STEP IT ACADEMY

Циклы и функции

СОЗДАНИЕ WEB–ПРИЛОЖЕНИЙ, ИСПОЛНЯЕМЫХ НА СТОРОНЕ СЕРВЕРА ПРИ ПОМОЩИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PHP, СУБД MYSQL И ТЕХНОЛОГИИ AJAX

МОДУЛЬ 01. ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ PHP

Содержание

1. Понятие циклов

ОПРЕДЕЛЕНИЕ И ИСПОЛЬЗОВАНИЕ

Определение

Циклы позволяют несколько раз повторять некоторые команды, которые называются *телом цикла*. Проход цикла называется *итерацией*. PHP поддерживает следующие виды циклов:

1. Цикл с предусловием.
2. Цикл с постусловием.
3. Цикл со счётчиком.
4. Цикл перебора массивов.

Использование циклов

- Использование циклов упрощает и укорачивает код. Также циклы незаменимы в ситуациях, когда заранее неизвестно сколько раз должен выполниться блок кода, если число зависит от множества условий и вычисляться в момент выполнения сценария.

2. Виды циклов

2.1. Цикл с предусловием

ОПРЕДЕЛЕНИЕ, НАЧАЛО И КОНЕЦ ЦИКЛА

Определение и начало оператора

Такой цикл сначала вычисляет значение логического выражения, и в случае `true` выполняет тело цикла, иначе пропускает его.

Классический синтаксис

```
while (cond) {
    // Body
}
```

Альтернативный синтаксис

```
<? while (cond): ?>
    // Alt body
```

Конец оператора

Используется только в альтернативном синтаксисе в качестве завершающего элемента

Классический синтаксис	Альтернативный синтаксис
–	`<? endwhile; ?>`

STEP IT ACADEMY

2. Виды циклов

2.1. Цикл с постусловием

ОПРЕДЕЛЕНИЕ, НАЧАЛО И КОНЕЦ ЦИКЛА

Определение и начало оператора

В отличие от предыдущего цикла, этот вид сначала выполняет тело, а потом проверяет условие. Таким образом, тело цикла выполнится хотя бы один раз.

Классический синтаксис

```
do {
    // Body
}
```

Альтернативный синтаксис

—

Конец оператора

Данный тип цикла **не имеет** поддержки альтернативного синтаксиса.

Классический синтаксис	Альтернативный синтаксис
`while (cond)`	–

STEP IT ACADEMY

2. Виды циклов

2.1. Цикл со счётчиком

ОПРЕДЕЛЕНИЕ, НАЧАЛО И КОНЕЦ ЦИКЛА

Определение и начало оператора

Этот цикл используется для выполнения тела некоторого количества раз. Он начинается с **инициализации** (1), проверяет **условие** (2) и начинает итерацию, а после **выполняет заданного действия инкремента или декремента** (3).

Классический синтаксис

```
for (1; 2; 3) {
    // Body
}
```

Альтернативный синтаксис

```
<? for (1; 2; 3): ?>
    // Alt body
```

Конец оператора

Используется только в альтернативном синтаксисе в качестве завершающего элемента

Классический синтаксис	Альтернативный синтаксис
–	`<? endfor; ?>`

2. Виды циклов

2.1. Цикл перебора массивов

ОПРЕДЕЛЕНИЕ, НАЧАЛО И КОНЕЦ ЦИКЛА

Определение

Существует цикл для последовательного перебора всех элементов массива или объекта, который не может быть использовать с переменными другого типа. Он бывает двух видов:

1. **Простой перебор** — переменные перебираются только по индексам: `$arrExp as $value`.
2. **Парный перебор** — используется для перебора элемента в паре с ключом: `$arrExp as $key => $value`.

Начало оператора

Начало цикла

Классический синтаксис

```php
foreach ($a as $item) {
    // Body
}
```

Альтернативный синтаксис

```php
<? foreach ($a as $item): ?>
    // Alt body
```

Такой цикл будет продолжаться, пока не пройдёт через каждый элемент массива. Также он использует копию массива и любые изменения в оригинальный массив не будут применяться.

Конец оператора

Конец цикла

Используется только в альтернативном синтаксисе в качестве завершающего элемента

Классический синтаксис

—

Альтернативный синтаксис

```
<? endforeach; ?>
```

STEP IT ACADEMY

3. Прерывание цикла и итерации

ОПРЕДЕЛЕНИЕ И ИСПОЛЬЗОВАНИЕ

Прерывание цикла

Конструкция `break` используется для немедленного выхода из цикла.

Конструкция может задана с *необязательным* параметром — номером вложенного цикла, из которого нужно выйти:

- `break;`
- `break(1);`

Прерывание итерации

Конструкция `continue` используется для выхода из итерации. Как и `break`, она может задаваться с номером цикла для выхода и позволяет сэкономить количество фигурных скобок в коде, чтобы увеличить его читаемость.

Чаще всего используется в циклах-фильтрах, где нужно выбрать только значения, что удовлетворяют заданным условиям.

STEP IT ACADEMY

4. Что такое функция?

ОПРЕДЕЛЕНИЕ И ОПИСАНИЕ

Определение

Функция — это важный элемент программирования, который позволяет группировать и использовать один и тот же код множество раз. Функции также называют *подпрограммой*, поэтому она имеет свою точку входа и свой «выход».

Описание

- Функция может быть названа *чёрным ящиком*.
- Она определяет свою локальную область видимости, куда входят входные параметры, и переменные, что объявляются в теле.
- Главная особенность функции — это её *вызов*.
- Функция также позволяет делать **декомпозицию** — разделение на более простые части

STEP IT ACADEMY

5. Типы функций

ВСТРОЕННЫЕ И ПОЛЬЗОВАТЕЛЬСКИЕ

Существующие типы

Существуют два типа функций — *встроенные* и *пользовательские:*

- **Встроенные функции** уже написали создатели языка программирования.
- **Пользовательские** функции создаёт разработчик. Они используются только внутри одного проекта или сценария.

STEP IT ACADEMY

6. Анатомия функций

ПРИНЦИП РАБОТЫ И ПРИМЕР

Принцип работы

Работа с функциями состоит из *объявления* и *определения:*

- **Объявление** (`declaration`) функции содержит список входных аргументов.
- **Определение** (`definition`) функции содержит исполняемый код функции.

Пример

```
function <имя функции> (<аргументы функции>) {
    // Тело функции
}
```

Возвращение значений

- Функция может *возвращать* некоторое значение.
- Такое значение называется **результатом функции**.
- Для возвращения значения применяется ключевое слово `return`.

Функция сложения

```php
function add (int $a, int $b = 2) {
    return $a + $b;
}
```

Использование значения

Для того, чтобы использовать возвращаемое значение, результат функции необходимо будет сохранить в переменную, либо использовать напрямую, например, вместе с оператором `echo`:

- `$result = add(1, 2);`
- `echo add(1, 2);`

Инструкции после `return`

Если после `return` в функции идут другие инструкции, то они не будут выполняться:

```
function add (int $a, int $b) {
    return $a + $b;
    echo $a + $b;
}
```

```
Unreachable code detected
PHP(PHP0419)
```

Конструкция выбора

```
function example ($var) {
    switch ($var) {
        case "value":
            // Тело кейса
            return true;
    }
}
```

7. Встроенные функции

ПРИМЕРЫ

Примеры

- `rand(int $min, int $max)` — генерирует случайное число.

  Например, вызов `rand(1, 12)` вернёт случайное число от 1 до 12.
- `strlen(string $string)` — возвращает длину строки
- `substr_count(string $haystack, string $needle)` — возвращает число вхождений подстроки.

  Например, вызов `substr_count("This is a function", "is")` вернёт число 2, так как в строке дважды повторяется is

Дополнительная литература

1. Документация по PHP: https://www.php.net
2. Цикл `while`: https://www.php.net/manual/ru/control-structures.while.php
3. Цикл `do-while`: https://www.php.net/manual/ru/control-structures.do.while.php
4. Цикл `for`: https://www.php.net/manual/ru/control-structures.for.php
5. Цикл `foreach`: https://www.php.net/manual/ru/control-structures.foreach.php
6. Прерывание цикла: https://www.php.net/manual/ru/control-structures.break.php
7. Прерывание итерации: https://www.php.net/manual/ru/control-structures.continue.php

Дополнительная литература

9. Функции для работы со строками: https://www.php.net/manual/ru/ref.strings.php
10. Математические модули: https://www.php.net/manual/ru/ref.math.php