PHP QuickSheet

Форматирование

// однострочный комментарий
/* многострочный
комментарий
в php коде*/

Теги

<?php
(программа или фрагмент)
?>

Переменные

$name = "Дмитрий";
$age = 25;
$**c** = $age * 4;
$c++;
$fam = "Леонидов";
$fio = $name . " " . $fam;
// результат: Дмитрий Леонидов

Присваивание прямое и по ссылке

$tmp = 5.0;
$tmp2 = **&**tmp;// указатель на ячейку памяти с данными переменной
$tmp = 7.0; // обе переменные == 7.0

Переменная в переменной

$tmp = "varX";
$$tmp = "Cj Bank";
...
print $varX;
или
print **$**(**$**tmp);

Типы

boolean: TRUE, True, fALSE, false
int: 5, 591, 52
octal: 0422, 0534
hexadecimal: 0x3FF, 0x22abc
float: 12.42, 98.6
double: 3e8, 5.9736e24
string: "welcome", "49ers"
special: NULL
object.

Объекты

class apple{
var power;
function set_power($on_off)
{
  $this -> power = $on_off;
}
  }
...
$blender = **new** apple;
...
$blender**->**set_power("on");
...

Работа с классами

конструктор **new**
ссылка на внутр. методы **$this->**
наследование **extends**
деструктор **unset**($WebPage)
get_class_methods() - массив имен всех методов
get_class_vars() - массив имен атрибутов
get_object_vars() - ассоциативный массив обо всех атрибутах
method_exist() - есть ли метод в объекте
get_class() - имя класса указанного объекта
get_parent_class() - родитель для объекта
is_subclass_of(,) - создать объект на базе родительского класса
get_declared_classes() - массив всех определенных классов

Объявления переменных

локальные - не требуют декларирования, тип определяется при первом присвоении.
GLOBAL - в теле функции обращаться к глобальным: **$GLOBALS**["somevar"];
STATIC - значения не уничтожаются при выходе из функции
константы:
define("PI","3.1415926");
$pi-2*PI;// вызов без $ в имени

Результаты присвоений без явного приведения

1+"1" равно 2
5+"105 bottle" равно 110
5+"bottle 105" равно 5

Явное преобразование типов

$var2 = (**double**)$var1;
(**float**)
(**int**)
(**integer**)
(**real**)
(**string**)
(**array**)
(**object**)
$var1 = 1114;
$arr1 = (**array**)$var1;// появляется массив с первым элементом
$arr1**[0]**==1114
$model = "bmw";
$new_obj = (**object**)$model;// новый объект с автоматически созданным атрибутом

Вывод информации

echo "...";
print "...";
в **" "** - подставляются значения переменных
в **' '** - все выводится буквально
\" - экранирование кавычек
\n и др - служебные символы
блок текста
echo **<<< HERE**
...
HERE;// завершение блока располагают всегда с начала строчки

Операторы

+ сложение
- вычитание
++ инкремент
– декремент
++**$var** прединкремент
$var-- постдекремент
***** умножение
/ деление
% - остаток от деления
&& логическое И
|| логическое ИЛИ
!= не равно
= равно
+= прибавление к переменной
-= вычитание из переменной
***=** умножение переменной
/= деление переменной
== сравнение
=== сравнение величины и типа
? - тернарный оператор для замены мелких if-else
(условие) ? если Да : если Нет
. - конкотенация (сложение) строк

Условия

if ($a == $b) {...}
else {…}
[**elseif** ($c == $d){…}
else {…}
...
]
или
switch ($a)
{
case 1: ...
        break;
case 2: ...
        break;

default: ...
break;
}

Циклы

while (...) {...}
do {...} **while**(...)
for ($i=1;$i<20;$i++) {...}
continue;- переход в новую итерацию (пропуск команд)
break — прервать цикл полностью.

Массивы - одна из сильнейших сторон PHP

одномерный индексируемый
$name[] = "Маша";
$name[8] = "Иван";
или
$name = **array**("Маша","Саша", ...);
...
$name[]="Паша";//добавление нового элемента в конец
одномерные ассоциативные
$city ["Russia"] = "Москва";
$city ["USA"] = "NY";
...
или
$city = **array**("Russia" => "Москва", "USA" => "NY", ...);
многомерный индексируемый
$pos = $tmp[5][4];
многомерные ассоциативные
$auto["bmw"] = **array**("color" => "red", ...);
$auto["daewoo"] = **array**(...);
...
$auto["kia"]["color"] = "blue";
смешанные
$users["id"][1]=272;
перебор массива
foreach($menu **as** $item) {...}

PHP обработка массивов

только самое необходимое из безумного множества доступных функций PHP

count($arrX) - кол-во элементов в массиве
sizeof($arrX) - тоже,что count()
SORT() - по возрастанию
RSORT() - по убыванию
ASORT() - значения по возрастанию ! но индексы 0,1,2,3... перетасовываются следом за значениями 3, 0, 2, 1...
ARSORT() - тоже, по убыванию
KSORT() - меняет элементы, выстраивая по возрастанию КЛЮЧИ массива
KRSORT() - тоже по убыванию
USORT($tmp, "my_sort") - пользовательская функция сортировки
SHUFFLE() - перетасовка в случайном порядке
ARRAY_PAD ($tmp, 10, 100) - добавляет блок из 10 новых эл-тов +(в конец) - (в начало), заполняет их значением 100.
ARRAY_UNSHIFT() - новые в начало, остальные сдвигаются
ARRAY_PUSH($tmp, "..", "..", "..") - дописывает элементы в конец
ARRAY_POP($tmp) - вытягивает последний и тутже удаляет его из массива
ARRAY_SHIFT($tmp) - вытягивает первый, удаляет его, остальные сдвигает к началу
ARRAY_WALK($tmp,"func_name", [keys]) - применить функцию к нескольким элементам
ARRAY_REVERSE($tmp) - диаметральная перестановка элементов
ARRAY_FLIP($tmp) - ключи и значения меняет местами
RESET($tmp) - переходит и возвращает первый элемент
EACH($tmp) - возвращает пару "ключ/значение" и передвигается к следующему
END($tmp) - в конец
NEXT($tmp) — следующий. Если по пути будет неинициализ. элемент - подумает, что вышел за границу массива
PREV($tmp) — предыдущий. Если по пути будет неинициализ. элемент - подумает, что вышел за границу массива
array_count_values($tmp) - подсчет совпадающих значений - результат массив
array_keys($tmp[, "vip67"]) - массив ключей. 1 - если был поиск и найден
array_values($tmp) - возвращает ВСЕ значения
array_merge($mas1, $mas2, $mas3...) - сливает в кучу по порядку
ARRAY_SLICE($tmp,$otstup, [$dlina]) - вернет часть
array_splice($pasta,$otstup,[$dlina], [$znach]) - замена части массива (либо удаление выбранного участка)
RANGE(from, to) - новый массив целых чисел от и до
IN_ARRAY($id, $id_users) - истина/ложь поиск в массиве
LIST($id, $name, $age) = split("|", $resource_line);- заполняет переменные одним махом

Вложенность ссылок

$des["brown"]["col"]
то же
{ $des[brown][col] }

Строки - массивы символов

trim($w) - убираем пробелы с двух сторон
split("|",$line) - разбить строку на подстроки по разделителю
chop() - удаляет завершающие пробелы и перевод строки
str_pad() - добавление в середину подстроки
strtolower() - к нижнему регистру
strtoupper() - к верхнему
ucfirst() - первый символ заглавным
ucwords() - первую букву каждого слова заглавной
strcmp() - сравнение строк с учетом регистра
strcasecmp() - сравнение без учета регистра
strtok(,) - разбивка на лексемы
parse_str() - выделяем пары var=value &... и создаем переменные
explode("|",$line) - массив фрагментов
implode("|",$line) - собирает строку из массива
strlen() - длина строки
str_replace(строка1, строка2, строка [, число]) - заменяет в строке все вхождения строка1 на строка2. Начиная с PHP5 можно задавать число замен.
stripslashes(строка) удаляет экранирование символов
substr(string str1, int start[, int length]) - возвращает часть строки. Первый аргумент – строка;второй – положение первого символа (с нуля);третий – длина строки в символах, которую надо вернуть. Если третий аргумент не указан, то возвращается вся оставшаяся часть строки.
strpos(string str1, string str2[, int

offset]) - возвращает позицию в строке str1, в которой найдена подстрока str2. Необязательный параметр offset позволяет указать в строке позицию, с которой надо начинать поиск:
strrpos(string str1, string str2) - ищет в строке str1 последнюю позицию, где встречается str2.
strstr(string str1, string str2) - возвращает часть str1, начиная с первого вхождения str2 и до конца строки. В случае неудачи функция возвращает false. Эта функция чувствительна к регистру. В случае, если str2 не является строкой, то значение преобразуется в целое число и используется как код искомого символа.
strchr(string str1, string str2) - работает абсолютно идентично функции strstr():
stristr(string str1, string str2) - аналогично функции strstr(), только является нечувствительной к регистру.
strrchr(string str1, string str2) - отличается тем, что осуществляет поиск последнего вхождения подстроки. Чувствительна к регистру. В случае, если str2 не является строкой, то значение преобразуется в целое и используется как код искомого символа.
substr_count(string str1, string str2) - находит количество вхождений фрагмента в строку.
strspn(string str1, string str2) - определяет присутствие начальных символов в строке. Она возвращает длину начального фрагмента строки str1, состоящего полностью из символов, которые есть в строке str2.
strcspn(string str1, string str2) - обратная функции strspn(). Определяет отсутствие начальных символов в строке. Функция strcspn() возвращает длину начального фрагмента строки str1, состоящего полностью не из символов, которые есть в строке str2.
Отдельное внимание функции **preg_match**()
Не используйте **preg_match**(), для проверки подстроки в строке. Используйте для этого **strpos**(), **strstr**(), поскольку они гораздо быстрее.
* Символ "i" после закрывающего ограничителя шаблона означает регистронезависимый поиск.
preg_match("/php/i", "PHP is the web scripting language of choice.")
*Специальная последовательность **\b** в шаблоне означает границу слова, следовательно, только изолированное вхождение слова 'web' будет соответствовать маске, в отличие от "webbing" или "cobweb".
preg_match("/\bweb\b/i", "PHP is the web scripting language of choice.")
*комбинация последовательностей
preg_match("/\bweb\b/i", "PHP is the website scripting language of choice.")
Пример кода. Извлекаем имя хоста из URL:

```
preg_match("/^(http:\/\/)?([^\/]+)/i",
"http://www.groupnimb.com/index.html", $matches);
$host = $matches[2];
```

Примеркода. Извлекаем две последние части имени хоста:

```
preg_match("/[^\.\/]+\.[^\.\/]+$/",
$host, $matches);
echo "domain name is: {$matches[0]}\n";
```

Результат работы примера:
domain name is: groupnimb.com
Еще пример:

```
if ((($Brobot == 'Googlebot' and
!preg_match("~googlebot\.com~",
$user_host)) or
    ($Brobot == 'Yandex' and
!preg_match("~yandex\.ru~",
$user_host)) or
    ($Brobot == 'Yahoo' and
!preg_match("~yahoo\.net~",
$user_host)) or
    ($Brobot == 'Rambler' and
!preg_match ("~rambler\.ru~",
$user_host)) or
    ($Brobot == 'Msnbot' and
!preg_match("~search\.live\.com~",
$user_host))
){
$_BName = 'Robot';
$_Brobot = 'Unknown';
}
}
```

Функции

Можно обращаться и до, и после их описания

```
function square($a)
{
 $result = $a * $a;
 echo $result;
 return $result;
}
```

Модули - библиотека функций
template.inc
include() - подключение
require() - принудительное подключение
include_once() - с проверкой был ли уже подключен
require_once() - гарантия, что подключается 1 раз

Файлы

fopen($file, **"r"**) только чтение
"R+" чтение и запись
"w" только запись (начало)
"W+" начало с уничтожением
"a" только запись конец
"a+" запись в конец без уничтожения, если файла нет - создает
is_writeable() - проверка W
file_exists()
is_tile() - проверка R/W
filesize() - размер в байтах
fclose()
fputs()
fgetc()
fgets()
fgetss() - с удалением тегов PHP и HTML
fread()
fwrite()

другие функции

exit("message") - остановка и выход
$d = **date**("d-m-Y");- дата и время
d - дата число 12
D - день недели Fri
H - час 0..23
i - минуты 0..59
L - высокос.год
m - месяц 01-12 (!)
M - месяц Jan
h - месяц без 00
w - день недели 0..6
y - год 4
Y - год 2
z - день года 0..365
isset($a) - проверка существования
unset($a) - уничтожить

Передача данных

GET
POST
http://sashakrasnoyarsk.no-ip.org/
По материалам сайта
http://www.100pravda.com/