**PHP语言的特点：**

**变量：**

1. **PHP的变量名称都是以$符号开头的**

**2.双引号和单引号**

在双引号中，变量名称将被变量值代替。

在单引号中，变量名称，或者任何其他文本都会不经修改而发送给浏览器。

1. **不要求在使用变量之前声明变量，当第一次给一个变量赋值时，才创建这个变量；并且PHP会“自动地”获得输入的数据类型。**
2. **支持改变变量名称**

如：

$a = ‘newName’;

$$a = 1;

相当于：$newName = 1;

1. **常量的声明和使用**

define定义全局常量，在任何地方都可以访问，define进行宏定义的时候，不会分配内存空间，**编译时**进行替换，只是单纯的替换，不会进行任何检查,比如类型，语句结构等，即宏定义常量只是纯粹的置放关系，

声明：define（‘常量名name---一般由大写字母组成’，value）；

使用：只需要使用常量名name，前面不需要加$符号。

const用于类成员变量定义,只能用类名访问

**操作符：**

1. **字符串连接操作符：点（.）**

如：

$a = ‘Hello ’;

$b = ‘World’;

$c = $a.$b;

则$c的值是‘Hello World’；

1. **引用操作符&**

引用（&）就像一个别名，而不是一个指针

通常，在将一个变量赋值给另一个变量时，先产生一个原变量的副本，再将该副本保存到内存的其他地方。

如：

$a = 1;

$b = $a;

改变$a的值，不会影响$b的值。

如：

$a = 1;

$b = &$a;

改变$a的值，会影响$b的值。

1. **->**

访问类的成员

1. **类型操作符instanceof**

检查一个对象是否是特定类的实例

如：

Class myClass{};

$a = new myClass();

If($a instanceof myClass){}

**其他：**

**1.exit**

用来终止PHP的执行，剩下的脚本不会继续执行。

1. **错误抑制操作符@**

通过抑制PHP的错误信息，并且根据自己的方式给出错误提示

**数组：**

**关联数组（PHP独有的）**

将索引与值关联起来，可以用有意义的名。

$A = array(‘a’=> 1, ‘b’=>2, ‘c’=>3);

相当于key => value;

与普通数组的区别为’=>’.

可以通过foreach（），list（），each（）循环输出；list比较好

注意：使用each（）函数时，数组将记录当前元素，如果要在同一个脚本中执行两次，需要用reset（）函数将当前元素重新设置到数组的开始处。reset($A)

**函数：**

**函数的声明：**

1.以关键字**function**开始，并且没有返回类型，如果不写访问修饰符，默认为public

2.函数名一般为全小写，因为PHP的函数名不区分大小写的。

如：function my\_function( ){ }

PHP不支持函数重载，所以自定义函数不能和内置函数或者用户已定义的函数重名。

**变量作用域：**

链接地址：<http://php.net/manual/zh/language.variables.scope.php>

<?php  
$a = 1;  
include 'b.inc';  
?>

这里变量 $a 将会在包含文件 b.inc 中生效。

<?php  
$a = 1; /\* global scope \*/  
function Test()  
{  
    echo $a; /\* reference to local scope variable \*/  
}  
Test();  
?>

但是，在用户自定义函数中，一个局部函数范围将被引入。任何用于函数内部的变量按缺省情况将被限制在局部函数范围内。

这个脚本不会有任何输出，因为 echo 语句引用了一个局部版本的变量 $a，而且在这个范围内，它并没有被赋值。你可能注意到 PHP 的全局变量和 C 语言有一点点不同，在 C 语言中，全局变量在函数中自动生效，除非被局部变量覆盖。这可能引起一些问题，有些人可能不小心就改变了一个全局变量。PHP 中全局变量在函数中使用时必须声明为 global。

<?php  
$a = 1; /\* global scope \*/  
$b = 2; /\* global scope \*/  
function Sum()  
{  
    global $a, $b;  
  
    $b = $a + $b;  
}  
Sum();  
echo $b;  
?>

以上脚本的输出将是“3”。在函数中声明了全局变量 $a 和 $b 之后，对任一变量的所有引用都会指向其全局版本。因为程序是按顺序执行的，所以先执行Sum（）函数，里面的$a,$b被声明为全局变量，对任一变量的所有引用都会指向其全局版本。

在函数之外的变量默认为全局变量，所以在函数外对变量用global是没有意义的，所以global是用于函数内部的，在函数结束时，该变量依旧存在，在函数外部可以使用，

被声明是global的不能初始化。

**类：**

**类的创建：**

1. 以关键字**class**开始，并且没有public，private等，以及返回类型，

如：class class\_name{ }

2.构造函数，名称必须是 **\_ \_construct( )**

如：function \_ \_construct( ){ }

PHP支持构造函数的重载（这里指的重载相当于C++的覆盖，指在子类再次声明和父类相同的属性和操作）

3.类里的属性（成员变量），有public，private，protected， 决定了变量的作用域

**类的继承：**

PHP支持单重继承，每个类至多一个父类

不支持多重继承，每个类可以有多个父类，可以通过接口解决

**Per-Class常量：（PHP的）**

每个类中可以定义自己类中的常量，用**const**关键字，在类外可以通过**：：（双冒号）**操作符指定常量所属的类来访问。

如：

<?Php

class Math{

const pi = 3.14159;

}

?>

Echo Math :: pi ; //允许在未初始化类的情况下就可以调用这个常量

**延迟静态绑定：**

允许一个静态继承的上下文中（**test( )**）对一个被调用类（**who( )**）的引用，父类可以使用子类重载的静态方法

class A{  
 public **static** function who(){  
 echo \_\_CLASS\_\_;  
 }  
 public **static** function test(){  
 **static::**who(); //延迟静态绑定  
 }  
}  
  
  
class B **extends** A{  
 public **static** function who(){  
 echo \_\_CLASS\_\_;  
 }  
}  
  
A::who();  
A::test();  
B::who();  
B::test();

输出为：AABB

test( )引用的who为他所在类的who（）

如果是self::who();

输出：AABA

取决于定义当前方法所在的类

**克隆对象：**

在变量前使用**clone**关键字，允许复制一个已有的对象（副本）；

如：$c = clone $b;

需要**\_ \_clone（）**方法，类似于\_ \_get( )和\_ \_set( )，在使用默认行为创建一个副本后能够被调用，所以在\_ \_clone（）方法定义自己需要的修改

**\_ \_call( )重载方法：**

\_ \_call( 被调用的方法名称，传递给该方法的参数数组)

当一个类中包含了\_ \_call( )方法，那么当调用这个类中不存在的方法，\_ \_call( )方法会被自动调用。

**异常：**

Php的异常必须手动抛出，利用**throw**关键字