# Основы PHP

http://php720.com/lesson/25

http://oftob.ru/%D1%83%D1%80%D0%BE%D0%BA%D0%B8-php/533-php-%D1%82%D0%B8%D0%BF%D1%8B-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85

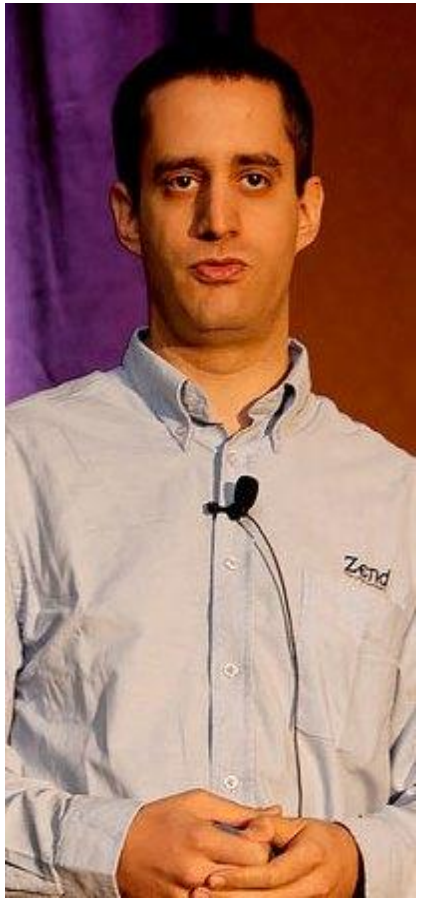
# Программирование на PHP.

- 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.

# Программирование на PHP. История развития.

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

# Программирование на PHP. История развития.




**1997 год** - два программиста Энди Гутманс и Зив Сураски взяли за основу идею Расмуса и переписали PHP с нуля. Название "Personal Home Page" было изменено на **Hypertext Preprocessor**

# Программирование на PHP. История развития.

- 2000 год – вышла 4-я версия PHP, ставшая стандартом для Веб-разработки. Сейчас разработн PHP5.
- PHP – это гибкий и легкий язык web-программирования, обладающий широкими возможностями и неоспоримыми преимуществами.

Сегодня PHP — это мощный кроссплатформенный набор средств, который располагается на сервере и предназначен для обработки специального кода, встраиваемого в HTML-страницу. Благодаря этому, появляется возможность легко создавать динамические сайты. Файлы, созданные таким образом, хранятся и обрабатываются на сервере, и когда посетитель запрашивает документ с PHP, скрипт обрабатывается не браузером посетителя, как, например, Java Script, а сервером, и посетителю передаются уже только результаты работы.

Динамический сайт, как правило, полностью управляется через несложный веб интерфейс. Управление возможно не только отдельными страницами, но и структурой разделов, а также информационной сеткой сайта.

Управление сайтом доступно (и рекомендуется) менеджерам, которые непосредственно общаются с клиентами и знают какие вопросы нужно оперативно осветить на сайте.

Важный плюс – это оперативность публикации новых материалов, объявлений и другой важной информации, которое делает общение с посетителями (клиентами) сайта «живым» и интересным.

# Изучаемые вопросы:

- Основные способы вставки PHP кода
- Особенности PHP кода
- Комментарии
- Переменные
- Типы переменных
- Присваивание и удаление значений переменной
- Особенности типа Boolean
- Особенности типов Integer и Float
- Особенности типа NULL

# Основные способы вставки PHP кода

```
<?php
    // Тут будет Php код
?>
```

PHP код состоит из инструкций, разделенных знаком ;

## Правильные записи

```
<?php
    инструкция1;
    инструкция2;
?>
```

```
<?php
    инструкция1;
инструкция2;
?>
```

# Вывод текста на экран. Оператор echo.

Когда нужно отобразить текст на веб-странице, то оператор echo является наиболее употребляемым оператором в PHP. Как его использовать - после слова echo нужно поместить строку текста в кавычки:

```
<?php
echo 'Привет от PHP';
?>
```

Отображение в браузере:

Привет от PHP

(для отображения текста можно использовать как двойные кавычки, так и одинарные).

Для чисел кавычки можно не использовать:

```
<?php
echo 2014;
?>
```

# Комментарии

**Однострочные**

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

```php
<?php
    /* Это комментарий
    многострочный
    echo "Привет";
    */
?>
```

# Переменные в PHP

PHP создан не только для форматирования статичного текста. Для того, чтобы обрабатывать различные данные были придуманы **переменные**.
Каждая переменная содержит определенное значение.
Синтаксис переменной состоит из знака доллара - **$** и "свободного" идентификатора которому присваивается какое-нибуть значение. Например:

```
<?php $name = "Виктор"; ?>
```

# Создание переменной

Переменная создается тогда, когда ей присваивают какое-нибуть значение. Для присвоения значения переменной используют оператор присвоения. Например:

```
<?php
$surname = "Петров";
$number = 1269794645;
$pi = 3.14159265;
$hello = "Hi all";
?>
```

Переменную можно вывести на экран с помощью оператора echo, вот так:

```
<?php
$name = "Виктор";
echo "Ваше имя ", $name, "<br>";
?>
```

Отображение в браузере:

Ваше имя Виктор

Создадим переменную которая будет содержать значение количества бананов, вторая переменная количество лимонов, а третья - их суммарное количество.

```php
<?php
$bann = 5;   // Бананы
$lim = 10;    // Лимоны
$together = $bann + $lim; // Всего
echo "Количество фруктов ", $together;
?>
```

Отображение в браузере:

Количество фруктов 15

# Интерполяция переменных в PHP

Значение переменной может быть отображено например так:

```
<?php
$capital = "Paris";
echo "The capital of France is", $capital,"<br />";
?>
```

Но есть способ сделать это проще. Если имя переменной заключено в двойные (**не одинарные**) кавычки, то переменная интерполируется. Например:

```
<?php
$capital = "Paris";
echo "The capital of France is $capital <br />";
?>
```

Отображение в браузере:

The capital of France is Paris

## Переменные, содержащие имена других переменных

В PHP возможно размещать в значение переменных не только обычные значения, но и имена других переменных.

```
<?php
$apples = 5;
$fruit = "apples"; /* Создаем переменную $fruit, которая содержит
                      имя переменной $apples */
// Сейчас мы можем вывести $apples, как $$fruit
echo "Число яблок - ", $$fruit; ?>
```

Для корректного отображения подобных переменных в строковых константах, заключенных в двойные кавычки, следует также использовать фигурные скобки: ${$fruit}.

Например:

```
<?php
echo "Число яблок - ${$fruit}";
?>Число яблок - 5
```

# Константы в PHP

Когда не нужно менять заданное значение для переменной, то имеет смысл создать константу и потом использовать ее в любой части скрипта. Для описания константы используют функцию define, которой передается ее имя и значение:

```
<?php
define("pi", 3.14); ?>
```

Имя константы нужно всегда заключать в кавычки, а ее значение только тогда когда оно является строкой.

Пример использования константы:

```
<?php
define("pi", 3.14);
echo "Математическая константа Пи равняется ", pi; ?>
```

Отображение в браузере:

Математическая константа Пи равняется 3.14

Попытка изменения константы приведет к неработоспособности скрипта .

# Типы данных PHP

PHP является языком динамической типизации (тип переменной определяется на основе её значения).
Типы, которые можно использовать в PHP:

- **Boolean**. Это логический тип, который содержит значение TRUE или FALSE.
- **Integer**. Содержит значения целого числа (Например: 4 или 10 или другое целое число).
- **String**. Содержит значение текста произвольной длины (Например: Олег, Киев, Австрия).
- **Float**. Вещественное число (Например: 1.2, 3.14, 8.5498777).
- **Object**. Объект.
- **Array**. Массив.
- **Resource**. Ресурс (Например: файл).
- **NULL**. Значение NULL.

```php
<?php
$bool = TRUE; // Значение Boolean
$int = 100; // Значение Integer
$string = "Переменная содержит текст"; // Значение String
$string2 = "5425"; // Значение String, так как число взято в кавычки
! $float = 44.122; // Значение Float
?>
```

Для предотвращения появления ошибок рекомендуется не смешивать разные типы данных. Если надо изменить тип данных переменной, то для этого нужно слева от имени переменной в круглых скобках указать нужный тип:

```php
<?php
$str = "50000"; // Значение String
$new_str = (integer) $str; // Теперь значение стало Integer
echo $new_str + $new_str;
?>
```

Отображение в браузере:

100000

# Математические операторы и математические функции PHP

Числовые данные обрабатываются при помощи таких операторов :

**+, -, *, /, %** (остаток от деления)

```
<?php
echo "2 + 2 = ", 2 + 2, "<br>";
echo "5 - 2 = ", 5 - 2, "<br>";
echo "10 * 10 = ", 10 * 10, "<br>";
echo "100 / 2 = ", 100 / 2, "<br>";
echo "10 % 2 = ", 10 % 2, "<br>";
?>
```

Отображение в браузере

2 + 2 = 4

5 - 2 = 3

10 * 10 = 100

100 / 2 = 50

10 % 2 = 0

**Abs**. Модуль числа, **Sin**. Синус, **Sinh**. Гиперболический синус. **Cos**. Косинус **Cosh**. Гиперболический косинус. **Acos**. Арккосинус. **Acosh**. Гиперболический арккосинус. **Asin**. Арксинус. **Asinh**. Гиперболический арксинус. **Atan2**. Арктангенс частного двух переменных. **Tan**. Тангенс. **Tanh**. Гиперболический тангенс. **Atan**. Арктангенс. **Atan**. Гиперболический арктангенс
**Base_convert**. Преобразование числа в строковом представлении из одной системы счисления в другую. **Decoct**. Преобразование числа в восьмеричное представление в виде строки. **Bindec**. Преобразование строки, предоставленной в двоичном числе, в целое значение. **Octdec**. Преобразование строки, представляющей восьмеричное число, в целое число. **Hexdec**. Преобразование строки, которая представляет шестнадцатеричное число, в целое число. **Ceil**. Округление числа в большую сторону. **Floor**. Округление числа в меньшую сторону.**Deg2rad**. Градусы в радианы. **Exp**. Вычисление экспоненты числа.
**Fmod**. Остаток от деления двух чисел. **Getrandmax**. Макс. значение, которое получают функцией ***rand().*** **Hypot**. Вычисление гипотенузы по двум катетам.
**Is_finite**. Проверка, является ли число конечным вещественным числом.
**Is_infinite**. Проверка, является ли число бесконечностью.
**Is_nan**. Проверка, является ли значение Не числом(Not-A-Number).
**Lcg_value**. Генератор случайных чисел.
**Log10**. Десятичный логарифм. **Log**. Натуральный логарифм. **Max**. Максимум заданных чисел. **Min**. Минимум заданных чисел. **Mt_getrandmax**. Макс. значение, которое можно получить функцией ***mt_rand***. **Mt_rand**. Генератор псевдослучайных чисел по алгоритму. **Pi**. Значение числа π.
**Pow**. Возведение в степень. **Round**. Округляет число типа float. **Sqrt**. Квадратный корень.

```php
<?php
echo "round(4.2) = ", round(4.2), "<br>";
?>
```

Отображение в браузере

round(4.2) = 4

# Операторы присвоения в PHP

Основным оператором присвоения является знак равенства =.
Он присваивает значение определенной переменной:

```
<?php $fruits=14; ?>
```

В одной строке можно присвоить одно значение сразу нескольким переменным, например:

```
<?php $n = $m = $p = 3; echo $n, $m, $p; ?>
```

Отображение в браузере:

333

Также в PHP есть комбинированные операторы, которые делают код более компактным. Вот их перечень:

**+=, -=, /=, .=, %=, &=, |=, ^=, <=, >=**

Например, если нужно прибавить 55 к значению переменной $number, это можно записать как:
$number = $number + 55, а если использовать комбинированный оператор, то так:
$number += 55.

## Увеличение и уменьшение на 1

Если есть переменна $a = 0, то чтобы добавить 1 к этой переменной нужно написать: $a++, если нужно отнять 1, то нужно записать так: $a--. Оператор ++ называют инкрементом, а -- декрементом.

## Оператор исполнения PHP

В PHP существует такой оператор, как **оператор исполнения**, он нужен для того чтобы выполнять команды ОС и использовать результат этого выполнения.

Любая строка, которая заключена в обратные апострофы — ` считаются как команда ОС. Например:

```
<?php
$d = `dir d:\\`;
echo $d;
?>
```

Как результат вы получите список директорий диска D.

# Строковые операторы PHP

PHP имеет два строковых операторы.

**Первый** - оператор конкатенации, который объединяет две строки в одну.

**Второй** - конкатенирующий оператор присвоения .=, добавляет к строке нужное значение. Например:

```
<?php
$d = "Hello";
$f = $d." world"; // Теперь $f = "Hello world"
echo $f;
echo "<br/>";
$f .= " ! ! !"; // Теперь $f = "Hello world ! ! !"
echo $f;
?>
```

Отображение в браузере:

Hello world

Hello world ! ! !

# Условный оператор IF в PHP

Во всех высокоуровневых языках программирования есть оператор if, в PHP синтаксис этого оператора такой:

**if** (exp) statement

**exp** (выражение) - логическое выражение, которое может быть истиной (TRUE) или ложью (FALSE).

**statement** (инструкция) выполняется тогда, когда **exp** — истина, и не выполняется когда **exp** ложь!

Например, если скорость машины будет больше 60 то это значит, что водитель превышает скорость:

```
<?php
$speed = 80;
if ($speed > 60)
    echo "Превышение скорости !";
?>
```

Если нужно чтобы при выполнении условия выполнялись сразу несколько операторов, то нужно заключить их в фигурные скобки { }:

```
<?php
$speed = 80;
if ($speed > 60)
    {echo "Превышение скорости! <br>";
     echo "Пожалуйста, уменьшите скорость!"; }
?>
```

Отображение в браузере:

Превышение скорости!

Пожалуйста, уменьшите скорость!

# Операторы сравнения PHP

Все операторы сравнения PHP указанные в таблице:

| | | |
|---|---|---|
| == | Равенство | Истина, если $a равно $b |
| === | Идентичность | Истина, если $a равно $b, и они одного и того же типа |
| != | Неравенство | Истина, если $a не равно $b |
| <> | Неравенство | Истина, если $a не равно $b |
| !== | Неидентичность | Истина, если $a не равно $b, или они не одного типа |
| < | Меньше | Истина, если $a меньше $b |
| > | Больше | Истина, если $a больше $b |
| <= | Меньше или равно | Истина, если $a меньше или равно $b |
| >= | Больше или равно | Истина, если $a больше или равно $b |

```php
<?php
$speed = 45;
if ($speed != 60)
    echo "Скорость в пределах нормы";
?>
```

Если нужно применить к выражению несколько условий, то используют логические операторы:

```php
<?php
$speed = 40;
if ($speed > 35 && $speed < 55)
   { echo "Скорость в пределах нормы"; }
?>
```

| | | |
|---|---|---|
| && | Логическое "И" | Истина, если истинно $a и $b |
| \|\| | Логическое "ИЛИ" | Истина, если истинно $a или $b |
| xor | Логическое "Исключающее ИЛИ" | Истина, если истинно $a или $b, но не оба одновременно |
| ! | Логическое "НЕ" | Истина, если $a ложь |

# Оператор ELSE в PHP

Синтаксис оператора:

```
if(exp) statement1 else statement2
```

Пример:

```
<?php
$speed = 50;
if ($speed > 60)
   echo "Превышение скорости !";
else
   echo "Скорость в пределах нормы"
?>
```

В этом случае будет выведено сообщение

Скорость в пределах нормы

# Оператор ELSEIF в PHP

Оператор if имеет еще одно расширение, это оператор elseif, он используется для последовательной проверки условий.

Синтаксис:

```
if (exp)
    statement1
elseif (exp2)
    statement2
```

Также можно записывать так:

```
if (exp)
    statement1
else if (exp2)
    statement2
```

Пример:

```php
<?php $speed = 50;
if ($speed < 30)
   echo "Скорость в пределах нормы";
elseif ($speed == 30)
   echo "Ваша скорость 30 км/час";
elseif ($speed == 40)
   echo "Ваша скорость 40 км/час";
elseif ($speed == 50)
   echo "Ваша скорость 50 км/час";
elseif ($speed == 60)
   echo "Ваша скорость 60 км/час";
else echo "Превышение скорости !"; ?>
```

Также, такой кусок кода можно записать и так:

```
<?php
$speed = 50;
if ($speed < 30)
      echo "Скорость в пределах нормы";
elseif ($speed >= 30 && $speed <= 60)
       echo "Ваша скорость {$speed} км/час";
else echo "Превышение скорости !";
?>
```

В этом случае будет выведено сообщение "Ваша скорость 50 км/час". А если бы не одно условие не подошло бы , то сработал бы оператор else и мы увидели "Превышение скорости !".

# Тернарный оператор PHP

Тернарный оператор работает почти также как и оператор if, но при использовании тернарного оператора, мы вместо ключевых слов пишем ? и:. Синтаксис:

```
$var = condition ? exp1 : exp2;
```

Если условие выполняется, то переменной $var присваивается результат вычисления exp1, иначе exp2.

Пример:

```
<?php
$speed = 55;
echo ($speed <= 60) ? "Скорость в пределах нормы" :
"Превышение скорости !"; ?>
```

В результате мы увидим строку - "Скорость в пределах нормы".

# Оператор switch

Иногда использование конструкции операторов if .. elseif несколько утомляет.
Чтобы исправить эту ситуацию есть оператор switch.
Синтаксис:

```
switch (exp)
{ case condition1: exp1; break;
case condition2: exp2; break;
case condition3: exp3; break;
default: exp4;
break; }
```

Сначала записывается ключевое слово switch, после которого в скобках записывается некоторое выражение.
Далее, после слова case нужно перечислить возможные варианты значений, если значение истина, то выполняется группа операторов, которые записаны до оператора break. Если ни одно условие не походит, то выполняется оператор default (если оператор default не записывать, то при не выполнении никаких других условий ничего не произойдет).

```php
<?php
$speed = 55;
switch($speed)
{  case 30 : echo "Ваша скорость 30 км/час"; break;
   case 58 : echo "Ваша скорость 50 км/час"; break;
   case 70 : echo "Превышение скорости !"; break;
   default : echo "Скорость в пределах нормы"; break; }
?>
```

Также, при использовании оператора switch, мы можем записать несколько условий для некоторого действия:

```php
<?php
$speed = 55;
switch($speed)
{ case 30 : case 58 : echo "Скорость в пределах нормы"; break;
  case 70 : echo "Превышение скорости !"; break;
  default : echo "Скорость в пределах нормы"; break; } ?>
```

В результате мы увидим — "Скорость в пределах нормы".

# Цикл FOR в PHP

Основной задачей компьютеров есть обработка большого количества информации, которое у человека заняло бы очень много времени. Для обработки таких задач компьютер использует циклы. Первым циклом которым мы начнем главу будет цикл for. Ниже приведен его синтаксис:

**for** (exp1; exp2; exp3) statement

В выражение **exp1** вставляют начальное значение для счетчика цикла — переменная, которая считает количество раз выполнения тела цикла.

**exp2 —** задает условие повторения цикла. Цикл будет выполнятся пока это условие будет true.

**exp3 —** выполняется каждый раз после выполенения тела цикла. Обычно, оно используется для изменения (увеличение или уменьшение) счетчика.

Пример:

```
<?php
   for ($i = 0; $i < 10; $i++)
    { echo "Вывод строки. 10 раз <br>"; }
?>
```

Отображение в браузере:

Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз
Вывод строки. 10 раз

# Циклы WHILE в PHP

Цикл WHILE, вместо использования счетчика цикла проверяет некоторое условие до того, пока это условие Истина (TRUE).

Синтаксис:

**while** (exp) statement

Условие проверяется перед выполнением цикла, если оно будет Ложным в начале, то цикл не выполнится ни разу!

В теле цикла должна быть переменная которая будет оказывать влияние на условие, чтобы предотвратить зацикливание.

Пример:

```
<?php
   $counter = 0;
   while ($counter < 5)
   { echo "Эта строка выведется 5 раз <br>"; $counter++; }
?>
```

# Цикл DO... WHILE в PHP

Главное отличие цикла DO ... WHILE от WHILE в том, что первый сначала выполняется тело цикла, а потом проверяет условие. Т.е., если условие сразу Ложь, то цикл выполнится один раз.

Синтаксис

**do** statement **while** (condition)

Использование цикла DO... WHILE:

```
<?php
   $counter = 6;
   do
       { echo "Эта строка выведется 1 раз <br>";
         $counter++;
        }
while ($counter < 5);
?>
```

Эта строка выведется 1 раз

Так как условие цикла сразу Ложь (6 > 5), цикл выполнился всего один раз, так как сначала выполняется тело цикла, а потом проверяется условие цикла.

# Цикл FOREACH в PHP

Цикл FOREACH представлен для упрощения работы с массивами. Массивы состоят из отдельных элементов, цикл FOREACH предназначен для перебора этих элементов без счетчика.

Синтаксис:

```
foreach (array as $value) statement
foreach (array as $key => $value) statement
```

Использование цикла:

```
<?php
$array = array ("Apple", "Limon", "Chery", "Oranges");
foreach ($array as $value)
    { echo "Вы выбрали фрукт - $value <br>"; }
?>
```

Отображение в браузере:

Вы выбрали фрукт - Apple
Вы выбрали фрукт - Limon
Вы выбрали фрукт - Chery
Вы выбрали фрукт - Oranges

# Функции для обработки строк в PHP

С помощью этих функций можно, например, обрезать строку, дописывать строку, заменить часть строки и много другое. Это очень полезный инструмент при разработке скриптов.
Все функции для обработки строк перечислены ниже:

## Функция substr

Функция substr используется для получения части строки.
Синтаксис:
string substr (string $string, int $start [, int $length ])
Первый параметр $string - строка из которой нужно получить подстроку начиная с позиции $start и длинной в $length.
Пример:

```
<?php echo substr("Hello world", 6, 5); ?>
```

Отображение в браузере:
world
Последний параметр $length необязательный

```
<?php echo substr("Hello world !!!", 6); ?>
```

# Функция strpos

Функция возвращает позицию первого вхождения подстроки в строку

int strpos (string $string , mixed $needle [, int $offset = 0 ])

$string - строка в которой будет произведен поиск,

$needle - строка, которую нужно найти,

$offset - необязательный параметр, если этот параметр указан, то поиск будет начат с указанного количества символов с начала строки

Пример:

```
<?php
echo strpos("Hello world", "world"); // получим 6
?>
```

В результате, получим 6, так как строка "world" впервые встречается на 6 позиции

# Создание массивов в PHP

Массив - это набор данных, которые объединены под одним именем. Массив состоит из нескольких элементов, которые имеют свой определенный индекс. Массивы создаются при помощи оператора присвоения, также, как и переменная.

Имена массивов начинаются со знака **$**, после которого следует произвольный идентификатор, далее идут квадратные скобки: **$**arr[0] = "php";

Данная конструкция создает массив и присваивает его элементу с индексом 0 значение "php", после чего мы можем обращаться к этому элементу как к обычной переменной: echo $arr[0]. В результате мы увидим слово *php*.

Также, мы можем добавить еще элементы к массиву:

```
<?php $arr[1] = "html"; $arr[2] = "css"; ?>
```

В качестве индекса элементов массива мы можем использовать не только числа:

```
<?php
$arr["Kiev"] = 3000000;
$arr["Paris"] = 5000000;
$arr["LA"] = 15000000;
?>
```

В качестве значений индексов элементов и самих элементов мы можем использовать одинаковые типы данных **одновременно**!

Также существует сокращенная запись для индексирования:

```
<?php
$arr[] = 3000000; $arr[] = 5000000; $arr[] = 15000000;
?>
```

В этом случае первый элемент (3000000) получит индекс 0! Нужно иметь это ввиду.

Для создания массива можно использовать функцию array:

```
<?php
$arr = array("php", "html", "css");
?>
```

В этом случае первый элемент получит индекс 0. Если нужно присвоить какой-то другой номер, то можно воспользоваться конструкцией =>:

```
<?php
$arr = array(1 => "php", "html", "css"); ?>
```

Теперь элемент под номером 1 это "php", а не "html"! Также, можно создать массив со строковым индексом:

```
<?php
$arr = array("first" => "php", "second" => "html", "third" => "css");
?>
```

Начиная с версии PHP 5.4, массивы можно создать через квадратные скобки:

```
<?php $arr = ["php", "laravel", "yii", "zend", "cakephp"]; ?>
```

# Модификация элементов массива в PHP

Есть массив:

```
<?php
$arr[0] = "PHP";
$arr[1] = "HTML";
$arr[2] = "CSS";
?>
```

Для того, чтобы изменить значение элемента используем оператор присвоения:

```
<?php
$arr[1] = "JAVASCRIPT";
?>
```

Для того, чтобы добавить новый элемент в конец массива использует конструкцию:

```
<?php
$arr[] = "JQUERY";
?>
```

Для того, чтобы вывести на экран массив можно использовать foreach:

```
<?php
$arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS"; $arr[1] = "JAVASCRIPT"; $arr[] = "JQUERY";
foreach($arr as $key => $value) { // при переборе: $key - индекс элемента массива, $value - значение элемента массива
echo $value.'<br/>'; } ?>
```

Отображение в браузере:

PHP
JAVASCRIPT
CSS
JQUERY

# Удаление элементов массива в PHP

Если нам нужно удалить один из элементов массива, то для этого мы должны использовать функцию unset

```
<?php
$arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
unset($arr[1]);
foreach($arr as $key => $value)
                { echo $value.'<br/>'; }
?>
```

# Перебор элементов массива в PHP

Кроме использования цикла for для вывода всех элементов массива на экран мы можем использовать функцию print_r, которая выведет все элементы массива вместе с их индексами.

```
<?php
$arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
print_r($arr);
?>
```

Отображение в браузере:

```
Array ( [0] => PHP [1] => HTML [2] => CSS )
```

Также в PHP присутствует специальный цикл для обработки массивов - цикл foreach

```php
<?php
$arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
foreach($arr as $value)
{
echo $value, "<br>";
}
?>
```

Отображение в браузере:

PHP
HTML
CSS

Для вывода **индекса** элемента нужно использовать второй вариант синтаксиса цикла foreach

```
<?php
$arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
foreach($arr as $key => $value)
{
// $key - индекс эл.массива, $value - значение эл.массива
echo "[{$key}] => {$value} <br/>";
}
?>
```

Отображение в браузере:

[0] => PHP
[1] => HTML
[2] => CSS

# Сортировка массивов в PHP

Очень часто нужно отсортировать массив по индексу его элементов, по алфавиту его элементов, по возрастанию, по убыванию и т. д. В PHP для этого существуют функции.

Функция - sort, которая сортирует массив по возрастанию значений его элементов, при этом изменяя индекс после сортировки:

```
<?php
   $arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
   sort($arr);
   print_r($arr);
?>
```

Отображение в браузере:

```
Array ( [0] => CSS    [1] => HTML    [2] => PHP )
```

Функция - rsort, которая сортирует массив по убыванию значений его элементов, при этом изменяя индекс после сортировки:

```
<?php
   $arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
   rsort($arr);
   print_r($arr);
?>
```

Отображение в браузере:

```
Array ( [0] => PHP    [1] => HTML     [2] => CSS )
```

Третья функция - ksort, которая сортирует массив по ключам, сохраняя отношения между ключами и значениями:

```
<?php
  $arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS"; ksort($arr);
  print_r($arr); ?>
```

Отображение в браузере:

Array ( [0] => PHP  [1] => HTML  [2] => CSS )

Функция - krsort, которая сортирует массив по убыванию индексов его элементов:

```php
<?php
   $arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
   krsort($arr);
   print_r($arr);
?>
```

Отображение в браузере:

Array ( [2] => CSS  [1] => HTML  [0] => PHP )

# Навигация по массивам в PHP

Навигация по массива дает возможно узнать текущий , следующий, предыдущий, последний элемента массива.

**Текущий элемент массива** определяет функция current:

```
<?php echo "Now is: ", current($arr), "<br>"; ?>
```

**Следующий элемент массива** определяет функция next:

```
<?php echo "Next is: ", next($arr), "<br>"; ?>
```

**Предыдущий элемент массива** определяет функция prev:

```
<?php echo "Previously is: ", prev($arr), "<br>"; ?>
```

**Последний элемент массива** определяет функция end:

```
<?php echo "The end is: ", end($arr), "<br>"; ?>
```

Для определения первого (**возврата указателя**) элемента массива используют функцию reset:

```
<?php echo "First is: ", reset($arr), "<br>"; ?>
```

**Пример навигации по массивам:**

```
<?php $arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
   echo "Тек_элемент: ", current($arr), "<br>";
   echo "След_элемент:  ", next($arr), "<br>";
   echo "Предыдущ_элемент: ", prev($arr), "<br>";
   echo "Последний_элемент: ", end($arr), "<br>";
   echo "Первый_элемент: ", reset($arr), "<br>";
?>
```

Отображение в браузере:

Тек_элемент:  PHP
След_элемент: HTML
Предыдущ_элемент: PHP
Последний_элемент:  CSS
Первый_элемент: PHP

# Преобразование строк в массивы и наоборот

PHP умеет преобразовывать данные из строки в массив и наоборот, для этого в PHP есть функция implode и explode.
implode - формирует строку из массива.
explode - формирует массив из строки.

Использование функции implode:

```
<?php
$arr[0] = "PHP"; $arr[1] = "HTML"; $arr[2] = "CSS";
$string = implode(", ", $arr);
echo $string;
?>
```

Отображение в браузере:
PHP,  HTML,  CSS

Использование функции explode:

```
<?php
  $string = "PHP, HTML, CSS";
  $arr = explode(", ", $string);
  print_r($arr);
?>
```

Отображение в браузере:

Array ( [0] => PHP  [1] => HTML  [2] => CSS )

Спасибо за внимание!
Удачи в WEB –
Программировании!

# Используемые источники


- Сайт Pro-web. Продвижение сайтов. Статья о программе PHP DevelStudio. - http://bogat.com.ua/php_develstudio.php
- Сайт DevelStudio. Описание программы PHP DevelStudio – http://develstudio.ru/downloads/
- Сайт PHP5.RU. Основы веб-программирования. – http://www.php5.ru/study/webbasics
- Сайт ITTRAND. Изображение Энди Гутманс, Зив Сураски и Расмус Лердорф – http://ittrend.am/ru/2011/10/04/10-%D0%BE%D1%82%D1%86%D0%BE%D0%B2-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D0%B5%D0%B9-%D0%B2%D0%B5%D0%B1%D0%B0/
- Сайт ZNAMUS. Статья «Язык программирования для начинающих» – http://znamus.ru/page/php_for_beginners