Что такое PHP?

PHP - это серверный язык сценариев. это используется для разработки статических веб-сайтов или динамических веб-сайтов или веб-приложений. PHP расшифровывается как Hypertext Pre-процессор, который ранее обозначал персональные домашние страницы.

Сценарии PHP могут интерпретироваться только на сервере, на котором установлен PHP.

Клиентским компьютерам, получающим доступ к сценариям PHP, требуется только веб-браузер.

Файл PHP содержит теги PHP и заканчивается расширением «.php».

Что такое язык сценариев?

Сценарий - это набор инструкций по программированию, который интерпретируется во время выполнения.

Язык сценариев - это язык, который интерпретирует сценарии во время выполнения. Скрипты обычно встраиваются в другие программные среды.

Целью сценариев обычно является повышение производительности или выполнение рутинных задач для приложения.

Серверные сценарии интерпретируются на сервере, в то время как клиентские сценарии интерпретируются клиентским приложением.

PHP - это серверный скрипт, который интерпретируется на сервере, а JavaScript - пример клиентского скрипта, который интерпретируется клиентским браузером. И PHP, и JavaScript могут быть встроены в HTML-страницы.

Язык программирования и язык сценариев

Язык программирования	**Язык сценариев**
Имеет все функции, необходимые для разработки законченных приложений.	В основном используется для рутинных задач
Код должен быть скомпилирован, прежде чем он может быть выполнен	Код обычно выполняется без компиляции
Не нужно встраивать в другие языки	Обычно встраивается в другие программные среды.

Что означает PHP?

PHP означает - **персональная домашняя страница** , но теперь она обозначает рекурсивный обратный PHP: препроцессор гипертекста.

Код PHP может быть встроен в код HTML или может использоваться в сочетании с различными системами веб-шаблонов, системой управления веб-контентом и веб-платформами.

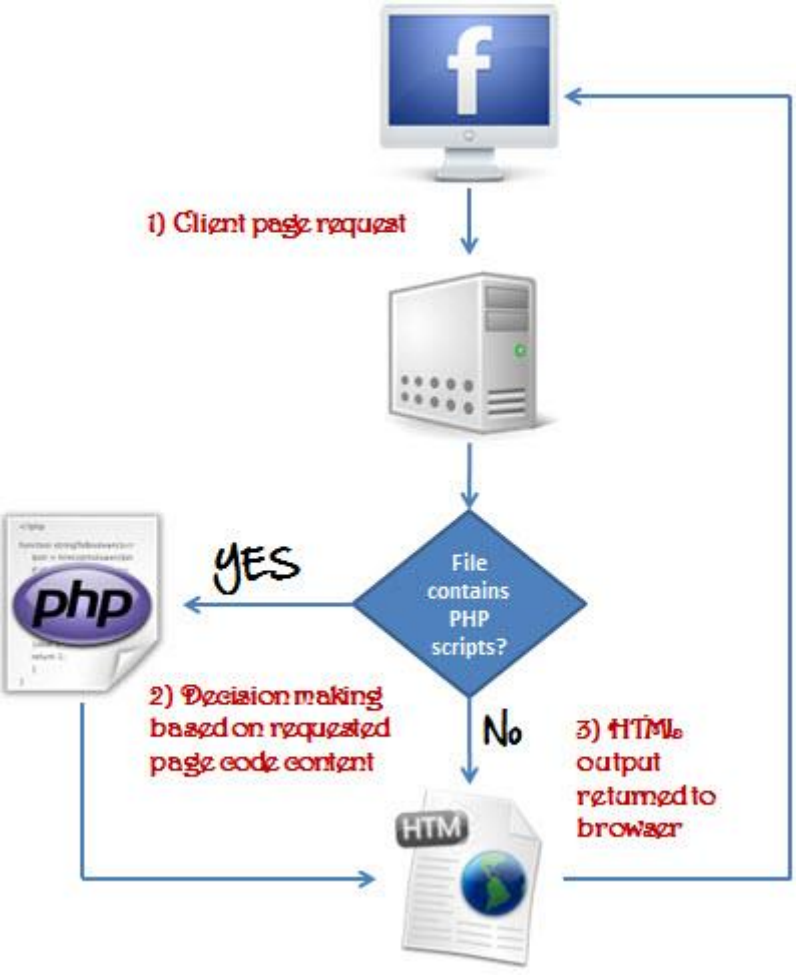
Синтаксис Php

```
<?php
    echo 'Hello World';
?>
```

Файл PHP также может содержать теги, такие как HTML, и сценарии на стороне клиента, такие как JavaScript.

- **HTML является дополнительным преимуществом** при изучении языка PHP. Вы даже можете изучать PHP, не зная HTML, но рекомендуется, по крайней мере, знать основы HTML.
- **Системы управления базами данных** СУБД для приложений на базе данных.
- Для более сложных тем, таких как интерактивные приложения и веб-службы, вам понадобятся **JavaScript и XML** .

Блок-схема, показанная ниже, иллюстрирует базовую архитектуру веб-приложения PHP и то, как сервер обрабатывает запросы.



Зачем использовать PHP?

Вы, очевидно, слышали о множестве языков программирования; Вы можете быть удивлены, почему мы хотели бы использовать PHP как наш яд для веб-программирования. Ниже приведены некоторые убедительные причины.

- PHP с **открытым исходным кодом и бесплатно.**
- Короткая кривая обучения по сравнению с другими языками, такими как JSP, ASP и т. Д.
- Большой документ сообщества
- Большинство серверов веб-хостинга поддерживают PHP по умолчанию, в отличие от других языков, таких как ASP, для которых требуется IIS. Это делает PHP экономически эффективным выбором.
- PHP регулярно обновляется, чтобы быть в курсе последних технологических тенденций.

- Другое преимущество, которое вы получаете с PHP - это **язык сценариев на стороне сервера** ; это означает, что вам нужно только установить его на сервер, и клиентским компьютерам, запрашивающим ресурсы у сервера, не нужно устанавливать PHP; только веб-браузер будет достаточно.
- PHP имеет **встроенную поддержку для работы рука об руку с MySQL** ; это не значит, что вы не можете использовать PHP с другими системами управления базами данных. Вы все еще можете использовать PHP с
    - Postgres
    - Oracle
    - MS SQL Server
    - ODBC etc.

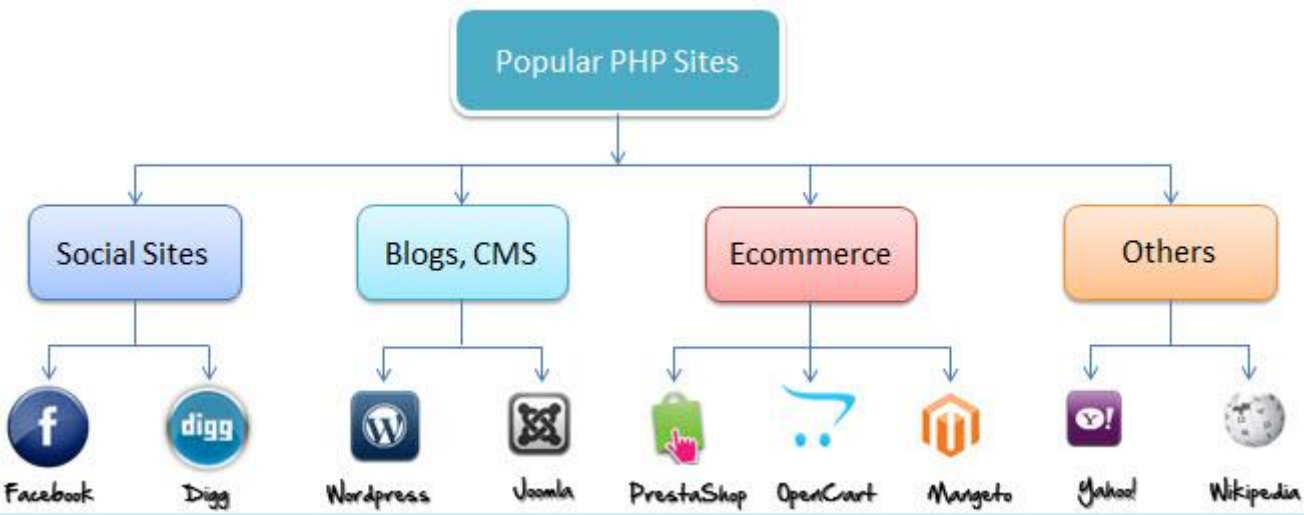
- PHP **кроссплатформенный;** это означает, что вы можете развернуть свое приложение в различных операционных системах, таких как Windows, Linux, Mac OS и т. д.

Для чего нужен PHP и доля рынка

Что касается доли рынка, в Интернете существует более 20 миллионов веб-сайтов и приложений, разработанных с использованием языка сценариев PHP.

Это может быть связано с вопросами, поднятыми выше;

Диаграмма ниже показывает некоторые из популярных сайтов, которые используют PHP



PHP против Asp.Net VS JSP VS CFML

ASP - Active Server Pages, JSP - Java Server Pages, CFML - язык разметки Cold Fusion В таблице ниже сравниваются различные языки сценариев на стороне сервера с PHP

ХАРАКТЕРНАЯ ЧЕРТА	**PHP**	**ASP**	**JSP**	**CFML**
Кривая обучения	короткая	Дольше чем PHP	Дольше чем PHP	Дольше чем PHP
веб хостинг	Поддерживается практически всеми хостинг-серверами	Требуется выделенный сервер	Довольно поддерживается	Требуется выделенный сервер
Открытый источник	да	нет	да	Как коммерческий, так и с открытым исходным кодом
Поддержка веб-сервисов	Встроенный	Использует .NET Framework	Использует добавить на библиотеки	Встроенный
Интеграция с HTML	Легко	Довольно сложный	Довольно сложный	Легко
Поддержка MySQL	Родной	Требуются сторонние драйверы	Требуются сторонние драйверы	Текущая версия имеет встроенную поддержку. Старые версии используют ODBC
Легко расширяется другими языками	да	нет	Расширен с использованием классов и библиотек Java.	да

Расширения файлов PHP

Расширение файла и теги для того , чтобы **сервер** для **идентификации** наших **PHP файлов** и **сценариев** , мы должны **сохранить** в **файл** с **расширением «.php»** . Старые расширения PHP включают

- .phtml
- .php3
- .php4
- .php5
- .phps

PHP был разработан для работы с HTML, и поэтому он может быть встроен в код HTML.

Вы можете создавать файлы PHP без каких-либо тегов HTML, и это называется Pure PHP file.

Сервер интерпретирует код PHP и выводит результаты в виде кода HTML в веб-браузеры.

Чтобы сервер мог идентифицировать код PHP из кода HTML, мы всегда должны заключать код PHP в теги PHP.

Тег PHP начинается с символа «меньше», за которым следует знак вопроса, а затем слова «php».

PHP является регистрозависимым языком, «VAR» - это не то же самое, что «var».

Сами теги PHP не чувствительны к регистру, но мы настоятельно рекомендуем использовать строчные буквы. Код ниже иллюстрирует вышеуказанный момент.

<? php…?>

Мы будем ссылаться на строки кода PHP как на операторы. PHP-операторы заканчиваются точкой с запятой (;). Если у вас есть только одно утверждение, вы можете опустить точку с запятой. Если у вас есть более одного оператора, то вы должны заканчивать каждую строку точкой с запятой. Для согласованности рекомендуется всегда заканчивать свои утверждения точкой с запятой. PHP-скрипты выполняются на сервере. Вывод возвращается в виде HTML.

PHP Привет, мир

Показанная ниже программа представляет собой простое PHP-приложение, которое выводит слова «Hello World!» При просмотре в веб-браузере.

```
<? PHP

echo "Hello world"

?>
```

Выход:

Hello world

Резюме

- PHP выступает за препроцессор гипертекста
- PHP - это серверный язык сценариев. Это означает, что он выполняется на сервере. Клиентские приложения не должны иметь установленный PHP.
- Файлы PHP сохраняются с расширением «.php», а код разработки PHP заключен в теги.
- PHP с открытым исходным кодом и кросс-платформенный

Типы данных PHP, переменные, константы, руководство по операторам

Типы данных PHP

Тип данных - это классификация данных в категории в соответствии с их атрибутами;

- Буквенно-цифровые символы классифицируются как строки
- Целые числа классифицируются как целые числа
- Числа с десятичными точками классифицируются как числа с плавающей запятой.
- Истинные или ложные значения классифицируются как логические.

PHP является свободно типизированным языком; он не имеет явно определенных типов данных. PHP определяет типы данных путем анализа атрибутов предоставленных данных. PHP неявно поддерживает следующие типы данных

- Целое число - целые числа, например, -3, 0, 69. Максимальное значение целого зависит от платформы. На 32-битной машине это обычно около 2 миллиардов. 64-битные машины обычно имеют большие значения. Константа PHP_INT_MAX используется для определения максимального значения.

```
<? PHP

echo PHP_INT_MAX;

?>
```

Выход:

```
9223372036854775807
```

- Число с плавающей запятой - десятичные числа, например, 3.14. они также известны как двойные или действительные числа. Максимальное значение с плавающей точкой зависит от платформы. Числа с плавающей точкой больше целых.
- Строка символов - например, Hello World
- Boolean - например, True или false.

Прежде чем мы углубимся в обсуждение типов данных PHP, давайте сначала обсудим переменные.

Переменная PHP

Переменная - это имя, данное области памяти, в которой хранятся данные во время выполнения.

Область действия переменной определяет ее видимость.

Глобальная переменная Php доступна для всех сценариев в приложении.

Локальная переменная доступна только для сценария, в котором она была определена.

Думайте о переменной как о стакане, содержащем воду. Вы можете добавить воду в стакан, выпить все, наполнить снова и т. Д.

То же самое относится и к переменным. Переменные используются для хранения данных и предоставления сохраненных данных при необходимости. Как и в других языках программирования, PHP также поддерживает переменные. Давайте теперь посмотрим на правила, которым следовали при создании переменных в PHP.

- Все имена переменных должны начинаться со знака доллара, например
- $my_var
- Имена переменных чувствительны к регистру; это означает, что $ my_var отличается от $ MY_VAR
- $my_var ≠ $MY_VAR
- Все имена переменных должны начинаться с буквы, следующей за другими символами, например, $ my_var1. $ 1my_var не является допустимым именем переменной.
- ✔ $my_var1; ✘ ~~$1my_var;~~
- Имена переменных не должны содержать пробелов, «$ first name» не является допустимым именем переменной. Вместо этого вы можете использовать подчеркивание вместо пробела, например, $ first_name. Вы не можете использовать такие символы, как знак доллара или минус, для разделения имен переменных.
- ✔ $my_var; ✘ $my var

Давайте теперь посмотрим, как PHP определяет тип данных в зависимости от атрибутов предоставленных данных.

```
<? PHP

$ my_var = 1;

echo $ my_var;

?>
```

Выход:

```
1
```

Числа с плавающей точкой

```
<? PHP

$ my_var = 3,14;

echo $ my_var;

?>
```

Выход:

```
3,14
```

Строки символов

```
<? PHP

$ my_var = "Hypertext Pre Processor ";

echo $ my_var;

?>
```

Выход:

```
Hypertext Pre Processor
```

Использование переменных

Переменные помогают отделить данные от алгоритмов программы.

Один и тот же алгоритм может использоваться для разных значений входных данных.

Например, предположим, что вы разрабатываете программу калькулятора, которая складывает два числа, вы можете создать две переменные, которые принимают числа, а затем использовать имена переменных в выражении, которое выполняет сложение.

Приведение типов

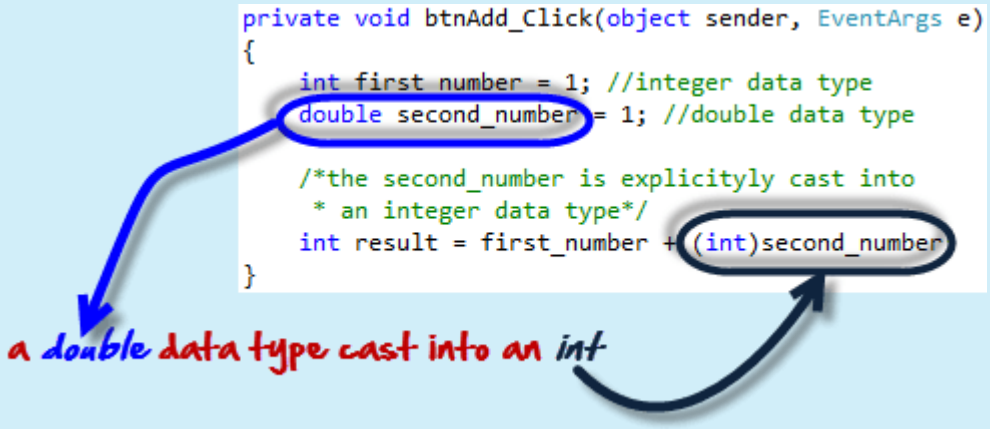
Выполнение арифметических вычислений с использованием переменных на языке, таком как C #, требует, чтобы переменные были одного типа данных.

Приведение типов - это преобразование переменной или значения в желаемый тип данных.

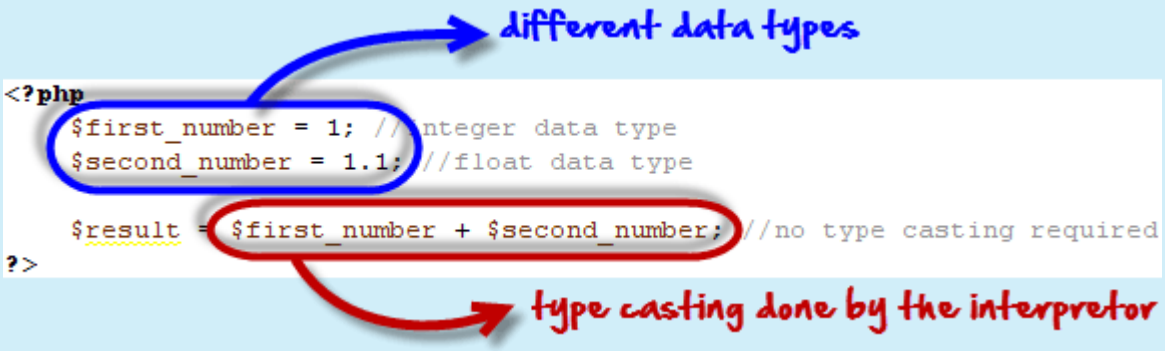
Это очень полезно при выполнении арифметических вычислений, которые требуют, чтобы переменные были одного типа данных.

Приведение типов в PHP выполняется интерпретатором.

В других языках, таких как C #, вы должны приводить переменные. Код ниже показывает приведение типов в C #.



На диаграмме ниже показан PHP, реализующий приведенный выше пример.



PHP также позволяет вам приводить тип данных. Это известно как явное приведение. Код ниже демонстрирует явное приведение типов.

```
<? PHP

$ a = 1;

$ b = 1,5;

$ c = $ a + $ b;

$ c = $ a + (int) $ b;

echo $ c;

?>
```

Выход:

```
2
```

Выше кода Выход 2 Функция var_dump используется для определения типа данных.

Код ниже демонстрирует, как использовать функцию var_dump.

```
<? PHP

$a = 1;

var_dump ($a);

$b = 1,5;

var_dump ($b);

$c = " I Love PHP ";

var_dump ($ c);

$d = true;

var_dump ($d);

?>
```

Выход:

```
int (1) float (1.5) string (10) " I Love PHP " bool (true)
```

PHP константа

Определить константу - константа - это переменная, значение которой нельзя изменить во время выполнения.

Предположим, что мы разрабатываем программу, которая использует значение PI 3.14, мы можем использовать константу для хранения ее значения.

Давайте теперь посмотрим на пример, который определяет константу. DEFINE ('PI', 3,14); // создаем константу со значением 3.14. Как только вы определили PI как 3.14, написание кода, подобного приведенному ниже, приведет к ошибке PI = 4; // PI был определен как константа, поэтому присвоение значения недопустимо.

Операторы PHP

Арифметические операторы

Арифметические операторы используются для выполнения арифметических операций с числовыми данными. Оператор конкатенации также работает со строковыми значениями. PHP поддерживает следующие операторы.

оператор	название	Описание	пример	Выход
+	прибавление	Суммирование x и y	1 + 1;	2
-	Вычитание	Разница между x и y	1 - 1;	0
*	умножение	Умножает x и y	3 * 7;	21
/	разделение	Отношение x и y	45/5;	9
%	Php Модуль	Дает напоминание о дайвинге x и y	10% 3;	1

оператор	название	Описание	пример	Выход
-n	Отрицание	Превращает n в отрицательное число	- (- 5);	5
X.Y	конкатенация	Соединяет x и y	"PHP" . " ROCKS";10 . 3;	PHP ROCKS103

Операторы присваивания

Операторы присваивания используются для присвоения значений переменным. Их также можно использовать вместе с арифметическими операторами.

оператор	название	Описание	пример	Выход
x =?	назначение	Назначает значение x?	$ x = 5;	5
x + =?	прибавление	Увеличивает значение x на?	$ x = 2; $ x + = 1;	3
X - =?	вычитание	Вычитает? от значения x	$ x = 3; $ x - = 2;	1
X * =?	умножение	Умножает значение x? раз	$ x = 0; $ x * = 9;	0
X / =?	деление	Коэффициент x и?	$ x = 6; $ x / = 3;	2
X% =?	модуль	Деление по модулю x на?	$ x = 3; $ x% = 2;	1
X. =?	конкатенация	Собирает предметы	"$ x = 'Pretty'; $ x. = 'Cool!';"	Pretty Cool!

Операторы сравнения Операторы сравнения используются для сравнения значений и типов данных.

оператор	название	Описание	пример	Выход
X == y	равных	Сравнивает x и y, затем возвращает true, если они равны	1 == "1";	True или 1
X === y	идентичный	Сравнивает как значения, так и типы данных.	1 === "1";	False или 0. Так как 1 - целое число, а «1» - строка
X! = Y, x <> y	PHP не равно	Сравнивает значения x и y. возвращает true, если значения не равны	2! = 1;	True или 1
X> y	Больше чем	Сравнивает значения x и y. возвращает true, если x больше y	3> 1;	True или 1
X <y	Меньше, чем	Сравнивает значения x и y. возвращает true, если x меньше y	2 <1;	False или 0
X> = y	Больше или равно	Сравнивает значения x и y. возвращает true, если x больше или равно y	1> = 1	True или 1
X <= y	Меньше или равно	Сравнивает значения x и y. возвращает true, если x больше или равно y	8 <= 6	False или 0

Логические операторы При работе с логическими операторами любое число больше или меньше нуля (0) оценивается как истинное. Ноль (0) оценивается как ложное.

оператор	название	Описание	пример	Выход
X and y, x && y	И	Возвращает true, если оба x и y равны	1 and 4; True && False;	True или 1 False или 0
X or y, x || Y	Или	Возвращает true, если x или y верны	6 or 9; 0 || 0;	True или 1 False или 0
X xor Y	Исключающая или, XOR	Возвращает true, если только x истинно или только y истинно	1 xor 1; 1 xor 0;	False или 0, True или 1
!X	Отрицание	Возвращает истину, если x ложь, и ложь, если x истина	! 0;	True или 1

Резюме

- PHP является свободно типизированным языком.
- Переменные - это области памяти, используемые для хранения данных
- Значение констант не может быть изменено во время выполнения
- Приведение типов используется для преобразования значения или переменной в желаемый тип данных.
- Арифметические операторы используются для манипулирования числовыми данными
- Операторы присваивания используются для назначения данных переменным
- Операторы сравнения используются для сравнения переменных или значений
- Логические операторы используются для сравнения условий или значений

Зачем использовать комментарии?

- Если вы некоторое время не работаете с исходным кодом, легко забыть, что делает этот код. Комментирование исходного кода помогает вспомнить, что делает код.
- Комментирование исходного кода также очень важно, когда нескольким разработчикам приходится работать над одним проектом. Изменения, сделанные одним разработчиком, могут быть легко поняты другими разработчиками, просто прочитав комментарии.
- В качестве лучшей практики вы должны иметь 3 строки комментариев для каждых 10 строк кода.

Комментарии PHP

- Комментарии помогают нам понять код
- Комментарии - это объяснения, которые мы включаем в наш исходный код. Эти комментарии для человеческого понимания.
- Однострочные комментарии начинаются с двойной косой черты // и заканчиваются на одной строке.
- ```
  //
  ```
- Многострочные комментарии начинаются с косой черты, за которой следует звездочка / *, и заканчиваются звездочкой, за которой следует косая черта * /.
- ```
  /*this is a multiple-line
  *comment
  /*example
  ```

На рисунке ниже показан файл PHP с многострочными и однострочными комментариями Пример PHP

```php
<?php

/**
 * Computer Value added tax
 *
 * @access public
 * @param float $amount, float $tax_rate
 * @return float $tax_amount
 */
function compute_tax($amount, $tax_rate) {
    $tax_amount = 0; //computed tax amount variable

    $tax_amount = $amount * ($tax_rate / 100); //tax computation

    return $tax_amount; //output tax amount as the function value
}
?>
```

Multi-line comments

Single line comment

Что такое массив PHP?

Массив PHP - это переменная, которая хранит более одной части связанных данных в одной переменной.

Думайте о массиве как о коробке конфет с щелями внутри.

Блок представляет собой сам массив, а пробелы, содержащие шоколадные конфеты, представляют значения, хранящиеся в массивах.

Диаграмма ниже иллюстрирует приведенный выше синтаксис.

Числовые массивы

Числовые массивы используют число в качестве ключей доступа.

Ключ доступа - это ссылка на слот памяти в переменной массива.

Ключ доступа используется всякий раз, когда мы хотим прочитать или присвоить новое значение элементу массива.

Ниже приведен синтаксис для создания числового массива в php. Пример массива

```
<?php

$variable_name[n] = value;

?>

Или

<?php

$variable_name = array(n => value, …);

?>
```

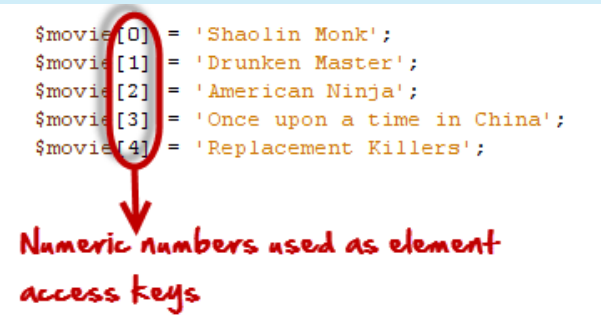
- “$variable_name…” - это имя переменной
- «[N]» - индекс доступа к элементу
- «value» - это значение, присвоенное элементу массива.

Давайте теперь посмотрим на пример числового массива.

Предположим, у нас есть 5 фильмов, которые мы хотим сохранить в переменных массива.

Мы можем использовать приведенный ниже пример, чтобы сделать это.

```
<? PHP

$movie[0] = 'Shaolin Monk';

$movie[1] = 'Drunken Master';

$movie[2] = 'American Ninja';

$movie[3] = 'Once upon a time in China';

$movie[4] = 'Replacement Killers';?>
```



Каждому фильму присваивается порядковый номер, который используется для извлечения или изменения его значения. Соблюдайте следующий код:

```
<? PHP

$movie[0]="Shaolin Monk";

$movie[1]="Drunken Master";

$movie[2]="American Ninja";

$movie[3]="Once upon a time in China";

$movie[4]="Replacement Killers";

echo $movie[3];

$movie[3] = " Eastern Condors";

echo $movie[3];?>
```

Выход:

```
Once upon a time in China Eastern Condors
```

Как видно из приведенных выше примеров, работа с массивами в PHP при работе с несколькими значениями одинаковой природы очень проста и гибка.

В качестве альтернативы вышеупомянутые переменные массива также могут быть созданы с использованием следующего кода.

```
<? PHP

$movie = array(0 => "Shaolin Monk",

        1 => "Drunken Master",

        2 => "American Ninja",

        3 => "Once upon a time in China",

        4 =>"Replacement Killers" );

echo $movie[4];?>
```

Выход:

```
Replacement Killers
```

PHP Ассоциативный Массив

Ассоциативный массив отличается от числового массива в том смысле, что ассоциативные массивы используют описательные имена для ключей идентификаторов.

Ниже приведен синтаксис для создания ассоциативного массива в php.

```
<? PHP

$variable_name['key_name'] = value;

$variable_name = array('keyname' => value);?>
```

ВОТ,

- «$ Variable_name…» - это имя переменной
- «['Key_name']» - индекс доступа к элементу
- «value» - это значение, присвоенное элементу массива.

Давайте предположим, что у нас есть группа людей, и мы хотим назначить пол каждого человека по имени.

Для этого мы можем использовать ассоциативный массив. Приведенный ниже код помогает нам в этом.

```
<? PHP

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

print_r($persons);

echo "";

echo "Mary is a " . $persons["Mary"];?>
```

ВОТ,

Выход:

```
Array ( [Mary] => Female [John] => Male [Mirriam] => Female ) Mary is a Female
```

Ассоциативный массив также очень полезен при извлечении данных из базы данных.

Имена полей используются в качестве ключей идентификатора.

PHP Многомерные массивы

Это массивы, которые содержат другие вложенные массивы.

Преимущество многомерных массивов состоит в том, что они позволяют группировать связанные данные вместе.

Давайте теперь посмотрим на практический пример, который реализует многомерный массив php.

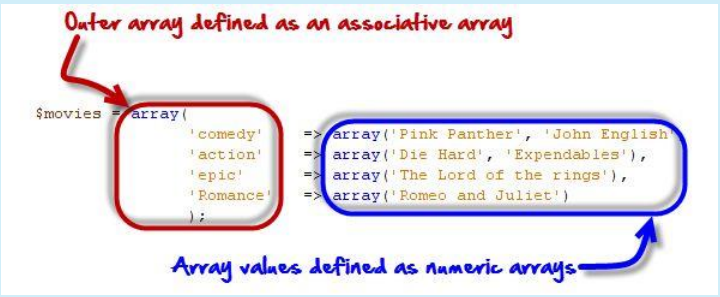
В таблице ниже приведен список фильмов по категориям.

Название фильма	**категория**
Pink Panther	Comedy
John English	Comedy
Die Hard	Action
Expendables	Action
The Lord of the rings	Epic
Romeo and Juliet	Romance
See no evil hear no evil	Comedy

Приведенная выше информация может быть представлена в виде многомерного массива.

Код ниже показывает реализацию.

```
<? PHP

$movies =array(

"comedy" => array("Pink Panther", "John English", "See no evil hear no evil"),

"action" => array("Die Hard", "Expendables"),

"epic" => array("The Lord of the rings"),

"Romance" => array("Romeo and Juliet")

);

print_r($movies);?>
```



Выход:

```
Array ( [comedy] => Array ( [0] => Pink Panther [1] => John English [2] => See no evil hear no evil ) [action] => Array ( [0] => Die Hard [1] => Expendables ) [epic] => Array ( [0] => The Lord of the rings ) [Romance] => Array ( [0] => Romeo and Juliet ) )
```

Другой способ определить тот же массив заключается в следующем

```
<?php

$film=array(

        "comedy" => array(

                0 => "Pink Panther",

                1 => "john English",

                2 => "See no evil hear no evil"

                ),

        "action" => array (

                0 => "Die Hard",

                1 => "Expendables"

                ),

        "epic" => array (

                0 => "The Lord of the rings"

                ),

        "Romance" => array

                (

                0 => "Romeo and Juliet"

                )

);

echo $film["comedy"][0];

?>
```

Выход:

```
Pink Panther
```

Примечание: числовой массив фильмов был вложен в ассоциативный массив категорий

Массивы PHP: операторы

оператор	название	Описание	Как это сделать	Выход

x + y	Union	Объединяет элементы из обоих массивов	<? PHP
			$ x = array ('id' => 1);
			$ y = array ('value' => 10);
			$ z = $ x + $ y;
			?>
				Array ([id] => 1
				[value] => 10)

X == y	Equal	Сравнивает два массива, если они равны, и возвращает true, если да.	<? PHP
			$ x = array ("id" => 1);
			$ y = array ("id" => "1");
			if ($ x == $ y) {
			echo "true"; }
			else { echo "false"; }
			?>
				Правда или 1

X === y	Identical	Сравнивает значения и типы данных	<? PHP
			$ x = array ("id" => 1);
			$ y = array ("id" => "1");
			if ($ x === $ y) {
			echo "true"; }
			else {
			echo "false"; }
			?>
				Ложь или 0

X! = Y, x <> y	Not equal		<? PHP
			$ x = array ("id" => 1);
			$ y = array ("id" => "1");
			if ($ x! = $ y) {
			эхо "правда"; }
			else
			{
			эхо "ложь";
			}
			?>
				Ложь или 0

оператор	название	Описание	Как это сделать	Выход

X! == y	Non identical		<? PHP
			$ x = array ("id" => 1);
			$ y = array ("id" => "1");
			if ($ x! == $ y)
			{ echo "true"; }
			else { echo "false"; }
			?>
				Правда или 1

PHP Array Функции

Функция подсчета

Функция count используется для подсчета количества элементов в массиве php. Код ниже показывает реализацию.

```
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo count($lecturers);
?>
```

Выход:

```
3
```

функция is_array

Функция is_array используется для определения, является ли переменная массивом или нет. Давайте теперь посмотрим на пример, который реализует функции is_array.

```
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo is_array($lecturers);
?>
```

Выход:

```
1
```

Сортировать

Эта функция используется для сортировки массивов по значениям.

Если значения буквенно-цифровые, они сортируются в алфавитном порядке.

Если значения являются числовыми, они сортируются в порядке возрастания.

Он удаляет существующие ключи доступа и добавляет новые числовые ключи.

Выход этой функции представляет собой числовой массив

```
<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

sort($persons);

print_r($persons);

?>
```

Выход:

```
Array ( [0] => Female [1] => Female [2] => Male )
```

ksort

Эта функция используется для сортировки массива по ключу. Следующий пример иллюстрирует его использование.

```
<? PHP

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

ksort($persons);

print_r($persons);?>
```

Выход:

```
Array ( [John] => Male [Mary] => Female [Mirriam] => Female )
```

asort

Эта функция используется для сортировки массива по значениям. Следующий пример иллюстрирует его использование.

```
<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

asort($persons);

print_r($persons);

?>
```

Выход:

```
Array ( [Mary] => Female [Mirriam] => Female [John] => Male )
```

Зачем использовать массивы?

- Содержимое массивов можно растянуть,
- Массивы легко помогают группировать связанную информацию, такую как данные для входа на сервер, вместе
- Массивы помогают писать более чистый код.

Резюме

- Массивы - это специальные переменные с возможностью хранения нескольких значений.
- Массивы являются гибкими и могут легко растягиваться, чтобы вместить больше значений.
- Числовые массивы используют числа для ключей массива
- PHP Ассоциативный массив использует описательные имена для ключей массива
- Многомерные массивы содержат внутри себя другие массивы.
- Функция count используется для получения количества элементов, которые были сохранены в массиве
- Функция is_array используется для определения, является ли переменная допустимым массивом или нет.
- Другие функции массива включают сортировку, ксорт, сортировку и т. Д.

Управляющие конструкции PHP: If else, Switch Case

Что такое Управляющие конструкции?

Выполнение кода можно сгруппировать по категориям, как показано ниже

- **Последовательный** - этот включает в себя выполнение всех кодов в том порядке, в котором они были написаны.
- **Решение** - это включает в себя выбор с учетом ряда вариантов. Выполненный код зависит от значения условия.

Управляющие конструкции - это блок кода, который определяет путь выполнения программы в зависимости от значения заданного условия.

Давайте теперь посмотрим на некоторые структуры управления, которые поддерживает PHP.

PHP IF ELSE

IF ... then ... else самая **простая структура управления** . Он оценивает условия, используя булеву логику. **Когда использовать if… then… else**

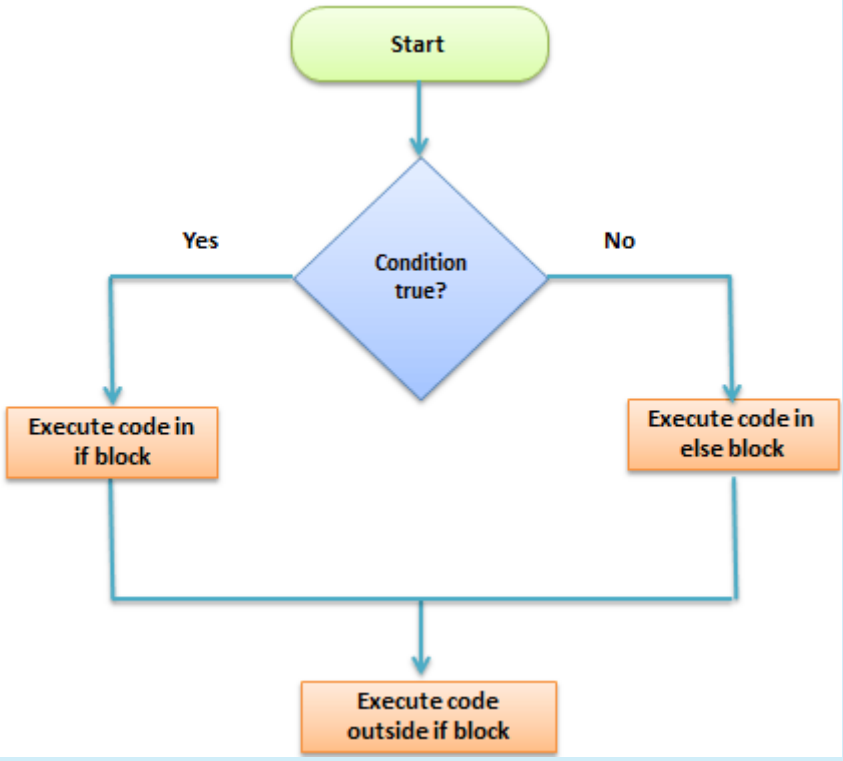
- У вас есть блок кода, который должен быть выполнен, только если выполняется определенное условие
- У вас есть два варианта, и вы должны выбрать один.
- If ... then ... else, if ... используется, когда вам нужно выбрать более двух вариантов, и вам нужно выбрать один или несколько

Синтаксис Синтаксис if… then… else ;

```
<?php

if (condition is true) {

block one }

else {

block two

}

?>
```

- « **If (условие истинно)»** является управляющей структурой
- « **block one**» - это код, который будет выполнен, если условие истинно
- **{… Else…}** - это запасной вариант, если условие ложно
- « **block two** » - блок кода, выполняемый, если условие ложно

Как это работает Блок-схема, показанная ниже, иллюстрирует, как работает структура управления if then… else



Давайте посмотрим на это в действии. Приведенный ниже код использует «if… then… else» для определения большего значения между двумя числами.

```
<?php

$first_number = 7;

$second_number = 21;

if ($first_number > $second_number){

echo "$first_number is greater than $second_number";

}else{

echo "$second_number is greater than $first_number";

}

?>
```

Выход:

```
21 is greater than 7
```

PHP Switch Case

Переключатель… case аналогичен структуре управления **if then… else** .

Он **выполняет** только один блок кода в зависимости от **значения** условия.

Если условия не были выполнены, выполняется блок кода по умолчанию.

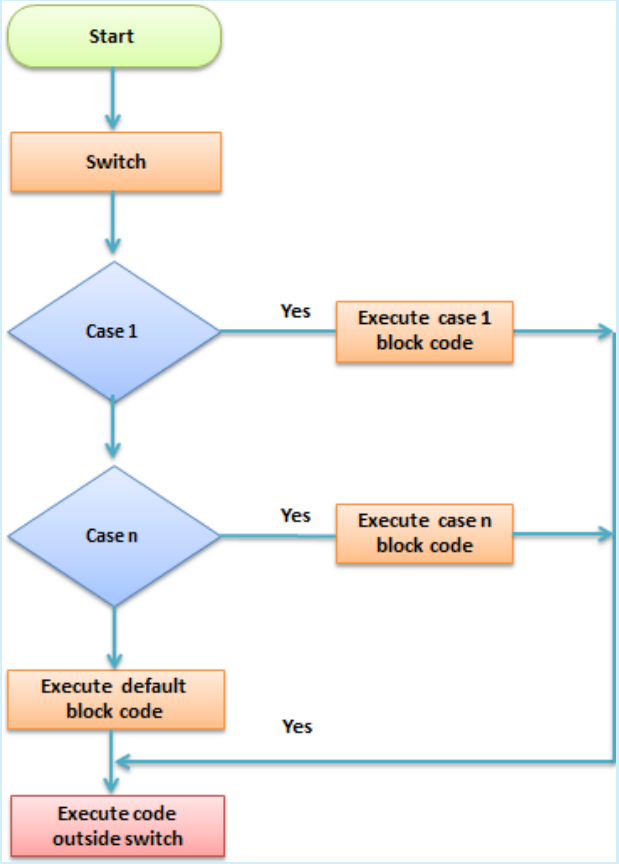
Он имеет следующий основной синтаксис.

```
<?php
switch(condition){
case value:
        //block of code to be executed

        break;
case value2:
        //block of code to be executed

        break;
default:
        //default block code

        break;
}
?>
```

- **«Switch (…) {…}»** - код блока структуры управления
- **«Case value: case…»** - блоки кода, которые должны быть выполнены в зависимости от значения условия
- **«Default:»** - блок кода, который будет выполнен, когда никакое значение не соответствует условию

Как это работает

Блок-схема, показанная ниже, иллюстрирует, как работает структура управления коммутатором.



Практический пример

Приведенный ниже код использует структуру управления коммутатором для отображения сообщения в зависимости от дня недели.

```
<?php

$today = "wednesday";

switch($today){

case "sunday":

echo "pray for us sinners."; break;

case "wednesday":

echo "ladies night, take her out for dinner"; break;

case "saturday":

echo "take care as you go out tonight."; break;

default:

echo "have a nice day at work"; break;

}?>
```

Выход:

```
ladies night, take her out for dinner
```

Резюме

- Управляющие структуры используются для контроля выполнения программы
- Если тогда ... то еще, когда у вас есть больше, чем маршрутный блок кода для выполнения в зависимости от значения условия
- Переключатель… регистр используется, когда у вас есть несколько кодов блоков, и вам нужно выполнить только один из них в зависимости от значения установленного регистра.

PHP Loop: For, ForEach, while, Do While

Цикл - это итеративная структура управления, которая включает выполнение одного и того же количества кода несколько раз, пока не будет выполнено определенное условие.

PHP для цикла

Вышеприведенный код выводит «21 больше, чем 7». Для циклов For ... циклы выполняют блок кода указанное число раз. Есть в основном два типа циклов for;

- для
- для каждого.

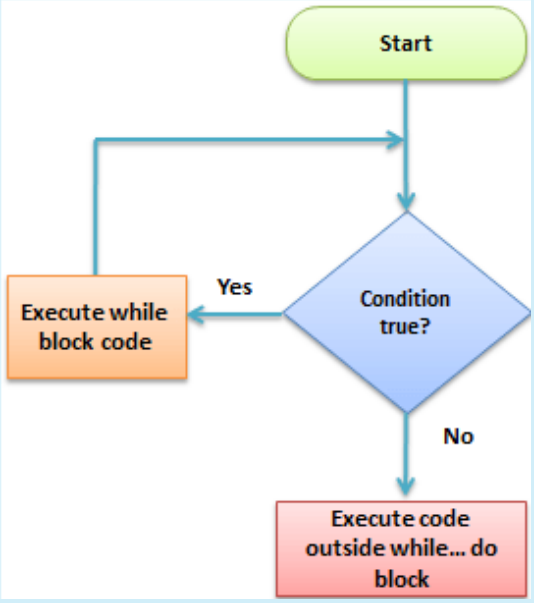
Давайте теперь посмотрим на них отдельно. **Цикл For** имеет следующий основной **синтаксис**

```
<?php

for (initialize; condition; increment){

//code to be executed

}

?>
```

- **«for…{…}»** - блок цикла
- « **initialize** » обычно целое число; используется для установки начального значения счетчика.
- **«condition»** условие, которое оценивается для каждого выполнения php. Если значение равно true, выполнение цикла for ... прекращается. Если значение равно false, выполнение цикла for ... продолжается.
- **«increment»** используется для увеличения начального значения целого числа счетчика.

Как это работает

Блок-схема, показанная ниже, иллюстрирует, как работает цикл for в php.



Как кодировать

В приведенном ниже коде используется цикл for… для вывода значений, умноженных с 10 на 0 до 10

```
<?php

for ($i = 0; $i < 10; $i++){

$product = 10 * $i;

echo "The product of 10 * $i is $product <br/>";

}

?>
```

Выход:

```
The product of 10 x 0 is 0

The product of 10 x 1 is 10

The product of 10 x 2 is 20

The product of 10 x 3 is 30

The product of 10 x 4 is 40

The product of 10 x 5 is 50

The product of 10 x 6 is 60

The product of 10 x 7 is 70

The product of 10 x 8 is 80

The product of 10 x 9 is 90
```

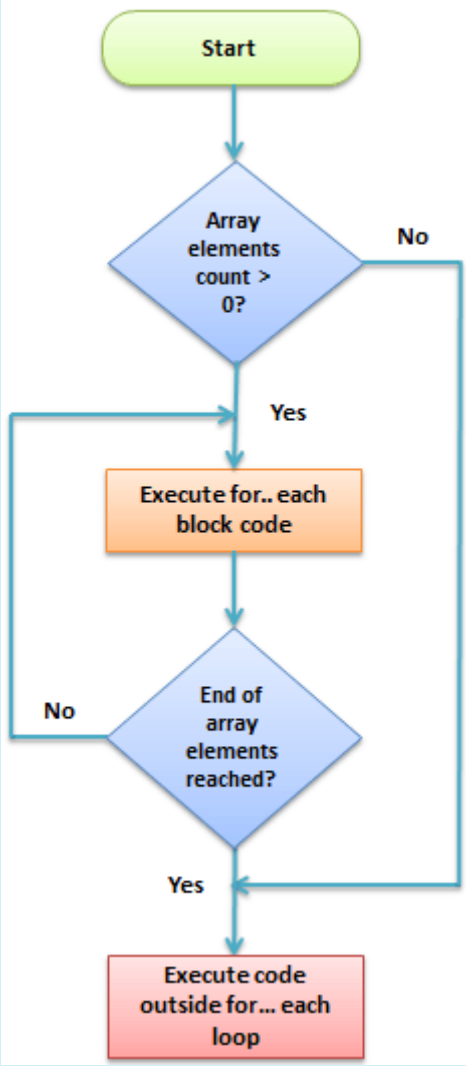
PHP для каждого цикла

Цикл php foreach используется для перебора значений массива. Имеет следующий основной синтаксис

```
<?php

foreach($array_variable  as $array_values){

block of code to be executed

}

?>
```

- **«Foreach (…) {…}»** - код блока цикла foreach php
- **«$ Array_data»** - переменная массива, которая будет проходить через
- **«$ Array_value»** - временная переменная, которая содержит текущие значения элемента массива.
- «block of code …» - это фрагмент кода, который работает со значениями массива

Как это работает На приведенной ниже блок-схеме показано, как работает цикл for… each…



Практические примеры

Приведенный ниже код использует для ... каждого цикла для чтения и печати элементов массива.

```
<?php

$animals_list = array("Lion","Wolf","Dog","Leopard","Tiger");

foreach($animals_list as $array_values){

echo $array_values . "<br>";

}

?>
```

Выход:

```
Lion

Wolf

Dog

Leopard

Tiger
```

Давайте посмотрим на другой пример, который проходит через **ассоциативный массив** .

Ассоциативный массив использует буквенно-цифровые слова для ключей доступа.

```
<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

foreach($persons as $key => $value){

echo "$key is $value"."<br>";

}

?>
```

Имена использовались как ключи массива, а пол - как значения.

Выход:

```
Mary is Female

John is Male

Mirriam is Female
```

While цикл

PHP while цикл

Они используются для многократного выполнения блока кода, пока не будет выполнено заданное условие

Когда использовать циклы while

- Циклы while используются для выполнения блока кода, пока определенное условие не станет истинным.
- Вы можете использовать цикл while для чтения записей, возвращаемых из запроса к базе данных.

Типы циклов

- **Do ... while** - выполнить блок кода хотя бы один раз, прежде чем оценивать условие
- **Пока ...** - сначала проверяет состояние. Если он принимает значение true, блок кода выполняется до тех пор, пока условие выполняется. Если значение равно false, выполнение цикла while прекращается.

While цикл

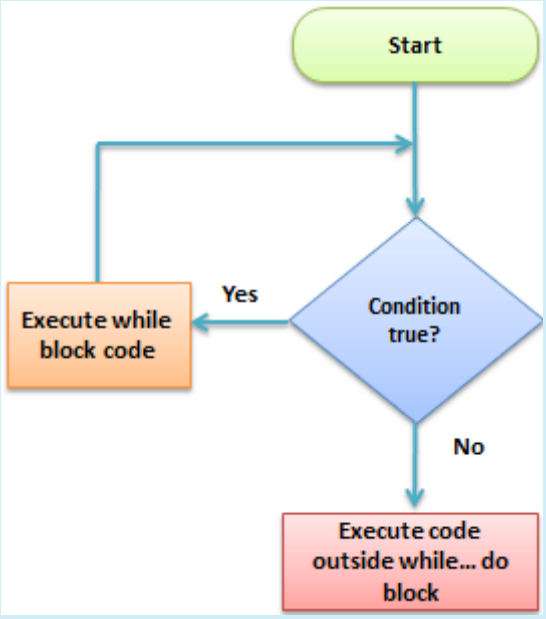
Имеет следующий синтаксис

```
<?php

while (condition){

block of code to be executed;

}

?>
```

- **«While (…) {…}»** - это код блока цикла while
- **«condition»** - это условие, которое будет оцениваться циклом while
- **«block of code…»** - это код, который будет выполнен, если условие выполнено

Как это работает

Блок-схема, показанная ниже, показывает, как работает цикл while ...



Практический пример

Приведенный ниже код использует цикл while… для печати чисел от 1 до 5.

```
<?php

$i = 0;

while ($i < 5){

echo $i + 1 . "<br>";

$i++;

}

?>
```

Выход:

```
1

2

3

4

5
```

PHP do while

Разница между циклами while… и do…, в то время как цикл do… while выполняется хотя бы один раз перед оценкой условия.

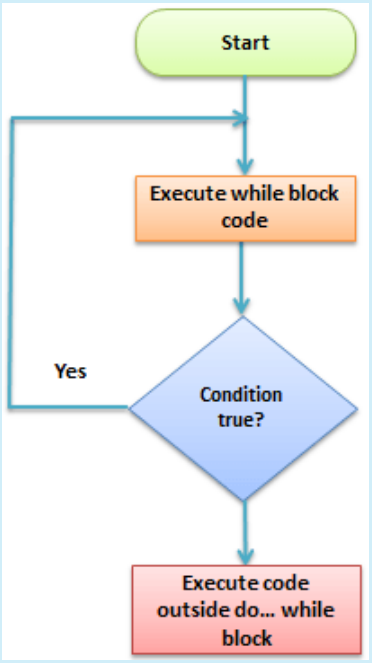
Давайте теперь посмотрим на основной синтаксис цикла do… while

```
<?php

do{

block of code to be executed

} while(condition);

?>
```

- **«Do {…} while (…)»** - это код блока цикла do… while
- **«condition»** - это условие, которое будет оцениваться циклом while
- **«block of code …»** - это код, который выполняется хотя бы один раз циклом do… while

Как это работает

Блок-схема, показанная ниже, показывает, как работает цикл while ...



Практический пример

Теперь мы собираемся изменить пример цикла while… и реализовать его с помощью цикла do… while и установить начальное значение счетчика равным 9.

Код ниже реализует приведенный выше модифицированный пример

```
<?php

$i = 9;

do{

   echo "$i is"." <br>";

}while($i < 9);

?>
```

Вышеприведенный код выводит:

```
9
```

Обратите внимание, что *в приведенном выше примере выводятся только 9.*

Это связано с тем, что цикл do… while выполняется хотя бы один раз, даже если заданное условие оценивается как ложное.

Резюме

- Цикл for… используется для выполнения блока указанное количество раз
- Цикл foreach… используется для обхода массивов
- Цикл while… используется для выполнения блока кода, если заданное условие считается ложным
- Цикл do… while используется для выполнения блока кода хотя бы один раз, тогда как остальная часть выполнения зависит от оценки установленного условия