**trimSettype(… , “тип”)** – смена типа переменной

**Gettype($...)** – вывод типа

**Define(“…”, …)** – создание константных переменных

**Foreach($... as $... => $...) {…}** – создание цикла для массивов ( в первую вносим массив, во вторую ключ, а в третью значение ключа )

**Continue(2)** – пропуск родительського цикла ( 1 – этот цикл, 2 – родительський и т. д. )

**Continue(2)** – прерыв родительського цикла ( 1 – этот цикл, 2 – родительський и т. д. )

**List($... , $... , …..)** – этой фигне можно присвоить функцию, которая возвращает массив и элементы массива распределяться соответственно элементам списка

**Global $...** – в функции из локальной переменной делает глобальную

**Var\_damp($...)** – вывод всей информации об переменной

**get\_defined\_vars()** – вывод всех переменных и массивов файла ( возвращает массив всего, по этому для вывода нужна функция **print\_r()** )

**var\_dump(debug\_backtrace())** – вся информация про функцию, из которой вызываться

**empty($...)** – проверка на пустоту

**time()** – текущее время

**die(…);** - полное закрытие php кода и выход из страницы с текстом в душках

**json\_encode(…);** - массив PHP в JSON

**unset($...)** – удаление переменной, объекта, массива

**ini\_set('display\_errors', 1);** - задание значений в ini file

**error\_reporting(E\_ALL);** - тип ошибок

**file\_exists($\_SERVER['DOCUMENT\_ROOT'] . …)** – проверка на существование файла

**filter\_var($email, FILTER\_VALIDATE\_EMAIL)** – проверка по стандартам, 1 вход – переменная, 2 -константа

**function …( array/имя класса/тип… параметр)** – строгий запрос на какой-то тип данных или класс

**set\_error\_handler(“…”);** - функция, которая принимает функцию-обработчик ошибок, она срабатывает при появлении ошибки некорректного передачи аргумента в функцию, описанную выше

**move\_uploaded\_file($\_FILES[…]['tmp\_name'], $\_SERVER['DOCUMENT\_ROOT'] . "…/новое имя….");** - берет высланную в форме фотку или музыку и т. д. и перемещает в нужную папку с нужным именем

**namespace**

**namespace …;** - создание пространства имен в отдельном файле

**имя пространства\...();** - вызов функции из другого пространство имен

**use имя пространства\...;** - выписать функцию из пространства

**Строки**

**“{$...}…” “$... …”** – переменные в строках

**… .= …** - конкатенация

**strtolower($...)** – снятие загланых букв

**strtoupper($...)** – все буквы загланые

**ucfirst($...)** – первая буква загланая

**ucwords($...)** – все слова начинаются с заглавных

**strlen($...)** – длина строки

**strstr($..., «…»)** – вывод строки с указанного слова/символа

**str\_replace(“…”, “…”, $...)** – замена строки ( что, на что, где )

**preg\_replace(“…”, “…”, $..., $limit, $count)** – замена строки через регулярку ( что, на что, где ) важно двойные кавычки! Принимает массв, в последние ограничение

**Strpos(где, что)** – возвращает позицию элемента в тексте

**trim(…)** – убирает пробелы сначала и в конце строи

**str\_repeat($..., …)** – повтор строки

**substr($..., …, …)** – обрезать строку

**trops($..., «…»)** – позиция слова/символа

**strchr($..., «…»)** – псевдоним функции strstr

**explode(‘…’, $str)** – разбить строку на массив

**preg\_split(‘регулярка’, где, ограничение)** – разбить строку на массив через регулярку

**crypt($..., “…”)** – создание хеш функции

**preg\_match("…", $..., $array, … , $offset)** – проверка регулярным выражением 1 – да, 0 – нет, false – ошибка в 3 переменную массив для сохранения файлов, в 4 флаг (тип выхода через константу) и в 5 сдвиг

**preg\_match\_all($pattern, $str, $array, … , $offset)** – как и **preg\_match**, но не останавливается на одном и все записывает в 3 переменную , в 4 флаг (тип выхода через константу) и в 5 сдвиг

**strops(«…», $...)** – возвращает позицию элемента из регулярного выражения, или false

**Math**

**Abs(…)** – модуль

**Pow(число, степень)** – число в степень

**Sqrt(…)** – корень

**Fmod(что, на что)** – остаток от деления ( % )

**Rand(от …, до …)** – рандомное число

**round(… , …)** – округлить до такого-то количества знаков после запятой

**ceil(…)** – округлить в большую сторону

**floor(…)** – округлить в меньшую сторону

**is\_int(…)** – целое?

**Is\_float(…)** – дробное?

**Is\_null($...)** – пустая ли переменная?

**Arrays**

**$... = array(… , … , …..) или $... = [… , … , …..]** – создание массива

**print\_r($...)** – возвращает весь масив

**count($...)** – количество

**max($...)** – максимальное

**min($...)** – минимальное

**sort($...)** – сортировка

**rsort($...)** – перестановка

**implode(«…»,$..)** – превратить в строку и разбить на “…”

**in\_array(…, $...)** – проверка на наличие в массиве

**isset($...)** – заполнена ли переменная

**unset($...[…])** – удаление ячейки, или затирание

**end($...)** – переместить курсор в конец

**reset($...)** – переместить курсор в начало

**next($...)** – переместить курсор вперед

**prev($...)** – переместить курсор назад

**current($...)** – вывести текущее положение курсора

**array\_key\_exists($id, $...)** – проверка на наличие ключа

**Objects**

**$... = array(“…” => … , “…” => … , …..) или $... = [“…” => … , “…” => … , …..]** – создание объекта

**URL запросы**

**$\_GET** – массив, содержащий ключи и их значения, переданные в url строке

**$\_POST** – массив, содержащий ключи и их значения, переданные в форме ( проверка на наявность через функию isset($\_POST[“…”]) )

**$\_FILES** – массив файлов как фото видео итд…

**$\_FILES[…]['tmp\_name']** – сам файл

**rawurlencode($…)** – для url типа строки кодировка текста ( в часть до знака вопроса )

**urlencode($...)** – для url типа строки кодировка текста ( в часть после знака вопроса )

**htmlspecialchars($...)** – автокодировка спецзнаков html (только 4 )

**htmlentities($...)**– автокодировка спецзнаков html (все)

**Header(“…”)** – задание значений заголовка

**header(«Location: « . …)** – изменение директории ( url ) (после смены адреса нужно вызвать функцию

**exit;** которая е передаст лишних данных вродь )

( при передачи файла на сервер заголовок можно менять только с самого начала, перед передачей каких либо файлов в html, но можно подключить буферную страницу, в которой это можно делать в любой части кода, и по завершению ее работы скинуть все на сервак )

**ob\_start()** – ручное открытие буфера в коде ( можно задать автоматом в php.ini файле )

**ob\_end\_flush()** – ручное открытие буфера в коде

**headers\_list()** – возвращает массив значений заголовка

**Cookies & Sessions**

**$\_SESSION** – массив, содержащий cookie и их значения, переданные как обыкновенному массиву

SESSION имеют обработку на сервере, а не браузере пользователя и н имеют ограничений, в куках оно записываеться как ссылка на файл в срвре, не дая хакерам инфы

**$\_COOKIE** – массив, содержащий cookie и их значения, переданные в setcookie функции

**$\_SERVER** – данные сайта

COOKIE имеют обработку в браузере и могут быть вызваны тольк в самом начале работы кода, до первого пробела в html, они держат читаемую для всех информацию и их можно удалить вручную

**Serialize($...) / unserialize($...)** – создать из массива строку и наоборот, используется для куки и сесий

**setcookie($name, $value, time)** – установка cookies ( имя, значение, время жизни ) ( для времени используют функцию time() + … для установки время жизни от момента записи, время в секундах )

**setcookie($name, null, time() - …)** – удаление cookie, можно просто задать имя или имя + null( но сказали лучше **имя, null, time() – 3600** – или любое другое число )

**session\_destroy();** - удалить все сессии

**session\_name()** – получить или установить имя текущей сессии

**$\_SERVER['REQUEST\_URI']** – строка запроса

**Запросы**

**include(«…»)** – добавление файла php в другой код ( не выдаст ошибки при отсутствии части файла, например функции )

**require(“…”)** – добавление файла php в другой код (выдаст ошибки при отсутствии части файла, например функции )

**include\_once(«…»)** – добавить лишь раз ( при следующих вызовах игнорируется )

**require\_once(“…”)** – добавить лишь раз ( при следующих вызовах игнорируется )

**SQL**

**1.**

**$connection = mysqli\_connect(«localhost», «имя пользователя», «пароль», «имя базы»);** - соединение с базой данных ( в начале стр. ) ( в первом водиться тип сервера или id )

**2.**

**$... = «select \* …»;** - создание переменной для хранения данных

**$... = mysqli\_query($connection, $...);** - функция mysql в первый параметр передаеться база данных, а во второй переменная с текстом кода

создание переменной которая хранит значение было ли успешное захватывание файлов из базы, если нет то через if и функцию die скинуть все

**3.**

**$... = mysqli\_fetch\_row($...)** – записывает в переменную первую или следующую строку из базы, при окончании запишет null, что выкинет из цикла while

**$... = mysqli\_fetch\_assoc($...)** – записывает в переменную первую или следующую ассоциативную (как массив но цельный ) строку из базы, при окончании запишет null, что выкинет из цикла while

**$... = mysqli\_fetch\_array($... , MYSQL\_NUM/MYSQL\_ASSOC/MYSQL\_BOTH)** – записывает в переменную первую или следующую сразу двумя типами, описанные выше, строку из базы, при окончании запишет null, что выкинет из цикла while

**В $**... храниться массив элементов

**4.**

**mysqli\_free\_result($...);** - возвращает обновленные данные в базу, переменная из функции mysqli\_query($connection, $...);

**5.**

**mysqli\_close($connection);** - закрытие базы данных

**mysqli\_connect\_errno($connection)** – возвращает код ошибки при подключении, можно использовать в if ибо при отсутствии ошибок вернет 0 и условие слетит

**mysqli\_connect\_error($connection)** – возвращает текст ошибки

**mysqli\_insert\_id($connection)** – возвращает последнее установленное id

**die(…);** - полное закрытие php кода и выход из страницы с текстом в душках

**mysqli\_affected\_rows($connection)** – количество измененных строк в sql

**mysqli\_real\_escape\_string($connection, $...)** – защита от sql injections, в строке можно передать незаметную команду (для взлома сайта), это функция экранирует все одинарные кавычки прочие спец символы, вторая переменная это та, которую нужно изменить и ей же присвоить результат

**mysqli\_num\_rows($result)** – возвращает количество рядков

**echo «selected», “checked”** – в формах через цикл и условие можно зафиксировать выбор

**strip\_tags($...)** – удаляет все html теги и знаки в тексте

**nl2br($...)** – включает переходы строк в тексте ( по умолчанию переходы не работают )

**Hashing**

**Md5(…)** – хеширование (старое )

**hash(‘…’, $...)**  - быстрое хеширование, не для паролей

**crypt($password, $salt)** – медленное хеширование для паролей (в соли больше 22 знаков)

**mt\_rand()** – рандомное значение, принимает 2 значения min и max

**uniqid($..., true)** – Сгенерировать уникальный ID, второй параметр, чтоб дольше было, генерация в зависимости от времени

**base64\_encode($...)** – Эта кодировка предназначена для корректной передачи бинарных данных по протоколам, не поддерживающим 8-битную передачу, например, для отправки тела письма. Данные, закодированные base64 занимают на 33% больше места по сравнению с оригинальными данными.

**ООП**

**Class … {…}** – создание класса

**Class … extends …{}** – наследование

**function \_\_construct(){}** – конструктор

**function \_\_destruct(){}** – деструктор

**Public/protected/private $... = …;** создание свойства со значением по умолчанию

**Public/protected/private function …() {};** создание метода

**Public/protected/private static $... = …**; - статический метод

**Public/protected/private static function() {}** – статическая функция

**Abstract …** - создание абстрактного класса или функции, при переопределении сигнатуры должны совпадать, но если в функцию передать объекты с значением по умолчанию, то все будет ок. При хоть одном объявлении абстракции класс становиться абстрактным **Abstract function …(…);**, можно создавать реализацию функций для наследования **extends …, …**

**Interface …** - класс интерфейс, в нем нельзя создавать обработку функций, можно наследовать много раз, содержит только обложку, для наследования **implements …, …**

**Final …** - финальный класс или функция, которые нельзя переопределить или наследовать

**Const … = …;** - создание констант

**Имя класса::…** или **self::…** - обращение к константам из класса и статическим

**Parent::…** - вызов функции базового класса, если при наследовании она переопределялась

**Self** – сам класс ( он один )

**This** – объект класса ( их много )

**$...::…** - обращение к константам и статическим

**$... = new имя класса;** - создание объекта класса

**$...->…** - обращение к свойствам класса

**$this->…** - ткущий объект

**call\_user\_func\_array(array($объект, $его функция), $параметры);** - фунция выполняет метод объекта и закидывает параметры

**Constants**

|  |  |
| --- | --- |
| **\_\_LINE\_\_** | Текущий номер строки в файле. |
| **\_\_FILE\_\_** | Полный путь и имя текущего файла с развернутыми симлинками. Если используется внутри подключаемого файла, то возвращается имя данного файла. |
| **\_\_DIR\_\_** | Директория файла. Если используется внутри подключаемого файла, то возвращается директория этого файла. Это эквивалентно вызову *dirname(\_\_FILE\_\_)*. Возвращаемое имя директории не оканчивается на слеш, за исключением корневой директории. |
| **\_\_FUNCTION\_\_** | Имя функции. |
| **\_\_CLASS\_\_** | Имя класса. Это имя содержит название пространства имен, в котором класс был объявлен (например, *Foo\Bar*). Обратите внимание, что начиная с PHP 5.4 \_\_CLASS\_\_ также работает в трейтах. При использовании в методах трейтов \_\_CLASS\_\_ является именем класса, в котором эти методы используется. |
| **\_\_TRAIT\_\_** | Имя трейта. Это имя содержит название пространства имен, в котором трейт был объявлен (например, *Foo\Bar*). |
| **\_\_METHOD\_\_** | Имя метода класса. |
| **\_\_NAMESPACE\_\_** | Имя текущего пространства имен. |

**Регулярные выражения**

**‘/…/’** – основной рахделитель

**‘/…[…-…]/’** – диапазон или через кому перечислить

**‘/…{…}/’** – количество повторений, или **‘/…{…,…}/’** – с ограничением, или **‘/…{…,}/’** – первое значение и более, или **‘/…{,…}/’** – до значения

**$str = "28-01-2001";** - строка со значениями

**$reg = "~([0-9]{2})-([0-9]{2})-([0-9]{4})~";** - регулярное с отдельными блоками, эти блоки в функции ниже идут под индексами соответственно

**$replacement = "year: $3, month: $2, day: $1";** - строка в которой будующие объекты заменены переменными, связь со строкой выше

**$replacement = preg\_replace($reg, $replacement, $str);** - функция для соединения ( регуляка, то куда будут записаны переменные, строка из которой берут переменные (практика ур2) )

**‘/[…]+/’** – от одного и больше