昨日回顾

php基本语法

标签4个: 标准推荐大家使用<?php ?>

语句结束符 ;

注释: //

#

/\*\*/

建议大家在晚上写代码的时候把一些概念可以抄一遍.

常见输出语句

print

echo

var\_dump: 可以打印数组,另外可以打印详细信息

printf:

print\_r: 可以打印数组

变量

$a = 100;

在内存中的区域分配问题.

变量的基本操作

增删改查

删: unset. 只做2步.将变量名移除,将引用移除.

变量间的传值

值传递

引用传递: 把一个地址传递给一个变量. 我们如何获取一个变量的地址? & 取地址符

预定义变量

认识9个预定义变量就可以.

$\_SERVER

$\_GET,$\_POST,$\_REQUEST

$\_COOKIE,$\_SESSION

$\_FILES;

$GLOBALS

$\_ENV

常量

define(“名”,值);

预定义常量

PHP\_OS;PHP\_VERSION;PHP\_INT\_SIZE;PHP\_INT\_MAX;

\_\_LINE\_\_; \_\_DIR\_\_; \_\_FILE\_\_;

# 第16讲 进制

现实生活中的进制: 十进制

一天 = 24H 二十四进制

1H = 60分钟 六十进制

半斤八两 1斤 = 16两 十六进制

一圈 = 360度

## 进制的概念

不同的进制。

X进制：满X进1，借1当X

### 二进制

就有2个数字 1和0

|  |  |  |  |
| --- | --- | --- | --- |
| 二进制 | 十进制 |  |  |
| 0 | 0 |  |  |
| 1 | 1 |  |  |
| 10 | 2 |  |  |
| 11 | 3 |  |  |
| 100 | 4 |  |  |

### 八进制

八进制只有8个数 ,从 0~7 .

0

1

2

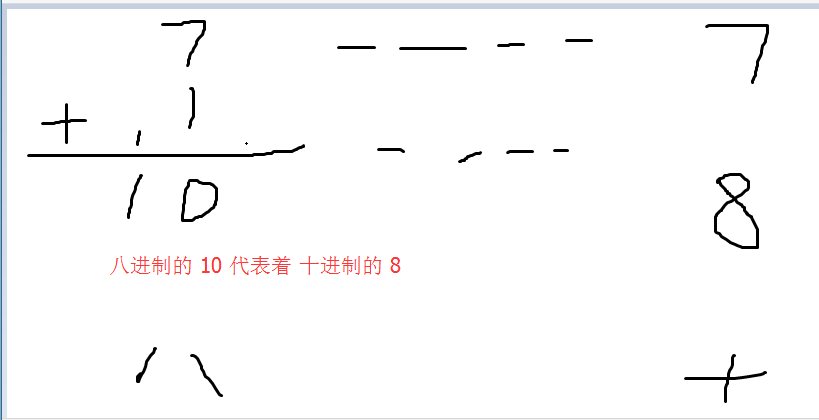
3

4

5

6

7



我想表示八进制的065.

033 表示 八进制

以0开头，此时的0不属于数值的一部分，也只是语法的一部分！

注意: 0开头的 八进制 数值. 里边数字的范围肯定是 0~7

如果我写一个 096 对不对??? 错误的!!!不是八进制

### 十进制

我们理解的十进制

### 十六进制

从0~9 十个数

从10开始，用字母来表示，10-15分别是A-F

0

1

2

3

4

5

6

7

8

9

a

b

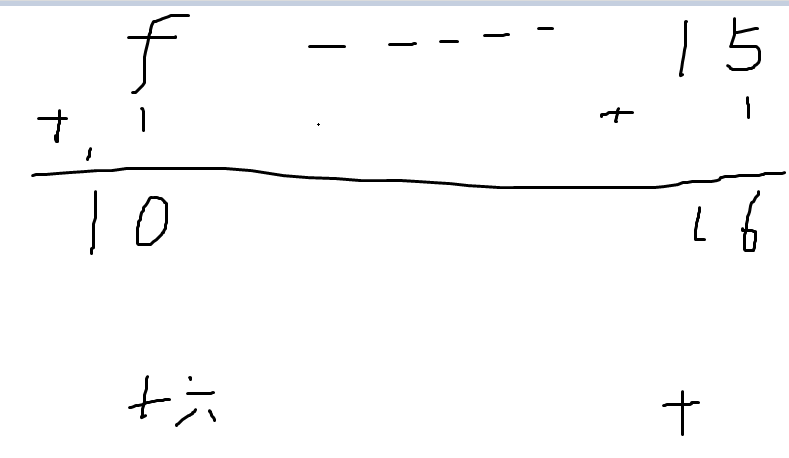
c

d

e e如果转换成十进制表示 14

f f如果转换成十进制表示 15

以0x开头，从10开始，用字母来表示，10-15分别是A-F



一定要注意 当我们学习了 进制之后 10 就已经不再是 我们理解的十进制10!!!

如果 10 是二进制那么 10这个二进制转换成十进制 表示 2

如果 10 是八进制那么 10这个八进制转换成十进制 表示 8

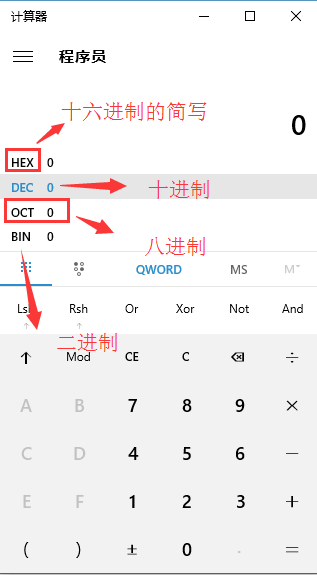
如果10 是十进制那么 10 这个十进制就是表示 10

如果10 是十六进制 那么10 这个十六进制就是表示十进制的 16

我写一个4f,大家猜一猜是多少进制? 十六进制 (不标准)

我写一个 89. 就不确定是哪个进制了!

0x 89 大家就知道是 十六进制



一般都是 其他进制转换成十进制.

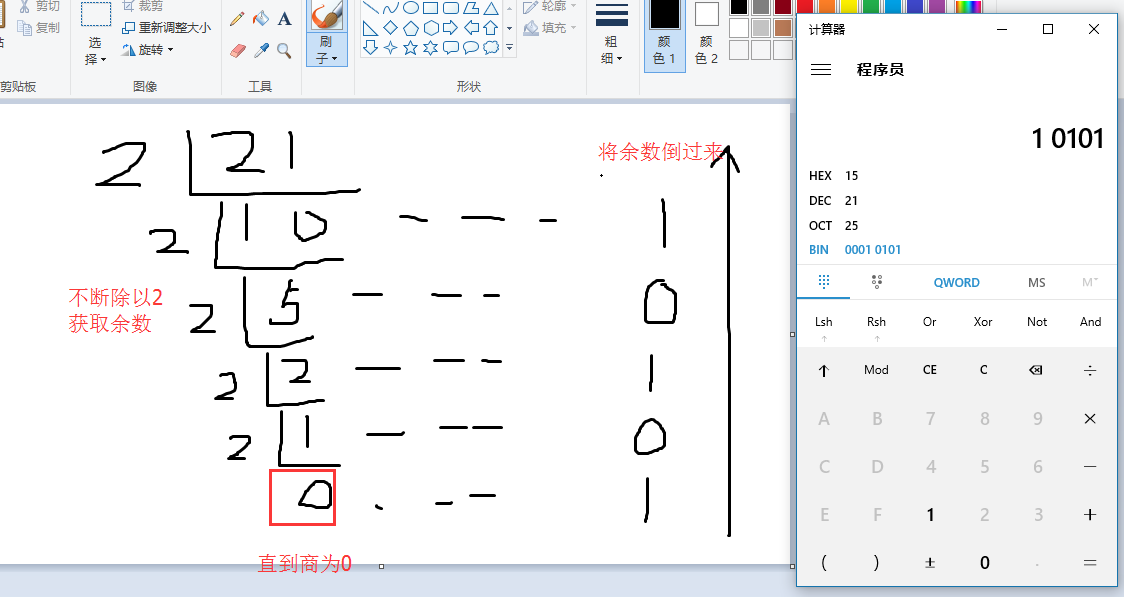
## 扩展

把一个十进制转换成二进制

十进制分整数和小数

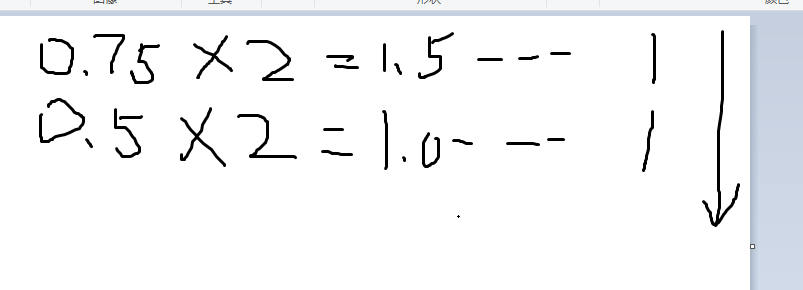
### 整数

我们把这个整数不断的除以2来取余,直到余数为0, 然后将 所有的余数倒过来.

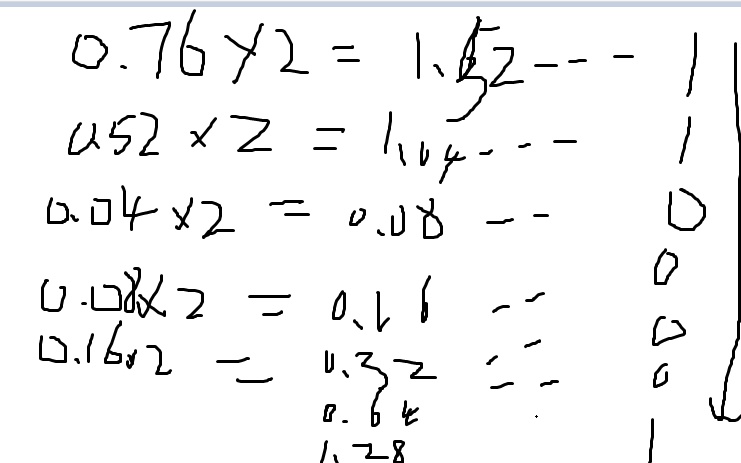


### 小数部分

小数部分不断的乘以2,来获取整数部分,剩余的小数部分再不断的乘以2.直到小数部分为0



小数部分 是以接近于真实值的形式存在的



二进制转换成十进制

加权计算

100 = 1

101 = 10

102 = 10\*10

10-1 = 0.1

10-2 = 0.01

(1234.56)10 = 1\*1000 + 2\*100 + 3\*10 + 4 + 5\*0.1 + 6\*0.01

= 1\*104-1 + 2\*103-1 + 3\*102-1 + 4\*101-1 + 5\*10-1 + 6 \* 10-2

= 数值 \* (进制)位数 -1

以上就是一个规律,我们按照这个规律来做事就行

(0110)2 = 0\*24-1+1\*23-1 + 1\*22-1 + 0\*21-1

= 0 + 4 + 2 + 0

= 6

(1111 1111)2 = 1\*28-1 + 1\*27-1 + 1\*26-1 + … 1\*22-1 + 1\*21-1

= 128+64+32+16+8+4+2+1

= 255

# 第17讲 整形数据

int，integer

整型数据占用了4个字节！也就是32个bit位！

一个字节 = 8个bit位

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

bit：比特，计算机中最小的存储单元，简称b

Byte：字节，计算机中最小的信息单元，简称B

1KB = 1K =1024B

1MB = 1M = 1024B

## 整形数据的表示形式

### 十进制

就是我们平时的十进制

### 八进制

以0开始的 ,数字范围是 0~7

037

089 错误

### 十六进制

以0x开始,数字范围是0~9,a,b,c,d,e,f

0x98

0xaf

fa 也是十六进制,只不过不标准

我们来打印一下0~100的 十进制十六进制和八进制

printf 函数,分两部分.第一部分如果遇到变量的时候我们采用占位符%

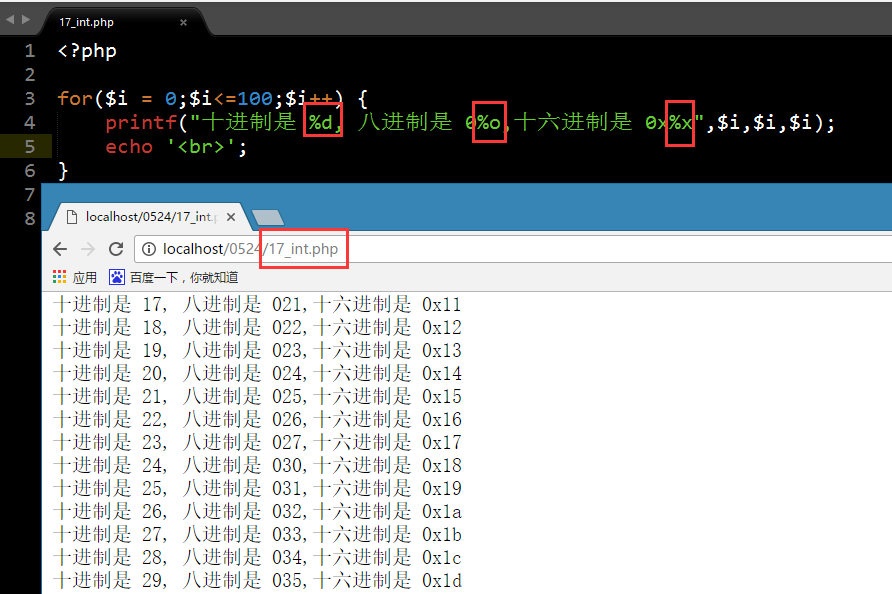
%d 表示 十进制整形

%s 表示字符串

%f 表示 浮点数

%o 表示八进制

%x 表示十六进制

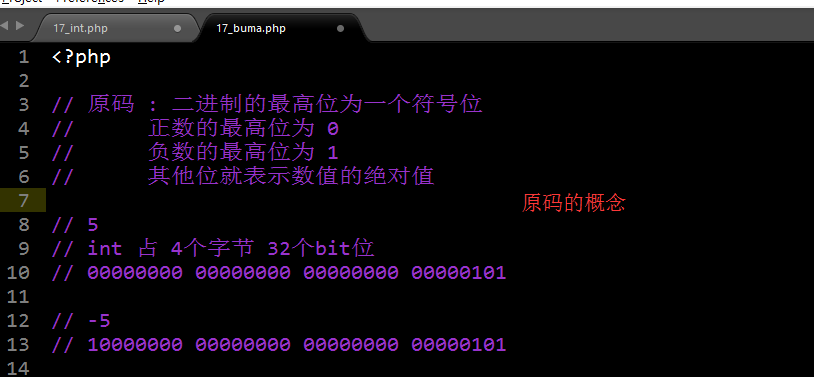


## 整形数据在内存中的形式(了解)

是以二进制的补码的形式存放的！

### 原码

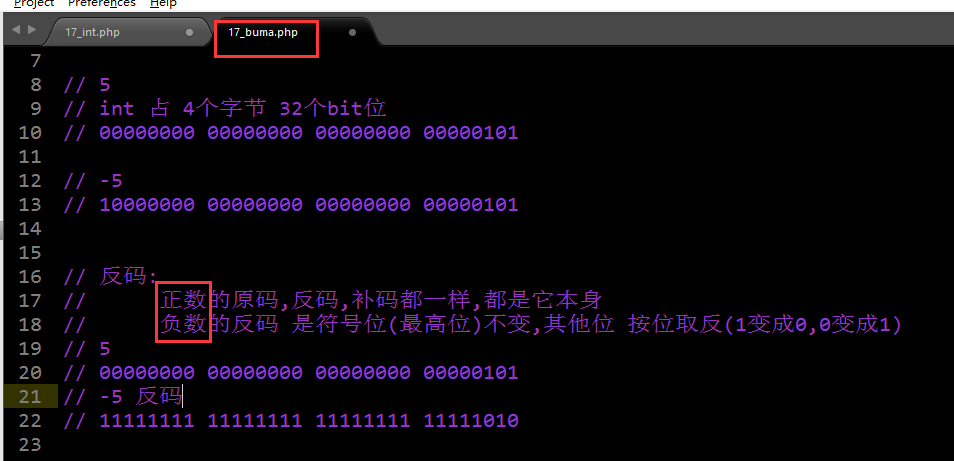
就是拿最高位作为符号位，其中1代表为负，0代表为正！



### 反码

正数：正数的原、反、补码全部相同！

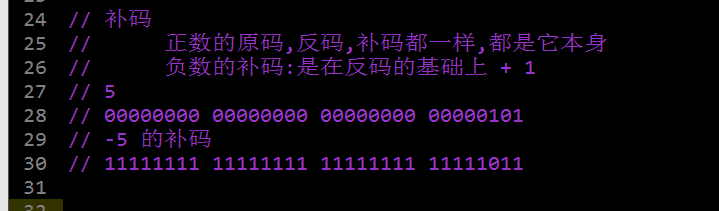
负数：除了符号位不变，其余的按位取反，所谓按位取反，是将1变成0，将0变成1

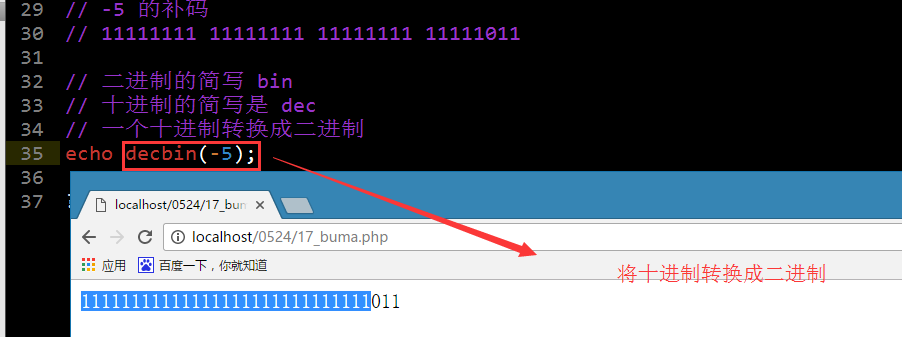


### 补码

正数：正数的原、反、补码全部相同！

负数：反码+1





# 第18讲 浮点型数据



* 小数。一般用float,double表示。
* 一般程序语言有单精度(占4个字节)、双精度(8个字节)之分（精度为小数的位数）。PHP只实现了双精度 ,占 8个字节。
* 一般小数部分为14位有效数字，表示范围为-1.8\*10^-308---- 1.8\*10^308

占据的空间为8字节

## 浮点型的表示形式

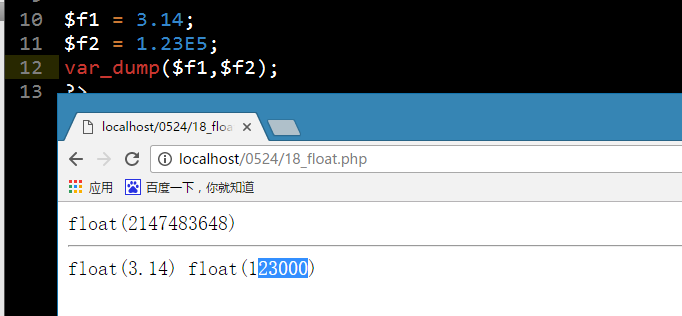
浮点数其实就是小数

小数有2种表示形式:

第一种: 小数点 3.14 ,123.456

第二种: 指数形式. 2.34e3 🡺 2.34\*103 = 2340

-1.23e-5 🡺 -1.23 \* 10-5 = -1.23\*0.00001

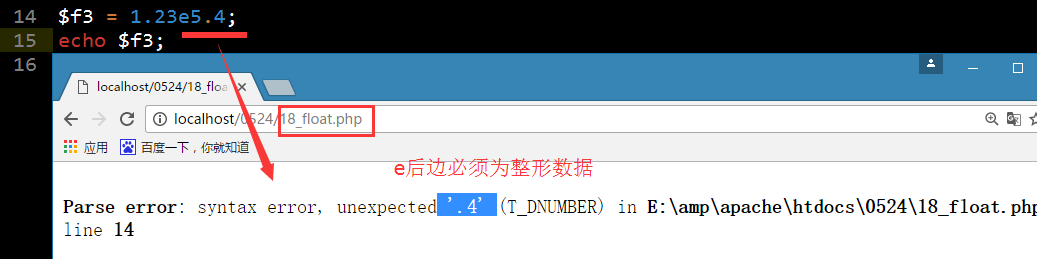


注意三点:

①e不区分大小写

② e的前后都需要数值

③e后边必须为整形



## 浮点数在内存中存放形式

以指数的形式存在的.

-1.23E4

1.23E-4

指数分为四部分: 小数部分,整数部分, 小数符号,整数符号

浮点数占8个字节,总共是64个bit位.

小数符号 和 整数符号分别占用一个 bit位

64 – 1 -1 = 62bit

因为 总的bit位是确定的. 如果 小数部分占的多,则 整数部分占的就少.说明精度高

如果小数部分占的少,则整数部分占的多,说明 范围大.

浮点数的有效位是 14位.

十进制的255转换成二进制占 8个bit



浮点数的范围和有效位要区分开:

1.8e1000000000000000000

一般我们的浮点数都在范围之内,很容易出了它的有效位数

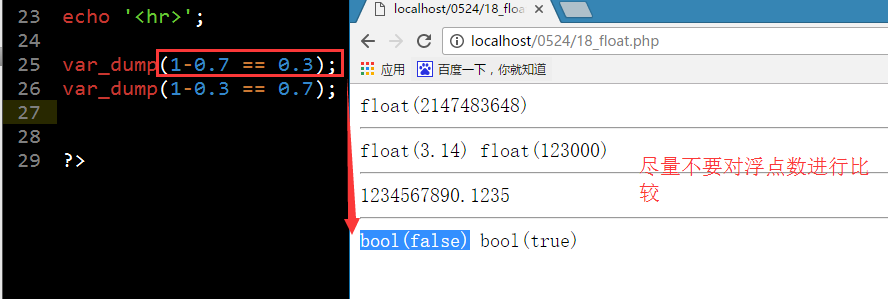
12345678901234567890 变成指数形式

1.2345678901234567890e19

## 浮点数的比较

尽量不要对浮点数进行比较.

因为浮点数的小数部分是 无限接近于真实值



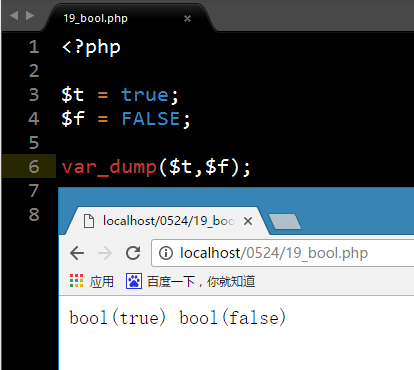
# 第19讲 布尔型数据

真 或者是假

true

false

不区分大小写.



以下哪些情况(值)会转换成bool中的false

整形的 0

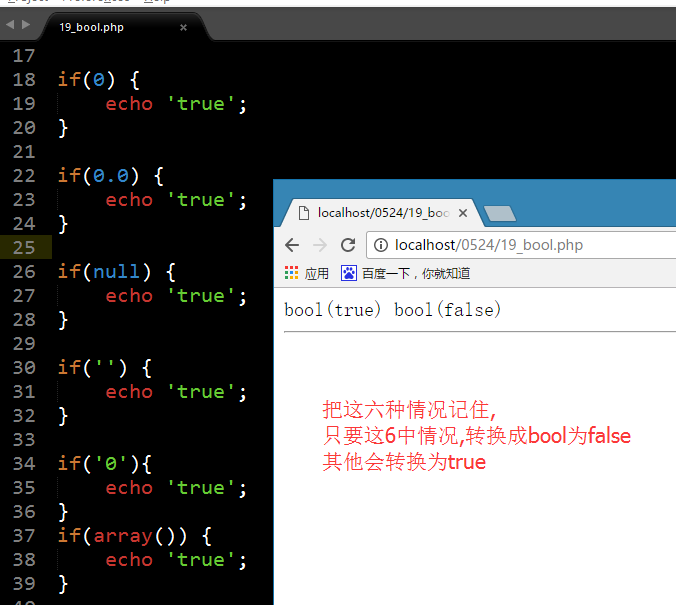
浮点型的 0.0

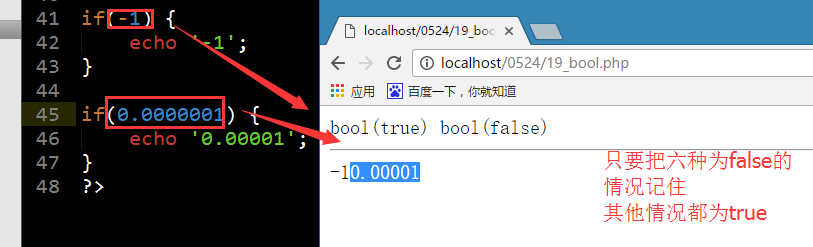
空 null

字符串的 空 ‘’

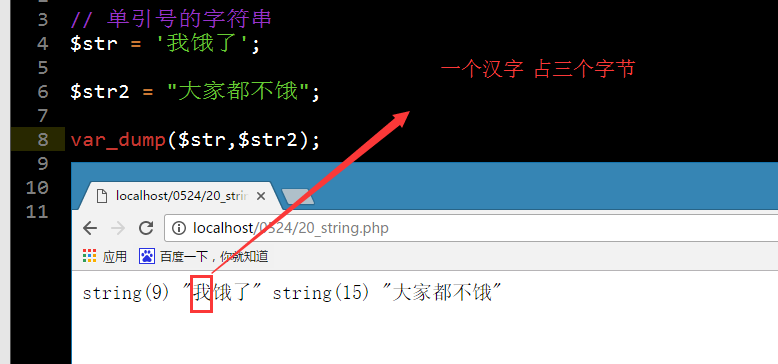
字符串的 0 ‘0’

数组的空 array()



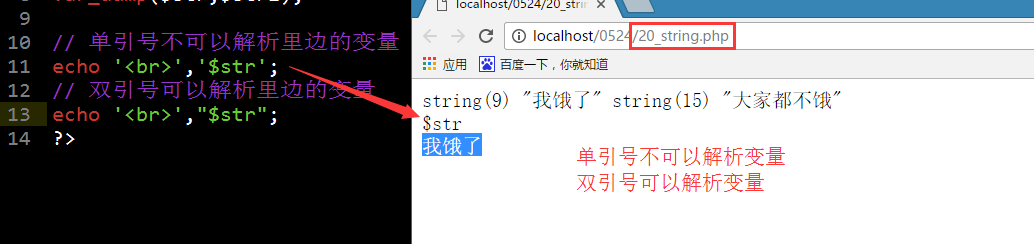


# 第20讲 字符串

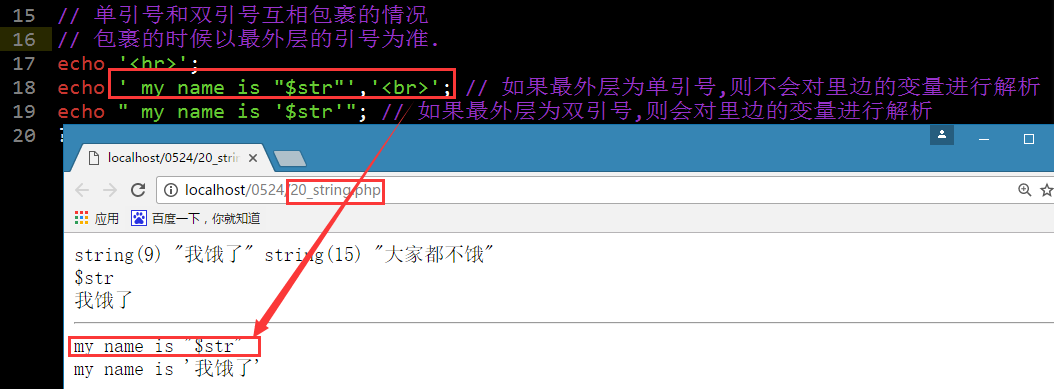


## 单引号

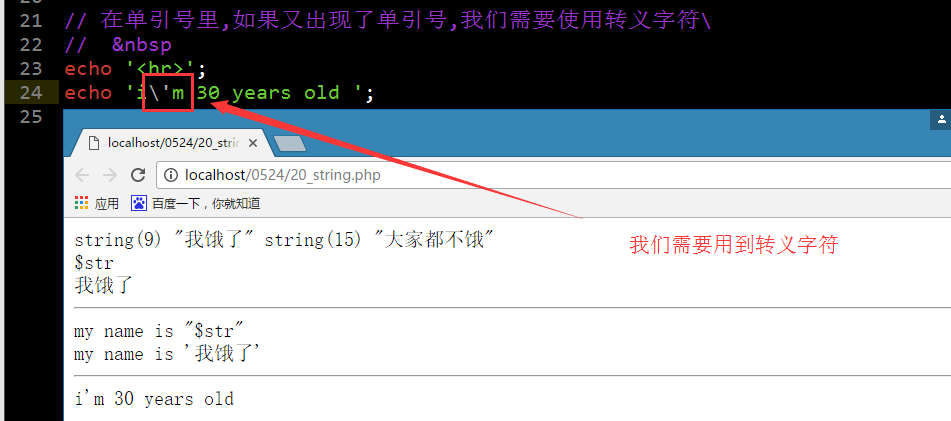
### 单引号不可以解析其中的变量！



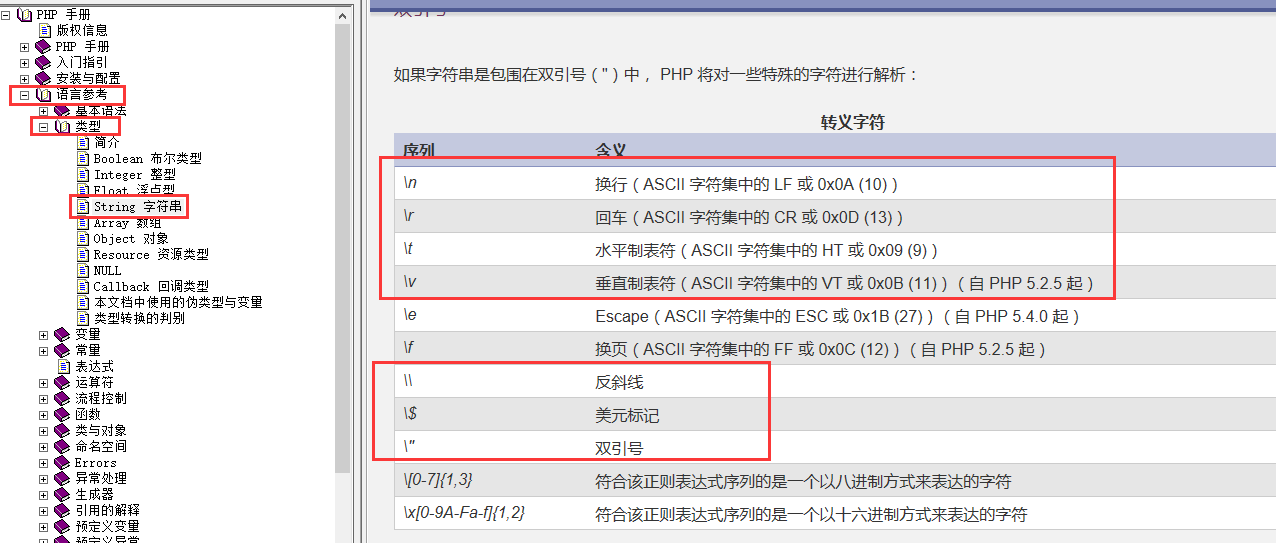
如果有单引号和双引号嵌套的时候,以最外层的引号为准!!!

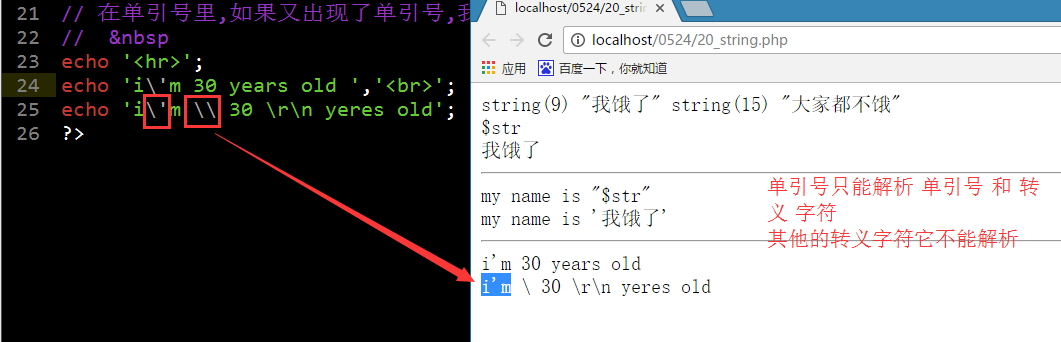


### 当使用单引号定义字符串时候，如果字符串里面又有单引号，需要用转义字符进行转义



### 单引号内除了可以转义单引号和转义字符外，就不能处理其他的转义字符了！



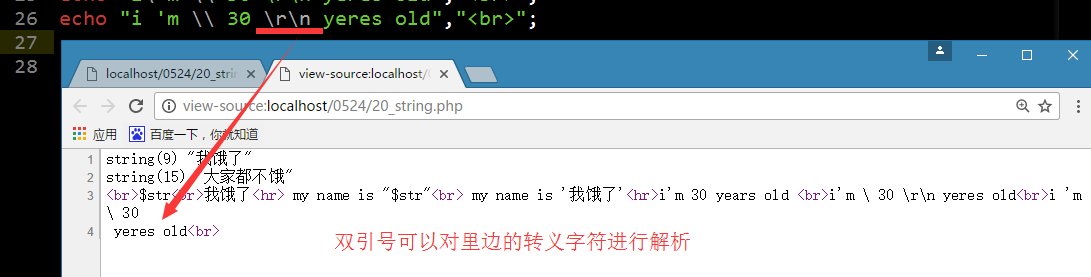


## 双引号

### 双引号可以解析其中的变量

### 当使用双引号定义字符串时候，如果字符串里面又有双引号，需要用转义字符进行转义

### 双引号内除了可以转义双引号和转义字符外，还能处理其他的转义字符！



### 双引号里面有变量名，最好先确定变量名的边界！

就是使用一对大花括号将变量名给包裹起来，此时，这里的花括号只是一种语法形式！



## 定界符(Heredoc –无单引号)

单引号及双引号，在多行字符串、复杂字符串的定义时显得不便

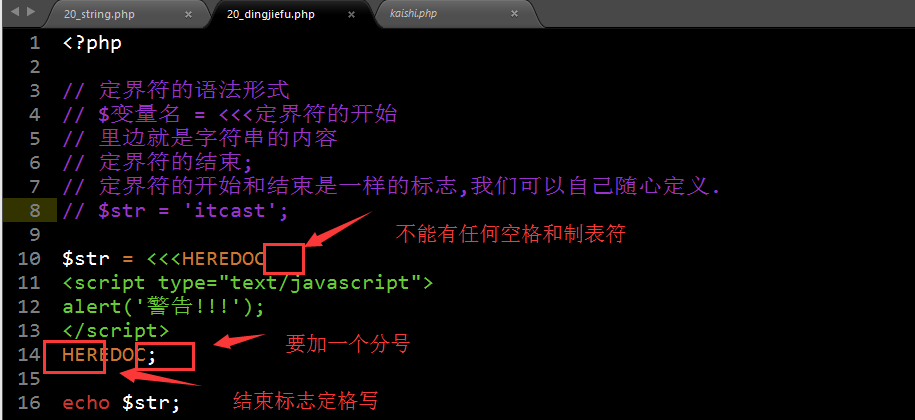
语法：

$str =<<<heredoc

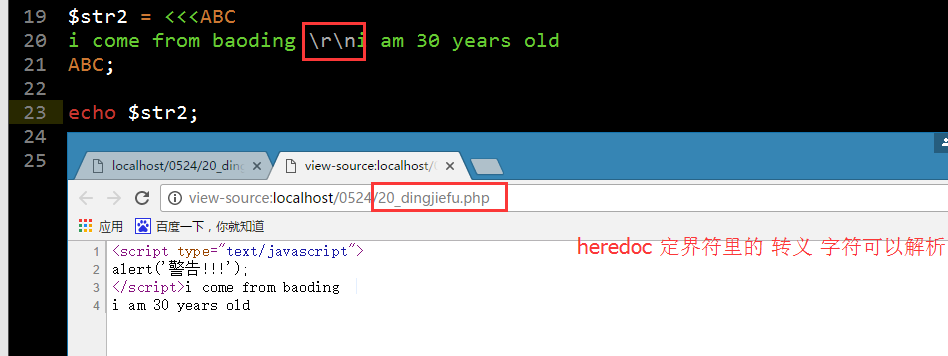
字符串内容

heredoc;

heredoc有三个注意的地方:



heredoc可以对里边的转义字符进行解析



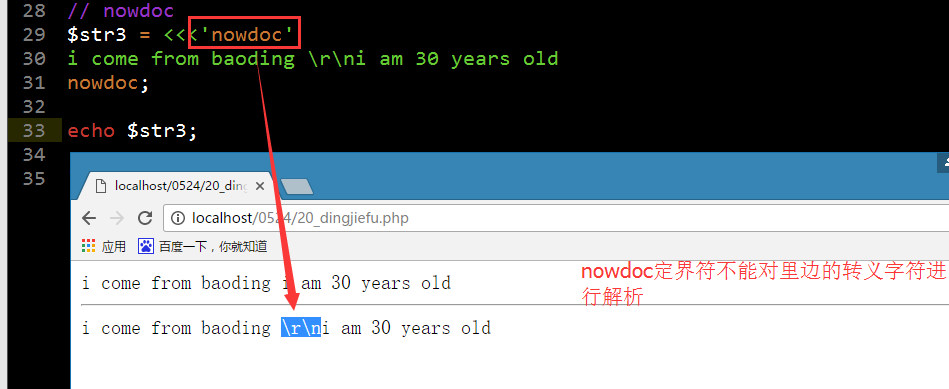
## 定界符(Nowdoc—有单引号)

语法形式:

$变量名 = <<<’定界符的开始’

内容

定界符的结束;



# 第21讲 特殊数据类型

## NULL型

null就是空,更多的是一种描述.

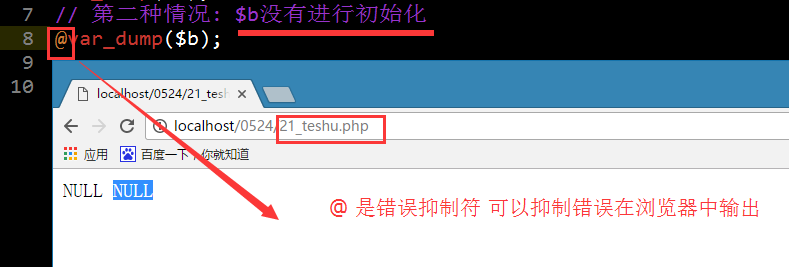
NULL不区分大小写

为null的几种情况

① 直接给一个变量赋值为null



② 我们打印未定义的变量或者是 被unset过的变量



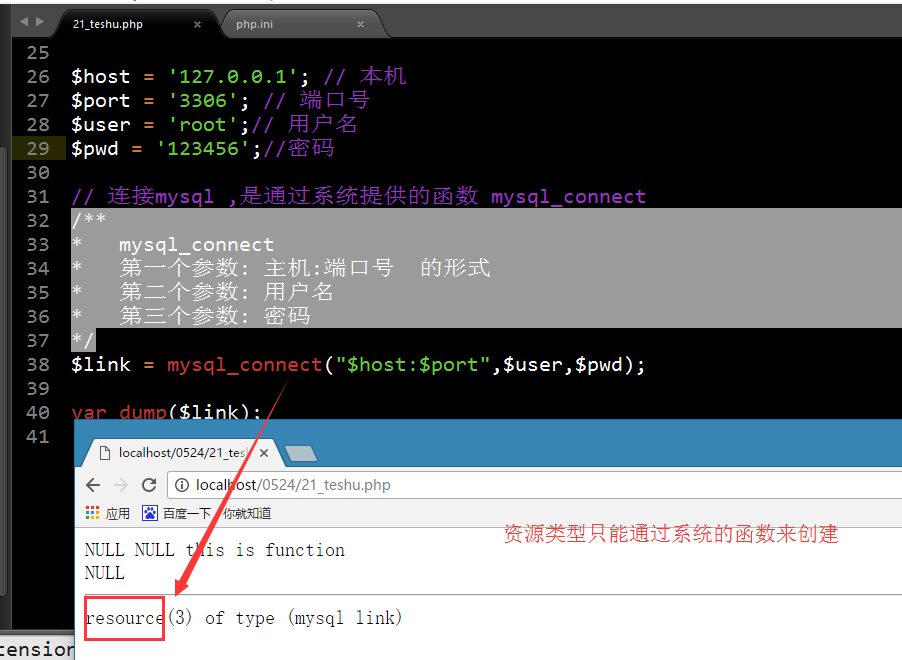
③ 如果我们打印一个没有返回值的函数也会出现null



## resource型

资源型,资源型只能通过系统的内置函数来创建,我们不能通过其他的途径来创建

我们php一般获取外界数据的时候会涉及到 资源型.例如 数据库



# 第22讲 类型转换

开发过程中会出现一些类型转换的情况

类型的转换可以分成两种：

一种是类型自动转换

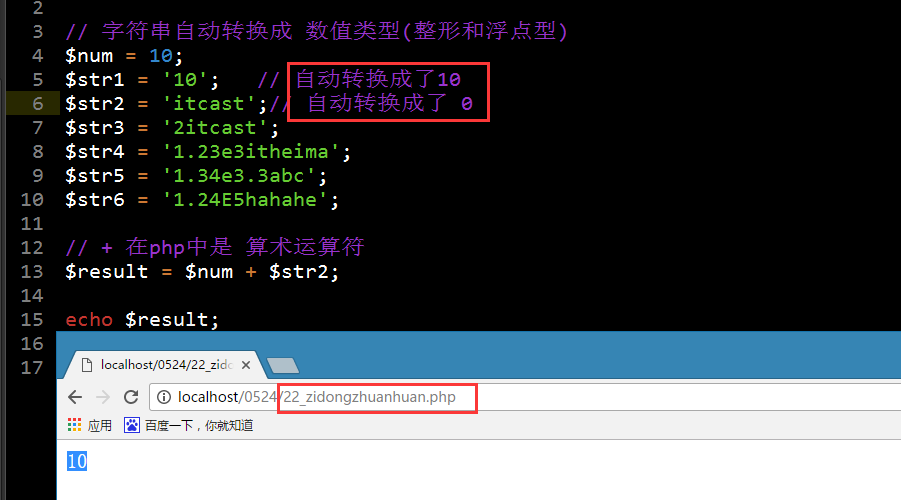
一种是类型的强制转换

## 类型自动转换

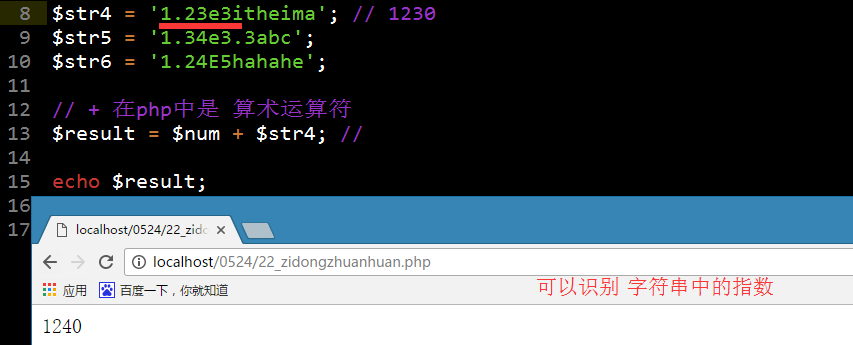
### 字符串类型自动转换成数值类型

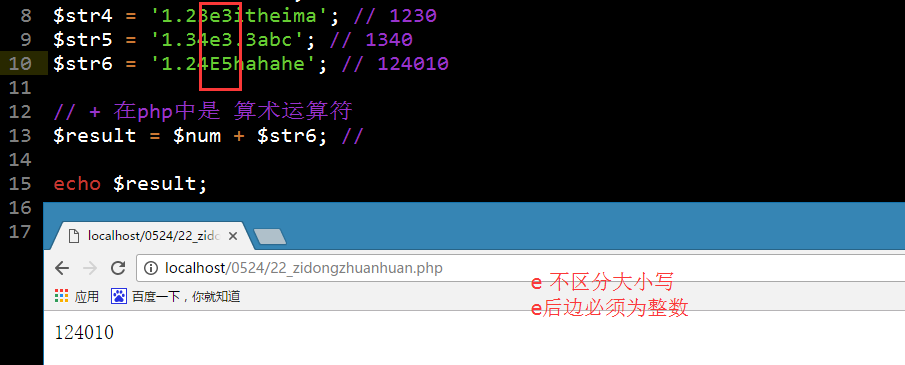
① 字符串如果是以数值开头的,会自动从开始获取到非数字为止

② 只要不是以数值开头,都取0



③ 自动转换中可以识别指数形式





### 其他类型自动转换成布尔型

0

0.0

‘0’

null

‘’

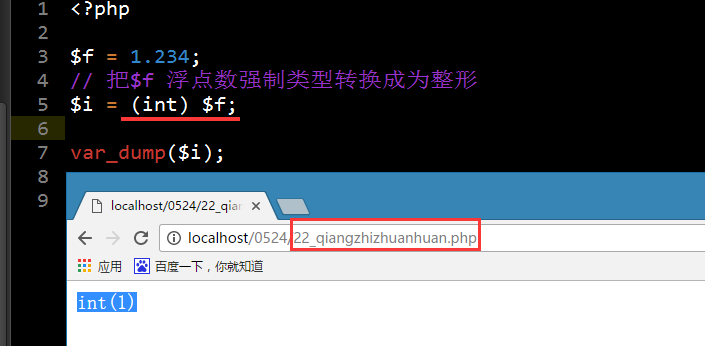
空数组

详见 布尔型数据

## 类型强制转换

语法形式:

$变量名 = (强制转换类型英文单词) 变量;



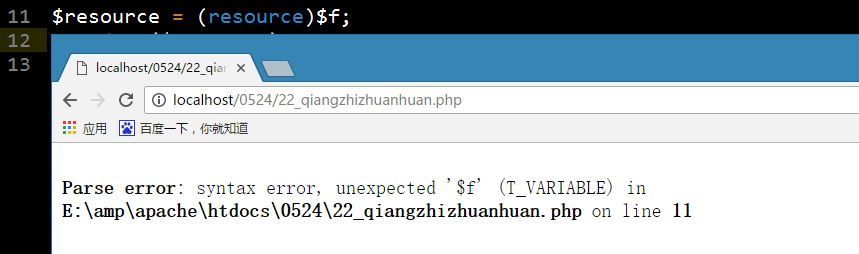
### 各种类型转换符

|  |  |
| --- | --- |
| 强制转换语法 | 强制转换成为的类型 |
| (int) | 整形 |
| (float)/(double) | 浮点数 |
| (string) | 字符串 |
| (bool) | 布尔型 |
| (array) | 数组型 |
| (object) | 对象类型 |

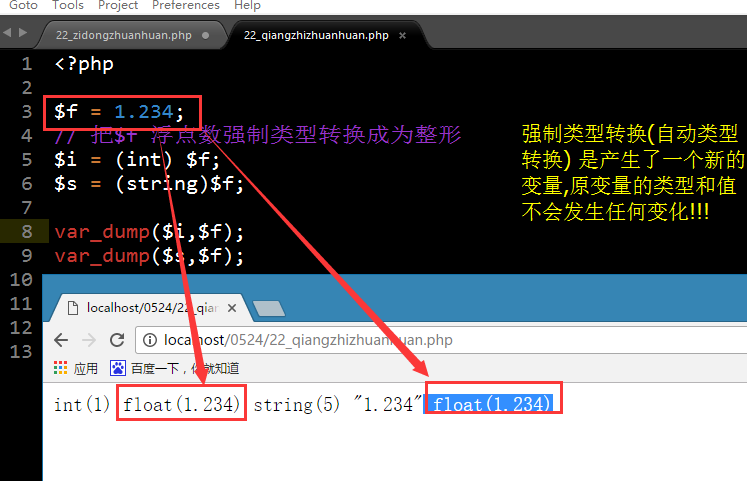
不能强制转换为 资源类型.

### 特别注意

* 无法把数据强制转换为资源类型



* 一个数据由A类型转换为B类型，无论是自动转换还是强制转换，转换后的结果是一样的！
* 类型转换的时候,原数据的类型和值不会发生改变!!!



# 第23讲 类型相关的函数

什么是函数？

函数可以看成是程序中独立的模块，可以完成独立的功能！

什么时候需要封装函数？

如果有一段代码在脚本中经常使用，就需要封装成函数，以实现代码的重用！

函数分成系统函数和用户自定义函数！

function 函数名(参数列表) {

函数体

}

学习系统函数的时候,尽量掌握 函数名的含义 以及查找手册

## gettype

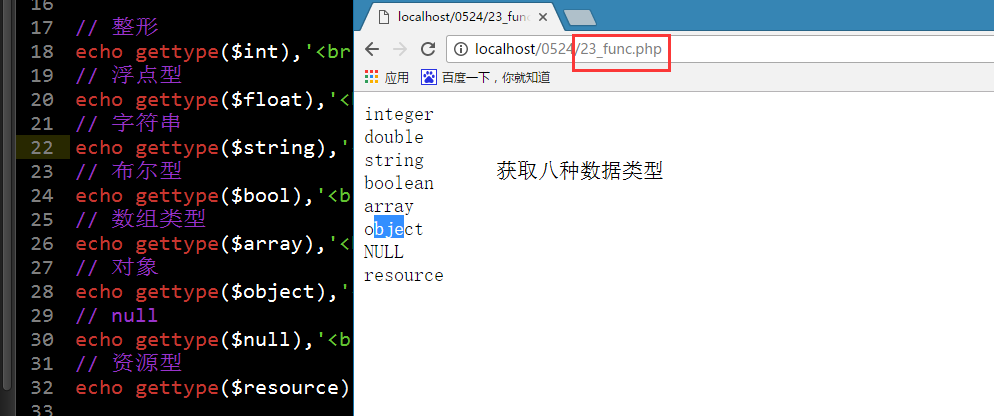
获取类型.

先分析函数名,带着目的去分析.这样可以加深印象

然后分析参数列表

最后分析返回值

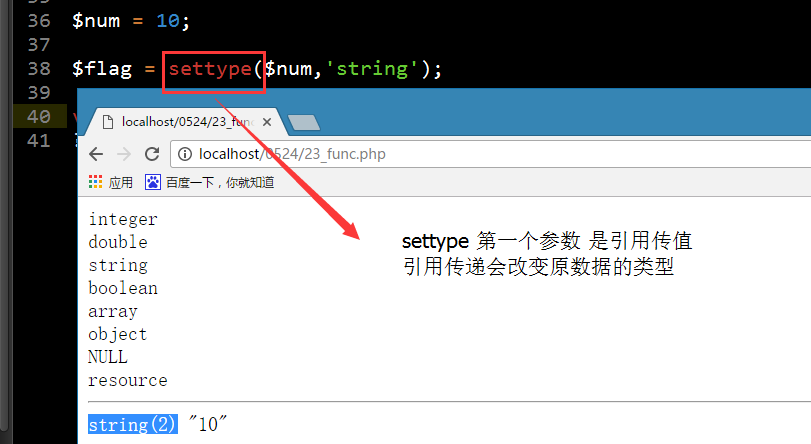




## settype

设置类型





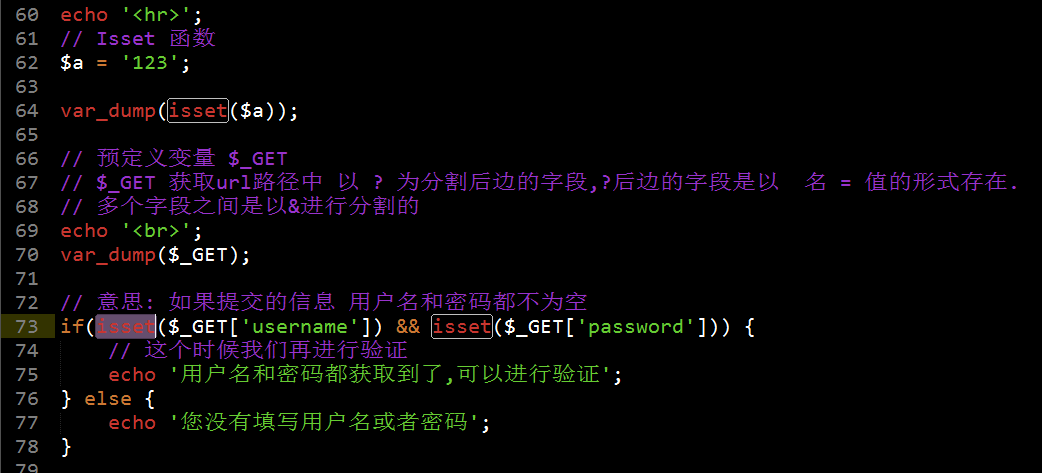
我们在填写settye的第一个参数的时候,变量名前边不需要添加 & 取地址符

## isset

isset 判断变量是否存在. 判断变量是否存在的依据是 这个变量的值不为null

变量的值不为null,则存在

变量的值为null,则不存在



## empty

空.就是判断变量是否为空.

null 是一种空的情况.

0

0.0

‘’

‘0’

array()

判断一个变量是否为空的技巧: 就是 这个变量转换为bool为false

## is\_type系列

### is\_int/is\_integer

判断一个变量是否为整型

### is\_float/is\_double

判断一个变量是否为浮点数

### is\_numeric

用于判断一个变量是否为数值型！等于is\_int+is\_float

### is\_string

判断一个变量是否为字符串

### is\_bool

判断一个变量是否为布尔值

### is\_scalar

等于is\_int + is\_float + is\_bool + is\_string

### is\_array

判断一个变量是否为数组

### is\_object

判断一个变量是否为对象

### is\_resource

判断一个变量是否为资源类型

### is\_null

判断一个变量是否为是否为空

# 第24讲 算数运算符

## 几个概念

### 什么是运算符

运算符连接操作数或者表达式

* 按照操作数的数目进行分类

①单目运算符：操作数只有1个。++,--

②双目运算符：操作数只有2个。+，-，\*,/

③三目运算符 ：又称条件运算符。? :

* 按照功能

算术运算符：+,-,\*,/,%（取余，取模）

赋值运算符：=，

自操作：++，--，+=，-=，\*=，/=,%=

比较运算符：>,<,===（全等于），==

逻辑运算符：&&，||，!

位运算符：&，|，~,^,<<,>>.

条件运算符：？ ：

字符串连接运算符：.

错误抑制符：@

### 运算符的优先级和结合性

1+2\*3

1+1+1+3

1+1+(1+3)

优先级: 当一个表达式中有很多运算符的时候，就要考虑运算符的先后运算顺序的问题

结合性: 如果一组运算符具有相同的优先级，就要考虑运算符的运算方向的问题了，如果是从左往右运算，就叫作左结合（比较多）， 如果是从右往左运算，就叫作右结合！



### 表达式和语句

1+2\*3

1+1+1+3

1+1+(1+3)

表达式: 用各种运算符和括号（）连接起来的式子就叫作表达式，一般来说，用XX运算符连接起来的式子就叫作XX表达式！

语句: 把表达式加上一个分号；就变成了一条语句！

本质区别:

表达式为了就是获取一个值

语句是 一条指令;

## 常见的算数运算符

+

-

\*

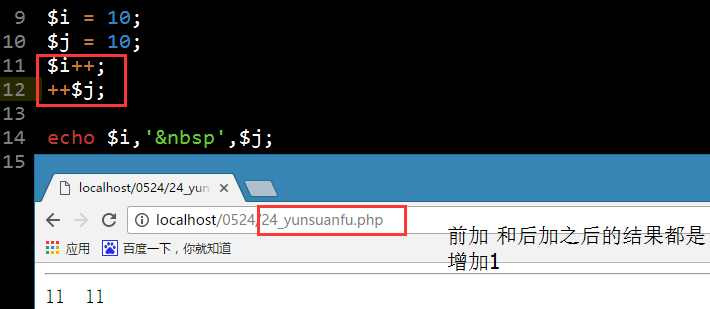
/ : 除法中 除数不能为 0

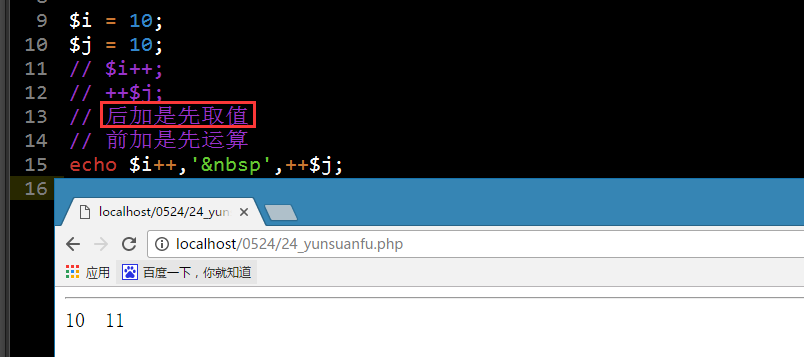
% : 取余 获取余数

## 自增自减运算符

i++,i—

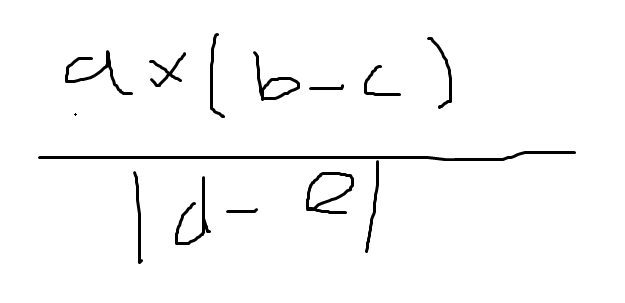
++I,--I;





## 算数表达式

我们可以通过系统提供的一些数学函数实现一些运算表达式



a\*(b-c)/abs(d-e);

abs是数学中的绝对值

# 第25讲 赋值运算符

赋值运算符就是 =

## 作用

$a = 100;

$b = $a;

$b = &$a;

赋值运算符的作用就是为了将 表达式右边的值|变量|地址 赋值给等号的左边

## 复合的赋值运算符

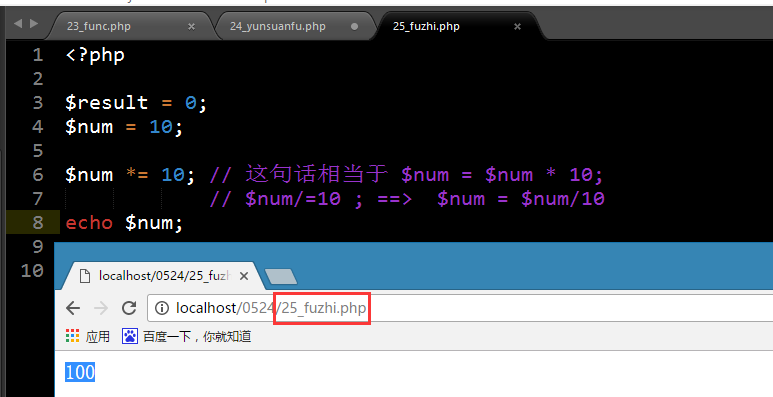
+=

-=

\*=

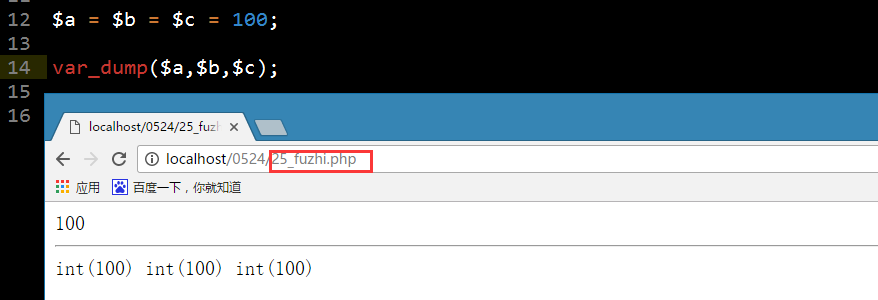
/=

%=



## 结合性

$a = $b = $c = 100;



从数学概念中来说,这三个数都是100.

但是 它们的结合性是从 右往左.

① 先去计算 $c = 100;

②再去计算 $b = ($c=100);

③最后 $a = ($b = ($c =100));

## 赋值表达式的值

赋值表达式的值就是等号右边的值!!!

# 第26讲 字符串运算符

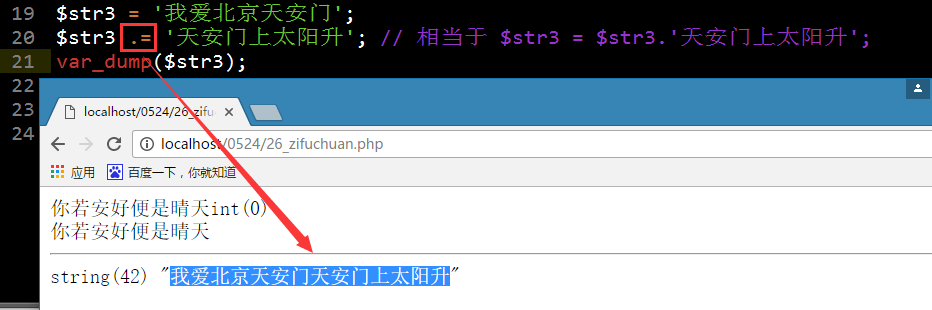
字符串运算符是 点(.)

## 作用

就是将字符串连接起来.

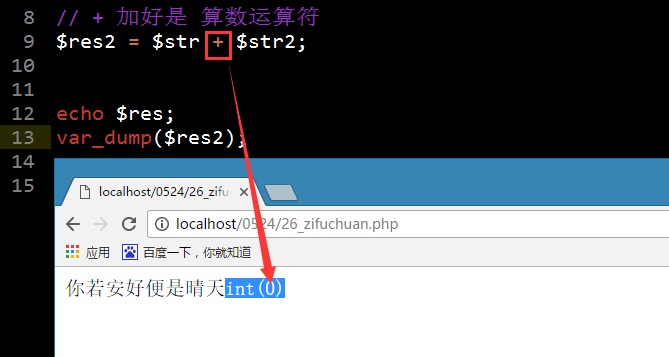


字符串运算符还有一个 复合运算符 .=

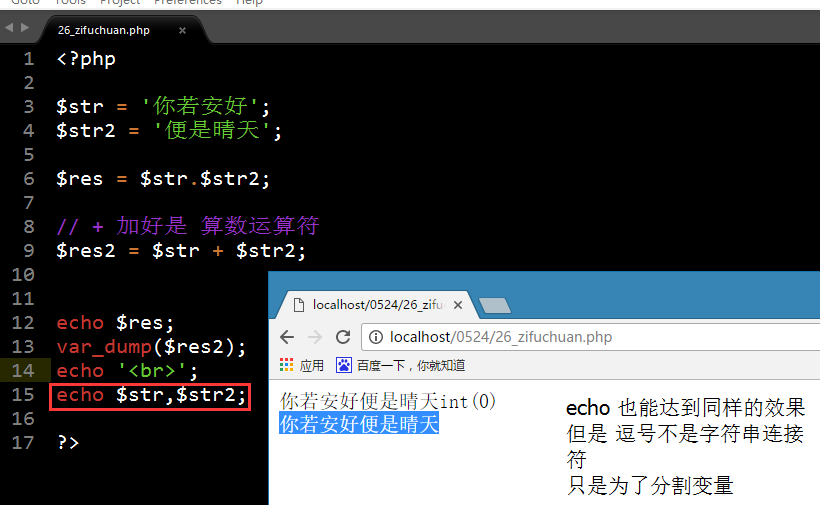


## 注意事项

① + 加好是 算数运算符



②我们在之前学习过echo ,echo也能达到拼接字符串的效果,但是 逗号只是变量的分隔符!!!



③ 如果 echo 实现的效果 和 字符串运算符一样,那么推荐大家使用 echo

因为echo消耗的资源小,性能快.

# 第27讲 关系运算符

## 概念

关系运算符其实就是比较运算符

比较的结果就2个,要么是真,要么是假

## 各种关系运算符

<

>

<=

>=

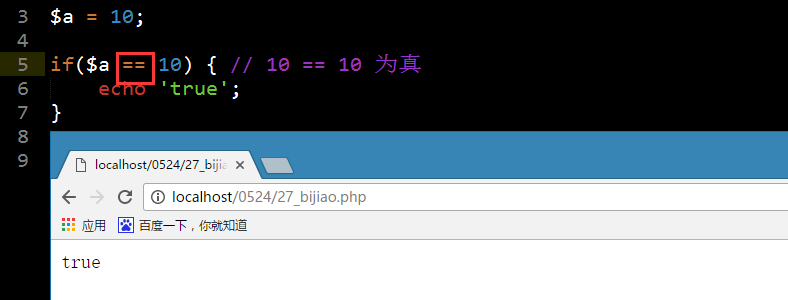
== 和 !=

=== 和 !==

## 注意事项

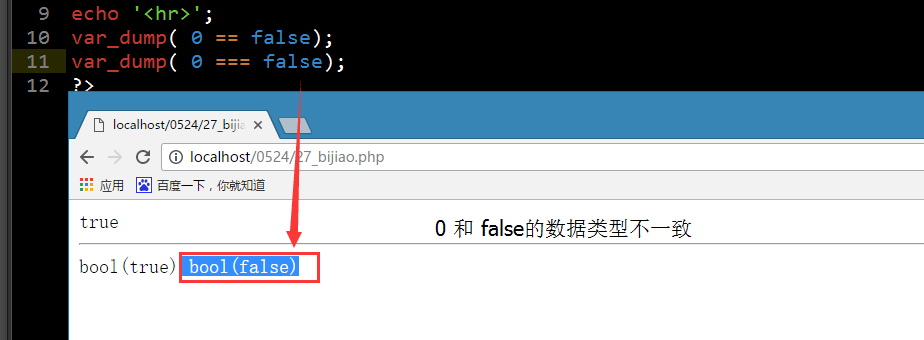
① = 和 == 的区别

= 是赋值运算符 == 是比较运算符



② == 和 ===

=== 是先进行类型的比较,再进行数值的比较



# 逻辑运算符

## 逻辑与

## 逻辑或

## 逻辑非

## 异或

## 短路运算

# 条件运算符

## 语法形式

## 注意事项

# 错误抑制符

## 错误抑制符

# 位运算符(了解)

## 什么是位运算

## 各种位运算符的含义

## 位运算符举例

## 位运算符在实际开发中的应用