WEB后端架构-PHP概述

【学习目标、重难点知识】

1. 环境安装
2. 基本语法
3. 变量
4. 常量
5. 输出
6. 数据类型及比较
7. 运算符
8. 分支语句
9. 循环语句
10. 数组
11. 函数

什么是网站?

浏览器里面输入一个网址/域名所看到页面就可以理解为是一个网址

前端： html+css+js

后端：数据处理/逻辑处理

数据库：MySQL，Oracle，MSSql.....

数据的产生，数据的传输，数据存储

BS架构，CS架构

BS： Brower<---->Server

- 只要有浏览器就可以访问系统
- 更新系统只需要对应服务商直接再服务器更新即可---->程序只有一套放在服务器的--->只用维护一套
- 为什么没有全部用BS架构？--->网速不够
- 如果以后网速够了---->90%都会转成BS

CS：Client<-------->Server

- 必须要安装一个客户端软件
- 更新软件客户端必须要更新----->需要用户统一----->程序有两套----->同时要维护两套程序（游戏软件原神LOL...社交软件QQ微信...视频软件爱奇艺腾讯....）
- 需要提前下载很多资源
- 现在手机上的CS架构其实是伪CS架构： 套了一个壳子，里面还是网页

网页： 静态网页，动态网页

静态网页：里面的数据是写死的，不和后端进行交互

动态网页：数据是动态渲染的，数据是和后端进行交互的

动态网页必定是有后端：

后端有哪些技术呢?

PHP: LAMP（Linux+Apache+MySQL+PHP）

JAVA: Spring+SpringBoot+SpringCloud+MyBatis/Hibernate+MySQL/Oracle+Linux JSP

Python: Flask框架,Django框架

Perl：LAMP（Linux+Apache+MySQL+Perl）

C#：.NET, ASP C#+Windows+MSSQL

Node.js: 昙花一现

主流遇到的后端的语言： PHP， JAVA

PHP: 主要是网站， 开放给所有人访问的（网站首页、门户网站等等）

JAVA: 主要系统, 业务系统，数据量巨大，用户是专业的用户（各大功能业务线的网站，需要高并发）

PHP单纯从功能的角度，没有任何一点比Java差

Java的并发性能很高

Java是编译性的语言，超级浪费时间，时间成本很高

1. PHP 介绍

1.1. PHP 概述

1.1.1. PHP 是什么?

- 我们应用的所有软件都是由**计算机语言**编写的。
- 目前流行的编程语言有很多，例如：`PHP`、`Java`、`Python`、`JavaScript`、`C/C++`和`Go`语言等，全世界有`**600**`多种编程语言，`PHP`则是众多计算机编程语言中的一种，用于网络开发，尤其适用于`Web`开发领域，主要目标是**快速编写动态网页**。
- 用`PHP`做出的动态页面与其他的编程语言相比，`PHP`是将程序嵌入到`HTML`（标准通用标记语言下的一个应用）文档中去执行，执行效率比完全生成`HTML`标记的其他编程语言要高许多。
- **`PHP`能运行在`Windows`、`Linux`等绝大多数操作系统环境中，常与开源免费的`Web`服务器（`Apache`或`Nginx`）和数据库（`Mysql`及`Redis`）配合使用，用于`Linux`平台上（简称`LAMP/LNMP`），具有最高的性价比，号称“`Web`架构黄金组合”，形成了现在非常流行的`Web`开发技术。**
- `PHP`是`Hypertext Preprocessor`（超文本预处理器）的缩写，是一种在服务器端运行的的开源的、可以嵌入在 HTML 页面中的脚本语言。
- `PHP`的默认文件扩展名是以`.php`结尾的。

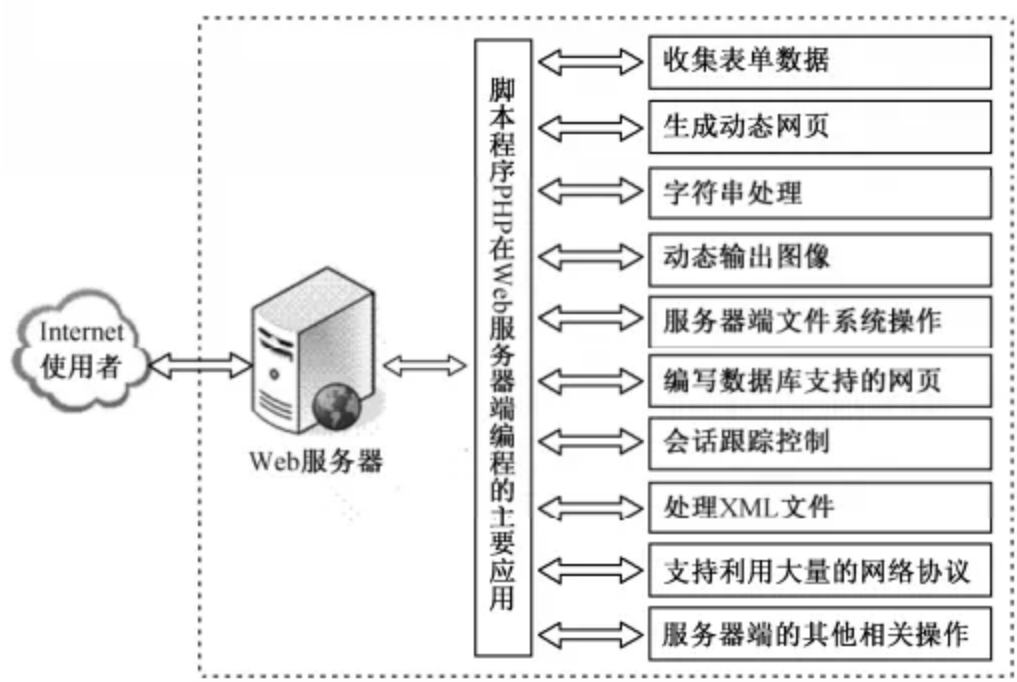
问：上面的`Hypertext Preprocessor`是怎么简写成`PHP`的?

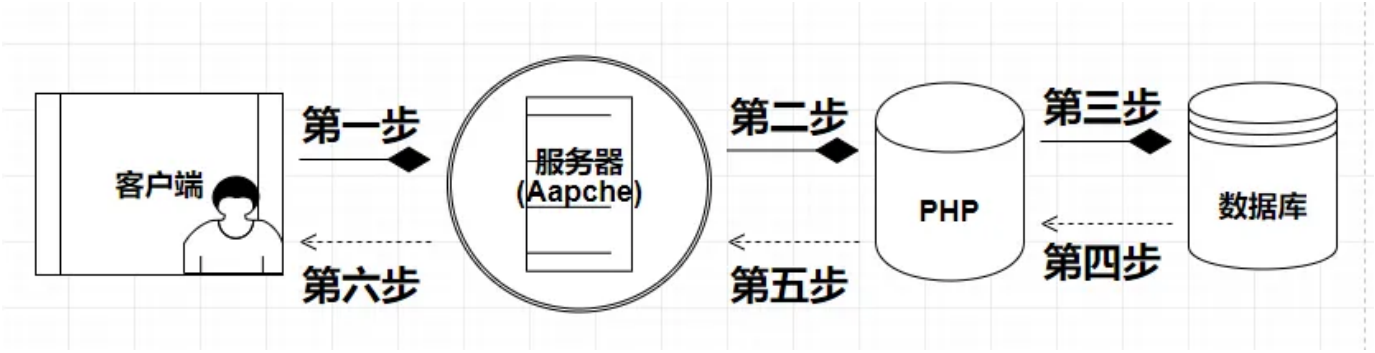
- `PHP（Personal Home Page）`是`PHP`最早的名字。

1.1.2. PHP 都能做什么?

- `PHP`能做很多事，但`PHP`主要是在`Web`开发中用于服务器端的脚本程序。

- `PHP` 需要安装 `PHP` 应用程序服务器去解释执行，是用来协助 `Web` 服务器工作的编程语言，也可以说是对 `Web` 服务器功能的扩展，并外挂在 `Web` 服务器上一起工作。
- 用户如果通过浏览器访问 `Web` 服务器需要得到动态响应的结果， `Web` 服务器就要委托 `PHP` 脚本编程语言来完成了。





1.2. PHP 的发展

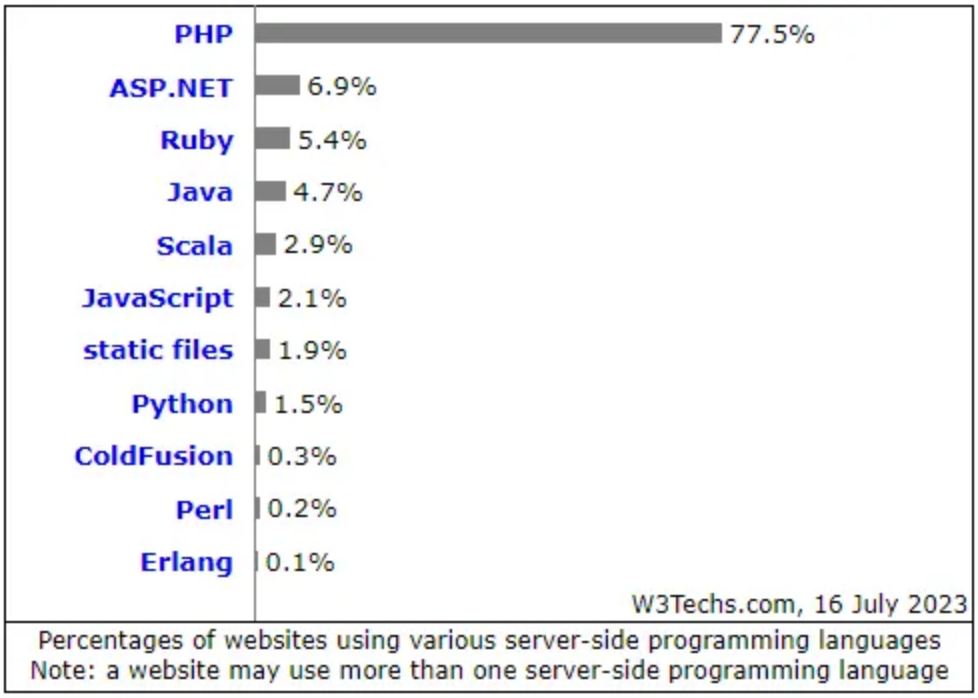
- 最初创建时， `PHP` 是一个简单的用 `Perl` 语言编写的程序，只是为了统计自己的网站有多少访问者。
- 后来又用 `C` 语言重新编写，多年来 `PHP` 经过无数开源贡献者的不断迭代，历经数个版本，已经成为当前最热门的 `Web` 开发语言。
- 像 `Facebook` 、淘宝等早期都是用 `PHP` 写的，在中国 `PHP` 在百度、新浪、腾讯等大型互联网公司中应用都比较多。

1.2.1. PHP 的诞生

- `1994` 年丹麦人 `Rasmus Lerdorf`（雷斯莫斯·勒道夫）创建了 `PHP`，最初只是一套简单的 `Perl` 脚本，用来跟踪访问他主页的人们的信息。
- 他给这一套脚本取名为 `Personal Home Page Tools`，后来他又用 `C` 语言重新编写，包括可以访问数据库。
- 在 1995 年以 `Personal Home Page Tools（PHP Tools）` 开始对外发表第一个版本，`Lerdorf` 写了一些介绍此程序的文档，并且发布了 `PHP1.0`。
- 在这个早期的版本中，只提供了像访客留言本、访客计数器等简单的功能。以后越来越多的网站使用了 `PHP`，并且强烈要求增加一些特性，比如循环语句和数组变量等。
- 现在是 `PHP 8` 的时代，2020 年 11 月 `PHP 8.0` 版本的发布取得了重大突破，同时将带来大幅的性能改进和新的特性，以及改进一些过时的功能。

1.2.2. PHP 的地位

- 在 `Web` 开发中 `PHP` 是王者，现在应用终端多方面发展，互联网用户爆发式增长。
- 如今不否认 `PHP` 在有些地方存在欠缺，比如：微服务的构建、常驻内存的服务级系统、密集计算、大数据的生态构建等。



- `PHP` 语言入门简单，容易掌握，程序健壮性好，不容易出现像 `Java`、`C++` 等其他语言那样复杂的问题。

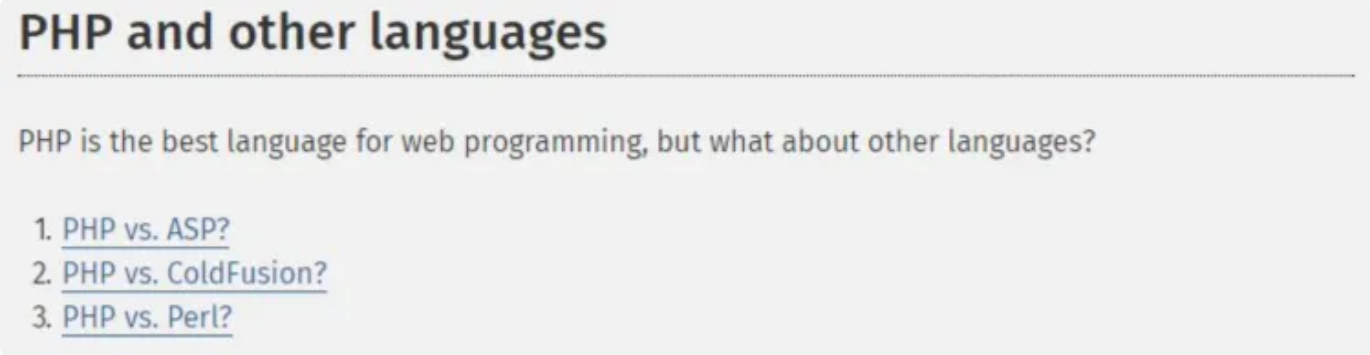
注：

- CTF 赛题：PHP 代码为主，代码审计，各种各样。
- FOFA 上的资产数：app="php"。

1.3. PHP 的梗

- `PHP` 是最好的语言，这个梗不是出自别处，而就是出自 `PHP` 的官方文档！

```
PHP is the best language for web programming, but what about other languages?
PHP 是网络编程最好的语言，但其他语言又怎样呢?
—— PHP and other languages
```



- 而这句话，最早出现在 2001 年 7 月的 `PHP` 文档中，不久后更激进的言论出现了。

```
Because PHP is the best language ever, ever. It's fast, very powerful, and free.
因为 PHP 是有史以来最好的语言，没有之一。它快速，非常强大，而且自由。
—— Project Beehive Forum
```

- `PHP` 官网地址：

https://www.php.net/

- `PHP` 官方在线手册：

https://www.php.net/manual/zh/

- PHP 官方手册下载：

https://www.php.net/download-docs.php

1.4. PHP 环境搭建

1.4.1. PhpStudy

- 官网地址：https://www.xp.cn/
- 简单，基本没有系统环境要求

1.4.2. BT（宝塔）

- 官网地址：https://www.bt.cn/download/linux.html
- 需要联网使用，一般用于云服务器上的环境搭建

1.4.3. wampserver/xmapp

- 官网地址：https://www.wampserver.com/en/
- 用于 `Windows`，图形化界面，个人感觉没有 `XP` 好用

1.4.4. Docker

- 官网地址：https://registry.hub.docker.com/search?q=lamp&type=image
- `Docker` 提供 `Linux` 环境下的 `PHP` 集成环境，一键启动，不受环境干扰，但无法任意切换环境

1.4.5. 本地/源码/程序安装

- 爱折腾的可以尝试搭建一下，但不建议用（吃饱了撑的）

2. PHP 基本语法

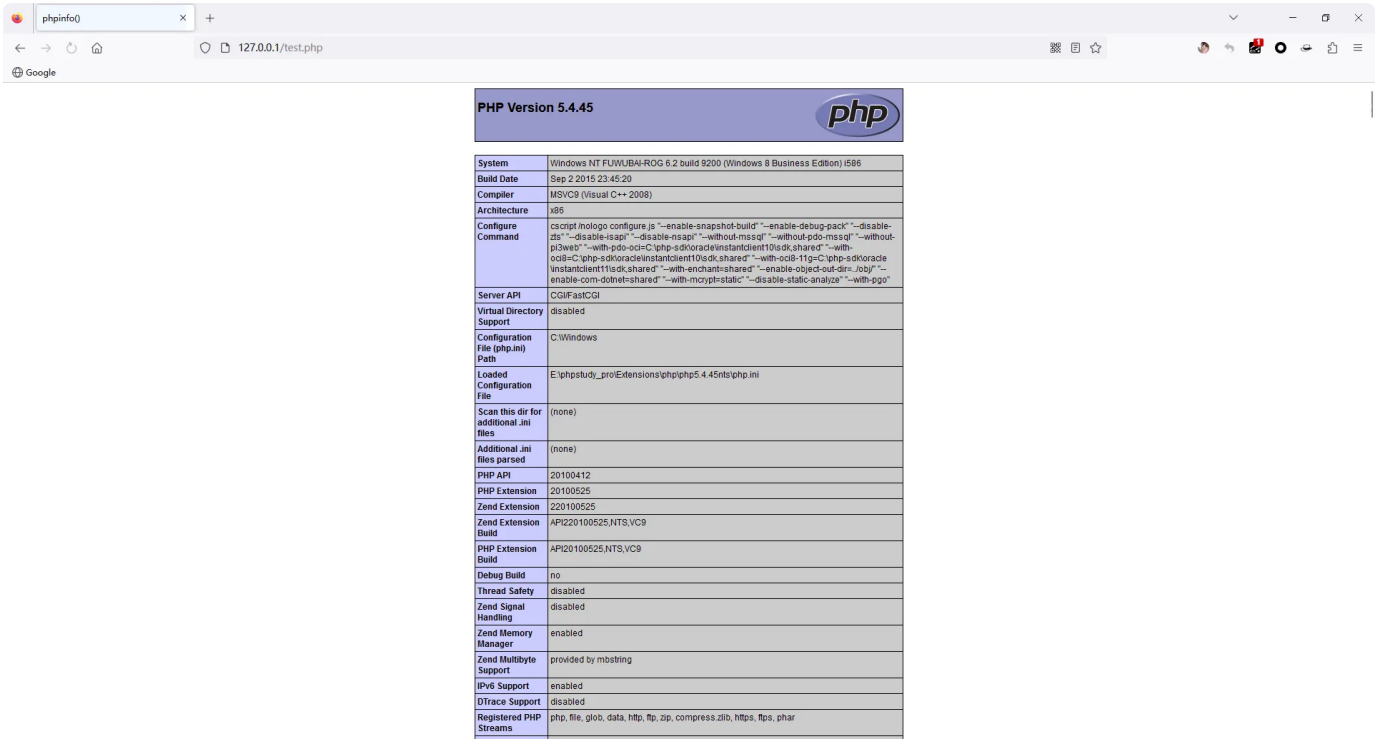
2.1. PHP 语法入门

2.1.1. 第一个 PHP 程序

- 新建一个 `txt` 文档，在里面写入如下代码：

```php
<?php phpinfo(); ?>
```

- 将上述代码保存，文件名改为 `test.php` （注意开启文件扩展名）。
- 启动 `PhpStudy`，在浏览器中输入 `http://127.0.0.1/test.php` 访问结果如下：



2.1.2. PHP 语言标记

- 通常情况 `PHP` 脚本以 `<?php 开头，以 ?> 结尾`，这是我们最常见的写法，其实标识脚本为 `PHP` 的方法有四种。

2.1.2.1. 标准标记 *

- 写法： `<?php Code ?>`

```php
<?php echo "Hello World"; ?>
```

- 标准标记是 `PHP` 最常用的标记类型，具有更好的兼容性、可移植性、可复用性。
- 提问： `;` 能不能不写
  - 可以， `?>` 起到结束的作用。
- 如果整个页面都是 `PHP` 代码， `PHP` 结束符时可以省略的（推荐）

2.1.2.2. 短标记 *

- 写法：`<? Code ?>`

```
<? echo "Hello World" ?>
```

- 短标记非常简单，但是使用短标记需要在配置文件 `php.ini` 中启用 `short_open_tag` 选项。
- 短标记在许多环境的默认设置中是不支持的，因此 `PHP` 不推荐使用这种标记。

2.1.2.3. ASP 标记（默认关闭）

- 写法：`<% Code %>`

```
<% echo "Hello World" %>
```

- `ASP` 标记与短标记类似，必须在配置文件 `php.ini` 中启用 `asp_tags` 选项。
- `ASP` 标记在许多环境的默认设置中是不支持的，因此 `PHP` 不推荐使用这种标记，且在 `PHP7` 以上该写法已废弃。

2.1.2.4. Script 标记

- 写法：`<script language="php"> Code </script>`
- `Script` 标记类似于 `javascript` 语言标记，由于 `PHP` 一般不推荐使用该标记，了解即可。

2.1.3. PHP 注释

- `PHP` 注释分为：
  - 单行注释
  - 多行注释

```
<?php
  // 单行注释
  # 单行注释
  /*
多行注释
*/
```

2.1.4. PHP 语句输出

- `echo`：输出，无返回值 *

```php
<?php
  echo "Hello World";
```

- `print`：输出，输出成功返回 1

```php
<?php
  print "Hello World";
echo print "Hello World";
```

- `print_r()`：输出数组 *

```php
<?php
  $arr = array('A'=>'a',123);
print_r($arr);
print_r($arr['A']);
```

- `var_dump()`：输出数据的详细信息，带有数据类型和数据长度 *

```php
<?php
  $arr = array('A'=>'a',123);
var_dump($arr);
```

2.2. 变量

- 变量是指在程序的运行过程中随时可以发生变化的量，是程序中数据的临时存放场所。
- 变量的**本质**就是内存中的一段空间。

2.2.1. 变量命名

- 变量必须以 `$` 开头，`$` 不是变量的一部分，仅表示后面的标识符是变量名。
- 除了 `$` 以外，以字母、下划线开头，后面跟着数字、字母、下划线。
- 变量名区分大小写，`$a` 和 `$A` 是两个变量。

2.2.2. 可变变量

- 变量名可以变，将变量名存储在另一个变量中：

```php
<?php
  $a = 10;
$b = 'a';
echo $$b;
```

2.2.3. 销毁/确认/检查变量

- 用 `unset()` 来销毁变量，销毁的是变量名，变量值由 `PHP` 垃圾回收机制销毁：

```php
<?php
  $name1 = 'yz';
$name2 = 'gok';
unset($name1);
echo $name1;
```

- 用 `isset()` 检测变量是否设置：

```php
<?php
  $name1 = 'yz';
$name2 = 'gok';
isset($name1);
echo $name1;
```

- 用 `empty()` 函数检查一个变量是否为空：

```php
<?php
  $name1 = '';
$name2 = 'gok';
echo empty($name1);
```

2.2.4.预定义变量（重中之重）

预定义变量又叫超全局变量，它们不需要提前声明就可以在所有的作用域中使用。通过这些预定义变量可以获取用户会话、用户操作系统的环境和本地操作系统的环境等信息。

常用的预定义变量如下所示：

- $GLOBALS：全局作用域中的全部可用变量；
- $_SERVER：服务器和执行环境的信息；
- $*REQUEST*：包含了* *$*GET，$*POST和$*COOKIE 的所有信息；
- $*POST：通过 POST 方法提交的数据；$a=$*POST['name']

- $_GET：通过 GET 方法提交的数据；
- $_FILES：通过 POST 方式上传到服务器的文件数据；
- $_ENV：通过环境方式传递给当前脚本的变量组成的数组；
- $_COOKIE：通过 HTTP Cookies 方式传递给当前脚本的变量所组成的数组；
- $_SESSION：当前脚本可用 SESSION 变量组成的数组。

```
<form method="get" action="./get.php">
  <input name="a" value="请输入" type="text">
  <button type="submit">提交</button>
  </form>
  <?php
  $a = $_GET['a'];
echo $a
?>
```

2.3. PHP 数据类型

- 数据类型分为两种：
  - 强类型
  - 弱类型
- `PHP` 和 `JavaScirpt` 一样都是弱类型的语言，变量的数据类型由右边的值类型来定义。
- 大部分数据类型，之前在 `JavaScript` 那都有讲过了，触类旁通这里不过多赘述，重点讲解一下：
  - 字符串
  - 数组

2.3.1. 字符串

- 在 `PHP` 中**单引号**字符串和**双引号**字符串是由区别的。
  - 单引号字符串是**真正的字符串**。
  - 双引号字符串要**解析字符串中的变量**。
- 示例：

```php
<?php
  $a = 'hello';

echo '$a';
echo "$a";
```

- 以下代码怎么输出?

```php
<?php
  $a = 'hello';

echo "$aworld";
echo "$a world";
```

- 两种方式进行变量拼接:

```php
<?php
  $a = 'hello';

echo "$a".'world';
echo "{$a}world";
```

2.3.2. 数组

- 在 `PHP` 中数组有两种形式:
  - 索引数组:用整数做下标,默认从 0 开始,后面一次加一。
  - 关联数组:用字符串做下标,通过 `=>` 符号将下标和值关联起来。
- 示例如下:

```php
<?php
  //1. 索引数组的声明
  $arr1 = array('tom', 'perry', 'ketty'); //索引数组
print_r($arr1); // 输出数组 Array ( [0] => tom [1] => berry [2] => ketty )

//2、关联数组
$arr2 = array('name' => '李白', 'sex' => '男', 'age' => 22);
print_r($arr2); //Array ( [name] => 李白 [sex] => 男 [age] => 22 )
```

- 数组的声明

```
直接赋值的方式声明数组
$数组变量名[下标] = 值
  $array[0] = 'GOK';
$array[1] = 'PHP 学习';
$array[2] = 'PHP 数组'

  使用 array() 函数声明数组
$数组变量名 = array(key1 => value1, key2 => value2, ..., keyN => valueN);
$array = array(0 => 'GOK', 1 => 'PHP 学习', 2 => 'PHP 数组');
```

- 可以使用 `foreach` 遍历数组：

```
<?php
  $arr = array('tom', 'berry', 'ketty');
foreach ($arr as $key => $value) {
  echo $key . '=>' . $value;
}
```

2.4. PHP 运算符/表达式

- 之前在 `JavaScript` 那都有讲过了，触类旁通这里不过多赘述。

3. PHP 流程控制

- 流程控制（`if`、`switch`、`while` 等），之前在 `JavaScript` 那都有讲过了，触类旁通这里不过多赘述。

```
if (判断条件) {
语句块 1;
} else {
语句块 2;
}
```

4. PHP 函数应用

- 函数就是一段代码块
- 函数可以实现模块化编程

4.1. 函数声明

- 在 `PHP` 中声明一个自定义的函数可以使用下面的语法格式：

```
<?php
  function 函数名（参数 1,参数 2,…,参数 n）{
  函数体;
  return 返回值;
  }
```

- 通过 `函数名()` 调用函数：

```
<?php
  function show(a,b)
{
  echo '锄禾日当午<br>';
  }
  show();
SHOW();
```

4.2. 可变函数

- 将函数名存储到变量中：

```
<?php
  function show($args)
{
  echo $args, '<br>';
}
$str = 'show';
$str('锄禾日当午');
```

4.3. 匿名函数

- 匿名函数就是没有名字的函数：

```
<?php
  $fun = function () {
  echo '锄禾日当午<br>';
  };
$fun();
```

4.4. 函数的参数

- 函数的参数有
  - 形参：定义函数时候的参数，只起形式作用，没有具体的值
  - 实参：调用函数时候的参数，有具体的值

```php
<?php
  //这里的$num1, $num2为形式参数
  function fun($num1, $num2)
{
  echo $num1 + $num2;
}
fun(10, 20);
```

4.4.1. 参数默认值

- 在定义函数的时候给形参赋值就是参数的默认值：

```php
<?php
  function fun($name, $add = '地址不详')
{
  echo '姓名：' . $name, '<br>';
echo '地址：' . $add, '<hr>';
}
fun('tom', '北京');
fun('berry');
```

- 默认值必须是值，不能用变量代替。

4.4.2. 参数个数不匹配

- 当参数不匹配时，默认往前取值：

```php
<?php
  //这里的$num1, $num2为形参
  function fun($num1, $num2)
{
  echo $num1, '<br>';
echo $num2, '<br>';
}
fun(10, 20, 30);
//这里的10，20为实参
```

4.5. header() 函数

- `header()` 函数向客户端发送原始的 `HTTP` 报头。

注：最好在任何实际的输出被发送之前调用 header() 函数。

- 掌握以下两种：
  - 页面跳转

```php
<?php
  header("location:https://www.baidu.com");
```

  - 页面编码

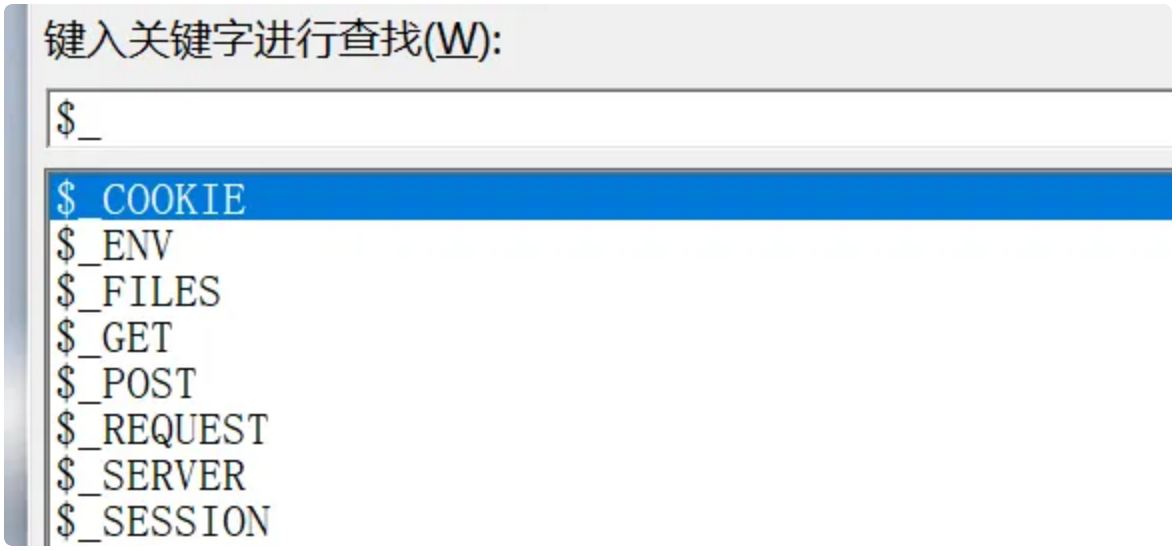
```php
<?php
  header("Content-Type:text/html;charset=utf8");
echo "国科";
```

5. 变量作用域

- 局部变量：在函数里面，默认情况下函数内部是不会去访问函数外部的变量。函数外面也无法调用里面的变量。
- 全局变量：在函数外面。

```php
<?php
  $num1 = 10;
function fun()
{
  $num2 = 20;
  echo $num1;
  echo $num2;
}
fun();
echo $num2;
```

- 超全局变量：可以在函数内部和函数外部访问




```php
<?php
  $_GET['num1'] = 10;
function fun()
{
  $_GET['num2'] = 20;
  echo $_GET['num1'];
}
fun();
echo $_GET['num2'];
```

- 还有一种方式可以使函数内部访问外部：全局变量 – `$GLOBALS`

```php
<?php
  $num1 = 10;
function fun()
{
  echo $GLOBALS[num1];
}
fun();
```

6. 函数返回值

- `PHP` 函数中 `return` 作用如下：
  - 终止脚本执行：中断 `return` 之后的代码执行。

```php
<?php
  echo '锄禾日当午<br>';
return;
echo '汗滴禾下土<br>';
```

  - 返回结果，需要赋予一个返回值。

```php
<?php
  function add($num)
{
  return $num * 2;
}
echo add(100);
```

7. PHP 面向对象

- `PHP5` 以上版本的最大特点是引入了面向对象的全部机制，保留了向下兼容性。

7.1. 面向过程 & 面向对象

- 面向过程和面向对象都是对软件分析、设计和开发的一种思想，它指导着人们以不同的方式去分析、设计和开发软件。
- 早期先有面向过程思想，随着软件规模不断的扩大，问题的复杂性的提高，面向过程的弊端越来越明显，随即出现了面向对象的思想并成为目前主流的模式。

- 面向过程思想思考问题时，我们首先思考怎么按步骤实现？，并将步骤对应成方法，一步一步最终完成。
- 这个适合简单任务，不需要过多协作的情况下。
    - 比如：如何开车?
        - 插钥匙
        - 发动
        - 挂挡
        - 油门
        - 走你
- 面对过程适合简单，不需要协作的事务。但是当我们思考比较复杂的问题，比如：如何造车?就会发现列出步骤是不太现实的。正是应为这样，造车太复杂，需要很多协作才能完成，此时面向对象思想就营运而生了。
- 面向对象思想思考造车，发现车由如下对象组成：
    - 轮胎
    - 发动机
    - 车架
    - 玻璃
    - ……
- 为了便于协作，我们从各大零件工厂进行材料采购，各大工厂同时进行车的制造，最终进行组装，大大提高了效率。但是，具体到工厂的零件生产时，仍是有步骤的，最终离不开面向过程思想。
- 千万不要把面向过程和面向对象两者进行对立，他们是相辅相成的，面向对象离不开面向过程！
- 总结：
    - 两者都是解决问题的思维方式，都是代码组织的方式。
    - 解决简单问题可以使用面向过程。
    - 解决复杂问题：宏观上使用面向过程把控，微观上使用面向过程

7.2. 抽象一个类

- 面向对象程序的单位就是对象，但对象又通过类的实例化出来，所以我们首先要做的就是如何来声明类。

7.2.1. 类的声明

- 类的声明非常简单，和函数的声明比较类似。只需要使用一个关键字 `class`，后面加上一个自定义的类别名称，并加上一对花括号就可以了。有时也需要在 `class` 关键字的前面加一些修饰类的关键字，例如 `abstract` 或 `final` 等。类的声明格式如下：

```
[ 一些修饰类的关键字 ] class 类名 {
  类中成员;
}
```

7.2.2. 成员属性

- 在类中直接声明变量就称为成员属性，可以在类中声明多个变量，即对象中有多个成员属性，每个变量都存储对象不同的属性信息。
- 示例如下：定义个人

```
<?php
  class Person{
  var $name;
  var $age;
var $sex;
function(){
  id
    }
    b
    }
  class qwe
```

- 关键字修饰符：

修饰符	说明
public	用于修饰成员变量或方法，表示它们可以在任何地方被访问

private	用于修饰成员变量或方法，表示它们只能在所属类的内部访问，无法在类的外部或子类中直接访问
protected	用于修饰成员变量或方法，表示它们只能在所属类的内部以及其子类中访问，无法在类的外部直接访问
static	用于修饰成员变量或方法，表示它们属于类本身而不是实例化对象，可以直接通过类名访问，而无需创建对象
var	在 PHP 5 中已经被废弃，并且在 PHP 7 中已经移除

- 示例如下：

```
<?php
  class Person{
  public $name;
private $age;
protected $sex;
}
```

7.2.3. 成员方法

- 在对象中需要声明可以操作本对象成员属性的一些方法来完成对象的一些行为。在类中直接声明的函数就称为成员方法，可以在类中声明多个函数，对象中就有多个成员方法。
- 示例如下：

```php
<?php

  class zhuc{
  var $name;
  var $age;
var $sex;

function say(){
  echo "你真是个帅比";
}
}
```

7.3. 实例化对象

- 面向对象程序的单位就是对象，但对象又是通过类的实例化产生出来的。

7.3.1. 实例化对象

- 将类实例化成对象非常容易，只需要使用 `new` 关键字并在后面加上一个和类名同名的方法即可。

```php
<?php
  class Person
{
  var $name;
  var $age;
var $sex;

function say()
  {
    echo "你真是个帅比";
  }
}



$xiaoming = new Person();
```

7.3.2. 成员访问

- 对象中包含成员属性和成员方法，访问对象中的成员则包括成员属性的访问和成员方法的访问。而对成员属性的访问又包括赋值操作和获取成员属性值的操作。
- 访问对象中的成员和访问数组中的元素类似，只能通过对象的引用来访问对象中的每个成员。但还要使用一个特殊的运算符号 `->` 来完成对象成员的访问。

```php
<?php
  class Person
{
  var $name;
  var $age;
var $sex;

function say()
  {
    echo "你真是个帅比";
  }
}
$xiaoming = new Person();
$xiaoming->say();
echo $xiaoming->name;
$xiaoming->name = '葫芦娃';
echo $xiaoming->name;
```

7.3.3. 特殊的引用 this

- 对象一旦被创建，在对象中的每个成员方法里都会存在一个特殊的对象引用 `$this` 。
- 成员方法属于哪个对象， `$this` 引用就代表哪个对象，专门用来完成对象内部成员之间的访问。
- 示例：没有 `this` 的情况

```php
<?php
  class Person
{
  var $name;
  var $age;
var $sex;

function say($name)
  {
    echo $name . "你真是个帅比";
  }
}
$xiaoming = new Person();
$xiaoming->say('葫芦娃');
echo $xiaoming->name;
```

- 示例：使用 `this` 赋值

```php
<?php
  class Person
{
  var $name;
  var $age;
var $sex;

function say($name)
  {
    $this->name = $name;
    echo $this->name . "你真是个帅比";
  }
}
$xiaoming = new Person();
$xiaoming->say('葫芦娃');
echo $xiaoming->name;
```

8. 会话与权限管理

8.1. Cookie 与 Session

8.1.1. Cookie

- `Cookie` 的工作机制是用户识别及状态管理， 用来管理服务器和客户之间的状态，由服务器生成保存在客户端的数据载体，用于会话跟踪。
- `Web` 网站为了管理用户的状态会通过 `Web` 浏览器，把一些数据临时写入用户的计算机内。
- 接着当用户访问该 `Web` 网站时，可通过通信方式取回之前发放的 `Cookie` 。
- `Cookie` 比喻成身份信息，国家没有保留你的身份信息，你的身份信息你自己带着， `Cookie` 就相当于你的身份证，去哪都要出示身份证，证明你是你。

8.1.2. Session

- `Session` 会在 `Cookie` 中出现和传输，属于 `Cookie` 的一部分，由服务器生成，保存在服务器。
- 当 `Web` 服务器开启会话机制时，用户登陆成功后，会将会话保存在服务器中。
- `Session` 比喻成银行卡，你的钱在银行，你拿只是一张卡，你可以随时去银行取，因为钱属于你。

8.2. PHP Cookie

8.2.1. setcookie()

- `setcookie()` 函数向客户端发送一个 `HTTP cookie`， `cookie` 的名称自动指定为相同名称的变量。
- 例如，如果被发送的 `cookie` 名为 `user`，则会自动创建一个名为 `$user` 的变量，包含 `cookie` 的值。，必须在任何其他输出发送到客户端前对 `cookie` 进行赋值。
- 语法：

```
1   setcookie(name,value,expire,path,domain,secure)
```

参数	描述
name	必需。规定 cookie 的名称。
value	必需。规定 cookie 的值。

expire	可选。规定 cookie 的过期时间。time()+3600 24 30 将设置 cookie 的过期时间为 30 天。如果这个参数没有设置，那么 cookie 将在 session 结束后（即浏览器关闭时）自动失效。
.....	.....

- 示例：

```php
<?php
  $value = "my cookie value";
setcookie("NAME", 'TOM');
<?php
  $value = "my cookie value";
setcookie("TestCookie", $value, time() + 3600);
```

- 删除 Cookie：

```php
<?php
  $value = "my cookie value";
setcookie("TestCookie", $value, time() - 3600);
```

8.2.2. $_COOKIE

- `PHP` 的 `$_COOKIE` 变量用于取回 `cookie` 的值。
- 示例：

```php
<?php
  $value = "my cookie value";
setcookie("TestCookie", $value, time() + 3600);
echo $_COOKIE['TestCookie'];
```

- 可以使用 `isset()` 函数确认 `Cookie` 是否存在：

```php
<?php
  $value = $_POST['name'];
setcookie("TestCookie1", $value, time() + 3600);
if (isset($_COOKIE['TestCookie'])) {
  echo $_COOKIE['TestCookie'];
} else {
  echo "Cookie not exist!";
}
```

8.3. PHP Session

8.3.1. session_start()

- 在您把用户信息存储到 `PHP session` 中之前，首先必须启动会话。

注：session_start() 函数必须位于 `<html>` 标签之前。

```php
<?php
  session_start();
```

- 上面的代码会向服务器注册用户的会话，以便您可以开始保存用户信息，同时会为用户会话分配一个 `UID` 。

8.3.2. $_SESSION

- `PHP` 的 `$_SESSION` 变量用于存储和取回 `session` 的内容。
- 示例：

```php
<?php
  session_start();
$_SESSION['username'] = 'admin';
echo $_SESSION['username'];
```

8.3.3. 删除 Session

- 如果需要删除某些 `session` 数据，可以使用 `unset()` 或 `session_destroy()` 函数。
- `unset()` 函数用于释放指定的 `session` 变量：

```php
<?php
  session_start();
$_SESSION['username1'] = 'admin1';
$_SESSION['username2'] = 'admin2';
unset($_SESSION['username1']);
echo $_SESSION['username1'];
echo $_SESSION['username2'];
```

- `session_destroy()` 将重置 `session`，将失去所有已存储的 `session` 数据：

```php
<?php
  session_start();
$_SESSION['username1'] = 'admin1';
$_SESSION['username2'] = 'admin2';
session_destroy();
echo $_SESSION['username1'];
echo $_SESSION['username2'];
```

9. 一个小实验

- 题目地址：https://ctf.bugku.com/challenges/detail/id/70.html
- 题目地址：https://ctf.bugku.com/challenges/detail/id/71.html

JS作业，提取码：GOKT

百度网盘为您提供文件的网络备份、同步和分享服务。空间大、速度快、安全稳固，支持教育网加速，...

https://pan.baidu.com/s/1hhvzR1cxuE3evwW-U5qjcw