Часть 2. Практическая работа №1

Основы протокола HTTP. Установка веб­сервера. Основы PHP.

В этом Практическая работа вы:

* познакомитесь с протоколом HTTP
* узнаете, что происходит между тем, как вы набрали адрес в браузере и увидели запрашиваемую страницу на экране монитора
* познакомитесь с аббревиатурами URL и URI
* исследуете клиент-серверную архитектуру взаимодействия при HTTP-соединении
* научитесь устанавливать веб-сервер на локальный компьютер
* приступите к изучению PHP
* ознакомитесь с базовым синтаксисом
* узнаете про переменные языка
* про константы
* про типы данных в PHP
* научитесь преобразовывать переменные одного типа в переменные другого
* научитесь писать выражения с помощью операторов языка

1. Основы протокола HTTP

Прежде чем перейти к изучению такого популярного и мощного языка веб­программирования как PHP, важно иметь четкое представление о том, как вообще функционирует интернет. Что из себя представляют PHP-скрипты и HTML-код? Как взаимодействуют наши компьютеры с серверами, на которых располагаются веб-страницы? И, наконец, как эти замысловатые скрипты, превращаются в те прекрасные интернет-сайты, которые мы каждый день видим на экранах своих мониторов.

К сожалению, огромное количество начинающих PHP-программистов даже не отдает себе отчет в том, где выполняются написанные ими скрипты, и вообще, что из себя представляет процесс выполнения PHP-скрипта. Однако без этих знаний очень трудно достичь каких-либо существенных успехов в PHP-разработке. Более того, эти знания должны быть получены в первую очередь.

Представьте себе архитектора, который абсолютно не разбирается в строительных материалах. Можно ли назвать его настоящим профессионалом? Скорее всего, такой человек не сможет построить что-нибудь сложнее сарая. Так и веб-программист не понимающий основных принципов интернет-взаимодействия просто не способен решать более менее серьезные задачи.

* 1. Что происходит между тем, как вы набрали адрес сайта и увидели страницу на экране

Все мы знакомы с браузерами. Это те самые программы, которые позволяют нам просматривать страницы в интернете. Но если вы спросите десять программистов, как работает браузер, то восемь из них не смогут дать внятного ответа. Давайте попробуем самостоятельно разобраться, что же происходит в промежутке между запросом какого- либо сайта и отображением страницы на экране.

Давайте рассмотрим адрес какой-нибудь реальной веб-страницы, например [http://prog- school.ru/catalog](http://prog-school.ru/catalog). Она однозначно идентифицирует веб-страницу, которую должен отобразить для вас браузер. Эту строку также называют URL или URI. Давайте подробнее разберем обе эти аббревиатуры.

* 1. URL и URI

URL (англ. Uniform Resource Locator) - единообразный локатор (определитель местонахождения) ресурса. URL — это стандартизированный способ записи адреса ресурса в сети Интернет.

URI (англ. Uniform Resource Identifier) — унифицированный (единообразный) идентификатор ресурса. URI — это последовательность символов, идентифицирующая абстрактный или физический ресурс.

URI - это более общее понятие, нежели URL. URI не всегда указывает то, как получить ресурс, в отличие от URL, а только идентифицирует его. URL — это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса. Действительно, любой URL-адрес несет достаточно информации для точного нахождения страницы. Далее в этой книге при использовании адресов сайтов будем придерживать аббревиатуры URL.

* 1. Структура адреса сайта

Вернемся к URL-адресу <http://prog-school.ru/catalog>. Его можно разделить на 3 части:

* http://
* prog-school.ru
* /catalog

Первая часть адреса «http://» определяет протокол взаимодействия браузера с сервером. В нашем случае это протокол HTTP, о нем речь пойдет ниже.

Вторая часть адресной строки называется доменом и служит для идентификации конкретного сайта с помощью службы DNS. DNS (англ. Domain Name System — система доменных имён) — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства). В сети существует большое количество DNS-серверов, которые по доменному имени ресурса могут «подсказать» его реальное местоположение, определяемое IP-адресом.

Третья часть адресной строки определяет подзапрос к сайту. Это может быть путь до определенного файла или каталога на сервере. Здесь же могут быть заданы различные параметры HTTP-запроса.

* 1. Исходный код HTML-страницы

Теперь давайте разберем, что же получает браузер в ответ на сформированный HTTP запрос. Страница может состоять из текста, картинок, гиперссылок, полей ввода, кнопок и других элементов. Информация обо всем этом была передана от веб-сервера браузеру, который и сгенерировал конечный внешний вид страницы. Передаваемые данные описываются с помощью протокола HTML.

HTML (от англ. HyperText Markup Language — «язык разметки гипертекста») — стандартный язык разметки документов в интернете. Язык HTML интерпретируется браузером и отображается в виде документа, в удобной для человека форме.

Можно сказать, что браузеры выполняют две основные функции - это взаимодействие с веб-серверами посредством HTTP-запросов, а также преобразование полученного от сервера HTML-кода в визуальное представление.

* 1. Схема клиент-серверного взаимодействия при работе браузера

Итак, давайте теперь ответим на главный вопрос: что же происходит в промежутке между тем, как мы набрали адрес сайта в браузере и моментом, когда его содержимое отобразилось на экране.

После того, как URL сайта введен в адресную строку, и нажата кнопка Enter, браузер формирует пакет данных, который посылает по сети. Этот пакет содержит URL запрашиваемого сайта, а также прочие данные запроса, оформленные согласно протоколу HTTP. Содержание пакета нас сейчас не интересует, важно то, что переданный URL позволяет промежуточным узлам в интернете доставить наш пакет с HTTP-запросом по адресу до нужного сервера.

На физическом веб-сервере (то есть неком реальном компьютере) должна быть запущена соответствующая программа, которая также называется веб-сервером и служит для обработки входящих HTTP-запросов. Самым популярным веб-сервером на данный момент является программа Apache, логотип которой приведен на рисунке 1.1.

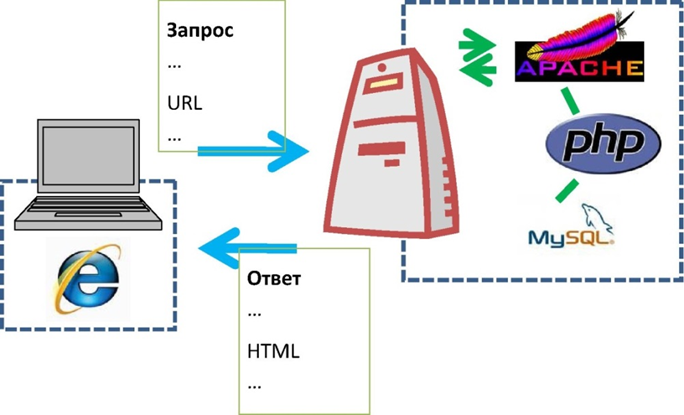


Рисунок 1.1

После получения пакета с HTTP-запросом веб-сервер определяет, какие действия необходимы для его обработки. Если HTTP-запрос осуществляется к обычной HTML- странице, то веб-сервер просто передает ее содержимое браузеру. Если же HTTP-запрос осуществляется к какому-либо скрипту (например, PHP-скрипту), веб-сервер передает запрос на обработку соответствующей программе, отвечающей за обработку этого типа скриптов. Обработчик скрипта в свою очередь может вызывать другие программы в ходе своей работы, например, СУБД MySQL.

Результатом работы обработчика скриптов является HTML-код, который веб-сервер посылает обратно на компьютер пользователя. Сгенерированный HTML-код веб-сервер упаковывает в HTTP-пакет, который и передается по сети обратно клиенту. Полученный HTTP-ответ попадает в браузер клиента, который извлекает из него HTML-код и генерирует на его основе графическое представление запрошенной страницы.

* 1. Протокол HTTP

Давайте дадим более четкое определение протоколу HTTP и разберемся, зачем он нужен.

HTTP (сокр. от англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов). Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом. HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов.

Иными словами, HTTP - это язык, на котором общаются браузер и веб-сервер при обмене пакетами с данными. Обмен сообщениями в ходе HTTP-соединения идёт по обыкновенной схеме «запрос-ответ»

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

1. Стартовая строка — определяет тип сообщения;
2. Заголовки — характеризуют тело сообщения, параметры передачи и прочие сведения;
3. Тело сообщения — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа.

Стартовые строки HTTP-запросов и HTTP-ответов различны. Для HTTP-запроса стартовая строка имеет следующий вид:

Метод URI HTTP/Версия

Здесь:

* Метод — название запроса, одно слово заглавными буквами.
* URI определяет путь к запрашиваемому документу.
* Версия — пара разделённых точкой арабских цифр. Например: 1.0. Определяет версию используемого HTTP-протокола. На данный момент наиболее актуальной является версия 1.1.

Для HTTP-ответа стартовая строка выглядит следующим образом:

HTTP/Версия КодСостояния Пояснение

Здесь:

* Версия — пара разделённых точкой арабских цифр как в запросе.
* КодСостояния — три цифры. По коду состояния определяется дальнейшее содержимое сообщения и поведение клиента.
* Пояснение — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.
  1. Заголовки HTTP

Заголовки HTTP (англ. HTTP Headers) — это строки в HTTP-сообщении, содержащие разделённую двоеточием пару параметр-значение. Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Примеры заголовков:

Server: Apache/2.2.11 (Win32) PHP/5.3.0 Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT Content-Type: text/plain; charset=windows-1251 Content-Language: ru

В примере выше каждая строка представляет собой один заголовок. При этом то, что находится до первого двоеточия, называется именем (англ. name), а что после неё — значением (англ. value).

Все заголовки разделяются на четыре основных группы:

1. General Headers (русск. Основные заголовки) — должны включаться в любое сообщение клиента и сервера.
2. Request Headers (русск. Заголовки запроса) — используются только в запросах клиента.
3. Response Headers (русск. Заголовки ответа) — только для ответов от сервера.
4. Entity Headers (русск. Заголовки сущности) — сопровождают каждую сущность сообщения.

Заголовки запроса и ответа, как и основные заголовки, описывают всё сообщение в целом и размещаются только в начальном блоке заголовков, в то время как заголовки сущности характеризуют содержимое каждой части в отдельности, располагаясь непосредственно перед её телом.

Мы не будем углубляться в изучение форматов HTTP-сообщений. На данном этапе достаточно иметь общее представление о функционировании механизмов, обеспечивающих клиент-серверное взаимодействие между браузерами и веб-серверами. Дополнительную информацию о HTTP-заголовках вы сможете найти в Википедии: [http://ru.wikipedia.org/wiki/Заголовки\_HTTP](http://ru.wikipedia.org/wiki/%d0%97%d0%b0%d0%b3%d0%be%d0%bb%d0%be%d0%b2%d0%ba%d0%b8_HTTP)

1. Установка веб-сервера на локальный компьютер

Как нам стало известно, за словом «веб-сервер» скрывается сразу несколько понятий. С одной стороны, это физический компьютер в сети, к которому клиенты направляют свои

HTTP-запросы. С другой стороны, это программа на этом компьютере, которая занимается обработкой HTTP-запросов. И, наконец, в более общем смысле, это целая совокупность программ (таких как Apache, PHP-интерпретатор, MySQL), которая участвует в формировании HTTP-ответа для клиента.

Для того, чтобы превратить наш домашний локальный компьютер в полноценный веб­сервер нам требуется установка трех программ:

1. веб-сервера Apache
2. PHP-интерпретатора
3. Системы управления базами данных MySQL

Помимо установки этого комплекта программ требуется их настройка для корректного взаимодействия между собой. Для новичков, только начинающих разбираться в веб­программировании, такая задача может оказаться слишком сложной. Однако есть простое решение, которое позволит автоматизировать процесс установки всех необходимых компонент веб-сервера, и называется оно Denwer.

Denwer - это самораспаковывающийся набор дистрибутивов (Apache, PHP, MySQL, Perl и т.д.) и оболочка для разработки сайтов на «домашней» (локальной) Windows- машине без выхода в Интернет.

Установка этого пакета не требует специальных знаний и не занимает много времени. Выполните следующие простые шаги для инсталляции Denwer^ на своем компьютере:

1. Перейдите на сайт <http://www.denwer.ru/>
2. Скачайте наиболее свежую версию дистрибутива.
3. Запустите инсталлятор и следуйте его инструкциям.

Если у вас возникли сложности, посмотрите это видео на сайте Школы Программирования [http://prog-school.ru/2010/02/kak-ustanovit-veb-server-i-napisat-pervvi- php-skript/](http://prog-school.ru/2010/02/kak-ustanovit-veb-server-i-napisat-pervvi-php-skript/).

1. Основы PHP
   1. О PHP

PHP представляет собой мощный язык написания сценариев для Web и поддерживает невероятно большой диапазон Internet-технологий. Это делает его на сегодняшний день ведущим языком сценариев для Web.

PHP скрипт - это набор инструкций для программы PHP-интерпретатора на сервере. Результатом работы PHP-скрипта обычно является HTML-код, который возвращается клиенту в браузер. Тем не менее, PHP-интерпретатор может выполнять не только генерацию HTML-кода, но и другие полезные действия, например, работать с базой данных или файловой системой сервера.

* 1. Базовый синтаксис PHP

Первый PHP-сценарий мы уже написали на первом Практическая работае. Приведем его ещё раз здесь, но на этот раз несколько упростим:

<?php

echo "Hello!";

?>

Теперь давайте подробнее рассмотрим, из чего же он состоит.

Все PHP-сценарии пишутся в виде блоков кода. Эти блоки при необходимости могут быть встроены в HTML, и обычно определяются с помощью строки <?php в начале и ?> — в конце. Все, что вне этих идентификаторов блока, интерпретатор РНР игнорирует и передает обратно Web-серверу для отображения на стороне клиента.

Для иллюстрации видоизмените свой файл index.php следующим образом:

<html>

<head><title>nepBbrn РНР-сценарий</title></head>

<body>

Пример работы php:<br/>

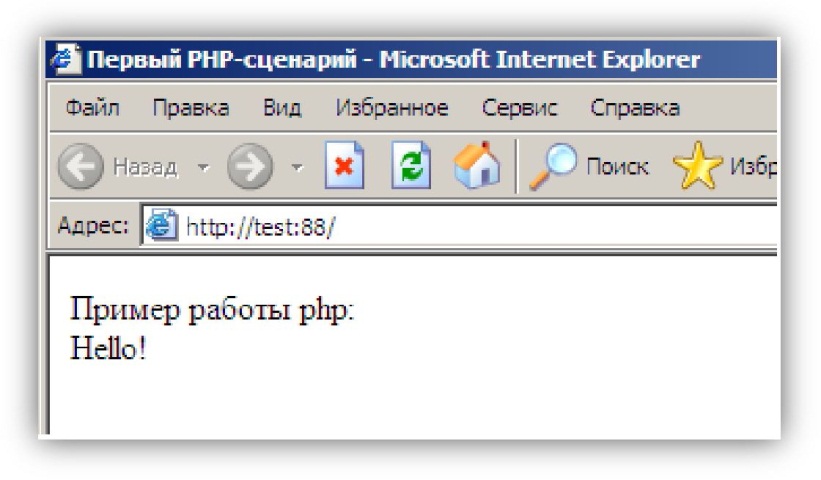
<?php

echo "Hello!";

?>

</body>

</html>

После этого зайдите на свой локальный сайт и посмотрите результат. Должно получиться что-то вроде этого:

Здесь вы знакомитесь с первым оператором РНР — echo. Этот оператор представляет базовый метод РНР для отправки содержимого клиенту, и он будет интенсивно использоваться на протяжении всего курса.

Как и во всех Си-подобных языках, в PHP каждое предложение завершается точкой с запятой.

* 1. Переменные

Несмотря на то, что это PHP-код, предыдущий сценарий не выполняет ничего такого, что не могло бы быть сделано средствами стандартного HTML. Чтобы сделать что-то более полезное, вам придется изучить переменные РНР.

В РНР имена переменных всегда начинаются с символа $ и содержат произвольную комбинацию символов, при условии, что первый символ после $ будет буквой или знаком подчеркивания. В число допустимых символов входят заглавные и прописные латинские буквы, цифры, символ подчеркивания («\_»), а также символы с ASCII-кодами в диапазоне от 127 до 255 (переменную можно назвать по-русски, например, «строка», но мы это настоятельно не рекомендуем). Переменные в РНР могут быть определены либо присвоением им значения, либо с помощью оператора var.

Измените PHP код своей страницы следующий образом:

|  |  |  |
| --- | --- | --- |
| <?php |  |  |
| $string var = | "string variable"; // | строковая переменная |
| $int var = 5; | /\* числовая переменная | \*/ |
| echo "Hello! | " . $string var . " " . | $int var; |
| ?> |  |  |

Перезапустите страницу, и вы увидите, что в браузере отобразилась строчка «Hello! string variable 5». Здесь в третьей строке мы вывели значения двух переменных, которые объявили в первых двух строчках.

В РНР все, что находится между /\* и \*/, трактуется как комментарий, используемый для пояснений в теле сценария, и игнорируется интерпретатором. Для однострочных комментариев применяются символы //, которые комментируют весь текст до конца строки.

Точка - это оператор конкатенации, то есть склеивания строки. О нем подробнее мы поговорим чуть позже.

* 1. Константы

Константы представляют собой контейнеры для данных, как и переменные, но после присваивания константе значения его уже нельзя изменить.

Константы создаются в РНР с помощью функции define():

define("INDEPENDENCEDAY", "4th July");

В теле кода на константу можно сослаться просто по имени:

echo(INDEPENDENCEDAY);

Принято записывать имена констант буквами верхнего регистра, хотя возможно выбирать любые имена, отвечающие правилам именования переменных. Константы РНР фактически действуют так же, как директива #define в препроцессоре С: можно определить их в некотором месте, а затем выполнять разный код в зависимости от того, определена ли константа и какое значение она имеет. Проверка выполняется с помощью функции defined():

if (defined("INDEPENDENCEDAY"))

{

echo ("INDEPENDENCEDAY is defined”);

}

else

{

echo("INDEPENDENCEDAY is not defined");

}

Константы бывают нужны достаточно редко. Использовать их имеет смысл для того, чтобы не «зашивать» в код абсолютные пути, IP адреса, какие-то числовые значения и т.д. Хорошим примером использование константы было бы хранение в ней пути к файлу, в который пишется лог событий. Или же имя базы данных и пароль для доступа к ней.

* 1. Типы данных

С точки зрения типизации переменных РНР можно классифицировать, как свободно- типизированный язык. Это значит, что изначально нет необходимости задавать определенный тип переменной. Вместо этого, когда ей присваивается значение, РНР трактует ее соответствующим образом в зависимости от значения и контекста, в котором она применяется. В РНР существует три простых типа переменных (целые, строковые, действительные с плавающей точкой) и два сложных типа (объекты и массивы). В этом Практическая работае мы будем работать только с простыми типами.

Первый тип данных, который будет представлен, это целые числа. Целые числа — фундаментальный числовой тип РНР, представляющий значения со знаком величиной до чуть более 2 миллиардов. На практике РНР воспринимает целые значения с использованием трех математических представлений: десятичные, восьмеричные и шестнадцатеричные. В большинстве ситуаций PHP-сценарии пишутся в десятичной нотации. Однако в некоторых случаях восьмеричные и шестнадцатеричные числа могут существенно облегчить жизнь.

Ниже показано, как использовать числа в разных нотациях.

<?php

// Стандартная десятичная нотация $my int 10 = 50;

//То же число в восьмеричной нотации (начинается с нуля)

$my int 8 = 062;

// Шестнадцатеричная нотация (начинается с 0x)

$my int 16 = 0x32;

?> \_ \_

Когда РНР работает с дробными числами, он представляет значения в виде типа данных с плавающей точкой. Числа с плавающей точкой — это любые числа, которые содержат десятичную дробную часть и могут быть выражены в десятичном или экспоненциальном представлении.

<?php

// Стандартная нотация с десятичной точкой $my float = 5.1;

// То же число в экспоненциальном представлении с плавающей точкой $my float = .051е2;

?> ~~

И последний базовый тип данных, о котором говорится в этой главе — строковый. Для начала отметим, что существует два типа строк. Первый тип строк определяются с использованием двойных кавычек, в то время как второй тип строк представляются в одинарных кавычках.

Разница между этими строками в том, что если внутри строки первого типа написать имя переменной, то интерпретатор PHP подставит значение этой переменной при результирующем выводе. В случае же со вторым типом строк (одинарные кавычки) подстановки не происходит. Для лучшего понимания напишите следующий пример и посмотрите результат.

<?php

$my int = 50;

$string one = "Значение переменной равно $my int<br/>";

$string two = 'Значение переменной равно $my int<br/>'; echo $string one; echo $string two;

?>

Если этот сценарий исполнить, его вывод будет таким:

|  |  |  |  |
| --- | --- | --- | --- |
| Значение | переменной | равно | 50 |
| Значение | переменной | равно | $my int |

Также вы можете обратить внимание, что обе строки содержат тег HTML переноса строки (<br>). Поскольку вы отправляете выходной текст Web-браузеру, это необходимо для отображения упомянутых двух значений на разных строках экрана.

* 1. Преобразование типов

Преобразование типов используется для того, чтобы указать интерпретатору на необходимость работать со значением переменной одного типа так, как если бы она имела другой тип. Оператор преобразования типа - это имя типа, заключенное в круглые скобки:

* (string) - строка
* (integer) - целое число
* (double) - число с плавающей точкой
* (boolean) - булевская переменная (true/false)

У некоторых есть также сокращенные версии:

* (int) - сокращение (integer)
* (bool) - сокращение (boolean)

Пример использования:

<?php

$a = "432.123"; echo((int)$a); ?>

Этот код выведет «432», потому что переменная $а будет восприниматься как целое число (с типом int), и дробная часть будет отброшена.

* 1. Операторы и операции

Арифметические операции

Основные операторы, предназначенные в РНР для выполнения математических действий, всем хорошо знакомы. Вот они

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| <?php $answer = | 5 + 4; |  |  | /\* | сложение \*/ |
| $answer = | $answer | - | 5; | /\* | вычитание \*/ |
| $answer = | $answer | / | 2; | /\* | деление \*/ |
| $answer = | $answer | \* | $answer; | /\* | умножение \*/ |
| $answer = ?> | $answer | О  о | 3; | /\* | остаток от деления \*/ |

Если оба аргумента - целые числа, то результат также будет целым числом. В то же время, если хотя бы один из операндов - число с плавающей точкой, то и результат будет величиной с плавающей точкой, даже если получается целое число. Пример: 0.5 + 1.5 = 2.0 , а не 2.

Для этих пяти операторов также существуют сокращенные версии для случая, когда одним из операндов выступает переменная, которой присваивается значение. Например, чтобы не писать $а = $а + $b, можно сокращенно записать $а += $b. А для того чтобы удвоить значение переменной $a, достаточно написать $а \*= 2 (тоже самое, что $a = $а \* 2).

Для увеличения или уменьшения значения переменной на единицу есть еще два сокращенных оператора: инкремента (++) и декремента (--). В применении этих унарных (требуется один операнд вместо двух) операторов есть одна тонкость, связанная с их расположением перед операндом или после него. Напишите и выполните следующий скрипт:



В первом случае переменная $a увеличивается на единицу уже после выполнения команды вывода echo. Напротив, во втором случае сначала $а становится равной 2, а уже потом происходит вывод с помощью echo.

Логические операции

Ещё одну группу операторов для работы с числами представляют операторы сравнения: «меньше» (<), «меньше или равно» (<=), «больше» (>) и «больше или равно» (>=). Все они сравнивают два заданных значения и возвращают логическое значение true или false. Например:

<?php

$a = 3 < 4;

echo($ a);

?>

В случае, если значение $a верно, скрипт выведет единицу, иначе - не выведет ничего. Это связано с особенностью представления логических переменных в PHP.

Операции над строками

Наиболее важной операцией над строками - является операция их объединения (конкатенации). В наших примерах она уже не раз встречалась, и мы будем пользоваться ей постоянно. Для конкатенации строк PHP использует символ точки («.»). Пример:

<?php

$a = "Hello, "; $b = "World";

$ c = $a . $b;

echo($ c);

?>

По аналогии с арифметическими операциями есть сокращенный оператор конкатенации .= $а .= $b;

эквивалентна

$а = $а . $Ь;

С помощью оператора «точка» можно конкатенировать несколько строк:

$d = $а . $b . "some text" . $с;

Язык PHP предоставляет богатый арсенал функций для работы со строками, но так как обсуждение функций у нас ещё впереди, то дальнейшие средства обработки строк оставим на потом.

Логические операции

Логические операторы проверяют булевы условия. Существует четыре главных булевых условия: И (and или &&), ИЛИ (or или || ) , НЕ (!) и исключающее ИЛИ (хог).

Варианты использования логических операций.

$a = true;

$b = false;

$c = $a && $b;

$d = $a || $b;

$e = !$a;

$f = $a xor $b;

В данном случае получим следующие значения:

$c - false (один из операндов ложный),

$d - true (один из операндов истинный),

$e - false (не $a), $f - true (один из операндов истинный в то время как второй - ложный).

Приоритетность операторов

Когда операторы образуют последовательность, например:

$с = $а++ + 2 <= — $Ь \* 4 + $е && $f;

то все они выполняются согласно приоритетности. В любом справочнике по PHP можно найти порядок приоритетов операторов. Тем не менее, мы настоятельно не рекомендуем допускать таких последовательностей, так как они сильно снижают читабельность кода.

Всегда используйте скобки, для облегчения дальнейшего понимания собственного скрипта. В нашем случае может получиться что-то вроде:

$с = ($а++ + 2) <= ( ( —$Ь \* 4) + ($е && $f));

Согласитесь, здесь, по крайней мере, с первого взгляда видно, что $с - это результат операции сравнения.

Резюме

В этом Практическая работа вы познакомились с языком PHP и его ключевыми моментами. Теперь вы умеете объявлять переменные и использовать константы. Вы знаете, что язык PHP свободно типизирован, что, кстати, очень удобно для тех задач, которые он решает. Это вы обязательно оцените в дальнейшем.

Мы не рассмотрели битовые операции, поскольку нужно она крайне редко, а тема отнюдь не самая простая. Однако имейте в виду, что они есть и что PHP свободно может их осуществлять.

Информации освещенной в Практическая работа пока не достаточно для создания полноценного скрипта, но запаситесь терпением! Скоро начнется реальная практика!

Задания:

Перед выполнением домашнего задания ознакомьтесь со статьей «3 правила расстановки пробелов». Пишите код, руководствуясь этими правилами!

1. С помощью оператора echo выведите на страницу:
2. Строковую переменную
3. Целочисленную переменную
4. Переменную с дробного типа
5. константу
6. число в восьмеричной нотации
7. число в шестнадцатеричной нотации
8. Повторите вывод, заключив переменные в двойные кавычки (“). Посмотрите, что получится.
9. Повторите вывод, заключив переменные в одинарные кавычки (‘). Посмотрите, что получится.
10. Выведите в восьмеричной системе числа от 10 до 20 (echo 010; echo 011; echo 012; ... echo 019; echo 020;) Объясните результат.
11. Выведите 16 чисел в шестнадцатеричной системе, так чтобы в браузере отобразилось «0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15»
12. Выведите на экран любое четверостишие. Пример:

”Я помню чудное мгновенье:

Передо мной явилась ты,

Как мимолетное виденье,

Как гений чистой красоты.”

А.С. Пушкин

Для каждой новой строки используйте отдельный оператор echo. Каждая строчка должна быть отдельной строковой переменной. Также необходимо использовать переводы строки. После четверостишия поставьте инициалы автора и выделите их курсивом.

1. Выполните эти же действия, с помощью одного оператора echo.
2. Попробуйте в выражении использовать разные типы, например, сложите число «10» и строку «привет». Что получится? Объясните результат.
3. Дайте ответ на вопрос, как работает оператор xor? В каких случаях он возвращает значение true, в каких - false? Для этого напишите скрипт, который выводит значения операций со всеми возможными вариантами операндов (4 варианта). Чему равно $a xor $a для любых значений $a?