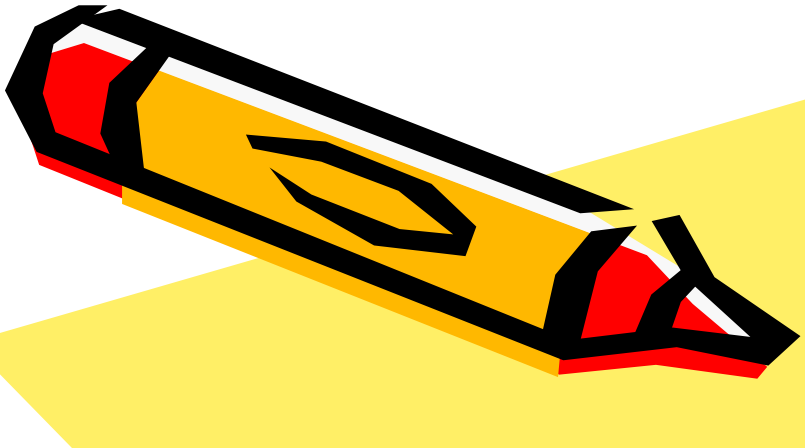
Курс

«Программирование в компьютерных сетях»

Лекция 8

Приходько Татьяна Александровна
доцент кафедры
Вычислительных технологий КубГУ

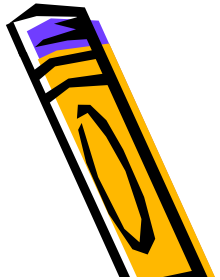


Лекция 8
PHP

Часть первая

История и синтаксис языка

Выбор технологий (и языка) для WEB-проекта



1. ***PHP*** — его используют в основном для простых и средних проектов. Очень много коробочных решений. Относительно дешевые программисты. Антитренд последних лет, хотя с выходом последней версии языка под номером 7, он получил действительно мощные возможности.
2. ***Python*** — современный язык, разработка на нем быстрая и качественная. Используют его для средних и больших проектов. Программистов найти проблематично, и стоят они не дешево.
3. ***Ruby*** — современный язык, разработка на нем также быстрая. Его используют в основном для разработки простых и средних проектов, часто разрабатывают стартапы. Программистов также мало, и они дорогие.

Выбор технологий (и языка) для WEB-проекта

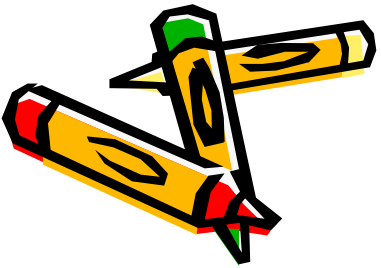
- ***Java*** — разработка на нем очень долгая и дорогая. Его используют в основном для больших проектов со специфическими требованиями.
- ***C#*** — аналог Java, также используют для больших проектов, часть в сфере FinTech.
- ***JS*** — очень быстро развивается, тренд последних лет. Огромное количество наработок, и можно писать все, что угодно, даже игры. Его используют для средних и больших проектов, но действительно мощные возможности этот язык получил недавно, потому примеров больших проектов пока мало, специалисты самые дорогие, и найти их сложнее всего.

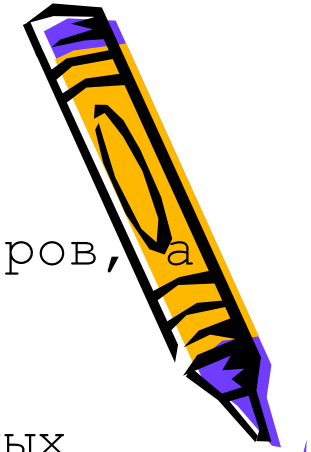
https://habr.com/ru/company/SECL_GROUP/blog/315734/

PHP

расшифровывается как *Hypertext Preprocessor* — «PHP: препроцессор гипертекста»

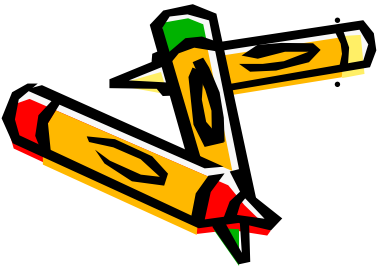
Тип исполнения: Интерпретатор компилирующего типа
Появился в:1994 г.
Автор: Расмус Лердорф
Релиз: 7.0 (декабрь 2015)
Предыдущая версия:5.4 (2011)
Испытал влияние:Perl, C, C++, Java



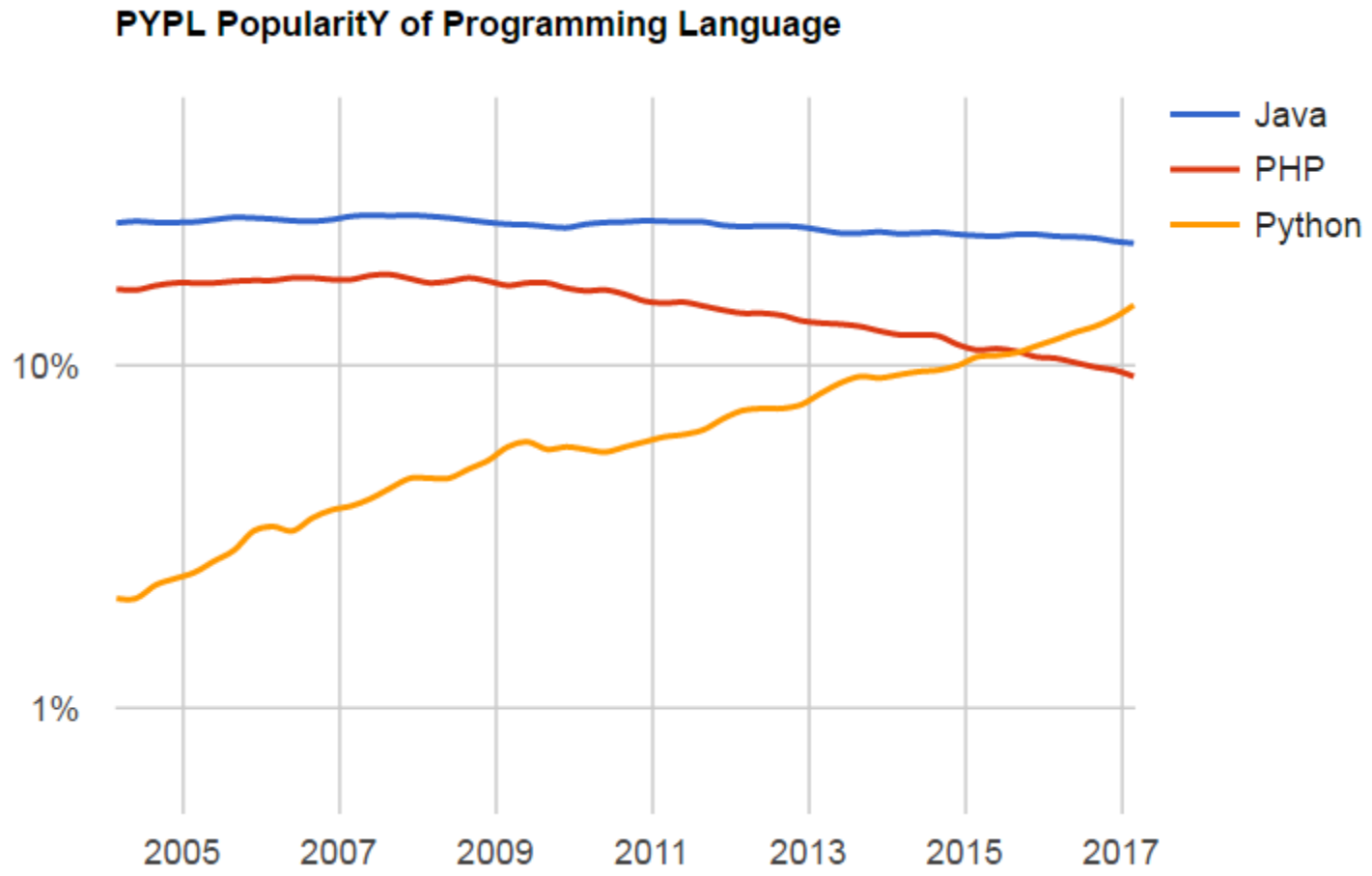


Возможности:

- автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
- взаимодействие с большим количеством различных систем управления базами данных (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape и Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase и др…);
- автоматизированная отправка HTTP-заголовков;
- работа с HTTP-авторизацией;
- работа с cookies и сессиями;
- работа с локальными и удалёнными файлами, сокетами.
- обработка файлов, загружаемых на сервер;
- работа с XForms;
- создание GUI-приложений.

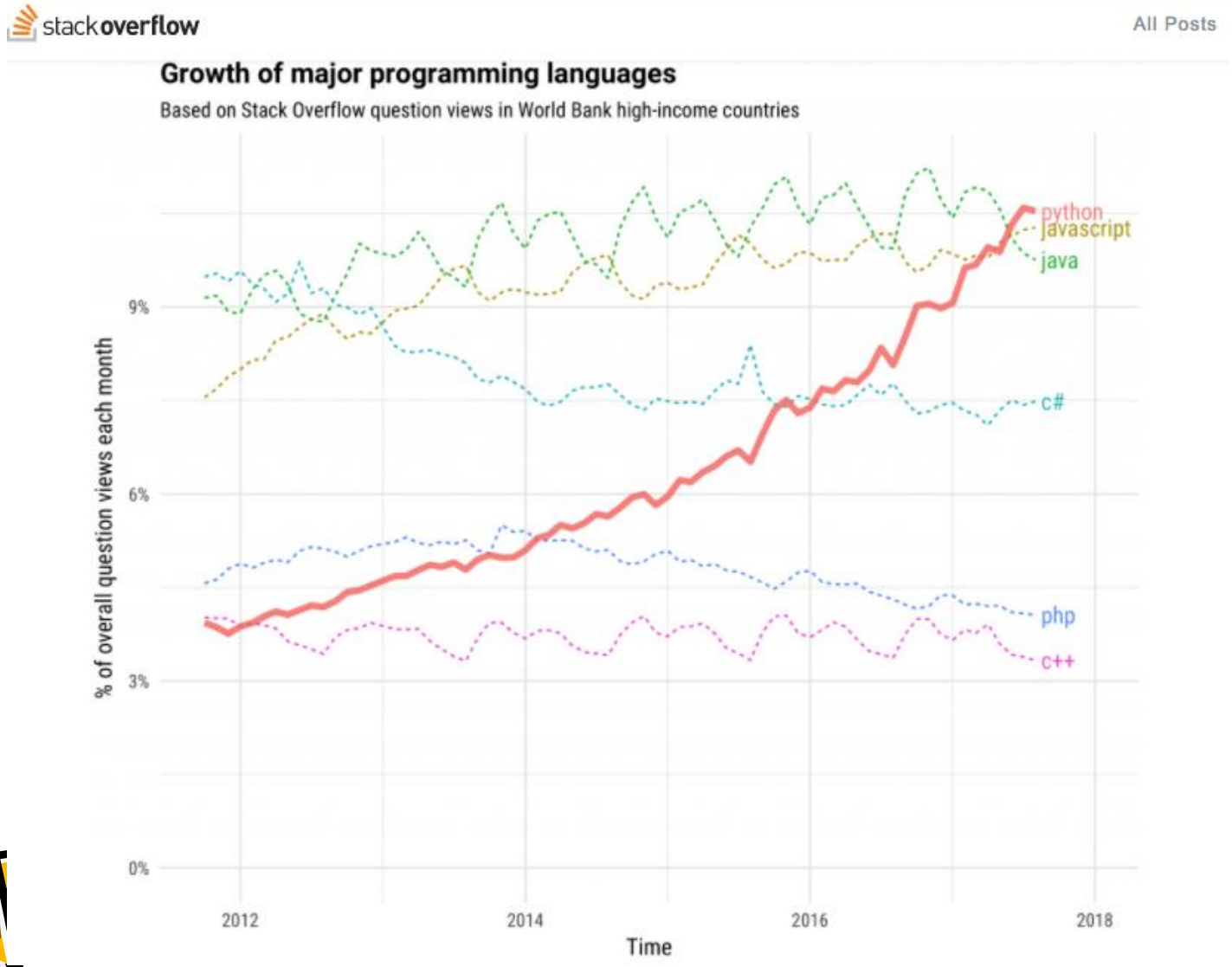


WEB languages



http://itmentor.by/articles/top-10-yazykov-programmirovaniya-v-2017-godu-po-versii-github

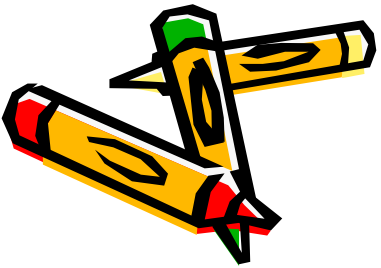
WEB languages



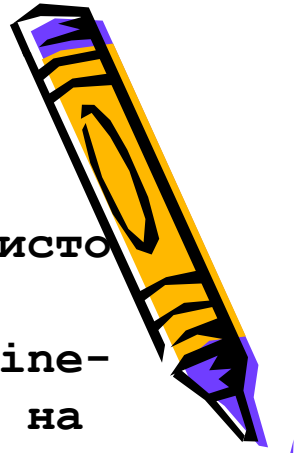
Примеры больших сайтов:

PHP: Facebook, Вконтакте, КиноПоиск

- Python: Instagram, Pinterest, Reddit
- Ruby: 500px, Groupon, Airbnb
- Java: Ebay, Amazon, Alibaba
- C#: Guru, Stack Overflow, Bank of America
- JS: LinkedIn, Walmart, PayPal

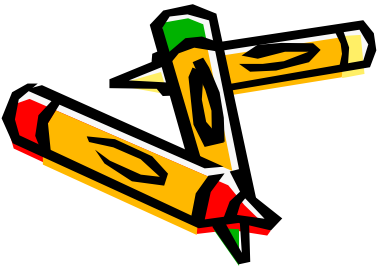


История



Язык PHP был разработан как инструмент для решения чисто практических задач в 1994г. Его создатель, Расмус Лердорф, хотел знать, сколько человек читают его online-резюме, и написал для этого простенькую CGI-оболочку на языке Perl, т.е. это был набор Perl-скриптов, предназначенных исключительно для определенной цели – сбора статистики посещений.

Для справки. CGI (Common Gateway Interface – общий интерфейс шлюзов) - стандарт для создания серверных приложений, работающих по протоколу HTTP.
Такие приложения (шлюзы или CGI-программы) запускаются сервером в режиме реального времени. Сервер передает запросы пользователя CGI-программе, которая их обрабатывает и возвращает результат своей работы на экран пользователя.



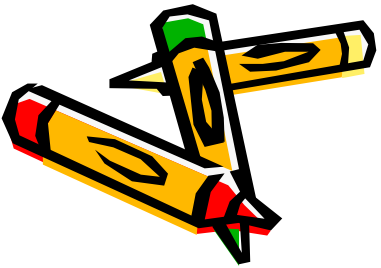
Таким образом, посетитель получает динамическую информацию, которая может изменяться в результате влияния различных факторов. Сам шлюз (скрипт CGI) может быть написан на различных языках программирования – Си/C++, Fortran, Perl, TCL, UNIX Shell, Visual Basic, Python и др.

История

1997г. – PHP2.0, написанный на C

1998г. – PHP3.0, ядро языка было снабжено дополнительными модулями, что впоследствии дало PHP возможность работать с огромным количеством баз данных, протоколов, поддерживать большое число API. Начался стремительный рост его популярности. С этой версии акроним php расшифровывается как «hypertext Preprocessor», вместо устаревшего «Personal Home Page».

2000г. – PHP 4.0.(С новым движком – Zend Engine в честь израильских программистов Энди Гутманса и Зээва Сураски). В дополнение к улучшению производительности, PHP 4.0 имел ещё несколько ключевых нововведений, таких как поддержка сессий, буферизация вывода, более безопасные способы обработки вводимой пользователем информации.



История

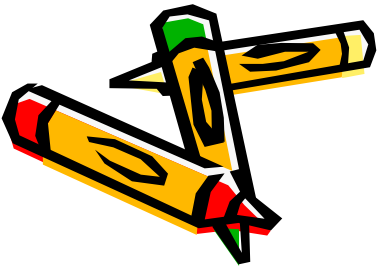
2004г. – PHP 5.0, Введена поддержка языка разметки XML. Полностью переработаны функции ООП, которые стали во многом схожи с моделью, используемой в Java. В частности, введён деструктор, открытые, закрытые и защищённые члены и методы, final члены и методы, интерфейсы и клонирование объектов. В последующих версиях также были введены пространства имён, замыкания и целый ряд других достаточно серьёзных изменений.

Шестая версия PHP разрабатывалась с октября 2006 года. Было сделано множество нововведений, как, например, исключение из ядра регулярных выражений.
В марте 2010 года разработка PHP6 была признана бесперспективной из-за сложностей с поддержкой Юникода.Основная ветвь разработки стала v 5.4

История

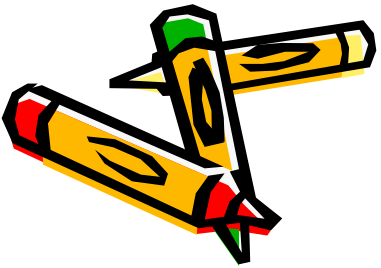
Новая версия ***PHP 7 (2015)*** основывается на экспериментальной ветви PHP, которая изначально называлась ***phpng*** (PHP Next Generation), и разрабатывалась с упором на увеличение производительности и уменьшение потребления памяти.

В новой версии добавлена возможность указывать тип возвращаемых из функции данных, добавлен контроль передаваемых типов для скалярных данных, а также новые операторы.



Отличие PHP от JavaScript

PHP-скрипт выполняется на сервере, а клиенту передается результат работы, тогда как в JavaScript-код полностью передается на клиентскую машину и только там выполняется.



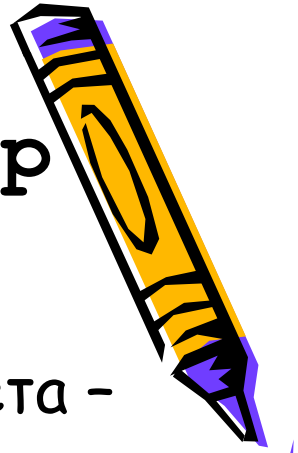
Необходимое ПО

Для работы с PHP нам нужно установить **web-сервер и интерпретатор PHP.** В качестве web-сервера часто выбирают Apache – он наиболее популярен среди web-разработчиков. Для просмотра результатов работы программ нам понадобится web-браузер. Для работы с базами данных часто устанавливают MySQL.

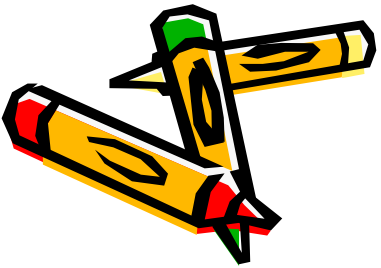
PHP - фреймворки

Yii2, Laravel, Zend и Symfony на данный момент соответствуют современным стандартам и требованиям. Они пользуются спросом и имеют огромную функциональность. Для изучения этих фреймворков, нужно иметь представление о MVC, хорошо знать PHP, включая ООП, и уметь работать с базами данных. Новичкам лучше начинать с освоения Laravel и Yii, а не Symfony или Zend.

Зачем нужен локальный web-сервер



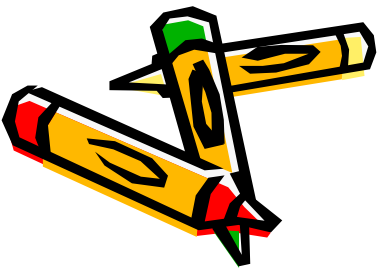
- Можно, конечно, использовать сервер Интернета – выбрать хостинг с поддержкой PHP, загрузить туда свой сайт, и, можно работать.
- Но, очень часто, это неудобно. Например, если вы создаете учебный сайт и в дальнейшем не собираетесь загружать его в Интернет.
- Неудобно работать с таким сервером и в том случае, если скорость подключения к Интернету мала. Вот тогда вам и нужен локальный сервер.



Зачем нужен локальный web-сервер

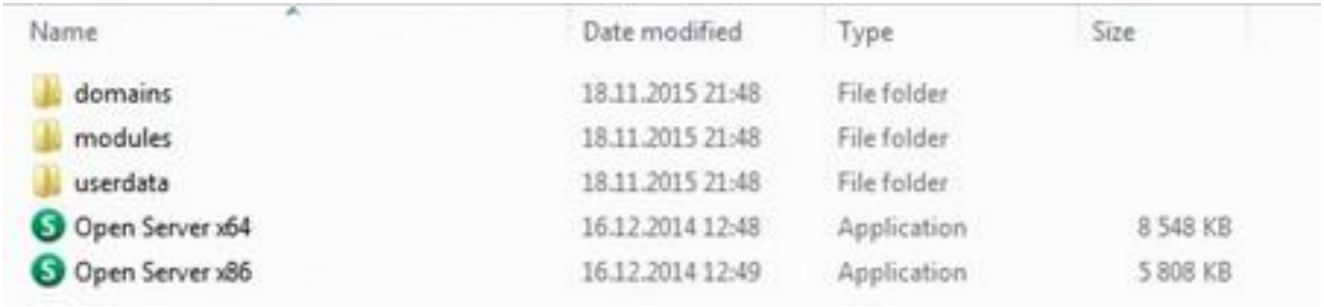
- Можно создать сервер самостоятельно. Для этого понадобиться скачать сервер Apache, пакеты PHP и MySQL, установить их на свой компьютер и начать работать.
- Можно использовать готовые установочные пакеты, которые содержат все необходимое для полноценной работы сервера, например:

  - Open Server https://ospanel.io/
  - Сервер AppServ
  - Сервер Apache Swissknife
  - Сервер XAMPP
  - Локальный сервер Denver

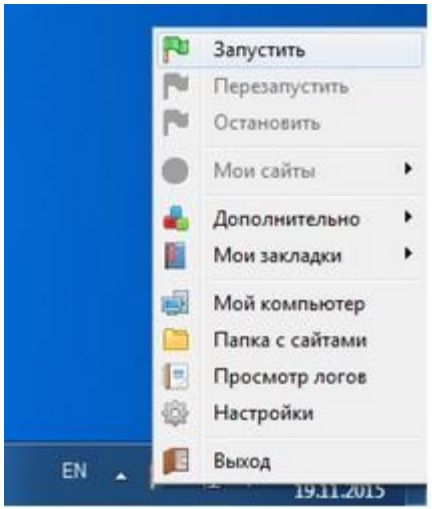


Установка Open Server

1. Рекомендуется скачать полную версию (максимальная редакция) – там много полезных программ для разработчиков.
2. После распаковки:



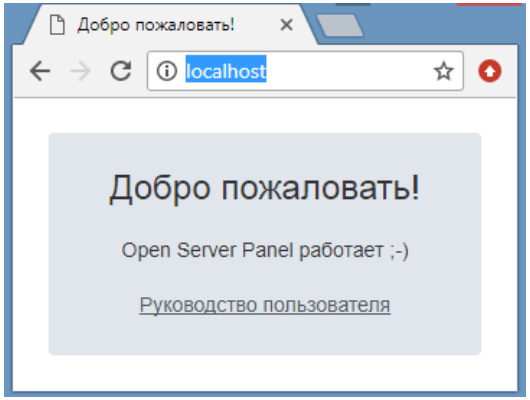
3. После инсталляции:

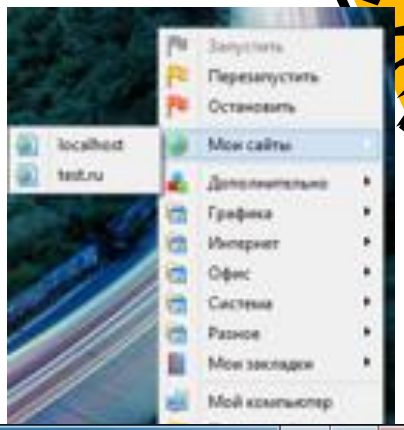


Здесь подробно об установке, настройке и первом запуске: https://www.youtube.com/watch?v=QyjRcJVTEhk

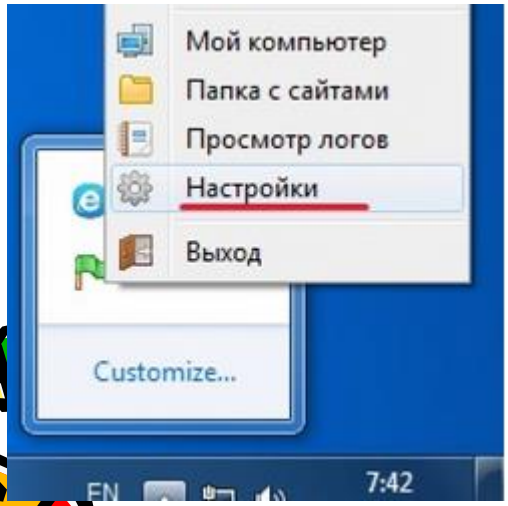
Запуск Open Server

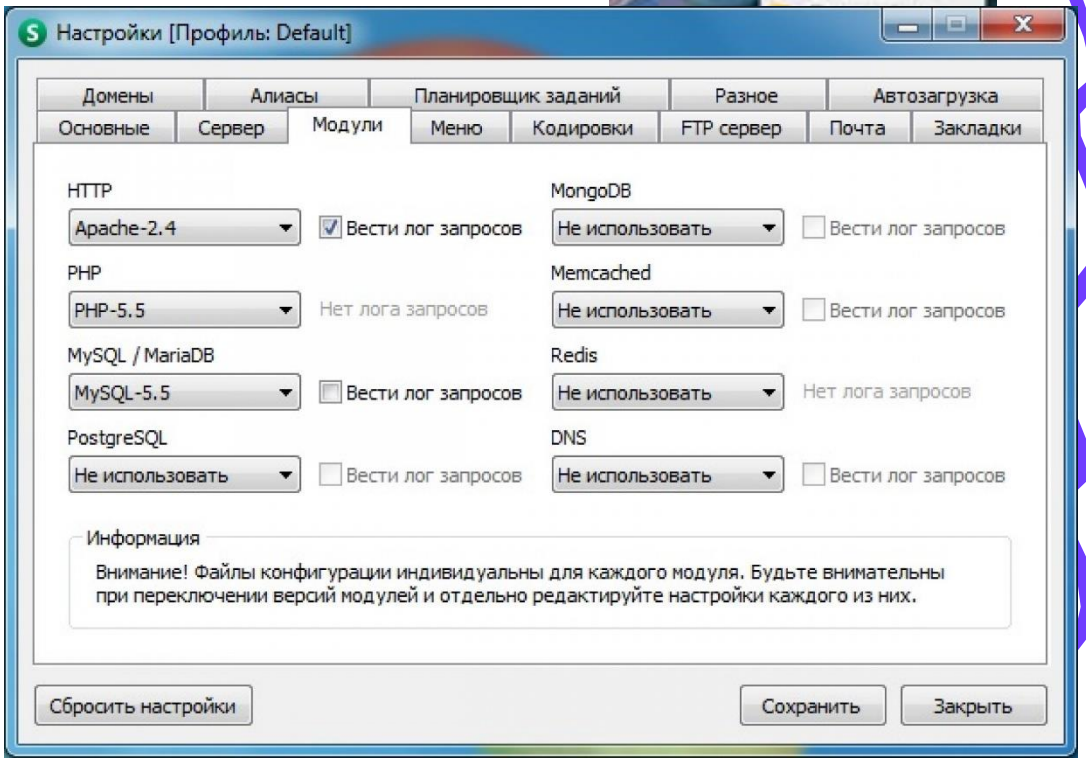
4. Проверяем работу запуском localhost:





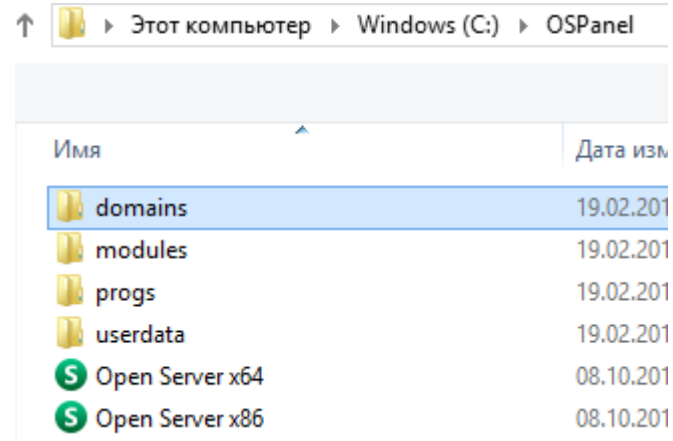
5. Из настроек загляните сюда:



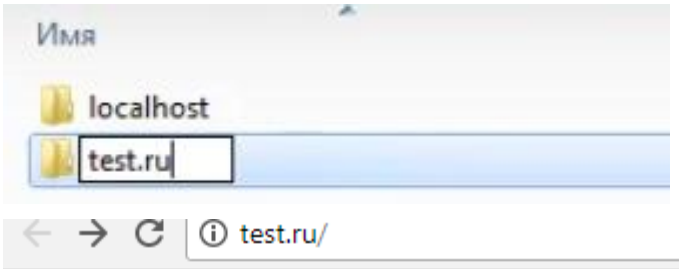


Создание нового домена

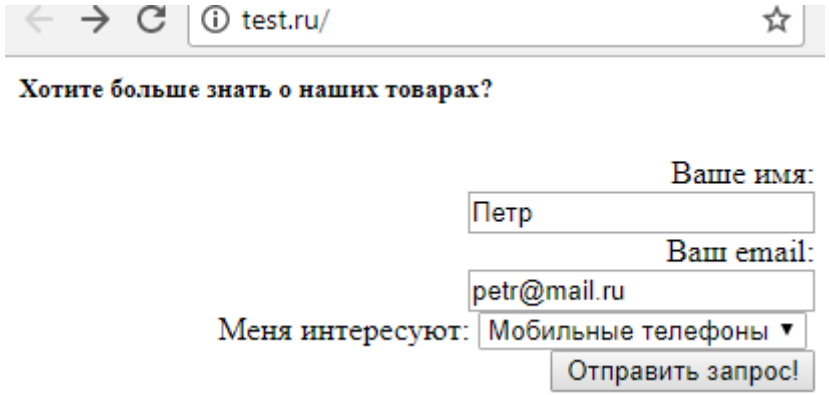
1. Находим, где у нас лежит сервер, заходим в папку domains:



2. Создаем новую директорию – это и будет новый домен



3. В ней – новый файл: index.php



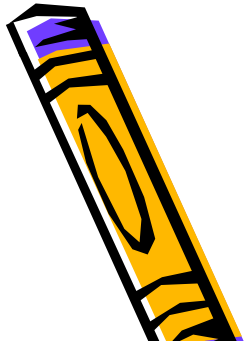
4. Перезапустите сервер.

Создание нового Web-приложения

index.php

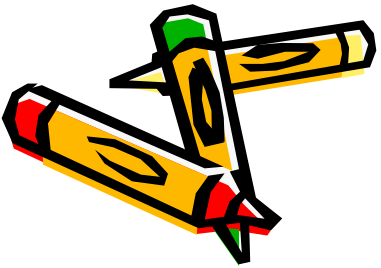
```
<body>
<h5>Хотите больше знать о наших товарах? </h5>
<table width = 400><tr><td align = right>
  <form action="email1.php" method="POST">
Ваше имя:<br>
<input type="text" name="name" size="20" maxlength="30"><br>
Ваш email:<br>
<input type="text" name ="email" size="20" maxlength
="30"><br>
Меня интересуют:
       <select name="preference">
   <option value = “Компьютеры"> Компьютеры</option>
   <option value = “Мобильные_телефоны"> Мобильные телефоны
       </option>
       </select> <br>
<input type ="submit" value="Отправить запрос!">
  </form>
</td></tr></table></body>
```

Создание нового Web-приложения



email1.php

```
<?
echo "<!DOCTYPE html><html><head><meta charset='utf-8'/>
</head><body>";
  echo '<h5>Этот скрипт получает переменные из
index.html</h5>';
       print "<div text-align:'center'>";
       print "Привет, ".$_POST['name'];
       print "<br>Спасибо за ваш интерес.<br>";
       print "Вас интересуют ".$_POST['preference']."",
 информацию о них мы пошлем вам на email:".$_POST['email'];
print "</div>";
echo "</body></html>";
?>
```



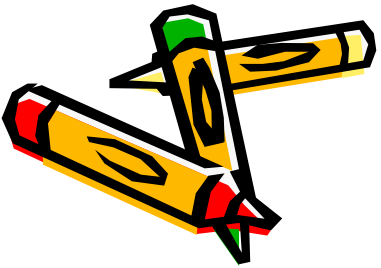
Этот скрипт получает переменные из index.html

Привет, Петр
Спасибо за ваш интерес.
Вас интересуют “Мобильные_телефоны", информацию о них мы пошлем вам на email: petr@mail.ru

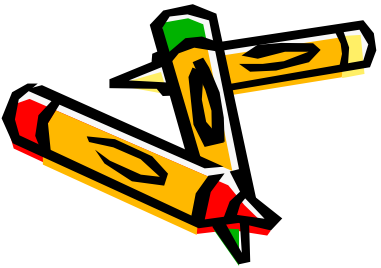
Создание нового Web-приложения

Обратите внимание, что, если в нашем примере в обоих файлах заменить метод ***POST*** Методом ***GET***, то при отправке формы мы увидим вот такую адресную строку:

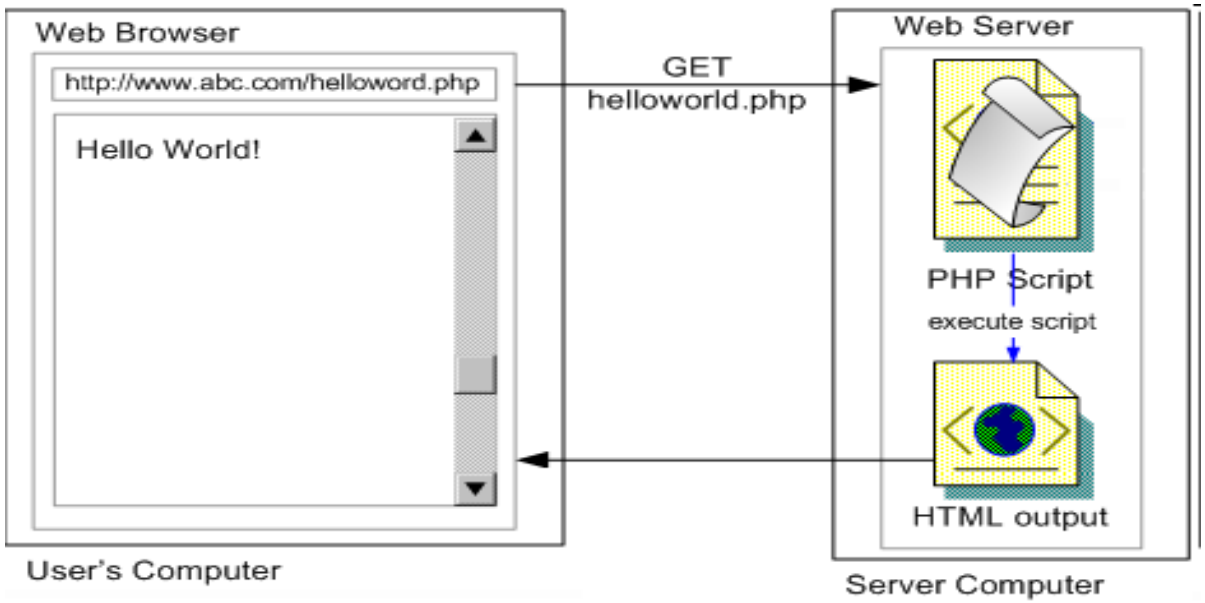
```
http://test.ru/email1.php?name=Петр&email=petr%40mail.ru
&preference="Мобильные_телефоны"
```



HTTP – протокол и PHP



Типичный запрос WEB-сервера с использованием PHP



1. Браузер запрашивает .html- файл (статическое содержимое): сервер только посылает этот файл;
2. Браузер запрашивает .php - файл (динамичное содержимое): сервер читает его, запускает код сценария внутри него, затем посылает результат обратно через сеть;
3. Результатом выполнения сценария становится часть сформированной страницы HTML.

Протокол HTTP и способы передачи данных на сервер

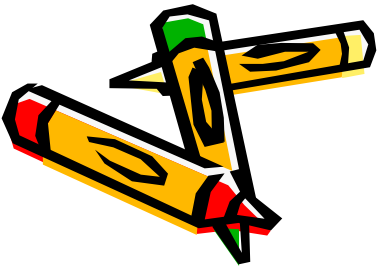
HTTP используется для коммуникаций между различными пользовательскими программами и программами-шлюзами, которые предоставляют доступ к существующим Internet-протоколам, таким как SMTP (протокол электронной почты), NNTP (протокол передачи новостей), FTP (протокол передачи файлов)и др.

HTTP разработан для того, чтобы позволять таким шлюзам через промежуточные программы-серверы (proxy) передавать данные без потерь.

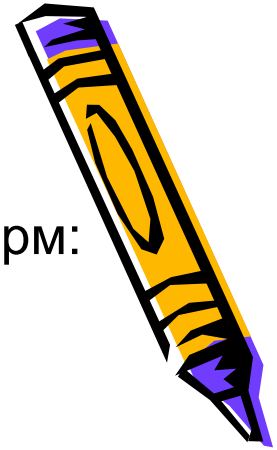
Протокол HTTP и способы передачи данных на сервер

Протокол реализует принцип запрос/ответ. Запрашивающая программа-клиент инициирует взаимодействие с отвечающей программой-сервером, и посылает **запрос**, содержащий:

- метод доступа;
- адрес URL;
- версию протокола;
- сообщение с информацией о типе передаваемых данных, информацией о клиенте, пославшем запрос, и, возможно, с телом сообщения.



Формы запроса клиента

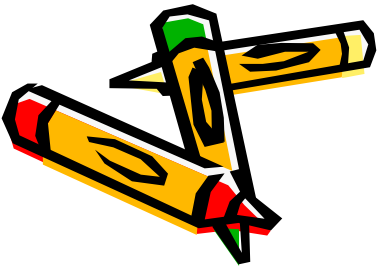


Клиент отсылает серверу запрос в одной из двух форм: в **полной или сокращенной**.

Простой запрос (сокращенный) содержит **метод доступа и адрес ресурса.** Формально это можно записать так:

<Простой-Запрос> := <Метод><символ пробела> <Запрашиваемый-URI><символ новой строки>

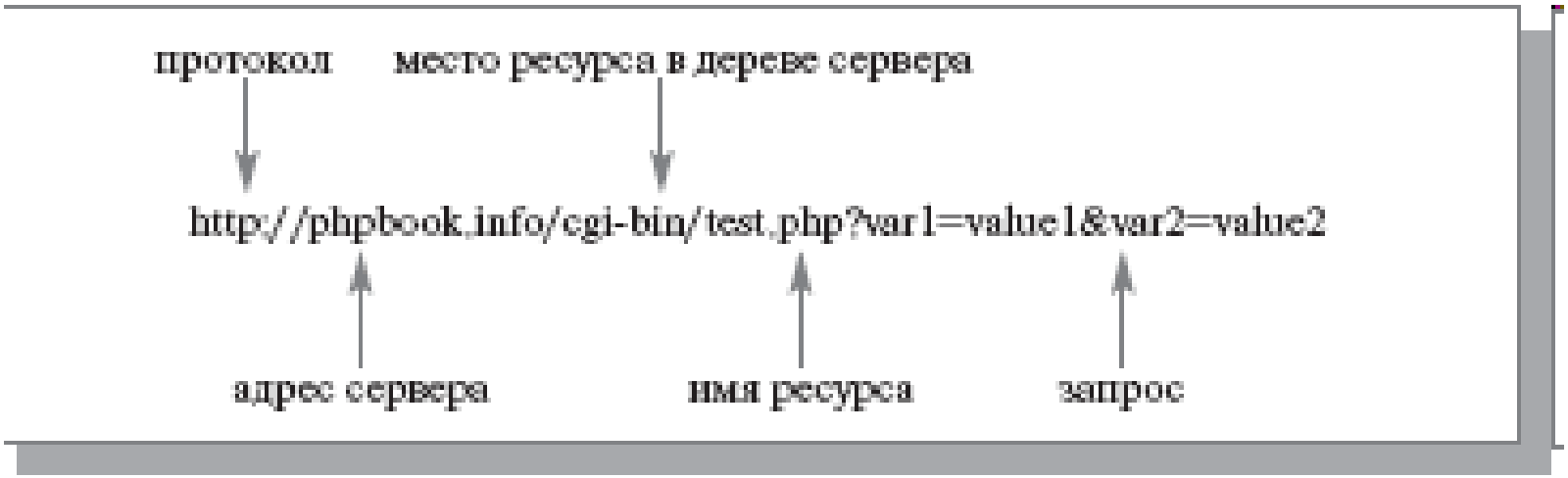
В качестве метода могут быть указаны GET, POST, HEAD, PUT, DELETE и другие.
В качестве запрашиваемого URI чаще всего используется URL-адрес ресурса.



```
Пример простого запроса:
GET http://phpbook.info/
```

Формы запроса клиента

Полная форма запроса содержит **тип протокола доступа, адрес сервера ресурса и адрес ресурса на сервере.** В сокращенной форме опускают протокол и адрес сервера, указывая только местоположение ресурса от корня сервера.

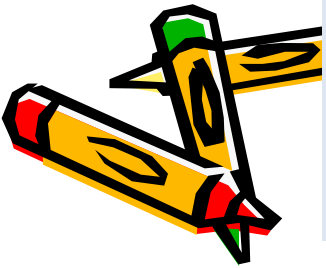


Полную форму используют, если возможна пересылка запроса другому серверу. Если же работа происходит только с одним сервером, то чаще применяют сокращенную форму.

Методы

Протокол HTTP поддерживает достаточно много методов, но реально используются только три: POST, GET и HEAD.

Метод GET позволяет получить любые данные, идентифицированные с помощью URI(Universal Resource Identifier) в запросе ресурса. Если URI указывает на программу, то возвращается результат работы программы, а не ее текст (если, конечно, текст не есть результат ее работы). Дополнительная информация, необходимая для обработки запроса, встраивается в сам запрос (в строку статуса). При использовании метода GET в поле тела ресурса возвращается собственно затребованная информация (текст HTML-документа, например).



Существует разновидность метода GET – условный GET. Этот метод сообщает серверу о том, что на запрос нужно ответить, только если выполнено условие, содержащееся в поле if-Modified-Since заголовка запроса (тело ресурса передается в ответ на запрос, если этот ресурс изменялся после даты, указанной в if-Modified-Since).

Методы

Метод HEAD аналогичен методу GET, только не возвращает тело ресурса и не имеет условного аналога. Метод HEAD используют для получения информации о ресурсе. Это может пригодиться, например, при решении задачи тестирования гипертекстовых ссылок.

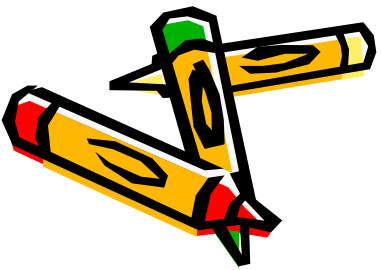
Метод POST разработан для передачи на сервер такой информации, как аннотации ресурсов, новостные и почтовые сообщения, данные для добавления в базу данных, т.е. для передачи информации большого объема и достаточно важной. В отличие от методов GET и HEAD, в POST передается тело ресурса, которое и является информацией, получаемой из полей форм или других источников ввода.

Методы

При отправке данных формы с помощью метода **GET** содержимое формы добавляется к URL после знака вопроса в виде пар имя=значения, объединенных с помощью амперсанда &:

action?name1=value1&name2=value2&name3=value3

Здесь action – это URL-адрес программы, которая должна обрабатывать форму (это либо программа, заданная в атрибуте action тега form, либо сама текущая программа, если этот атрибут опущен). Имена name1, name2, name3 соответствуют именам элементов формы, а value1, value2, value3 – значениям этих элементов.

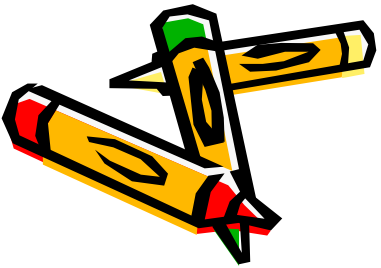


Методы

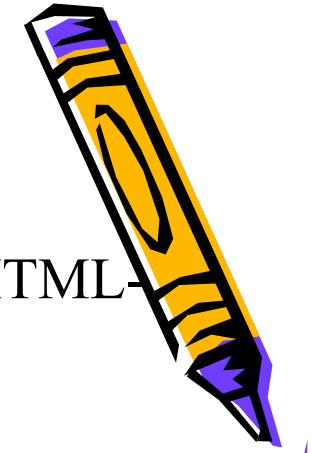
В принципе создавать HTML-форму для передачи данных методом GET не обязательно. Можно просто добавить в строку URL нужные переменные и их значения.

http://phpbook.info/test.php?id=10&user=pit

В связи с этим у передачи данных методом GET есть один существенный недостаток – любой может подделать значения параметров. Поэтому не рекомендуется использовать этот метод для доступа к защищенным паролем страницам, для передачи информации, влияющей на безопасность работы программы или сервера.



Методы

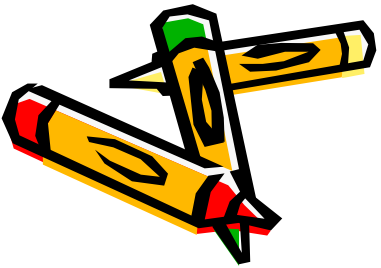


Передать данные методом POST можно только с помощью HTML-формы, поскольку данные передаются ***в теле запроса***, а не в заголовке, как в GET. Соответственно и изменить значение параметров можно, только изменив значение, введенное в форму. При использовании POST пользователь не видит передаваемые серверу данные.

Основное преимущество POST запросов – это их б***о***льшая безопасность и функциональность по сравнению с GET-запросами. Поэтому метод POST чаще используют для передачи важной информации, а также информации большого объема. Тем не менее не стоит целиком полагаться на безопасность этого механизма, поскольку данные POST запроса также можно подделать, например создав html-файл на своей машине и заполнив его нужными данными. Кроме того, не все клиенты могут применять метод POST, что ограничивает варианты его использования.

Использование HTML-форм для передачи данных на сервер

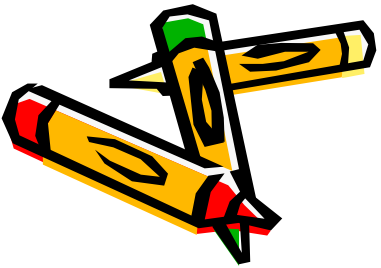
Для создания формы в языке HTML используется тег FORM. Внутри него находится одна или несколько команд INPUT. С помощью атрибутов action и method тега FORM задаются имя программы, которая будет обрабатывать данные формы, и метод запроса, соответственно. Команда INPUT определяет тип и различные характеристики запрашиваемой информации. Отправка данных формы происходит после нажатия кнопки input типа submit.



Обработка запросов с помощью PHP

Создадим простой HTML файл. Назовем этот файл index.html В нем мы указали, что данные форм будут обрабатывать-ся файлом email1.php

```
<html>
<head>
<title> Запрос информации</title> </head>
<body>
<h5>Хотите больше знать о наших товарах? </h5>

<table width = 400><tr><td align = right>
<form action="email.php" method="POST">
Ваше имя:<br>
<input type="text" name="name" size="20" maxlength="30"><br>
Ваш email:<br>
<input type="text" name ="email" size="20" maxlength ="30"><br>
Меня интересуют:
        <select name="preference">
<option value = "Компьютеры"> Компьютеры</option>
<option value = "Мобильные_телефоны"> Мобильные телефоны
   </option>
        </select> <br>
<input type ="submit" value="Отправить запрос!">
</form>
</td></tr></table>
</body>
</html>
```



Теперь, если пользователь вызовет request.html и наберет в форме имя "***", email: "*****" и отметит какой именно товар его интересует, а после этого нажмет "отправить запрос!".**

Хотите больше знать о наших товарах?

Ваше имя:

Ваш email:

Меня интересуют: Компьютеры

Отправить запрос!

В ответ вызовется email.php, который выведет на экран.

http://test1.ru/email.php

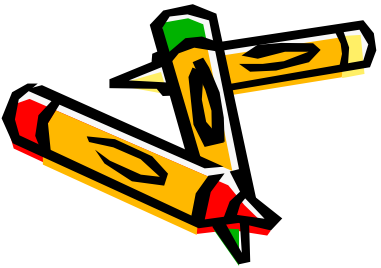
Привет, *****

Спасибо за ваш интерес.

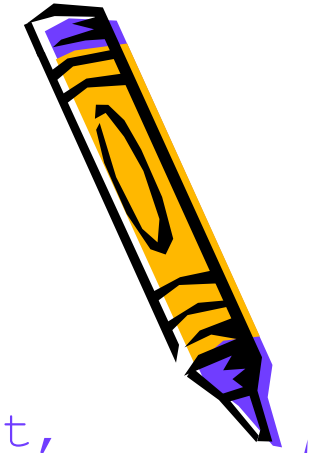
Вас интересуют Компьютеры, информацию о них мы пошлем вам на email: *****

Такая реакция на submit возможна, если содержимое файла email1.php будет таким:

```
<?
/* Этот скрипт получает переменные из request.html */
header('Content-Type: text/html; charset= utf-8');
        print "<div text-align:'center'>";
        print "Привет, ".$_POST['name'];
        print "<br><br>";
        print "Спасибо за Ваш интерес.<br><br>";
        print "Вас интересуют ".$_POST['preference']."
    информацию о них мы пошлем вам на email:
".$_POST['email']; print "</div>";
?>
```



$_POST используется при передаче данных из формы
$_GET - при передаче данных из адресной строки



Теперь мы должны выслать email.
Для этого в PHP есть функция MAIL.

Синтаксис: void mail (string to, string subject, string message, string add_headers);

to - email адреса получателя.

subject - тема письма.

message – собственно текст сообщения.

add_headers – другие параметры заголовка письма.

Допишем в файл email1.php следующий код:

```
<?
$subj = "Запрос на информацию";
$text = "Уважаемый ".$_POST['name']."! Спасибо за ваш
интерес!
Вас интересуют ".$_POST['preference']." Мы их
доставляем бесплатно.
Обратитесь в ближайший филиал нашей компании и получите
подробную информацию по интересующей вас продукции.";

mail($_POST['email'], $subj, $text);
$subj = "Поступил запрос на информацию";
$text = $_POST['name'].", Вас интересовали
".$_POST['preference']." Информацию о них мы отправим
на Ваш email-адрес: ".$_POST['email']";
mail($adminaddress, $subj, $text);
?>
```

В результате пользователь получит письмо с более подробной информацией.

Базовый синтаксис PHP

Команды на языке PHP обрамляются специальными дескрипторами — тэгами языка PHP. Все, что находится вне этих тегов, игнорируется интерпретатором. Поддерживаются следующие стили написания тэгов:

- XML-стиль (рекомендуемый);

```
<?php код на PHP ?>
```

- HTML-стиль;

```
<script language="php"> код на PHP </script>
```

- Краткий стиль;

```
<? код на PHP ?>
```

- ASP-стиль.

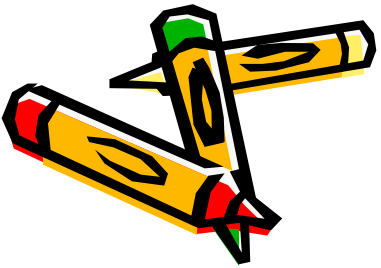
```
<% код на PHP %>
```

Базовый синтаксис

Существует ряд требований, которые необходимо соблюдать при программировании на PHP:

- Каждая команда заканчивается точкой с запятой (;);
- Одну команду можно записывать в несколько строк или несколько команд в одну строку;
- PHP чувствителен к регистру символов в именах переменных и функций;

```
<?php
    $index = 10;
    print($Index);      // Ошибка
?>
```

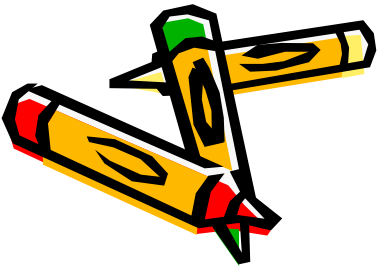


- PHP нечувствителен в отношении ключевых слов, к пробелам, переводам строки, знакам табуляции.

Комментарии

PHP поддерживает три вида комментариев: один многострочный и два однострочных.

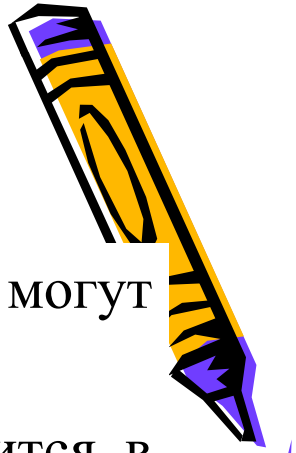
```
<?php
      /*
            Первый
            вид
            комментария
       */

      //    Второй

      #     Третий
?>
```



Переменные

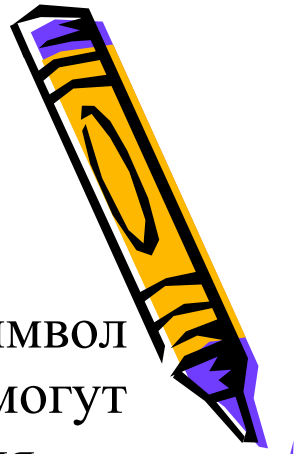
- Все имена переменных должны начинаться со знака доллара ($);
- Объявления не являются обязательными. Переменная начинает существовать с момента присвоения ей значения или с момента первого использования. Если использование начинается раньше присвоения, то переменная будет содержать значение по умолчанию;
- Тип переменной определяется динамически присвоенным значением и текущей операцией.
- В PHP существует два способа задания строк в двойных, либо одинарных кавычках.
- Разница между ними заключается в том, что если внутри строки первого типа написать имя переменной, то интерпретатор PHP подставит значение этой переменной при результирующем выводе. Во втором случае (одинарные кавычки) подстановки не происходит.

Переменные



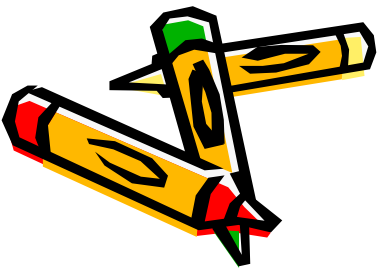
- Переменные в PHP — особые объекты, которые могут содержать в буквальном смысле все, что угодно.
- Если в программе что-то хранится, то оно всегда хранится в переменной (исключение — константа, которая, впрочем, может содержать только число, строку, а начиная с PHP 7 — массив).
- Такого понятия, как указатель (как в C), в языке не существует — при присваивании переменная в большинстве случаев копируется один в один, какую бы сложную структуру она ни имела. Единственное исключение из этого правила — копирование переменной, ссылающейся на объект или массив: в этом случае объект остается в единственном экземпляре, копируется лишь ссылка на него.
- В PHP также присутствует понятие ссылки. Всего существуют три вида ссылок: жесткие, символические и ссылки на объекты.

Переменные



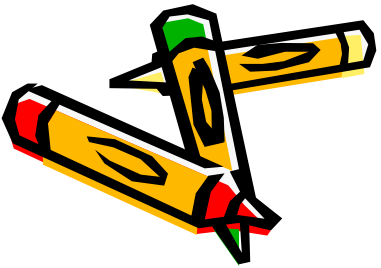
Первым символом после $ должна быть буква или символ подчеркивания. Далее в имени переменной могут присутствовать буквы, цифры и символ подчеркивания.

```
<?php
        $I;                 // Допустимо
        $1;                 // Недопустимо
        $_1;                // Допустимо
        $firstName;         // Допустимо
        $7Lucky;            // Недопустимо
        $~password;         // Недопустимо
        $Last!Visit;        // Недопустимо
        $Compute-Mean;      // Недопустимо
?>
```



Переменные. Пример

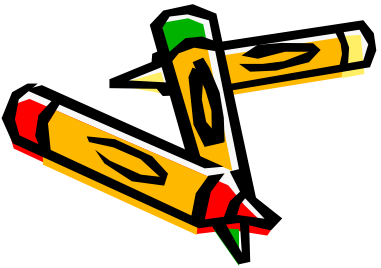
```php
<?php
      $f1 = 'Bob';          // Присваивает $f1 значение 'Bob'
      $f1 = "My name is Mike";// Изменение $f1
      $bar = 25;                // Присваивает $bar значение 25
      $bar = 2 + 2;             // Присваивает $bar 4
      $tmp = $f1;               // Присваивает $tmp значение $f1
      $tmp = &$f1;              // Ссылка на $f1 через $tmp
      $f1 = "John";             // Изменение $f1
      echo $tmp;                    // Выведет на экран "John"
      $f1 = "Mike";             // Изменяем значение $f1
      unset($f1);               // Удаляем переменную $f1
      echo $tmp;                    // Выведет на экран "Mike"
?>
```



Переменные

Print() и echo

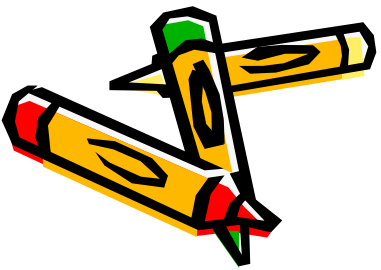
1. **print()** *ведет себя как функция*, которая всегда возвращает значение ***1***(<?php $a=print('test');/* $a=1; */ ?>).
2. **echo** - не является функцией (это языковая конструкция), ничего не возвращает;
3. **print** и **echo** различаются по передаваемым параметрам. Функции **print** можно передавать только один параметр, в то время как **echo** принимает их неограниченное количество.



Предопределенные переменные

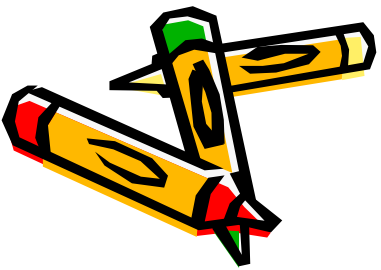
$GLOBALS — Массив, содержащий все глобальные переменные.
$_ENV — Массив переменных окружения.
$_COOKIE — Массив файлов cookie, отправленных на сервер.
$_GET — Массив переменных, отправленных методом GET.
$_POST — Массив переменных, отправленных методом POST.
$_FILES — Массив, содержащий информацию о загруженных файлах.
$_REQUEST — Массив, содержащий $_GET, $_POST, $_FILES, $_COOKIE.
$_SESSION — Массив переменных, размещенных в сессиях PHP.
$_SERVER — Массив, содержащий информацию о сервере.

Эти предопределённые переменные также являются "суперглобальными", что означает, что они доступны в любом месте скрипта.



Предопределенные переменные
(продолжение)

- **$php_errormsg** – Предыдущее сообщение об ошибке;
- **$HTTP_RAW_POST_DATA** – Необработанные данные POST;
- **$http_response_header** – Заголовки ответов HTTP;
- **$argc** – Количество аргументов переданных скрипту;
- **$argv** – Массив переданных скрипту аргументов.



Типы данных

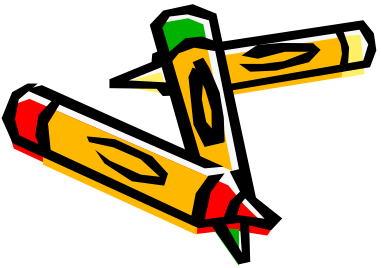
PHP поддерживает восемь типов данных.

Четыре скалярных типа:

- **`boolean`** — логический;
- **`integer`** — целое число;
- **`float`** (**`double`**) — число с плавающей точкой;
- **`string`** — строка.

Два специальных типа:

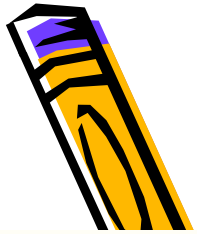
- **`resource`** — ссылка на внешний по отношению к скрипту источник данных (файл на диске, изображение в памяти и т.п.);
- **`NULL`** — отсутствие какого либо значения.



Два смешанных типа:

- **array** — массив;
- **object** — экземпляр класса.

Типы данных. Пример



```php
<?php
     $f1 = TRUE;                  // Логический
     $int = 1234;                 // Целое число
     $flt = 1.234;                // Число с плавающей точкой

     echo "Это простая строка"; // Это простая строка
          // Это добавит новую строку:
     echo "Это добавит: \n новую строку";
          // Переменная OK добавится в текст:
     $a = "OK"; echo "Переменная $a добавилась в текст";

     echo 'Это простая строка'; // Это простая строка

          Он сказал "I'll be back"
     echo 'Он сказал: "I\'ll be back"';
          // Это не добавит: \n новую строку
     echo 'Это не добавит: \n новую строку';
          //Переменная $a не подставляется
     $a = "OK"; echo 'Переменная $a не подставляется';
?>
```

Полезные функции

isset(имя_переменной) - сообщает, существует ли переменная.

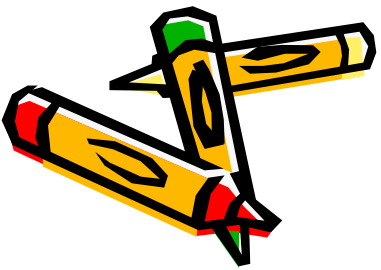
unset(имя_переменной) - уничтожает указанную переменную

empty(имя_переменной) - сообщает, присвоено ли переменной какое-либо значение.

gettype(имя_переменной) - возвращает тип указаной переменной

settype(имя_переменной, тип) - конвертирует переменную в другой тип.

is_bool(имя_переменной) - проверяет является ли тип переменной логическим.



Функции is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array() работают по аналогии с is_bool().

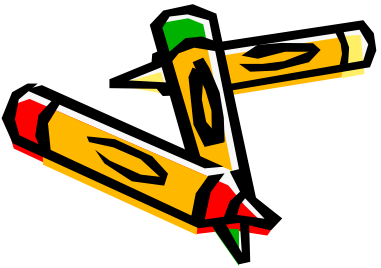
Константы

Для задания значений, которые не будут меняться в ходе выполнения сценария можно использовать константы. Так же, как и переменные, константы могут быть определены и доступны в любом месте сценария, но у них есть и ряд особенностей:

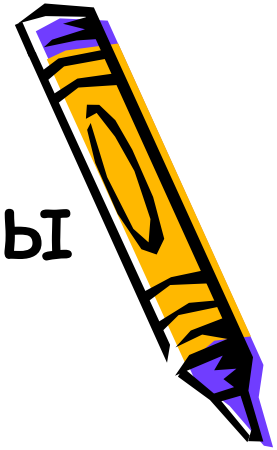
- У констант нет префикса в виде знака доллара;
- Константам нельзя присваивать значения, их можно определить вызовом функции define();
- Константы не могут быть определены или аннулированы после первоначального объявления.

Пример:

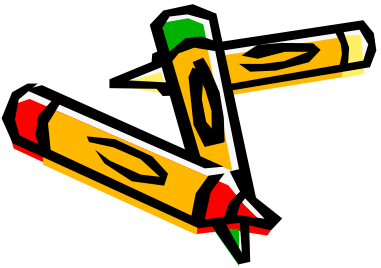
```
<?php
    define ('PI', 3.14);
    $index = 10 * PI;           // Верно
    PI = 10 * 3.14;             // Ошибка!

    define("CONSTANT", "Здравствуй, мир.");
    echo CONSTANT;          // Выведет "Здравствуй, мир."
    echo Constant; // Выведет "Constant" и
    предупреждение

?>
```



Предопределенные константы

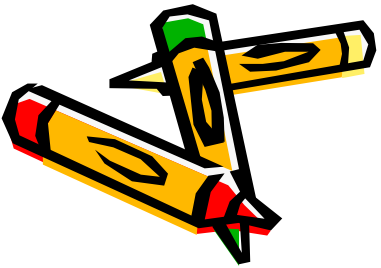


_ _LINE_ _	- Номер текущей строки.
_ _FILE_ _	- Полный путь и имя текущего файла.
_ _FUNCTION_ _	- Имя текущей функции.
_ _CLASS_ _	- Имя текущего класса.
PHP_EXTENSION_DIR	- Каталог расширений PHP
PHP_OS	- Операционная система
PHP_VERSION	- Версия PHP
PHP_CONFIG_FILE_PATH	- Каталог размещения php.ini



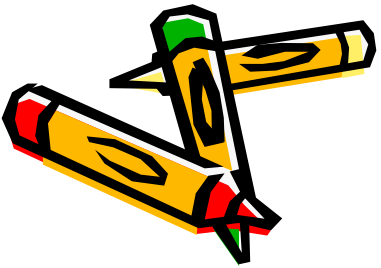
Арифметические операции

```
-$a             // Смена знака
$a + $b         // Сумма
$a - $b         // Разность
$a * $b         // Произведение
$a / $b         // Частное
$a % $b         // Остаток от деления
$a += $b        // Аналогично $a = $a + $b
$a -= $b        // Аналогично $a = $a - $b
$a *= $b        // Аналогично $a = $a * $b
$a /= $b        // Аналогично $a = $a / $b
$a %= $b        // Аналогично $a = $a % $b
```



Операции сравнения

```
$a == $b        // TRUE если $a равно $b.
$a === $b       // TRUE если $a равно $b И имеет тот же тип
$a != $b        // TRUE если $a не равно $b.
$a !== $b       // TRUE если $a не равно $b ИЛИ у них разные типы.
$a < $b         // TRUE если $a строго меньше $b.
$a > $b         // TRUE если $a строго больше $b.
$a <= $b        // TRUE если $a меньше или равно $b.
$a >= $b        // TRUE если $a больше или равно $b.
```

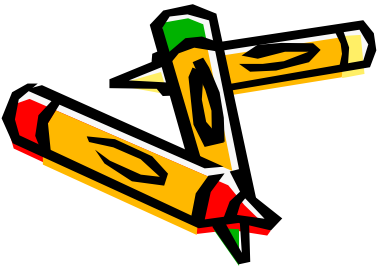


Логические операции

```
$a and $b      // TRUE если и $a, и $b TRUE.

$a or $b       // TRUE если или $a, или $b TRUE.

$a xor $b      // TRUE если $a, или $b TRUE, но не оба.

!$a            // TRUE если $a не TRUE.

$a && $b       // TRUE если и $a, и $b TRUE.

$a || $b       // TRUE если или $a, или $b TRUE.
```

Побитовые операции

```
$a & $b        // Побитовое И

$a | $b        // Побитовое ИЛИ

$a ^ $b        // Исключающее или

~ $a           // Отрицание

$a << $b       // Побитовый сдвиг влево

$a >> $b       // Побитовый сдвиг вправо
```



Специфичные операции

Конкатенация

```
$a = "Hello ";
$b = $a."World!"; // $b содержит строку "Hello World!"
$a .= "World!";       // $a содержит строку "Hello World!"
```

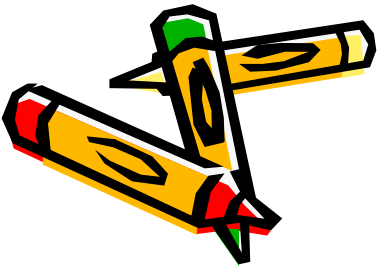
Подавление ошибки

```
@$a = 1 / 0;                    // Ошибка не будет сгенерирована
```

Тернарная операция

По сути является аналогом условной конструкции **if...else**

```
условие ? значение, если условие истинно : значение, если ложно
```

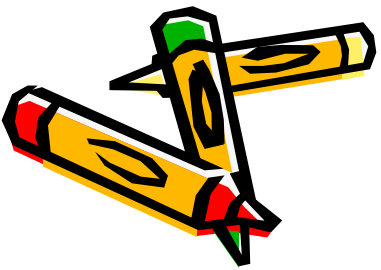


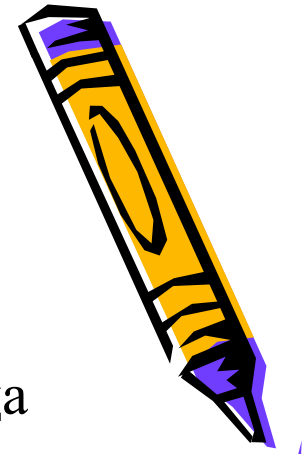
```
<?php
      $grade = 3;
      $result = ($grade > 2 ? 'Сдал' : 'Не сдал');
      echo $result;
?>
```

Инкремент / Декремент

```
++$a      // Увеличивает $a на единицу и возвращает
          значение $a
$a++      // Возвращает значение $a, а затем увеличивает
          $a на единицу
--$a      // Уменьшает $a на единицу и возвращает
          значение $a
$a--      // Возвращает значение $a, а затем уменьшает
          $a на единицу
```

```
<?php
        $a = 5;
        echo "Должно быть 5: " . $a++ . "<br>";
        echo "Должно быть 6: " . $a . "<br>";
        $a = 5;
        echo "Должно быть 6: " . ++$a . "<br>";
        echo "Должно быть 6: " . $a . "<br>";
?>
```





Управляющие конструкции

Конструкция if

Указанные действия выполняются тогда и только тогда, когда условие истинно.

```
if(условие){
      Действие;
}
```

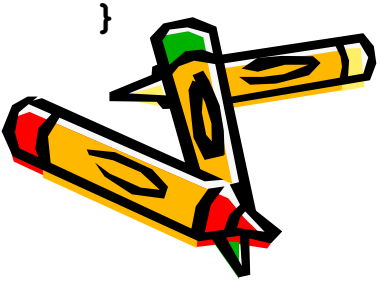
```
if($index > 0)   {
      echo 'Index > 0';
      }
```

Конструкция if...else

Если условие истинно, выполнятся действия из блока if, в противном случае — из блока else.

```
if(условие){
      Действие;
}else{
      Действие;
}
```

```
if($index > 0){
      echo 'Да';
}else{
      echo 'Нет';
      }
```



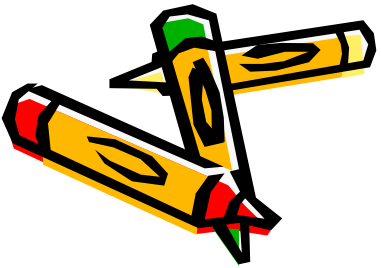
Управляющие конструкции

Конструкция elseif

Если условие блока if истинно, выполнятся действия блока if. В противном случае, если условие блока elseif истинно, выполнятся действия блока elseif. Во всех остальных случаях выполнятся действия из блока else.

```
if(условие){
      Действие;
}elseif(условие){
      Действие;
}else{
      Действие;
}
```

```
if($numb < 5){
                  $discount = 0;
elseif($numb >= 5 && $numb <=10)
                  $discount = 5;
else{
                  $discount = 10;
                  }
```



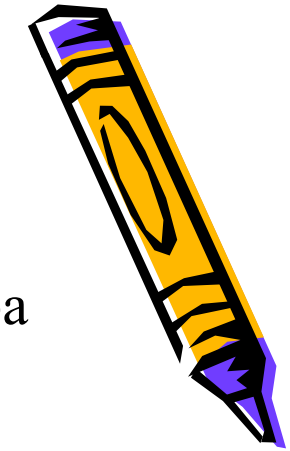
Управляющие конструкции

Конструкция switch

Если значение переменной соответствует значению одного из блоков case, выполнятся действия из этого блока. В противном случае - из блока default.

```
switch(Переменная){
      case Значение 1:
            Действие 1;
            [break;]
      case Значение 2:
            Действие 2;
            [break;]
      [default: Действие;]
}
```

```
switch($day){
      case 1:
            echo 'Понедельник';
break;
      case 2:
            echo 'Вторник'; break;
      case 3:
            echo 'Среда'; break;
      case 4:
            echo 'Четверг'; break;
      case 5:
            echo 'Пятница'; break;
      case 6:
            echo 'Суббота'; break;
      case 7:
            echo 'Воскресенье';
break;
      default:
            echo 'Нет такого дня';
}
```



Циклы

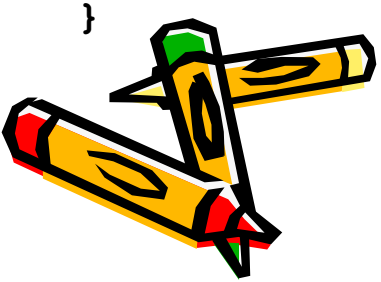
Циклы предназначены для многократного исполнения набора инструкций.

Цикл for

В цикле for указывается начальное и конечное значения счетчика, а так же шаг, с которым счетчик будет изменяться. Изменяться счетчик может как в положительную, так и отрицательную сторону. Действия выполнятся столько раз, сколько итераций пройдет от начального значения счетчика до достижения конечного, с указанным шагом.

```
for(начало;конец;шаг){
      Действие;
      ...
}
```

```
for ($i = 1; $i <= 5; $i++) {
        $sum += $i;
            echo $sum;
        }
```



Циклы

Цикл while

Действия будут выполняться до тех пор, пока условие истинно. Цикл while является циклом с предусловием.

```
while (условие) {
      Действие;
      ...
}
```

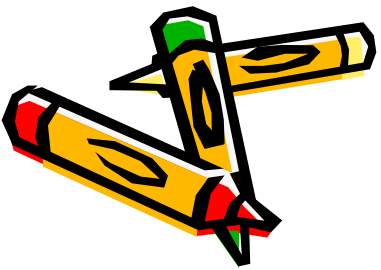
```
while ($state == 'Солнце высоко') {
      echo 'Рабочий день продолжается';
      $state = 'Солнце заходит';
}
```

Цикл do...while

Цикл do...while является циклом с постусловием. Это значит, что сначала будет выполняться действие, а потом проверяться условие. Таким образом действие всегда выполнится минимум один раз.

```
do{
      Действие;
      ...

} while (условие);
```

```
do{
   echo 'Пиф-паф';
   }
while ($state == 'Живой');
```

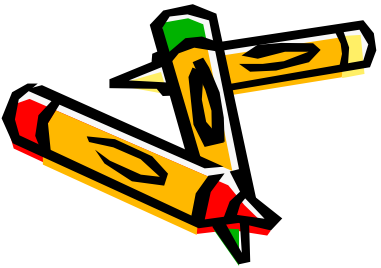


Управление циклами

Break прерывает работу цикла. Интерпретатор перейдет к выполнению инструкций, следующих за циклом.

```
$index = 1;
while ($index < 10){
echo "$index <br>";
$index++;
if ($index == 5)
        break;

}

$index = 0;

while ($index < 10){
$index++;
if ($index == 5)
        continue;
        echo "$index <br>";

}
```

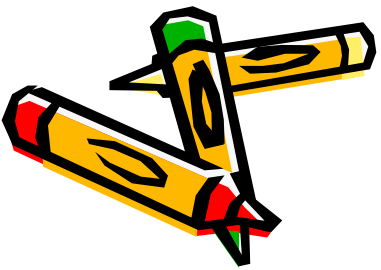
Continue прерывает выполнение текущей итерации цикла. Цикл продолжит выполняться со следующей итерации.

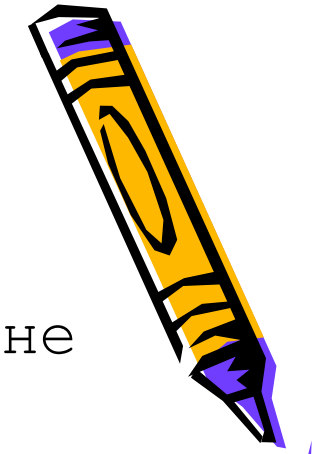


Массивы

Массив — это структура, в которой хранится упорядоченный набор данных. В PHP массив можно создать следующими способами:

```
<?php
      $zoo[0] = 'слон';
      $zoo[6] = 'крокодил';
      $zoo[4] = 'жираф';
      $zoo[] = 'осел';          // Индекс равен 7

      // или

      $zoo = array ('лев', 'медведь', 'обезьяна');
      echo count ($zoo); // Количество элементов массива
?>
```

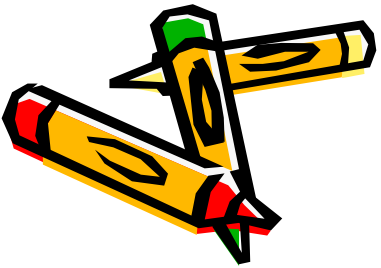




Ассоциативные массивы

В ассоциативных массивах используется не числовой, а строковый индекс. Индексы ассоциативных массивов называются ключами.

```php
<?php
  $pets['dog'] = 'Бульдог';
  $pets['cat'] = 'Шиншилла';
  $pets['fish'] = 'Золотая';

  // или

  $pets = array ('lizard' => 'Игуана',
                 'spider' => 'Черная вдова',
                 'parrot' => 'Ара');
  print_r ($pets);          // Печать массива
?>
```



Многомерные массивы

Для создания массивов в PHP существует специальная инструкция **array()**. Ее удобно использовать для создания многомерных массивов.

```
<?php
     $users = array (
          0 => array ( 'login' => 'admin',
          'password' => 'hskdfuegefdjfdg'),
          1 => array ('login' =>'tel',
          'password' => 'ppqmcnvkfghye')
     );
     echo $users[0]['login'];// admin
?>
```

Можно так:

```
$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");
```

Многомерные массивы похожи на записи в языке Pascal или структуры в языке C.

Цикл foreach

Очень удобен при работе с массивами. Указанные действия выполняются для **каждого** элемента массива $array, при этом $key — номер элемента массива $array, $value — значение этого элемента.

```
foreach ($array as [ $key => ] $value){
      Действия;
      ...
}
```

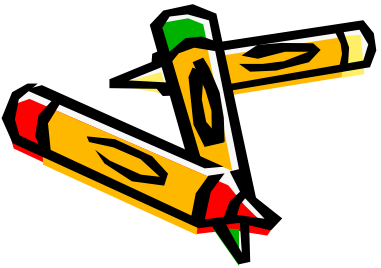
```
<?php
  $arr["first"] = "PHP";
  $arr["second"] = "MySQL";
  $arr["third"] = "Apache";

foreach($arr as $key => $value)   //эквив ($arr as $value)
  {
     echo "$key = $value <br />";
  }
?>
```

```
first = PHP
second = MySQL
third = Apache
```

Тип object (объекты)

```
// Новый класс Coor:
class Person {
var $name; // данные (свойства)

// методы:
function Getname() {
  echo $this->name; }

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

```
$object = new Person;
// Создаем объект класса Coor:
$object->name = "Alex";
// Получаем доступ к членам класса
echo $object->name;   // Выводит 'Alex'
// А для доступа, как и прежде, Getname():
$object->Getname();
```



Задания для лабораторной

1. Создать форму (или использовать форму с предыдущих занятий),

2. Выполнить проверку информации, введенной в поля формы (e-mail на наличие символа @, имя и фамилия должны содержать только буквы, почтовый индекс должен состоять из 6 символов).

Варианты: (№ вар. mod 6)+1

Задания для лабораторной

2. В html документе создайте форму с текстовым полем, в которое пользователь вводит свой логин и кнопкой типа Submit. Далее после нажатия копки Submit вызывается php скрипт который проверяет, зарегистрирован ли этот пользователь. При этом таких пользователей (разных логинов должно быть 4, сохраненных в массиве). Если введен один из существующих логинов, должно выводится приветствие для этого человека. Например: введен логин Ivan_php , должно вывестись приветствие «Здравствуйте Иванов Иван Иванович». Если введен неизвестный логин должно появиться сообщение – «Вы не зарегистрированный пользователь!»

- Для зарегистрированного пользователя должен проверятся еще и пароль.

Варианты: (№ вар. mod 6)+1

Задания для лабораторной

3.Создать HTML-форму, которая содержит 3 поля для ввода фамилии, имени и номера телефона в формате +8(XXX-XXX-XX-XX). Создать PHP-сценарий, который проверяет на корректность входные данные и выдает соответствующее сообщение, а также имя владельца телефона, содержащееся в массиве-записной книжке.

4. В html документе создайте форму с текстовым полем, в которое пользователь вводит название зверушки, и кнопкой типа Submit. Далее после нажатия копки Submit вызывается php скрипт который проверяет, содержится ли эта зверушка в заранее определённом массиве-зоопарке и выводит результат.

Если зверушка есть, извлечь из ассоциативного массива ее кличку.

Варианты: (№ вар. mod 6)+1

Задания для лабораторной

5. На странице с формой пользователю предлагается отгадать число (ввести с текстовое поле). После нажатия на кнопку ГОТОВО запускается PHP скрипт, который проверяет, отгадал ли пользователь число и если нет пишет текст: 7 – не верно. И далее ссылка на документ с формой и тестом – попробуй еще раз! Если пользователь угадал число, то большими красными буквами должно писаться - ВЕРНО.

- В процессе угадывания предусмотрите подсказки типа – загадано больше или загадано меньше.

6. Создайте массив со следующими элементами: Name, Address, Phone, Mail и заполните его. С помощью цикла foreach осуществите форматированный вывод массива в виде: «элемент: значение», предварительно проверив правильность и наличие значений.

Варианты: (№ вар. mod 6)+1

Используемые литературные источники

1. Самые популярные PHP-фреймворки в 2017. https://geekbrains.ru/posts/php_frameworks_17
2. Сравнение популярных PHP-фреймворков. http://www.cmsmagazine.ru/library/items/programming/php-frameworks-compare/
3. Бретт Маклафлин **Исчерпывающее руководство PHP и MySQL**. — СПб.: Питер, 2013. — 512 с.: ил.
4. **PHP и MySQL:** http://www.php.su/learnphp/vars/?types