pages:
  - http://screencast.com/t/ukzqIzBcOBmS
  - http://screencast.com/t/HeiBen4zaM

3 CMS pages

- Customer account includes many pages and many elements to be styled and can take up much time to style BUT if you use basic css cascading principles and style first page which is account dashboard all other pages will get styled as well and your development time will decrease by half or more.

For this to work you need to add a new class name for all the pages in customer account. Add class name “custumer-account-page” to the body tag. Use following reference in local.xml file:

```
<customer_account>
  <reference name=”root”>
      <action method=”addBodyClass”>
<className>customer-account-page</className>
</action>
   </reference>
</customer_account>
```

- Before styling the customer account please create an actual custumer account, enter at least 3 addresses, buy and complete (invoice & ship) at least 3 purchases (simple product, configurable product, multiple products in one purchase), add 3 different products to the wish-list.

  - Pages which ALWAYS need to be styled in customer account:
  - Account dashboard
    http://screencast.com/t/zvg2hX3Hw
  - Account information
    http://screencast.com/t/6GM2uBsBL
  - Address book
    http://screencast.com/t/ZGeFwqX6Je
  - My orders
    http://screencast.com/t/exMo81Rv
    - View order page
      http://screencast.com/t/ ZER58aHklB
    - Invoices page
      http://screencast.com/t/8W9kbUDZ
    - Shipments page
      http://screencast.com/t/Kh1of09Z

- For styling please use base css styles everywhere where it is possible. If style is very specif and base css does not include style you need - create a very specific css rule for the element so it gets displayed ONLY where you need it.
- For example: .customer-account-index .my-account .welcome-msg {font size:13px;}
- Pagination must be styled for order table, page review table, etc.

To understand which element are included in typical customer account, which do repeat all over customer account and which are the elements that need to be styled in a typical Magento custumer account where no blocks are taken away please use this style guide - https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B_JShG-1RnR7ZDBlOTJlMmEtMmEwNi00YWUzLWJkMzYtNTRiZDhkMmUyY2Zk&hl=en_US

Download it, print it and keep it in front of you when working on a customer account.

The main rule - all elements that have the same name in the style guide must look the same on the page as well - that includes their font size, padding, margins, width & height! If you have styled one element, for example a table - then all other tables in customer account must appear styled right away as well, if that does not happen - something is wrong with your css or you have forgotten to add the general class to the body tag of all pages in customer account.