Практическое занятие №21 Строки, Логический тип данных, приведение типов

Задание №1

Проверьте запущен ли у Вас Open Server. Папка с материалами урока (php21.loc) должна быть размещена в директории /domains сервера. Заходим на http://php21.loc и начинаем урок.

Задание №2

Строка в PHP - это набор символов любой длины. Строки могут быть заключены в одинарные (') или двойные кавычки ("). Рассмотрим и протестируем код создания строк в PHP:

```php
$string1 = 'первая строка';
$string2 = "вторая строка";
```

В отличии от JavaScript в PHP для конкатенации (склеивания) строк используется оператор «.», пример склеивания строк:

```php
// неправильный вариант склеивания строк
echo $string1 + $string2;

echo '<br>';

// правильное склеивание строк
echo $string1.' '.$string2;
```

У оператора конкатенации есть особенности использования, так есть он находиться между двумя числами, то он будет разделителем целой и дробной части:

```php
// разделитель целой и дробной части
echo 4.56;

echo '<br>';

// ошибка
echo 4.'56';
echo 4. 56;

echo '<br>';

// конкатенатор строк
var_dump(4 . 56);
```

В последнем примере используется встроенная функция **var_dump**, которая выводит значение и тип, чтобы показать Вам, что в результате мы получили строковый тип.

Задание №3

Управляющие последовательности в PHP очень похоже на JavaScript. В PHP есть особенность - строки в двойных кавычках «"» функционируют так же как и в JavaScript, строки в «'» выводятся «как есть».

Ниже представляем Вам таблицу управляющих последовательностей PHP:

Последовательность	Значение
\n	новая строка (LF или 0x0A (10) в ASCII)
\r	возврат каретки (CR или 0x0D (13) в ASCII)
\t	горизонтальная табуляция (HT или 0x09 (9) в ASCII)
\v	вертикальная табуляция (VT или 0x0B (11) в ASCII) (с версии PHP 5.2.5)
\e	escape-знак (ESC или 0x1B (27) в ASCII) (с версии PHP 5.4.4)
\f	подача страницы (FF или 0x0C (12) в ASCII) (с версии PHP 5.2.5)
**	обратная косая черта
\$	знак доллара
\"	двойная кавычка
\[0-7]{1,3}	последовательность символов, соответствующая регулярному выражению символа в восьмеричной системе счисления, который молча переполняется, чтобы поместиться в байт (т.е. "\400" === "\000")
\x[0-9A-Fa-f]{1,2}	последовательность символов, соответствующая регулярному выражению символа в шестнадцатеричной системе счисления
\u{[0-9A-Fa-f]+}	последовательность символов, соответствующая регулярному выражению символа Unicode, которая отображается в строка в представлении UTF-8 (добавлено в PHP 7.0.0)

Рассмотрим и протестируем код:

```
echo '<pre>';

echo "\t\t строка с табуляцией и  \n переносом на новую строку \n";

echo '\t\t строка без табуляции и  \n переноса на новую строку';
```

Единственным исключением экранирования строки в одинарных кавычках является сама одинарная кавычка:

```
echo 'O\'relly';
```

Задание №4

В PHP есть возможность, которой нет в JavaScript, это возможность подставлять переменные непосредственно в строку, разберем и протестируем пример ниже:

```
$name = 'John';
$age = 29;


echo "My name is $name. I am $age";
echo 'My name is $name. I am $age';
```

Какие выводы мы можем сделать?

Возникает проблема, особенно характерна для Английского языка, когда нужно явно указать в тексте строки где начинается и заканчивается имя переменной. Рассмотрим и протестируем следующий пример:

```php
$juice = "apple";

echo "He drank some juice made of $juices.";

echo '<br>';

// Решение
echo "He drank some juice made of {$juice}s.";
echo '<br>';
echo "He drank some juice made of ${juice}s.";
```

Этот приём называется выделение (экранирование) переменных в строке.

Задание №5

Так же в PHP существует альтернативный синтаксис строк, он используется довольно редко, но его можно иногда встретить. Его могут использовать, например, для того что бы вывести большие куски текста в консоль операционной системы при вызове php скрипта. Рассмотри и протестируем следующий код:

```php
$name = 'Lena';
$age = 89;

echo <<<HEREDOC
Эквивалент строки в двойных кавычках.
подставятся в строку, а сама строка развернётся

Значения переменных $name и $age

HEREDOC;

echo '<br>';

echo <<<'NOWDOC'
Эквивалент строки в одинарных кавычках.
Переменные $name и $age не развернутся.
И строки тоже.
NOWDOC;
```

Задание №6

Получить длину строки (кол-во символов) в PHP можно с помощью встроенной функции **strlen,** рассмотрим и протестируем пример кода:

```php
$string = 'simple string';

echo strlen($string);
```

Так же PHP позволяет получить произвольный доступ к символу в строке. Для этого используется следующий синтаксис:

```php
// получение первого символа в строке
echo $string{0};
```

Теперь самостоятельно напишите код, который в строке *$string* заменяет последний символ на «!».

Задание №7

В PHP встроено множество функций для работы со строками. В этом задании мы разберем некоторые из них.

number_format — форматирует число с разделением групп.

Подробнее http://php.net/manual/ru/function.number-format.php .

Рассмотрите и протестируйте пример кода:

```php
$million_dollars = 9999999;

// простой вариант разделения групп
echo number_format($million_dollars).' $';

echo '<br>';

// красивое разделение групп, очень удобное для вывода денежных сумм
echo number_format($million_dollars, 2 , '.', ' ').' $';
```

explode — Разбивает строку с помощью разделителя (указанного символа). Возвращает массив PHP.

Подробнее тут http://php.net/manual/ru/function.explode.php .

Рассмотрите и протестируйте пример кода:

```php
// берем строку, в которой находится информация о системе и браузере пользователя
$userAgent = $_SERVER['HTTP_USER_AGENT'];

echo '<pre>';

// делим строку по пробелу
$arr = explode(' ', $userAgent);
var_dump( $arr );
```

Обратная функция **implode** – она склеивает массив в строку:

```php
echo '<br>';
echo implode(' ', $arr);
```

trim – удаляет пробелы (или другие пробельные символы) из начала и конца строки.

Подробнее http://php.net/manual/ru/function.trim.php .

Рассмотрите и протестируйте пример кода:

```php
echo '<pre>';
var_dump( trim("\t\t string   ") );
```

str_replace — Заменяет кусок строки на другой.

Подробнее http://php.net/manual/ru/function.str-replace.php .

Рассмотрите и протестируйте пример кода:

```php
$str = "I love HTML !";
echo '<br>';
echo str_replace('HTML', 'PHP', $str);
```

Задание №8

Логический тип данных в PHP такой же как и в JavaScript. Логические операторы автоматически приводят свои операнды в логический тип (об особенностях приведения типов в PHP мы поговорим позже).

Логические операторы PHP:

Пример	Название	Результат
$a and $b	И	**TRUE** если и *$a*, и *$b* **TRUE**.
$a or $b	Или	**TRUE** если или *$a*, или *$b* **TRUE**.
$a xor $b	Исключающее или	**TRUE** если *$a*, или *$b* **TRUE**, но не оба.
!$a	Отрицание	**TRUE** если *$a* не **TRUE**.
$a && $b	И	**TRUE** если и *$a*, и *$b* **TRUE**.
$a || $b	Или	**TRUE** если или *$a*, или *$b* **TRUE**.

Обратите внимание что, логические операторы AND и OR в PHP имеют два варианта использования AND (&&) и OR (||). Оба варианта не одинаковы - они имеют разный приоритет выполнения (у AND и OR он ниже).

Приоритет выполнения операторов можно посмотреть на странице тут
http://php.net/manual/ru/language.operators.precedence.php

Рассмотрим и протестируем следующий код:

```php
$result = false || true;

$anomaly = false or true;

var_dump($result, $anomaly);
```

Почему мы получили такие результаты?

Операторы сравнения служат для сравнения значений, операторы сравнения в PHP представлены в таблице ниже:

Операторы сравнения:

Пример	Название	Результат
$a == $b	Равно	**TRUE** если *$a* равно *$b* после преобразования типов.
$a === $b	Тождественно равно	**TRUE** если *$a* равно *$b* и имеет тот же тип.
$a != $b	Не равно	**TRUE** если *$a* не равно *$b* после преобразования типов.
$a !== $b	Тождественно не равно	**TRUE** если *$a* не равно *$b* или в случае, если они разных типов
$a < $b	Меньше	**TRUE** если *$a* строго меньше *$b*.
$a > $b	Больше	**TRUE** если *$a* строго больше *$b*.
$a <= $b	Меньше или равно	**TRUE** если *$a* is меньше или равно *$b*.
$a >= $b	Больше или равно	**TRUE** если *$a* больше или равно *$b*.

Следует знать несколько особенностей сравнения в PHP.

Строки между собой сравниваются через механизм «лексического сравнения». PHP в этой роли используется алгоритм **Natural Order String Comparison** (http://sourcefrog.net/projects/natsort/). Рассмотрим пример сравнения строк:

```
// примеры сравнения строк

var_dump('xxx' == 'xxx');
var_dump('xXX' == 'xxx');
var_dump('xxx' === 'xxx');

var_dump('a' < 'aa');
var_dump('a1' < 'a0');
```

Если один из операндов число, то второй будет приведен к числу, протестируйте следующий пример:

```
// примеры числового сравнения

var_dump( 0 == 'string' );
var_dump( 0 === 'string' ); // сравнение типов

var_dump(1 > 'string');
var_dump(1 > '10');
```

Если один из операндов логическое значение, то второй будет приведен в логическое значение. Пример:

```php
// примеры логического сравнения

var_dump( false == 0 );
var_dump( false === 0 );


var_dump( true == 'string' );
```

Детальнее о сравнении массивов и объектов можно посмотреть тут
http://php.net/manual/ru/language.operators.comparison.php

Задание №9

Специальное значение NULL представляет собой переменную без значения. NULL - это единственно возможное значение типа null. Переменная считается null, если:

- ей была присвоена константа NULL.
- ей еще не было присвоено никакого значения.
- она была удалена с помощью unset().

Проверьте следующий код, посмотите с помощью var_dump какие значения у переменных получились.

```php
$a = NULL;

$b = 5;
unset($b);


$c;
```

Встроенная функция unset() удаляет переменную.

Задание №10

Перед тем как рассмотреть и протестировать как PHP преобразует типы данных, мы разберём несколько важных встроенных функций для работы с типами данных:

gettype($var) – возвращает строку с названием типа переменной $var. Рассмотрите и протестируйте следующий код:

```php
// Проверка типов
$str = "John";
$int = 10;
$bool = true;

// использование gettype
echo gettype($str);
echo gettype($int);
echo gettype($bool);
echo gettype($x);
```

Так же существуют отдельные встроенные функции для проверки отдельных типов:

```php
// функции проверки отдельных типов
echo is_string($str);
echo is_integer($int);
echo is_boolean($bool);
echo is_null($x);
```

isset($value) - определяет, была ли установлена переменная значением отличным от NULL. Фактически это проверка на существование значения.

empty($value) - проверяет, пуста ли переменная.

Результаты функций связанных с типами представлены в таблице ниже:

Выражение	gettype()	empty()	is_null()	isset()	boolean : if($x)
$x = "";	string	TRUE	FALSE	TRUE	FALSE
$x = null;	NULL	TRUE	TRUE	FALSE	FALSE
var $x;	NULL	TRUE	TRUE	FALSE	FALSE
$x не определена	NULL	TRUE	TRUE	FALSE	FALSE
$x = array();	array	TRUE	FALSE	TRUE	FALSE
$x = array('a', 'b');	array	FALSE	FALSE	TRUE	TRUE
$x = false;	boolean	TRUE	FALSE	TRUE	FALSE
$x = true;	boolean	FALSE	FALSE	TRUE	TRUE
$x = 1;	integer	FALSE	FALSE	TRUE	TRUE
$x = 42;	integer	FALSE	FALSE	TRUE	TRUE
$x = 0;	integer	TRUE	FALSE	TRUE	FALSE
$x = -1;	integer	FALSE	FALSE	TRUE	TRUE
$x = "1";	string	FALSE	FALSE	TRUE	TRUE
$x = "0";	string	TRUE	FALSE	TRUE	FALSE
$x = "-1";	string	FALSE	FALSE	TRUE	TRUE
$x = "php";	string	FALSE	FALSE	TRUE	TRUE
$x = "true";	string	FALSE	FALSE	TRUE	TRUE
$x = "false";	string	FALSE	FALSE	TRUE	TRUE

Задание №11

В PHP допускаются следующие приведения типов:

- **(int), (integer)** - приведение к integer
- **(bool), (boolean)** - приведение к boolean
- **(float), (double), (real)** - приведение к float
- **(string)** - приведение к string
- **(array)** - приведение к array
- **(object)** - приведение к object
- **(unset)** - приведение к NULL (PHP 5+)

В этом задании Вы должны самостоятельно скрипт, который будет в табличной форме выводить информацию о переменной, включая:

- Имя переменной
- Значение
- Текущий тип
- результат empty для переменной
- результат is_null для переменной
- результат isset для переменной
- вывести значения, которые мы получаем при приведении переменной к следующим типам: string, int, float, array, object, null, bool.

Задание №11

PHP позволяет очень гибко настроить оповещение об ошибках. В PHP все ошибки разбиты на различные уровни, каждому уровню соответствует определённая константа как идентификатор этого уровня. Сейчас мы разберём как некоторые из них.

E_PARSE – ошибки на этапе первичного анализа кода, например классическая ошибка – пропущенный разделитель инструкций «;». Введите и запустите данный код:

```
$a = 10
$b = 15;
```

Какое сообщение об ошибке мы получили? Что произошло с точки зрения PHP?

E_ERROR – фатальные ошибки, которые прекращают выполнение сценария PHP, пример такой ошибки – вызов неопределенной функции:

```
foo();
```

E_WARNING - предупреждения времени выполнения (не фатальные ошибки). Выполнение скрипта в таком случае не прекращается. Примером такой ошибки может служить попытка открыть несуществующий файл:

```
$file = fopen('index.txt');

echo 'I am alive!';
```

Главное, что Вы должны взять из этого примера это то, что выполнение программы после ошибки **E_WARNING** продолжается и это может привести к «каскаду ошибок». Если такое уже произошло, то нужно всегда смотреть на первую ошибку.

E_NOTICE – это что-то вроде извещение, PHP кажется, что Вы делаете что-то не так. Например, обращаетесь к не существующему элементу массива:

```
$array['spirit'];
```

Сейчас на экране Вы возможно не увидите предупреждение об этой ошибке, потому что очень часто это оповещение отключено.

E_DEPRECATED – ошибки, связанные с использование устаревших функций.

Полный список констант, описывающий уровни ошибок можно найти тут http://php.net/manual/ru/errorfunc.constants.php

Первоначально задать как будут отображаться ошибки можно в php.ini за это отвечают следующие директивы:

; on или off. Если стоит значение off то никакие ошибки не будут выводиться

```
display_errors = on
```

; показывать все ошибки кроме E_NOTICE

```
error_reporting = E_ALL & ~E_NOTICE
```

Но если у Вас на проекте нет доступа к настройкам php.ini , то не отчаивайтесь есть замечательные встроенные функции, которые могут «на лету» изменить настройки PHP:

```php
// разрешаем показывать ошибки
ini_set('display_errors','on');


// Включаем вывод всех ошибок
error_reporting(E_ALL);

// Отключаем вывод всех ошибок
error_reporting(0);

// Включаем определённые уровни ошибок
error_reporting(E_ERROR | E_WARNING);
error_reporting(E_ALL & ~E_DEPRECATED);
```

ini_set - устанавливает значения настройки конфигурации в скрипте. Следует помнить что далеко не все настройки можно менять подобным образом.

error_reporting – задаёт какие ошибки следует отображать.

Задание №12

Последнее на сегодня задание — это побитовые операции в PHP. Побитовые операторы позволяют считывать и устанавливать конкретные биты целых чисел. Побитовые операции сведены в таблицу ниже:

Пример	Название	Результат
$a & $b	И	Устанавливаются только те биты, которые установлены и в $a, и в $b.
$a | $b	Или	Устанавливаются те биты, которые установлены в $a или в $b.
$a ^ $b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в $a, либо только в $b, но не в обоих одновременно.
~ $a	Отрицание	Устанавливаются те биты, которые не установлены в $a, и наоборот.
$a << $b	Сдвиг влево	Все биты переменной $a сдвигаются на $b позиций влево (каждая позиция подразумевает "умножение на 2")
$a >> $b	Сдвиг вправо	Все биты переменной $a сдвигаются на $b позиций вправо (каждая позиция подразумевает "деление на 2")

Рассмотрим пример использования побитовых операций в PHP. Предположим, что нам нужно сгенерировать 4 бита данных, которые определяют модификаторы доступа к директории в следующем порядке: чтение, создание, изменение, удаление. Рассмотрим и протестируем пример кода, как это можно реализовать через побитовые операции:

```php
// Определяем константы для каждой опции

define('READ', 1 << 0);   // 0001
echo decbin(READ), '<br>';

define('CREATE', 1 << 1); // 0010
echo decbin(CREATE), '<br>';

define('EDIT', 1 << 2);   // 0100
echo decbin(EDIT), '<br>';

define('DELETE', 1 << 3); // 1000
echo decbin(DELETE), '<br>';


// опеределяем константу, в случае если всё включено
define('ALL', READ | CREATE | EDIT | DELETE); // 1111
echo decbin(ALL), '<br>';
```

Встроенная функция **decbin** переводит число из десятеричного в двоичный формат. Таким способом через созданные константы удобно управлять правами доступа. Чтобы комбинировать несколько прав используем побитовый оператор ИЛИ (|).

Другой пример использования побитовых операторов Вы уже встречали на этом уроке:

```php
$error = E_ALL & ~E_DEPRECATED;

error_reporting( $error );

echo decbin($error), '<br>';
```

На самом деле функция error_reporting принимает целое число, затем в двоичном формате, считывая биты, и по ним определяет какие настройки были ей переданы.

Домашнее задание

Ознакомитесь со встроенными функциями для обработки строк
http://php.net/manual/ru/ref.strings.php