МДК 02.02. Web-программирование. Язык PHP

Функции в PHP

Функция – это блок программного кода, который принимает некоторые значения, обрабатывает их и выполняет определенные действия.

Преимущества использования функций

- избавляют от многократного повторения одних и тех же фрагментов кода в программах;
- облегчают чтение кода и позволят свести к минимуму правку

Создание пользовательской функций

Синтаксис

function имя_функции([аргументы])

{

программный код (тело функции)

}

Аргументами называются неопределенные входящие данные.

Параметрами являются известные данные.

Параметры – это способ передачи информации в функцию, позволяющий не заботиться об области видимости переменных.

Требования, предъявляемые к именам функций:

1. Не рекомендуется использовать русские буквы в именах функций.
2. Имена функций не должны содержать пробелов.
3. Имя каждой пользовательской функции должно быть уникальным.
4. Регистр при объявлении функций и обращении к ним не учитывается. То есть, например, функции funct() и FUNCT() имеют одинаковые имена.
5. Знак $ в начале имен функций не указывается.

Вызов функций

Синтаксис

имя_функции(параметры);

Функции допускается определять в любом месте программы

Пример

```
<?php
function hi()
{
echo ("Привет из мира функций!");
}
hi(); // Вызов функции
?>
```

Особенности пользовательских функций PHP

1. Доступны параметры по умолчанию. Одну и ту же функцию с переменным числом параметров.
2. Пользовательские функции могут возвращать любой тип.
3. Есть возможность изменять переменные, переданные в качестве аргумента.

Определение значений по умолчанию для параметров функции

Пример

```
<?php
function wrap_in_html_tag($text, $tag = 'strong')
{
return "<$tag>$text</$tag>";
}
echo wrap_in_html_tag("строка");
?>
```

При присваивании значений по умолчанию следует помнить о двух важных обстоятельствах.

1. Все параметры со значениями по умолчанию должны перечисляться после параметров, не имеющих значений по умолчанию.
2. Присваиваемое значение по умолчанию должно быть константой, строкой или числом. Оно не может быть переменной.
3. Если по умолчанию переменная не должна содержать ничего, можно присвоить параметру пустую строку: $tag = ''.

Конструкции включений в PHP

Конструкции включений позволяют собирать PHP программу (скрипт) из нескольких отдельных файлов.

Функции

1. include;
2. require;
3. include_once;
4. require_once.

Конструкция включений require

Конструкция **require** позволяет включать файлы в сценарий PHP до исполнения сценария PHP.

Синтаксис

require имя_файла;

Конструкция **require** позволяет собирать сценарии PHP из нескольких отдельных файлов, которые могут быть как html-страницами, так и php-скриптами.

Конструкция **require** поддерживает включения удаленных файлов

Пример

Файл header.html:

```
<html>
    <head>
        <title> Интернет-магазин канцтоваров</title>
    </head>
    <body bgcolor=green>
        <h1>Интернет-магазин "Канцтовары"</h1>
        <img  src="1.png">
    </body>
</html>
```

Пример

Файл script.php

```
<?php
    require "header.html";
?>
```

Конструкция включений include

Предназначена для включения файлов в код сценария PHP во время выполнения сценария.

Синтаксис

include имя_файла;

Конструкции require_once и include_once

Работают аналогично инструкциям include и require.

Позволяют включать файлы однократно.

Создание структуры страниц с помощью функций включения

Пример файл index.php

```
<!DOCTYPE html>
<html>
    <head>  …  </head>
    <body>
<?php
    include_once "include/header.php";
    include_once "include/nav.php";
?>
Основная часть страницы
<?php
    include_once "include/footer.php";
?>
    </body>
</html>
```

Файл header.php

```
<header>
    <div>
        …
    </div>
</header>
```

Файл nav.php

```
<nav>
    <div>
        …
    </div>
</nav>
```

Файл footer.php

```
<footer>
    <div>
        …
    </div>
</footer>
```

Включение содержимого текстовых файлов в документ с помощью функции include

Пример

```
<?php
    for($i=1; $i<=5; $i++)
    {
    include "text/$i.txt";
    }
?>
```

Проверка существования функции

Функция function_exists

Пример

```
<?php
function test_this()
{}
$test=function_exists("test_this");
if ($test == TRUE)
{echo "Функция test_this существует.";}
else
{echo "Функция test_this не найдена.";}
?>
```

Способы записи функций

1. в операторной форме

```
echo 'AAA';
```

2. в функциональной форме

```
echo('AAA');
```

Область видимости переменной

Локальные переменные

Локальные переменные – переменные, определенные внутри подпрограммы (пользовательской функции). Они доступны только внутри функции, в которой они определены.

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

Глобальные переменные – это переменные, которые доступны всей программе, включая подпрограммы (пользовательские функции).

Пример

```
<?php
$a = 100;          //глобальная переменная
function funct() {
$a = 70;           //локальная переменная
 echo "$a<br>";
}
funct();
echo $a;
?>
```

Инструкция global позволяет пользовательской функции работать с глобальными переменными

Пример

```
<?php
$a = 1;
$b = 2;
function Sum() {
    global $a, $b;
    $b = $a + $b;
}
Sum();
echo $b;
?>
```

Пример

```
<?php
$a = 1;       // глобальная область видимости
function Test()
{
    echo $a;      // локальная область видимости
}

Test();
?>
```

Статические переменные в PHP

Если в теле пользовательской функции объявлена статическая переменная, то компилятор не будет ее удалять после завершения работы этой функции.

Объявление статических переменных:

```
static $int = 0;
```

Пример

```
<?php
  function funct()
  {
    static $a;
    $a++;
    echo "$a";
  }
  for ($i = 0; $i++<10;) funct();
?>
```

Конструкции возврата значений

Конструкция return

Конструкция **return** используется для возврата значений пользовательскими функциями.

Возвращаемые значения могут быть любого типа, в том числе это могут быть списки и объекты

Условно определяемые функции

Пример

```
<?php
$phpver = phpversion();
if ($phpver[0] === "5")
{function getversion() { return "Вы используете PHP5"; }}
if ($phpver[0] === "4")
{function getversion() { return "Вы используете PHP4"; }}
if ($phpver[0] === "3")
{function getversion() { return "Вы используете PHP3"; }}
echo @getversion();
?>
```

Стандартные функции PHP

Категории функций:

- ✔ функции для работы со строками;
- ✔ функции для работы с массивами;
- ✔ функции для работы с файловой системой;
- ✔ функции баз данных;
- ✔ математические функции;
- ✔ календарные функции и функции времени;
- ✔ графические функции;
- ✔ функции для работы с сессиями;
- ✔ функции управления выводом;
- ✔ функции для работы с ядром php;
- ✔ функции для работы с документами;
- ✔ криптографические функции php;
- ✔ функции авторизации и аутентификации.