

Лекция 2

Переменные в PHP.

Типы данных

Что такое переменная?

Это область оперативной памяти, доступ к которой осуществляется по имени.

- ✓ Все данные, с которыми работает программа, хранятся в виде переменных (исключение – константа, которая, впрочем, может содержать только число или строку).
- ✓ Такого понятия, как указатель (как в C), в PHP не существует. При присвоении переменная копируется один-в-один, какую бы сложную структуру она ни имела.
- ✓ В PHP существует понятие ссылок – жестких и символических.

Особенности использования переменных в PHP

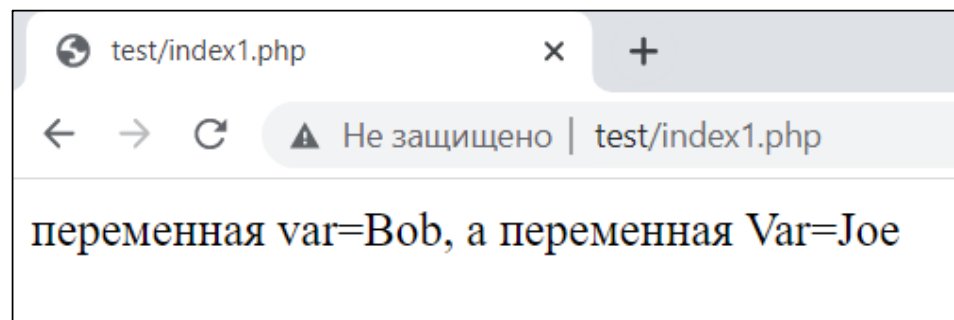
- Переменную не надо предварительно определять с заданием типа. Тип переменной определяет сам интерпретатор при вводе ее значения.
- Имена переменных обязательно начинаются со знака \$.
- В именах переменных можно использовать:
 - латинские символы,
 - любой символ с кодом ASCII >127.
- Имена переменных не могут начинаться с цифр.
- Имена переменных чувствительны к регистру.
- Имя переменной не может содержать пробелов.

Примеры использования имен переменных в PHP

```
<?php
$var = "Bob";
$Var = "Joe";
echo "переменная var=$var, а переменная Var=$Var"; // выведет "Bob, Joe"
?>
```

```
<?php
$4site = 'not yet'; // неверно; начинается с цифры
$_4site = 'not yet'; // верно; начинается с символа подчеркивания
$tдыte = 'mansikka'; // верно; 'д' это (Дополнительный) ASCII 228.
$this = 'hello'; // неверно; ссылка на текущий объект.
?>
```

Примеры использования имен переменных в PHP



Удаление переменной в PHP

```
<?php
```

```
$a=1000;
```

```
echo "a=$a", "<br>";
```

```
unset($a);
```

```
echo "a=$a"; //в данной строке будет ошибка, т.к. переменной $a уже не существует
```

```
?>
```



Что такое константа?

Константы в PHP - это «постоянные», значения которых указывается всего один раз и затем не может быть изменено. Примеры: параметры конфигурации, такие как имя пользователя и пароль базы данных, базовый URL-адрес веб-сайта, название компании и т.д.

2 способа задания константы:

```
// 1 define( 'NAME', 'VALUE' );  
// 2 const NAME = 'VALUE';      (с версии 5.3)
```

Константы могут хранить только скаляры.

Скалярные переменные - это переменные с типами integer, float, string и boolean.

Что такое константа?

PHP меньше 5.3

```
// скаляры
define( 'FOO', 10 );
define( 'FOO', 10.9 );
define( 'FOO', 'val' );
define( 'FOO', true );

// не скаляры
define( 'FOO', array(1) );
// константа не установиться и получим Warning
define( 'FOO', (object) array(1) );
// константа не установиться и получим Warning
```


Что такое константа?

PHP с версии 5.3

Появилось ключевое слово `const` и теперь константу можно определять еще и с помощью него.

Однако, в `const` нельзя указать переменную, функцию или какое-то выражение, а нужно передавать скаляр «напрямую»:

```
const FOO = 'val';           // нет ошибок
const FOO = $var;           // Parse error
const FOO = home_url();     // Parse error
const FOO = 5 + 10;         // Parse error
const FOO = 'foo'. 'bar';   // Parse error
```

Тогда как для `define()` таких ограничений нет:

```
define('FOO', 'val');       // нет ошибок
define('FOO', $var );       // нет ошибок
define('FOO', home_url() ); // нет ошибок
define('FOO', 5 + 10 );     // нет ошибок
define('FOO', 'foo'. 'bar' ); // нет ошибок
```

Что такое константа?

PHP с версии 5.6

Стало возможным указывать в значения const примитивные PHP выражения (выражения из скаляров):

```
const FOO = 1 + 2;  
const FOO = 'foo' . 'bar';
```

Стало возможным хранить массивы в константах:

```
const FOO = [1, 2, 3];           // работает  
define( 'FOO', [1, 2, 3] );      // не работает в  
PHP 5.6, работает в PHP 7.0
```

Разница между const и define

#1 const должны быть объявлены в верхней области

Потому что они определяются при компилировании скрипта. Это значит, что const нельзя использовать внутри функций / циклов / выражений if или try/catch блоков.

```
if ( 1 ) {  
    const NAME = 'VALUE'; // не работает  
}  
// но  
if ( 1 ) {  
    define('NAME', 'VALUE'); // работает  
}
```

#2 const всегда регистрозависима

В то время как define() позволяет создать регистро-независимые константы:

```
define( 'NAME', 'VALUE', true );  
echo NAME;           // VALUE  
echo name;           // VALUE
```

Разница между const и define

#3 const понимает только скаляры

const нельзя передать переменные, функции, выражения, а define() можно:

```
const FOO = $var;           // Parse error
const FOO = home_url();     // Parse error
define('FOO', $var );       // нет ошибок
define('FOO', home_url() ); // нет ошибок
```

С версии PHP 5.6 в const также можно указывать примитивные выражения, а не только скаляры.

#4 const может хранить массивы с версии PHP 5.6, а define с PHP 7.0

```
const FOO = [1, 2, 3];      // работает с PHP 5.6
define( 'FOO', [1, 2, 3] ); // работает с PHP 7.0
```

Типы данных в PHP

Четыре скалярных типа:

- boolean (двоичные данные);
- integer (целые числа);
- float (числа с плавающей точкой или 'double');
- string (строки).

Два смешанных типа:

- array (массивы);
- object (объекты).

Два специальных типа:

- resource (ресурсы);
- NULL («пустой» тип).

Существуют также несколько псевдотипов:

- mixed (смешанный);
- number (числовой);
- callback (обратного вызова);
- ITERABLE.

Приведение 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 и выше)

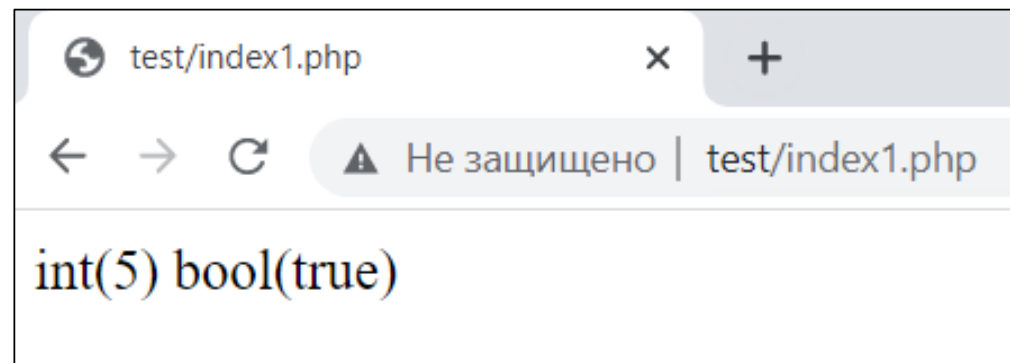
```
<?PHP
    $bar = "17";
    $foo = (int) $bar;
    $foo = ( int ) $bar;
?>
```

Тип boolean

```
<?php
    $x = True; // присвоить $x значение TRUE
?>
```

Функция `var_dump()` выводит в браузер структурированную информацию о переменной, а именно тип и значение. Например, выполнение следующего кода дает вот такой результат:

```
<?php
    $a=5;
    var_dump($a);
    $b=true;
    var_dump($b);
?>
```



test/index1.php

← → ↻ Не защищено | test/index1.php

int(5) bool(true)

Преобразование в тип boolean

При преобразовании в логический тип, следующие значения рассматриваются как **FALSE**:

- Сам булев FALSE
- целое 0 (ноль)
- число с плавающей точкой 0.0 (ноль)
- пустая строка и строка "0"
- массив с нулевыми элементами
- объект с нулевыми переменными-членами
- специальный тип NULL (включая неустановленные переменные)

Все остальные значения рассматриваются как **TRUE** (включая любой ресурс).

Внимание! -1 считается TRUE, как и любое ненулевое (отрицательное или положительное) число!

Преобразование в тип boolean

```
<?php
var_dump((bool) "");           // bool(false)
var_dump((bool) 1);            // bool(true)
var_dump((bool) -2);           // bool(true)
var_dump((bool) "foo");        // bool(true)
var_dump((bool) 2.3e5);         // bool(true)
var_dump((bool) array(12));     // bool(true)
var_dump((bool) array());       // bool(false)
var_dump((bool) "false");      // bool(true)
?>
```

Тип integer (целые числа)

Целое это число из множества $Z = \{..., -2, -1, 0, 1, 2, ...\}$, обычно длиной 32 бита (от $-2\ 147\ 483\ 648$ до $2\ 147\ 483\ 647$).

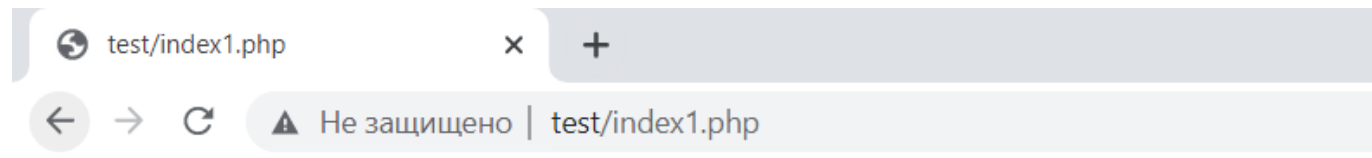
Целые могут быть указаны в десятичной, шестнадцатеричной, восьмеричной и двоичной системе счисления, по желанию с предшествующим знаком (- или +).

PHP не поддерживает беззнаковые целые.

```
<?php
$dec=17;          // десятичное число
$bin=0b10001;     // двоичное число
$oct=021;         // восьмеричное число    0o или 0O с PHP 8.1.0
$hex=0x11;        // шестнадцатеричное число

echo "переменная dec, введенная в десятичной системе счисления =$dec", "<br>";
echo "переменная dec, введенная в двоичной системе счисления =$bin", "<br>";
echo "переменная dec, введенная в восьмеричной системе счисления =$oct", "<br>";
echo "переменная dec, введенная в шестнадцатеричной системе счисления =$hex", "<br>";
?>
```

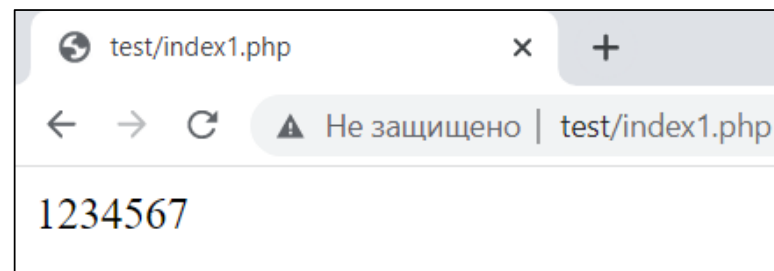
Тип integer (целые числа)



переменная dec, введенная в десятичной системе счисления =17
переменная dec, введенная в двоичной системе счисления =17
переменная dec, введенная в восьмеричной системе счисления =17
переменная dec, введенная в шестнадцатеричной системе счисления =17

Начиная с PHP 7.4.0, целочисленные литералы могут содержать подчёркивания (_) между цифрами для лучшей читаемости литералов. Эти подчёркивания удаляются сканером PHP.

```
<?php
$a=1_234_567;
echo $a;
?>
```



Превышение размера целого

Переполнение целых на 32-битных системах

```
<?php
$large_number = 2147483647;
var_dump($large_number); // вывод: int(2147483647)

$large_number = 2147483648;
var_dump($large_number); // вывод: float(2147483648)

// это справедливо и для шестнадцатеричных целых:
var_dump( 0x80000000 ); // вывод: float(2147483648)

$million = 1000000;
$large_number = 50000 * $million;
var_dump($large_number); // вывод: float(500000000000)
?>
```

Превышение размера целого

Переполнение целых на 64-битных системах

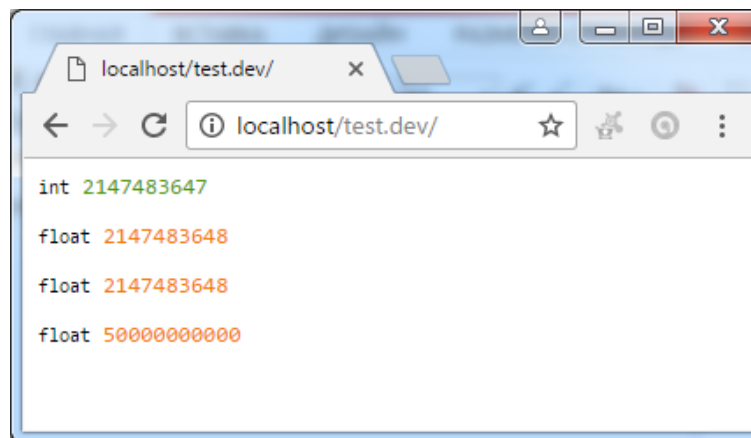
```
<?php
$large_number = 9223372036854775807;
var_dump($large_number); // вывод: int(9223372036854775807)

$large_number = 9223372036854775808;
var_dump($large_number); // вывод: float(9.223372036854776E+18)

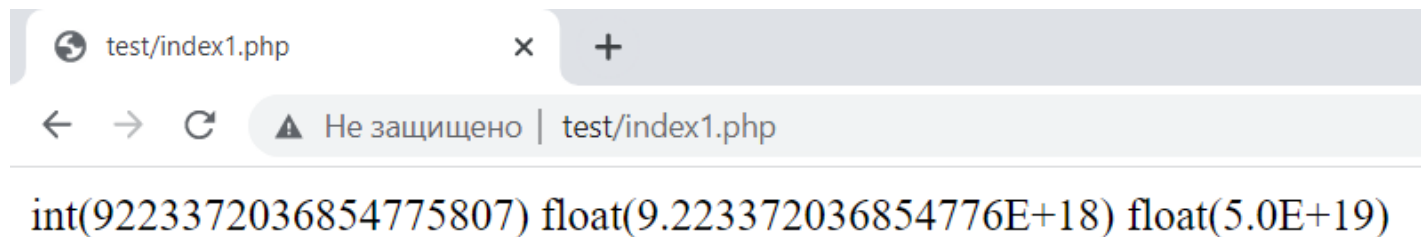
$million = 1000000;
$large_number = 5000000000000000 * $million;
var_dump($large_number); // вывод: float(5.0E+19)
?>
```

Превышение размера целого

Переполнение целых на 32-битных системах



Переполнение целых на 64-битных системах

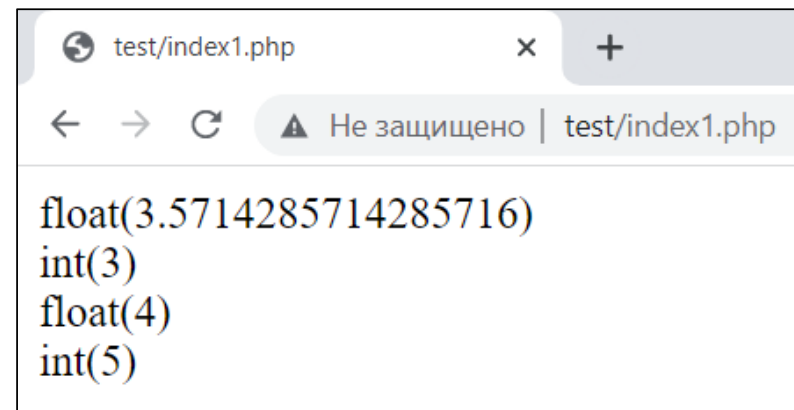


Деление целых чисел и округление до целого

В PHP, начиная с версии 7, при делении целых чисел возможен результат типа как `int`, так и `float`. Результатом $1/2$ будет число с плавающей точкой 0.5. Вы можете привести значение к целому, что всегда округляет его в меньшую сторону, либо использовать функцию `round()`, округляющую по традиционным правилам математики. Однако, если в результате деления двух чисел предполагается целое число, например, $25/5=5$, то тип данных, действительно получится `integer`.

Деление целых чисел и округление до целого

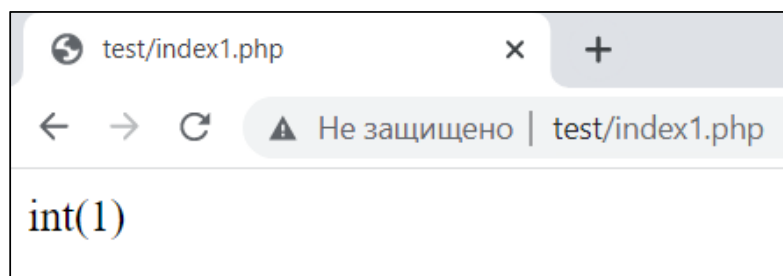
```
<?php
    $a=25/7;
    var_dump($a); // float(3.5714285714286)
?>
<br>
<?php
    $b=(int) $a;
    var_dump($b); // int(3)
?>
<br>
<?php
    $c=round($a);
    var_dump($c); //float (4)
?>
<br>
<?php
    $a=25/5;
    var_dump($a); // int(5)
?>
```



Деление целых чисел и округление до целого

Для однозначного преобразования значения в целое используйте приведение типа (int) или (integer). Однако в большинстве случаев вам нет необходимости использовать приведение типа, поскольку значение будет автоматически преобразовано, если оператор, функция или управляющая конструкция требует целый аргумент. Вы также можете преобразовать значение в целое при помощи функции `intval()`.

```
<?php
    $a=1.55;
    $b=intval($a); //преобразование типа float в integer
    var_dump($b); // b будет равно 1
?>
```

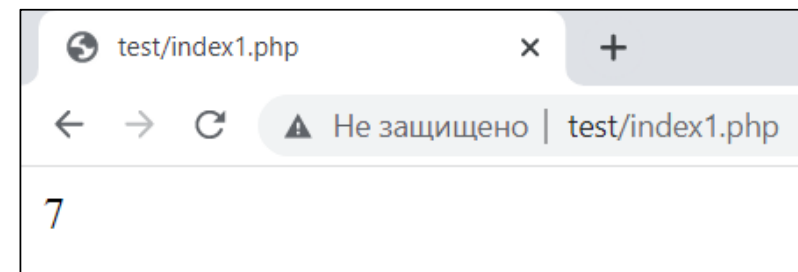


Преобразование других типов данных в целое (integer)

Если число с плавающей точкой превышает пределы целого (как правило, это $\pm 2.15 \times 10^9 = 2^{31}$), результат будет неопределенным, так как целое не имеет достаточной точности, чтобы вернуть верный результат. В этом случае не будет выведено ни предупреждения, ни даже замечания!

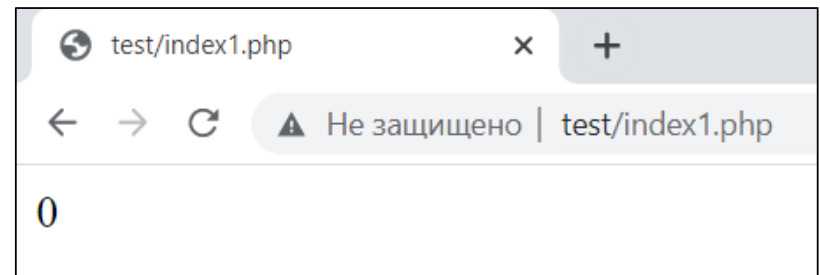
Внимание! Никогда не приводите неизвестную дробь к целому, так как это может иногда дать неожиданные результаты. Пример ошибочного преобразования:

```
<?php
echo (int) ( (0.1+0.7) * 10 );
// выводит 7! (ошибка!)
?>
```



Преобразование других типов данных в целое (integer)

```
<?php
$a=false;
$b=intval($a);//преобразов. типа boolean
в integer. Результат будет равен 0
echo $b;
?>
```



Тип float (числа с плавающей точкой)

Тип float (числа с плавающей точкой). Вещественное число довольно большой точности (должно хватить для подавляющего большинства математических вычислений). Числа с плавающей точкой (они же числа двойной точности или действительные числа) могут быть определены при помощи любого из следующих синтаксисов.

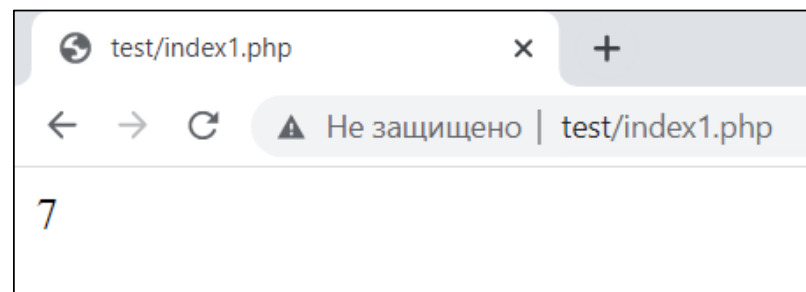
```
<?php
    $a = 1.234;
    $b = 1.2e3;
    $c = 7E-10;
?>
```

Размер переменной типа float зависит от платформы, хотя максимум, как правило, $\sim 1.8e308$ с точностью около 14 десятичных цифр (это 64-битный IEEE-формат).

Точность числа с плавающей точкой

```
<?php
$a=floor((0.1+0.7)*10);
echo $a; // b будет равно 7
?>
```

floor – округляет дробь в меньшую сторону.

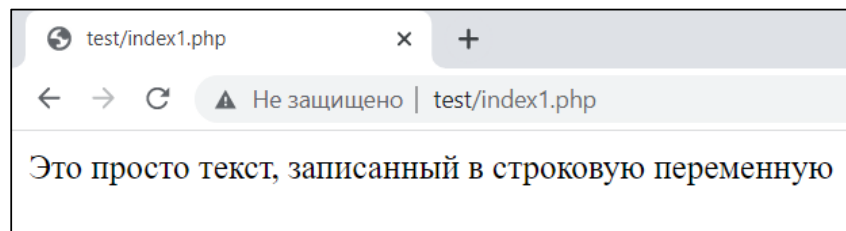


Тип string (строки)

Строка может быть определена следующими способами:

- одинарными кавычками;
- двойными кавычками;
- heredoc-синтаксисом;
- nowdoc-синтаксисом.

```
<?php
$a = "Это просто текст, записанный в строковую переменную";
echo $a; //Выводит 'Это просто текст, записанный в строковую переменную'
?>
```



Тип string (строки)

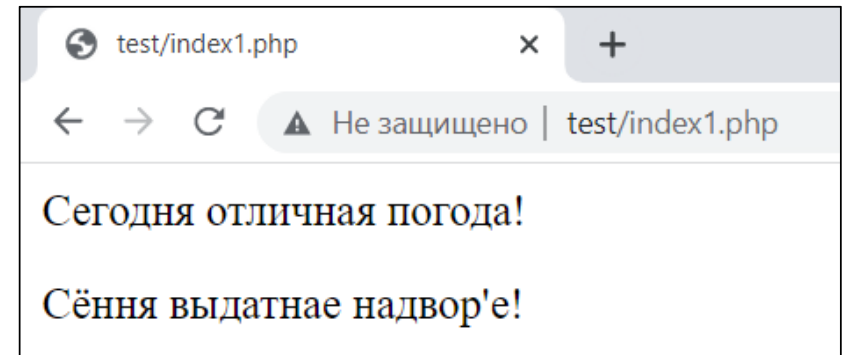
Простейший способ определить строку – это заключить ее в одинарные кавычки (символ `'`).

Чтобы использовать одинарную кавычку внутри строки, как и во многих других языках, ее необходимо предварить символом обратной косой черты (`\`), т. е. экранировать ее. Если обратная косая черта должна идти перед одинарной кавычкой либо быть в конце строки, вам необходимо продублировать ее. Обратите внимание, что если вы попытаетесь экранировать любой другой символ, обратная косая черта также будет напечатана! Так что, как правило, нет необходимости экранировать саму обратную косую черту.

В отличие от двух других синтаксисов, переменные и экранирующие последовательности для специальных символов, встречающиеся в строках, заключенных в одинарные кавычки, не обрабатываются.

Тип string (строки)

```
<?php
    echo '<p>';
    echo 'Сегодня отличная погода!';
    echo '</p>';
    echo '<p>';
    echo 'Сёння выдатнае надвор\'е!';
    echo '</p>';
?>
```



Определение строк двойными кавычками

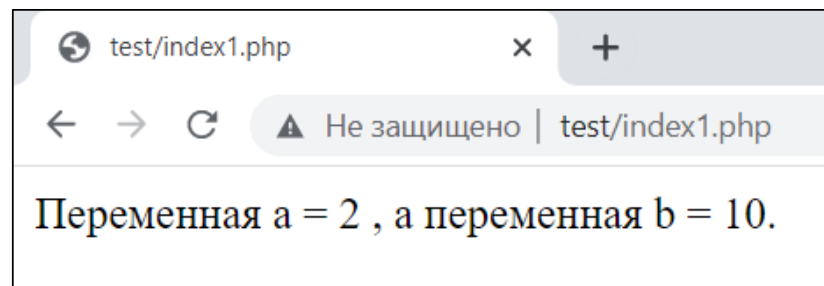
Если строка заключена в двойные кавычки ("), PHP распознает большее количество управляющих последовательностей для специальных символов

Последовательность	Значение
<code>\n</code>	новая строка (LF или 0x0A (10) в ASCII)
<code>\r</code>	возврат каретки (CR или 0x0D (13) в ASCII)
<code>\t</code>	горизонтальная табуляция (HT или 0x09 (9) в ASCII)
<code>\\</code>	обратная косая черта
<code>\\$</code>	знак доллара
<code>\"</code>	двойная кавычка
<code>\[0-7]{1,3}</code>	последовательность символов, соответствующая регулярному выражению, символ в восьмеричной системе счисления
<code>\x[0-9A-Fa-f]{1,2}</code>	последовательность символов, соответствующая регулярному выражению, символ в шестнадцатеричной системе счисления

Определение строк двойными кавычками

Самым важным свойством строк в двойных кавычках является обработка переменных. Например, данную особенность можно использовать при выводе значений переменных и т.д.

```
<?php
    $a=2;
    $b=10;
    echo "Переменная a = $a , а переменная b = $b.";
    //Выводит значения a и b, встроенные в текст
?>
```



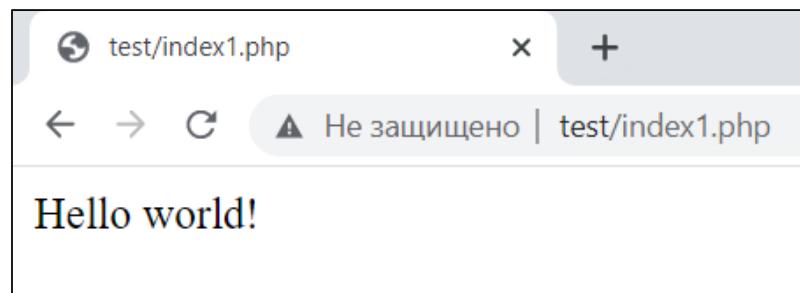
Определение строк двойными кавычками

Еще один способ определения строк – это использование heredoc-синтаксиса ("<<<").

После <<< необходимо указать идентификатор, затем идет строка, а потом этот же идентификатор, закрывающий вставку.

Закрывающий идентификатор должен начинаться в первом столбце строки. Кроме того, идентификатор должен соответствовать тем же правилам именования, что и все остальные метки в PHP: содержать только буквенно-цифровые символы и знак подчеркивания, и должен начинаться с не цифры или знака подчеркивания.

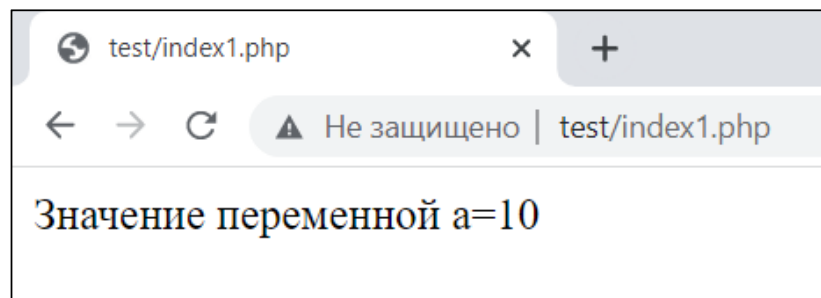
```
<?php
echo <<<TEXT
Hello world!
TEXT;
?>
```



Определение строк двойными кавычками

Heredoc-текст ведет себя так же, как и строка в двойных кавычках, при этом их не имея. Это означает, что нет необходимости экранировать кавычки в heredoc, но по-прежнему можно использовать вышеперечисленные управляющие последовательности. Переменные обрабатываются, но с применением сложных переменных внутри heredoc нужно быть также внимательным, как и при работе со строками.

```
<?php
$a=10;
echo <<<TEXT
Значение переменной a=$a
TEXT;
?>
```



Определение строк одинарными кавычками

Nowdoc – это то же самое для строк в одинарных кавычках, что и *heredoc* для строк в двойных кавычках. *Nowdoc* похож на *heredoc*, но внутри него *не осуществляется никаких подстановок*. Эта конструкция идеальна для внедрения PHP-кода или других больших блоков текста без необходимости его экранирования.

Nowdoc указывается той же последовательностью <<<, что используется в *heredoc*, но последующий за ней идентификатор заключается в одинарные кавычки, например, <<<'EOT'. Все условия, действующие для идентификаторов *heredoc*, также действительны и для *nowdoc*, особенно те, что относятся к закрывающему идентификатору.

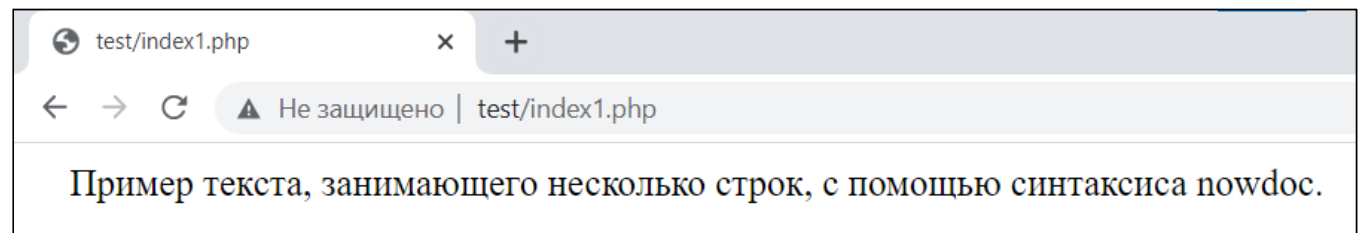
```
<?php
```

```
echo <<<'EOD'
```

```
    Пример текста,  
    занимающего несколько строк,  
    с помощью синтаксиса nowdoc.
```

```
EOD;
```

```
?>
```

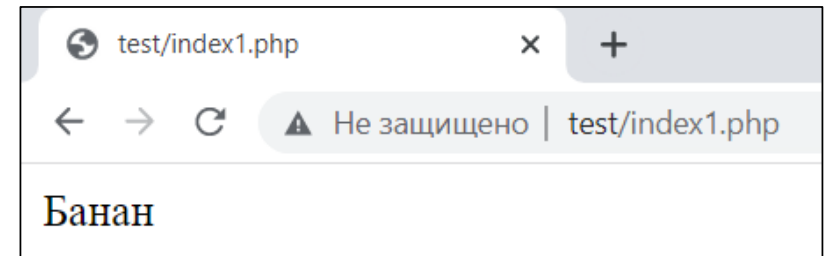


Тип array (массивы)

Массив в PHP – это упорядоченный набор данных, в котором установлено соответствие между значением и ключом. Индекс (ключ) служит для однозначной идентификации элемента внутри массива. В одном массиве не может быть двух элементов с одинаковыми индексами.

Простой массив (список)

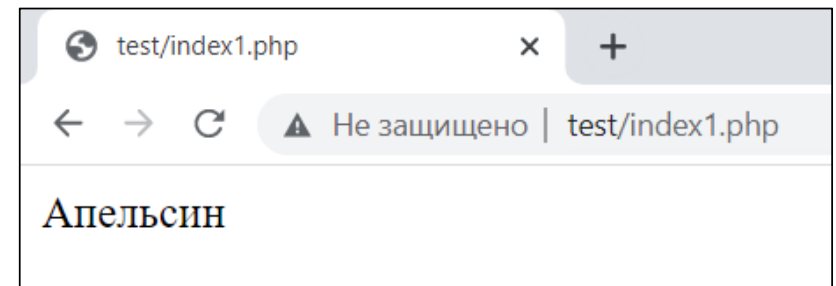
```
<?php
// Простой способ инициализации массива
$names[0]="Апельсин";
$names[1]="Банан";
$names[2]="Груша";
$names[3]="Помидор";
// Здесь: names - имя массива, а 0, 1, 2, 3
- индексы массива
echo $names[1]; // Выведет на экран Банан
?>
```



Тип array (массивы)

Простой массив (список)

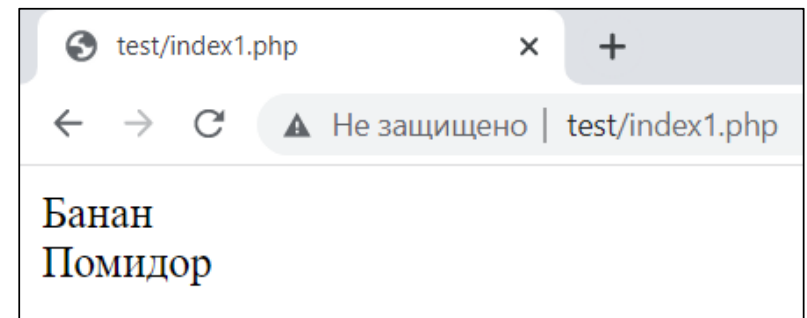
```
<?php
// Простой способ инициализации массива
$names[1]="Апельсин";
$names[2]="Банан";
$names[3]="Груша";
$names[4]="Помидор";
// Здесь: names - имя массива, а 0, 1, 2, 3
- индексы массива
echo $names[1]; // Выведет на экран Апельсин
?>
```



Тип array (массивы)

Простой массив (список)

```
<?php
// Простой способ инициализации массива
$names []="Апельсин";
$names []="Банан";
$names []="Груша";
$names []="Помидор";
// Здесь: names - имя массива, а 0, 1, 2, 3
- индексы массива
echo $names[1]; // Выведет на экран Банан
echo "<br>";
echo $names[3]; // Выведет на экран Помидор
?>
```



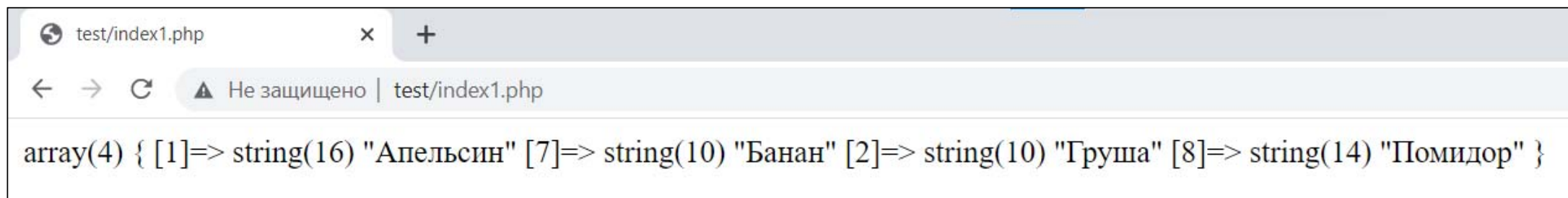
Тип array (массивы)

Простой массив (список)

```
<?php
// Простой способ инициализации массива
$names[1]="Апельсин";
$names[7]="Банан";
$names[2]="Груша";
$names[?]="Помидор";
// Здесь: names - имя массива, а 0, 1, 2, 3 - индексы массива
echo $names[1]; // Выведет на экран Апельсин
echo "<br>";
echo $names[?];
?>
```

Тип array (массивы)

Простой массив (список)



```
test/index1.php x +  
← → ↻ ⚠ Не защищено | test/index1.php  
array(4) { [1]=> string(16) "Апельсин" [7]=> string(10) "Банан" [2]=> string(10) "Груша" [8]=> string(14) "Помидор" }
```

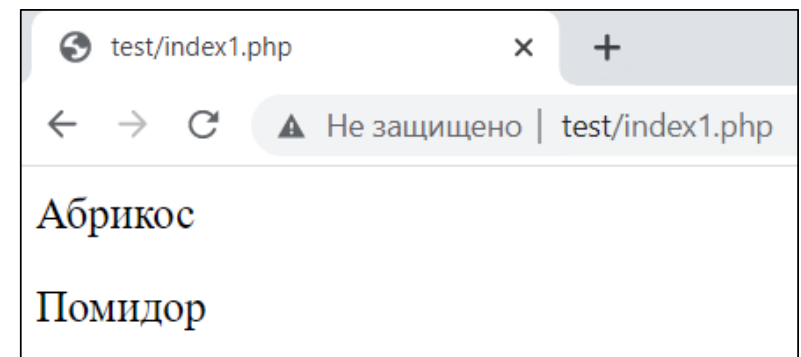
```
<?php  
$names = array(1=>"Апельсин", 2=>"Банан", 3=>"Груша", 4=>"Помидор");  
echo $names[1]; // Выведет на экран Апельсин  
?>
```

Тип array (массивы)

Простые многомерные массивы

\$имя [индекс1] [индекс2] .. [индексN] ;

```
<?php
// Многомерный простой массив:
$arr[0][0]="Овощи";
$arr[0][1]="Фрукты";
$arr[1][0]="Абрикос";
$arr[1][1]="Апельсин";
$arr[1][2]="Банан";
$arr[2][0]="Огурец";
$arr[2][1]="Помидор";
$arr[2][2]="Тыква";
// Выводим некоторые элементы массива:
echo "<h3>".$arr[1][0].":</h3>";
echo "<h3>".$arr[2][1].":</h3>";
?>
```

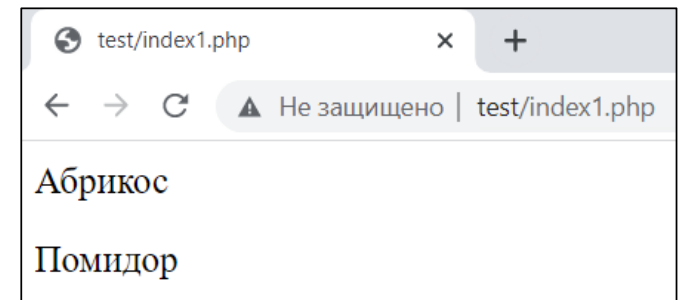


Тип array (массивы)

Простые многомерные массивы

\$имя [индекс1] [индекс2] .. [индексN] ;

```
<?php
// Многомерный простой массив:
$arr = array(
    array("Овощи", "Фрукты"),
    array("Абрикос", "Апельсин", "Банан"),
    array("Огурец", "Помидор", "Тыква"),
);
echo "<h3>".$arr[1][0]."</h3>";
echo "<h3>".$arr[2][1]."</h3>";
?>
```



Тип array (массивы)

Многомерные ассоциативные массивы

Многомерные ассоциативные массивы могут содержать несколько ключей, соответствующих конкретному индексу ассоциативного массива. Рассмотрим пример многомерного ассоциативного массива

	name	age	email
Ivanov	Иванов И.И.	25	ivanov@mail.ru
Petrov	Петров П.П.	34	petrov@mail.ru
Sidorov	Сидоров С.С.	47	sidorov@mail.ru

Тип array (массивы)

Многомерные ассоциативные массивы

```
<?php
// Многомерный массив
$A["Ivanov"] = array("name"=>"Иванов И.И.", "age"=>"25",
"email"=>"ivanov@mail.ru");
$A["Petrov"] = array("name"=>"Петров П.П.", "age"=>"34",
"email"=>"petrov@mail.ru");
$A["Sidorov"] = array("name"=>"Сидоров С.С.", "age"=>"47",
"email"=>"sidorov@mail.ru");

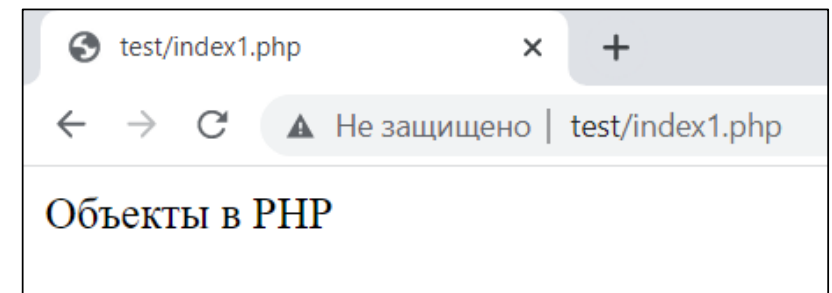
echo $A["Petrov"]["name"]; // Выводит Петров П.П.
echo "<br>";
echo $A["Petrov"]["age"]; // Выводит возраст
echo "<br>";
echo $A["Petrov"]["email"]; // Выводит e-mail
?>
```

Тип object (объекты)

Объект является одним из базовых понятий объектно-ориентированного программирования. Внутренняя структура объекта похожа на хэш, за исключением того, что для доступа к отдельным элементам и функциям используется оператор `→`, а не квадратные скобки.

Для инициализации объекта используется выражение *new*, создающее в переменной экземпляр объекта.

```
<?php
class ob
{
    function do_ob()
    {
        echo «Объекты в PHP»;
    }
}
$n_ob = new ob;
$n_ob->do_ob();
?>
```



Псевдотип RESOURCE

Resource - это специальная переменная, содержащая ссылку на внешний ресурс. Ресурсы создаются и используются специальными функциями.

Поскольку тип **resource** содержит специальные указатели на открытые файлы, соединения с базой данных, области изображения и тому подобное, преобразование в этот тип не имеет смысла.

Благодаря системе подсчета ссылок, введенной в Zend Engine, определение отсутствия ссылок на ресурс происходит автоматически, после чего он освобождается сборщиком мусора. Поэтому очень редко требуется освобождать память вручную.

Тип NULL (пустой тип)

Специальное значение NULL говорит о том, что эта переменная не имеет значения. NULL – это единственно возможное значение типа NULL (пустой тип).

Переменная считается NULL если:

- ей была присвоена константа NULL;
- ей еще не было присвоено какое-либо значение;
- она была удалена с помощью *unset()*.

Псевдотип ITERABLE

Итерируемым является любое значение, которое можно перебрать в цикле `foreach()`. Псевдотип `iterable` был введен в PHP 7.1, и он может быть использован в качестве типа данных для аргументов функции или в качестве возвращаемого типа функции. Если значение не является массивом или экземпляром `Traversable`, будет выдана ошибка `TypeError`.

Mixed, Number, Callback-функции

mixed говорит о том, что параметр может принимать много (но не обязательно все) типов.

Например, функция `gettype()` принимает все типы PHP, тогда как `str_replace()` принимает только типы `string` и `array`.



number говорит о том, что параметр может быть либо `integer`, либо `float`.



Callback-функции могут быть отмечены в коде подсказкой типа `callable`, появившейся в версии PHP 5.4. Некоторые функции, такие как `call_user_func()` или `usort()`, принимают определенные пользователем callback-функции в качестве параметра. Callback-функции могут быть как простыми функциями, так и методами объектов, включая статические методы классов.

Вопросы:

1. Правила именования переменных?
2. Способы задания константы?
3. Особенности использования констант?
4. Типы данных в PHP?
5. Что делает функция `var_dump()`?
6. Преобразование в тип `boolean`. Какие значения рассматриваются как **FALSE**?
7. Способы задания строк в PHP?
8. Отличия этих способов?
9. Какие виды массивов в PHP узнали?
10. Когда переменные считаются `NULL`?