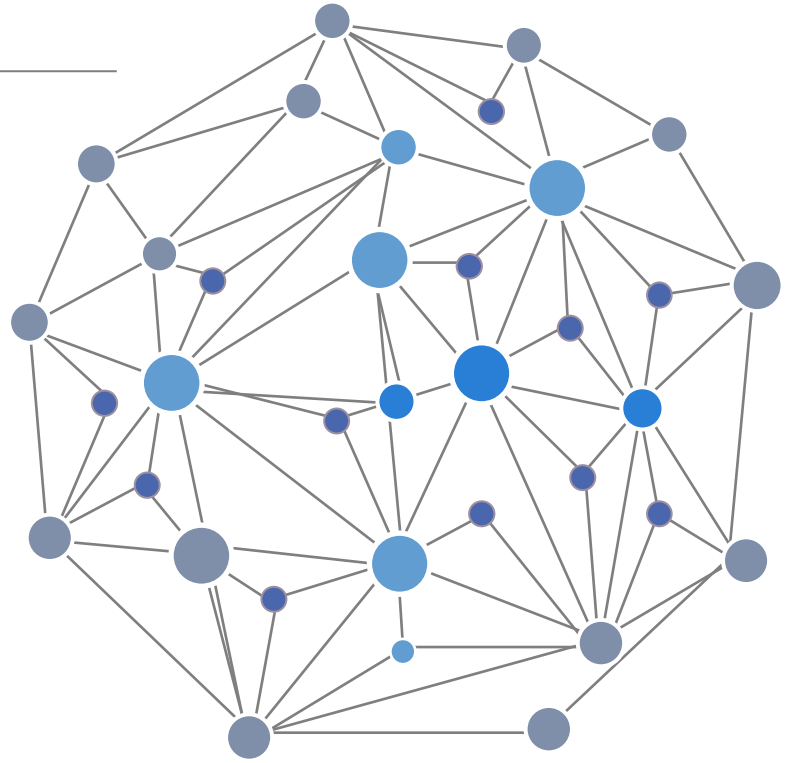
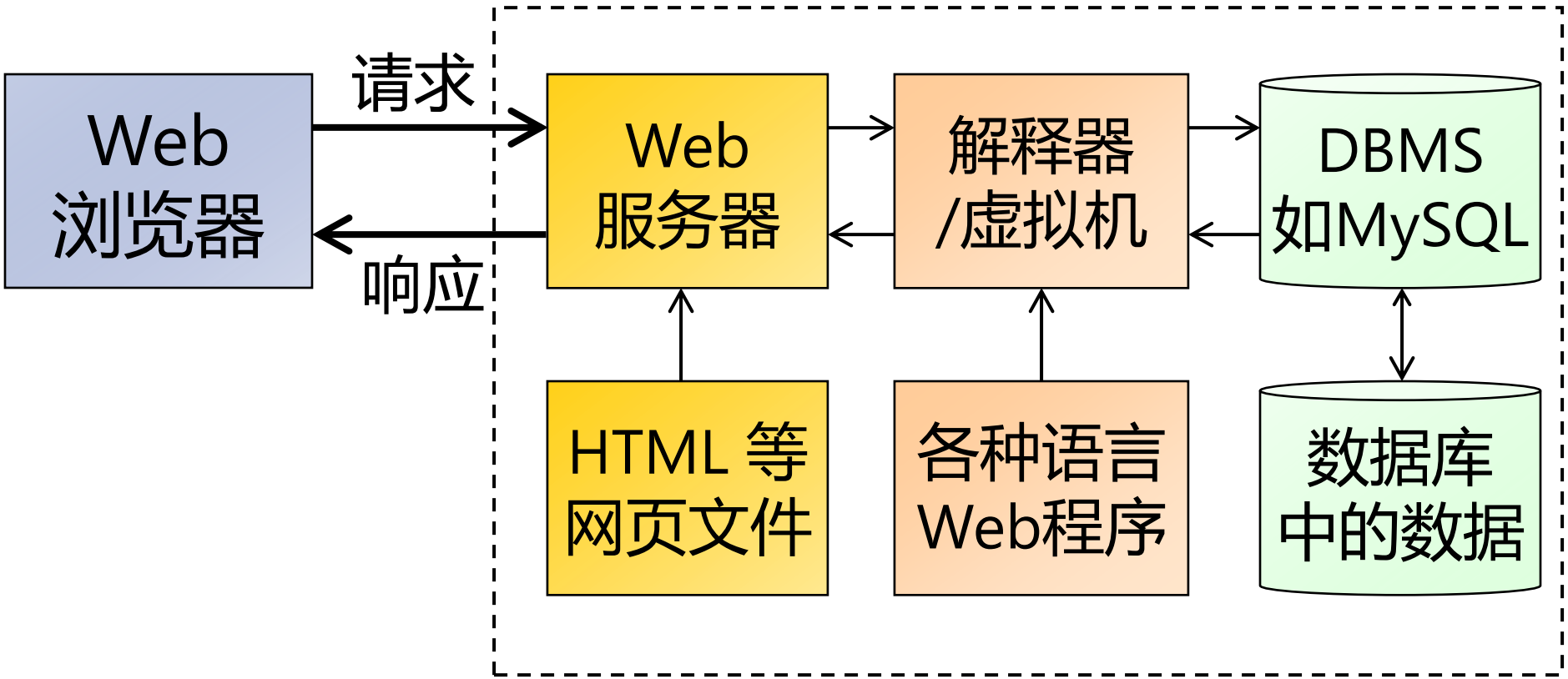
Web 程序设计



第八讲 PHP 编程 (1): 语法基础

福州大学 计算机与大数据学院
软件工程系 陈昱

Web Server Side Technologies



URL 和 Web 服务器

- 通常，你在浏览器输入 URL 时:
  - 你的电脑会通过DNS寻找服务器的 IP 地址
  - 你的浏览器连接到那个 IP 地址并请求指定文件
  - Web 服务器的软件（例如 Apache）会从服务器的本地文件中读取该文件，并把内容反馈给你

- 一些 URL 指定了要让 Web 服务器运行的程序，然后将输出作为结果返回给客户端：
  - http://php.net/manual/en/function.sqrt.php
  - 上面这个 URL 告诉服务器 php.net 运行 manual/en/function.sqrt.php 这个程序并返回输出结果

动态 vs 静态

- 静态网页
  - 客户端/用户的观点: 一个指向不变的 html 文件的 url
  - 服务器/开发者的观点: 一个存储在 Web 服务器根目录或者子目录下的 html 文件...
  - 可以直接在浏览器上显示
- 动态网页
  - 客户端/用户的观点: 一个指向动态内容的 url (每次请求访问都可能不一样)
  - 服务器/开发者的观点: 一个生成 html 的程序/脚本
  - 不是 html，但是程序生成 html
  - 不能直接在浏览器上显示

服务端 Web 编程

- 服务器端 Web 程序是用以下其中一种 Web 编程语言 / 框架编写的程序：
  - 例如：PHP，Java/JSP，Ruby on Rails，ASP.NET，Python，Perl
- Web 服务器端有个服务器软件，能够执行那些程序，并将输出结果作为响应传送回相应的 Web 请求
- 每种语言 / 框架都有利有弊
  - 本课程中，我们使用 PHP 来进行服务器端编程

内容提要

- PHP 的历史，特点和安装配置
- PHP 基础语法
- 控制结构
- 数组使用
- 函数
- 代码重用
- 面向对象的 PHP，异常处理
- 参考资料

PHP 的诞生与应用范畴

- 缘起
  - Rasmus Lerdorf - 1994
  - 为统计他的个人网站的访问者而开发
- 最初 PHP 是 Personal Home Page 的缩写，不过后来变成了：

  PHP → PHP Hypertext Preprocessor

  just as GNU → GNU is Not UNIX ...
- PHP 是一个开源的软件产品

PHP 发展历史

- PHP 4
  - 开始普及的版本
- PHP 5.x
  - 增强了面向对象特性
  - 最后一个版本 PHP 5.6
- PHP 7.x
  - 运行性能有极大提高
  - 语法有一定变动

    http://www.runoob.com/php/php7-new-features.html
- 目前最新
  - PHP 8.x
  - https://www.php.net/

PHP 技术特点

- PHP 是一种服务器端的脚本语言，代码可以嵌入 HTML 文档中

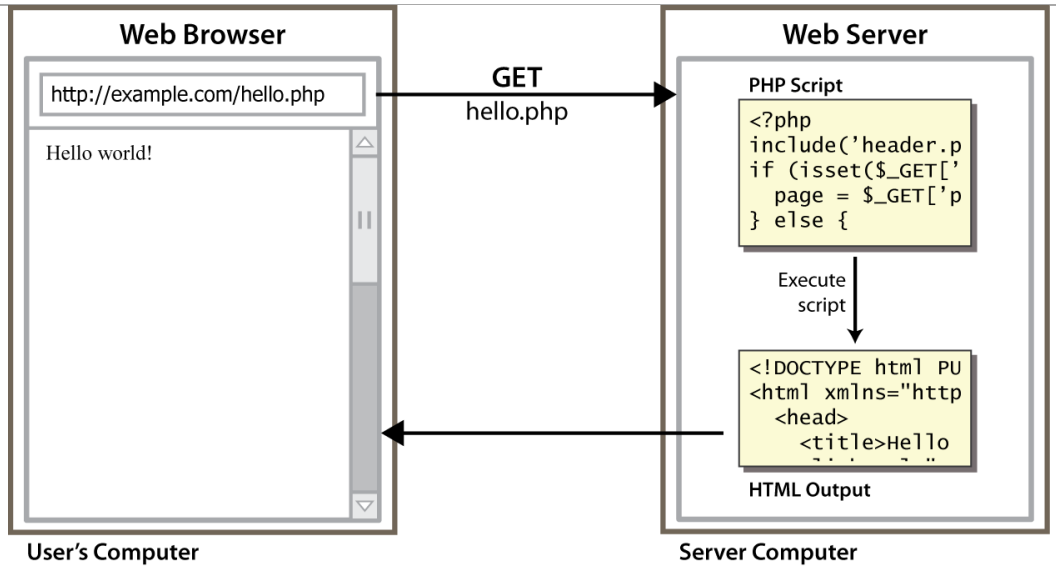
- 用来动态的生成网页:
  - 根据上下文提供不同的内容
  - 验证用户
  - 处理表单信息
  - 其它服务的接口：文件系统, 数据库等等

- 与其他服务器端编程技术的作用是一样的

PHP 技术特点

- PHP 的语法类似
  - C++, Java, JavaScript, Pascal, Perl ...
- PHP 是弱类型语言
  - 在运行时才能进行类型检查
- PHP 是解释型语言
  - 会先编译成中间代码(php 中称为 Opcode，**op**eration **code)**，然后再在虚拟机上解释执行
  - 目前有些编译器可编译成本地代码，但不成熟

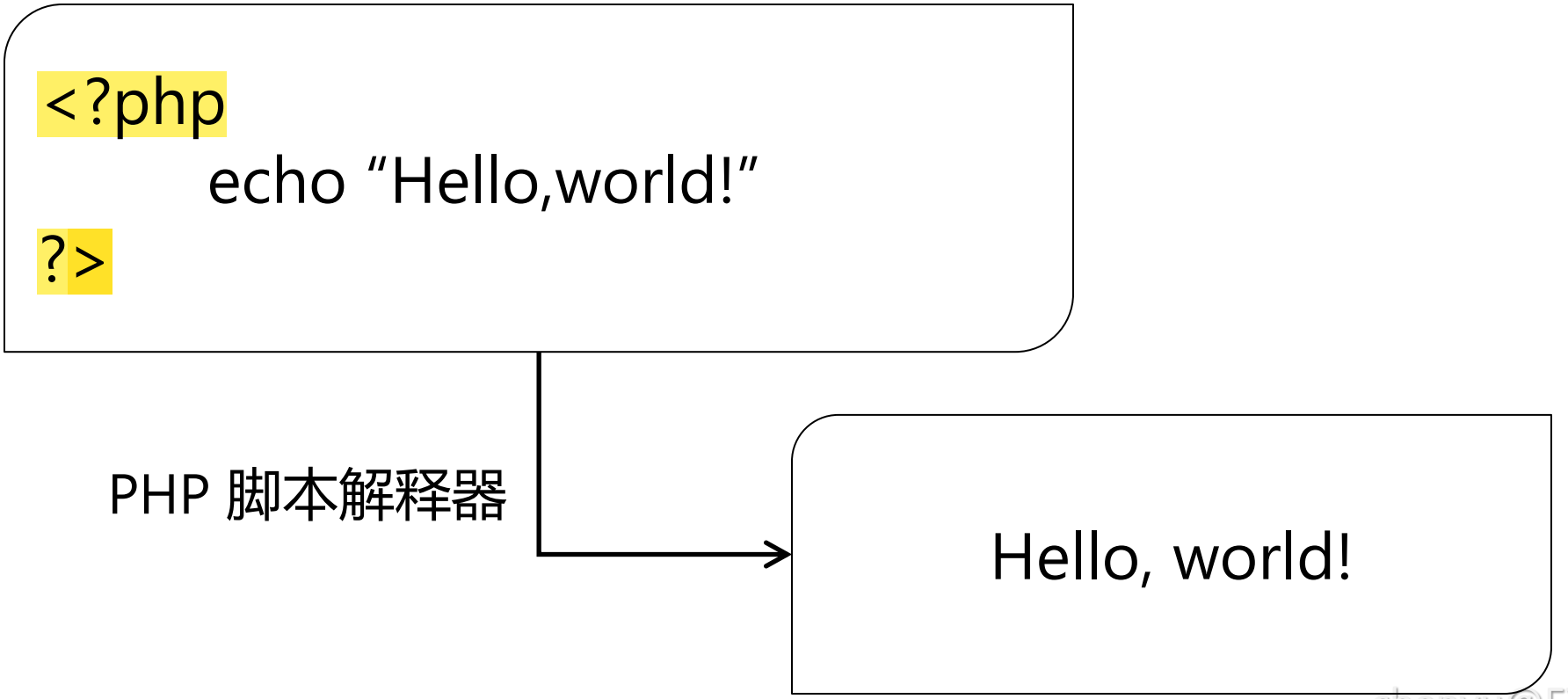
PHP 运行机制



- 浏览器请求一个 .html 文件 (静态内容)：服务器仅发送那个文件
- 服务器请求一个 .php 文件 (动态内容)：服务器读取，并执行里面的所有脚本代码，然后通过网络传送输出结果
  - 脚本产生的输出结果作为响应被返回

Hello, world!

- 以下内容可以放在一个 hello.php 的文件中：



观察 PHP 的输出

- 你的PHP代码经过服务器端PHP脚本解释器的运行，才能在浏览器输出。

获取PHP配置信息

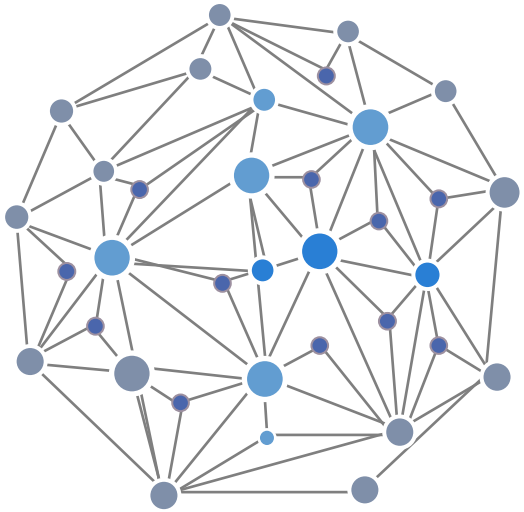
- phpinfo();

- 输出 PHP 当前状态的大量信息，包含了PHP 编译选项、启用的扩展、PHP 版本、服务器信息和环境变量（如果编译为一个模块的话）、PHP环境变量、操作系统版本信息、path 变量、配置选项的本地值和主值、HTTP 头和PHP授权信息(License)。

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

PHP 脚本引擎

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTM
<html><head>
<style type="text/css">
body {background-color: #ffffff; color: #000000
body, td, th, h1, h2 {font-family: sans-serif;}
pre {margin: 0px; font-family: monospace;}
a:link {color: #000099; text-decoration: none; b
a:hover {text-decoration: underline;}
table {border-collapse: collapse;}
.center {text-align: center;}
.center table { margin-left: auto; margin-right: a
.center th { text-align: center !important; }
td, th { border: 1px solid #000000; font-size: 75
h1 {font-size: 150%;}
h2 {font-size: 125%;}
.p {text-align: left;}
.e {background-color: #ccccff; font-weight: bol
.h {background-color: #9999cc; font-weight: bo
.v {background-color: #cccccc; color: #000000;
.vr {background-color: #cccccc; text-align: right
img {float: right; border: 0px;}
                ......
```



PHP 基础语法

基本语法

- PHP 代码使用 <?php ... ?> 标记嵌入(X)HTML/XML 文档中
  - 当 PHP 解析一个文件时，会寻找开始和结束标记，并执行其中的代码
  - 凡是在一对开始和结束标记之外的内容都会被 PHP 解析器当成文本直接输出给客户端
- 每个语句后用分号结束指令

```
<p>This is going to be ignored.</p>
<?php echo 'While this is going to be parsed.'; ?>
<p>This will also be ignored.</p>
```

PHP 程序的注释风格

```
<?php
   echo "这是第一种方法的例子。\n";
   //本例是C++的注释风格

   echo "这是第二种方法的例子。\n";
   /*本例采用多行的注释方式*/

   echo "这是第三种方法的例子。\n";
   #本例使用UNIX Shell注释风格
?>
```

数据类型

- PHP 支持八种原始类型：
- 四种标量类型：
  - boolean（布尔型），integer（整型）
  - float（浮点型，也作“double”）
  - string（字符串，有单引号和双引号两种）
- 两种复合类型：
  - array（数组），object（对象）
- 最后是两种特殊类型：
  - resource（资源），NULL

字符串

- 单引号和双引号括起来的字符串稍有不同
- 最大不同是：使用双引号的字符串中可以加入特殊的转义字符，生成转义序列
- 此外，双引号字符串中的变量名会被变量的值替代

PHP 变量

- PHP 是一种弱类型的程序语言
- 也就是说一个变量可以存储任意类型的数据，使用变量之前无须声明变量是字符型还是整型
- 与 C/C++/Java 语言有本质区别

PHP 变量

- 变量用一个美元符号后面跟变量名来表示
- 变量名前加 & 表示引用 reference

```
<?php
$foo = 'Bob';  // Assign the value 'Bob' to $foo
$bar = &$foo;        // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $bar;
echo $foo;                 // $foo is altered too.
?>
```

变量类型可以自动改变

```php
<?php
$foo = "0";  // $foo is string (ASCII 48)
$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies";
          // $foo is integer (15)
$foo = 5 + "10 Small Pigs";
          // $foo is integer (15)
?>
```

PHP 变量相关函数

- 未被赋值的变量有一个特殊的值，NULL
- ***unset*** 函数可以将一个变量的值设置为 NULL
- ***isset*** 函数可以用来检测一个变量是否是 NULL
- ***gettype*** 函数 -- 获取变量的类型
- ***settype*** 函数 -- 设置变量的类型

PHP 常量

- 常量使用 ***define()*** 定义

```php
<?php
    define("CONSTANT", "Hello world.");
    echo CONSTANT;
?>
```

常量和变量的不同

- 常量前面没有美元符号（$）
- 常量只能用 define() 函数定义，而不能通过赋值语句
- 常量可以不用理会变量作用域的规则而在任何地方定义和访问
- 常量一旦定义就不能被重新定义或者取消定义
- 常量的值只能是标量（整数，浮点数，字符串）

预定义变量

- PHP 提供了大量的预定义变量
- 包括主机和操作系统的许多环境变量
- 可以通过调用 phpinfo() 查看预定义变量列表

预定义常量

- __FILE__ 执行中的 PHP 程序文件名
- __LINE__ 执行中 PHP 程序行数
- PHP_VERSION PHP 的版本
- PHP_OS 执行 PHP 的操作系统名称
- TRUE 真值
- FALSE 假值
- E_ERROR 指向最近的错误处

可变变量

- 动态定义一个变量的名称

```
$a = 'hello';
$$a = 'world';
echo "$a $hello";
```

- 两个变量都被定义了：$a 的内容是“hello”
  并且 $hello 的内容是“world”

PHP 中的变量作用域

- 和 C 语言稍有不同
- 全局变量在函数中是无法直接访问的

```
<?php
$a = 1; /* global scope */

function Test()
{
    echo $a; /* reference to local scope variable */
}

Test();
?>
```

关键字 global

- 如果要在函数中引用一个全局变量，可以使用 global 关键字

```php
<?php
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
?>
```

静态变量 static

- 静态变量仅在局部函数域中存在，但当程序执行离开此作用域时，其值并不丢失。

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

PHP 运算符

- PHP 中的运算符与C语言中的相似，包括：
  - 算术运算符
  - 字符串运算符
  - 赋值运算符
  - 比较运算符
  - 逻辑运算符
  - 位运算符
  - 其他一些运算符

运算符

- 算术运算符： + - * / %

- 字符串运算符
  - 字符串运算符（string operator）只有一个，为英文句点 "."，其功能是将字符串连接起来，合并成新字符串
  - $str = ‘hello ’ . $name;

运算符

- 赋值运算符 = ， 还有复合赋值运算符 +=...

- 比较运算符： == , !=, <>, <, > ...
  - 恒等运算符 ===
  - 0=="0" 为真，但是 0==="0" 不为真

运算符

- 逻辑运算符 !, &&, ||, and ,or
- 位运算符 &, !,~,^,<<,>>
- 其他一些运算符
  - 三元运算符 ? :
  - 错误控制运算符 @
  - 执行运算符 ``

错误控制运算符 @

- 当将其放置在一个 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
- 放在变量，函数调用，常量等之前
- 不能放在函数或类的定义之前，也不能用于条件结构例如 if 和 foreach 等

执行运算符``

- 注意这不是单引号!
- 而是反单引号，位于"1"键左边的那个键

```
<?php
    $output = `ls -al`;
    echo "<pre>$output</pre>";
?>
```

PHP 表达式

- 表达式是标识符和运算符的组合

```
-12
$a=$b=5;
$str_name='Tom';
$arr_a=array('one', 'tow', 'three');
$int_total=++$int_number;
```

算术运算符与表达式

- 按通常的运算优先级计算
- 如果整数相除的结果不是一个整数，那么就返回一个 double
- 如果整数操作的结果导致溢出，那么就生成一个 double
- 取模操作会强迫将它的操作数转换成整数
- 浮点数舍入(rounding) 时是朝 0 的方向
- 数学函数
  - floor, ceil, round, abs, min, max, rand, etc.

字符串到数值的强制类型转换

- 当字符串参与算术运算时，PHP 会将字符串转换成数值
- 如果一个字符串包含 e 或是 E，它将被转换成浮点数，否则转换成整数
- 如果字符串没有以一个符号或是数字开始，那么会被当成 0

显式类型转换

- 例如:
  (int)$total 或 intval($total) 或
  settype($total, "integer")
- 变量的当前类型可以通过 gettype 函数或是 is_***type*** 函数获得

- gettype($total) - 可能返回 "unknown"
- is_integer($total) – 明确告诉你是还是不是

输出

- PHP 的输出通常是被送往浏览器的页面代码（HTML, XML 等）
- HTML 通过标准输出送往浏览器
- PHP 中有三种输出：echo, print 和 printf

```
echo "whatever";   # Only one parameter
echo "first <br />", $sum # More than one
print "Welcome to my site!";  # Only one
```

Short open tag

- PHP使用短标记 <?= 作为 <?php echo 的简写

```
<?php
        $username = "Geek";
        echo "$username";
?>
```

```
<?php
    $username = "Geek";
?>
<?= $username ?>
```

基本调试函数

- var_dump()
  - 打印变量的相关信息
- print_r()
  - 打印关于变量的易于理解的信息
  - 两者输出稍微有所差别，请实践中自行比较

```
<?php
    $a = array (1, 2, array ("a", "b", "c"));
    var_dump ($a);
    print_r($a);
?>
```

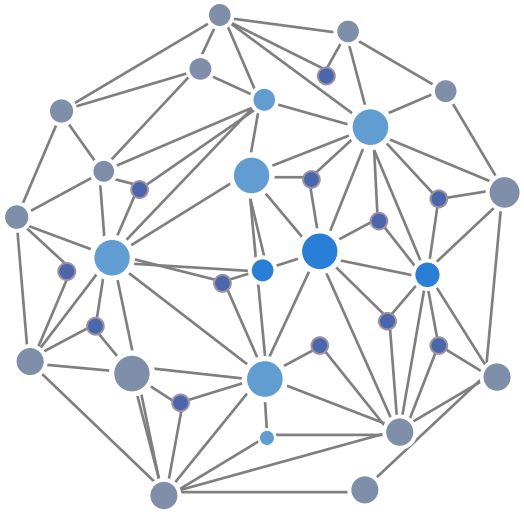
基本调试函数

var_dump 输出

```
array(3) {
      [0]=>
      int(1)
      [1]=>
      int(2)
      [2]=>
      array(3) {
        [0]=>
        string(1) "a"
        [1]=>
        string(1) "b"
        [2]=>
        string(1) "c"
      }
    }
```

print_r 输出

```
Array
(
    [0] => 1
    [1] => 2
    [2] => Array
        (
            [0] => a
            [1] => b
            [2] => c
        )

)
```



控制结构

PHP 程序的流程控制

- 流程控制对任何编程语言都是很重要的，对PHP来说当然也不例外

- 程序设计语言有3种基本的流程结构：
  - 顺序
  - 分支
  - 循环

分支结构

if, if-else, elseif, switch

if (expression)

statement

- 测试表达式
  - 与JavaScript相同的关系运算符，包括 === 和 !==
  - 和 Perl 类似的逻辑运算符 （ &&，and 等）
- **switch 表达式的类型必须为整数，浮点数，或是字符串**

循环结构

- while，do-while，for ，foreach

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
?>
```

循环控制

- break - 可以用于 for, foreach, while, do-while 或是 switch

- continue - 可以用于任何循环

```
<body>
  <table border = "border">
    <caption> Powers table </caption>
    <tr>
      <th> Number </th>        <th> Square Root </th>
      <th> Square </th>
      <th> Cube </th>        <th> Quad </th>
    </tr>
    <?php
     for ($number = 1; $number <=10; $number++) {
       $root = sqrt($number);
       $square = pow($number, 2);
       $cube = pow($number, 3);
       $quad = pow($number, 4);
       print(<tr align = 'center'> <td> $number </td>);
       print(<td> $root </td> <td> $square </td>);
       print(<td> $cube </td> <td> $quad </td> </tr>);
     }
    ?>
  </table>
</body>
```

powers.php

Powers table

Number	Square Root	Square	Cube	Quad
1	1	1	1	1
2	1.4142135623731	4	8	16
3	1.7320508075689	9	27	81
4	2	16	64	256
5	2.2360679774998	25	125	625
6	2.4494897427832	36	216	1296
7	2.6457513110646	49	343	2401
8	2.8284271247462	64	512	4096
9	3	81	729	6561
10	3.1622776601684	100	1000	10000

流程控制的替代语法

- PHP 提供了一些流程控制的替代语法，包括 if，while，for，foreach 和 switch。

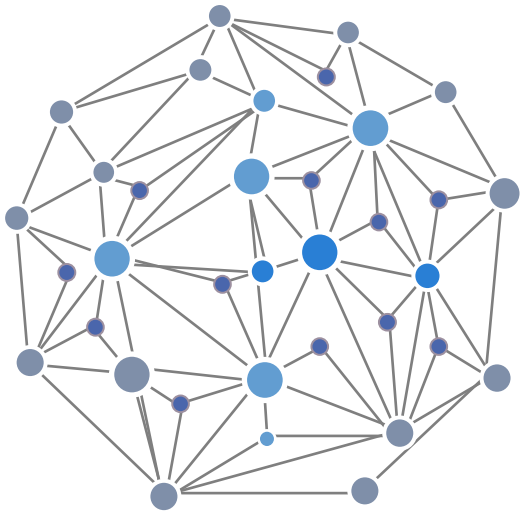
- 替代语法的基本形式是把左花括号（{）换成冒号（:），把右花括号（}）分别换成 endif; 等

- 新语法的优点是容易嵌入 HTML，可读性强
  - <?php if($i < 100) : ?>

流程控制的替代语法

```php
<?php
while ($a < 100) {
        $a = $a * $b + 7;
        $b++;
} ?>
```

```php
<?php
while ($a < 100) :
        $a = $a * $b + 7;
        $b++;
endwhile; ?>
```



数组的使用

数组

- 与其他编程语言中的数组的概念不同，PHP 中的数组是个——映射的 map，也就是一个 key 到 value 的映射

- 除了可以使用数字作为索引之外，它还能使用字符串作为索引

数组的创建

- 使用 array() 创建，用多个 **key => value**对作为参数，返回值是一个数组
- key 可以是整数或是字符串
- value 可以是任何类型数据

```
// 一个字符串数组
$list = array(0 => "apples",
              1 => "oranges",
              2 => "grapes")
```

数组的创建

- 如果给出的 value 没有指定索引，则取当前最大的整数索引值加一
- 如果指定的索引已经有了 value，则该值会被覆盖
- 数组中可以混合各种类型元素

```php
<?php
// This array is the same as ...
$a = array(5 => 43, 32, 56, "b" => 12);

// ...this array
$b = array(5 => 43, 6 => 32, 7 => 56,
      "b" => 12);

var_dump($a);
var_dump($b);
?>
```

输出：

```
array(4) {
    [5] =>  int(43)
    [6] =>  int(32)
    [7] =>  int(56)
    ["b"] =>  int(12)
}
```

短数组定义语法

- 自 5.4 起可以使用短数组定义语法，用 *[ ]* 替代 *array()*。

```php
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);

// 自 PHP 5.4 起
$array = [
    "foo" => "bar",
    "bar" => "foo",
];
?>
```

赋值数组元素

- 通过 [索引/Key] 访问赋值数组元素
- 如果数组不存在，数组将会被创建
- 如果一个数组元素不存在，它将会被创建

```
$arr[4] = 7;
$arr["day"] = "Tuesday";
$arr[] = 17;
var_dump($arr);
```

数组运算符

- 比较运算符

==	相等
===	全等
!= ,<>	不等
!==	不全等

- + : Union 联合

  联合运算符把右边的数组附加到左边的数组后面，但是重复的键值不会被覆盖。

数组的联合

```php
<?php
$a = array("a" => "apple", "b" => "banana");
$b = array("a" => "pear", "b" => "strawberry",
           "c" => "cherry");

$c = $a + $b; // Union of $a and $b
var_dump($c);

$c = $b + $a; // Union of $b and $a
var_dump($c);
?>
```

数组的联合

- Union of $a and $b:
  array(3) {
    ["a"]=>
    string(5) "apple"
    ["b"]=>
    string(6) "banana"
    ["c"]=>
    string(6) "cherry"
  }
- Union of $b and $a:
  array(3) {
    ["a"]=>
    string(4) "pear"
    ["b"]=>
   string(10)
  "strawberry"
    ["c"]=>
    string(6) "cherry"
  }

数组的提取

- 可以将 keys 和 values 从数组中提取出来

```
$highs = array( "Mon" => 74, "Tue" => 70,
                "Wed" => 67, "Thu" => 62,
                "Fri" => 65);

$days = array_keys($highs);
$temps = array_values($highs);
```

删除数组元素

- 使用函数 unset 可以删除数组中的元素

```
unset($list[4]);
    // 现在索引为4的元素不存在了
unset($list); // 删除整个数组
```

数组的排序函数

- 正向，逆向，是否保持索引关系等
- sort
- asort
- rsort
- ksort
- krsort

```php
$fruits = array("lemon", "orange", "banana", "apple");
sort($fruits);
foreach ($fruits as $key => $val) {
    echo "fruits[".$key."] = " . $val . "\n";
}
```

输出：

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

其他数组常用函数

- **is_array($list)**
  - 检测变量是否是数组
- **in_array(17, $list)**
  - 检测17是否是$list的一个元素
- **explode(" ", $str)**
  - 将一个字符串分割成数组，以空格为分隔符
- **implode(" ", $list)**
  - 将数组转换成字符串，以空格为分隔符

多维数组

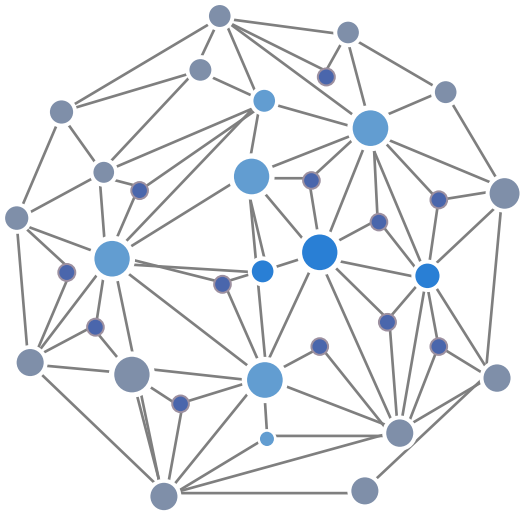
- 多维数组的创建

```
$arr_books=array(
   '0-679-7681-9'=>array(
    'name'=>'The New History', 'pages'=>400,
    'publisher=>'Vintage Books')
   '0-512-85585-5'=>array(
    'name'=>'Children of the Mind',
    'pages'=>549 , 'publisher'=>'Tor Books')
);
```

多维数组的访问

- 要查询 The New History 一书的页数有多少，可以使用以下的表达式：

- $arr_books[‘0-51-85595-5’][‘pages’]



函数

函数

- PHP函数的定义语法为：
  function 函数名(参数列表)
  {
  　　函数功能;
  }
- 函数不能被重载、重复定义，但可以嵌套
- **函数名和关键字忽略大小写区别！**
- 可以有返回值(return)，也可以没有

函数允许嵌套定义

```
function foo() {
        function bar() { … }
        …
}
```

函数定义

```
<?php
// prints input in strong
// $stVar is var of type string
function inBold ($stVar)
    {print “<strong>” . $stVar . “</strong>”;}
// ---------------------

inBold(“soup”);
$var =“baked beans”;
inBold($var);
?>
```

PHP 函数原型

- PHP 内部函数拥有相应的原型定义：

- 函数可能有些可选的参数，例如下面这个原型里的 **timestamp**：

```
string date(string format [, integer timestamp])
```

- 还有些函数可能有不定数目的参数

```
string fun1(string p1,...)
```

参数的引用传递

- 参数声明前加 & 表示引用传递

```
function addOne(&$param) {
        $param++;
}

$it = 16;
addOne($it);  // $it is now 17
```

参数的引用传递

- 如果函数声明是没有使用引用的形式，也可以通过引用的形式将参数传递给函数，从而达到引用传递的目的

```
function subOne($param)
    { $param--; }
$it = 16;
subOne(&$it);   // $it is now 15
```

返回值

- 函数的返回值可以是任意类型，包括数组和对象，例如以下定义的函数就返回一个数组：

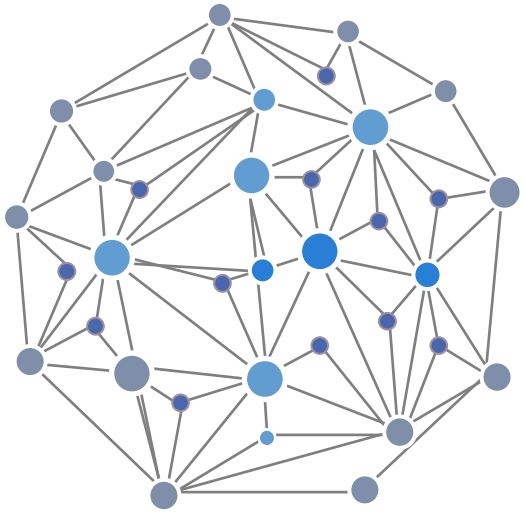
```
<?php
  function array_list()  {
    return [10,20,30,40,50];
  }
?>
```

- 利用赋值语句：$arr_temp=array_list()；就得到一个表示数组的变量$arr_temp，它的各个元素分别为：10，20，30，40，50

返回值

- 函数不能返回多个值
- 从函数返回一个引用，必须在函数声明和指派返回值给一个变量时都使用引用运算符 &

```
function &returns_reference() {
    $someref = "hello";
    return $someref;
}

$newref =& returns_reference();
echo $newref;
```



代码重用

代码重用

- 将常用的功能写成一个函数，预先存放在文件中，然后在编制含有PHP程序的页面中引用相应的文件，就可以调用适用的函数
  - 这样既可加强代码的灵活性和可读性，也有利于代码重用
  - 例如实现简单的模版机制
- 引用外部文件的函数有两种：
  - require()函数及include()函数

在 PHP 中引用文件

- require()的使用方法，如

```
require("MyRequireFile.php");
```

- 该函数一般放在 PHP 程序的最前面，PHP 程序在执行前，就会先读入 require 语句所指定引入的文件，使它变成 PHP 文件的一部分

在 PHP 中引用文件

- include() 的使用方法和 require() 一样

```
include("MyIncludeFile.php");
```

- 两者的区别在于，当 require 引用的文件丢失时会产生一个严重错误，程序会停止运行
- include 引用的文件丢失时只会产生一个警告，程序继续运行

home.php

```
<?php
 require('header.inc');
?>
  <!-- page content -->
  <p>Welcome to the home of TLA Consulting.
  Please take some time to get to know us.</p>
  <p>We specialize in serving your business needs
  and hope to hear from you soon.</p>
<?php
 require('footer.inc');
?>
```

```
<!-- header.inc -->
<html>
<head>
  <title>TLA Consulting Pty Ltd</title>
.............
```

```
<!-- footer.inc-->
        ......
  </body>
</html>
```

模版 Template

- 将网站的页眉，页脚这些标准的组成部分放入单独的文件中，再用 require 引用
- 可以很容易使得网站具有统一的风格
- 而且易于修改

- 这就是最简单的 Web 网站 “模版”

PHP 模板引擎

- 如果要使得网站能够长期使用和易于维护，就需要将逻辑和内容相分离
- 也就是 PHP 和 HTML 相分离
- 较好的方法是使用模板引擎

常用 PHP 模板引擎

- Smarty
  - http://smarty.php.net/
- PHPTemplate
  - http://drupal.org/project/phptemplate
- PHPlib
- Template Lite
- XTemplate

面向对象的 PHP

```php
<?php
class SimpleClass {

    // 成员声明
    public $var = 'a default value';

    // 方法声明
    public function displayVar() {
        echo $this->var;
    }
}
?>
```

面向对象的 PHP：继承

```php
<?php
class ExtendClass extends SimpleClass
{
    // Redefine the parent method
    function displayVar()
    {
        echo "Extending class\n";
        parent::displayVar();
    }
}

$extended = new ExtendClass();
$extended->displayVar();
?>
```

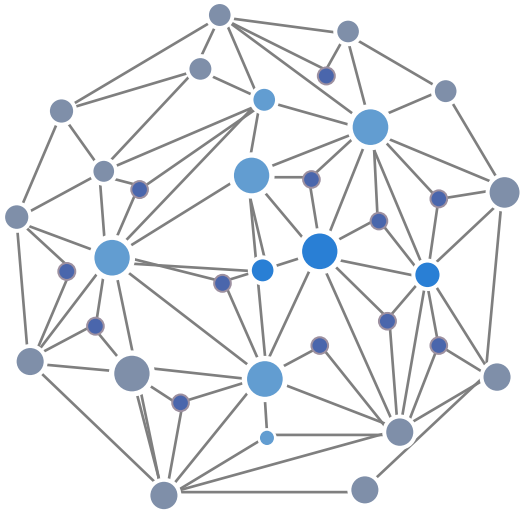
异常处理

```php
<?php
try {
    $error = 'Always throw this error';
    throw new Exception($error);

    // 下面的代码将不会被执行
    echo 'Never executed';

} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}

// 继续执行
echo 'Hello World';
?>
```

Things to do

- 学习 w3school 在线教程
- 包括了在线教程和参考手册
  - https://www.w3school.com.cn/php/index.asp

- 编程作业
  - http://chenyv.gitee.io/webprogramming/homeworks/php



PHP 参考资料

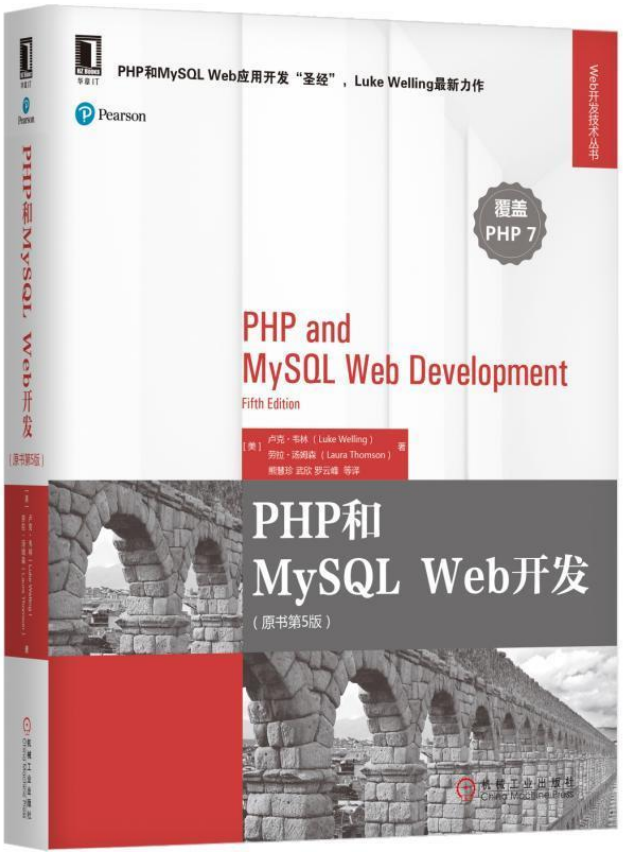
PHP IDE

- JetBrains PphStorm
- Zend Studio
- Eclipse PDT
- VS Code, Sublime Text
- ...

PHP 官方手册

- http://php.net/manual/zh/
- http://php.net/download-docs.php

PHP和MySQL Web开发



- PHP和MySQL Web开发 （原书第5版）
- 入门书
- 前面语法，后面实例

进阶级与一些专门主题讨论书籍

- PHP5 Power Programming（PHP5权威编程）
- Essential PHP Security （PHP 安全基础）
  - 电子书，有中文版
- PHP 设计模式（电子书，有中文版）
- Ajax 与 PHP Web 开发
- Extending and Embedding PHP
- Pro PHP: Patterns, Frameworks, Testing and More

THANKS

本章结束

陈昱

福州大学 计算机与大数据学院 软件工程系