PHP

> 详细参考文档：http://php.net/manual/zh/index.php

1. 起步

> - PHP 是什么？
> - PHP 写在哪？
> - PHP 能做啥？

> 超文本标记是用普通文本描述富文本的一种方式

PHP（PHP: Hypertext Preprocessor）是一种被广泛应用的脚本语言，它可以被嵌入到 HTML中，尤其适合做动态网站开发开发。

我们接下来会在 PHP 中看到的许多代码特性和其他编程语言类似，例如：变量、函数、循环，等等。 代码语法看起来不同，但是在概念上是基本类似的。

我们使用 PHP 的目的就是能让静态网页变成动态网页，能称之为动态网页的核心就是让 HTML 上的内容不再被写死，而是通过在 HTML 中嵌入一段可以在服务端执行的代码，从而达到动态网页的目标。

例如：我们需要有一个网页，这个网页每次打开都可显示当前的年月日，如果采用 HTML 处理：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>当前日期</title>
</head>
<body>
  <h1>2020-01-01</h1>
</body>
</html>
```

我们必须每天到服务器上修改这个网页，从而让它保持显示最新日期，但是有了 PHP 这种能够在服务端执行的脚本语言就可以很轻松实现：

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>当前日期</title>
</head>
<body>
  <h1><?php echo date('Y-m-d'); ?></h1>
</body>
</html>
```

从以上这个最最简单的基础案例就能看出：PHP 无外乎为了可以在网页中动态输出最新内容的一种技术手段。

> 历史使人明智：http://php.net/manual/zh/history.php.php

1.1. PHP 标记

> http://php.net/manual/zh/language.basic-syntax.phpmode.php

- `<?php` 可以让代码进入“PHP 模式”
- `?>` 可以让代码退出“PHP 模式”

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>这是一个包含 PHP 脚本的网页</title>
</head>
<body>
  <h1>这是一个包含 PHP 脚本的网页</h1>
  <p>这里原封不动的输出</p>

  <?php
  // 这里是 PHP 代码，必须满足 PHP 语法
  $foo = 'bar';
  echo $foo;
  ?>

  <p>这里也不变</p>

  <p><?php echo '<b>这还是 PHP 输出的</b>'; ?></p>
</body>
</html>
```

类似于在 HTML 中使用 JavaScript，但是不同的是 JavaScript 运行在客户端，而 PHP 运行在服务端。

只有处于 PHP 标记内部的代码才是 PHP 代码，PHP 标记以外都原封不动。

1.1.1. 省略结束标记

如果 PHP 代码段处于整个文件的末尾，建议（必须）删除结束标记，这样不会有额外的空行产生。

1.2. 输出内容方式

- echo：

```php
<?php
// echo 是 PHP 中的一个特殊的“指令”，
// 不一定需要像函数那样通过 `()` 去使用
// 注意：echo 后面紧跟着一个空格
echo 'hello php';
echo 'hello', 'world';
// => `helloworld`
```

- print：

```php
<?php
// print 与 echo 唯一区别就是只能有一个参数
print 'hello php';
// print 'hello', 'world';
// => Parse error: syntax error ...
```

- var_dump：

```php
<?php
// var_dump 是一个函数，必须跟上 () 调用
// 可以将数据以及数据的类型打印为特定格式
var_dump('hello php');
// => 'string(9) "hello php"'
```

> 还有一些输出函数（可以通过查手册自学，用到再说），例如：`exit()` / `print_r()` 等等

1.3. 与 HTML 混编

- 普通嵌入

```
<p><?php echo 'hello'; ?></p>
```

- 语句混编

```
<?php if ($age >= 18) { ?>
  <p>成年人</p>
<?php } else { ?>
  <p>小朋友</p>
<?php } ?>
```

更常见的用法：

```
<?php if ($age > 18): ?>
  <p>成年人</p>
<?php else: ?>
  <p>小朋友</p>
<?php endif ?>
```

1.4. 注释

你可以在代码中添加注释，从而增强我们代码的可阅读性。PHP 中注释有两种方式（与 JavaScript 相同）：

- 单行注释

```
<?php
// 这是一条单行注释
# 井号也可以做注释（不要用，有点生僻）
$foo = 'hello';
```

- 多行注释

```
<?php
/*
......
这里可以添加任意行数的注释内容
......
*/
$foo = 'hello';
```

2. 语法

编程语言常见的语法

- 变量 —— 用于临时存放数据的容器
- 顺序结构 —— 先干什么再干什么
- 分支结构 —— 如果怎样就怎样否则怎样
- 循环结构 —— 不断的做某件相同的事
- 函数 —— 提前设计好一件事怎么干，然后想什么时候干就什么时候干
- 运算符 —— 数学运算和字符串拼接
- 字面量 —— 在代码中用某些字符组成，能够表达一个具体的值 这些字符之间表示数据的方式叫做字面量

PHP 也是另外种编程语言，作为另外一种编程语言，PHP 也具备着绝大多数语言都有的特点，例如变量、条件分支、循环、函数等等，不同的是每个语言都会有自己的语法规定。这里不用太过担心，这些语法规定与之前学习的编程语言大同小异，对我们来说不会有太大的压力。

1. 变量
2. 双引号字符串和单引号字符串的差异
3. 指令式的语法
4. foreach
5. 函数作用域问题
6. 字符串拼接

2.1. 变量

> 变量是编程语言中临时存放数据的容器。

PHP 中申明一个变量是用一个美元符号后面跟变量名来表示。变量名同样是区分大小写的。

PHP 中变量无需声明类型，变量的类型根据值的类型来推断。

```php
<?php
$foo; // 申明一个变量，变量名为 `foo`，未对其进行赋值
$bar = 'baz'; // 申明一个变量，将一个值为 `baz` 的字符串赋值给它
echo $foo; // 输出一个变量名为 `foo` 的变量
fn($bar); // 将一个变量名为 `foo` 的变量作为 `fn` 的实参传递
```

2.1.1. 数据类型

常见的 PHP 数据类型与 JavaScript 基本一致：

- string（字符串）

- integer（整型）—— 只能存整数
- float（浮点型）—— 可以存带小数位的数字
- boolean（布尔型）
- array（数组）
- object（对象）
- NULL（空）
- Resource（资源类型）
- Callback / Callable（回调或者叫可调用类型）

字符串

PHP 有多种创建字符串的方式：单引号、双引号等。

- 单引号字符串
  - 不支持特殊的转义符号，例如 `\n`
  - 如果要表示一个单引号字符内容，可以通过 `\'` 表达
  - 如果要表示一个反斜线字符内容，可以通过 `\\` 表达
- 双引号字符串
  - 支持转义符号
  - 支持变量解析

```php
<?php
// ====== 单引号 ======
echo 'hello\nworld';
// => `hello\nworld`
echo 'I\'m a better man';
// => `I'm a better man`
echo 'OS path: C:\\Windows';
// => `OS path: C:\Windows`

// ====== 双引号 ======
echo "hello\nworld";
// => `hello
// world`
$name = 'zce';
echo "hello $name";
// => `hello zce`
```

> 字符串函数
>
> - http://php.net/manual/zh/ref.strings.php
> - http://www.w3school.com.cn/php/php_string.asp

数组

PHP 中数组可以分为两类：

- 索引数组

  与 JavaScript 中的数组基本一致

```php
<?php
// 定义一个索引数组
$arr = array(1, 2, 3, 4, 5);
var_dump($arr);

// PHP 5.4 以后定义的方式可以用 `[]`
$arr2 = [1, 2, 3, 4, 5];
var_dump($arr2);
```

- 关联数组

  有点类似于 JavaScript 中的对象

```php
<?php
// 注意：键只能是`integer`或者`string`
$arr = array('key1' => 'value1', 'key2' => 'value2');
var_dump($arr);

// PHP 5.4 以后定义的方式可以用 `[]`
$arr2 = ['key1' => 'value1', 'key2' => 'value2'];
var_dump($arr2);
```

2.1.2. 数据类型转换

> 参考：http://php.net/manual/zh/language.types.type-juggling.php

```php
<?php
$str = '132';
// 将一个内容为数字的字符串强制转换为一个整形的数字
$num = (int)$str;
// 将一个数字强制转换为布尔值
$flag = (bool)$num;
```

2.2. 运算符

数学运算符，逻辑运算符与 JavaScript 基本一致，无额外特殊情况。

字符串连接（拼接）采用的是比较特殊的 `.`

```php
<?php
$name = 'zce';
// 拼接 `hey ` 和 `zce`
$message = 'hey ' . $name;
// 相当于 +=
$foo .= $message
```

2.3. 语句

- 分号分割
- if、switch、while、for、 foreach、function……

2.4. 流程控制

- 顺序结构
- 分支结构
  - if ... else
  - switch ... case
- 循环结构
  - for
  - while
  - foreach --- 专门用来遍历数组

```php
<?php
$arr = array('name' => 'zhangsan', 'age' => '18');

foreach ($arr as $key => $value) {
  echo $key . ' ' . $value;
}
```

指令式的 if、for、foreach、while 单独掌握

```php
<?php
// 指令式就是将开始 { 换成 :  结束 } 换成 endif;
if ($i > 0) :
  echo 'ok'
endif;

// for foreach while 也是一样
for ($i = 0; $i < 10; $i++) :
  echo $i;
endfor;
```

2.5. 函数

定义与使用函数的方式与 JavaScript 相同：

```php
<?php
// 函数名不区分大小写
function foo ($name, $title) {
  echo "$name ($title)";
}

// 调用
foo('zce', 'UFO');
Foo('zgd', 'SBO'); // 大小写不区分
```

注意：使用方式有点差异（函数名不区分大小写），但是不要这么搞！！！

> 建议在 PHP 中采用下划线式（snake_case）做命名规则，不管是函数还是变量

3. 特性

3.1. 变量作用域

关于变量作用域这一点，PHP 与绝大多数语言也都不同：**默认函数内不能访问函数所在作用域的成员。**

在 JavaScript 中，我们可以在函数作用域中使用父级作用域中的成员：

```
var top = 'top variable'

function foo () {
  var sub = 'sub variable'

  console.log(top)
  // => `top variable`

  function bar () {
    console.log(top)
    // => `top variable`
    console.log(sub)
    // => `sub variable`
  }

  bar()
}

foo()
```

而在 PHP 中：

```
<?php
$top = 'top variable';

function foo () {
  $sub = 'sub variable';

  echo $top;
  // => 无法拿到

  function bar () {
    echo $top;
    // => 无法拿到

    echo $sub;
    // => 无法拿到
  }

  bar();
}

foo();
```

如果需要访问全局变量，可以通过 `global` 关键字声明：

```php
<?php
$top = 'top variable';

function foo () {
  // 声明在当前作用域中获取全局作用域中的 `$top`
  global $top;

  $sub = 'sub variable';

  echo $top;
  // => `top variable`

  function bar () {
    // 声明在当前作用域中获取全局作用域中的 `$top` 和 `$bar`
    global $top, $bar;

    echo $top;
    // => `top variable`

    echo $sub;
    // => 任然无法拿到，因为 `$sub` 不再全局范围，而是在 `foo` 函数中定义
  }

  bar();
}

foo();
```

3.2. 超全局变量

> http://www.w3school.com.cn/php/php_superglobals.asp

PHP 中的许多预定义变量都是“超全局的”，这意味着它们在一个脚本的全部作用域中都可用。在函数或方法中无需执行 global $variable; 就可以访问它们。

这些超全局变量是：

- $GLOBALS — 引用全局作用域中可用的全部变量　　类似JS中的window
- $_SERVER — 获取服务端相关信息
- $_REQUEST — 获取提交参数
- $_POST — 获取 POST 提交参数
- $_GET — 获取 GET 提交参数
- $_FILES — 获取上传文件
- $_ENV — 操作环境变量

- $_COOKIE — 操作 Cookie
- $_SESSION — 操作 Session

本节会介绍一些超全局变量，并会在稍后的章节讲解其他的超全局变量。

3.2.1. $GLOBALS

$GLOBALS 这种全局变量用于在 PHP 脚本中的任意位置访问全局变量（从函数或方法中均可）。

PHP 在名为 $GLOBALS[index] 的数组中存储了所有全局变量。变量的名字就是数组的键。

下面的例子展示了如何使用超级全局变量 $GLOBALS：

```php
<?php
$x = 75;
$y = 25;

function foo () {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

foo();
echo $z;
// => 100
```

3.3. 常量定义与使用

> 常量跟变量一样也是一个数据容器，但是不同的是一旦申明过后就不允许被修改。

3.3.1. 定义常量

```php
<?php
// 定义常量使用的是内置的 `define` 函数
// 第一个参数是常量的名称，建议采用全大写字母命名，多个单词下划线分隔
// 第二个参数是常量中存放的数据，可以是任意类型
// 第三个参数是常量名称是否区不分大小写，默认 false 区分大小写
define('SYSTEM_NAME', '阿里百秀');
define('SYSTEM_ENABLE', true);
```

3.3.2. 使用常量

```php
<?php
// 直接通过常量的名称访问常量
// 与变量不同的是不需要用 $
echo SYSTEM_NAME;
echo SYSTEM_ENABLE;
```

3.4. 载入其他文件

通常情况下，当一个文件中的代码过长，自然会想到要拆分到多个文件中。随着开发经验的积累，慢慢的会发现，除了文件过程应该拆分文件，更应该做的事情是根据用途去划分。

不管你是怎样想的，核心问题都是一样：怎么将代码拆分到多个文件中？

PHP 中引入其他 PHP 文件有四种方式：

- require
- require_once
- include
- include_once

四种方式的对比：

	require	require_once	include	include_once
被载入文件如果不存在是否影响继续运行	Y	Y	N	N
多次调用是否会重复执行被载入的文件	Y	N	Y	N

总结来说：

- 横向分为两类：require 和 include 两种，区别在于 require 会因为载入文件不存在而停止当前文件执行，而 include 不会。
- 纵向分为两类：xxx 和 xxx_once，区别在于代码中每使用一次 xxx 就执行一次载入的文件，而 xxx_once 只会在第一次使用是执行。

使用层面：

- include 一般用于载入公共文件，这个文件的存在与否不能影响程序后面的运行
- require 用于载入不可缺失的文件
- 至于是否采用一次载入（once）这种方式取决于被载入的文件

4. 常用 API

> 任何编程语言本身并没有太多的能力，具体的能力大多数都来源于 API。

PHP 的能力来源于它有 1000+ 内置函数，不是每一个函数都默认直接可以使用，有一些需要安装或者启用额外的"插件" 扩展

4.1. 字符串处理

> 宽字符集需要开启 php_mbstring 扩展

4.1.1. 开启 PHP 扩展

1. 将PHP目录中的 php.ini-development 复制一个 修改为 php.ini
2. 修改扩展文件所在目录 extension_dir
3. 修改文件中的部分选项（; 是注释符）
4. 在 Apache 配置文件中申明一下 php.ini 的所在目录

4.1.2. 字符串处理函数

- 字符串截取
  - `string substr ( string $string , int $start [, int $length ] )`
  - `string mb_substr ( string $str , int $start [, int $length = NULL [, string $encoding = mb_internal_encoding() ]] )`
- 字符串长度
  - `int strlen ( string $string )`
  - `mixed mb_strlen ( string $str [, string $encoding = mb_internal_encoding() ] )`
- 大小写转换
  - `string strtolower ( string $string )`
  - `string strtoupper ( string $string )`
- 去除首尾空白字符
  - `string trim ( string $str [, string $character_mask = " \t\n\r\0\x0B" ] )`
  - `string ltrim ( string $str [, string $character_mask ] )`
  - `string rtrim ( string $str [, string $character_mask ] )`
- 查找字符串中某些字符首次出现位置
  - `mixed strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )`
  - `int mb_strpos ( string $haystack , string $needle [, int $offset = 0 [, string $encoding = mb_internal_encoding() ]] )`

- 字符串替换
  - `mixed str_replace ( mixed $search , mixed $replace , mixed $subject [, int &$count ] )`
- 重复字符串
  - `string str_repeat ( string $input , int $multiplier )`
- 字符串分割
  - `array explode( string $input, string $char )`

```
Interactive shell

php > substr('1234567', 3, 2);
php > ehco substr('1234567', 3, 2);
PHP Parse error:  syntax error, unexpected 'substr' (T_STRING) in php shell code on line 1

Parse error: syntax error, unexpected 'substr' (T_STRING) in php shell code on line 1
php > echo substr('1234567', 3, 2);
45
php > echo strtolower('DSFSDF');
dsfsdf
php > echo trim('  sdfsdf  ');
sdfsdf
php > echo trim('  sdfsdf  ', 'a');
  sdfsdf
php > echo trim('111sdfsdf111', '1');
sdfsdf
php > echo trim('   111abc111', '1');
   111abc
php > echo trim('   111abc111 ', '1');
   111abc111
php > echo strpos('12345671234567', '5');
4
php > echo strpos('12345671234567', '5', 5);
11
php > echo str_replace('a', 'b', '123abcabc');
123bbcbbc
php >
```

4.2. 数组处理

- 获取关联数组中全部的键 / 值
  - `array_keys() / array_values()`
- 判断关联数组中是否存在某个键
  - `array_key_exists()`
- 去除重复的元素
  - `array_unique()`
- 将一个或多个元素追加到数组中
  - `array_push()`
  - `$arr[] = 'new value'`
- 删除数组中最后一个元素
  - `array_pop()`

- 数组长度
  - `count()`
- 检测存在
  - `in_array()`

· 获取元素在数组中的下标
· array_search()

4.3. 时间处理

- 时间戳：`time()`
  - 从 Unix 纪元（格林威治时间 1970-01-01 00:00:00）到当前时间的秒数
- 格式化日期：`date()`
  - 获取有格式的当前时间
  - 格式化一个指定的时间戳
  - 可以通过 `strtotime()` 将有格式的时间字符串转换为时间戳

4.4. 文件操作

函数	描述	PHP
basename()	返回路径中的文件名部分。	3
copy()	复制文件。	3
dirname()	返回路径中的目录名称部分。	3
disk_free_space()	返回目录的可用空间。	4
disk_total_space()	返回一个目录的磁盘总容量。	4
fclose()	关闭打开的文件。	3
file()	把文件读入一个数组中。	3
file_exists()	检查文件或目录是否存在。	3
file_get_contents()	将文件读入字符串。	4
file_put_contents()	将字符串写入文件。	5
filesize()	返回文件大小。	3
fopen()	打开一个文件或 URL。	3
glob()	返回一个包含匹配指定模式的文件名/目录的数组。	4
is_dir()	判断指定的文件名是否是一个目录。	3
is_file()	判断指定文件是否为常规的文件。	3
mkdir()	创建目录。	3
move_uploaded_file()	将上传的文件移动到新位置。	4
pathinfo()	返回关于文件路径的信息。	4
rename()	重名名文件或目录。	3
rmdir()	删除空的目录。	3
unlink()	删除文件。	3

> 参考：http://www.w3school.com.cn/php/php_ref_filesystem.asp