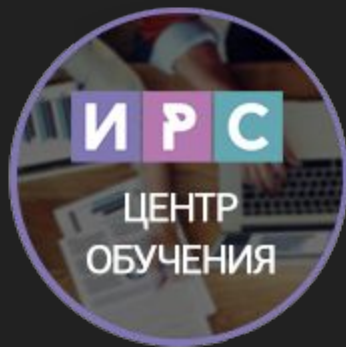


Курс вёрстки и программирования сайтов

goo.gl/gD76o7



Ямасыпов Виталий

вконтакте: vk.com/snake_yava

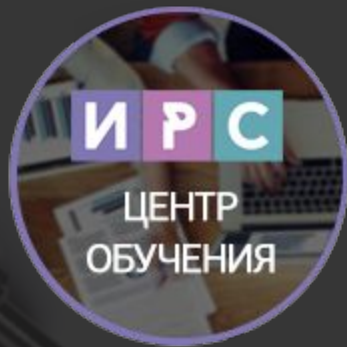
почта: snake-yava@mail.ru

skype: [snake-yava](https://www.skype.com/people/snake-yava)

icq: 366696661

Скачать Sublime Text

<https://www.sublimetext.com/3>



PHP

```
<!DOCTYPE [html PUBLIC  
<html xmlns="http://www  
<head>  
  <meta name="TITLE"  
  <meta http-equiv="c
```

Принцип работы PHP

В основе работы службы WWW лежит технология «клиент-сервер».

Веб-сервером называют специальную программу, обеспечивающую работу сайта, а также компьютер, на котором она работает.

Веб-сервер обрабатывает запросы, полученные через Интернет от браузера (клиента) и выдает в ответ нужный ресурс: HTML-код страниц, изображения, видеоролики и т.п.

Часто требуется не просто выдавать пользователю готовые ресурсы, а производить вычисления «на лету» и выдавать их результат. В таких случаях возможностей HTML недостаточно и необходимо создавать специальные программы с помощью языков программирования. Одним из таких языков является PHP.

Принцип работы PHP

PHP (PHP Hypertext Preprocessor) – серверный язык создания приложений, ориентированный на веб-разработку. PHP код может быть внедрен в HTML-страницу и будет выполняться при каждом ее посещении.

Код PHP интерпретируется веб-сервером и генерирует HTML-код или другой вывод (например, графику), который отсылается браузеру пользователя.

Так как PHP интерпретируемый язык, он не требует компиляции (преобразования в машинный код в файле .exe) – программы хранятся на веб-сервере как обычные текстовые файлы.

Архитектура веб-приложений



Принцип работы PHP

PHP-программа запускается при вводе ее адреса в строке браузера или отправке ей данных формы.

1. Пользователь вводит логин и пароль в HTML-форму и жмет кнопку отправки.
2. Данные через Интернет отправляются на веб-сервер. Браузер начинает ждать ответа от сервера.
3. Веб-сервер запускает PHP-программу и передает ей введенный логин и пароль.
4. PHP-программа:
 - а. Подключается к базе данных.
 - б. Делает запрос к базе данных «существует ли пользователь с таким логином и паролем?».
 - в. Если ответ положительный, программа выводит страницу приветствия. Если отрицательный – сообщение об ошибке.
5. Веб-сервер отправляет ответ PHP-программы назад в браузер в виде HTML-кода.
6. Браузер обрабатывает HTML-код и выводит страницу на экран компьютера пользователя.

Синтаксис PHP

`<?php`

```
    echo "Мама мыла раму";
```

`?>`

Весь текст внутри тегов `<?php ... ?>` обрабатывается как PHP-код, вне тегов – как обычный HTML.

```
<h1><?php echo "Мама мыла раму"; ?></h1>
```

Синтаксис PHP

Разумеется, выводить текст в браузер можно и без использования PHP – для этого достаточно HTML. Но PHP предоставляет массу других замечательных функций, которые HTML недоступны. Например, функция `date()` выдает дату и время в различных форматах:

Код:

```
<?php
```

```
    echo "Текущее время на сервере:<BR>";
```

```
    echo date("r");
```

```
?>
```

Переменные

Требования к именам переменных:

- должны начинаться с \$.
- могут иметь любую длину и состоять из букв, цифр и знака подчеркивания (" _")
- не могут начинаться с цифры

Имена переменных чувствительны к регистру:

`$studentname` и `$StudentName` – разные переменные.

Типы данных

RНР поддерживает 4 базовых типа данных:

- булевский (boolean),
- целочисленный (integer),
- действительный с плавающей точкой (double),
- строковый (string)

и 2 комплексных:

- массивы (array)
- объекты (object)

Строки

Существует 2 типа строк: разбираемые и неразбираемые. Разбираемые строки заключаются в двойные кавычки. В таких строках происходит подстановка значений переменных и обработка управляющих последовательностей символов. В неразбираемых строках, заключенных в одинарные кавычки, этого не происходит. Пример: PHP-код:

```
<?php
    $age = 2011 - 1936; //74
    $string1 = "МГДДЮТ $age года<BR>";
    $string2 = 'МГДДЮТ $age года<BR>';
    echo $string1;
    echo $string2;
?>
```

В браузере: МГДДЮТ 74 года
 МГДДЮТ \$age года

Строки

Существует 2 типа строк: разбираемые и неразбираемые. Разбираемые строки заключаются в двойные кавычки. В таких строках происходит подстановка значений переменных и обработка управляющих последовательностей символов. В неразбираемых строках, заключенных в одинарные кавычки, этого не происходит. Пример: PHP-код:

```
<?php
    $age = 2011 - 1936; //74
    $string1 = "МГДДЮТ $age года<BR>";
    $string2 = 'МГДДЮТ $age года<BR>';
    echo $string1;
    echo $string2;
?>
```

В браузере: МГДДЮТ 74 года
 МГДДЮТ \$age года

Строки

Управляющие последовательности позволяют включить в строку специальные символы:

`\n` – перевод строки

`\r` – возврат каретки

`\\` - обратный слеш

`\$` - знак доллара

`'` – одинарная кавычка

`"` – двойная кавычка

Операции сравнения

Используются для сравнения двух значений. Выражения, в которых используются эти операции, возвращают значения истина (true) или ложь (false).

== равно

=== равно и относятся к одному типу

!= неравно

< меньше

> больше

<= меньше или равно

>= больше или равно

Пример: \$a = 5;

 \$b = 5.0;

\$a == \$b вернет true

\$a === \$b вернет false

Логические операции

! НЕ

&& И

|| ИЛИ

Полезные ссылки

[google.com](https://www.google.com)

www.php.su/learnphp

phpclub.ru/manrus/index.html

Задания

а) Напишите программу, которая вычисляет:

- площадь круга и длину окружности, исходя из радиуса
- площадь и периметр прямоугольника по двум сторонам

б) Напишите программу, которая переводит градусы Фаренгейта в градусы Цельсия.

Условные операторы

```
$age = 19;  
if($age >= 60)  
{  
    echo "Пенсионер<BR>";  
}  
if($age >= 18)  
{  
    echo "Совершеннолетний<BR>";  
}  
else  
{  
    echo "Несовершеннолетний<BR>";  
}
```

Условные операторы

Когда для принимаемых решений существует более двух возможностей применяется оператор `elseif`.

Пример:

```
if (($hour >= 6) && ($hour < 12)) {  
    echo "Доброе утро!";  
} elseif (($hour >= 12) && ($hour < 18)) {  
    echo "Добрый день!";  
} elseif (($hour >= 18) && ($hour < 0)) {  
    echo "Добрый вечер!";  
} else {  
    echo "Доброй ночи!";  
}
```

Условные операторы

Существует также тернарный условный оператор, который возвращает второй или третий операнд в зависимости от логического значения первого операнда.

Синтаксис:

условие ? выражение 1 : выражение 2

Например:

```
$x = 5;
```

```
echo "Число " . $x . " – " . ($x % 2) ? "нечетное" : "четное";
```

Оператор switch

предоставляет упрощенный способ выполнения задач, которые можно выполнить с помощью оператора if. Синтаксис оператора switch:

```
switch (выражение) {  
    case выражение1:  
        операторы  
        break;  
    case выражение2:  
        операторы  
        break;  
    ...  
    default:  
        операторы  
        break;  
}
```

Задания

а) Создайте логическое выражение, проверяющее, что число является

- отрицательным

- нечетным

б) С помощью примера из занятия, создайте страницу, фон которой меняется в зависимости от времени суток.

Циклы

Управляющие структуры для повторения (циклы) - позволяют выполнять повторения (итерации) блока кода.

```
while(условие)
```

```
{
```

```
    код, выполнение которого повторяется, пока условие истинно
```

```
}
```

Пример:

```
 $\$i = 1;$ 
```

```
while( $\$i \leq 5$ )
```

```
{
```

```
    echo  $\$i++$ ;
```

```
}
```

Циклы

Управляющие структуры для повторения (циклы) - позволяют выполнять повторения (итерации) блока кода.

```
while(условие)
```

```
{
```

```
    код, выполнение которого повторяется, пока условие истинно
```

```
}
```

Пример:

```
 $\$i = 1;$ 
```

```
while( $\$i \leq 5$ )
```

```
{
```

```
    echo  $\$i++$ ;
```

```
}
```

Циклы

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

Цикл `for` обычно используют, когда заранее известно сколько раз выполнится цикл, а в остальных случаях удобнее использовать `while`.

В любой момент можно прервать выполнение циклов с помощью ключевого слова `break`.

Можно пропустить оставшуюся часть операторов цикла и перейти к следующей итерации с помощью оператора `continue`.

Задания

- а) Создайте программу, которая выводит квадраты и кубы чисел от 2 до 20.
- б) Модернизируйте программу из п. 1: оформите вывод в виде таблицы.
- в) Используя вложенные циклы, выведите таблицу умножения в браузер.

Массивы

Массив – это именованная ячейка памяти, в которой хранится набор значений.

Хранящиеся в массиве значения называются элементами массива. Каждый элемент имеет связанный с ним индекс (также называемый ключом), который используется для доступа к элементу.

PHP поддерживает 2 типа массивов:

- с численными индексами
- и ассоциативные, которых в качестве индекса могут использоваться практически любые значения (обычно строки).

Массивы

- Численно индексируемые массивы
- Ассоциативные массивы

Для доступа к элементам массива используется конструкция
`$имя_массива[индекс]`

Так как такой тип массивов индексируется последовательными номерами, для его обхода удобно использовать цикл `for`. Следующий код выводит содержимое всех элементов массива `$fruits`:

```
for ( $i = 0; $i < 3; $i++ )  
    echo $fruits[$i] . "<BR>";
```

Ассоциативные массивы

В ассоциативном массиве можно задавать осмысленные названия. Создадим массив сотрудников, индексами которых будет должность, а значением – оклад.

```
$employees = array( "Сисадмин" => 15000, "Бухгалтер" => 20000, "Директор" => 50000);
```

Ключи и значения разделяются символами =>.

Доступ к массиву осуществляется аналогично численно индексированному. Например, чтобы добавить элемент «Уборщица» можно использовать такой код:

```
$employees["Уборщица"] = 10000;
```

Цикл foreach

Для обхода массива используется цикл foreach.

Цикл foreach поэлементно проходит численно индексированный или ассоциативный массив от начала до конца. Синтаксис:

```
foreach ( $имя_массива as $ключ => $значение )  
{  
    код  
}
```

**Переменная для хранения ключа может быть опущена.*

Многомерные массивы

Каждый элемент массива может представлять собой массив. Таким образом, можно создавать многомерные массивы. Создадим двумерный массив работников предприятия:

```
$employees = array(  
    array("Фамилия" => "Иванов", "Должность" => "Сисадмин", "Оклад" => 15000),  
    array("Фамилия" => "Петрова", "Должность" => "Бухгалтер", "Оклад" => 20000),  
    array("Фамилия" => "Сидоров", "Должность" => "Директор", "Оклад" => 50000)  
);
```

Многомерные массивы

Для вывода массива используем следующий код:

```
echo '<TABLE border="1">';
echo '<TR><TD> Фамилия</TD><TD> Должность </TD><TD> Оклад
</TD></TR>';
for($i = 0; $i < 3; $i++)
{
    echo '<TR><TD>' . $employees[$i]["Фамилия"] . '</TD><TD>' .
    $employees[$i]["Должность"] . '</TD><TD>' .
    $employees[$i]["Оклад"] . '</TD></TR>';
}
echo '</TABLE>';
```

Задания

а) Создайте и выведите на экран массив, содержащий имена учащихся в группе.

Функции

Функция – это именованная последовательность операторов, которая при необходимости может принимать параметры и возвращать значение. Для определения функции используется следующий синтаксис:

```
function имя_функции($параметр1, $параметр2 ...) {  
    операторы  
}
```

Функция может вернуть значение в программу при помощи оператора `return`.

При его вызове выполнение функции прекращается.

На имена функций накладываются следующие ограничения:

- Функция не может иметь то же имя, что и существующая функция
- Имя функции может содержать только буквы, цифры и символ подчеркивания
- Имя функции не может начинаться с цифры

* Не чувствительны к регистру

Область действия переменных

- определяет доступность переменной в программе:

Переменные, создаваемые внутри функции, доступны только внутри этой функции (локальные переменные). Параметры функции относятся к этому же типу.

Переменные, создаваемые вне функции, действуют до конца файла, но не внутри функций (глобальные переменные).

Для доступа к глобальным переменным внутри функции используется ключевое слово `global`.

Функции для работы с массивами

`count($имя_массива)` - возвращает количество элементов в массиве

С помощью функции `count()` можно легко обойти целочисленный массив целиком:

```
for ( $i = 0; $i < count($fruits); $i++ )  
    echo $fruits[$i] . "<BR>";
```

`shuffle($имя_массива)` - располагает элементы массива в случайном порядке

Сортировка массивов

а) Численно индексированные массивы

`sort($имя_массива)` - сортирует массив в порядке возрастания

`rsort($имя_массива)` - сортирует массив в порядке убывания

б) Ассоциативные массивы

`asort($имя_массива)` - сортирует массив по значениям в порядке возрастания

`arsort($имя_массива)` - сортирует массив по значениям в порядке убывания

`krsort($имя_массива)` - сортирует массив по ключам в порядке убывания

`ksort($имя_массива)` - сортирует массив по ключам в порядке возрастания

ДЗ

а) Создайте массив со списком учащихся группы. Отсортируйте его и выведите в браузер.

б*) Создайте функцию, возвращающую наибольший общий делитель двух чисел. Напишите программу, использующую эту функцию.

Передача переменных из HTML-форм

При отправке форм PHP-сценариям данные формы становятся доступны в программе в массивах `$_GET` или `$_POST` в зависимости от выбранного метода отправки формы. Массивы `$_GET` и `$_POST` являются суперглобальными, т.е. доступны во всей программе.

Это ассоциативные массивы, содержащие список ключей, представляющих имена элементов формы, указанные в атрибутах `name`, и ассоциированных с ними значений.

Создание форм

Формы HTML используются для передачи данных от пользователя PHP-приложениям.

Границы формы определяются тегом `<form>...</form>`

Атрибуты: `action="URL"` - адрес приложения, которому будут переданы данные формы (по умолчанию текущий URL)

`method="..."` - метод передачи параметров: `get` (по умолчанию) или `post`

Замечание: При отсылке параметров методом `get` данные присоединяются к URL запроса после знака вопроса (?) парами `ключ=значение`, разделенными

Создание форм

Замечание: При отсылке параметров методом `get` данные присоединяются к URL запроса после знака вопроса (?) парами ключ=значение, разделенными символом амперсанда (&).

Например: `http://www.yandex.ru/yandsearch?rpt=rad&text=HTML`

Такой способ отсылки небезопасен (например, при передаче паролей), т.к. все данные видны в строке браузера.

При использовании метода `post` данные передаются в теле запроса и не видны в браузере.

Создание форм

Пример формы (регистрация пользователя на портале):

HTML-код:

```
<FORM method="GET" action="test.php">  
  Имя пользователя  
  <INPUT type="text" name="username"  
  value="Гость"><BR>  
  Пароль  
  <INPUT type="password" name="password"><BR>  
</FORM>
```

Полезные ссылки

<http://www.php.su/phphttp/forms/>

<http://htmlbook.ru/faq>

Задания

а) Создайте программу, позволяющую пользователю выполнять основные арифметические действия, используя форму ввода. Задать проверку входных данных: делитель не должен быть равен 0.

б) Напишите программу, вычисляющую корни квадратного уравнения. Коэффициенты уравнения вводятся пользователем в форме.

в*) Создайте программу для перевода единиц измерения (напр. Футы-см, фунты-кг и т.д.)

Структуры включения PHP-файлов (библиотек)

`require`(имя_файла) и `include`(имя_файла) – включают и исполняют PHP-файл.

Отличие между ними состоит в том, что `require` при ошибке останавливает программу, а `include` только генерирует предупреждение.

Пример использования

```
<?php
```

```
    /*
```

Эта функция передаёт массив переменных в файл шаблона,
и вставляет содержимое этого файла на странице с помощью include

```
    */
```

```
function includeFileWithVariables($fileName, $variablesArray) {
```

```
    extract($variablesArray);
```

```
    include($fileName);
```

```
}
```

```
/* выводим содержимое файла page, передав в него массив с переменными */
```

```
includeFileWithVariables('views/page.php', $pageData);
```

```
?>
```


Пример использования

```
<?php
```

```
/*
```

Эта функция передаёт массив переменных в файл шаблона, возвращает в результате строку из html-тегов с "подставленными" переменными

```
*/
```

```
function template($templateName, $variablesArray = array()) {
```

```
    extract($variablesArray);
```

```
    ob_start();
```

```
    include(__DIR__ . '/../views/' . $templateName . '.php');
```

```
    $contents = ob_get_contents(); // данные сейчас здесь
```

```
    ob_end_clean();
```

```
    return $contents;
```

```
}
```

```
?>
```

Задание

- 1) Используя предыдущие примеры, разделить на шаблоны наш сайт сделанный на HTML.
- 2) В зависимости от GET запроса показывать соответствующее содержимое. В качестве оператора выбора можно использовать switch.

Задание

- 1) Дописать методы для получения данных из БД (добавление/редактирование/удаление статей и разделов блога).
 - 2) Вывести заголовки статей на странице Блог. При клике на ссылку должна открываться страница с полным текстом статьи.
-
- 1) Добавить форму для создания статьи на странице Блог.
 - 2) Добавить форму редактирования статей под каждой статьёй, а также ссылки для удаления.

Сессии в PHP

Сессии являются простым способом хранения информации для отдельных пользователей с уникальным идентификатором сессии. Это может использоваться для сохранения состояния между запросами страниц.

Идентификаторы сессий обычно отправляются браузеру через сессионный cookie и используются для получения имеющихся данных сессии. Отсутствие идентификатора сессии или сессионного cookie сообщает PHP о том, что необходимо создать новую сессию и сгенерировать новый идентификатор сессии.

Сессии в PHP

index.php

```
<?php
$a = "Меня задали на index.php";
?>
<html><body>
<?php
echo $a;
?>
</body></html>
```

dothings.php

```
<html><body>
<?php
echo $a;
?>
</body></html>
```

Если выполнить эти два скрипта, то на первой странице мы увидим надпись "Меня задали на index.php", а вторая страница будет пустой.

Сессии в PHP

При использовании сессий вся информация хранится не на стороне клиента, а на стороне сервера, и потому лучше защищена от манипуляций злоумышленников.

Любой скрипт, который будет использовать переменные (данные) из сессий, должен содержать следующую строчку:

```
session_start();
```

Эта команда говорит серверу, что данная страница нуждается во всех переменных, которые связаны с данным пользователем (браузером). Сервер берёт эти переменные из файла и делает их доступными. Очень важно открыть сессию до того, как какие-либо данные будут посылаться пользователю; на практике это значит, что функцию `session_start()` желательно вызывать в самом начале страницы, например так:

```
<?php
```

```
session_start();
```

```
?>
```

```
<html>
```

```
<head>
```

```
</head>
```

```
...
```

Сессии в PHP

index.php

```
<?php
// открываем сессию
[REDACTED]
// задаём значение переменной
$ SESSION 'a' "Меня задали на
index.php"
?>
[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]
"dothings.php"
[REDACTED]
[REDACTED]
```

dothings.php

```
<?php
// открываем сессию
[REDACTED]
?>
[REDACTED]
[REDACTED]
<?php
echo $ SESSION 'a'
?>
[REDACTED]
[REDACTED]
```

Сессии в PHP

Другие полезные функции и приемы для работы с сессиями:

unset(\$_SESSION['a']) - сессия "забывает" значение заданной сессионной переменной;

session_destroy() - сессия уничтожается (например, если пользователь покинул систему, нажав кнопку "выход");

session_set_cookie_params(int lifetime [, string path [, string domain]]) - с помощью этой функции можно установить, как долго будет "жить" сессия, задав `unix_timestamp` определяющий время "смерти" сессии. По умолчанию, сессия "живёт" до тех пор, пока клиент не закроет окно браузера.

session_write_close() - запись переменных сессии и закрытие ее. Это необходимо для открытия сайта в новом окне, если страница выполняет длительную обработку и заблокировала для вашего браузера файл сессий.

Задание

1) Доработать систему авторизации пользователя с помощью сессий.

Не показывать административные ссылки и формы редактирования для неавторизованных пользователей.

Полезные ссылки:

<https://google.com>

https://htmlweb.ru/php/php_session.php

<http://php.net/manual/ru/session.examples.basic.php>

<http://phpfaq.ru/sessions>