

**Coding Conventions**

Table of Content

[PHP Coding Standard – Reference 3](#_Toc322680741)

[Overview of Reference 3](#_Toc322680742)

[PHP File 3](#_Toc322680743)

[Naming Zend Classes 3](#_Toc322680744)

[Variables 3](#_Toc322680745)

[Comments 3](#_Toc322680746)

[Coding 4](#_Toc322680747)

[HTML Coding convention 4](#_Toc322680748)

[Styling 4](#_Toc322680749)

[HTML 4](#_Toc322680750)

[Images 4](#_Toc322680751)

# PHP Coding Standard – Reference

<http://framework.zend.com/wiki/display/ZFDEV/PHP+Coding+Standard+%28draft%29>

# Overview of Reference

## PHP File

* For files that contain only PHP code, the closing tag ("?>") is to be omitted.
* Mixed php/html files use never short tags for <?php ?>!
* The target line length is 80 characters. However, longer lines are acceptable, but the maximum length of any line of PHP code is 120 characters.
* Use an indent of 4 spaces with no tag characters (configure your editor)

## Naming Zend Classes

* Controller are named like “IndexController” (postfix Controller, Controllername first letter uppercase)
* Actions are named like “indexAction” (postfix Action, Actionname first letter lowercase)
* Forms area also named with postfix Form (first letter uppercase) like “CampaignForm”
* Only Model Classes have none postfix just their name, like “Vision”
* Methods and variables are named in camel cases
* These naming conventions also apply to naming php-files.

## Variables

* Variable names may only contain alphanumeric characters. Underscores are not permitted. Numbers are permitted in variable names but are discouraged.
* Like function names, variable names must always start with a lowercase letter and follow the "camelCaps" capitalization convention.
* For String concatenation the “.” Operator should be used.
* Associative arrays should be declared in multiple lines like this:   
  $sampleArray = **array**('firstKey' => 'firstValue',   
   'secondKey' => 'secondValue');

## Comments

* Every Class has above its declaration several multiline comments with class name and its usage.  
  /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  
  /\* \*\*\*\*\*\*\* class controller \*\*\*\*\*\*\* \*/  
  /\* description text \*/  
  /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/
* Code comments are single comment lines permitted, but also if necessary multiline comments can be used (may these comments should be at the beginning comment of the class).
* Also every file should have at the beginning a multi-line comment, containing Filename, description, author, creation date, copyright  
  /\* \*\*\* File Name.php \*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  
  /\* Description: \*/  
  /\* Description Text \*/  
  /\* Author: Author Name \*/  
  /\* creation-date: 2012-04-20 \*/  
  /\* copyright: MassIdea, Zend \*/
* Every Function, Method or Operation should have a comment above its declaration, which tells the usage and the return type/value of the Function (etc).

## Coding

* The opening brace for function or method is always written on the line underneath the function or method name.
* Every loop or case (if, else, switch) should be start and end with a curly bracket. Code, that has only one line, mustn’t have brackets. [ if (...) { multi lines; } else { single line; } ]
* Inline functions shouldn’t be used too much, cause of higher complexity of reading. Sure, at return values it does make sense to use them, either longer inline functions cannot be read again.

# HTML Coding convention

<https://wiki.orangehrm.com/index.php/HTML_Coding_Conventions>

## Styling

* Only use CSS for styling, and do not use inline style attributes
* Use external CSS files instead of style tags in the HTML page
* Merge similar styling definition of classes together to cut the length.
* Using transformations like border-radius make sure you also add the definitions for all browsers.

## HTML

* All tags must end with an end tag.
* Use id only for container, that are need to edit by scripting

## Images

* Images used for the styling (like icon-image for navigation) should be realized by CSS
* All images end with the /> tag
* Content-images must have an alt-attribute.
* Use the .png-format for small images with less color
* And .jpg format for images with a lot of color
* Always compress images with <http://kraken.io/> to a maximum data-size of 250 kb
* Save content images into the root/images directory of the website, for component based images it is also permitted to save them in the component’s directory.