Стиль кодирования

Обрамление PHP-кода

PHP-код должен всегда обрамляться полными PHP-тегами:

```
1. <?php
2.
3. ?>
```

Короткие теги не допустимы. В файлах, содержащих только PHP-код, закрывающий тегдолжен быть опущен.

Строки

Строковые литералы

Когда строка является литеральной (не содержит подстановок переменных), для ее обрамления должны использоваться апострофы или "одинарные кавычки":

```
1. $a = 'Example String';
```

Строковые литералы, содержащие апострофы

Когда строка литералов сама содержит апострофы, разрешается для обрамления строки использовать "двойные кавычки". Это особенно актуально для **SQL**-запросов:

```
1. $sql = "SELECT `id`, `name` from `people` WHERE `name`='Fred' OR
   `name`='Susan'";
```

Синтаксис выше является более предпочтительным, чем экранирование апострофов.

Подстановка переменных

Подстановка переменных разрешается с использованием нижеприведенных форм:

```
1. $greeting = "Hello $name, welcome back!";
2. $greeting = "Hello {$name}, welcome back!";
```

Для надежности, эта форма не допустима:

```
1. $greeting = "Hello ${name}, welcome back!";
```

Конкатенация строк

Строки должны объединятся с помощью оператора ".". Пробел должен всегда добавляться до и после оператора "." для улучшения читабельности:

```
1. $company = 'SLMax' . ' ' . 'Technologies';
```

Когда производится конкатенация строк с помощью оператора ".", разрешается разрывать выражение на несколько строк для улучшения читабельности. В этом случае, каждая следующая строка должна быть дополнена пробелами так, чтобы оператор "." был выровнен под оператором "=":

```
1. $sql = "SELECT `id`, `name` FROM `people` "
2.       . "WHERE `name` = 'Susan' "
3.       . "ORDER BY `name` ASC ";
```

Массивы

Массивы с числовыми индексами

Отрицательные числа в качестве индексов запрещены.

Хотя индекс массива может начинаться с любого неотрицательного числа, но не приветствуется и не рекомендуется начинать индексирование не с 0.

Когда определяется индексированный массив с помощью конструкции Array , завершающий пробел должен быть добавлен после каждой запятой для улучшения читабельности:

```
1. $sampleArray = array(1, 2, 3, 'SLMax', 'Tech');
```

Также разрешается определять многострочные индексированные массивы, используя конструкцию "array". В этом случае, каждая следующая строка должна быть дополнена пробелами так, чтобы начало каждой строки было выравнено как показано ниже:

```
1. $sampleArray = array(1, 2, 3, 'SLMax', 'Tech',
2.                      $a, $b, $c,
3.                      56.44, $d, 500);
```

В качестве альтернативы, начальный элемент может располагаться на следующей строке. В этом случае он должен быть сдвинут на один уровень отступа больше, чем строка содержащая объявление массива. Все последующие строки должны иметь аналогичный отступ. Закрывающая скобка должна быть на отдельной строке с уровнем отступа, что и

строка, содержащая объявление массива:

```
$sampleArray = array(
    1, 2, 3, 'SLMax', 'Tech',
    $a, $b, $c,
    56.44, $d, 500,
);
```

При использовании последнего способа мы рекомендуем ставить запятую после последнего элемента массива. Это упрощает добавление новых строк и обеспечивает отсутствие ошибок из-за пропущенной запятой.

Ассоциативные массивы

Когда определяется ассоциативный массив с помощью конструкции Array , приветствуется разделение выражения на несколько строк. В этом случае, каждая следующая строка должна быть дополнена с помощью пробелов так, чтобы и ключи и значения были выровнены:

```
$sampleArray = array('firstKey'  => 'firstValue',
                     'secondKey' => 'secondValue');
```

В качестве альтернативы, начальный элемент может располагаться на следующей строке. В этом случае он должен быть сдвинут на один уровень отступа больше, чем строка содержащая объявление массива. Все последующие строки должны иметь аналогичный отступ. Закрывающая скобка должна быть на отдельной строке с уровнем отступа, что и строка, содержащая объявление массива. Для удобочитаемости, "=>" должен быть выравнен пробелами относительно остальных:

```
$sampleArray = array(
    'firstKey'  => 'firstValue',
    'secondKey' => 'secondValue',
);
```

При использовании последнего способа мы рекомендуем ставить запятую после последнего элемента массива. Это упрощает добавление новых строк и обеспечивает отсутствие ошибок из-за пропущенной запятой.

Классы

Определение класса

Классы должны быть именованы согласно своей логической сущности, основываясь на популярных наименованиях на английском языке (т.е не $polzovatel, а $user).
Фигурная скобка всегда пишется на следующей строке под именем класса.
Каждый класс должен иметь блок документации (doc-блок).
Код внутри класса должен иметь отступ в одну табуляцию.
Только один класс разрешен внутри одного PHP-файла.

Размещение дополнительно кода в файле с классом разрешено, но не приветствуется. В таких файлах, две пустые строки должны разделять класс и дополнительный PHP-код.

Это пример допустимого определения класса:

```
1. /**
2. Пример: Просто класс
3. Подробное описание класса, что он делает, как он делает, что, кому
   куда передает.
4. */
5. class Database
6. {
7.     // содержимое класса должно быть
8.     // с отступом в 1 табуляцию
9. }
```

Классы, расширяющие другие классы или реализующие интерфейсы, должны объявлять свои зависимости на той же строке, если возможно.

```
1. class SampleClass extends FooAbstract implements BarInterface
2. {
3. }
```

Если в результате такого объявления, длина строки превышает максимальную длину строки ЧИТАБЕЛЬНОСТИ, сделайте перенос перед ключевыми словами "extends" и/или "implements" и сделайте отступ в один уровень.

```
1. class SampleClass
2.     extends FooAbstract
3.     implements BarInterface
4. {
5. }
```

Если класс реализует несколько интерфейсов и объявление превышает максимальную длину строки - сделайте перенос после запятой, разделяющей интерфейсы, и выровняйте их имена пробелами:

```
1. class SampleClass
2.     implements BarInterface,
```

```
3.                 BazInterface
4. {
5. }
```

Переменные-члены классов

Переменные-члены классов должны быть именованы согласно своему логическому назначению на английском языке.

Любые переменные, определенные в классе, должны быть определены в начале класса, до определения любого метода.

Ключевое слово *var* не разрешено. Члены класса должны всегда определять их область видимости, используя ключевое слово **`private`**, **`protected`** или **`public`**.

Функции и методы

Определение функций и методов

Функции должны быть наименованы согласно их логическому действию на английском языке, все слова в названии должны быть прописаны без сокращений с заглавной буквы.
Функции внутри классов должны всегда определять свою область видимости с помощью одного из префиксов **`private`**, **`protected`** или **`public`**.
Как и у классов, фигурная скобка всегда пишется на следующей строке под именем функции. Пробелы между именем функции и круглой скобкой для аргументов не допускаются.
В случаях, когда в функции находится много кода (более 20 строк), необходимо писать doc-блок на строчку выше названия функции.
Функции в глобальной области видимости крайне не приветствуются.
Это пример допустимого определения функции:

```
1. /**
2. Пример: Просто класс
3. Подробное описание класса, что он делает, как он делает, что, кому
   куда передает.
4. */
5. class Foo
6. {
7.     public function bar()
8.     {
```

```
        // содержимое класса должно быть
        // с отступом в 1 табуляцию
    }
}
```

В случае, если список аргументов превышает максимальную длину строки, можно делать перенос строки. Аргументы, перенесенные на следующую строку, должны иметь отступ на один уровень больше чем у объявления метода или функции. Закрывающая скобка должна быть на новой строке с отступом, как у объявления функции/метода, а после нее, через пробел, должна находиться открывающая фигурная скобка. Ниже приведен пример такой ситуации:

```
/**
 * Doc-блок здесь
 */
class Foo
{
    public function bar($arg1, $arg2, $arg3,
        $arg4, $arg5, $arg6
    ) {
        // содержимое класса должно быть
        // с отступом в 1 табуляцию
    }
}
```

Замечание: Передача по ссылке допустима только в определениях функций:

```
/**
 * Doc-блок здесь
 */
class Foo
{
    public function bar(&$baz)
    {}
}
```

Передача по ссылке во время вызова строго запрещена.

Возвращаемое значение не должно обрамляться в круглые скобки, иначе это ухудшает читабельность, а также может нарушить код, если метод позже станет возвращать результат по ссылке.

```
/**
 * Doc-блок здесь
 */
```

```
class Foo
{
    /**
     * ПЛОХО
     */
    public function bar()
    {
        return($this->bar);
    }

    /**
     * ХОРОШО
     */
    public function bar()
    {
        return $this->bar;
    }
}
```

Использование функций и методов

Аргументы функции разделяются одним завершающим пробелом после каждой запятой. Это пример допустимого вызова функции для функции, которая принимает три аргумента:

```
threeArguments(1, 2, 3);
```

Передача по ссылке во время вызова запрещена. Смотрите секцию определения функций для правильного способа передачи аргументов функции по ссылке.

Для функций, чьи аргументы допускают массив, вызов функции может включать конструкцию "array" и может быть разделено на несколько строк для улучшения читабельности. В этом случае, применим стандарт описания массивов:

```
threeArguments(array(1, 2, 3), 2, 3);

threeArguments(array(1, 2, 3, 'SLMax', 'Tech',
                     $a, $b, $c,
                     56.44, $d, 500), 2, 3);

threeArguments(array(
    1, 2, 3, 'SLMax', 'Tech',
    $a, $b, $c,
    56.44, $d, 500
), 2, 3);
```

Управляющие структуры

If / Else / Elseif

Управляющие структуры, основанные на конструкциях *if* и *elseif*, должны иметь один пробел до открывающей круглой скобки условия, и один пробел после закрывающей круглой скобки.

Внутри выражения условия между круглыми скобками операторы должны разделяться пробелами для читабельности. Внутренние скобки приветствуются для улучшения логической группировки больших условий.

Открывающаяся фигурная скобка пишется на той же строке, что и условие.

Закрывающаяся фигурная скобка пишется на отдельной строке. Все содержимое между скобками пишется с отступом в 1 табуляцию.

```
1. if ($a != 2) {
2.     $a = 2;
3. }
```

Если условное выражение заставляет строку превысить максимальную длину строки и имеет несколько условий, вы можете разбить его на несколько строк. В таком случае, делайте перенос до логического оператора и выравнивайте пробелами до первого символа условного выражения. Закрывающая скобка должна быть на новой строке, с уровнем отступа как у управляющей структуры. На той же строке, через пробел, должна находиться открывающая фигурная скобка.

```
1. if (($a == $b)
2.     && ($b == $c)
3.     || (Foo::CONST == $d)
4. ) {
5.     $a = $d;
6. }
```

Цель последнего формата - предотвратить сложности, при добавлении или удалении условий из условного выражения в последующих изменениях.

Для выражения "if", включающего "elseif" или "else", форматирование должно быть таким, как в следующем примере:

```
1. if ($a != 2) {
2.     $a = 2;
3. } else {
```

```
    $a = 7;
}

if ($a != 2) {
    $a = 2;
} elseif ($a == 3) {
    $a = 4;
} else {
    $a = 7;
}

if (($a == $b)
    && ($b == $c)
    || (Foo::CONST == $d)
) {
    $a = $d;
} elseif (($a != $b)
          || ($b != $c)
) {
    $a = $c;
} else {
    $a = $b;
}
```

PHP допускает написание таких выражений без фигурных скобок при некоторых условиях. Стандарт кодирования не делает различий - для всех "if", "elseif" или "else" выражений необходимо использовать фигурные скобки.

Switch

Управляющие структуры, написанные с использованием "switch" конструкции, должны иметь один пробел до открывающей круглой скобки условного выражения, и также один пробел после закрывающей круглой скобки.
Все содержимое между фигурными скобками пишется с отступом в четыре пробела. Содержимое каждого "case" выражения должно писаться с отступом в 1 дополнительную табуляцию.

```
switch ($numPeople) {
    case 1:
        break;
```

```
4.
5.      case 2:
6.          break;
7.
8.      default:
9.          break;
10.}
```

Ключевое слово **default** никогда не должно опускаться в выражении **switch**.

Замечание: Иногда полезно писать **case** выражения, которые передают управление следующему **case** выражению, опуская **break** или **return**. Для того, чтобы отличать такие случаи от ошибок, каждое **case** выражение, где опущен **break** или **return**, должно содержать комментарий, указывающий, что это сделано преднамеренно.

Файлы

Каждый файл, содержащий PHP-код должен иметь заголовочный doc-блок в начале файла, содержащий как минимум следующую информацию:

```
1. /**
2. * Автор: Виктор Козинин
3. *
4. * Дата реализации: 01.01.1990 13:00
5. *
6. * Дата изменения: 02.01.1990 15:40
7. *
8. * Утилита для работы с базой данных
```

Классы

Каждый класс должен иметь doc-блок, содержащий как минимум следующую информацию:

```
1. /**
2. Просто класс
3. Подробное описание класса, что он делает, как он делает, что, кому
   куда передает.
4. */
```

Функции

Каждая функция, состоящая из более чем 20 строк, включая методы объектов, должна иметь doc-блок, содержащий как минимум:

- Описание функции
- Все аргументы
- Все возможные возвращаемые значения

Задачи

При коммите класса/файла который не был завершен - обязательно указывайте в начале класса/файла какие функции ещё предстоит сделать (TODO). Где +/* - логически соотносятся с файлом ‘Работа с репозиториями’.

```
1. /**
2.  * TODO:
3.  * [+] Создать функцию авторизацию пользователя
4.  * [*] Исправить проверку ключа доступа.
5.  */
```