1.15

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

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：

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；

....

小数转二进制的做法：乘 2 并顺序取整数部分(了解)：

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）；

单引号字符串：

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。

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：

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》

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。

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；

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”)；

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”