Процедуры и функции PHP

Вопрос

- Чем отличаются процедуры от функций?

Особенности пользовательских функций PHP

- Доступны параметры по умолчанию. Можно вызывать одну и ту же функцию с переменным числом параметров;
- Пользовательские функции могут возвращать любой тип;
- Область видимости переменных внутри функции является иерархической (древовидной);
- Есть возможность изменять переменные, переданные в качестве аргумента.

- Невозможность объявления локальных функций

Нельзя создать функцию внутри другой функции таким образом, чтобы первая (вложенная) функция была видна только во второй функции. В PHP вложенная функция будет доступна всей программе (скрипту), а значит не будет локальной.

Пример

```
<?php
function first_function()
{
  echo "<h4>Первая пользовательская
функция</h4>";
    function second_function()
      {
         echo "<h5>Вторая
пользовательская   функция</h5>";
        }
}
first_function();
second_function();
?>
```

Какой будет результат?

Результат

Первая пользовательская функция
Вторая пользовательская функция

- Все объявленные и используемые в функции переменные по умолчанию локальны для функции.

То есть, по умолчанию нет возможности изменить значение глобальной переменной в теле функции.

Пример

```
<?php
$a = 100;
function funct() {
$a = 70;
 echo "<h4>$a</h4>";
}
funct();
echo "<h2>$a</h2>";
?>
```

Что выведет сценарий?

Ответ

Сценарий выведет спервa 70, а затем 100

В PHP существует специальная инструкция **global**, позволяющая пользовательской функции работать с глобальными переменными.

Исправляем пример

```
<?php
$a = 100;
function funct() {
global $a;
$a = 70;
 echo "<h4>$a</h4>";
}
funct();
echo "<h2>$a</h2>";
?>
```

Другой вариант (используем $GLOBALS)

```
<?php
$a = 100;
function funct() {
$GLOBALS(“a”) = 70;
 echo "<h4>$a</h4>";
}
funct();
echo "<h2>$a</h2>";
?>
```

Создание пользовательских функций

Пользовательская функция может быть объявлена в любой части программы (скрипта), до места ее первого использования.

Синтаксис

```
function Имя
(аргумент1[=значение1],...,
аргумент1[=значение1])
{
тело_функции;
}
```

Объявление функции начинается служебным словом **function**, затем следует имя функции, после имени функции - список аргументов в скобках. Тело функции заключается в фигурные скобки и может содержать любое количество операторов.

Требования к неймингу

- Имена функций могут содержать русские буквы, но давать функциям имена, состоящие из русских букв не рекоммендуется;
- Имена функций не должны содержать пробелов;
- Имя каждой пользовательской функции должно быть уникальным. Регистр при объявлении функций и обращении к ним не учитывается. То есть, например, функции funct() и FUNCT() имеют одинаковые имена;
- Функциям можно давать такие же имена, как и переменным, только без знака $ в начале имен.

- Для передачи результата работы пользовательских функций в основную программу (скрипт) используется конструкция **return**.
- Если функция ничего не возвращает, конструкцию **return** не указывают.
- **return** может возвращать все, что угодно, в том числе и массивы.

Пример

```
<?php
function funct() {
$number = 777;
 return $number;
}
$a = funct();
echo $a;
?>
```

Что выведет данный скрипт?

Передача аргументов пользовательским функциям

При объявлении функции можно указать список параметров, которые могут передаваться функции

Пример

- <?php
  function funct($a, $b, $z) { ... };
  ?>

- По умолчанию функции не могут изменить параметр.

если вы измените значение аргумента внутри функции, то вне ее значение все равно останется прежним

Пример

- ```
  <?php
  function funct($string)
  {
      echo "<h3>Параметр = $string </h3>";
  }

  $str = 777;
  funct(777);
  funct($str);

  // Функция 'funct' выведет строку 'Параметр = 777' дважды

  ?>
  ```
  ```

- Если необходимо разрешить функции модифицировать свои аргументы, вы должны передавать их по ссылке, то есть использовать переменную-ссылку.

Пример

```
<?php
function funct(&$string)
{
     $string .= 'а эта внутри.';
}
$str = 'Эта строка за пределами
функции, ';
funct($str);
echo $str;    // Выведет 'Эта строка
за пределами функции, а эта внутри.'
?>
```

Что выведет данный скрипт?

Параметры по умолчанию

- В PHP можно определить значение параметров по умолчанию, тогда нет необходимости вводить значение этого параметра при вызове функции. Подставится значение по умолчанию.

Пример

```
<?php
function makecup($type = "Чая")
{
     return "Сделайте чашечку
$type.\n";
}
echo makecup();
echo makecup("Кофе");
?>
```

Что выведет данный скрипт?

Особенности объявления параметров

- PHP также позволяет использовать массивы и специальный тип NULL в качестве значений по умолчанию.
- Все аргументы, для которых установлены значения по умолчанию, должны находиться правее аргументов, для которых значения по умолчанию не заданы.

Неправильно:

- ```
  <?php
  function makecup($type = "чая", $c
  ond)
  {
       return "Сделайте
  чашечку $type $cond.\n";
  }
  echo makecup("горячего");
  ?>
  ```

Выведет ошибку
  ```

Правильно:

```
<?php
function makecup($cond, $type = "ча
я")
{
     return "Сделайте
чашечку $type $cond.\n";
}
echo makecup("горячего");
?>
```

Выведет Сделайте чашечку чая горячего

Закрепление материала

- Как объявить функцию?
- Можно ли вызвать отдельно подфункцию?
- Изменится ли переменная, объявленная вне функции, если мы изменили ее внутри функции?
- Можно ли использовать в качестве параметров функции массивы? NULL?
- Как создать функцию, которая ничего не возвращает?