父元素高度确定的单行文本的竖直居中的方法是通过设置父元素的 height 和 line-height 高度一致来实现的。

父元素高度确定的多行文本、图片等的竖直居中的方法有两种：

方法一：使用插入 table (包括tbody、tr、td)标签，同时设置 vertical-align：middle。

方法二：在 chrome、firefox 及 IE8 以上的浏览器下可以设置块级元素的 display 为 table-cell（设置为表格单元显示），激活 vertical-align 属性

.first span, #second span{color:green;} 分组选择器

层叠就是在html文件中对于同一个元素可以有多个css样式存在，当有相同权重的样式存在时，会根据这些css样式的前后顺序来决定，处于最后面的css样式会被应用。

letter-spacing 字母间距 word-spacing 单词间距

echo关键字与字符串之间至少要有一个空格

在php中字符串连接符是用点（.）来表示的

**PHP注释语句：用双斜杠（//）来表示，注释语句不能写在<?php ?>外面**

**Html中使用<!--注释语句-->**

**CSS中使用/\*注释语句\*/**

**网站根目录的自拟定**，默认放到wamp安装文件夹的www目录下

Php文件被apache执行

修改httpd conf 有个路径读取php文件（查找DocumentRoot,Directory也修改，配置修改都要重启服务）

WampManager.ini文件（查找menu.left，修改其中的www目录路径） WampManager.tpl（查找menu.left，修改directory）

Php的变量名用$xxx声明，以字母或者下划线开头，**可以包含汉字**...（不要使用汉字！）

**var\_dump()可以显示变量**

声明变量的时候不用注明数据类型，会自动转换

Echo指令输出bool类型的时候，true为1（不区分大小写），false什么都不输出

字符串型可以用三种方法定义：单引号形式、双引号形式和Heredoc结构形式

字符串中有单引号或者双引号：

第一种方案：在单引号中嵌入双引号，如上图第2行所示；

第二种方案：在双引号中嵌入单引号，如上图第3行所示；

第三种方案：使用转义符“\”

wamp多站点配置（如何使用一个web应用程序管理和运行多个网站）

bin/apache/conf/extra/httpd-vhosts.conf

修改documentroot和servername为项目路径和服务器名

在apache中默认禁止其他web主机地址访问服务器的资源

打开apache的httpd.conf，查找httpd-vhost，把注释去掉

找到自己修改的directory，把deny from all 改成allow，下面一行注释掉

再去system\drivers\etc\hosts 添加域名，表示从127.0.0.1优先请求资源

apache默认80端口，默认的时候输入网址不需要端口号，端口冲突会无法访问

http.conf 修改端口 查找80 改掉listen，改掉servername localhost

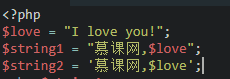
zend studio：php开发软件

zend工作目录设置为apache运行目录

当双引号中包含变量时，变量会与双引号中的内容连接在一起；

当单引号中包含变量时，变量会被当做字符串输出（直接输出$xxx）

“$var” 跟



使用Heredoc结构形式的方法来解决**字符串很长**的问题：先使用定界符表示字符串（<<<），接着在“<<<“之后提供一个标识符GOD，然后是字符串，最后以提供的这个标识符结束字符串。

特殊类型：资源

$file = fopen(“f.txt”, “r”); //打开文件

$con = mysql\_connect(“localhost”, “root”, “root”); //连接数据库

$img = imagecreate(100, 100); //图形画布

注意： f.txt文本文件保存时一定要设置为UTF-8格式

特殊类型：空类型 null不区分大小写

当被赋值为NULL，或者尚未被赋值，或者被unset()，这三种情况下变量被认为为NULL

PHP中的常量分为自定义常量和系统常量

自定义常量是根据我们开发的需要，而定义的常量，它通过使用PHP中的函数define()定义。

Define有三个参数，第一个为变量名，第二个为常量值，第三个为bool类型，决定是否区分大小写

注意：defined()函数也可以帮助我们判断一个常量是否已经定义 define($xxx)

系统常量是php已经定义好的常量，常见的有：

（1）\_\_FILE\_\_ :php程序文件名。它可以帮助我们获取当前文件在服务器的物理位置。

（2）\_\_LINE\_\_ :PHP程序文件行数。它可以告诉我们，当前代码在第几行。

（3）PHP\_VERSION:当前解析器的版本号。它可以告诉我们当前PHP解析器的版本号，我们可以提前知道我们的PHP代码是否可被该PHP解析器解析。

（4）PHP\_OS：执行当前PHP版本的操作系统名称。它可以告诉我们**服务器所用的操作系统**名称，我们可以根据该操作系统优化我们的代码。

常量取值：1.直接使用常量名 2.使用constant函数

Php的比较运算符



字符串连接运算符是为了将两个字符串进行连接，PHP中提供的字符串连接运算符有：

（1）连接运算符(“.”)：它返回将右参数附加到左参数后面所得的字符串。

（2）连接赋值运算符(“.=”)：它将右边参数附加到左边的参数后

可以将@放置在一个PHP表达式之前，该表达式可能产生的任何错误信息都被忽略掉

如果激活了track\_error（在php.ini中设置）特性，表达式所产生的任何错误信息都被存放在变量$php\_errormsg中

**php数组：**

PHP有两种数组：索引数组、关联数组

索引和关联两个词都是针对数组的键而言的

**索引数组**是指数组的键是整数的数组，并且键的整数顺序是从0开始，依次类推

索引数组赋值有三种方式:

**第一种：**用数组变量的名字后面跟一个中括号的方式赋值，当然，索引数组中，中括号内的**键一定是整数**。比如，$arr[0]='苹果';

**第二种：**用array()创建一个空数组，使用=>符号来分隔键和值，左侧表示键，右侧表示值。当然，索引数组中，键一定是整数。比如，array('0'=>'苹果');

**第三种：**用array()创建一个空数组，直接在数组里用英文的单引号'或者英文的双引号"赋值，数组会默认建立从0开始的整数的键。比如array('苹果');这个数组相当于array('0'=>'苹果');

关联数组赋值有两种方式:

**第一种：**用数组变量的名字后面跟一个中括号的方式赋值，当然，关联数组中，中括号内的**键一定是字符串**。比如，$arr['apple']='苹果';

**第二种：**用array()创建一个空数组，使用=>符号来分隔键和值，左侧表示键，右侧表示值。当然，关联数组中，键一定是字符串。比如，array('apple'=>'苹果');

PHP中foreach循环语句，常用于遍历数组，一般有两种使用方式:

不取下标 取下标

foreach (数组 as 值) foreach (数组 as 下标 => 值)

**类与面向对象：**

使用关键字static修饰的，称之为静态方法，**静态方法不需要实例化对象**，可以通过类名直接调用，操作符为双冒号::

一般通过->对象操作符来访问对象的属性或者方法，当在类成员方法内部调用的时候，可以使用$this伪变量调用当前对象的属性。

PHP5可以在类中使用\_\_construct()定义一个构造函数

在子类中如果定义了\_\_construct则不会调用父类的\_\_construct，如果需要同时调用父类的构造函数，需要使用parent::\_\_construct()显式的调用

PHP5支持析构函数，使用\_\_destruct()进行定义，当PHP代码执行完毕以后，会自动回收与销毁对象，因此一般情况下不需要显式去销毁对象

静态属性不允许对象使用->操作符调用

静态方法可以通过变量来进行动态调用

静态方法中，$this伪变量不允许使用。可以使用self，parent，static在内部调用静态方法与属性

PHP中的重载指的是动态的创建属性与方法

属性的重载通过\_\_set，\_\_get，\_\_isset，\_\_unset来分别实现对**不存在的属性**的赋值、读取、判断属性是否设置、销毁属性

方法的重载通过\_\_call来实现，当**调用不存在的方法**的时候，将会转为参数调用\_\_call方法，当调用不存在的静态方法时会使用\_\_callStatic重载

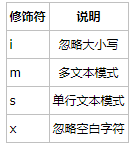
对象比较，当同一个类的两个实例的所有属性都相等时，可以使用比较运算符==进行判断，当需要判断两个变量是否为同一个对象的引用时，可以使用全等运算符===进行判断

对象复制，在一些特殊情况下，可以通过关键字**clone**来复制一个对象，这时\_\_clone方法会被调用，通过这个魔术方法来设置属性的值。

对象序列化，可以通过**serialize**方法将对象序列化为字符串，用于存储或者传递数据，然后在需要的时候通过**unserialize**将字符串**反序列化**成对象进行使用。

正则匹配模式使用分隔符与元字符组成，分隔符可以是非数字、非反斜线、非空格的任意字符。经常使用的分隔符是正斜线(/)、hash符号(#) 以及取反符号(~)

分隔符后面可以使用模式修饰符，模式修饰符包括：i, m, s, x等



当转换为 boolean 时，以下值被认为是 FALSE：

布尔值 FALSE 本身

整型值 0（零）

浮点型值 0.0（零）

空字符串，以及字符串 "0"

不包括任何元素的数组

特殊类型 NULL（包括尚未赋值的变量）

从空标记生成的 SimpleXML 对象

所有其它值都被认为是 TRUE（包括任何资源 和 NAN）

**正则表达式：**

对字符串进行操作的一种逻辑公式，用一些特定的字符组合成一个规则字符串

正则表达式使用**分隔符**与**元字符**：分隔符可以是除了数字、字母、以及反斜线以外的任何字符，常用的是 **/，#，~**

用了分隔符以后，匹配的即为分隔符中间的模式串**，如果模式串里面有分隔符需要用转义字符 \ ，如模式串为[/\] 需要写成[\/\\]**

正则表达式中含有特殊含义的叫元字符：

\ 一般用于转义字符

^ 断言目标的开始位置(或在多行模式下是行首)

$ 断言目标的结束位置(或在多行模式下是行尾)

. 匹配除换行符外的任何字符(默认)

[ ] 字符类定义

| 开始一个可选分支

( ) 子组定义

? 量词，表示 0 次或 1 次匹配。**位于量词后面用于改变量词的贪婪特性**

\* 量词，0 次或多次匹配

+ 量词，1 次或多次匹配

{ } 自定义量词

有的元字符可以在方括号里面使用，比如：

\ 转义字符

**^ 仅在作为第一个字符(方括号内)时，表明字符类取反**

- 标记字符范围

例子：

\s：匹配任意的空白符，包括空格、制表符、换行符

\w：匹配字母或数字或下划线

\d：匹配数字

[\w\.\-]：括住的内容只匹配一个单一的字符

(bc)\*：bc被定义成一个子组（视作一个字符）

(bc){2,5}：表示2到5个bc

^abc$：表示以某个abc为开头与结尾的字符串，此处只有abc能匹配

在php中，提供了preg\_match()和preg\_match\_all函数进行正则匹配

这两个函数原型如下：

int preg\_match|preg\_match\_all ( string $pattern , string $subject [, array &$matches [, int $flags = 0 [, int $offset = 0 ]]] )

**pattern:**要搜索的模式，字符串类型。

**subject :**输入字符串。

**matches:**如果提供了参数matches，它将被填充为搜索结果。 matches[0]将包含完整模式匹配到的文本，matches[1]将包含第一个捕获子组匹配到的文本，以此类推。

**flags:**flags可以被设置为以下标记值：**PREG\_OFFSET\_CAPTURE** 如果传递了这个标记，对于每一个出现的匹配返回时会附加字符串偏移量(相对于目标字符串的)。 注意：这会改变填充到matches参数的数组，使其每个元素成为一个由 第0个元素是匹配到的字符串，第1个元素是该匹配字符串 在目标字符串subject中的偏移量。

**offset:**通常，搜索从目标字符串的开始位置开始。可选参数 offset 用于 指定从目标字符串的某个未知开始搜索(单位是字节)。

返回值：preg\_match()返回 pattern 的**匹配次数**。它的值将是0次（不匹配）或1次，因为 **preg\_match()在第一次匹配后将会停止搜索**。preg\_match\_all()不同于此，它会**一直搜索subject直到到达结尾**。 如果发生错误 preg\_match()返回 FALSE。

单词定界符

\b\b 匹配的必须是个完整的单词

\B\B 匹配的不能是一个完整的单词

当用某种边界来查找，而所要获取的内容又不包含边界时，可使用小括号来指定所要的范围

带 "\*" 和 "{m,n}" 的表达式都是尽可能地多匹配，带 "?" 的表达式在可匹配可不匹配的时候，也是尽可能的 "要匹配"。这 种匹配原则就叫作 **"贪婪" 模式** 。

在修饰匹配次数的特殊符号后再加上一个 "?" 号，则可以使匹配次数不定的表达式尽可能少的匹配，使可匹配可不匹配的表达式，尽可能的 "不匹配"。这种匹配原则叫作 "非贪婪" 模式（**懒惰模式**）

**反向引用**

在php正则表达式中，通过\n，来表示第n次匹配到的结果，要注意的是，在使用反向匹配的时候都需要使用()，反向匹配时，匹配()里面出现的字符或字符串

正则表达式在字符串的搜索和替换中很有用

preg\_replace('/\s+/', ' ', $str); 会把匹配到的串都替换

注意：如果仅仅想要检查一个字符串是否包含另外一个字符串, 不要使用preg\_match()，使用strpos()或strstr()会更快。

正则匹配常用在表单验证上，一些字段会有一定的格式要求，比如用户名一般都要求必须是字母、数字或下划线组成，邮箱、电话等也都有自己的规则，因此使用正则表达式可以很好的对这些字段进行验证。

**Cookie：**

Cookie是存储在客户端浏览器中的数据，我们通过Cookie来跟踪与存储用户数据。Cookie是存在于HTTP的标头之中。

PHP设置Cookie最常用的方法就是使用setcookie函数，setcookie具有7个可选参数，我们常用到的为前5个：

**name（Cookie名）** 可以通过$\_COOKIE['name'] 进行访问

**value（Cookie的值）**

**expire（过期时间）** Unix时间戳格式，默认为0，表示浏览器关闭即失效

**path（有效路径）** 如果路径设置为'/'，则整个网站都有效

**domain（有效域）** 默认整个域名都有效，如果设置了'www.imooc.com',则只在www子域中有效

cookie仍然具有一些局限：

cookie相对不是太安全，容易被盗用导致cookie欺骗

单个cookie的值最大只能存储4k

每次请求都要进行网络传输，占用带宽

session是将用户的会话数据存储在服务端，没有大小限制，通过一个session\_id进行用户识别，PHP默认情况下session\_id是通过cookie来保存的，因此从某种程度上来说，seesion依赖于cookie

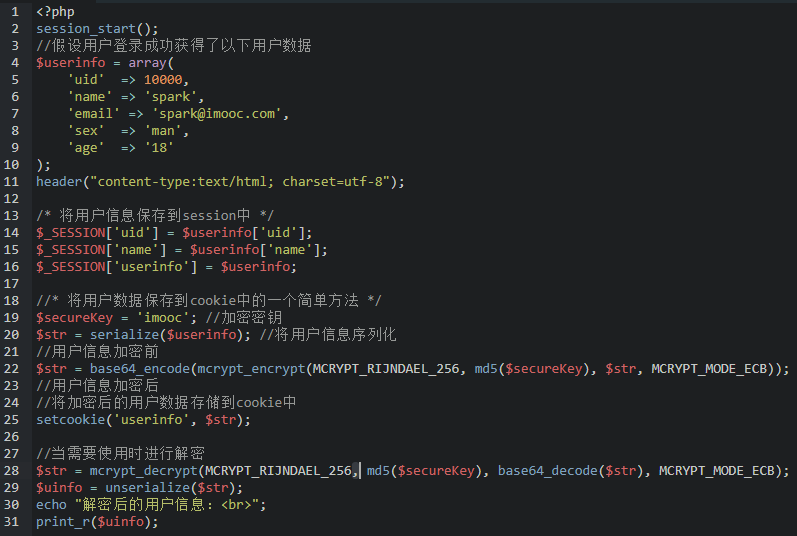
如果要删除所有的session，可以使用session\_destroy函数，session\_destroy会删除所有数据，但是session\_id仍然存在。

注意：session\_destroy并不会立即的销毁全局变量$\_SESSION中的值，只有当下次再访问的时候，$\_SESSION才为空，因此如果需要立即销毁$\_SESSION，可以使用unset函数。

如果需要同时销毁cookie中的session\_id，则还需要显式调用setcookie方法删除cookie中的session\_id值。

登录信息既可以存储在sessioin中，也可以存储在cookie中，他们之间的差别在于session可以方便的存取多种数据类型，而cookie只支持字符串类型，同时对于一些安全性比较高的数据，cookie需要进行格式化与加密存储，而session存储在服务端则安全性较高

一个用session存储用户信息的好例子：



**php文件系统：**

PHP具有丰富的文件操作函数，最简单的读取文件的函数为file\_get\_contents，可以将整个文件全部读取到一个字符串中，也可以通过参数控制读取内容的开始点以及长度：

$content = file\_get\_contents('./test.txt', null, null, 100, 500);

PHP也提供类似于C语言操作文件的方法，使用fopen，fgets，fread等方法，fgets可以从文件指针中读取一行，freads可以读取指定长度的字符串

注意：使用fopen打开的文件，最好使用fclose关闭文件指针，以避免文件句柄被占用

一般情况下在对文件进行操作的时候需要先判断文件是否存在，PHP中常用来判断文件存在的函数有两个is\_file与file\_exists

file\_exists不仅可以判断文件是否存在，同时也可以判断目录是否存在。is\_file是确切的判断给定的路径是否是一个文件。更加精确的，可以使用is\_readable与is\_writeable在文件是否存在的基础上，判断文件是否可读与可写

fileowner：获得文件的所有者

filectime：获取文件的创建时间

filemtime：获取文件的修改时间

fileatime：获取文件的访问时间

其中最常用的是文件的修改时间，通过文件的修改时间，可以判断文件的时效性，经常用在静态文件或者缓存数据的更新

number\_format(number,decimals,decimalpoint,separator)

描述

number 必需。要格式化的数字。如果未设置其他参数，则数字会被格式化为不带小数点且以逗号（,）作为千位分隔符。

decimals 可选。规定多少个小数。如果设置了该参数，则使用点号（.）作为小数点来格式化数字。

decimalpoint 可选。规定用作小数点的字符串。

separator 可选。规定用作千位分隔符的字符串。仅使用该参数的第一个字符。比如 "xxx" 仅输出 "x"。 注：如果设置了该参数，那么所有其他参数都是必需的。

返回值： 返回已格式化的数字

通过filesize函数可以取得文件的大小，文件大小是以字节数（bytes）表示的

若要转换文件大小的单位，可以自己定义函数实现

注意：没法通过简单的函数来取得目录的大小，目录的大小是该目录下所有子目录以及文件大小的总和，因此需要通过递归的方法来循环计算目录的大小

与读取文件对应，PHP写文件也具有两种方式，最简单的方式是采用file\_put\_contents。

PHP也支持类似C语言风格的操作方式，采用fwrite进行文件写入。

PHP使用unlink函数进行文件删除。删除文件夹使用rmdir函数，文件夹必须为空，如果不为空或者没有权限则会提示失败。如果文件夹中存在文件，可以先循环删除目录中的所有文件，然后再删除该目录，循环删除可以使用glob函数遍历所有文件：

foreach (glob("\*") as $filename) {

unlink($filename);

}

**php异常处理：**

异常处理是面向对象一个重要特性，PHP代码中的异常通过throw抛出，异常抛出之后，后面的代码将不会再被执行。异常抛出被用于在遇到未知错误，或者不符合预先设定的条件时，通知客户程序，以便进行其他相关处理，不至于使程序直接报错中断。当代码中使用了try catch的时候，抛出的异常会在catch中捕获，否则会直接中断

。

Try - 使用异常的函数应该位于 "try" 代码块内。如果没有触发异常，则代码将照常继续执行。但是如果异常被触发，会抛出一个异常。

Throw - 这里规定如何触发异常。注意：每一个 "throw" 必须对应至少一个 "catch"，当然可以对应多个"catch"

Catch - "catch" 代码块会捕获异常，并创建一个包含异常信息的对象。

通过exception 对象调用 $e->getMessage()，输出来自该异常的错误消息

PHP具有很多异常处理类，其中Exception是所有异常处理的基类。

Exception具有几个基本属性与方法，常用的属性包括：

message 异常消息内容

code 异常代码

file 抛出异常的文件名

line 抛出异常在该文件的行数

常用的方法有：

getTrace 获取异常追踪信息

getTraceAsString 获取异常追踪信息的字符串

getMessage 获取出错信息

如果必要的话，可以通过继承Exception类来建立自定义的异常处理类。

**php 数据库操作：**

mysql扩展进行数据库连接的方法：

$link = mysql\_connect('mysql\_host', 'mysql\_user', 'mysql\_password');

该函数需要指定数据库的地址，用户名及密码。

连接成功以后，我们需要选择一个操作的数据库，通过mysql\_select\_db函数来选择数据库

通常我们会先设置一下当前连接使用的字符编码，一般的我们会使用utf8编码。

mysql\_query("set names 'utf8'");

通过上面的步骤，我们就与数据库建立了连接，可以进行数据操作了。

在数据库建立连接以后就可以进行查询，采用mysql\_query加sql语句的形式向数据库发送查询指令。**对于查询类的语句会返回一个资源句柄（resource），可以通过该资源获取查询结果集中的数据。**

默认的，PHP使用最近的数据库连接执行查询，但如果存在多个连接的情况，则可以通过参数指令从那个连接中进行查询。

$res = mysql\_query('select \* from user limit 1', $link1); //从第一个连接中查询数据

在mysql中，执行插入语句以后，可以得到自增的**主键id**，通过PHP的mysql\_insert\_id函数可以获取该id。

$uid = mysql\_insert\_id();

这个id的作用非常大，通常可以用来判断是否插入成功，或者作为关联ID进行其他的数据操作。

PHP有多个函数可以获取数据集中的一行数据，最常用的是mysql\_fetch\_array，可以通过设定参数来更改行数据的下标，默认的会包含数字索引的下标以及字段名的关联索引下标。

可以通过设定参数MYSQL\_NUM只获取数字索引数组，等同于mysql\_fetch\_row函数，如果设定参数为MYSQL\_ASSOC则只获取关联索引数组，等同于mysql\_fetch\_assoc函数。

如果要获取数据集中的所有数据，我们通过循环来遍历整个结果集。

在实际应用中，我们并不希望一次性获取数据表中的所有数据，那样性能会非常的低，因此会使用翻页功能，每页仅显示10条或者20条数据。

通过mysql的limit可以很容易的实现分页，**limit m,n表示从m行后取n行数据**

假定当前页为$page，每页显示$n条数据，那么m为当前页前面所有的数据

即$m = ($page-1) \* $n

只需要构建好相应的sql语句，然后调用mysql\_query执行就能完成数据的更新与删除操作。

对于删除与更新操作，可以通过**mysql\_affected\_rows**函数来获取有多少行的数据被更新，若结果为0，则说明数据没有发生变化。

当数据库操作完成以后，可以使用**mysql\_close**关闭数据库连接，默认的，当PHP执行完毕以后，会自动的关闭数据库连接。在对性能要求比较高的情况下，可以在进行完数据库操作之后尽快关闭数据库连接，以节省资源，提高性能。在存在多个数据库连接的情况下，可以设定连接资源参数来关闭指定的数据库连接。

**php字符串：**

标量类型，写到定界符（单引号、双引号、heredoc，nowdoc）中

双引号会解析所有的转义符

在PHP中有urlencode()、urldecode()、rawurlencode()、rawurldecode()这些函数来解决网页URL编码解码问题。

urlencode： 是指针对网页url中的中文字符的一种编码转化方式，最常见的就是Baidu、Google等搜索引擎中输入中文查询时候，生成经过 Encode过的网页URL

rawurlenconde和urlencode的区别：

1. 数字、大小写字母都不编码

2. 减号、点号、下划线 三个不编码

3. rawurlencode比urlencode多编码一个”加号”

可以用花括号来明确变量名的界限，建议把整体都放到花括号里面{$username}，花括号之间不要有空格

可以通过花括号对字符串中的指定字符做增删改查操作

echo str{0} 注意：中文字符占3个字节

删除可以通过用空字符串替换来完成

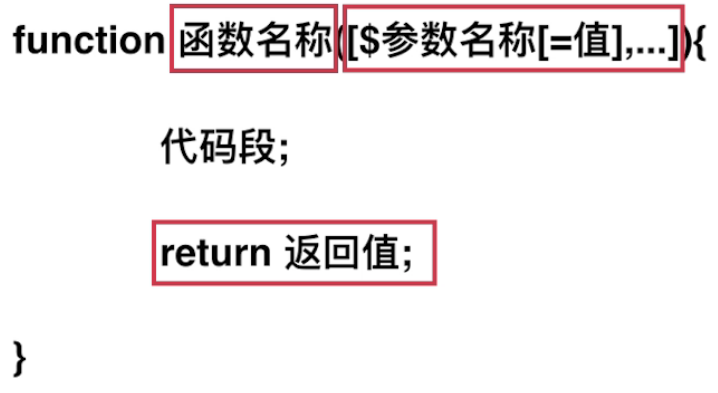
单引号与双引号的区别

一般情况下两者是通用的.但双引号内部变量会解析,单引号则不解析.

所以如果内部只有纯字符串的时候,用单引号(速度快),内部有别的东西(如变量)的时候,用双号引更好点.

**自定义函数：**

按照具体需求封装的函数



单一职责：一个函数只完成一个功能

**函数名称不区分大小写**

**函数名称不可以重名！！**

检测函数名是否存在的函数：function\_exists($funcName)

函数默认返回NULL

int intval ( mixed $var [, int $base = 10 ] )

通过使用指定的进制 base 转换（默认是十进制），返回变量 var 的 integer 数值。 intval() 不能用于 object，否则会产生 E\_NOTICE 错误并返回 1

strrpos() - 函数查找字符串在另一字符串中**最后一次**出现的位置（区分大小写）

strripos() - 查找字符串在另一字符串中最后一次出现的位置（不区分大小写）

strpos() - 查找字符串在另一字符串中**第一次**出现的位置（区分大小写）

stripos() - 查找字符串在另一字符串中第一次出现的位置（不区分大小写）

require 和 include 几乎完全一样，除了处理失败的方式不同之外。require 在出错时产生 E\_COMPILE\_ERROR 级别的错误。换句话说将导致脚本中止而 include 只产生警告（E\_WARNING），脚本会继续运行。