# 运算符

运算符：operator，是一种将数据进行运算的特殊符号，在PHP中一共有十种运算符之多。

## 赋值运算符

赋值运算：符号是“=”，表示将右边的结果（可以是变量、数据、常量和其它运算出来的结果），保存到内存的某个位置，然后将位置的内存地址赋值给左侧的变量（常量）。

## 算术运算符

算术运算：基本算术操作

+：执行数据累加

-：数据相减

\*：键盘上没有乘法符号，使用\*代替，两个数相乘

/：正斜杠代替，表示两个数相除

%：取余(模)运算，两个数（整数）相除，保留余数

在进行除法运算或者取余运算的时候，对应的除数（第二个数）不能为0



## 比较运算符

比较运算：比较两个数据的大小，或者两个内容是否相同，返回的结果都是布尔类型：满足返回true，不满足返回false

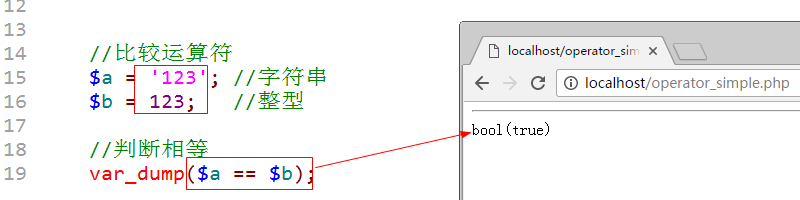
>：左边大于右边，返回结果true

>=：左边大于等于右边

<：左边小于右边

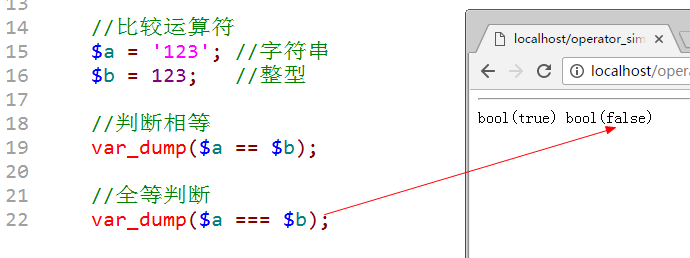
<=：左边小于或者等于右边

==：左边的与右边的相同（大小相同）



!=：左边的与右边的不同（大小不同）

===：全等于，左边与右边相同：大小以及数据的类型都要相同



!==：不全等于，只要大小或者类型不同就为true

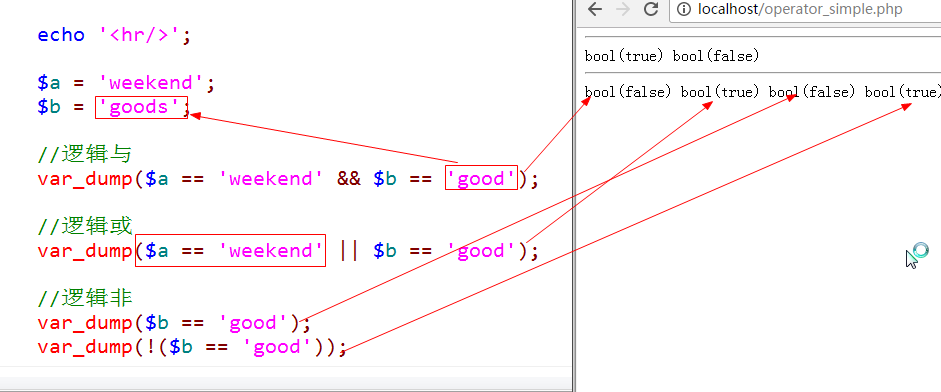
## 逻辑运算符

逻辑运算：针对不同的结果进行匹配。满足条件返回true，不满足返回false

&&：逻辑与，左边的条件与右边的条件同时成立（两边结果都为true）

||：逻辑或，左边的条件或者右边的条件只要有一个满足即可

！：逻辑非，对已有条件进行取反，本身为true，取反结果就是false



逻辑与和逻辑或又称之为**短路运算**：如果第一个表达式结果已经满足条件了，那么就不会运行逻辑运算符后面的表达式：**在书写代码的时候，尽量将出现概率最高的（能够直接判断出结果）的表达式放到第一位，提高运算效率**

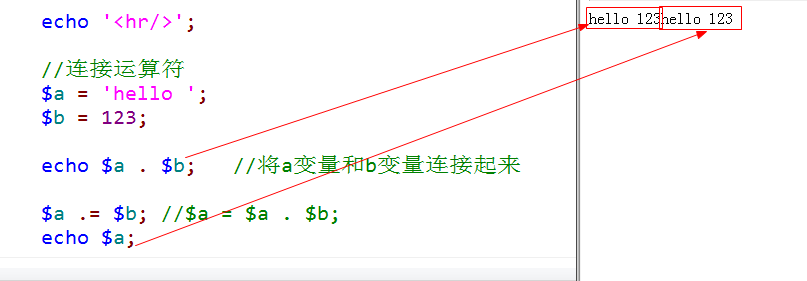
## 连接运算符

连接运算：是PHP中将多个字符串拼接的一种符号

. ：将两个字符串连接到一起

.= ： 复合运算，将左边的内容与右边的内容连接起来，然后重新赋值给左边变量

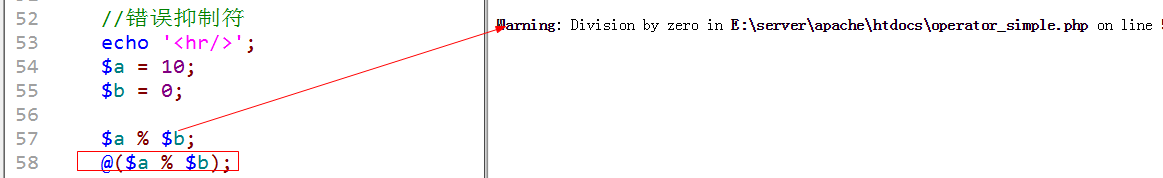
A .= b ⇔ A = A . b



## 错误抑制符(错误控制运算符)

在PHP中有一些错误可以提前预知，但是这些错误可能无法避免，但是又比希望报错给用户看，可以使用错误抑制符处理。

@：在可能出错的表达式前面使用@符号即可



错误抑制符通常在生产环境（上线）会用到，在开发的时候不会用：系统本身最好没有任何错误。

## 三目运算符

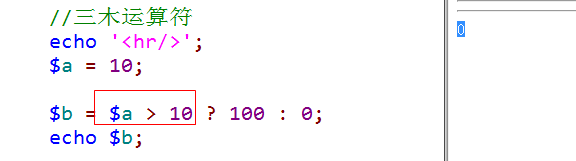
三目运算：有三个表达式参与的运算（简单的的分支结构缩写）

语法格式：

表达式1 ? 表达式2 ：表达式3；

运算：如果表达式1成立，那么执行表达式2，否则执行表达式3；

注意：如果表达式本身比较复杂，建议使用括号包起来。



三目运算可以进行复合三目运算：三目运算中的表达式2和3都是可以是另外一个三目运算。

表达式1 ？ (表达式2 ? 表达式4 : 表达式5) : (表达式3 ? 表达式5 : 表达式6);

## 自操作运算符

自操作：自己操作自己的运算符

++：在原来的值上+1

--：在原来的值上-1

$a = 1;

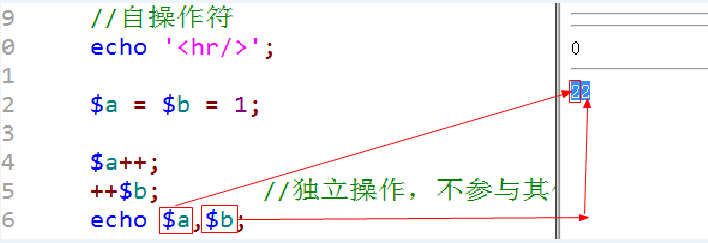
$a++; // $a = $a + 1;

在PHP中自操作符是可以放到变量前或者后：前置自操作和后置自操作

$a = 1;

$a++;

++$a; //前置或者后置如果本身只有自操作，不参与其他运算（自操作同时），那么效果是一样的。但是如果自操作同时还参与别的运算，那么效果就不一样



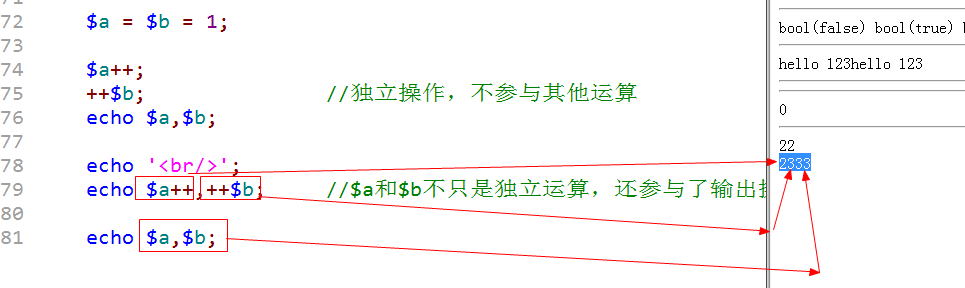
$a = 1;

$b = $a++; //$a++会导致$a = $a + 1; $a = 2;，上面的$b = 1

$c =++$a; //++$a会导致$a = $a + 1; $a = 2;，$c = 2;

后置自操作：先把自己所保存的值留下来，然后改变自己，自己给别人的值是原来的值；

前置自操作：先把自己改变，然后把改变后的值给别人。



衍生符号：类似自操作

+=：左边的结果与右边结果相加，然后赋值给左边

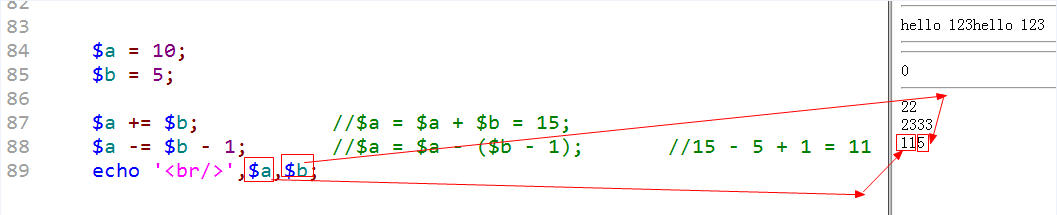
-=：左边的减去右边的结果，然后复制给左边

\*=：乘法操作

/=：除法操作

%=：模操作

注意：右边是一个整体 $a += $b; =🡺 $a = $a + ($b);



如果进行除法或者取余运算，那么要考虑右边表达式的结果是否为0（为0出错）

## 计算机码

计算机码：计算机在实际存储数据的时候，采用的编码规则（二进制规则）

计算机码：原码、反码和补码。数值本身最左边一位是用来充当符号位：正0负1

**原码**：数据本身从十进制转换成二进制得到的结果

正数：左边符号位为0（**正数的原码、反码和补码就是原码本身**）

负数：左边符号位为1

**反码**：**针对负数，符号位不变，其他位取反**

**补码**：**针对负数，反码+1**

系统中存在两个0：+0和-0

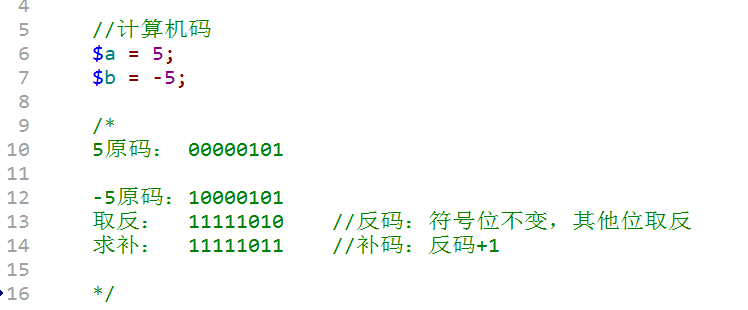
(**0的原码跟反码都有两个，因为这里0被分为+0和-0)**

+0： 00000000

-0： 10000000 原码

取反 11111111

补码 00000000

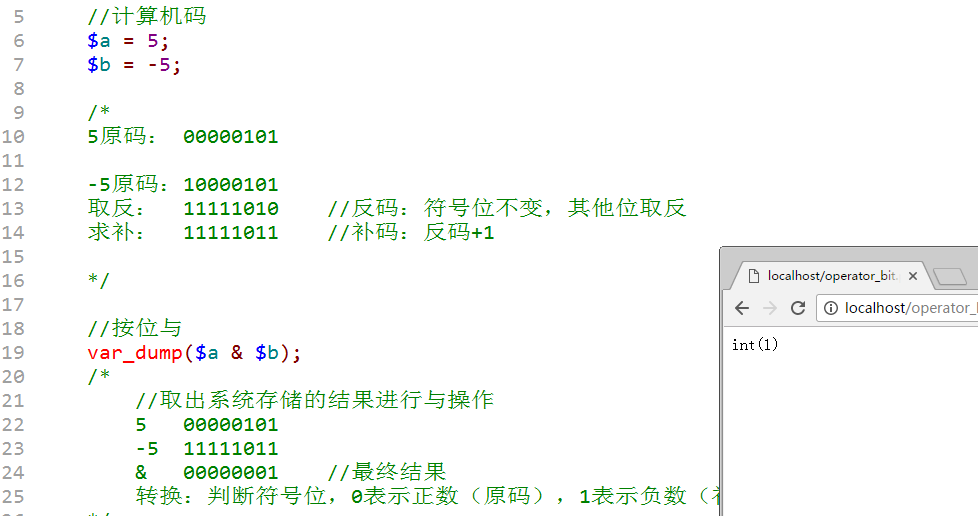


## 位运算符

位运算：取出计算机中最小的单位（位bit）进行运算

&：按位与，两个位都为1，结果为1，否则为0

|：按位或，两个有一个为1，结果为1



注意：

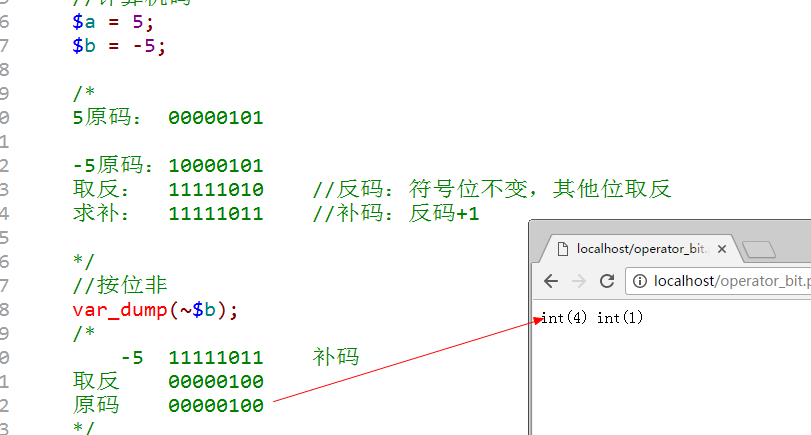
1. 系统进行任何位运算的时候都是使用的补码。

--计算机存储的是补码(为了实现化减为加--用加法处理减法)。用-5作为位运算时，-5在计算机内的二进制表示是二进制补码。

1. 运算结束之后都必须转换成原码才是最终要显示的数据

--补码的计算结果依然是补码，但是显示时需要转为原码：符号位为0(正数)的补码=原码 不用转换；符号位为1(负数)的需要把补码转换为原码。

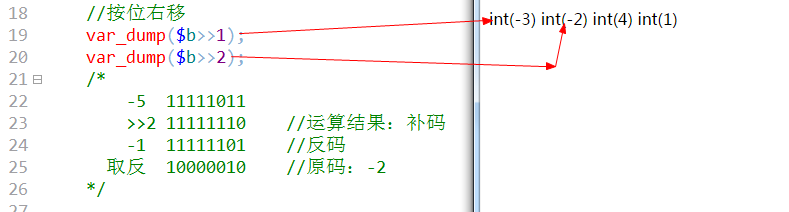
~：按位非，一个位如果为1则变成0，否则反之



^：按位异或，两个相同则为0，不同则为1

<<：按位左移，整个位（32位），向左移动一位，右边补0

>>：按位右移，整个位向右移动一位，左边补符号位对应内容（**正数补0，负数补1**）



按位左移：乘以2的操作

按位右移：除以2的操作（不完全正确）：整数除2会出现小数

## 运算符优先级

运算符优先级：在多种运算符同时存在的时候，如何结合运算



# 流程控制

流程控制：代码执行的方向

## 控制分类

顺序结构：代码从上往下，顺序执行。（代码执行的最基本结构）

分支结构：给定一个条件，同时有多种可执行代码（块），然后会根据条件执行某一段代码

循环结构：在某个条件控制范围内，指定的代码（块）可以重复执行

## 顺序结构

顺序结构：最基本结构，所有代码默认都是从上往下依次执行

## 分支结构

在PHP中，分支结构主要有两种：if分支和switch分支

### if分支

if：如果的意思，给定一个条件，同时为该条件设置多种（两种）情况，然后通过条件判断来实现具体的执行段

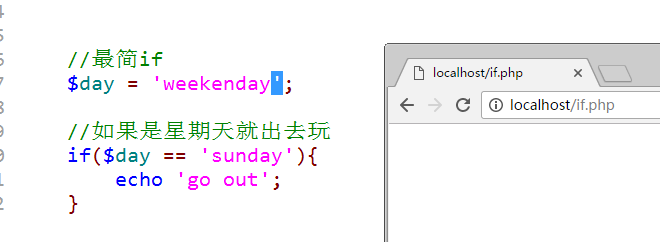
基本语法：if分支PHP也提供多种方式来实现

1)最简if：只有一段代码，但是可以选择是否执行

If(条件表达式){

//满足条件所要执行的内容; //顺序结构

}



2)基础if：有两面性，满足条件或者不满足条件都有对应的执行代码

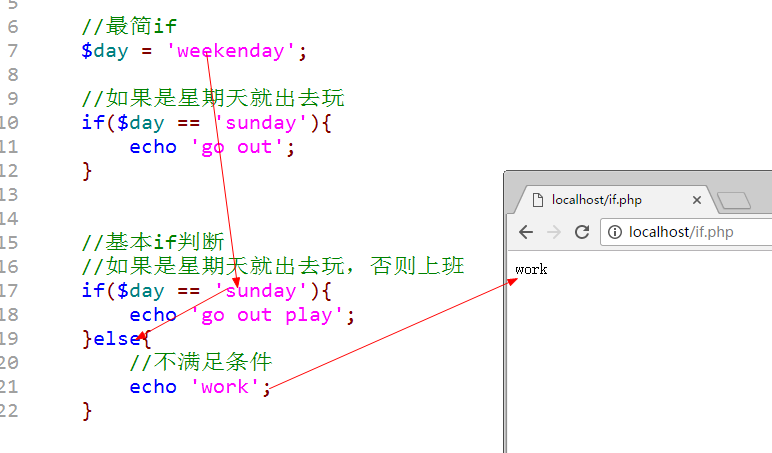
If(条件表达式){

//满足条件后执行的代码段;

}else{

//不满足条件执行的代码段;

}



3)复杂if结构：在判断条件之后，通常就有两种结果：满足或者不满足，在不满足之后还可以再次进行条件判断

if(条件表达式1){

//满足条件表达式1的代码段;

}elseif(条件表达式2){ //`else if 分开也可以

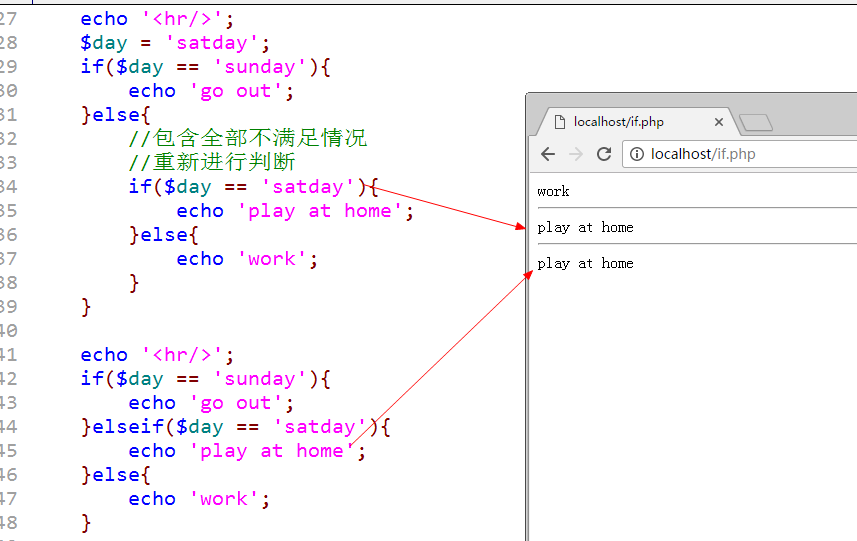
//不满足表达式1条件，但是满足表达式2的代码;

}…{ //可以使用多个elseif来进行再次条件筛选

}else{

//全部不满足要执行的代码;

}



注意：如果条件特别多才会采用复合if形式

1、 如果条件比较单一（同一个条件），会采用elseif复合方式

2、 如果判断条件不一致，建议使用嵌套语法（不宜有太多层嵌套：影响代码美观）

if分支，适用于所有的条件判断（分支结构）

### switch分支

switch分支：有一组情形存在，同过一条件，通常有多个值，但是每一个值都会有对应不同的代码要执行。

switch判断方式：是将条件放到分支结构内部判断

switch基本语法：

switch(条件表达式){

//所有条件判断：逐个进行

case 值1: //当前条件表达式的结果与值1相等（==）

要执行的代码段;

break; //在switch中，如果条件匹配成功，那么系统就不会再次匹配条件，会自动顺序执行向下的所有代码（case代码除外），需要中断执行：break表示中断switch（结束）。

case 值2:

要执行的代码段;

break;

…

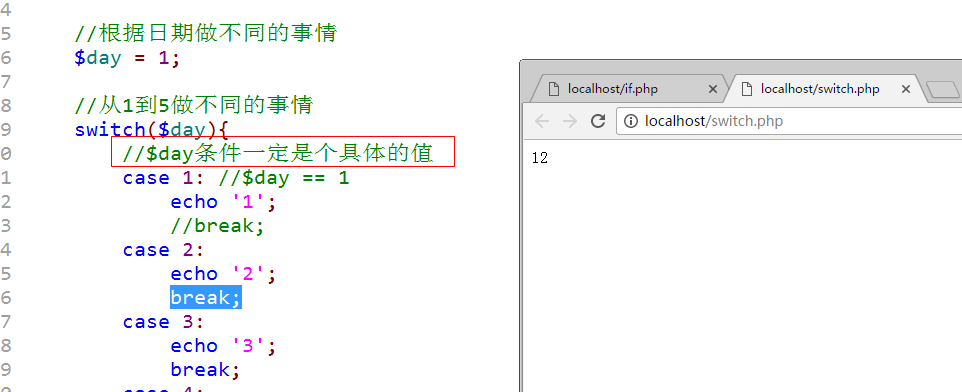
//可以使用类似else的语法：都不匹配

default:

//匹配失败的代码;

break;

}



if和switch的选择

1、if能做所有的分支结构事情

2、switch处理的是条件比较多，同时比较单一，而且是固定值匹配的分支结构

## 循环结构

循环结构：代码段在一定的控制下，可以多次执行

在PHP中循环结构有以下几种：

for循环：通过条件、起始和终止判断执行

while循环：通过判断条件终止

do-while循环：跟while差不多

foreach循环：专门针对数组

### for循环

for循环基本语法

for(条件表达式1;条件表达式2;条件表达式3){

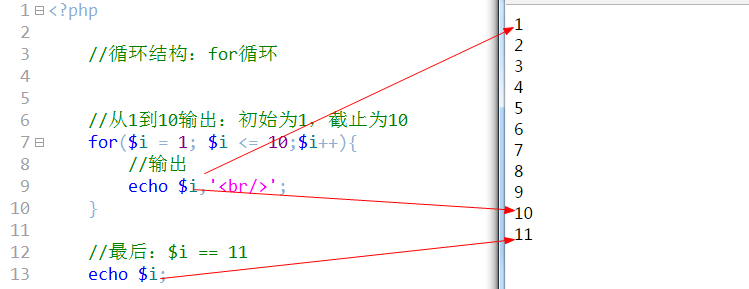
//条件表达式1：定义初始化条件，可以有多种赋值语句存在，使用逗号分隔即可

//条件表达式2：边界判定，限定循环执行的次数

//条件表达式3：用来执行条件变化（自操作）

//循环体

}



for循环执行原理：

1、执行条件表达式1：定义初始化条件（执行一次）

2、执行条件表达式2：判断条件（N次）

2.1 满足条件：执行循环体

2.2 不满足条件：循环结束

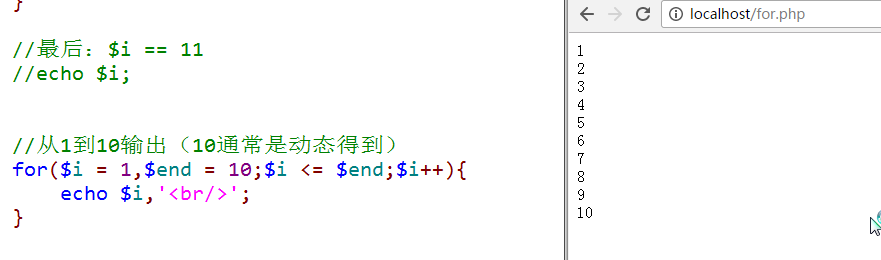
3、执行循环体：（N次）

4、执行条件表达式3：循环变量变化（N次）

5、执行条件表达式2：判断条件（N次）

6、重复执行3-4-2步骤：直到第2步不满足条件结束循环

for循环中条件表达式1的多变量定义

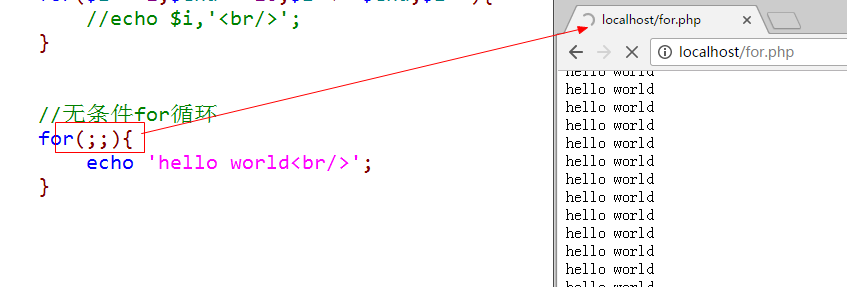


for循环特殊使用：for循环中对应的括号（条件）可以一个都没有（死循环）：一定要避免出现

for(;;){

//循环体

}



### while循环

while循环基本语法：

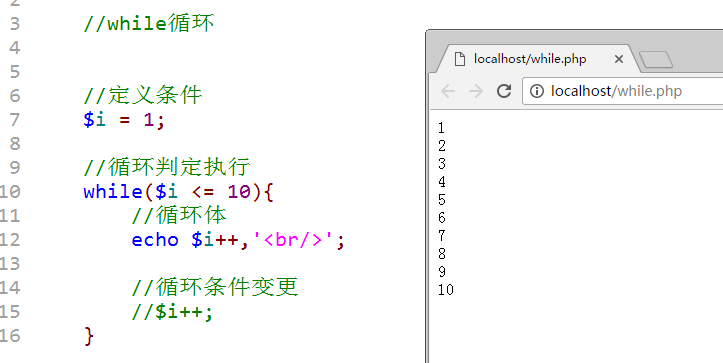
条件初始化;

while(条件表达式){

//条件表达式就是判断边界条件

循环体; //循环条件的变化

}



For与while的选择

1、如果是基于固定已知条件（数值而且是有规律的变化），使用for循环

2、while可以做灵活的条件判定（while使用的比较多）

### do-while循环

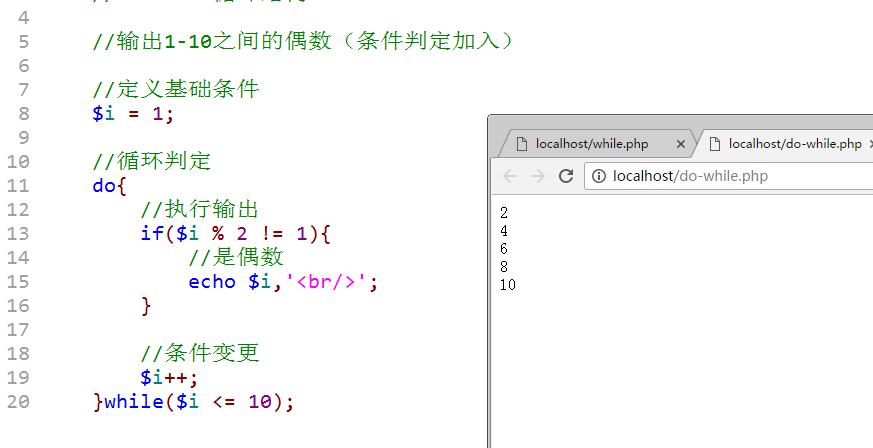
do-while：看着很像while，while首先进行条件判定然后执行循环体，有可能出现第一次就条件不满足，那么就会直接失败（循环体一次都不执行）。Do-while就是先干了再说（执行循环体），后判断条件。（至少会执行一次循环体）

do-while基本语法：

do{

//循环体

}while(条件表达式);

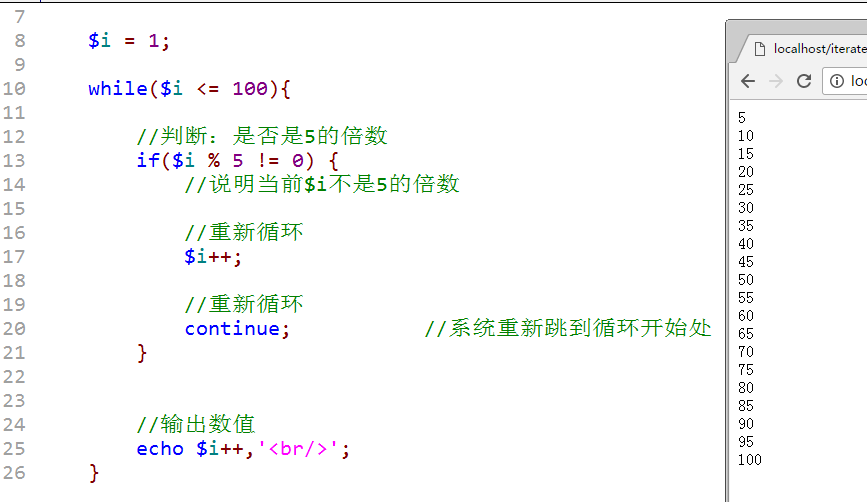


### 循环控制

循环控制：在循环内部对循环本身进行控制

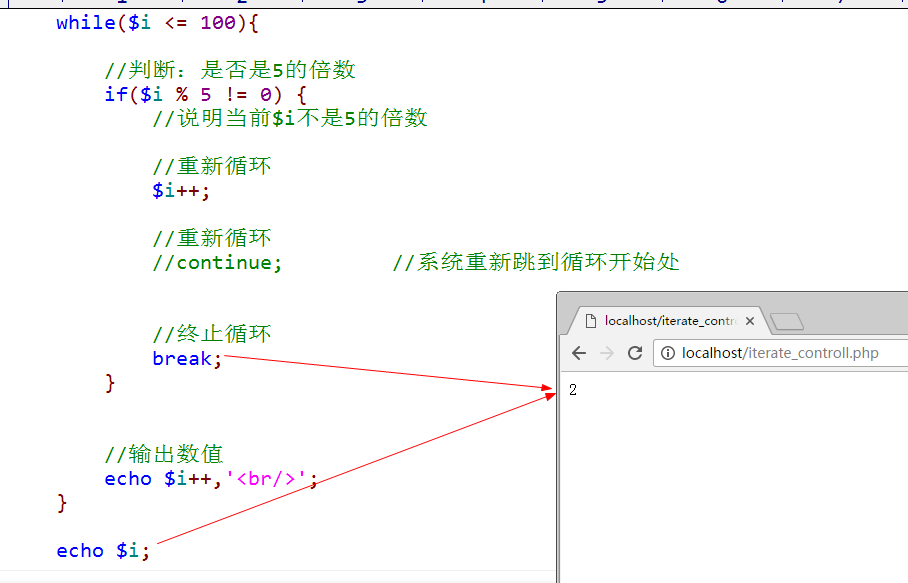
1)中断控制：重新开始循环，循环体中还有其他内容，也不执行，重新执行

Continue 层级; //层级默认是1（循环可以多层嵌套）



2)终止控制：循环直接结束

Break 层级; //层级默认是1



因为循环经常性会碰到嵌套（循环中间包含循环），如果在循环内部有些条件下，明确可以知道当前循环（或者说外部循环）不需要继续执行了，那么就是可以使用循环控制来实现：

其中内部循环也可以控制到外部，就是通过使用层级参数。

Continue 2; //当前自己循环后面内容不再执行，同时本层的外部循环如果还有循环体也不再执行，重新来过；

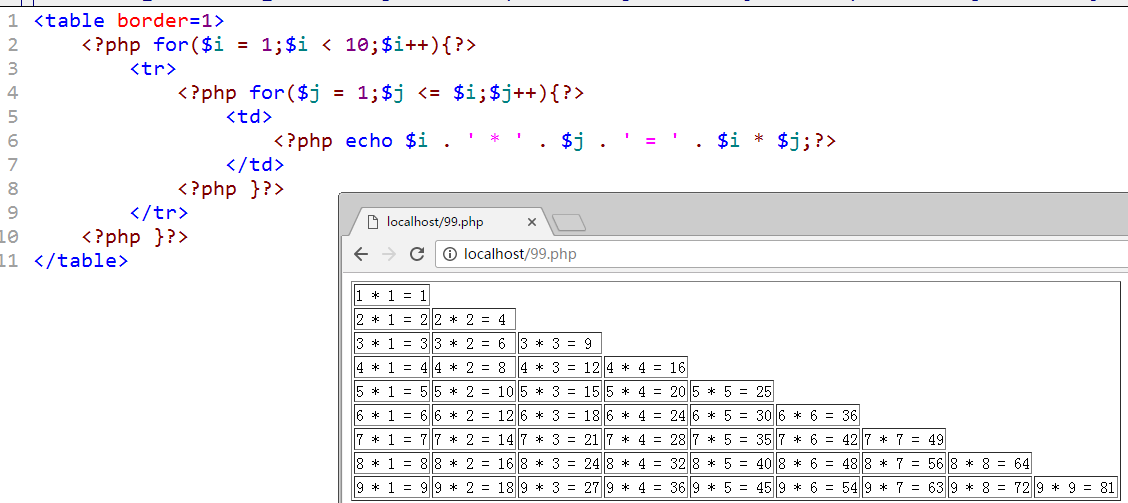
Break 2; //当前自己循环结束，同时本层的外部也结束（如果还有外部不受影响，继续执行）

## 流程控制替代语法

流程控制替代语法：分支和循环结构的替代语法

PHP本身是嵌入到HTML中的脚本语言，需要在HTML中书写一些关于判断或者循环的结构语法，必须符合PHP标签规范，需要HTML与PHP进行混搭，如果使用原始的PHP代码那么会非常不美观。

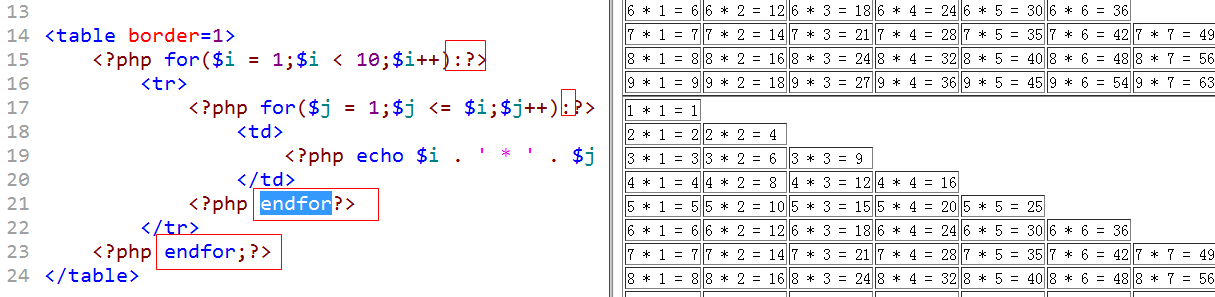
需求：打印一个九九乘法表，使用表格来展示



在PHP书写到HTML中的这些大括号{}非常不美观，所以PHP提供了一种替代机制，让其可以不用书写大括号：

For(;;){ =🡺 for(;;):

} =🡺endfor;



PHP中具体有哪些替代语法呢？

PHP应该在HTML中只做数据输出，输出通常伴有条件判断和循环操作，因此PHP提供了对应分支结构和循环结构的替代语法：全部都是对应的一个模式：

左大括号 { 使用冒号替代：

右大括号 } 使用end+对应的起始标记替代

if 🡺 if(;;): endif;

switch 🡺 switch(;;): endswitch;

for 🡺 for(;;): endfor;

while 🡺 while(;;): endwhile;

foreach🡺

# 完整版进qq群(300567032)获取