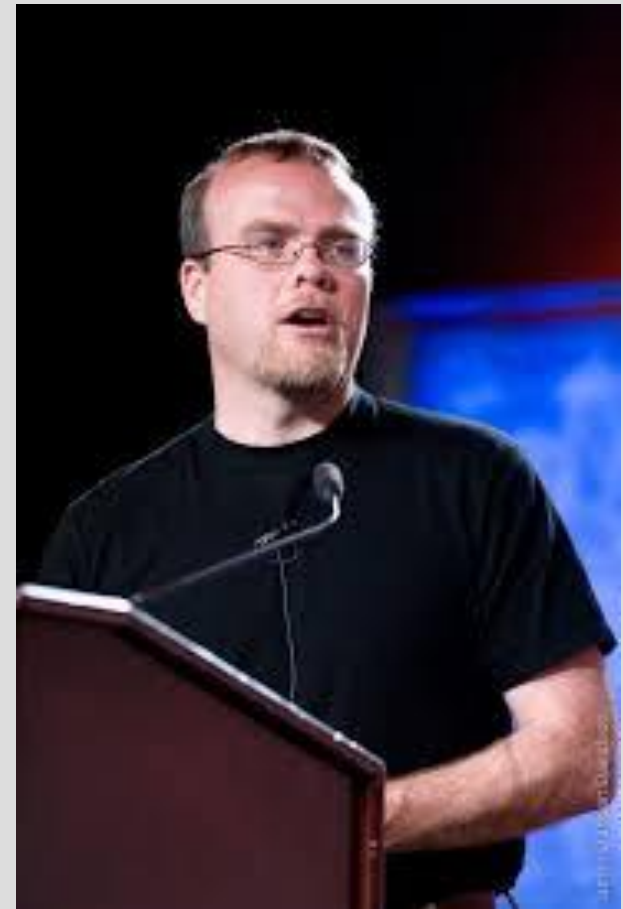


# Программирование на PHP.

- PHP – это гибкий и легкий язык web-программирования, обладающий широкими возможностями и неоспоримыми преимуществами.
- PHP это язык программирования, с помощью которого создают сайты, активно взаимодействующие с пользователем, например: Интернет-магазин, веб - каталоги, почтовую рассылку на сайте, новостную ленту, справочники, форумы и многое другое.
- PHP отличается от других подобных языков (JavaScript), тем, что код выполняется на сервере. Если вы имеете скрипт на сервере, то клиент получит результат работы этого скрипта, не имея возможности определить, каков был исходный код.
- Наилучшим качеством PHP является то, что он прост для новичка в программировании и предлагает много продвинутых возможностей для программиста-профессионала.

# История развития.

- **1994 год** – программист Рasmus Лердорф написал собственные Perl-скрипты для собственной страницы и назвал Personal Home Page (PHP)



# Основы PHP: синтаксис

PHP-код всегда начинается с `<?PHP` и заканчивается `?>`. Скриптовый блок PHP может быть размещен в любом месте документа.

```
<?php  
тело программы  
?>
```

Если на серверы с сокращенной поддержкой включены, то вы можете начать скриптовый блок с `<?>` и заканчивать `?>`, но в любом случае не рекомендую использовать короткие теги, потому что при переносе на сервер где их поддержка отключена, PHP будет восприниматься как

# Основы PHP: синтаксис

PHP файл обычно содержит HTML-теги, так же, как HTML-файл, и некоторый код PHP-скриптов. Вставьте следующий код, который посылает текст "Hello World" в браузер, в файл index.php:

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

# Основы PHP: синтаксис

- Каждый код строки в PHP должны заканчиваться точкой с запятой. Точка с запятой является разделителем и используется, чтобы отличить один набор инструкций от другого.
- Существуют два основных положения для вывода текста с помощью PHP: **echo()** и **print()**. В приведенном выше примере мы использовали **echo()** для вывода текста "Hello World".

- Можете попробовать использовать функцию print:

```
<html>
<body>
<?php
print "Hello World";
?>
</body>
</html>
```

Результат выполнения такого файла будет как и с echo()

Примечание: файл должен иметь расширение .php.

Если файл имеет расширение .html, PHP-код не будет выполнен.

# Комментарии в PHP

- В PHP, мы используем `//` , чтобы сделать однострочный комментарий или `/* и */`, чтобы сделать большой блок комментариев. Комментарии нужны чтобы писать пояснение в коде, сами комментарии на выполнение кода не влияют и не выводятся.

```
<?php
//Это комментарий
/*
Это
тоже
комментарий
*/
?>
```

# Переменные в PHP

- Переменные PHP используются для хранения значений, таких как текстовые строки, числа или массивы. Когда переменная объявлена, она может быть использована снова и снова в вашем скрипте. Все переменные в PHP начинаются с символа **\$**.

Правильный способ объявления переменных в PHP:

```
$var_name = value;
```

Одинарный знак **=** является оператором присваивания, то есть мы говорим на языке PHP: переменной `var_name` присваиваем значение `value`.



# Переменные в PHP

Когда мы объявляем строковую переменную, то мы должны строку заключить в кавычки, одинарные или двойные разницы не имеет, главное что если начинается с одинарной, то и заканчивается одинарной, начинается с двойной - заканчивается с двойной. Числовые переменные мы пишем без кавычек. Когда мы выводим строковую переменную, то не заключаем ее в кавычки.

Давайте попробуем создать переменную, содержащую строку, и переменную, содержащую число. Для этого код в файле index.php замените на следующий

```
<?php
$txt="Hello World!";
$x=16;
print $txt;
echo $x;
?>
```

# Переменные в PHP

- Еще одна особенность PHP, в нем не нужно объявлять переменные, то есть писать особой командой, что нужна новая переменная. Мы сразу переходим к делу и пишем новую переменную.
- Мало того переменная может из строковой стать числовой:

```
<?php
$var1 = '1 новая переменная';
$var2 = 6;
$var3 = $var1 + $var2;
print $var3;
?>
```

# Переменные в PHP

- Также можно обращаться к переменным и изменять их значение.

```
$var1 = 14;  
$var1 = $var1 + 1;  
print $var1;
```

Таким образом мы можем текущее значение переменной изменить и записать в эту же переменную.

Фактически в данном случае значение переменной подставляется в выражение и получается (14 + 1).

# Правила именования для переменных:

- Имя переменной должно начинаться с буквы или символа подчеркивания "\_"
- Имя переменной может содержать только буквенно-цифровые символы и знак подчеркивания (A-Я, AZ, 0-9, и \_)
- Имя переменной не должно содержать пробелов. Если имя переменной более одного слова, она должна быть разделена подчеркиванием (`$my_string`), или с капитализацией (`$myString`).

# Переменные в PHP

- PHP автоматически преобразует переменную в правильный тип данных, в зависимости от его значения. В строго типизированных языках программирования, вы должны объявить (определить) тип и имя переменной перед ее использованием.

# Операции со строковыми переменными

## Оператор сцепления (сложения) строк

Существует только одна строка оператора в PHP.

Оператор конкатенации (сложения, соединения) знак точки (.) Используется для перевода двух строковых значений вместе. Чтобы объединить две строковые переменные вместе, используйте оператор

```
<?php
$txt1="Привет, Мир!";
$txt2="Как дела?";
echo $txt1 . " " . $txt2;
?>
```

# Оператор сцепления (сложения) строк

Вы наверно заметили, что результат выводится в строку. Для того чтобы перенести текст на следующую строку, следует использовать HTML-теги `<br />` или `<p></p>`.

```
<?php
$txt1="Привет, Мир!";
$txt2="Как дела?";
print $txt1 . "<br />" . $txt2;
?>
```

```
<?php
$txt1="Привет, Мир!";
$txt2="Как дела?";
print "<p>" . $txt1 . "</p><p>" .
    $txt2 . "</p>";
?>
```

# StrLen() функция PHP

- StrLen () функция используется для возврата длины строки. Найдем длину строки:

```
<?php  
echo strlen("Привет, Мир!");  
?>
```

Длина строки часто используется в циклах или иные функциях, когда важно знать, когда строка заканчивается (то есть остановить цикл после последнего символа в строке).



# Strpos () функция

- Strpos () функция используется для поиска символов / текста в строке. Если найдено совпадение, то эта функция возвратит номер позиции первого совпадения. Если совпадений не обнаружится, то он вернет FALSE. Давайте посмотрим, сможем ли мы найти строку "Мир" в нашей строке:

```
<?php  
echo strpos("Привет, Мир!", "Мир");  
?>
```

Номер позиции строки "Мир" в приведенном выше примере 8. Причина того, что он 8 (а не 9), является то, что первая позиция символа в строке равна 0, а не 1.

# Числовые переменные PHP

- Целые числа в PHP

```
<?php
    $i = 456; // десятичное число
    $i = -895; // отрицательное число
?>
```

Дробные числа в PHP записываются в виде десятичных дробей:

```
$pi = 3.14;
```

```
$a = 1/3; print $a;
```

# Логические переменные PHP

Логическими они называются потому что используются для построение логики нашего кода. Допустим есть какая-то ситуация в которой при различных условиях нужно делать разные действия. У вас есть стакан с водой, из которого вы хотите пить, но если в нем будет мало воды, то вы не напьетесь, а если много воды, то когда вы стакан наклоните вода польется на вас. Поэтому нужно долить воды или отлить воду в зависимости от наполненности стакана.

Это можно представить в следующем виде:

```
$water = 100;  
$not_enough_water = TRUE;  
if($not_enough_water){  
    $water = $water + 20;  
}
```

# PHP операторы

## Арифметические операторы PHP

Оператор PHP	Описание	Пример	Результат
+	Сложение	$x=2$ $x+2$	4
-	Вычитание	$x=2$ $5-x$	3
*	Умножение	$x=4$ $x*5$	20
/	Деление	$15/5$ $5/2$	3 2.5
%	Остаток от деления	$5\%2$ $10\%8$ $10\%2$	1 2 0
++	Инкремент (увеличение на 1)	$x=5$ $x++$	$x=6$
--	Декремент (уменьшение на 1)	$x=5$ $x--$	$x=4$

# RНР операторы

## Операции присвоения RНР

Оператор RНР	Пример	Эквивалент
=	$x=y$	$x=y$
+=	$x+=y$	$x=x+y$
-=	$x-=y$	$x=x-y$
*=	$x*=y$	$x=x*y$
/=	$x/=y$	$x=x/y$
.=	$x.=y$	$x=x.y$
%=	$x%=y$	$x=x\%y$

Это альтернативный вариант записи арифметических действий

# PHP операторы

## Операции сравнения PHP

Оператор PHP	Описания	Пример
==	Равно	5==8 возвращает false
!=	Не равно	5!=8 возвращает true
<>	Не равно	5<>8 возвращает true
>	Больше чем	5>8 возвращает false
<	Меньше чем	5<8 возвращает true
>=	Больше или равно	5>=8 возвращает false
<=	Меньше или равно	5<=8 возвращает true

# RНР операторы

## Логические операторы RНР

Оператор RНР	Описание	Пример
&&	И	x=6 y=3 (x < 10 && y > 1) возвращает true
	ИЛИ	x=6 y=3 (x==5    y==5) возвращает false
!	НЕТ	x=6 y=3 !(x==y) возвращает true

# Оператор if

Довольно часто в зависимости от обстоятельств приходится принимать различные решения. Если пойдет дождь, я возьму зонт, если будет тепло, мы пойдем на пляж. Так в PHP мы проверяем выражения на истинность и выполняем соответствующие действия:

```
<?php  
if(условие){  
    Действие  
}  
?>
```



# Оператор if

Еще обычно рисуют блок схемы для того чтобы показать как работает алгоритм:



```
<?php  
if(5>3){  
    print '5 больше чем 3';  
}  
?>
```

# Оператор if

- PHP if с использованием && (И):

```
<?php
$x = 4;
$y = 5;
if($x==4 && $y==5){
    print 'x равен 4, y равен 5';
}
if($x==3 && $y==5){
    print 'x равен 3, y равен 5'; // не выводится, потому что x
    не равен 3
}
?>
```

# Оператор if

- PHP if с использованием || (ИЛИ):

```
<?php
$x = 4;
$y = 5;
if($x == 4 || $y == 5){
    print 'x равно 4 или y равно 5<br />';
}
```

```
if($x == 3 || $y == 5){
    print 'x равно 3 или y равно 5<br />';
}
```

```
if($x == 4 || $y == 6){
    print 'x равно 4 или y равно 6<br />';
}
```

Во всех трех случаях хотя бы одно условие выполняется, поэтому будут выведены все три строки.

# Использование логических переменных в операторе if

```
<?php
$condition = true;
if($condition){
    print 'Hello, World!';
}
?>
```

Но условие может быть не всегда верным, тогда мы добавляем else, чтобы задать два варианта использования if:

```
<?php
$x=10;
if(empty($x)){
    print 'переменная пустая';
}else{
    print 'значение переменной ' . $x;
}
?>
```

Функция empty() проверяет пустая или нет переменная. Если переменная не пустая, то функция возвращает FALSE, а если пустая то возвращает TRUE.

# Использование логических переменных в операторе if

Мы можем также использовать оператор отрицания !  
(восклицательный знак):

```
<?php
$x=10;
if(!empty($x)){
    print 'значение переменной ' . $x;
}else{
    print 'переменная пустая';
}
?>
```

Тогда то что было истинным станет ложным, то что было ложным станет истинным.

# Использование логических переменных в операторе if

И есть еще похожая функция `isset()`, которая проверяет наличие переменной вообще:

```
<?php
$x = "";
if(empty($x)){
    print 'переменная x пустая<br />';
}

if(isset($x)){
    print 'переменной x существует';
}

?>
```

# Оператор switch

Давайте теперь представим, что нам нужно выполнить десять проверок и в зависимости от результатов проверки выполнить одно из десяти действий. Конечно мы можем записать это и через if:

```
<?php
if($x == 1){
    // действия
}
if($x == 2){
    // действия
}
if($x == 3){
    // действия
}
if($x == 4){
    // действия
}
if($x == 5){
    // действия
}
if($x == 6){
    // действия
}
if($x == 7){
    // действия
}
if($x == 8){
    // действия
}
if($x == 9){
    // действия
}
if($x == 10){
    // действия
}
?>
```

```
<?php
switch ($x){
    case 1:
        // действия
        break;
    case 2:
        // действия
        break;
    case 3:
        // действия
        break;
    case 4:
        // действия
        break;
    case 5:
        // действия
        break;
    case 6:
        // действия
        break;
    case 7:
        // действия
        break;
    case 8:
        // действия
        break;
    case 9:
        // действия
        break;
    case 10:
        // действия
        break;
    default:
        // действия при условии, что другие условия не подошли
}
?>
```

Оператор switch позволяет проверить значение выражения при многих условиях.

Оба примера будут работать одинаково, только в случае со switch есть возможность добавить default, который будет срабатывать, когда значение \$x не будет равняться от 1 до 10.

# Массивы

Массивы могут включать в себя как числовые, так и строковые переменные.

Суть массива в следующем, есть множество ключей массива и есть множество значений массива. Каждому ключу массива принадлежит одно значение. Например у нас есть ключи 1,2,3, то значения могут быть 4, 'Hello World', true. Записываются массивы так же как и переменные, со знаком доллара:

```
$tree = array();
```

```
$tree = array(
```

```
    1 => 4,
```

```
    2 => 'Hello World',
```

```
    3 => true,
```

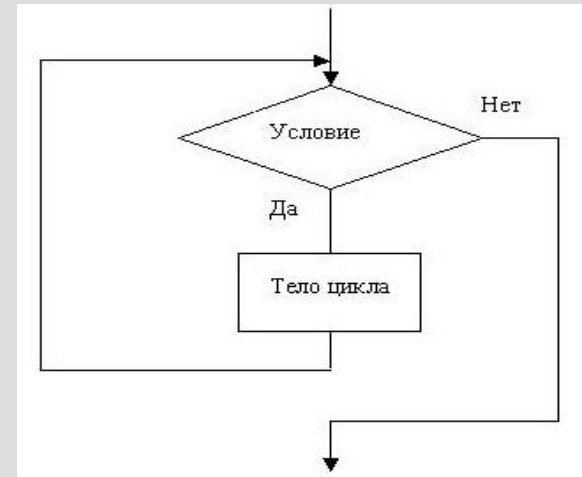
```
);
```



# PHP while циклы

Довольно часто в программирование нам приходится выполнять повторяющиеся действия, пока нас не удовлетворит результат.

В круглых скобках мы пишем условия, в фигурных что нужно сделать когда условие выполняется. Условие это логическое выражение, которое имеет два состояния TRUE или FALSE. Если хотите еще раз понять о логических выражениях, то вам следует почитать урок об операторе if. А теперь пример while:



```
<?php
$counter = 5;
$newArray = array();
while($counter > 2){
    $newArray[] = $counter;
    $counter--;
}
print_r($newArray);
?>
```

В результате будут выведены элементы массива 5,4,3.

1 шаг. \$counter = 5 (5 > 2)

2 шаг. \$counter = 4 (4 > 2)

3 шаг. \$counter = 3 (3 > 2)

4 шаг. \$counter = 2 (2 = 2, условие перестало выполняться, поэтому и действия не произошло и в массиве нет 2)

# PHP foreach циклы

Еще один необходимый цикл, он часто встречается когда нужно перебрать массив. Под перебрать я подразумеваю, взять один элемент массива, провести с ним действия, потом взять следующий элемент массива, провести с ним те же действия и т.д. пройти по всем элементам.

Давайте рассмотрим это на примере, допустим у нас есть десять элементов "красные" и "синие", давайте подсчитаем количество

```
<?php
$counter = 0;
$newArray = array(
    1=>'красный',
    2=>'синий',
    3=>'синий',
    4=>'красный',
    5=>'красный',
    6=>'синий',
    7=>'красный',
    8=>'синий',
    9=>'красный',
    10=>'синий',

    foreach($newArray as $key){
        if($key == 'красный'){
            $counter++;
        }
    }
    print 'У нас есть ' . $counter . ' красных
    элементов';
?>
```

# Цикл FOR

Цикл for похож на цикл while. For обычно используют, когда нужно заранее известное количество повторений цикла. Например 10 или 100 повторений:

```
for($i=0;$i<10;$i++){  
    //десять повторений  
}
```

Как и у прошлых циклов все параметры цикла for пишутся в круглых скобках. Обычно используют три параметра:

1.  $\$i=0$  - инициализация счетчика, переменной куда мы будем записывать номер текущего повторения.
2.  $\$i<10$  - условие при котором будет работать цикл for.
3.  $\$i++$  - увеличение счетчика на единицу, чтобы перейти на следующее повторение.