**Именование**

При именовании файлов, методов, переменных, классов и т.д. нужно стараться как можно более точно и лаконичнее выразить предназначение данного элемента.

**Имена файлов**

Все имена файлов должны иметь расширение .php и начинаться с заглавной буквы если это файл класса, каждое следующее слово тоже должно начинаться с заглавной буквы.

Если файл содержит константы, глобальные переменные или функции то следует добавить суффикс .inc :

сonfig.inc.php

Если файл содержит класс, то к названию следует добавлять суффикс .class :

Database.class.php

FileSystem.class.php

**Имена классов**

Имя класса должно начинаться с заглавной буквы, каждое следующее слово тоже должно начинаться с заглавной буквы, для разделения слов нельзя использовать знак "подчеркивания".

class UserInfo

{

...

}

Наличие имен классов, состоящих более чем из трех слов говорит о том, что скорее всего данный класс берет на себя больше задач, чем планировалось;

Не следует присваивать производному классу имя производное от имени родительского класса. Лучше будет, если у класса будет имя более соответствующее его предназначению;

**Имена методов и функций**

Имя метода или функции должно начинаться со строчной буквы, каждое следующее слово тоже должно начинаться с заглавной буквы, для разделения слов нельзя использовать знак "подчеркивания".

Как правило функция или метод совершают какое-либо действие, поэтому желательно, чтобы из имени было понятно, какое именно действие будет совершаться:

class SomeClass

{

function doIt() {};

function handleError() {};

}

Также не помешает использование префиксов:

* is - для обозначения вопроса.
* get - получить значение;
* set - установить значение.

Например:

isEmpty()

function getElement()

function setElement( $element )

Если в имени содержится аббревиатура, нужно вместо всех заглавных оставить только первую букву заглавной, а остальные написать строчными.

getXMLDocument() // неправильно

getXmlDocument() // правильно

**Имена переменных и членов класса**

Переменные именуются по следующим правилам:

* имя переменной или члена класса должно начинаться со строчной буквы
* каждое следующее слово должно начинаться с заглавной буквы
* для разделения слов нельзя использовать знак "подчеркивания"

$userInf

Для именования глобальных переменных следует использовать префикс g\_.

$g\_Log

**Имена констант**

Имена констант всегда должны быть в верхнем регистре с символами "подчеркивание" для разделения слов. Желательно, чтобы у имени константы был префикс, который бы выделял ее среди других и относил ее к какому-то перечислению или классу.

define( 'IMAGE\_TYPE\_JPEG', 1 );

define( 'IMAGE\_TYPE\_PNG', 2 );

define( 'IMAGE\_TYPE\_BMP', 3 )

define( 'DB\_USER', 'qweqwe' );

define( 'DB\_PASS', '123123' );

**Оформление кода**

**Отступы**

Следует использовать для отступа 4 пробела, а не табуляцию.

Использование пустых строк для отделения блоков кода улучшает его читаемость, но этим не стоит злоупотреблять.

**Комментарии**

Используйте строковые комментарии. Блочные комментарии лучше применять для документирования кода. Если используется система контроля версий, то неиспользуемый код комментировать не стоит – лучше его удалить из файла, так как это вносит путаницу.

Комментарии внутри кода классов должны соответствовать синтаксису комментариев PHPDoc, который напоминает Javadoc. За дополнительной информацией о PHPDoc обращайтесь сюда: [http://www.phpdoc.org/](http://www.phpdoc.org/" \o "http://www.phpdoc.org/)

Основное правило в данном случае - каждая часть кода повышенной сложности должна быть прокомментирована до того, как вы забыли как она работает.

/\*\*

\* fooFunction

\*

\* description foo function

\*

\* @access public

\* @param integer $arg1 integer data

\* @param string $arg2 some string

\* @return string

\*/

function fooFunction( $arg1, $arg2 = '' )

{

if ( $arg2 )

{

$val = $arg2;

}

else

{

$val = $arg1;

}

return $val;

}

**Пробелы**

Не надо забывать ставить пробелы перед и после оператора присвоения, а так же логических и арифметических операторов.

if ($a!=0 && $b\*$a>1.0) // неправильно, сложно понять

if ( $a != 0 && $b \* $a > 1.0 ) // пробелы расставлены правильно, но понять все равно сложно

if ( ( $a != 0 ) && ( ( $b \* $a ) > 1.0) ) // с точки зрения стиля, идеальн

$a="string"; // неправильно, сложно понять

$a = "string"; // правильно

**Расстановка фигурных скобок**

Открывающая скобка ставится под соответствующим оператором и на одном отступе с ним. Отступ закрывающей скобки должен быть таким же, что и у открывающей.

if ( ( condition0 ) || ( condition1 ) )

{

// some action

}

else if ( ( condition2 ) && ( condition3 ) )

{

// another action

}

else

{

// default action

}

В управляющих структурах между ключевым словом и открывающей круглой скобкой должен находиться пробел, чтобы отличать их от вызова функций.

Обязательно используйте фигурные скобки, даже в том случае, когда их использование не является необходимостью. Использование фигурных скобок увеличивает читабельность кода и уменьшает вероятность логических ошибок при изменении кода.

**Вызовы функций**

Вызовы функций должны быть написаны без отступов между именем функции и открывающей скобкой. Отступы в виде пробела должны присутствовать после каждой запятой в перечислении параметров.

$var = foo( $bar, $baz, $quux );

Как можно заметить, в примере используются пробелы с двух сторон от знака "=". Если подобные присвоения результатов функций переменным объединяются в блоки, то для повышения читабельности рекомендуется следующий синтаксис:

$short = foo( $bar );

$longVariable = foo( $baz );

**Конструкция if then else**

if ( condition0 ) // Комментарий

{

}

else if ( condition1 ) // Комментарий

{

}

else // Комментарий

{

}

При сравнении всегда ставьте константы слева. Например:

define( 'SOME\_CONSTANT', 7 );

...

if ( SOME\_CONSTANT == $errorNum ) ...

Первая причина такому поведению - это то, что парсер найдёт ошибку, если вы поставите только один знак равенства вместо двух. Вторая причина - при чтении кода вы находите нужное вам значение сразу в начале условия, а не ищите где-то в конце.

**Конструкция switch**

* При наличии соответствующего комментария допускаются блоки, передающие управление вниз;
* Рекомендуется всегда ставить блок default, который бы сообщал об ошибке в случаях, когда попадание на него должно быть исключено, но тем не менее имело место;

switch (...)

{

case 1:

...

// управление передается вниз

case 2:

$someVar = getNumber();

...

break;

default:

...

}

**Использование continue, break**

continue и break - это тот же самый goto, только названы по-другому, поэтому их использование рекомендуется свести до минимума.

Например, при использовании continue возникают следующие проблемы:

* continue может обойти условный блок;
* continue может обойти наращивание/уменьшение.

**Использование одинарных и двойных кавычек**

Двойные кавычки используются в том случае, если необходимо подчеркнуть тип **string**:

echo "Hello World!";

$arr = array("string1", "string2");

или когда это действительно необходимо:

$someVar = "World!";

echo "Hello $someVar";

В остальных случаях используются одинарные кавычки:

define( 'SOME\_CONST', 7 );

require\_once( 'includes/common.inc.php' );

$arr = array

(

'name' => "Alexey",

'email' => "email@cpslabs.net"

);

echo $arr['name'];