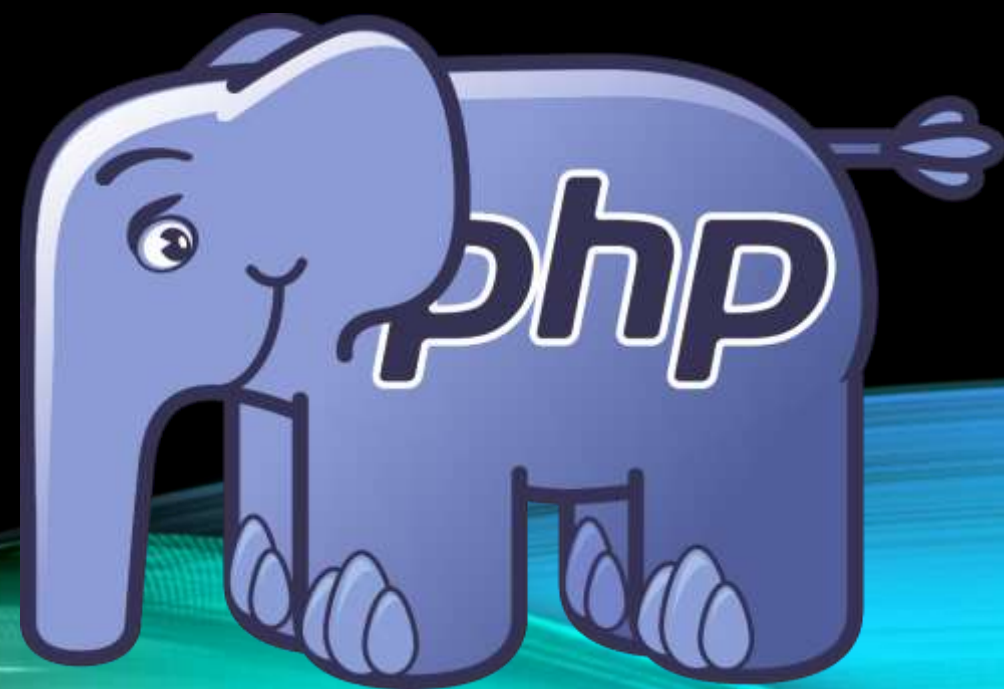


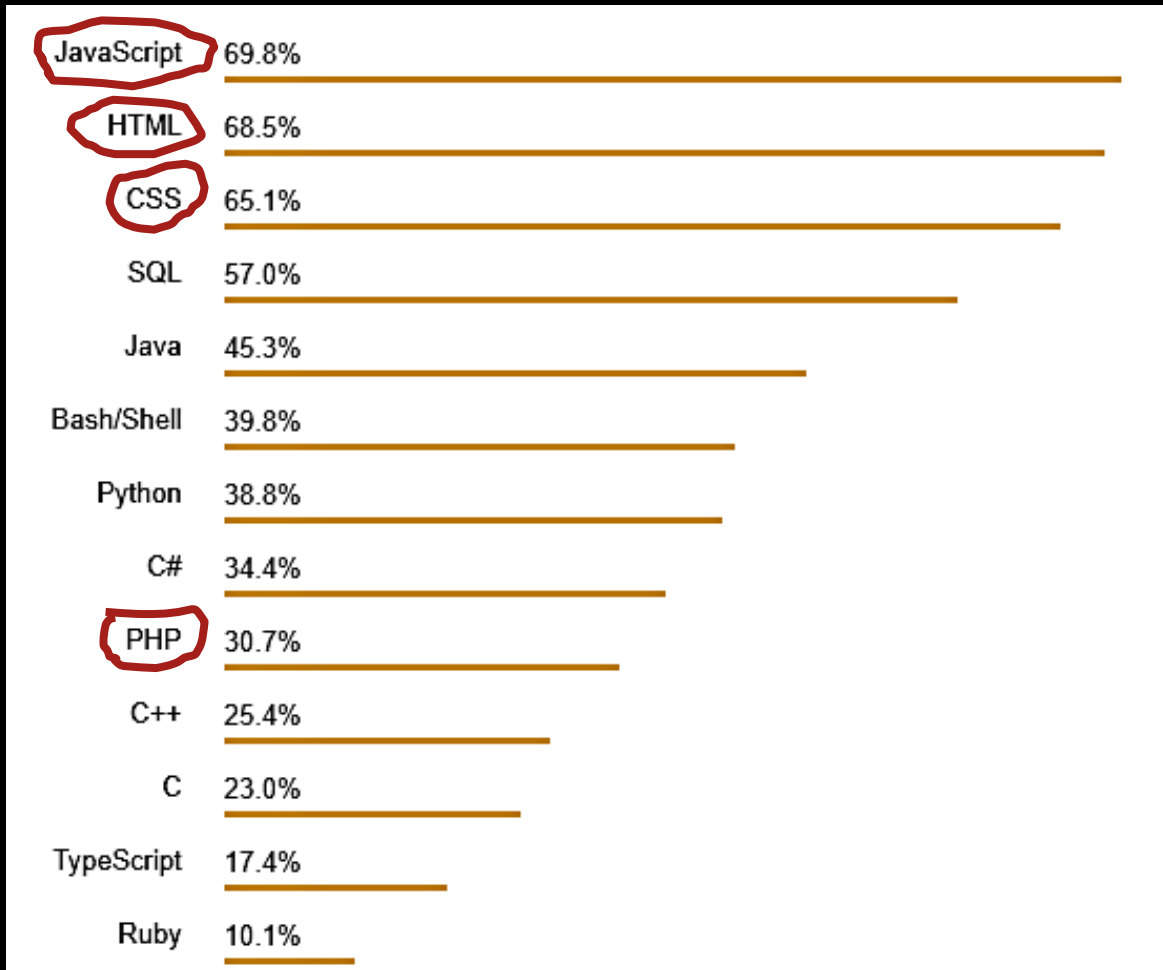
Базы данных

5^{bis}. Язык PHP



- Области применения
 - Скрипты, исполняемые на стороне сервера (классическое применение)
 - Скрипты, выполняющиеся в командной строке (парсер, отладка)
 - GUI – приложения на стороне клиента (PHP-GTK, необычное применение)
- **PHP – мультиплатформенный язык**
- Возможности
 - Генерация HTML (динамические web-страницы)
 - Поддержка различных баз данных и различных протоколов доступа к данным (ODBC, сокет, и пр.)
 - Работа со слабоструктурированными данными (в т.ч. текстовые файлы). Использование регулярных выражений. Генерация XML и PDF.
 - Работа с файловой системой, обработка и отображение файлов различных форматов.
 - Поддержка протоколов LDAP, IMAP, SNMP, NNTP, POP3.
 - Огромное количество расширений и готовых проектов для любых вариантов Интернет-технологий

Рейтинг технологий stackoverflow.com



Индекс популярности TIOBE

	Sep 2019	Sep 2018	Change	Programming Language
1	1	1		Java
2	2	2		C
3	3	3		Python
4	4	4		C++
5	6	6	▲	C#
6	5	5	▼	Visual Basic .NET
7	8	8	▲	JavaScript
8	9	9	▲	SQL
9	7	7	▼	PHP
10	10	10		Objective-C
11	34	34	▲▲	Groovy
12	14	14	▲	Assembly language
13	11	11	▼	Delphi/Object Pascal

Главной особенностью PHP является его практичность, т.к. язык предоставляет программисту средства для быстрого и эффективного решения поставленных задач. Данная особенность PHP обусловлена пятью важными характеристиками:

- Простота
- Гибкость
- Традиционность
- Эффективность
- Безопасность

Расмус Лердорф (Rasmus Lerdorf) — датский программист, написавший в 1994 году набор скриптов, обрабатывающих шаблоны HTML-документов, позже воплотившийся в интерпретатор языка сценариев PHP, с помощью которого можно было решать различные задачи веб-приложений, включая создание сайтов для различных целей и направлений.



Актуальная на 2021 год стабильная версия – **PHP 8.0**

Программа на php встраивается в HTML-документ

```
<?php  
echo "Это обычная инструкция для обработки PHP";  
?>
```

```
<?  
echo "Простейшая инструкция";  
?>
```

```
<script language="php">  
echo "Некоторые редакторы (FrontPage) предпочитают делать так";  
</script>
```

```
<%  
echo "Можно использовать теги в стиле ASP (но не нужно) ";  
// однострочный комментарий  
# тоже вариант однострочного комментария  
/* многострочный  
   комментарий */  
%>
```

Расширение файла с PHP-кодом зависит от настроек web-сервера:

- `example.php`
- `example.phtml`

Переменные и константы

Переменная в PHP обозначается знаком доллара, за которым следует её имя (`$my_var`). Правильное имя переменной должно начинаться с буквы или символа подчёркивания.

Имя переменной чувствительно к регистру.

```
<?php
    $first = ' Text ';
    $second = $first;
    echo "Значение second =" . $second;
    $zzz = & $first; // по ссылке
    $first = ' New text ';
    echo "Значение zzz =" . $zzz;
?>
```

Константы типа *boolean*: `True`, `False`
`echo (int)TRUE` даст в результате 1

```
<?php
    define("PASSWORD", "qwerty");
    // определяем константу PASSWORD
    define("PI","3.14", True);
    // регистронезависимая константа PI
    echo (PASSWORD);
    // выведем значение константы: qwerty
    echo pi;
    // выведет 3.14, т.к. PI регистронезависима
?>
```

+	Сложение	$\$c = \$a + \$b$
-	Вычитание	$\$c = \$a - \$b$
*	Умножение	$\$c = \$a * \$b$
/	Деление	$\$c = \$a / \$b$
%	Остаток от деления	$\$c = \$a \% \$b$
.	Конкатенация (сложение строк)	$\$c = \$a . \$b$
>=	Больше или равно	$\$c = \$a >= \$b$

=	$\$a = (\$b = 4) + 5;$ ($\$a$ будет равна 9, $\$b$ будет равна 4)
+=	$\$a += 5;$ // эквивалентно $\$a = \$a + 5;$
.=	$\$b = "Привет";$ $\$b .= "всем";$ (эквивалентно $\$b = \$b . "всем";$) В результате: $b = "Привет всем"$
&&	$\$c = \$a \&\& \$b;$ // логическое «И»
	$\$c = \$a \$b;$ // логическое «ИЛИ»
==	$\$a == \b // условие равенства значений
===	$\$a === \b // условие равенства значений и типов
++	$++\$a$ // Пре-инкремент $\$a++$ // Пост-инкремент

Скалярные:

- boolean (логический)
- integer (целый)
- float (с плавающей точкой)
- string (строковый)

Смешанные:

- array (массив)
- iterable (псевдотип)
- object (экземпляр класса)

Специальные:

- resource (ресурс)
- NULL
- callable (функция обр. вызова)

```
<?php
$a = 1234;    //десятичное число
$a = -123;   //отрицательное число
$a = 0123;   //восьмеричное число
$a = 0x1A;   //шестнадцатеричное число

$b = 1.234;  // по умолчанию «двойная точность»
$b = 1.2e3;
$b = 7E-10;

$str1 = 'Строка 1 '; // ограничения на размер нет
$str2 = "Строка 2 ";
$str3 = <<<EOD
Третья строка - это $str1 и $str2
EOD;
?>
```


Тип данных - массив

Определить массив можно либо с помощью конструкции `array()`, либо непосредственно задавая значения его элементам.

```
<?php
$books = array ("php" => "PHP users guide", 12 => True);
echo $books["php"]; //выведет "PHP users guide"
echo $books[12]; //выведет 1
?>
```

```
<?php
// обычный вариант одномерного массива
$A[0]="Апельсин";
$A[1]="Банан";
$A[2]="Груша";
$A[3]="Помидор";
echo $A[3]
?>
```

```
array (key1 => value1, key2 => value2, ... )
```

т.н. ассоциативный массив

```
<?php
$books[12]= 333;
$books[] = "test"; // ключ будет равен 13
unset($books[12]); //удаляет элемент № 12
unset($books); //удаляет массив полностью
?>
```

```
<?php
// Многомерный ассоциативный массив
$B["Ivanov"] = array("name"=>"Иванов И.И.", "age"=>"25";
$B["Petrov"] = array("name"=>"Петров П.П.", "age"=>"34";
$B["Sidorov"] = array("name"=>"Сидоров С.С.", "age"=>"47";
?>
```

Сложение массивов

<?

```
$a = array("и"=>"Информатика", "м"=>"Математика");  
$b = array("и"=>"История", "м"=>"Магия", "ф"=>"Физика");  
$c = $a + $b;  
$d = $b + $a;  
print_r($c); // получим: Array([и]=>Информатика [м]=>Математика [ф]=>Физика)  
print_r($d); // получим: Array([и]=>История [м]=>Магия [ф]=>Физика)
```

?>

Равенство массивов (==) – это когда совпадают все пары ключ / значение элементов массивов. Эквивалентность (===) – когда кроме равенства значений и ключей элементов требуется, чтобы элементы в обоих массивах были записаны в одном и том же порядке.

<?php

```
// Простой способ инициализации массива  
$names []="Апельсин";  
$names []="Банан";  
$names []="Груша";  
$names []="Помидор";
```

?>

Тип данных - объект

Класс – это набор объектов, обладающих определенными свойствами и методами работы с ним, а объект – экземпляр соответствующего класса.

```
<?php
```

```
class Person // Создаем класс
{
    function info()
        //Объявляем метод данного класса
        { echo "Это PHP-программист"; }
}

$bob = new Person; // Создаем объект класса
$bob -> info(); // Вызываем метод класса
}
```

```
?>
```

В PHP для доступа к методам объекта используется оператор `->`. Для инициализации объекта используется выражение `new`, создающее в переменной экземпляр объекта.

Объект = `new` Имя_класса;

```
function Getname() {
    echo $this->name;
}

function Setname($name) {
    $this->name = $name;
}
```

Смешанные типы данных

Ресурс – это специальная переменная, содержащая ссылку на внешний ресурс (например, соединение с базой данных). Ресурсы создаются и используются специальными функциями (например, `mysql_connect()`, `pdf_new()` и т.п.).

Специальное значение **NULL** говорит о том, что переменная не имеет значения.

Переменная считается NULL, если

- ей была присвоена константа NULL (`$var = NULL`);
- ей ещё не было присвоено какого-либо значения;
- она была удалена с помощью `unset()`.

Условные операторы

Оператор **if**

```
<?
$names = array("Иван", "Петр", "Семен");
if ($names[0] == "Иван")
{
    echo "Привет, Ваня!";
    $num = 1;  $account = 2000;
}
if ($num) echo "Появился первый в списке!";
?>
```

```
<?php
$a = 10;
$b = 5;
$c = 2;
if ($a + $b > $c and $c < $a) {
    echo 'Истина';
}
?>
```

```
<?
if ($a > $b) {
    echo "a больше, чем b";
} elseif ($a == $b) {
    echo "a равен b";
} else {
    echo "a меньше, чем b";
}
?>
```

```
<?php
$my = True;
echo $my ? 'Переменная истина' : 'Ложь';
?>
```

Оператор **switch**

```
switch (анализируемое_выражение) {  
  case значение1: блок_действий1 break;  
  case значение2: блок_действий2 break;  
  ...  
  default: блок_действий_по_умолчанию  
}
```

```
<?  
$names = array("Иван", "Петр");  
switch ($names[0]) {  
  case "Иван": echo "Привет, бро!";  
  break;  
  case "Петр": echo "Наше почтение!";  
  break;  
  default: echo "Не знаем, кто вы, $names[0], но всё равно здравствуйте!";  
}  
>
```

```
<?php  
switch ( $i ) {  
  case 0:  
  case 1:  
  case 2:  
    echo "i меньше чем 3, но неотрицательный";  
    break;  
  case 3:  
    echo "i равно 3";  
  }  
?>
```

ЦИКЛЫ

Цикл **for**

```
for (выражение1; выражение2; выражение3) {  
    блок_выполнения }
```

Выражение1 вычисляется безусловно один раз в начале цикла. В начале каждой итерации вычисляется выражение2. Если оно является True, то цикл продолжается и выполняются все команды блока выполнения. Если выражение2 вычисляется как False, то исполнение цикла останавливается. В конце каждой итерации (т.е. после выполнения всех команд блока выполнения) вычисляется выражение3.

```
<?php  
    for ($i = 0; $i < 10; $i++) {  
        if ($i % 2 == 0) print $i;  
    }  
?>
```

```
<?php  
    for ($i=0, $j=0, $k="Точки"; $i<10; $j++, $i+=$j) {  
        $k = $k."."; echo $k;  
    }  
    // Выводит Точки.Точки..Точки...Точки....  
?>
```

ЦИКЛЫ

```
<?
  $i = 1;
  while ($i < 10) {
    if ($i % 2 == 0) echo $i;
    $i++;
  }
?>
```

Значение выражения проверяется каждый раз в начале цикла, так что, даже если его значение изменилось в процессе выполнения блока, цикл не будет остановлен до конца итерации

```
<?
  $i = 1;
  do {
    if ($i % 2 == 0) print $i;
    $i++;
  } while ($i < 10)
?>
```

Истинность выражения проверяется в конце цикла. Благодаря этому блок цикла do...while гарантированно выполняется хотя бы один раз.

```
<?
  $arr = array(1, 2, 3, 4);
  foreach ($arr as &$value) {
    $value = $value * 2;
  }
  unset($value);
  // надо разорвать ссылку
  // на последний элемент
?>
```

Foreach работает только с массивами и объектами

Цикл **foreach**

```
foreach ($array as $value) {блок_выполнения}  
foreach ($array as $key => $value) {блок_выполнения}
```

В первом случае формируется цикл по всем элементам массива, заданного переменной \$array. На каждом шаге цикла значение текущего элемента массива записывается в переменную \$value, и внутренний счетчик массива передвигается на единицу. Выполнение блока выполнения происходит столько раз, сколько элементов в массиве \$array.

Во второй форме записи на каждом шаге цикла ключ текущего элемента массива записывается в переменную \$key, которую можно использовать в блоке выполнения.

```
<?php  
    $a = array(  
        "one" => 1,  
        "two" => 2,  
        "three" => 3,  
        "seventeen" => 17  
    );  
  
    foreach ($a as $k => $v) {  
        echo "\$a[$k] содержит $v.\n";  
    }  
?>
```

В языке программирования PHP существует 2 вида функций:

1. Функции, определяемые пользователем
2. Внутренние (встроенные) функции

```
function Имя_функции (параметр1, параметр2, ... параметрN) {  
    Блок_действий  
    return возвращаемое_значение;  
}
```

С помощью аргументов данные в функцию можно передавать тремя различными способами:

- по значению
- по ссылке
- значения по умолчанию

```
<?php  
// рекурсивная функция вычисления факториала  
function fact($n) {  
    if ($n==0) return 1;  
    else return $fact = $n * fact($n-1);  
}  
  
echo 'Факториал 5 равен ' .fact(5). '<br>';  
echo 'Факториал 50 равен ' .fact(50). '<br>';  
?>
```

Примеры

```
<?php
function sayHello($Name){
    echo "Привет, ". $Name . "!<br>";
}
?>
```

```
<?php
function makesoffee($type = "капучино")
{
    return "Готовим чашку $type.\n";
}

echo makesoffee(); // по умолчанию
echo makesoffee(null);
echo makesoffee("эспрессо");
?>
```

```
<?php
function mySum($numX, $numY) {
    $total = $numX + $numY;
    return $total;
}

$myNumber = 0;
$myNumber = mySum(3, 4);
echo "myNumber = " . $myNumber . "<br>";
?>
```

```
<?php
function fuf( &$string )
{
    $string .= 'а эта - внутри.';
}

$str = 'Эта строка за пределами функции, ';
fuf( $str );
echo $str;
?>
```

Ещё несколько примеров

```
<?php
// Использование позиционных аргументов:
X_fill(0, 100, 50);

// Использование именованных аргументов:
X_fill(start_index: 0, count: 100, value: 50);
?>
```

```
<?php
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
?>
```

Возврат нескольких значений в виде массива

```
<?php
function sum(...$numbers) {
    $acc = 0;
    foreach ($numbers as $n) {
        $acc += $n;
    }
    return $acc;
}
```

```
echo sum(1, 2, 3, 4);
?>
```

Список аргументов может содержать многоточие ..., чтобы показать, что функция принимает переменное количество аргументов. Аргументы в этом случае будут переданы в виде массива.

- В PHP содержится достаточно большое количество встроенных функций и языковых конструкций. Также есть функции, которые требуют, чтобы PHP был собран с определенными расширениями, в противном случае будут генерироваться фатальные ошибки, вызванные использованием неизвестной функции.
- Например, для того чтобы использовать функции для работы с изображениями, например, `imagecreatetruecolor()`, необходимо собрать PHP с поддержкой `GD`.
- Или же для того, чтобы воспользоваться функцией `mysqli_connect()`, необходима поддержка модуля `MySQLi`. Тем не менее, есть много встроенных функций, которые доступны всегда: например функции обработки строк и математические функции.
- Вызвав специальные функции `phpinfo()` или `get_loaded_extensions()`, можно узнать, поддержка каких модулей есть в используемом PHP.

Функции для работы с переменными

- `empty()` — Проверяет, пуста ли переменная
- `gettype()` — Возвращает тип переменной
- `is_array()` — Определяет, является ли переменная массивом
- `is_bool()` — Проверяет, является ли переменная булевой
- `is_int()` — Проверяет, является ли переменная целым числом
- `is_null()` — Проверяет, является ли значение переменной равным NULL
- `isset()` — Определяет, была ли установлена переменная значением, отличным от NULL
- `unset()` — Удаляет переменную
- `var_dump()` — Выводит информацию о переменной

Примеры функций для работы с массивами

```
<?
$del_items = array("langs" => array(
    "10"=>"Python", "12"=>"Lisp"),
    "other"=>"Информатика");

echo count($del_items) . "<br>";
    // выведет 2
echo count($del_items, COUNT_RECURSIVE);
    // выведет 4
?>
```

```
<?
$items = array(10 => "хлеб", 20 => "молоко",
    30 => "бутерброд");
sort($items);
// [0] => бутерброд [1] => молоко [2] => хлеб
print_r($items);
?>
```

Пример использования функции `count` для вычисления количества элементов массива. Фактически элементов массива 2, а именно пары ключ-значение:

"langs" => array() и "other"=>"Информатика". Однако, если учесть, что ключ "langs" содержит массив из двух элементов, то общее количество элементов будет равно 4.

Функция `sort` упорядочивает значения массива по возрастанию. Эта функция удаляет все существовавшие в массиве ключи, заменяя их числовыми индексами, соответствующими новому порядку элементов. В случае успешного завершения работы она возвращает true, иначе – false.

Примеры функций для работы со строками

Для того чтобы определить, входит ли подстрока в состав строки, используется функция `strpos()`

```
<?
$str = "Идея наносить данные на перфокарты
и затем считывать и обрабатывать их
автоматически принадлежала Джону Биллингсу,
а ее техническое решение осуществил Герман
Холлерит. Перфокарта Холлерита оказалась
настолько удачной, что без малейших изменений
просуществовала до наших дней.";
```

```
    $pos = strpos($str, "Холлерит");
    if ($pos !== false) echo "Искомая строка
        встречена в позиции номер $pos ";
    else echo "Искомая строка не найдена";
?>
```

Подстроку из строки можно выделить с помощью функции `substr()`

```
<?php
$word = "say <b>Hello, world!</b><br>";
echo $word
$pure_str = substr($word, 7, -8);
echo $pure_str;
?>
```

`str_replace` — Заменяет все вхождения строки поиска на строку замены

`substr_replace` — функция заменяет часть строки строкой, предназначенной для замены

PHP в настоящее время реализует полноценную объектную модель (классы и экземпляры классов – объекты, конструкторы и деструкторы, инкапсуляция, наследование, перегрузка, абстрактные классы, интерфейсы объектов). Для создания экземпляра класса используется директива `new`. Класс может содержать собственные константы, переменные (называемые свойствами) и функции (называемые методами).

```
<?php
class MyClass
{
    // объявление свойства
    public $var = 'значение по умолчанию';

    // объявление метода
    public function displayVar() {
        echo $this->var;
    }
}

$s = new MyClass();
$s->displayVar();
?>
```

```
<?php
class ExtendClass extends MyClass
{
    // Переопределение метода родителя
    function displayVar()
    {
        echo "Расширенный класс\n";
        parent::displayVar();
    }
}

$ext = new ExtendClass();
$ext->displayVar();
?>
```

```
<?php
```

```
class User
{
    public int $id;
    public ?string $name;

    public function __construct(int $id, ?string $name)
    {
        $this->id = $id;
        $this->name = $name;
    }
}
```

```
$user = new User(1234, null);
```

```
var_dump($user->id);
var_dump($user->name);
```

```
?>
```

PHP позволяет объявлять методы-конструкторы. Классы, в которых объявлен метод-конструктор, будут вызывать этот метод при каждом создании нового объекта, так что это может оказаться полезным, например, для инициализации какого-либо состояния объекта перед его использованием.

В PHP объектная переменная не содержит сам объект как значение. Такая переменная содержит только идентификатор объекта, который позволяет найти конкретный объект при обращении к нему.

Язык PHP — это удобный инструмент, который всё ещё продолжает развиваться, имеет большую поддержку в Интернет-сообществе, огромное количество готовых библиотек и решений. Немаловажно, что он прекрасно документирован на разных языках.

Идея работы с PHP — обладая нулевым уровнем подготовки быстро стартовать, без проблем сформировать хорошую команду и сделать продукт «не хуже чем у других».

Язык PHP имеет массу проблем, в том числе и на уровне идеологии. Считать его современным перспективным языком сейчас никто не станет. Но и игнорировать его существование пока ещё рано.



YOUR ATTENTION

I THANK YOU FOR