* [Welcome to CodeIgniter](http://docs.google.com/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/index.html)
  + [CodeIgniter URLs](http://docs.google.com/urls.html)
  + [Controllers](http://docs.google.com/controllers.html)
  + [Reserved Names](http://docs.google.com/reserved_names.html)
  + [Views](http://docs.google.com/views.html)
  + [Models](http://docs.google.com/models.html)
  + [Helpers](http://docs.google.com/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/libraries.html)
  + [Creating Libraries](http://docs.google.com/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/drivers.html)
  + [Creating Drivers](http://docs.google.com/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/autoloader.html)
  + [Common Functions](http://docs.google.com/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/routing.html)
  + [Error Handling](http://docs.google.com/errors.html)
  + [Caching](http://docs.google.com/caching.html)
  + [Profiling Your Application](http://docs.google.com/profiling.html)
  + [Running via the CLI](http://docs.google.com/cli.html)
  + [Managing your Applications](http://docs.google.com/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/alternative_php.html)
  + [Security](http://docs.google.com/security.html)
  + PHP Style Guide
* [Libraries](http://docs.google.com/libraries/index.html)
  + [Benchmarking Class](http://docs.google.com/libraries/benchmark.html)
  + [Caching Driver](http://docs.google.com/libraries/caching.html)
  + [Calendaring Class](http://docs.google.com/libraries/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/libraries/cart.html)
  + [Config Class](http://docs.google.com/libraries/config.html)
  + [Email Class](http://docs.google.com/libraries/email.html)
  + [Encrypt Class](http://docs.google.com/libraries/encrypt.html)
  + [Encryption Library](http://docs.google.com/libraries/encryption.html)
  + [File Uploading Class](http://docs.google.com/libraries/file_uploading.html)
  + [Form Validation](http://docs.google.com/libraries/form_validation.html)
  + [FTP Class](http://docs.google.com/libraries/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/libraries/image_lib.html)
  + [Input Class](http://docs.google.com/libraries/input.html)
  + [Javascript Class](http://docs.google.com/libraries/javascript.html)
  + [Language Class](http://docs.google.com/libraries/language.html)
  + [Loader Class](http://docs.google.com/libraries/loader.html)
  + [Migrations Class](http://docs.google.com/libraries/migration.html)
  + [Output Class](http://docs.google.com/libraries/output.html)
  + [Pagination Class](http://docs.google.com/libraries/pagination.html)
  + [Template Parser Class](http://docs.google.com/libraries/parser.html)
  + [Security Class](http://docs.google.com/libraries/security.html)
  + [Session Library](http://docs.google.com/libraries/sessions.html)
  + [HTML Table Class](http://docs.google.com/libraries/table.html)
  + [Trackback Class](http://docs.google.com/libraries/trackback.html)
  + [Typography Class](http://docs.google.com/libraries/typography.html)
  + [Unit Testing Class](http://docs.google.com/libraries/unit_testing.html)
  + [URI Class](http://docs.google.com/libraries/uri.html)
  + [User Agent Class](http://docs.google.com/libraries/user_agent.html)
  + [XML-RPC and XML-RPC Server Classes](http://docs.google.com/libraries/xmlrpc.html)
  + [Zip Encoding Class](http://docs.google.com/libraries/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

Toggle Table of Contents

[CodeIgniter](http://docs.google.com/index.html)

* [Welcome to CodeIgniter](http://docs.google.com/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/index.html)
  + [CodeIgniter URLs](http://docs.google.com/urls.html)
  + [Controllers](http://docs.google.com/controllers.html)
  + [Reserved Names](http://docs.google.com/reserved_names.html)
  + [Views](http://docs.google.com/views.html)
  + [Models](http://docs.google.com/models.html)
  + [Helpers](http://docs.google.com/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/libraries.html)
  + [Creating Libraries](http://docs.google.com/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/drivers.html)
  + [Creating Drivers](http://docs.google.com/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/autoloader.html)
  + [Common Functions](http://docs.google.com/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/routing.html)
  + [Error Handling](http://docs.google.com/errors.html)
  + [Caching](http://docs.google.com/caching.html)
  + [Profiling Your Application](http://docs.google.com/profiling.html)
  + [Running via the CLI](http://docs.google.com/cli.html)
  + [Managing your Applications](http://docs.google.com/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/alternative_php.html)
  + [Security](http://docs.google.com/security.html)
  + PHP Style Guide
* [Libraries](http://docs.google.com/libraries/index.html)
  + [Benchmarking Class](http://docs.google.com/libraries/benchmark.html)
  + [Caching Driver](http://docs.google.com/libraries/caching.html)
  + [Calendaring Class](http://docs.google.com/libraries/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/libraries/cart.html)
  + [Config Class](http://docs.google.com/libraries/config.html)
  + [Email Class](http://docs.google.com/libraries/email.html)
  + [Encrypt Class](http://docs.google.com/libraries/encrypt.html)
  + [Encryption Library](http://docs.google.com/libraries/encryption.html)
  + [File Uploading Class](http://docs.google.com/libraries/file_uploading.html)
  + [Form Validation](http://docs.google.com/libraries/form_validation.html)
  + [FTP Class](http://docs.google.com/libraries/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/libraries/image_lib.html)
  + [Input Class](http://docs.google.com/libraries/input.html)
  + [Javascript Class](http://docs.google.com/libraries/javascript.html)
  + [Language Class](http://docs.google.com/libraries/language.html)
  + [Loader Class](http://docs.google.com/libraries/loader.html)
  + [Migrations Class](http://docs.google.com/libraries/migration.html)
  + [Output Class](http://docs.google.com/libraries/output.html)
  + [Pagination Class](http://docs.google.com/libraries/pagination.html)
  + [Template Parser Class](http://docs.google.com/libraries/parser.html)
  + [Security Class](http://docs.google.com/libraries/security.html)
  + [Session Library](http://docs.google.com/libraries/sessions.html)
  + [HTML Table Class](http://docs.google.com/libraries/table.html)
  + [Trackback Class](http://docs.google.com/libraries/trackback.html)
  + [Typography Class](http://docs.google.com/libraries/typography.html)
  + [Unit Testing Class](http://docs.google.com/libraries/unit_testing.html)
  + [URI Class](http://docs.google.com/libraries/uri.html)
  + [User Agent Class](http://docs.google.com/libraries/user_agent.html)
  + [XML-RPC and XML-RPC Server Classes](http://docs.google.com/libraries/xmlrpc.html)
  + [Zip Encoding Class](http://docs.google.com/libraries/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

[CodeIgniter](http://docs.google.com/index.html)

* [Docs](http://docs.google.com/index.html) »
* [General Topics](http://docs.google.com/index.html) »
* PHP Style Guide

classic layout

[**PHP Style Guide**](#gjdgxs)[**¶**](#30j0zll)

The following page describes the coding styles adhered to when contributing to the development of CodeIgniter. There is no requirement to use these styles in your own CodeIgniter application, though they are recommended.

Table of Contents

* [PHP Style Guide](#30j0zll)
  + [File Format](#1fob9te)
    - [TextMate](#3znysh7)
    - [BBEdit](#2et92p0)
  + [PHP Closing Tag](#tyjcwt)
  + [File Naming](#3dy6vkm)
  + [Class and Method Naming](#1t3h5sf)
  + [Variable Names](#4d34og8)
  + [Commenting](#2s8eyo1)
  + [Constants](#17dp8vu)
  + [TRUE, FALSE, and NULL](#3rdcrjn)
  + [Logical Operators](#26in1rg)
  + [Comparing Return Values and Typecasting](#lnxbz9)
  + [Debugging Code](#35nkun2)
  + [Whitespace in Files](#1ksv4uv)
  + [Compatibility](#44sinio)
  + [One File per Class](#2jxsxqh)
  + [Whitespace](#z337ya)
  + [Line Breaks](#3j2qqm3)
  + [Code Indenting](#1y810tw)
  + [Bracket and Parenthetic Spacing](#4i7ojhp)
  + [Localized Text](#2xcytpi)
  + [Private Methods and Variables](#1ci93xb)
  + [PHP Errors](#3whwml4)
  + [Short Open Tags](#2bn6wsx)
  + [One Statement Per Line](#qsh70q)
  + [Strings](#3as4poj)
  + [SQL Queries](#1pxezwc)
  + [Default Function Arguments](#49x2ik5)

[**File Format**](#2p2csry)[**¶**](#1fob9te)

Files should be saved with Unicode (UTF-8) encoding. The BOM should *not* be used. Unlike UTF-16 and UTF-32, there’s no byte order to indicate in a UTF-8 encoded file, and the BOM can have a negative side effect in PHP of sending output, preventing the application from being able to set its own headers. Unix line endings should be used (LF).

Here is how to apply these settings in some of the more common text editors. Instructions for your text editor may vary; check your text editor’s documentation.

[**TextMate**](#147n2zr)[**¶**](#3znysh7)

1. Open the Application Preferences
2. Click Advanced, and then the “Saving” tab
3. In “File Encoding”, select “UTF-8 (recommended)”
4. In “Line Endings”, select “LF (recommended)”
5. *Optional:* Check “Use for existing files as well” if you wish to modify the line endings of files you open to your new preference.

[**BBEdit**](#3o7alnk)[**¶**](#2et92p0)

1. Open the Application Preferences
2. Select “Text Encodings” on the left.
3. In “Default text encoding for new documents”, select “Unicode (UTF-8, no BOM)”
4. *Optional:* In “If file’s encoding can’t be guessed, use”, select “Unicode (UTF-8, no BOM)”
5. Select “Text Files” on the left.
6. In “Default line breaks”, select “Mac OS X and Unix (LF)”

[**PHP Closing Tag**](#23ckvvd)[**¶**](#tyjcwt)

The PHP closing tag on a PHP document **?>** is optional to the PHP parser. However, if used, any whitespace following the closing tag, whether introduced by the developer, user, or an FTP application, can cause unwanted output, PHP errors, or if the latter are suppressed, blank pages. For this reason, all PHP files MUST OMIT the PHP closing tag and end with a single empty line instead.

[**File Naming**](#ihv636)[**¶**](#3dy6vkm)

Class files must be named in a Ucfirst-like manner, while any other file name (configurations, views, generic scripts, etc.) should be in all lowercase.

**INCORRECT**:

somelibrary.php  
someLibrary.php  
SOMELIBRARY.php  
Some\_Library.php  
  
Application\_config.php  
Application\_Config.php  
applicationConfig.php

**CORRECT**:

Somelibrary.php  
Some\_library.php  
  
applicationconfig.php  
application\_config.php

Furthermore, class file names should match the name of the class itself. For example, if you have a class named Myclass, then its filename must be **Myclass.php**.

[**Class and Method Naming**](#32hioqz)[**¶**](#1t3h5sf)

Class names should always start with an uppercase letter. Multiple words should be separated with an underscore, and not CamelCased.

**INCORRECT**:

class superclass  
class SuperClass

**CORRECT**:

class Super\_class

class Super\_class {  
  
 public function \_\_construct()  
 {  
  
 }  
}

Class methods should be entirely lowercased and named to clearly indicate their function, preferably including a verb. Try to avoid overly long and verbose names. Multiple words should be separated with an underscore.

**INCORRECT**:

function fileproperties() // not descriptive and needs underscore separator  
function fileProperties() // not descriptive and uses CamelCase  
function getfileproperties() // Better! But still missing underscore separator  
function getFileProperties() // uses CamelCase  
function get\_the\_file\_properties\_from\_the\_file() // wordy

**CORRECT**:

function get\_file\_properties() // descriptive, underscore separator, and all lowercase letters

[**Variable Names**](#1hmsyys)[**¶**](#4d34og8)

The guidelines for variable naming are very similar to those used for class methods. Variables should contain only lowercase letters, use underscore separators, and be reasonably named to indicate their purpose and contents. Very short, non-word variables should only be used as iterators in for() loops.

**INCORRECT**:

$j = 'foo'; // single letter variables should only be used in for() loops  
$Str // contains uppercase letters  
$bufferedText // uses CamelCasing, and could be shortened without losing semantic meaning  
$groupid // multiple words, needs underscore separator  
$name\_of\_last\_city\_used // too long

**CORRECT**:

for ($j = 0; $j < 10; $j++)  
$str  
$buffer  
$group\_id  
$last\_city

[**Commenting**](#41mghml)[**¶**](#2s8eyo1)

In general, code should be commented prolifically. It not only helps describe the flow and intent of the code for less experienced programmers, but can prove invaluable when returning to your own code months down the line. There is not a required format for comments, but the following are recommended.

[DocBlock](http://manual.phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tutorial_phpDocumentor.howto.pkg.html#basics.docblock) style comments preceding class, method, and property declarations so they can be picked up by IDEs:

/\*\*  
 \* Super Class  
 \*  
 \* @package Package Name  
 \* @subpackage Subpackage  
 \* @category Category  
 \* @author Author Name  
 \* @link http://example.com  
 \*/  
class Super\_class {

/\*\*  
 \* Encodes string for use in XML  
 \*  
 \* @param string $str Input string  
 \* @return string  
 \*/  
function xml\_encode($str)

/\*\*  
 \* Data for class manipulation  
 \*  
 \* @var array  
 \*/  
public $data = array();

Use single line comments within code, leaving a blank line between large comment blocks and code.

// break up the string by newlines  
$parts = explode("\n", $str);  
  
// A longer comment that needs to give greater detail on what is  
// occurring and why can use multiple single-line comments. Try to  
// keep the width reasonable, around 70 characters is the easiest to  
// read. Don't hesitate to link to permanent external resources  
// that may provide greater detail:  
//  
// http://example.com/information\_about\_something/in\_particular/  
  
$parts = $this->foo($parts);

[**Constants**](#2grqrue)[**¶**](#17dp8vu)

Constants follow the same guidelines as do variables, except constants should always be fully uppercase. *Always use CodeIgniter constants when appropriate, i.e. SLASH, LD, RD, PATH\_CACHE, etc.*

**INCORRECT**:

myConstant // missing underscore separator and not fully uppercase  
N // no single-letter constants  
S\_C\_VER // not descriptive  
$str = str\_replace('{foo}', 'bar', $str); // should use LD and RD constants

**CORRECT**:

MY\_CONSTANT  
NEWLINE  
SUPER\_CLASS\_VERSION  
$str = str\_replace(LD.'foo'.RD, 'bar', $str);

[**TRUE, FALSE, and NULL**](#vx1227)[**¶**](#3rdcrjn)

**TRUE**, **FALSE**, and **NULL** keywords should always be fully uppercase.

**INCORRECT**:

if ($foo == true)  
$bar = false;  
function foo($bar = null)

**CORRECT**:

if ($foo == TRUE)  
$bar = FALSE;  
function foo($bar = NULL)

[**Logical Operators**](#3fwokq0)[**¶**](#26in1rg)

Use of the || “or” comparison operator is discouraged, as its clarity on some output devices is low (looking like the number 11, for instance). && is preferred over AND but either are acceptable, and a space should always precede and follow !.

**INCORRECT**:

if ($foo || $bar)  
if ($foo AND $bar) // okay but not recommended for common syntax highlighting applications  
if (!$foo)  
if (! is\_array($foo))

**CORRECT**:

if ($foo OR $bar)  
if ($foo && $bar) // recommended  
if ( ! $foo)  
if ( ! is\_array($foo))

[**Comparing Return Values and Typecasting**](#1v1yuxt)[**¶**](#lnxbz9)

Some PHP functions return FALSE on failure, but may also have a valid return value of “” or 0, which would evaluate to FALSE in loose comparisons. Be explicit by comparing the variable type when using these return values in conditionals to ensure the return value is indeed what you expect, and not a value that has an equivalent loose-type evaluation.

Use the same stringency in returning and checking your own variables. Use **===** and **!==** as necessary.

**INCORRECT**:

// If 'foo' is at the beginning of the string, strpos will return a 0,  
// resulting in this conditional evaluating as TRUE  
if (strpos($str, 'foo') == FALSE)

**CORRECT**:

if (strpos($str, 'foo') === FALSE)

**INCORRECT**:

function build\_string($str = "")  
{  
 if ($str == "") // uh-oh! What if FALSE or the integer 0 is passed as an argument?  
 {  
  
 }  
}

**CORRECT**:

function build\_string($str = "")  
{  
 if ($str === "")  
 {  
  
 }  
}

See also information regarding [typecasting](http://php.net/manual/en/language.types.type-juggling.php#language.types.typecasting), which can be quite useful. Typecasting has a slightly different effect which may be desirable. When casting a variable as a string, for instance, NULL and boolean FALSE variables become empty strings, 0 (and other numbers) become strings of digits, and boolean TRUE becomes “1”:

$str = (string) $str; // cast $str as a string

[**Debugging Code**](#4f1mdlm)[**¶**](#35nkun2)

Do not leave debugging code in your submissions, even when commented out. Things such as var\_dump(), print\_r(), die()/exit() should not be included in your code unless it serves a specific purpose other than debugging.

[**Whitespace in Files**](#2u6wntf)[**¶**](#1ksv4uv)

No whitespace can precede the opening PHP tag or follow the closing PHP tag. Output is buffered, so whitespace in your files can cause output to begin before CodeIgniter outputs its content, leading to errors and an inability for CodeIgniter to send proper headers.

[**Compatibility**](#19c6y18)[**¶**](#44sinio)

CodeIgniter recommends PHP 5.6 or newer to be used, but it should be compatible with PHP 5.3.7. Your code must either be compatible with this requirement, provide a suitable fallback, or be an optional feature that dies quietly without affecting a user’s application.

Additionally, do not use PHP functions that require non-default libraries to be installed unless your code contains an alternative method when the function is not available.

[**One File per Class**](#3tbugp1)[**¶**](#2jxsxqh)

Use separate files for each class, unless the classes are *closely related*. An example of a CodeIgniter file that contains multiple classes is the Xmlrpc library file.

[**Whitespace**](#28h4qwu)[**¶**](#z337ya)

Use tabs for whitespace in your code, not spaces. This may seem like a small thing, but using tabs instead of whitespace allows the developer looking at your code to have indentation at levels that they prefer and customize in whatever application they use. And as a side benefit, it results in (slightly) more compact files, storing one tab character versus, say, four space characters.

[**Line Breaks**](#nmf14n)[**¶**](#3j2qqm3)

Files must be saved with Unix line breaks. This is more of an issue for developers who work in Windows, but in any case ensure that your text editor is setup to save files with Unix line breaks.

[**Code Indenting**](#37m2jsg)[**¶**](#1y810tw)

Use Allman style indenting. With the exception of Class declarations, braces are always placed on a line by themselves, and indented at the same level as the control statement that “owns” them.

**INCORRECT**:

function foo($bar) {  
 // ...  
}  
  
foreach ($arr as $key => $val) {  
 // ...  
}  
  
if ($foo == $bar) {  
 // ...  
} else {  
 // ...  
}  
  
for ($i = 0; $i < 10; $i++)  
 {  
 for ($j = 0; $j < 10; $j++)  
 {  
 // ...  
 }  
 }  
  
try {  
 // ...  
}  
catch() {  
 // ...  
}

**CORRECT**:

function foo($bar)  
{  
 // ...  
}  
  
foreach ($arr as $key => $val)  
{  
 // ...  
}  
  
if ($foo == $bar)  
{  
 // ...  
}  
else  
{  
 // ...  
}  
  
for ($i = 0; $i < 10; $i++)  
{  
 for ($j = 0; $j < 10; $j++)  
 {  
 // ...  
 }  
}  
  
try  
{  
 // ...  
}  
catch()  
{  
 // ...  
}

[**Bracket and Parenthetic Spacing**](#1mrcu09)[**¶**](#4i7ojhp)

In general, parenthesis and brackets should not use any additional spaces. The exception is that a space should always follow PHP control structures that accept arguments with parenthesis (declare, do-while, elseif, for, foreach, if, switch, while), to help distinguish them from functions and increase readability.

**INCORRECT**:

$arr[ $foo ] = 'foo';

**CORRECT**:

$arr[$foo] = 'foo'; // no spaces around array keys

**INCORRECT**:

function foo ( $bar )  
{  
  
}

**CORRECT**:

function foo($bar) // no spaces around parenthesis in function declarations  
{  
  
}

**INCORRECT**:

foreach( $query->result() as $row )

**CORRECT**:

foreach ($query->result() as $row) // single space following PHP control structures, but not in interior parenthesis

[**Localized Text**](#46r0co2)[**¶**](#2xcytpi)

CodeIgniter libraries should take advantage of corresponding language files whenever possible.

**INCORRECT**:

return "Invalid Selection";

**CORRECT**:

return $this->lang->line('invalid\_selection');

[**Private Methods and Variables**](#2lwamvv)[**¶**](#1ci93xb)

Methods and variables that are only accessed internally, such as utility and helper functions that your public methods use for code abstraction, should be prefixed with an underscore.

public function convert\_text()  
private function \_convert\_text()

[**PHP Errors**](#111kx3o)[**¶**](#3whwml4)

Code must run error free and not rely on warnings and notices to be hidden to meet this requirement. For instance, never access a variable that you did not set yourself (such as $\_POST array keys) without first checking to see that it isset().

Make sure that your dev environment has error reporting enabled for ALL users, and that display\_errors is enabled in the PHP environment. You can check this setting with:

if (ini\_get('display\_errors') == 1)  
{  
 exit "Enabled";  
}

On some servers where *display\_errors* is disabled, and you do not have the ability to change this in the php.ini, you can often enable it with:

ini\_set('display\_errors', 1);

Note

Setting the [display\_errors](http://php.net/manual/en/errorfunc.configuration.php#ini.display-errors) setting with ini\_set() at runtime is not identical to having it enabled in the PHP environment. Namely, it will not have any effect if the script has fatal errors.

[**Short Open Tags**](#3l18frh)[**¶**](#2bn6wsx)

Always use full PHP opening tags, in case a server does not have *short\_open\_tag* enabled.

**INCORRECT**:

<? echo $foo; ?>  
  
<?=$foo?>

**CORRECT**:

<?php echo $foo; ?>

Note

PHP 5.4 will always have the **<?=** tag available.

[**One Statement Per Line**](#206ipza)[**¶**](#qsh70q)

Never combine statements on one line.

**INCORRECT**:

$foo = 'this'; $bar = 'that'; $bat = str\_replace($foo, $bar, $bag);

**CORRECT**:

$foo = 'this';  
$bar = 'that';  
$bat = str\_replace($foo, $bar, $bag);

[**Strings**](#4k668n3)[**¶**](#3as4poj)

Always use single quoted strings unless you need variables parsed, and in cases where you do need variables parsed, use braces to prevent greedy token parsing. You may also use double-quoted strings if the string contains single quotes, so you do not have to use escape characters.

**INCORRECT**:

"My String" // no variable parsing, so no use for double quotes  
"My string $foo" // needs braces  
'SELECT foo FROM bar WHERE baz = \'bag\'' // ugly

**CORRECT**:

'My String'  
"My string {$foo}"  
"SELECT foo FROM bar WHERE baz = 'bag'"

[**SQL Queries**](#2zbgiuw)[**¶**](#1pxezwc)

SQL keywords are always capitalized: SELECT, INSERT, UPDATE, WHERE, AS, JOIN, ON, IN, etc.

Break up long queries into multiple lines for legibility, preferably breaking for each clause.

**INCORRECT**:

// keywords are lowercase and query is too long for  
// a single line (... indicates continuation of line)  
$query = $this->db->query("select foo, bar, baz, foofoo, foobar as raboof, foobaz from exp\_pre\_email\_addresses  
...where foo != 'oof' and baz != 'zab' order by foobaz limit 5, 100");

**CORRECT**:

$query = $this->db->query("SELECT foo, bar, baz, foofoo, foobar AS raboof, foobaz  
 FROM exp\_pre\_email\_addresses  
 WHERE foo != 'oof'  
 AND baz != 'zab'  
 ORDER BY foobaz  
 LIMIT 5, 100");

[**Default Function Arguments**](#1egqt2p)[**¶**](#49x2ik5)

Whenever appropriate, provide function argument defaults, which helps prevent PHP errors with mistaken calls and provides common fallback values which can save a few lines of code. Example:

function foo($bar = '', $baz = FALSE)

[Next](http://docs.google.com/libraries/index.html)   [Previous](http://docs.google.com/security.html)

© Copyright 2014 - 2016, British Columbia Institute of Technology. Last updated on Oct 28, 2016.

Built with [Sphinx](http://sphinx-doc.org/) using a [theme](https://github.com/snide/sphinx_rtd_theme) provided by [Read the Docs](https://readthedocs.org).