

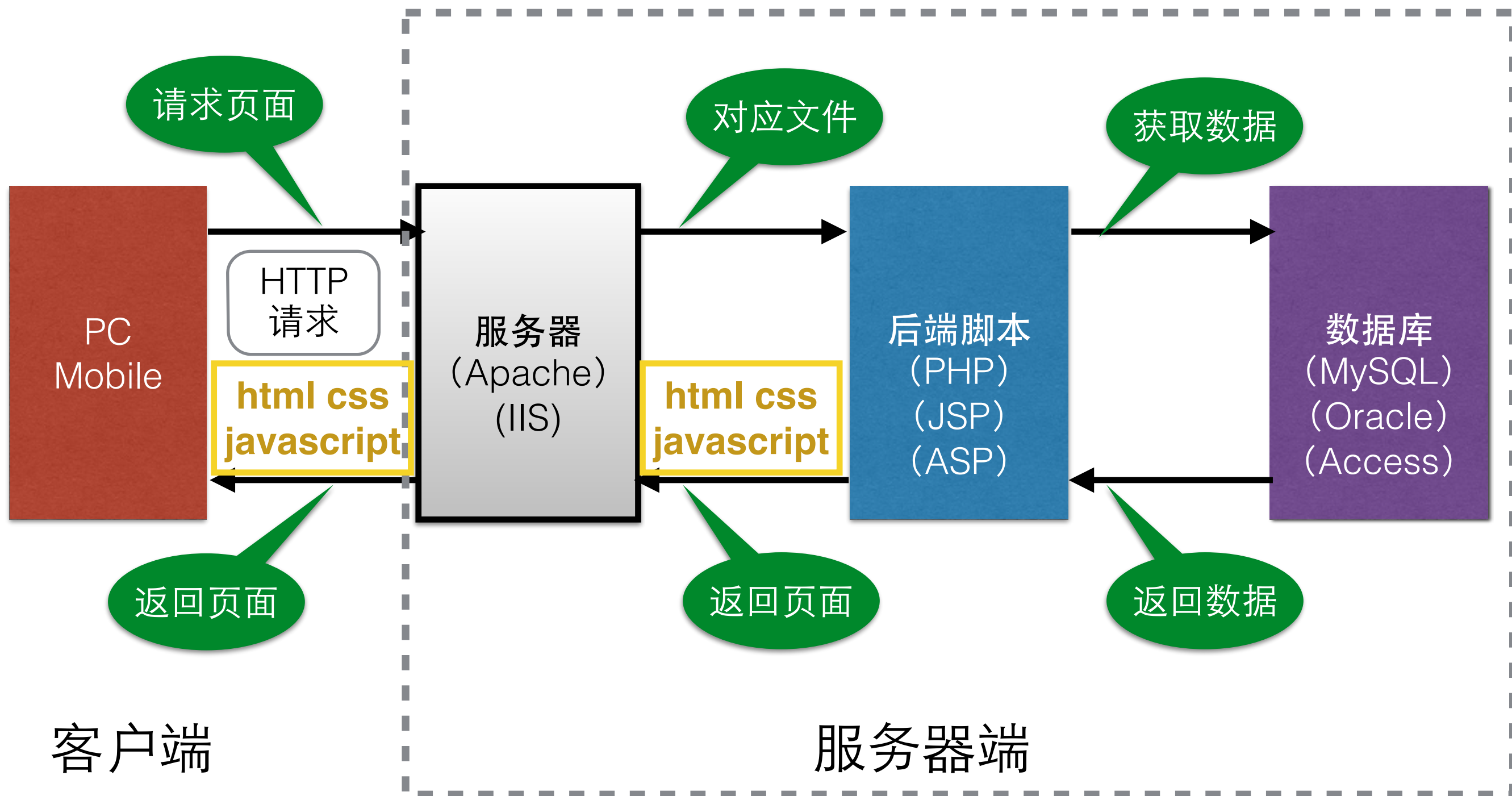


PHP基本语法

—变量、数组、字符串

杨亮

Web基本流程



PHP基本角色

前端工程师

PC
Mobile

html

页面内容

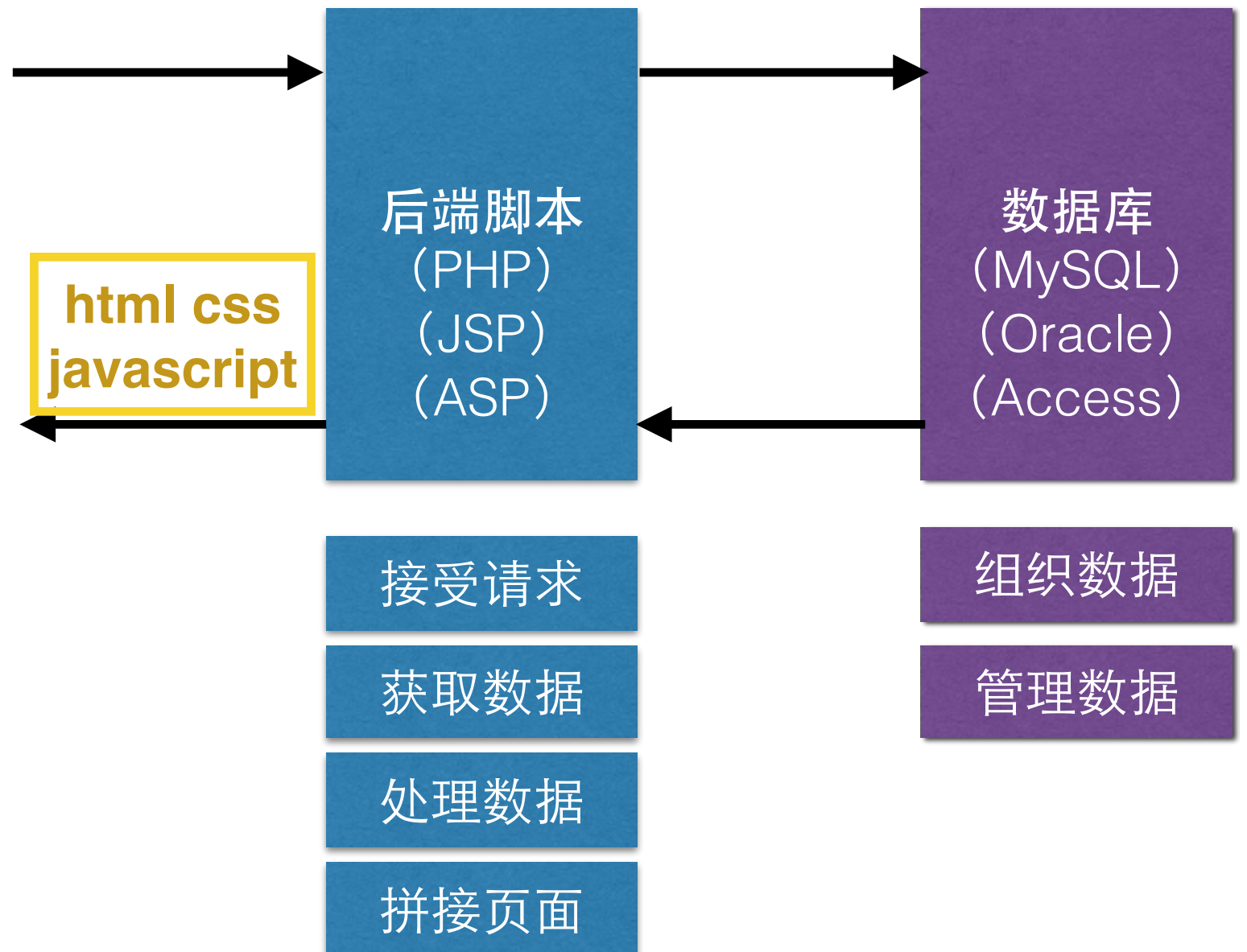
CSS

页面外观

JavaScript

页面行为

后端工程师



第一个PHP程序

拼接页面

From HTML

```
1 <html>
2 <head></head>
3 <body>
4 <p>This is my first PHP code :)</p>
5 </body>
6 </html>
```

This is my first PHP code :)

To PHP

```
1 <html>
2 <head></head>
3 <body>
4 <?php
5 echo "<p>This is my first PHP code :)</p>";
6 ?>
7 </body>
8 </html>
```

第一个有意义的PHP程序

```
1 <html>
2 <head></head>
3 <body>
4 <?php
5 echo "<p>This is my first PHP code :) ";
6 echo date('H:i, jS F Y');
7 echo "</p>\n";
8 ?>
9 </body>
10 </html>
```

动态内容

This is my first PHP code :) 13:47, 17th February 2015

```
1 <html>
2 <head></head>
3 <body>
4 <p>This is my first PHP code :) 13:47, 17th February 2015</p>
5 </body>
6 </html>
```

PHP中的变量

- 无需声明，直接使用
- 变量名：以\$开头
 - 字母（大小写敏感）
 - **下划线**
 - 数字（不能开头）
- 动态变量类型，赋值时动态改变
- 变量的变量，变量名动态改变

```
1 <?php
2 $thisIsMyName
3 $this_is_my_name
4 ?>
```

```
1 <?php
2 //dynamically typed
3 $value = 100;
4 $value = 'example';
5
6 //variable variable
7 $value = 'test';
8 $$value = 123;
9 echo $test;
10 ?>
```

常量

- 为什么要有常量：代码可读性和规范
- 通常大写，用以和变量区分
- 非\$开头，直接使用
- 只能存放标量
 - 布尔、整形、浮点和字符串

```
1 <?php
2 define("PRICE", 100);
3 $price_60 = PRICE*60;
4 $price_100 = PRICE*100;
5 echo PRICE;
6 echo $price_60;
7 echo $price_100;
8 ?>
```

运算符

- 与C语言的基本一致（别说都忘了）
- 组合运算符 +=, -=, *=, /=, %=
- 递增、递减运算符 ++, --,
- 赋值运算符 (=) 与比较运算符 (==)
- 赋值运算符 (=) 与引用运算符 (= &)
 - \$a = \$b; vs \$a = &\$b;
- 错误抑制运算符 (@)

```
2 //incremental
3 $a = 4;
4 echo $a++;
5 echo $a;
6 $b = 4;
7 echo ++$b;
8 echo $b;
9
10 //comparsion
11 $x == 6;
12 6 == $x;
13
14 //reference
15 $v = 5;
16 $y = $v;
17 $z = &$v;
18 $v = 6;
19 echo $y;
20 echo $z;
```


数组

- 数值索引数组-自动索引
 - `array(val1,val2,val3,...)`
 - 混合类型
 - 下表从0开始
- 自定义索引数组-人为指定索引
 - 任何字符串和整数值都可以做下标
 - 下标0和'0'是不同的
- 数组运算符
 - '+'合并两个数组，如果有索引冲突，保留前者

```

1 <?php
2 //automatically index
3 echo "test";
4 $products = array( 'Tires', 'Oil', 'Spark Plugs', 100 );
5 echo $products[0];
6 $products[2] = 'test';
7 echo $products[2];
8
9 //manually index
10 $prices = array( 'Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
11 $prices['Tires'] = 20;
12 echo $prices['Oil'];
13
14 //number indexed and string indexed
15 $mytest[0] = 'aaa';
16 $mytest['0'] = 'bbb';
17 echo $mytest[0];
18 echo $mytest['0'];
19
20 //union
21 $prices_new = array( 'Meat'=>30, 'Apple'=>15, 'Oil'=>10 );
22 $prices_all = $prices + $prices_new;
23 echo $prices['Oil'];
24 ?>


```

0	1	2	3
'Tires'	'Oil'	'Spark Plugs'	100

Tires'	Oil'	Spark Plugs'
100	10	4

多维数组

```
1 <?php
2 $products = array(    array( 'TIR', 'Tires', 100 ),
3                        array( 'OIL', 'Oil', 10 ),
4                        array( 'SPK', 'Spark Plugs', 4 ) );
5
6 $products = array(    array(      'Code' => 'TIR',
7                                'Description' => 'Tires',
8                                'Price' => 100
9                                ),
10                       array(      'Code' => 'OIL',
11                                'Description' => 'Oil',
12                                'Price' => 10
13                                ),
14                       array(      'Code' => 'SPK',
15                                'Description' => 'Spark Plugs',
16                                'Price' => 4
17                                )
18                       );
19 ?>
```



Code	Description	Price
TIR	Tires	100
OIL	Oil	10
SPK	Spark Plugs	4

数组顺序相关函数

- 排序函数
 - 针对自动索引数组sort, 可按数值顺序和字符顺序
 - 针对自定义索引数组asort(按value), ksort(按索引)
 - 相应的反向排序rsort, arsort, krsort
- 乱序函数 shuffle (洗牌)
- 反序函数 array_reverse

字符串

- 无需提前分配空间和之后的回收空间
- 多个字符串可以通过“.”来进行连接
- 用单引号还是双引号呢？（处女座绕行）
 - 单引号中的所有内容都当做普通的字符不进行解析（除去单引号本身），因此速度较快
 - 双引号中的变量（以\$开头）和转义符（\）都将被解析，实现拼接目的，因此速度较慢
 - 为了防止混淆，建议双引号中的变量前后加上一对{ }

```
1 <?php
2 //concatenate
3 $name = "Liang Yang";
4 $state = "My name is ".$name;
5
6 //single quote or double quote
7 $s = "I am a 'single quote string' inside a double quote string";
8 $s = 'I am a "double quote string" inside a single quote string';
9 $s = "I am a 'single quote string' inside a double quote string";
10 $s = 'I am a "double quote string" inside a single quote string';
11
12 $first_name = "Liang";
13 $last_name = "Yang";
14 $full_name = $first_name . ' ' . $last_name;
15 $full_name = "$first_name $last_name";
16 $full_name = "{$first_name} {$last_name}";
```

```
18 $foo = 2;
19 echo "foo is $foo"; // 打印结果: foo is 2
20 echo 'foo is $foo'; // 打印结果: foo is $foo
21 echo "foo is $foo\n"; // 打印结果: foo is 2 (同时换行)
22 echo 'foo is $foo\n'; // 打印结果: foo is $foo\n
23
24 $foo = array('liang', 'yang', 'stefyang');
25 $i = 0;
26 $j = 1;
27 $k = 2;
28 echo "My name is {$foo[$i]} {$foo[$j]} or {$foo[$k]}";
29
30 //double quote, \ and { in double quote
31 $var = 3;
32 echo "value = {$var}"; // 打印结果 "value = 3"
33 echo "value = \{$var}"; // 打印结果 "value = {3}"
34
35 $file = "c:\windows\system.ini";
36 echo $file; // 打印结果为: c:windowssystem.ini
37 $file = "c:\\windows\\system.ini";
38 echo $file; // 打印结果为: c:\windows\system.ini
39 ?>
```

字符串常用函数

- 删除首尾的空白字符 trim()
- 字符串与数组互相转换 explode, implode
- 英文分词 stoke
- 提取字符串substr
- 字符串长度strlen
- 字符串比较strcmp
- 字符串查找替换strstr, strpos, str_replace

自行阅读PMWD相关章节
所有出现过的函数自己写一个例子

一些普通查找无法完成的工作

- 提取某一文章中的出现过的邮件地址或电话号码
- 提取文章中出现过的所有3-8位的数字
- 提取某一网页中出现的所有图片的网址
- 提取日志中所有出现过的IP地址
- 匹配文章中出现过的英文日期
- 匹配文章中出现过的所有very, very very, very very very

正则表达式



- 从精确匹配到模式匹配
- 对原始的通配符（? *）的扩展
- 元字符(metacharacter),代表一组字符，可以匹配其中的任何一个（空白符，数字符等）
- 重复出现次数(*,+,?,{})
- 字符类（指定特殊字符集合）“[]”
- 分组（某些模式的组合）“()”
- 转义符“\”，表示已被特殊使用的字符（*,+,?,.,\）

代码	说明
.	匹配除换行符以外的任意字符
\w	匹配字母或数字或下划线或汉字
\s	匹配任意的空白符
\d	匹配数字
\b	匹配单词的开始或结束
^	匹配字符串的开始
\$	匹配字符串的结束

代码/语法	说明
*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次

正则表达式例子

同一行中有hi和lucy

- 如果搜“hi”，可能会找到high, him, history....

代码/语法	说明
*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次

- 可以使用 **\bhi\b**来搜索单词hi

- 同理可以通过**\blucy\b**搜索单词lucy

- 通过**\bhi\b.*\blucy\b**

代码	说明
.	匹配除换行符以外的任意字符
\w	匹配字母或数字或下划线或汉字
\s	匹配任意的空白符
\d	匹配数字
\b	匹配单词的开始或结束
^	匹配字符串的开始
\$	匹配字符串的结束

- 带区号的电话号码
 - 以0开头后面接两个数组，再接-，在接8个数字
 - `0\d\d-\d\d\d\d\d\d\d\d`
 - `0\d{2}-\d{8}`
- 匹配任何以a开头的单词 `\ba\w*\b`
- 匹配任何连续出现的数字 `\d+`
- 从开头匹配 `^`，匹配到结尾 `&`，全部匹配 `^ $`
 - 输入是5-12位数字 `^\d{5,12}$`
- 匹配doc文件名 `\w+\.doc`
- 匹配aeiou中的任何一个字符 `[aeiou]`
- `[a-zA-z0-9_]` 等价于 `\w`

代码/语法	说明
*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次

代码	说明
.	匹配除换行符以外的任意字符
\w	匹配字母或数字或下划线或汉字
\s	匹配任意的空白符
\d	匹配数字
\b	匹配单词的开始或结束
^	匹配字符串的开始
\$	匹配字符串的结束

精确的电话号码匹配

- 类可能的电话号码 (010)12345678, 010-12345678, 010 12345678, 01012345678
- `\(?0\d{2}[\)]-[\d]{8}`
- 可能会有错误 (010-12345678或者010)12345678
- 使用条件分支
- `(0\d{2})-\d{8}|0\d{2}[\]?[\d]{8}`

IP地址的匹配

- 202.113.29.4或者127.0.0.1
- $(\backslash d\{1,3\}\backslash.){3}\backslash d\{1,3\}$
- 但是这回匹配 012.300.900.00这种非IP地址
- 问题出现在 $\backslash d\{1,3\}$ 过于简单了
- 如何解决留作课后练习

PHP中使用正则表达式
是时候展示你们较强的自学能力了

变量相关函数

- 获得变量类型gettype
- 检测变量类型是否为
 - is_*(array,null,numeric,scalar)
- 查看变量是否存在isset，只要存在则返回true
- 清楚变量unset，为了节省内存空间
- 验证变量是否存在且或者为空，（0，空字符串，空数组，未设置）都返回true

```
1 <?php
2 $a = 0;
3 $b = '';
4 $c = array();
5 echo empty($a); //1
6 echo empty($b); //1
7 echo empty($c); //1
8 echo empty($d); //1
```