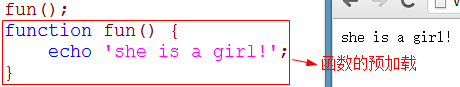


主讲教师：传智.神龙教主

传智播客 PHP学院 <http://php.itcast.cn> 发布

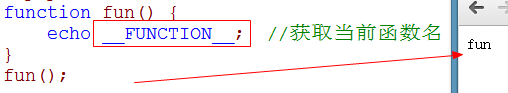
## 函数的预加载

代码的执行过程，词法分析——语法分析——编译——加载编译的代码——执行，在执行代码之前，函数已经加载到内存中了。



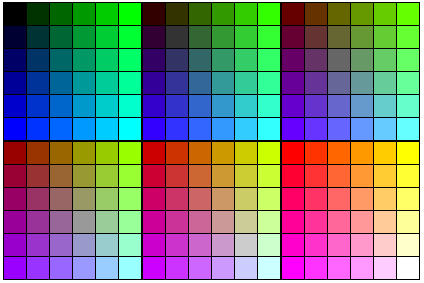
## 与函数有关的魔术常量

\_\_FUNCTION\_\_用来获取当前的函数名。



## 作业

### 打印颜色面板



<style type="text/css">

td{

border:#000 solid 1px;

width:20px;

height:20px;

}

table{

border-collapse:collapse;

float:left;

}

</style>

<div style="width:500px; margin:auto">

<?php

$arr=array('0','3','6','9','C','F'); //组成颜色的字符

$n=count($arr); //数组的个数钱

for($r=0;$r<$n;$r++){

echo '<table>';

for($b=0;$b<$n;$b++){

echo '<tr>';

for($g=0;$g<$n;$g++){

echo '<td style="background-color:#'.$arr[$r].$arr[$g].$arr[$b].'"></td>';

}

echo '</tr>';

}

echo '</table>';

}

?>

</div>

多学一招：DW中的常用快捷键

Ctrl+U：首选参数

Ctrl+J:页面属性

Ctrl+1：一级标题

Crl+2:二级标题 …

回车：换段

Shift+回车：换行

Ctrl+M:表格中插入一行

Ctrl+shift+M:删除表格的行

Ctrl+shift+A:在表格中插入一列

### 打印图形（一）

分析：

行数 \*的数量 变化后的行数 规则

1 1 1

2 3 2

3 5 3

4 7 4

5 9 5

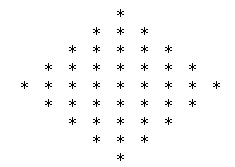
6 7 4 9-6+1

7 5 3 9-7+1

8 3 2 总行数-当前行+1

9 1 1

规律：星星的个数=行的编号\*2-1



代码

<style type="text/css">

span{

width:5px;

margin:0px 8px;

display:inline-block;

}

div{

text-align:center;

}

</style>

<div>

<?php

$total=9; //总行数

for($i=1;$i<=$total;$i++){

$row=$i; //$row表示变化后的行的编号

if($i>$total/2)

$row=$total-$i+1;

$n=2\*$row-1; //星星的个数

for($j=1;$j<=$n;$j++){

echo '<span>\*</span>';

}

echo '<br>';

}

?>

</div>

### 打印图形（二）

<div>

<?php

for($i=1;$i<=10;$i++){

if($i==1 || $i==2 || $i==9 || $i==10){

for($j=1;$j<=10;$j++){

echo '<span>\*</span>';

}

}else{

for($j=1;$j<=10;$j++){

if($j==1 || $j==2 ||$j==9 ||$j==10)

echo '<span>\*</span>';

else

echo '<span></span>';

}

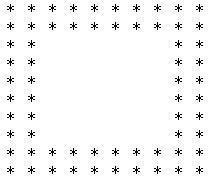
}

echo '<br>';

}

?>

</div>



### 自定义两个函数求数组中的最大值和最小值。



同理可以写出一个数组中的最小值。

### 递归法：因数分解



### 递归法：求最大公约数和最小公倍数（8251和6105）

分析：求最大公约数用更相减损法，比如$num1和$num2求最大公约数。$num2更到$num1的地方，$num2的位置保持$num1%num2的值。如果$num2的位置保存了0，则结束，$num1的保存值就是最大公约数。

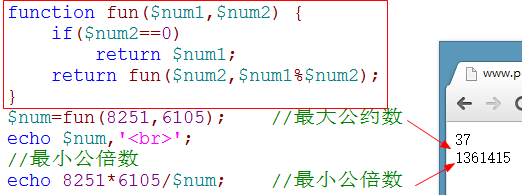
$num1 $num2

8 6

6 2

2 0

最小公倍数：$num1\*$num2/最大公约数



### 递归法：猴子吃桃子

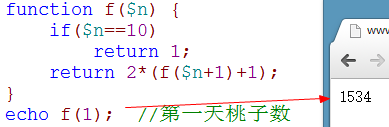
一个猴子守着一堆桃子，第一天吃了一半又多吃了一个，第二天一样，到第十天就剩下一个桃子，请问原来有几个桃子。

分析：

f(n)/2-1=f(n+1)

f(n)=(f(n+1)+1)\*2;

代码



## 包含文件

在网站中有相同的banner，版权等等页面，我们可以将相同的页面做成包含文件，在展示的页面中包含即可。

脚下留心：包含文件相当于将文件的内容复制到页面中，如果包含文件有

页面结构(<html></html>),包含后，页面中就不止一个页面结构了。这与W3C定义的规范不符合，所以在包含文件中不能出现页面的结构。

### 路径

相对路径：以自身的所在的目录为参照

绝对路径：以盘符或者站点根目录为参照



脚下留心：相对路径比较简洁，但可能由于页面之间多次相互包含，地址因此会出现问题。这种情况下，建议使用绝对路径。

../ 上一级目录

./ 当前目录

在html页面中/表示根目录。

### 包含文件语法

1. include()：可以包含多次
2. require()：可以包含多次
3. include\_once()：只能包含一次
4. require\_once()：只能包含一次

新建9-demo1.html

<?php

echo '锄禾日当午<br>';

?>

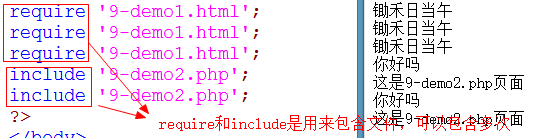
新建9-demo2.php

<?php

echo '你好吗<br>这是9-demo2.php页面<br>';

?>

新建9-demo3.php，用来包含9-demo1.html和9-demo2.php

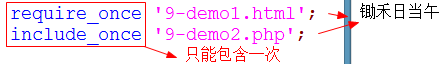
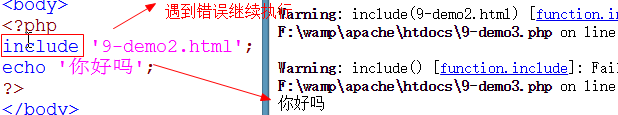
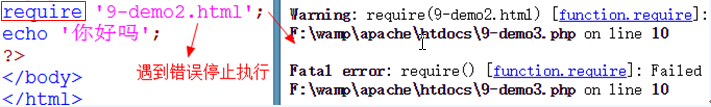


脚下留心：包含文件如果是html页面，在包含文件中有PHP代码，包含文件中的PHP代码是可以被解析的。

### require和include的区别

相同点：可以包含多次

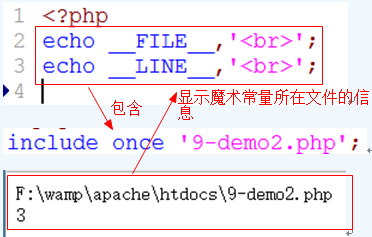
不同点：require遇到错误停止执行，include遇到错误继续执行。



思考：在项目中我们是选用require(include)还是选择require\_once (include\_once)?

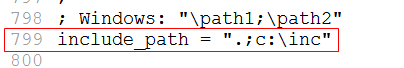
答：选择require(include),require\_once()在包含之前先要判断一下此文件是否被包含，如果没有包含则包含，否则不包含，在效率上显然没有requrie高。

多学一招：包含文件相当于拷贝包含文件的代码复制主文件中，魔术常量除外,魔术常量取的是魔术常量所在文件的信息。

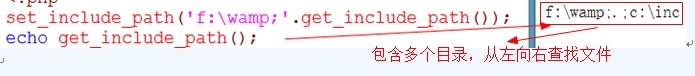


### ./目录/文件和目录/文件的区别

这两种写法的效果基本是一样的，不带./的路径受到include\_path配置的影响。



多学一招：可以通过set\_include\_path()来设置包含文件所在的目录。



set\_include\_path()：设置包含文件的目录

get\_include\_path()：获取包含文件的目录



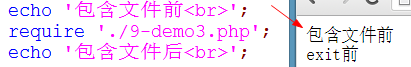
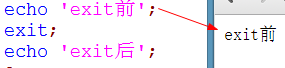
设置include\_path的作用：

1. 将项目的经常使用的文件放到一个统一的目录下，然后通过include\_path指向此文件夹，就不用写很长的地址字符串。
2. 将安全性高的文件放到站点外，通过include\_path指向此目录。比如连接数据库文件。

## 终止脚本执行

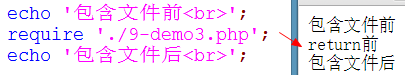
### exit()和die()

exit的die的别名，终止脚本执行，终止后可以输出一个字符串。



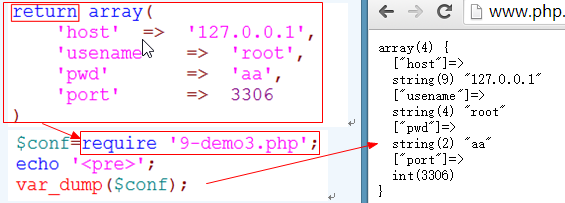
### return

终止当前页面的执行。



## return的作用

1. 终止函数执行
2. 从函数中返回值
3. 终止当前脚本的执行
4. 用来做配置文件



## 延时脚本执行

语法：sleep(秒)



## foreach数组指针

1. 指针其实也是一个变量，用来保存某个元素的地址。指针可以移动。
2. 默认情况下，打开一个数组，指针会指向数组的第一个元素



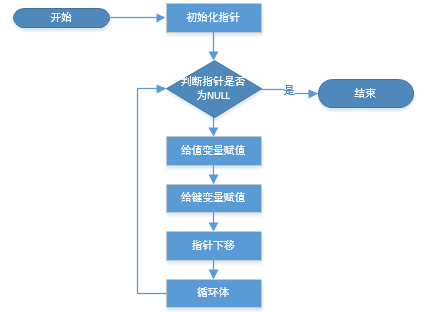
语法结构

foreach(数组 as 键=>值){

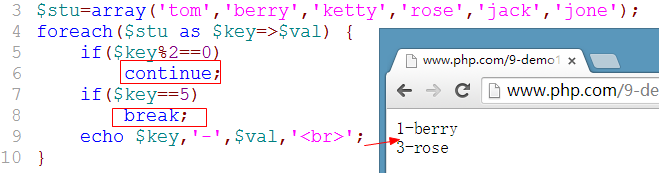
//循环体

}

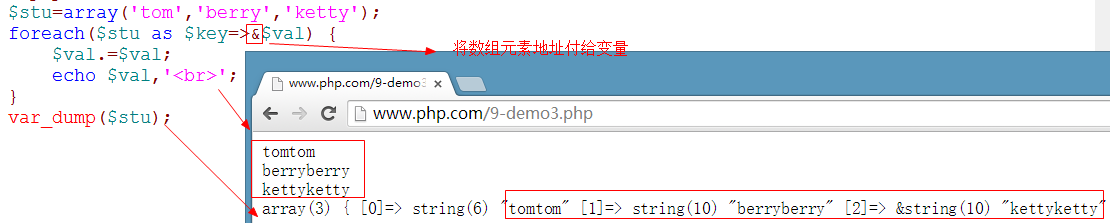
流程图



foreach是循环，break和continue也适用于foreach循环

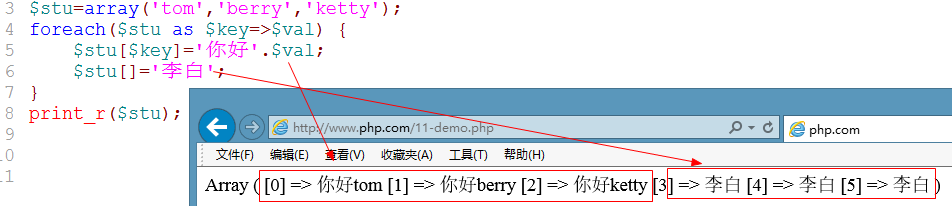


foreach值变量支持传地址



脚下留心：键变量不支持传地址。

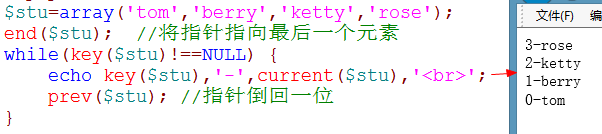
foreach循环变量数组的时候，遍历的是数组的拷贝



## 数组指针的操作

1. key()：从关联数组中取得键名，没有取到返回NULL。
2. current()：返回数组中的当前单元
3. next()：将数组中的内部指针向前移动一位
4. prev()：将数组的内部指针倒回一位
5. reset()：将数组的内部指针指向第一个单元
6. end()：将数组的内部指针指向最后一个单元

例题：将数组的元素从后往前取



脚下留心：如果一个数组连续输出两次，必须要将指针复位。

## each()

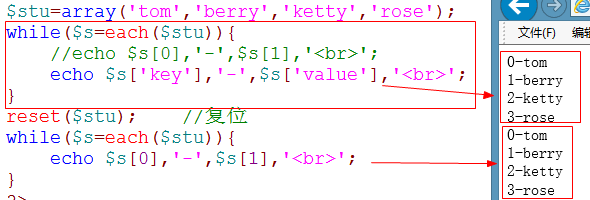
1、返回数组中当前的键／值对并将数组指针向前移动一步，如果内部指针穿越

了数组的末尾返回false。

1. 键值对被返回4个单元，其中包含索引数组和关联数组，下标是key和0的是键，下标是1和value的是值。
2. each()=key()+current()+next()
3. 如果第二次再遍历，必须通过reset()复位指针。



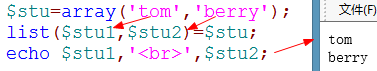
例题：通过while+each遍历数组



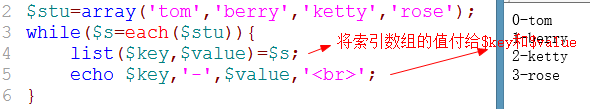
## list()

把索引数组中的值赋给一些变量

注意：只能是索引数组，不能是关联数组

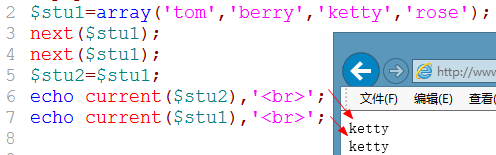


## each()+while()+list()



## 数组在复制指针的时候指针的处理

在数组复制的时候，如果指针合法，指针是一起复制的。



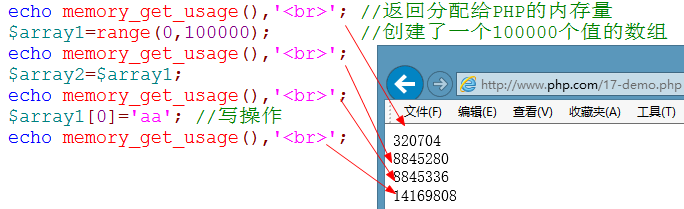
如果指针不合法，新数组指针会被重新初始化。**新数组就是先使用的数组。**



## COW(写时复制)

COW就是copy on write,在值传递的情况下，数据在赋值时候，内部存在一个优化，叫写时复制。

在值传递的赋值时，真实的数据空间不是立即复制的，而是在其中一个变量执行写操作的时候才会重新分配内存空间。



## 思考题

1. 阅读如下代码，输出什么？

$stu=array('tom','berry','ketty','rose');

foreach($stu as $key=>$val) {

}

var\_dump(current($stu)); //bool(false)

1. 阅读如下代码，输出什么？

$stu=array('tom','berry','ketty','rose');

foreach($stu as $key=>$val) {

$stu[4]='李白';

}

var\_dump(current($stu)); //string(5) "berry"

说明：foreach在第一循环时候，指针下移，指针指向berry,然后执行写操作，此时分配内存，由于指针合法，所有连同指针一起复制。所以输出berry.

1. 阅读如下代码，输出什么？

$stu=array('tom','berry','ketty','rose');

foreach($stu as $key=>$val) {

if($key==2)

$stu[4]='李白';

}

var\_dump(current($stu)); //string(4) "rose"

1. 阅读如下代码，输出什么？

$stu=array('tom','berry','ketty','rose');

foreach($stu as $key=>$val) {

if($key==3)

$stu[4]='李白';

}

var\_dump(current($stu)); //string(3) "tom"

1. 阅读如下代码，输出什么？

$stu=array('tom','berry','ketty','rose');

foreach($stu as $key=>$val) {

if($key==4)

$stu[4]='李白';

}

var\_dump(current($stu)); //bool(false)

## 作业

1、打印10行杨辉三角

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

2、打印等腰的杨辉三角