# 浮点类型

## 浮点熟的 2 种形式

1，常规写法：带小数点。

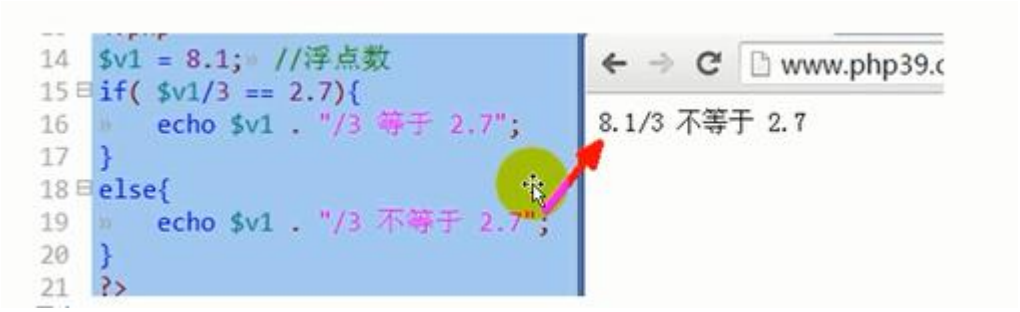
```
$v1 = 123.456;
```

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。
```

# 浮点数字使用的细节知识

## 浮点数不应该进行大小比较




因为，

1，所有数字，最终的表示形式，都是 2 进制

2，浮点数的 2 进制形式，不能完全表达准确，”以很高的精度接近理论值”

3，因此，浮点数比较是不可靠的

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”

**小数转二进制的做法：乘 2 并顺序取整数部分**

**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**

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);
```

```
整数的最大值为：2147483647
int(100000000)
float(1000000000000)
```

# 字符串

## 有如下四种形式

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;
```

单引号字符串：

```
//单引号字符串：
//其实需要或可以识别的转义符有：\\» \'
$str1 = 'ab\\cde\'fgabcd';
echo $str1;
```

ab\cde'fgabcd

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

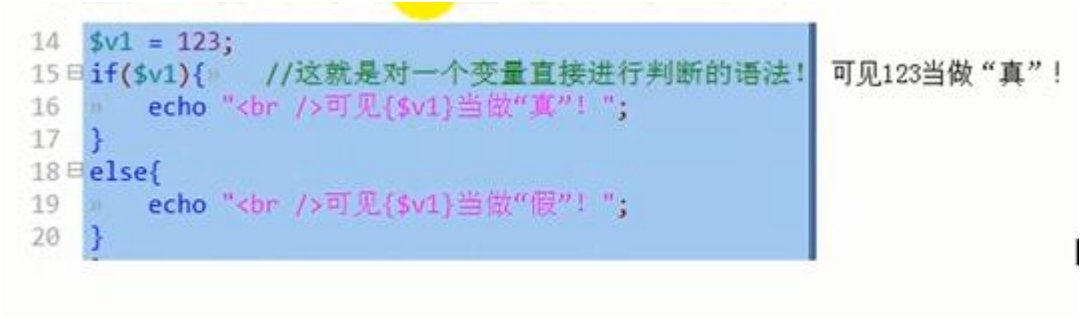
```
//双引号字符串：
//需要或可以识别的转义符有：\\» \"» \n（换行符）,\r(回车符，\t(tab符)
//还有一个： \$，表示“$”符，其实就是取消了其在双引号字符串中的“变量起始含义”
$str2 = "ab\"cde\nfg\tabcd $v1 \$v1 ";
echo $str2;
```

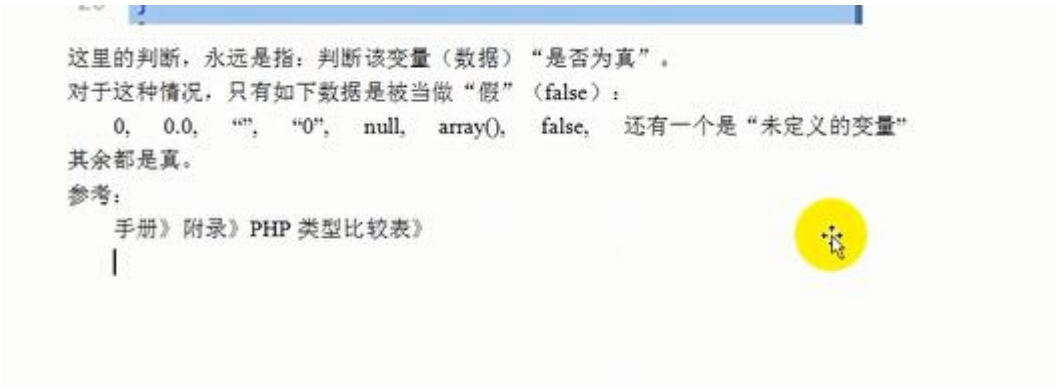
输出内容为：

```
ab"cde
fg      abcd 123 $v1
```

# 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情形是：对一个变量直接进行判断
例如：



# 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。

```php
//这里演示的就是一些常见的自动转换
$v1 = 1 + 2;    //3
$v2 = 1 + "2"; //3;
$v3 = "1" + 2; //3
$v4 = "1" + "2";    //3
//在php中，算术运算符，就只是对数值进行计算
$v5 = 1 + "2abc";   //3
$v6 = "1" + "2abc";//3
$v7 = "1def" + "2abc"; //3
$v7 = "1def" + "abc2"; //1
$v7 = "def1" + "abc2"; //0
//。。。可见，这种识别字符串中数字的转换规则
//是，只去“识别”字符串的前面数字部分
$v8 = 1 + true;    //2;
$v8 = 1 + false;    //1;

//以上运算中，也适用于-， *， / %号！
```

## 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：

```php
$v1 = 123;
$s1 = (float) $v1;//将v1的值转换为float类型
    //注意，此时$v1里面还是整数123
$s2 = (string)$v1;//将v1的值转换为string类型
echo "<br />s1为";  var_dump($s1);
echo "<br />s2为";  var_dump($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($x);
  - is_numeric($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”