# =======Day02 PHP基础=======

# PHP语法初步

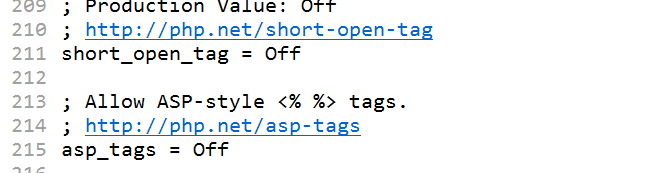
PHP是一种运行在服务器端的脚本语言，可以嵌入到HTML中。

## PHP代码标记

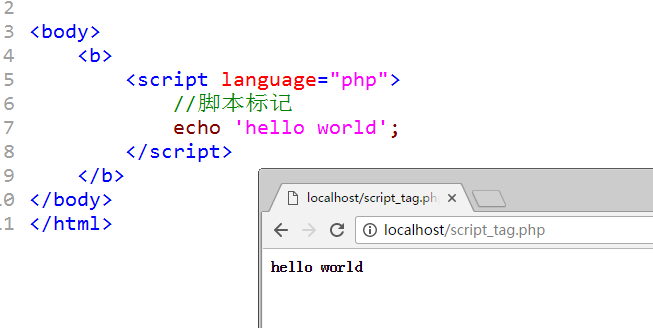
在PHP历史发展中，可以使用多种标记来区分PHP脚本

ASP标记：<% php代码 %>

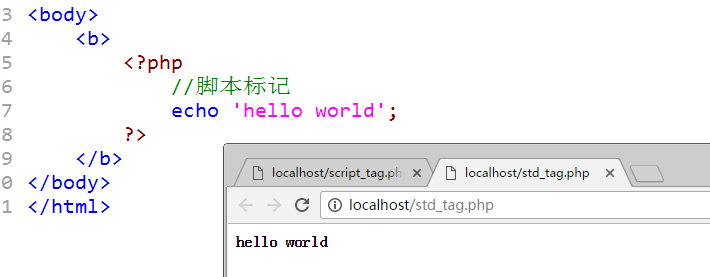
短标记：<? php代码 ?>，以上两种基本弃用，如果要使用那么需要在配置文件中开启



脚本标记：<script language=”php”>php代码</script> (php7已经弃用)



标准标记（常用）：<?php php代码?>



## PHP注释

习惯：所有的代码在写的过程中都必须进行注释，对于初学者而言，注释就是个人学习和写代码的一个思路说明

PHP中注释分为两种：行注释和块注释

行注释：一次注释一行

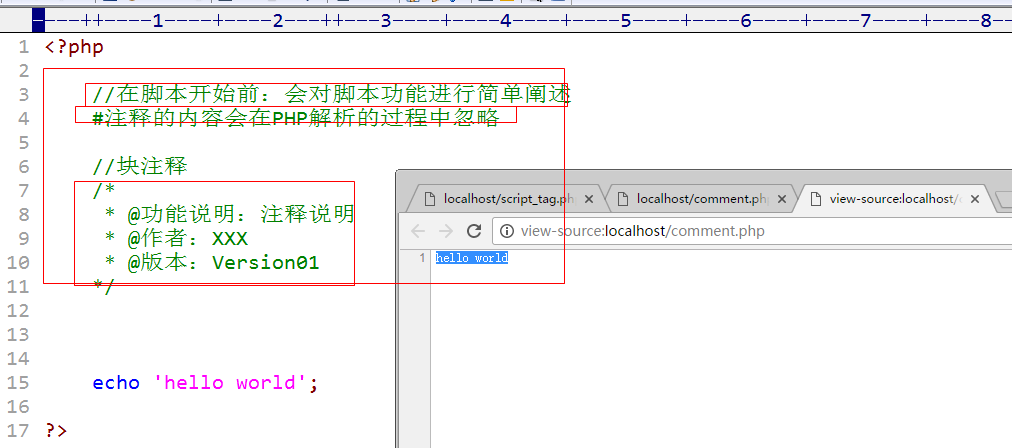
//：后面跟的所有内容都是注释

#：与//一样

块注释：一次注释多行

/\*：中间直到\*/出现之前，全部都是注释

\*/



## PHP语句分隔符

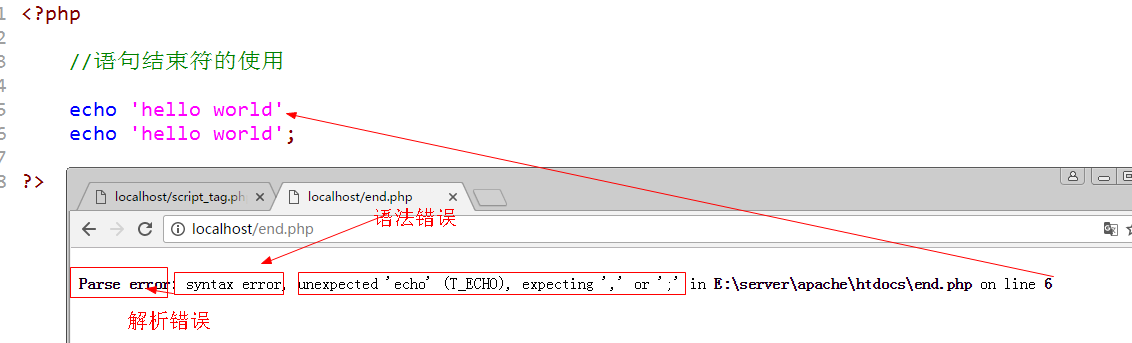
语句分隔符：在PHP中，代码是以行为单位，系统需要通过判断行的结束，该结束通常都是一个符号：分号“;”（英文状态下的分号）

定义

定义内容：

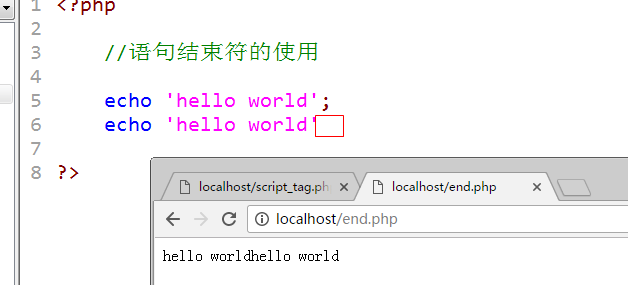
$a = 5;

Echo ‘hello world’;

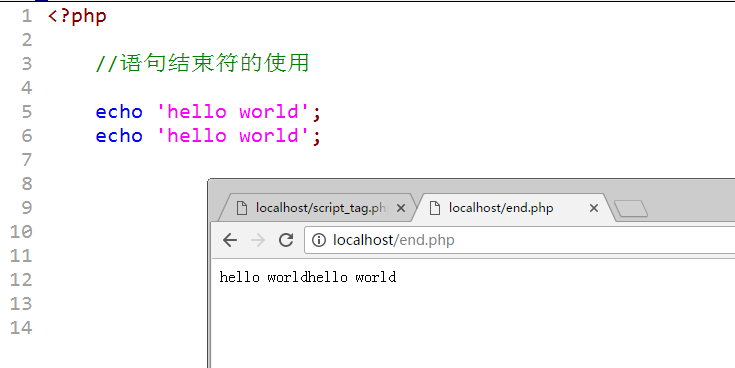


特殊说明：

1、 PHP中标记结束符?>有自带语句结束符的效果，最后一行PHP代码可以没有语句结束符“;”



2、 PHP中其实很多代码的书写并不是嵌入到HTML中，而是单独存在，通常书写习惯中就不建议使用标记结束符?>，PHP会自动从开始到最后全部认为是PHP代码，从而解析



# 变量

PHP是一种动态网站开发的脚本语言，动态语言特点是交互性，会有数据的传递，而PHP作为“中间人”，需要进行数据的传递，传递的前提就是PHP能自己存储数据（临时存储）

## 变量基本概念

变量来源于数学，是计算机语言中能储存计算结果或能表示值抽象概念。变量可以通过变量名访问。在指令式语言中，变量通常是可变的。

1、 变量是用来存储数据的；

2、 变量是存在名字的；

3、 变量是通过名字来访问的：数据；

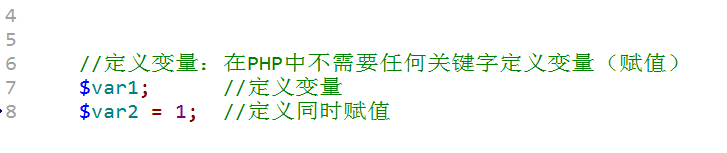
4、 变量是可以改变的：数据。

## 变量的使用

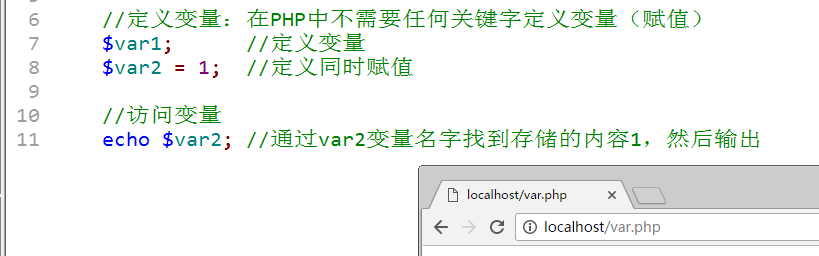
PHP中的所有变量都必须使用“$”符号

1、 定义：在系统中增加对应的变量名字（内存）

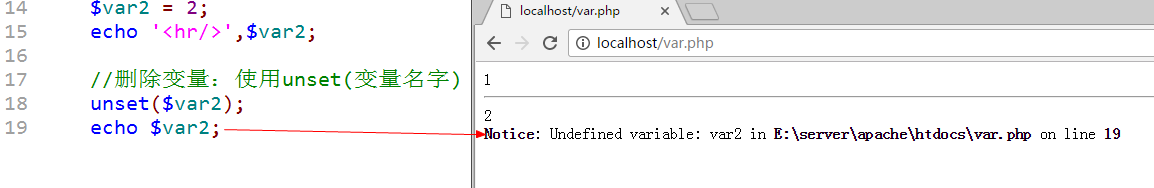
2、 赋值：可以将数据赋值给变量名（可以在定义的同时完成）



3、 可以通过变量名访问存储的数据



4、 可以将变量从内存中删除



## 变量命名规则

1、 在PHP中变量名字必须以“$”符号开始；

2、 名字由字母、数字和下划线“\_”构成，但是不能以数字开头；

3、 在PHP中本身还允许中文变量（不建议）。



## 预定义变量

预定义变量：提前定义的变量，系统定义的变量，存储许多需要用到的数据（预定义变量都是数组）

$\_GET：获取所有表单以get方式提交的数据

$\_POST：POST提交的数据都会保存在此

$\_REQUEST：GET和POST提交的都会保存

$GLOBALS：PHP中所有的全局变量

$\_SERVER：服务器信息

$\_SESSION：session会话数据

$\_COOKIE：cookie会话数据

$\_ENV：环境信息

$\_FILES：用户上传的文件信息

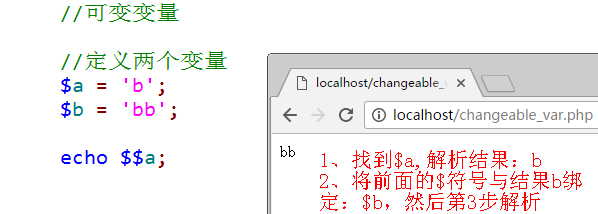
## 可变变量

可变变量：如果一个变量保存的值刚好是另外一个变量的名字，那么可以直接通过访问一个变量得到另外一个变量的值：在变量前面再多加一个$符号。

$a = ‘b’;

$b = ‘bb’;

$$a🡺bb

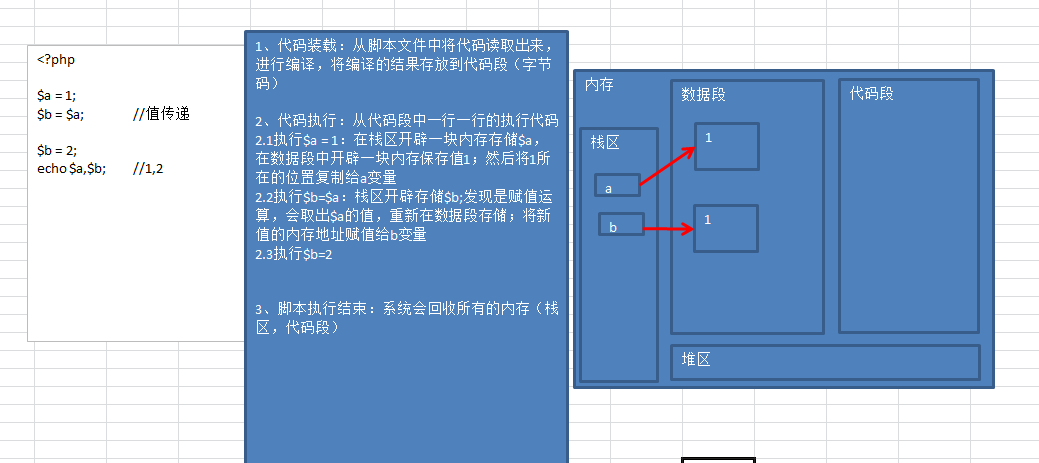


## 变量传值

将一个变量赋值给另外一个变量：变量传值

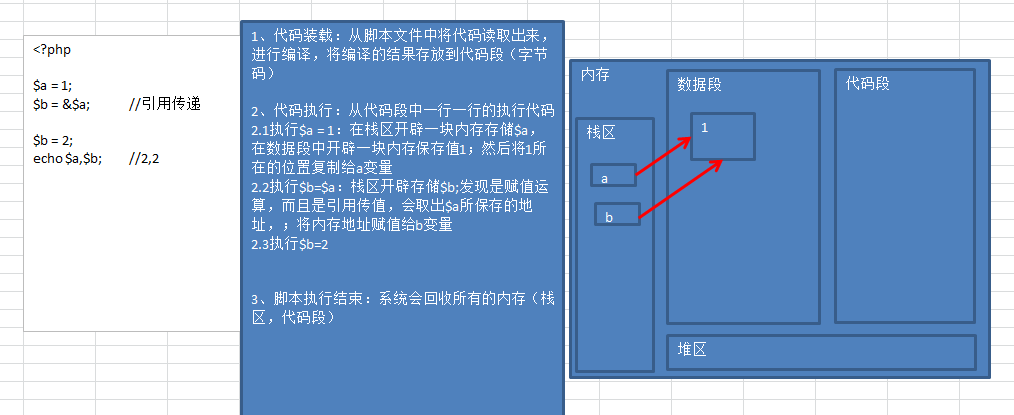
变量传值一共有两种方式：值传递，引用传递

值传递：将变量保存的值赋值一份，然后将新的值给另外一个变量保存（两个变量没有关系）



引用传递：将变量保存的值所在的内存地址，传递给另外一个变量：两个变量指向同一块内存空间（两个变量是同一个值）

$新变量 = &$老变量; (&地址符号)



在内存中，通常有以下几个分区

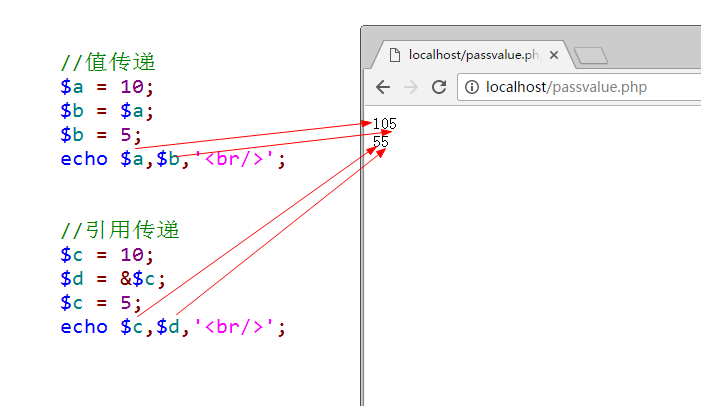
栈区：程序可以操作的内存部分（不存数据，运行程序代码），少但是快

代码段：存储程序的内存部分（只存储代码不执行代码）

数据段：存储普通数据（全局区和静态区）

堆区：存储复杂数据，大但是效率低

代码实现：



# 常量

常量与变量一样，都是用来保存数据的。

## 常量基本概念

常量：const/constant，是一种在程序运行当中，不可改变的量（数据）

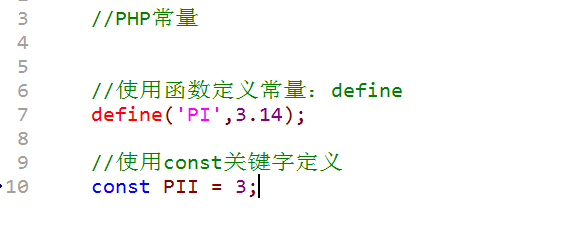
常量一旦定义，通常数据不可改变（用户级别）

## 常量定义形式

在PHP中常量有两种定义方式（5.3之后才有两种）

1、 使用定义常量的函数：define(‘常量名’,常量值);

2、 5.3之后才有的：const 常量名 = 值;



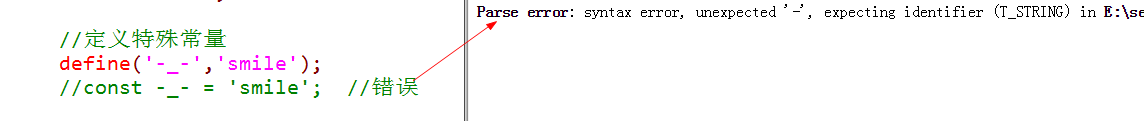
常量名字的命名规则

1、 常量不需要使用“$”符号，一旦使用系统就会认为是变量；

2、 常量的名字组成由字母、数字和下划线组成，不能以数字开头；

3、 常量的名字通常是以大写字母为主（与变量以示区别）；

4、 (第二点不完全对：)常量命名的规则比变量要松散，可以使用一些特殊字符，该方式只能使用define定义；



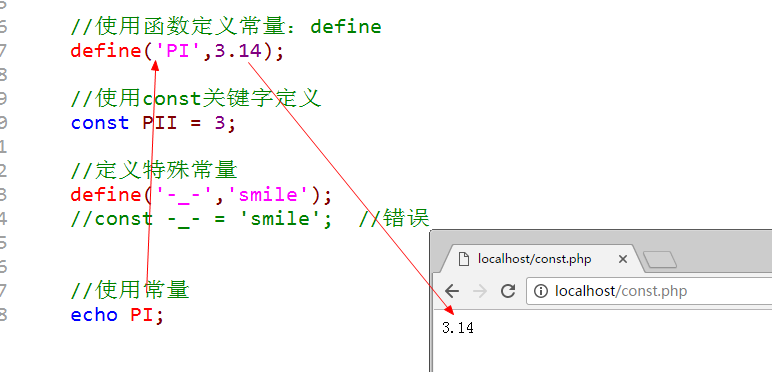
注意细节：

1、 Define和const定义的常量是有区别：在于访问权限区别

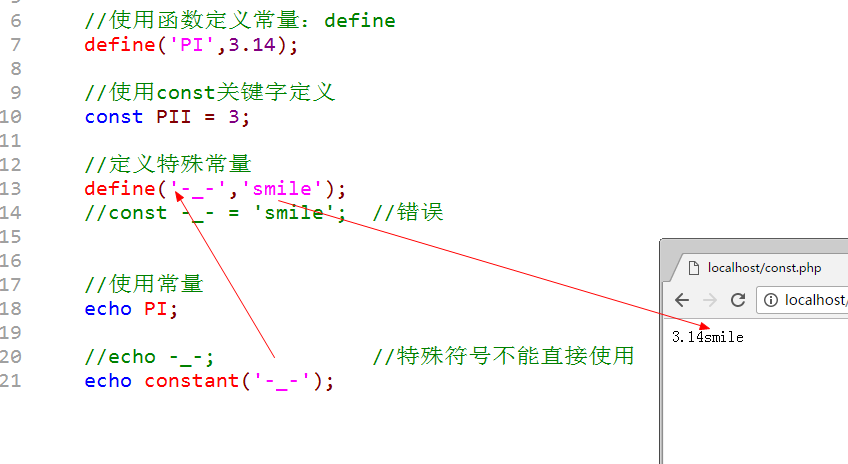
2、 定义常量通常不区分大小写，但是可以区分，可以参照define函数的第三个参数

## 常量使用形式

常量的使用与变量一样：不可改变值（在定义的时候必须赋值）



有的时候还需要使用另外一种形式来访问（针对的是特殊名字的常量），需要用到另外一个访问常量的函数：constant(‘常量名’)



说明：常量和变量的使用

1、 凡是数据会可能变化的，那么肯定是用变量

2、 数据不一定会变的，可以使用常量或者变量（变量居多）

3、 数据不允许被修改的，一定用常量

## 系统常量

系统常量：系统帮助用户定义的常量，用户可以直接使用

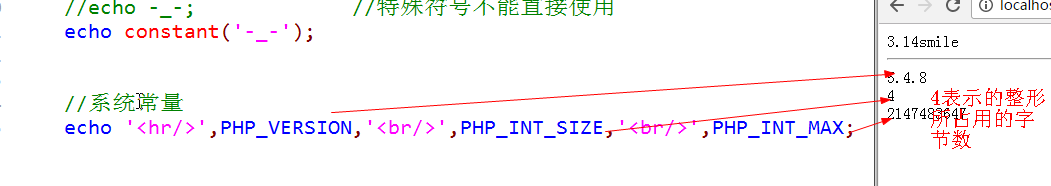
常用的几个系统常量

PHP\_VERSION：PHP版本号

PHP\_INT\_SIZE：整型大小。(1字节8位。32位4字节，64位8字节)

PHP\_INT\_MAX：整型能表示的最大值（PHP中整型是允许出现负数：带符号）

[32位和64位的size&max](https://blog.csdn.net/qq_34765852/article/details/51688869)

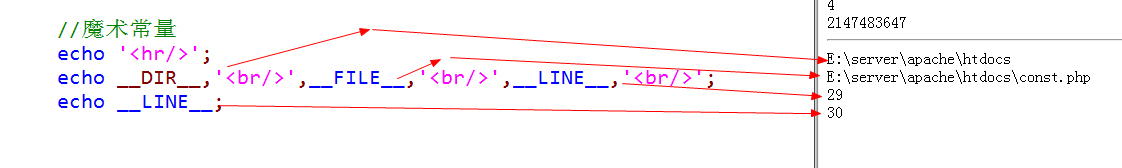


在PHP中还有一些特殊的常量，他们有双下划线开始+常量名+双下划线结束，这种常量称之为系统魔术常量：魔术常量的值通常会跟着环境变化，但是用户改变不了

\_\_DIR\_\_：当前被执行的脚本所在电脑的绝对路径

\_\_FILE\_\_：当前被执行的脚本所在的电脑的绝对路径（带自己文件的名字）

\_\_LINE\_\_：当前所属的行数



\_\_NAMESPACE\_\_：当前所属的命名空间

\_\_CLASS\_\_：当前所属的类

\_\_METHOD\_\_：当前所属的方法

# 数据类型

数据类型：data type，在PHP中指的是存储的数据本身的类型，而不是变量的类型。PHP是一种弱类型语言，变量本身没有数据类型。

## PHP的八种数据类型

在PHP中将数据分为三大类八小类：

简单（基本）数据类型：4个小类

整型：int/integer，系统分配4个字节存储，表示整数类型（有前提）

浮点型：float/double，系统分配8个字节存储，表示小数或者整型存不下的整数(比如32位存不下的整型)

字符串型：string，系统根据实际长度分配，表示字符串（引号）

布尔类型：bool/boolean，表示布尔类型，只有两个值：true和false

复合数据类型：2个小类

对象类型：object，存放对象（面向对象）

数组类型：array，存储多个数据（一次性）

特殊数据类型：2个小类

资源类型：resource，存放资源数据（PHP外部数据，如数据库、文件）

空类型：NULL，只有一个值就是NULL（不能运算）

## 类型转换

类型转换：在很多的条件下，需要指定的数据类型，需要外部数据（当前PHP取得的数据），转换成目标数据类型

在PHP中有两种类型转换方式：

1、 自动转换：系统根据需求自己判定，自己转换（用的比较多，系统自己判断需要的类型，效率偏低）

2、 强制（手动）转换：认为根据需要的目标类型转换

强制转换规则：在变量之前增加一个括号()，然后在里面写上对应类型：int/integer….其中NULL类型用到unset()

在转换过程中，用的比较多的就是转布尔类型（判断）和转数值类型（算术运算）

其他类型转布尔类型：true或者false，在PHP中比较少类型会变成false



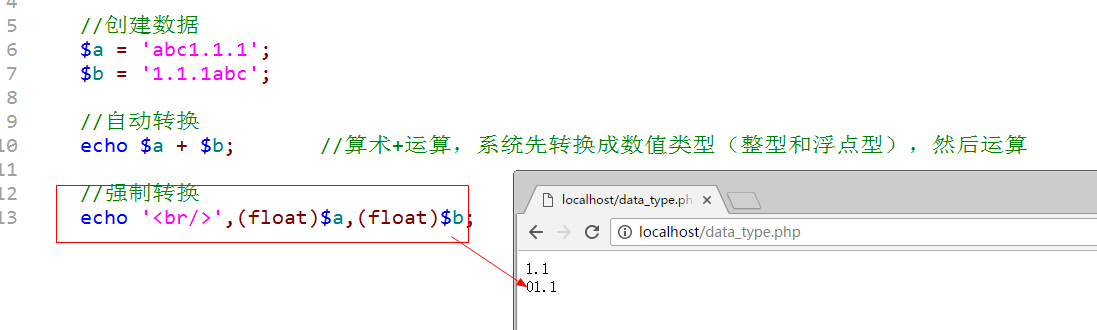
其他类型转数值的说明

1、 布尔true为1，false为0；

2、 字符串转数值有自己的规则

1)以字母开头的字符串，永远为0；

2)以数字开头的字符串，取到碰到字符串为止（不会同时包含两个小数点）

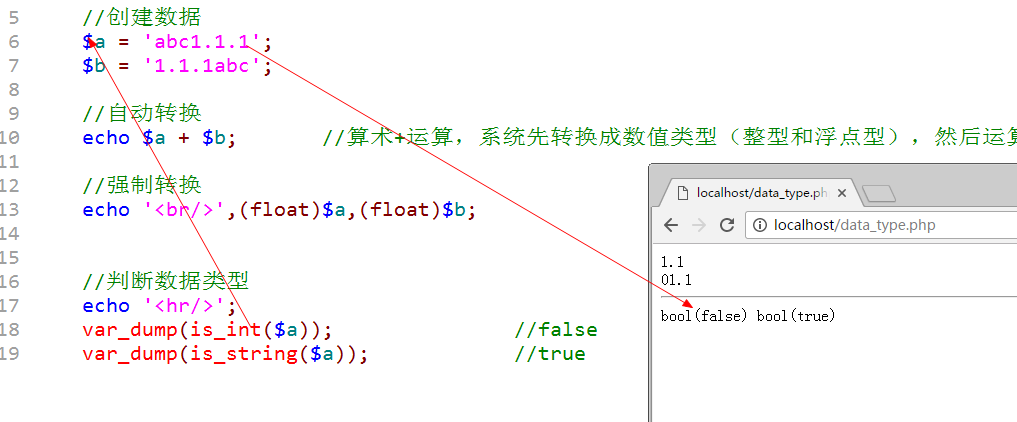


## 类型判断

通过一组类型判断函数，来判断变量，最终返回这个变量所保存数据的数据类型（相同结果为true，失败为false）：是一组以is\_开头后面跟类型名字的函数：is\_XXX(变量名)

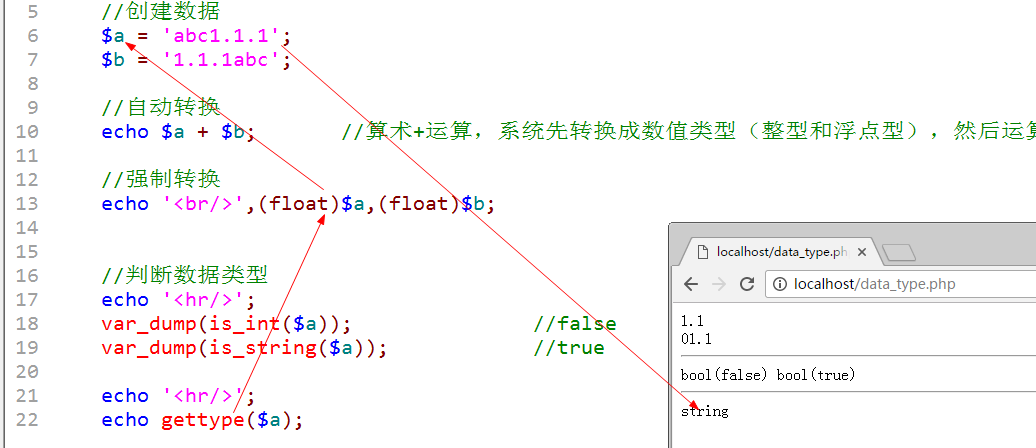
Bool类型不能用echo来查看(不能分辨是字符串的true还是bool的true)，可以使用var\_dump结构查看

Var\_dump(变量1,变量2…)



还有一组函数可以用来获取以及设定数据（变量）的类型

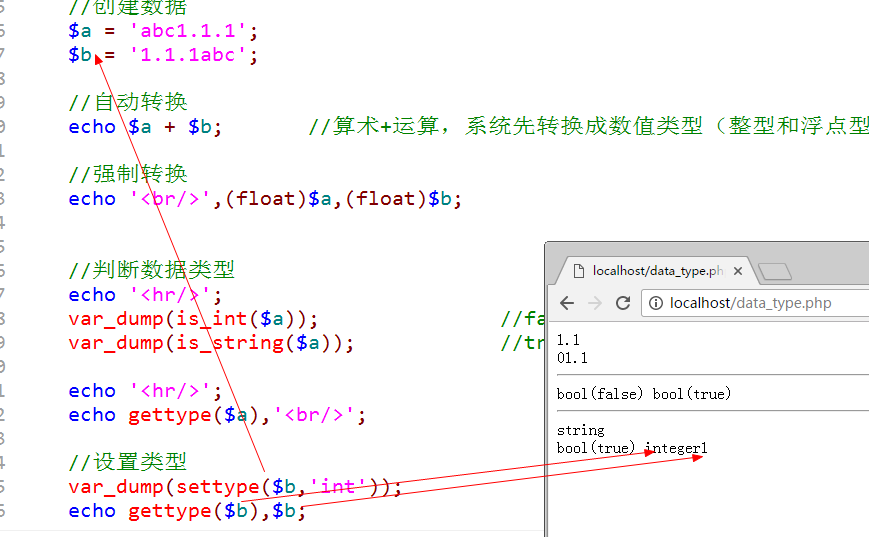
Gettype(变量名)：获取类型，得到的是该类型对应的字符串



Settype(变量名,类型)(返回bool值表示转换成功与否)：设定数据类型：与强制转换不同

1)强制转换(类型)变量名，是对数据值复制的内容进行处理（不会处理实际存储的内容）

2)settype会直接改变数据本身



# 整数类型

整数类型：保存整数数值（范围限制），4个字节存储数据，最大就是32位：42亿多。但是在PHP中默认是有符号类型（区分正负数）

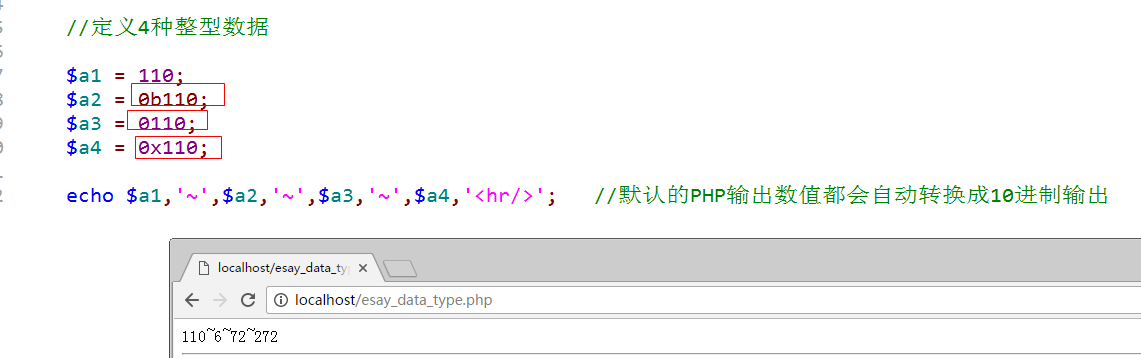
在PHP中提供了四种整型的定义方式：十进制定义，二进制定义，八进制定义和十六进制定义

$a = 120; //10进制

$a=0b110; //2进制 0b开头 b:binary

$a=0120; //8进制 0开头

$a=0x120; //16进制 0x开头



十进制：逢10进1，能够出现的数字是0-9

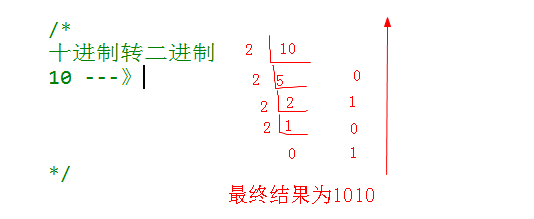
二进制：逢2进1，能够出现的数字是0-1

八进制：逢8进1，能够出现的数字是0-7

十六进制：逢16进1，能够出现的数字是0-9以及a-f，a表示10，依次类推

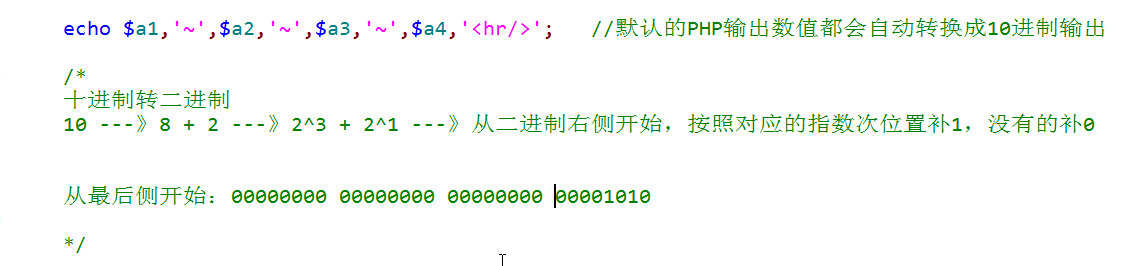
进制转换：手动转换

10进制转二进制(方法一)：除2倒取余法

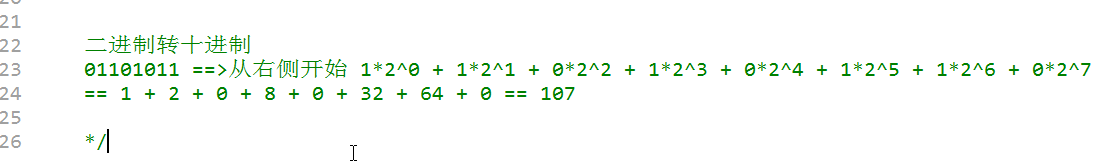


不管得到的结果如何，需要补足32位：前面补0：00000000 00000000 00000000 00001010

10进制转二进制(方法二)：取出最大的2的N次方，直到结果为0或1



二进制到十进制之间的转换：从右侧开始，将对应的第几位作为2的指数，然后将所有的结果相加



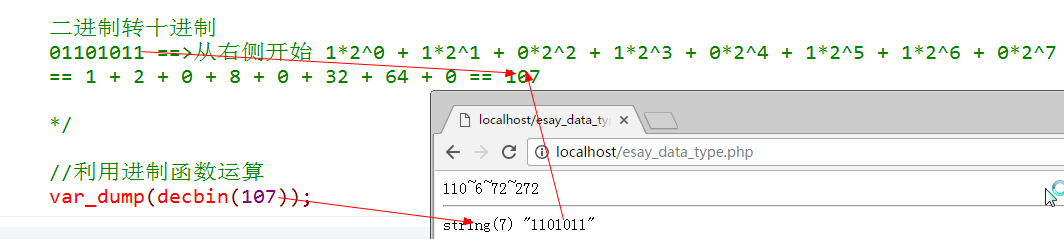
PHP中不需要用户这么复杂的去计算，提供了很多的函数进行转换

Decbin()：十进制转二进制

Decoct()：十进制转八进制

Dechex()：十进制转十六进制

Bindec()：二进制转十进制



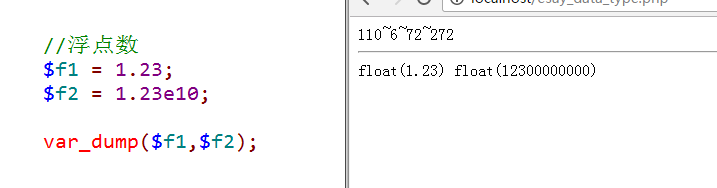
# 浮点类型

浮点型：小数类型以及超过整型所能存储范围的整数（不保证精度），精度范围大概在15个有效数字左右

浮点型定义有两种方式：

$f = 1.23;

$f = 1.23e10; //科学计数法，其中e表示底10。1.23\*10^10



简单说明浮点数为什么同样的字节数存储数据，但是却能表示更大的数据呢？

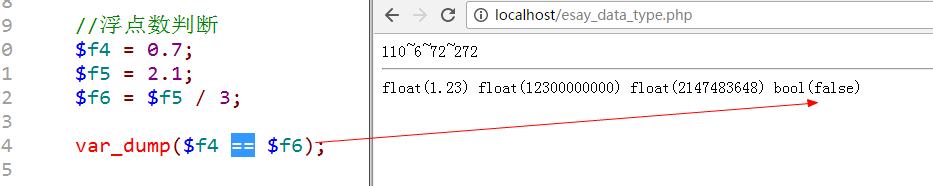
00000000 00000000 00000000 00000000 🡪 11111111 11111111 11111111 1111111 （整型最大值：所有位都是有效数据）

浮点数：红色7位算的结果是10的指数，后面三个字节存储表示具体数值

00000000 00000000 00000000 00000000 🡪 11111111 11111111 11111111 1111111

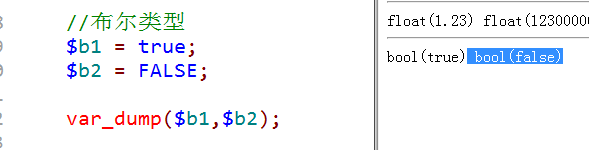
最后的结果是后三个字节的值\*10的指数，这样就能表示更大的数据。也正因此所以浮点数保存的数据不够精确

**尽量不要用浮点数做精确判断：浮点数保存的数据不够精确，而且在计算机中凡是小数基本上存的都不准确**



# 布尔类型

布尔类型：两个值true和false，通常是用于判断比较



在进行某些数据判断的时候，需要特别注意类型转换

Empty()：**判断数据的值是否为“空”**，不是NULL，如果为空返回true，不为空返回false

Isset()：**判断数据存储的变量**本身**是否存在**，存在变量返回true，不存在返回false

