

## 浮点类型

### 浮点数的 2 种表示形式

- 1, 常规写法: 带小数点。

```
$v1 = 123.456;
```

- 2, 科学计数法: 带一个特殊符号 “E”

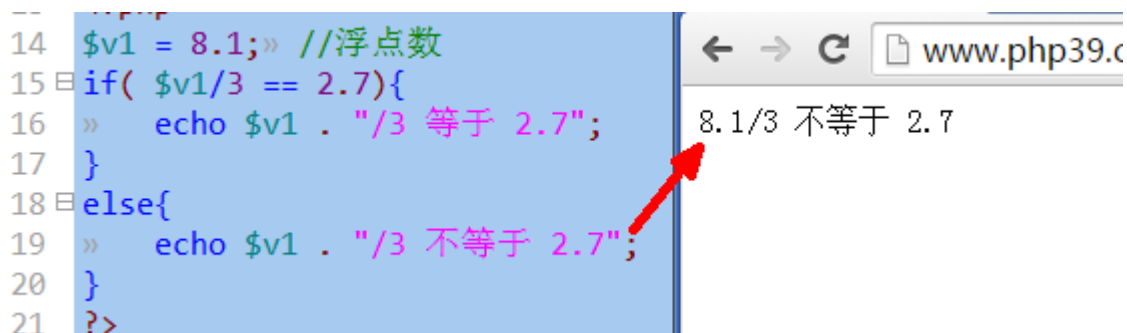
```
$v1 = 123.456E2; //含义为: 123.456 乘以 10 的 2 次方;
```

```
$v2 = 123.456E3; //含义为: 123.456 乘以 10 的 3 次方, 虽然结果是 123456, 但仍然是“浮点型”
```

```
$v3 = 123E4; //含义为: 123 乘以 10 的 4 次方, 还是浮点数。
```

### 浮点数使用的细节知识

- 浮点数不应进行大小比较



```
14 $v1 = 8.1; //浮点数
15 if( $v1/3 == 2.7){
16     echo $v1 . "/3 等于 2.7";
17 }
18 else{
19     echo $v1 . "/3 不等于 2.7";
20 }
21 ?>
```

8.1/3 不等于 2.7

因为:

- 1, 所有数字, 最终的表示形式, 都是 2 进制!!!
- 2, 大多数浮点数的 2 进制形式, 不能完全表达准确, 最终只能“以很高的精度接近理论值”
- 3, 因此, 浮点数比较是不可靠。

再从另一个角度证明浮点数的不准确性:

```
21 echo "<br />php中输出: 8.1/3的结果为: " . (8.1/3);
22 ?>
23 </body>
24 </html>
25 <script>
26     document.write("<br />但js中输出: 8.1/3的结果为: " + ( 8.1/3) );
27 </script>
```

php中输出：8.1/3的结果为：2.7  
但js中输出：8.1/3的结果为：2.6999999999999997

说明： php 中输出其实是做了一定的处理之后的显示结果，而 js 的输出是该计算结果的“真实反映”。

那应该怎么办？

考虑实际应用所需的精度的情况下，去将要比较的浮点数，转换为整数之后再比较。

比如：

要求精度为 3 为小数，则都乘以 1000，然后取整后比较；

要求精度为 4 为小数，则都乘以 10000，然后取整后比较；

....

```
22 echo "<hr />";
23 //以下为正确的浮点数比较方法:
24 //考虑精度要求为4位时:
25 if( round($v1/3 * 10000) == round(2.7 * 10000) ){
26     » echo $v1 . "/3 等于 2.7";
27 }
28 else{
29     » echo $v1 . "/3 不等于 2.7";
30 }
```




- 小数转二进制的做法：乘 2 并顺序取整数部分(了解)：

```

1 //目标：将小数0.6125转换为2进制小数形式：
2 //小数转二进制的做法：乘2并顺序取整数部分
3 0.6125» » 整数
4 *2
5 =0.225» » 1
6 *2
7 =0.45» » 0
8 *2
9 =0.9» » 0
10 *2
11 =0.8» » 1
12 *2
13 =0.6» » 1
14 *2
15 =0.2» » 1
16 *2
17 =0.4» » 0
18 *2
19 =0.8» » 0
20 *2
21 =0.6» » 1
22 .....循环了
23 可见其结果为：0.1001 1100 1100 1100 1100 ....（最多你就写32个了）
24


```



```

25 //再来一个，将小数0.625转换为2进制小数形式：
26 0.625» » 整数
27 *2
28 =0.25» » 1
29 *2
30 =0.5» » 0
31 *2
32 =0» » » 1
33 结果为：0.101
34

```



- 当整数运算的结果超出整数的范围后，会自动转换为浮点数（了解）。

获取一个数据（变量）的类型的函数有：

getType(\$变量);      返回的是该类型的名字（字符串）；  
 var\_dump(\$变量);      会输出该变量的类型，数据内容，（以及长度）；

```

33 $n1 = 10000;
34 $s1 = $n1 * $n1;» //两个整数相乘
35 $s2 = $n1 * $n1 * $n1;» //3个整数相乘
36 echo "<br />整数的最大值为: " . PHP_INT_MAX
37 echo "<br />"; var_dump($s1);
38 echo "<br />"; var_dump($s2);
20

```

---

整数的最大值为: 2147483647  
int(1000000000)  
float(1000000000000000)

---

## 字符串

有如下 4 形式:

形式 1: 双引号字符串:

\$str1 = "字符串内容.....";

形式 2: 单引号字符串:

\$str2 = '字符串内容.....';

形式 3: 双引号定界符字符串:

\$str3 = <<<"标识符 A"

字符串内容....

标识符 A;

形式 4: 单引号定界符字符串:

\$str4 = <<<'标识符 B'

字符串内容...

标识符 B;

## 双引号字符串

```

//双引号字符串:
//需要或可以识别的转义符有: \\» \" \n (换行符),\r(回车符),\t(tab符)
//还有一个: \$, 表示"$"符, 其实就是取消了其在双引号字符串中的"变量起始含义"
$str2 = "ab\"cde\nfg\tabcd $v1 \$v1 ";
echo $str2;

```

输出内容为:

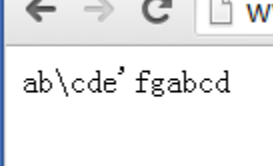
```

16 ab"cde
17 fg      abcd 123 $v1

```

## 单引号字符串：

```
14 //单引号字符串：
15 //实际需要或可以识别的转义符有：\\» \'
16 $str1 = 'ab\\cde\'fgabcd';
17 echo $str1;
```



说明：

实际上，单引号字符串中，只有最后一个“\”才是必须进行转义的。

## 双引号定界符字符串（heredoc）

```
33 //双引号定界符字符串：
34 //需要或可以识别的转义符有：\\» \n（换行符），\r（回车符），\t（tab符），\ $
35 //可见，该字符串的转义符，跟双引号非常类似！
36 //也可见，该字符串中的单引号双引号可以直接写出！
37 $str3 = <<<"ABCD"
38 abcdefg，这里其实就是字符串的内容！
39 ABCD;
40 //说明：
41 //1，上述标识符是我们自己取的一个类似常量名的"名字"，随便取
42 //2，该字符串的结束一行，只能出现该标识符本身及一个分号，其他什么都不可以出现！
43 //3，上述标识符的双引号可以省略（不推荐）|
44 echo $str3;
```

abcdefg，这里其实就是字符串的内容！

结果为：

## 单引号定界符字符串（nowdoc）：

```
49 //单引号定界符字符串：
50 //没有转义符！该字符串中的单引号双引号可以直接写出！
51 $str4 = <<<'ABCD'
52 这里是单引号字符串的内容！
53 ABCD;
54 //说明：
55 //1，上述标识符是我们自己取的一个类似常量名的"名字"，随便取
56 //2，该字符串的结束一行，只能出现该标识符本身及一个分号，其他什么都不可以出现！
57 echo $str4;
```

这里是单引号字符串的内容！

结果为：

## 布尔类型

单词是 bool, boolean。

其只有 2 个数据: true, false;

布尔类型的一个常见应用情形是: 对一个变量直接进行判断, 比如 if 判断, 示例如下:

```
14 $v1 = 123;  
15 if($v1){ //这就是对一个变量直接进行判断的语法! 可见123当做“真”!  
16     echo "<br />可见{$v1}当做“真”! ";  
17 }  
18 else{  
19     echo "<br />可见{$v1}当做“假”! ";  
20 }
```

这里的判断, 永远是指: 判断该变量 (数据) “是否为真”。

对于这种情况, 只有如下数据是被当做 “假” (false):

0, 0.0, “”, “0”, null, array(), false, 还有一个是 “未定义的变量”

其余都是真。

参考:

手册》附录》PHP 类型比较表》

使用 PHP 函数对变量 \$x 进行比较					
表达式	<a href="#">gettype()</a>	<a href="#">empty()</a>	<a href="#">is_null()</a>	<a href="#">isset()</a>	<a href="#">boolean : if(\$x)</a>
<code>\$x = "";</code> 还	<a href="#">string</a>	TRUE	FALSE	TRUE	FALSE
<code>\$x = null;</code> 有	<a href="#">NULL</a>	TRUE	TRUE	FALSE	FALSE
<code>var \$x;</code> 很多	<a href="#">NULL</a>	TRUE	TRUE	FALSE	FALSE
<code>\$x is undefined</code>	<a href="#">NULL</a>	TRUE	TRUE	FALSE	FALSE
<code>\$x = array();</code>	<a href="#">array</a>	TRUE	FALSE	TRUE	FALSE
<code>\$x = false;</code>	<a href="#">boolean</a>	TRUE	FALSE	TRUE	FALSE
<code>\$x = true;</code>	<a href="#">boolean</a>	FALSE	FALSE	TRUE	TRUE

## 类型转换

### 自动转换:

在任何运算中, 如果需要某种类型的数据, 而给出的数据不是该类型, 通常都会发生自动转换: 将该类型转换为目标需要的类型。

比如： `octdec($x), bindec($x), hexdec($x);` //这里就要求\$x 必须是字符串，如果不是，就会转换；  
`$v1 = 1 + "2";` //此时也发生了自动转换。

```
--
14 //这里演示的就是一些常见的自动转换
15 $v1 = 1 + 2;» //3
16 $v2 = 1 + "2";» //3;
17 $v3 = "1" + 2;» //3
18 $v4 = "1" + "2";» //3
19 //在php中，算术运算符，就只是对数值进行计算
20 $v5 = 1 + "2abc";» //3
21 $v6 = "1" + "2abc";»//3
22 $v7 = "1def" + "2abc";» //3
23 $v7 = "1def" + "abc2";» //1
24 $v7 = "def1" + "abc2";» //0
25 //。。。可见，这种识别字符串中数字的转换规则
26 //是，只去“识别”字符串的前面数字部分
27 $v8 = 1 + true;»» //2;
28 $v8 = 1 + false;» //1;
29
30 //以上运算中，也适用于-, *, / %号！
31
```

## 强制转换：

自动类型转换是由“运算符”或类似运算符的语句来决定的。

而：

强制类型转换，仅仅是一个简单的语法：

形式：（目标类型）数据；

含义： 将该数据转换为设定的目标类型；

例子：

```
14 $v1 = 123;
15 $s1 = (float) $v1; //将v1的值转换为float类型
16 » » » » //注意，此时$v1里面还是整数123
17 $s2 = (string) $v1; //将v1的值转换为string类型
18 echo "<br />s1为"; var_dump($s1);
19 echo "<br />s2为"; var_dump($s2);
```

通常的转换目标类型有：

(int), (float), (string), (bool), (array), (object)

上述强制类型转换，并不改变该变量的本身数据或类型。

对应，有一个语法是直接改变改变本的数据（及类型）：

`settype( $变量名, “目标类型”);`

```

21 $v2 = 123;
22 setType($v2, "string");//直接转换v2的类型为string
23 echo "<br />v2为"; var_dump($v2);

```

v2为string(3) "123"

## 类型相关的函数

- `var_dump()`：用于输出变量的“完整信息”，几乎只用于调试代码。
- `getType($变量名)`：获取该变量的类型名字，返回的是一个表示该类型名字的字符串，比如：“string”，“bool”，“double”，“int”
- `setType($变量名, “目标类型”)`：将该变量强制改变为目标类型；
- `isset()`, `empty()`, `unset()`；。。。省略！
- `is_XX 类型()` 系列函数：判断某个数据是否为某种类型，有如下一些：
  - `is_int($x)`； 判断\$x 是否是一个整数类型；
  - `is_float($x)`;
  - `is_string($x)`;
  - `is_bool($x)`;
  - `is_array($x)`;
  - `is_object($x)`;
  - `is_null($x)`;
  - `is_numeric($x)`； 判断\$x 是否是一个数字！
  - `is_scalar($x)`； 判断\$x 是否是一个“标量类型”