Основы работы с PHP

*В этом уроке мы разберем работу с **данными** на языке PHP.*

***Данные** - это объекты, которыми может оперировать PHP, к примеру, сообщение пользователя - это строка, а его возраст - это число. Данные могут быть разных **типов**.*

Строки и числа

Самыми простыми **типами данных** в PHP являются **строки** и **числа**.

Числа обозначают сами себя, например 1, 12, 145, а вот строки требуется брать в **кавычки**(одинарные или двойные - без разницы):

```
<?php
        'строка', "строка"; //это примеры строк
?>
```

Числа также могут быть **дробями**. Разделителем целой и дробной части служит знак "точка". К примеру, число **10.754** - это дробь.

Переменные

Одним из самых главных и распространенных объектов в программировании является *переменная*.

Переменная - это такой объект, который может хранить внутри себя различные **данные**, например, строки или числа.

Имя переменной должно состоять из английских букв: больших или маленьких, а также цифр и знака_подчеркивания.

В PHP перед переменной обязательно должен быть написан знак доллара **$**:

```
<?php
        $a, $a1, $isVar, $is_var; //это примеры переменных
?>
```

Операция присваивания

Очень важным элементом программирования является **операция присваивания**.

С помощью присваивания мы можем положить в переменную результат какой-либо операции, например, строку или число, а затем достать эти данные в нужном нам месте и воспользоваться ими.

Пример присваивания:

```
<?php
        $a = 4; //мы присвоили переменной $a значение 4
?>
```

Комментарии

К коде PHP, так же, как и в HTML и CSS, можно оставлять **комментарии**, которые будут проигнорированы PHP при выполнении программы.

Их обычно используют для того, чтобы оставлять какие-либо пометки в коде или временно убрать какие-то строки кода, которые вам мешают, а потом иметь возможность быстро их вернуть обратно.

Комментарии могут быть **многострочными** и **однострочными**:

```
<?php
        $a = 4; //это пример однострочного комментария
?>

<?php
        /*
                Это пример
                многострочного комментария.
        */
        $a = 4;
 ?>
```

Функции

В программировании есть такое понятие, как **функции**. В отличии от переменных, которые хранят внутри себя некоторые данные, функции **выполняют операции** над этими данными.

Функция отличается от переменной тем, что перед именем функции не ставится знак доллара (он нужен только для переменных), а также тем, что у нее после названия стоят круглые скобки **( )**, внутри которых через запятую можно перечислять параметры.

Параметры - это данные (например, строки или числа), которые мы даем функции, она производит над ними какие-то операции и возвращает нам результат.

Например, в PHP есть стандартная функция **sqrt()**, которая параметром принимает число, а возвращает квадратный корень этого числа.

То есть, если ей передать число **9**, вот так - **sqrt(9)**, то на выходе получим число **3**. Полученное число мы можем записать в какую-либо переменную или вывести на экран.

Давайте найдем корень из **25** и запишем в переменную **$a**:

```
<?php
        $a = sqrt(25); //в переменную $a запишется значение 5 (так как 25 - это 5 в квадрате)
?>
```

Функция echo

В PHP существует специальная функция **echo**, которая позволяет **вывести** какой-либо **текст** в окно браузера. В отличие от других функций, после echo не требуются круглые скобки ().

Следующий код выводит на экран заданный текст:

```
<?php
        echo 'Привет, мир!'; //выведет на экран фразу 'Привет, мир!'
?>
```

А в следующим коде переменной **$text** присваивается фраза, а затем содержимое этой переменной выводится на экран:

```
<?php
        $text = 'Привет, мир!';
        echo $text; //выведет на экран фразу 'Привет, мир!'
?>
```

Математические операции

В PHP между числами и переменными можно совершать различные **математические операции**: сложение (знак +), вычитание (знак -), умножение (знак *), деление (знак /).

Чтобы понять, как с ними работать, посмотрите на следующий пример:

```
<?php
        echo 2 + 3; //выведет 5
        echo 5 - 1; //выведет 4
        echo 2 * 3; //выведет 6
        echo 6 / 2; //выведет 3
?>
```

Можно сначала записать числа в переменные, а потом уже проводить операции с переменными:

```
<?php
        $a = 3; //в $a запишем число 3
        $b = 7; //в $b запишем число 7
        echo $a + $b; //выведет на экран число 10, результат сложения 3 и 7
?>
```

А можно также одновременно производить операции и с числами, и с переменными:

```
<?php
        $a = 3; //в $a запишем число 3
        echo $a + 7; //выведет на экран число 10, результат сложения 3 и 7
?>
```

Специальные значения

В PHP, как и в других языках программирования, существуют ключевые слова для некоторых специальных значений.

Значения true, false

Значения **true** и **false** обозначают истину и ложь соответственно. Они используются для таких вещей, которые предполагают два варианта ответа - да или нет.

К примеру, на вопрос 'вам уже есть 18 лет?' в можете ответить да, то есть **true**, или нет, то есть **false**.

Эти значения называются *логическими*.

Значение null

Значение **null** обозначает 'ничего'. К примеру, если мы попробуем обратиться к переменной, которой мы еще не задали значение - то ее значение и будет **null**.

Функция var_dump

Бывают ситуации, когда нам нужно посмотреть **содержимое переменной**, чтобы понять, почему наш код работает не так, как мы хотели. Обычно на помощь приходит функция **echo**, однако, бывают ситуации, когда echo не работает.

Например, если переменная является **пустой строкой ''**, то выведя ее содержимое на экран с помощью **echo**, мы ничего не увидим.

Вторая ситуация - когда переменная содержит **строку с пробелами по краям**, например:

```
<?php
        $a = '   строка  ';
        echo $a; //выведет 'строка' - без пробелов по краям
?>
```

Для решения описанных выше проблем следует использовать функцию **var_dump()**, которая предназначена для **отладки кода** и позволяет выводить значения переменных и их тип.

С помощью этой функции мы можем увидеть, что переменная пуста, можем подсчитать количество пробелов по краям строки, можем увидеть, что переменная логическая и имеет значение true или false (об этих значениях чуть позже).

Пример использования **var_dump()**:

```
<?php
        $a = '   строка  ';
        var_dump($a);
?>
```

PHP внутри HTML

PHP код можно вставлять **внутрь HTML кода**:

```
<p> <?php echo $text; ?> </p>
```

Сложение строк в PHP

В PHP, в отличие от многих других языков, строки складываются с помощью знака **"точка"**:

```
<?php
        $a = 'строка1';
        $b = 'строка2';
        $c = $a.$b; //в переменной $c будет лежать 'строка1строка2'
?>
```

Получение определенного символа строки

В PHP можно получить доступ к определенному символу строки по его номеру таким образом: **$a[n]** – n-ный символ строки (учтите, что **нумерация идет с нуля**):

```
<?php
        $a = 'abcde'; //в переменной $a будет хранится значение 'abcde'
        $b = $a[0]; //в переменной $b будет 'a'
        $b = $a[1]; //в переменной $b будет 'b'
        $b = $a[4]; //в переменной $b будет 'e'
?>
```

Сложности с операцией присваивания

Очень часто новички не понимают, что **присваивание отличается от обычного равенства**. Посмотрите следующий пример:

```
<?php
        $a = 1;
        $a = $a + 2;
?>
```

С точки зрения математики запись **$a = $a + 2** абсурдна, но не с точки зрения программирования.

В данном случае переменная **$a** имела значение **1**, а затем мы переменной **$a** присвоили новое значение - старое значение переменной **$a** плюс **2**.

Операции инкремента и декремента

Операция **$a++** или **++$a** – увеличивает переменную **$a** на единицу. Эта операция называется **инкремент**.

Операция **$a--** или **--$a** – уменьшает переменную **$a** на единицу. Эта операция называется **декремент**.

Примеры:

```
<?php
        $a = 1;
        $a++; //увеличит $a на 1, что соответствует коду $a = $a + 1;
        echo $a; //выведет 2

        $a = 1;
        $a--; //уменьшит $a на 1, что соответствует коду $a = $a - 1;
        echo $a; //выведет 0
?>
```

Давайте посмотрим, в каких случаях проявляется разница между **++a** и **a++**.

Пусть у нас есть код **echo ++a** и код **echo a++**.

В первом случае переменная сначала увеличится на единицу, а потом выведется, а во втором случае - сначала выведется, а потом увеличится.

Операции +=, -=, *=, /=

Мы уже рассматривали код, который демонстрирует сложности с операцией присваивания:

```
<?php
        $a = 2;
        $a = $a + 3;
?>
```

В данном случае мы присваиваем переменной **$a** ее текущее значение, увеличенное на 2. Однако PHP позволяет записать этот код еще короче с помощью оператора **+=**:

```
<?php
        $a = 1;
        $a += 3; //этот код полностью эквивалентен коду $a = $a + 3;
?>
```

Кроме того, существуют операторы -=, *=, /=, которые эквивалентны следующему коду:

```
<?php
        $a = 2;
        $a -= 3; //этот код полностью эквивалентен коду $a = $a - 3;

        $a = 2;
        $a *= 3; //этот код полностью эквивалентен коду $a = $a * 3;

        $a = 2;
```

```
        $a /= 3; //этот код полностью эквивалентен коду $a = $a / 3;
?>
```

Операция .=

Существует также операция **.=**:

```
<?php
        $str = 'abc';
        $str .= 'def'; //этот код полностью эквивалентен коду $str = $str . 'def';
?>
```

Типизация переменных

Что будет, если попробовать сложить, к примеру, число и строку, вот так: **3+'3'**? В результате вы получите число **6**. Это значит, что PHP автоматически осуществляет преобразование типов при необходимости, вам не нужно за это переживать.

То же самое случится, если, к примеру, сделать так: **3+true** или **3+false**. Попробуйте эти примеры сами и узнайте к чему преобразуется **true** в этом случае, а чему - **false**.

Задачи на основы языка PHP

Задача

Задача. Создайте переменную **$var** и присвойте ей значение 'hello'. Обращаясь к отдельным символам этой строки выведите на экран символ 'h', символ 'e', символ 'o'.

Решение: будем обращаться к отдельным символам этой строки, к примеру, буква 'h' имеет номер 0 и ее можно вывести так - **$var[0]**, буква 'e' имеет номер 1 и так далее:

```
<?php
        $var = 'abcde';
        echo $var[0]; //выведем букву 'h'
        echo $var[1]; //выведем букву 'e'
        echo $var[4]; //выведем букву 'o'
?>
```

Задача

Задача. Напишите скрипт, который считает **количество секунд** в часе.

Решение: так как в минуте **60** секунд, а в часе - **60** минут, то умножив **60** на **60** мы получим **количество секунд в часе**:

```
<?php
        echo 60 * 60;
?>
```

Если нам нужно получить количество секунд в **сутках**, то умножим еще и на **24** (так как в сутках **24**часа):

```
<?php
        echo 60 * 60 * 24;
?>
```

Задача

Задача. Переделайте приведенный код так, чтобы в нем использовались операции **+=**, **-=**, ***=**, **/=**, **++**, **--**. Количество строк кода при этом не должно измениться. Код для переделки:

```
<?php
        $var = 1;
        $var = $var + 12;
        $var = $var - 14;
```

```
        $var = $var * 5;
        $var = $var / 7;
        $var = $var + 1;
        $var = $var - 1;
        echo $var;
?>
```

Решение: заменим все подходящие места на сокращенную форму записи. К примеру, вместо **$var = $var + 12** можно написать **$var += 12**, а вместо **$var = $var + 1** будет **$var++**. Результат переделки будет выглядеть так:

```
<?php
        $var = 1;
        $var += 12;
        $var -= 14;
        $var *= 5;
        $var /= 7;
        $var++;
        $var--;
        echo $var;
?>
```

Задачи для решения

Работа с переменными

1. Создайте переменную **$a** и присвойте ей значение 3. Выведите значение этой переменной на экран.

2. Создайте переменные **$a=10** и **$b=2**. Выведите на экран их сумму, разность, произведение и частное (результат деления).

3. Создайте переменные **$c=15** и **$d=2**. Просуммируйте их, а результат присвойте переменной **$result**. Выведите на экран значение переменной **$result**.

4. Создайте переменные **$a=10**, **$b=2** и **$c=5**. Выведите на экран их сумму.

5. Создайте переменные **$a=17** и **$b=10**. Отнимите от **$a** переменную **$b** и результат присвойте переменной **$c**. Затем создайте переменную **$d**, присвойте ей значение 7. Сложите переменные **$c** и **$d**, а результат запишите в переменную **$result**. Выведите на экран значение переменной **$result**.

Работа со строками

6. Создайте переменную **$text** и присвойте ей значение **'Привет, Мир!'**. Выведите значение этой переменной на экран.

7. Создайте переменные **$text1='Привет, '** и **$text2='Мир!'**. С помощью этих переменных и операции сложения строк выведите на экран фразу 'Привет, Мир!'.

8. Создайте переменную **$name** и присвойте ей ваше имя. Выведите на экран фразу **'Привет, %Имя%!'**. Вместо **%Имя%** должно стоять ваше имя.

9. Создайте переменную **$age** и присвойте ей ваш возраст. Выведите на экран **'Мне %Возраст% лет!'**.

Обращение к символам строки

10. Создайте переменную **$text** и присвойте ей значение 'abcde'. Обращаясь к отдельным символам этой строки выведите на экран символ 'a', символ 'c', символ 'e'.

11. Дана произвольная строка, например, 'abcde'. Поменяйте первую букву (то есть букву 'a') этой строки на '!'.

12. Создайте переменную **$num** и присвойте ей значение '12345'. Найдите сумму цифр этого числа.

Практика

13. Напишите скрипт, который считает **количество секунд** в часе, в сутках, в месяце.

14. Создайте три переменные - час, минута, секунда. С их помощью выведите текущее время в формате **'час:минута:секунда'**.

15. Создайте переменную, присвойте ей число. Возведите это число в **квадрат** (*это значит нужно умножить его само на себя*). Выведите его на экран.

Работа с присваиванием и декрементами

16. Переделайте этот код так, чтобы в нем использовались операции **+=**, **-=**, ***=**, **/=**. Количество строк кода при этом не должно измениться.

```
$var = 47;
$var = $var + 7;
$var = $var - 18;
$var = $var * 10;
$var = $var / 20;
echo $var;
```

17. Переделайте этот код так, чтобы в нем использовалась операция **.=**. Количество строк кода при этом не должно измениться.

```
$text = 'Я';
$text = $text.' хочу';
$text = $text.' знать';
$text = $text.' PHP!';
echo $text;
```

18. Переделайте этот код так, чтобы в нем использовались операции **++** и **--**. Количество строк кода при этом не должно измениться.

```
$var = 10;
$var = $var + 1;
$var = $var + 1;
$var = $var - 1;
echo $var;
```

19. Переделайте этот код так, чтобы в нем использовались операции **++**, **--** , **+=**, **-=**, ***=**, **/=**. Количество строк кода при этом не должно измениться.

```
$var = 10;
$var = $var + 7;
$var = $var + 1;
$var = $var - 1;
$var = $var + 12;
$var = $var * 7;
$var = $var - 15;
echo $var;
```