Практическая работа №15
Тема: «Пользовательские функции. Вызов функции. Функции обработки строк. Функции даты и времени»

Цель: работать с наиболее популярными функциями PHP

Теоретическая часть

Введение

Функции – это особым образом оформленный фрагмент кода PHP, который можно вызвать из любого места любого сценария, присутствующего в данной серверной странице. Как правило, в виде функции оформляется код, выполняющий однотипные и часто используемые в сценариях задачи.

Создание функций

Прежде чем функция будет использоваться, ее нужно объявить Объявление выполняется с помощью ключевого слова ***function***.

```
<?php
function имя функции (параметры)
{
    тело функции;
}
?>
```

Для имен функции действуют те же правила, что и для имен переменных (только латинские буквы и цифры и знаки подчеркивания). Параметры, которые используются внутри функции называются ***локальными переменными***. Каждая функция имеет свои локальные переменные. Для того, чтобы переменные можно было использовать переменные в любой функции, нужно их задать как глобальные: **`global $a, $b`**;.
Для получения результата действия функции используется оператор ***return***:

```
retutn <переменная или выражение>;
```

Пример:

```
function divide ($a, $b) {
$c = $a / $b ;
return $c;
}
```

Вызов функций

Вызвать функцию можно в любом месте сценария серверной страницы:

```
имя функции (параметры);
```

Пример:
Нужно вызвать функцию **`divide`** и задать ей параметры **`$a=3; $b=2;:`**

```
echo divide (3,2);
```

Результат вывода будет **1,5**.

Пример вызова встроенной функции:

```
<?php
phpinfo();
?>
```

Параметры функции

Параметрами функции могут быть переменные, выражения и другие функции

1. Параметры – **переменные**:	2. Параметры – **выражения**:
`<? php` `function get_num ($left, $right)` `{` `$sum = $left + $right;` `echo $sum;` `}` `get_num (10,5);` `?>`	`<? php` `function func ($left, $ middle, $right) //объявление функции` `{` `echo $left " ";` `echo $ middle ." ";` `echo $right ." ";` `}` `$i=10` `func (++$i, $i=$i+2, --$i);` `//вызов функции` `?>`
Здесь **переменные `$left, $right`** являются *параметрами*, а их значения ***10 и 5*** – *аргументами*.	Здесь **выражения `$left, $ middle, $right`** являются *параметрами*, а их значения **`++$i, $i=$i+2, --$i`** – *аргументами*. Результатом выполнения данного скрипта будет последовательность чисел: 11 22 21

Строковые функции и операторы

Конкатенация строк.

В различных языках программирования используются различные операторы конкатенации (объединения) строк. Например, в Pascal используется оператор "+". В PHP есть два оператора, выполняющие конкатенацию.

Первый - оператор конкатенации (**`'.'`**), который возвращает объединение левого и правого аргумента.

Второй - **`оператор присвоения`** с конкатенацией, который присоединяет правый аргумент к левому.

Приведем конкретный пример:

```
<?php
$a = "Hello ";
$b = $a . "World!"; // $b содержит строку "Hello World!" - Это конкатенация

$a = "Hello ";
```

```
$a .= "World!";     // $a содержит строку "Hello World!" - Это присвоение с конкатенацией
?>
```

Функции проверки существования переменной

Функция empty

Зачастую в веб-программировании возникает следующая проблема: необходимо проверить переменную на пустоту.
Переменная будет пустая, если она равна нулю, " (пустой строке), или null (то есть не определена ранее).
Проверка выполняется с помощью функции empty():

```
$a = null;
if (empty($a))
{
echo 'Верно!';
}
else
{
echo 'Неверно!';
}
```
//если $a пустое, то...выведет 'Верно!'

```
$a = 0;
if (empty($a))
{
echo 'Верно!';
}
 else
{
echo 'Неверно!';
}
```
//если $a пустое, то... выведет 'Верно!'

```
$a = '';
if (empty($a))
{
echo 'Верно!';
}
else
{
echo 'Неверно!';
}
```
//если $a пустое, то... //выведет 'Верно!'

```
$a = 'hi';
if (empty($a))
{
echo 'Верно!';
```

```
}
else
{
echo 'Неверно!';
}
```
//выведет 'Неверно!', так как $a не пустое//если $a пустое, то...

Чаще возникает обратная задача — проверка на то, что переменная является НЕ пустой. Отрицание НЕ можно сделать с помощью оператора '!':

```
$a = null;
if (!empty($a))
{
echo 'Верно!';
}
else
{
echo 'Неверно!';
}
```
//если $a Непустое, то...
//выведет 'Неверно!' - $a пустое

Функция isset

Аналогом empty является функция isset. Она проверяет существует ли переменная (то есть равна null или не равна null).

```
$a = null;
if (isset($a))
{
echo 'Верно!';
}
else
{
echo 'Неверно!';
}
```
Если $a существует, то... выведет 'Верно!', так как $a существует.

```
Пусть переменную $b вообще не определяли выше в коде.
if (isset($b))
{
echo 'Верно!';
}
else
{
echo 'Неверно!';
}
```

Если $a существует, то...выведет 'Неверно!'

Аналогично проверятся на НЕ существование (через !isset):

```
$a = 3;
```

```
if (!isset($a))
{
echo 'Верно!';
}
else
{
echo 'Неверно!';
}
```
Если $a НЕ существует то.. выведет 'Неверно!', так как $a существует

Функции для работы со строками

Для работы со строками в PHP существует множество полезных функций.
Кратко разберем часть функций для работы со строками.
Базовые строковые функции

1)strlen(string $st)

Одна из наиболее полезных функций. Возвращает просто длину строки, т. е., сколько символов содержится в $st. Строка может содержать любые символы, в том числе и с нулевым кодом (что запрещено в Си). Пример:

```
$x = "Hello!";
echo strlen($x);
```
// Выводит 6

2)substr(string $str, int $start [,int $length])

Данная функция тоже востребуется очень часто. Ее назначение — возвращать участок строки **$str**, начиная с позиции **$start** и длиной **$length**.
Пример:

```
$str = "Programmer";
echo substr($str,0,2);  // Выводит Pr
echo substr($str,-3,3);  // Выводит mer
```

3)str_replace($from, $to, $str)

Заменяет в строке $str все вхождения подстроки $from (с учетом регистра) на $to и возвращает результат.
Пример:

```
$str = "PHP  и mySQL";
$st=str_replace("m","M",$str) // Заменяет m на M и возвращает «PHP и MySQL»
```

Функции для работы с отдельными символами

Как и в других языках программирования, в PHP можно работать с символами строк отдельно.
Обратиться к любому символу строки можно по его индексу:

```
1)$str = "PHP";
echo $str[0]; // Выводит 'P'
```

2) **chr(int $code)**

Данная функция возвращает строку, состоящую из символа с кодом $code.
Пример:

```
echo chr(75); //Выводит K
```

3) **ord($char)**
Данная функция возвращает код символа **$char**.
Пример:

```
echo ord('A'); // Выводит 65 - код буквы 'A'
```

Функции изменения регистра

1) strtoupper(string $str)

Перевод строки в верхний регистр, т. е. делать все прописные буквы в строке заглавными.

2) strtolower(string $str)

Перевод троки в нижний регистр.

Функция кодирования пароля

Иногда нужно закодировать пароль для ограничения доступа. Для этого существует функция:

```
md5(string $str)
```

Пример:

```
$str="Пароль";
$new_parol = md5 ($str);
echo $new_parol; // Выводит «5ebe553e01799a927b1d045924bbd4fd»
```

Функции для работы с датой и временем

1) date

```
date -- Форматирует системную дату/время
```

Описание:

```
string date ( string format [, int timestamp] )
```

Возвращает время, отформатированное в соответствии с аргументом **format**, используя метку времени, заданную аргументом **timestamp** или текущее системное время, если **timestamp** не задан. Другими словами, **timestamp** является необязательным и по умолчанию равен значению, возвращаемому функцией **time()**.
Пример форматирования с использованием **date()**

```
<?php
// Предположим, что текущая дата March 10th, 2001, 5:16:18 pm

$today = date("F j, Y, g:i a");                 // March 10, 2001,
5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, n, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 10-
03-01, 1631 1618 6 Fripm01
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');   // It is the 10th
day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10
15:16:08 MST 2001
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');     // 17:03:17 m is
month
$today = date("H:i:s");                         // 17:16:17
```

2) time

time -- Возвращает текущую метку времени
Описание
`int time ( void )`
Возвращает количество секунд, прошедших с начала Эпохи Unix (The Unix Epoch, 1 января 1970, 00:00:00 GMT) до текущего времени.

Практическая часть

Задание 1. Работа с empty и isset

1. Если переменная $a пустая, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 1, 0, -3, null, true, ''.
2. Если переменная $a НЕ пустая, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 1, 0, -3, null, true, ''.
3. Если переменная $a существует, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 1, 0, -3, null, true, ''.
4. Если переменная $a НЕ существует, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 1, 0, -3, null, true, ''.

Задание 2. Работа со строковыми функциями

1. **`Задать функцию по выводу на экран текста «Функции делятся на встроенные`** ***и пользовательские. К встроенным функциям относятся функции по работе с изображениями обработкой строк переменными базами данных и другие. Все они имеют собственные наименования, которые нельзя использовать при создании пользовательских функций.»*** Использовать оператор **`return`**..
2. Подсчитать количество байт и количество символов в выражении «Волоколамский колледж права, экономики и безопасности» с помощью функции **`strlen.`** (один русский символ занимает 2 байта.
3. Создайте строковую переменную **`login`** и присвойте ей значение **"root"**. Создайте строковую переменную **`password`** и присвойте ей значение **"megaP@ssw0rd".** Создайте строковую переменную **`email`** и присвойте ей значение **ivan@petrov.ru.** Используя строковые функции, сделайте первый символ значения переменной **login** прописной (большой), сделайте первый символ значения переменной **password** прописной (большой)

Задание 3. Работа с функциями даты и времени

4. Используя встроенную функцию **`date()`** выведите на экран текущую дату и время.
5. Создайте строковую переменную **`now`**. Создайте строковую переменную **birthday**. Присвойте переменной **now** значение **метки времени актуальной даты (сегодня)**. Присвойте переменной **`birthday`** значение метки времени **Вашего дня рождения**. Выведите **фразу "До моего дня рождения осталось "**. Выведите количество секунд, оставшееся до Вашего дня рождения Закончите фразу **" секунд"**