* 函数
  + 分类
    - 系统预定义
    - 用户自定义
  + 定义
    - Function 函数名 (形参1，形参2，形参N) {
      * 函数体；
    - }
    - 注意：
      * 1，函数可以先调用，后声明
      * 2，函数声明好之后，必须调用才会生效
  + 命名规则
    - 1.可以使用字母，数字，下划线
    - 2，数字不能开头
    - 3，下划线开头慎用
    - 4，命名要有意义
    - 5，小驼峰
    - 6，函数名不能重复
  + 形参&实参
    - 形参：定义函数时的参数，形式上的参数
    - 实参：调用函数时的参数，实际上的参数
  + 变量范围
    - 局部变量：函数内部定义的变量
    - 全局变量：函数外部定义的变量
    - 默认，：局部，全局互相不能调用
  + 局部调用全局：
    - 1.传参
    - 2，使用global关键词。
      * 使用global之后，相当于局部的变量是全局变量的引用
  + 全局调用局部
    - 1.return返回
  + Return注意：
    - 1.return返回的值，需要变量接收
    - 2，一个函数中，只能有一个return生效
    - 3.return之后的代码，不会执行，return结束，退出函数空间
  + 实参&形参数目问题
    - 默认，形参=实参，并且对应关系为从左到右一一对应
    - 实参>形参
      * 多的实参，丢弃
    - 实参<形参
      * 默认报错，丢参数
      * 使用默认值，注意：需要默认值的参数往后放
    - 实参个数不定
      * Func\_get\_arg()
      * Func\_get\_args()
      * Func\_num\_args()
  + 可变函数
    - 定义：把函数的定义赋值给变量
    - 使用：$变量名(实参1，实参2，实参N)；
  + 静态变量
    - 默认情况下，函数在退出空间的时候会销毁所有的局部变量。如果不想被销毁，可以在变量前加static
  + 检测
    - Function\_exists(),返回bool
  + 获取所有的函数
    - Get\_defined\_functions()
    - 返回2维数组
  + 匿名函数|闭包
    - 没有名字的函数就是匿名函数
  + 回调函数
    - 把某个函数当初另外一个函数的参数进行调用的方式就是回调函数
  + 递归函数
    - 自己调用自己的函数
    - 注意：有要求：必须有某个条件来限定，判断什么情况下才会调用自己，什么情况下会结束。说 白了和循环一个道理
* 数组
  + 定义方式：
    - $arr = array()
    - $arr = []
    - $arr[] = ‘’;
    - 没有区别，都可以
  + 键值对：
    - 数组由键值对组成，键值的关系是：键=>值
    - 注意：
      * 1.键名不能重复
      * 2，键值对之间使用逗号隔开
      * 3，如果不给键，默认从0开始，依次递增(索引数组)
      * 4，键只能是标量
      * 5，如果不给键，那么键就是前面的最大键的值+1
      * 6，定义数组的时候，如果键重复，后面的会覆盖前面的值
  + 分类：
    - 根据下标：
      * 索引数组
        + 下标从0开始，依次递增1，并且连续的数组
      * 关联数组
        + 不是索引数组，就是关联数组
    - 根据层级：
      * 一维数组
      * 二维数组
      * 多维数组
  + 运算符：
    - $a + $b
      * 联合
      * 注意：如果键重复，保留前面的值
      * Array\_merge():数组的合并
    - $a == $b
      * 相等
    - $a <> $b
      * 不等
    - $a != $b
      * 不等
    - $a === $b
      * 全等
    - $a !== $b
      * 不全等
    - 注意：记得需要对顺序的判断
  + 遍历：
    - For
      * 注意：
        + 1，只能遍历索引数组
        + 2，需要获取数组键值对数目，count()
    - While..each..list
      * 依据each移动数组指针遍历
      * 可以遍历所有的数组
      * 遍历之前，记的重置数组指针，reset()
    - Foreach
      * 推荐的，可以遍历所有的数组，且不需要重置指针位置
      * 格式：

Foreach($arr as $k=>$v) {

}

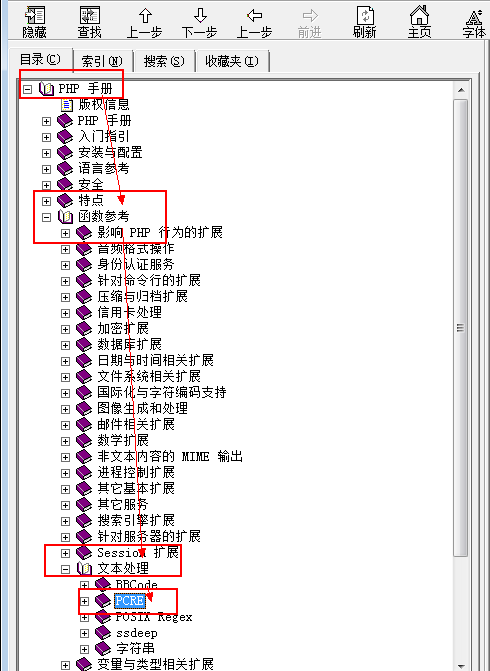
Foreach($arr as $v) {

}

* + - 手动
      * Reset()
      * Current()
      * Key()
      * Next()
      * Prev()
      * End()
  + 函数：
* 字符串
  + 输出方式对比
    - Echo:
      * 1.不是函数
      * 2，没有返回值
      * 3，只能输出标量，并且bool的false不能输出
    - Print:
      * 1.不是函数
      * 2，返回值是1
      * 3，只能输出标量，并且bool的false不能输出
    - Print\_r()
      * 1.是函数
      * 2，返回值是bool
      * 3.可以输出标量+数组，并且bool的false不能输出
    - Var\_dump()
      * 1.是函数
      * 2，没有返回值
      * 3，可以输出所有的类型
  + 格式化输出
    - Printf()
      * 返回格式化的结果长度，格式化结果直接输出
    - Sprintf()
      * 返回格式化的结果
    - 常见的格式化方式，占位符：
      * %s:字符串
      * %d:有符号的正整数
      * %u:无符号的整数
      * %e:科学计数法
      * %f:浮点数
      * %%:%
  + 函数
    - 加密：md5()

# 正则表达式

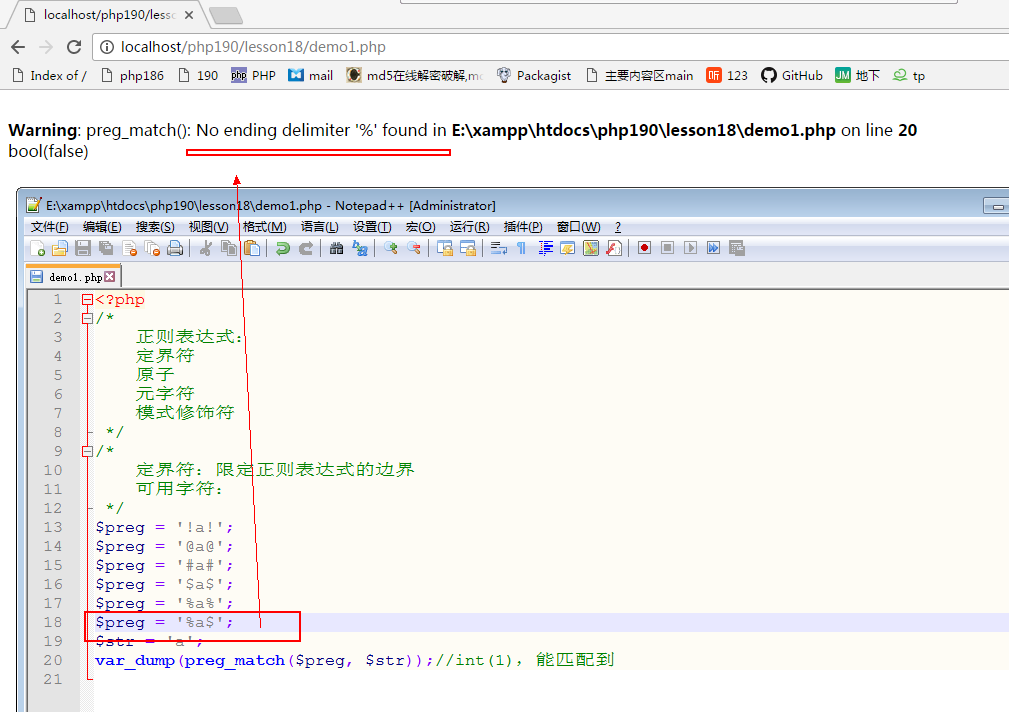
* 手册位置：



* 作用:可以匹配字符串
* **注意：能用字符串函数处理的，不要轻易用正则表达式，因为正则效率低**

## 定界符

* 作用：限定正则的边界
* 可用字符：好多，shift+1到+都可以
  + ！
  + @
  + #
  + $
  + %
  + ^
  + &
  + \*
  + ()
  + \_
  + +
  + /
* 注意：
  + 1.定界符必须成对出现
  + 2，必须一致



* **最常用的定界符：/ 正斜线**
* **注意：反斜线不行，有别的含义**

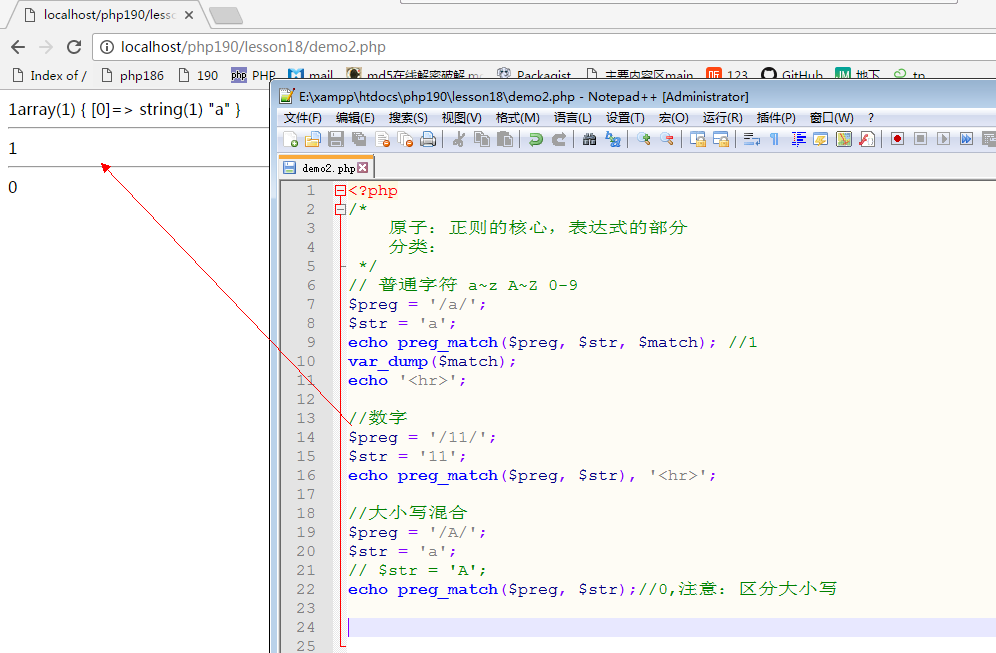
## 原子

* 就是正则的核心，要匹配的内容

### 分类

#### 普通字符

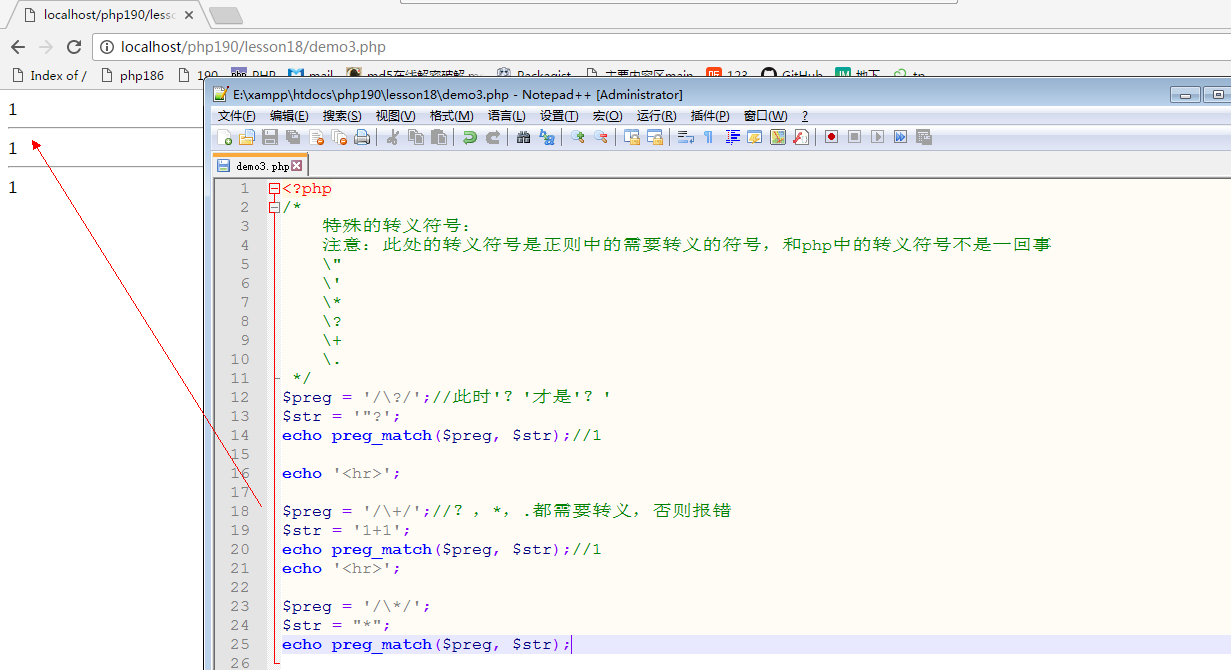
* A-Z ,a-z,0-9



* **注意：英文字母区分大小写**

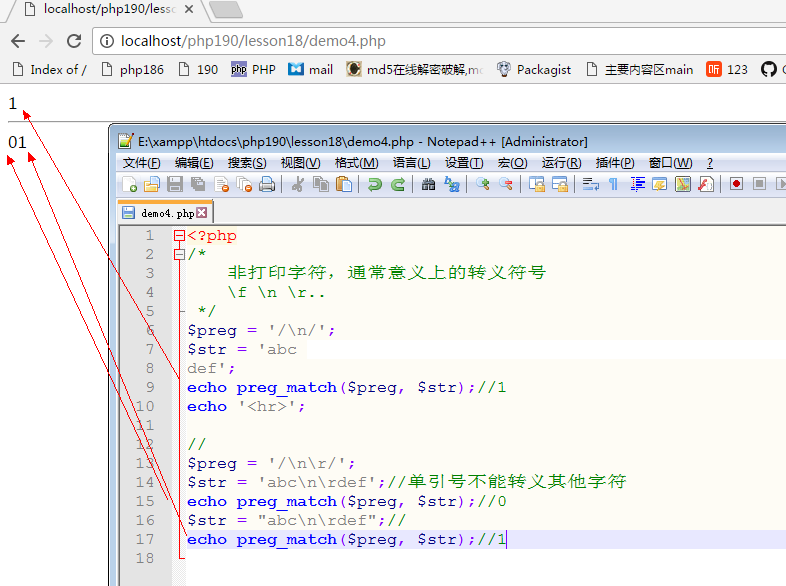
#### 转义字符—正则特有的特殊字符的转义

* \”
* \’
* \?
* \\*
* \+
* \.
* **注意：此处的转义字符和php中的转义字符不是一回事。他是指正则当中的转义符号。本身”,’,+,?,\*,.在正则中有特殊的含义**



#### 非打印字符—常规意义上的转义字符

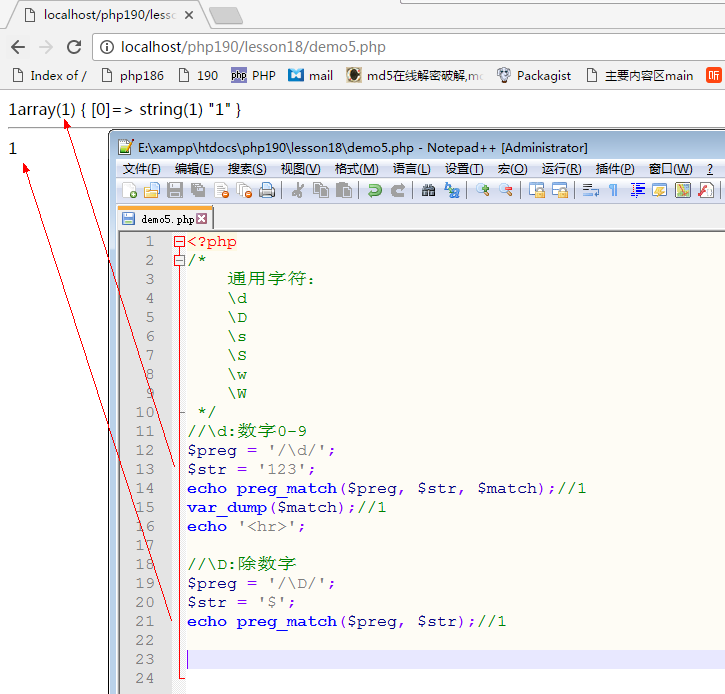
* \n
* \r
* \f
* \t
* ...



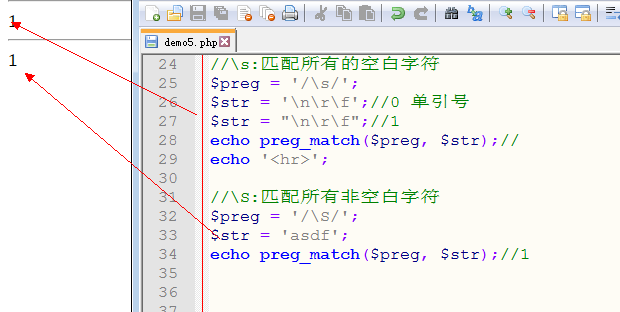
* **注意，如果字符串中有\n\r等转义符号，需要定义的时候使用双引号，否则不生效**

#### 通用原子—最常用

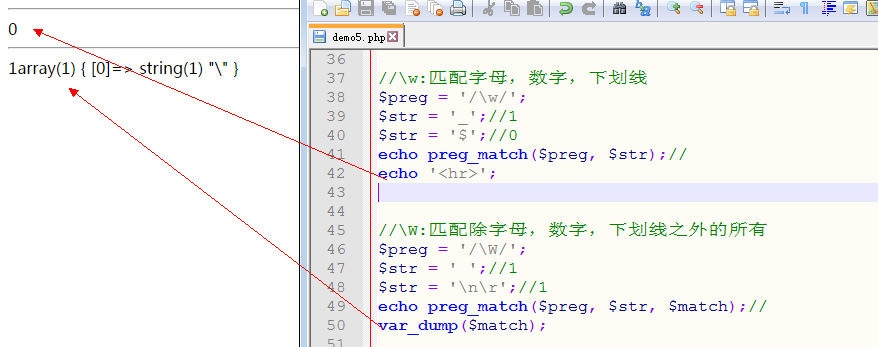
* **\d:匹配所有的数字，比如说：0-9**
* **\D：匹配所有的非数字**



* **\s：匹配所有的空白字符，也就是说：\n \r \f**
* **\S：匹配所有的非空白字符，除了转义字符**

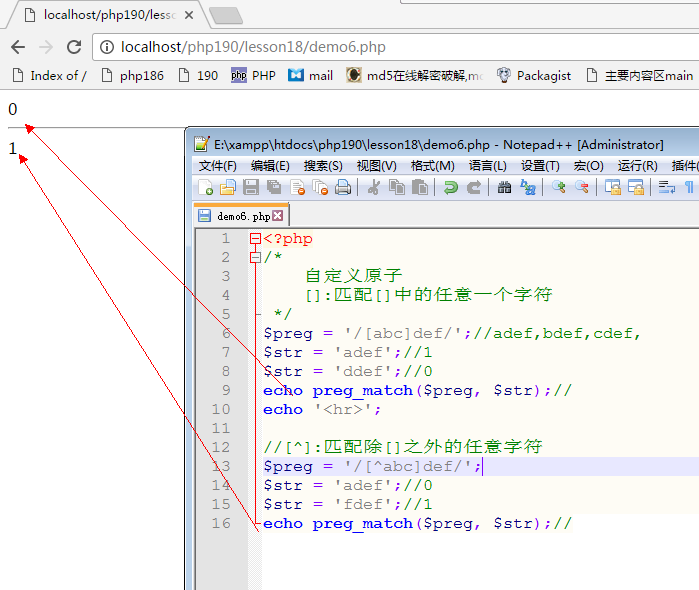


* **\w：匹配所有的字母，数字，下划线**
* **\W：匹配除字母，数字，下划线之外的所有字符**



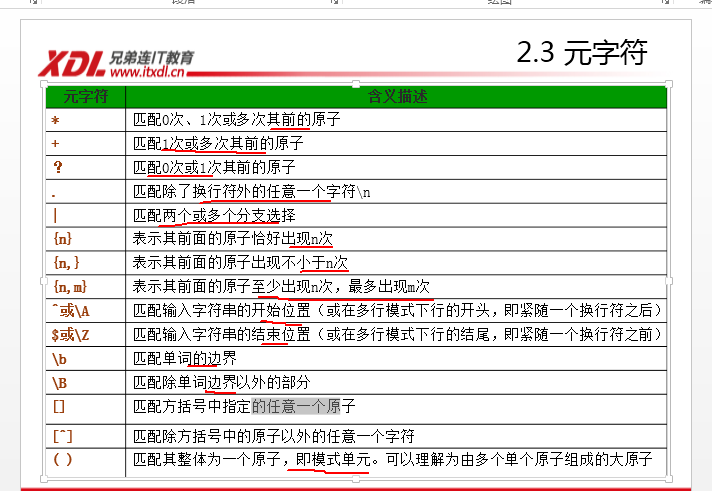
#### 自定义原子

* **使用[]把字符包括起来，匹配到的是[]里面的任意一个字符**
* **使用[^]把字符包含起来，表示，匹配除[]里面的任意一个字符**



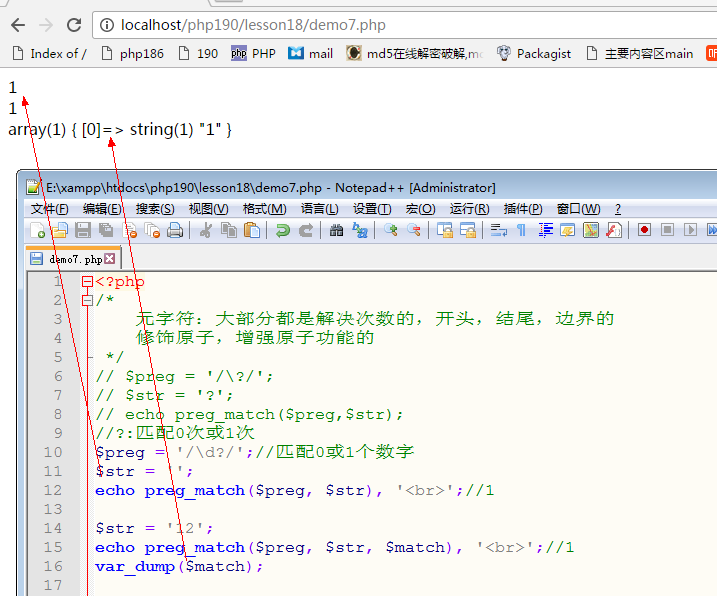
* **使用基本的原子是可以匹配字符，但是始终匹配到的是1个字符。并不能多次匹配，所以需要额外的东西来限定匹配的次数和长度，这个就是元字符**

## 元字符



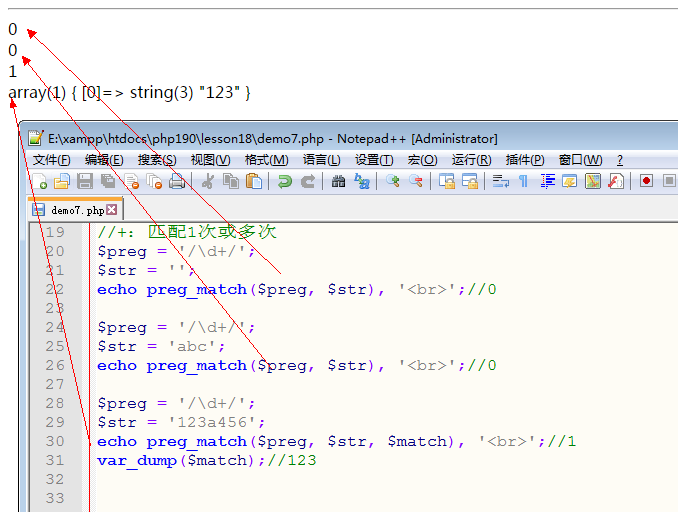
### ?:匹配0次或1次

* 结果几乎百分百会匹配到，因为可以匹配到0次



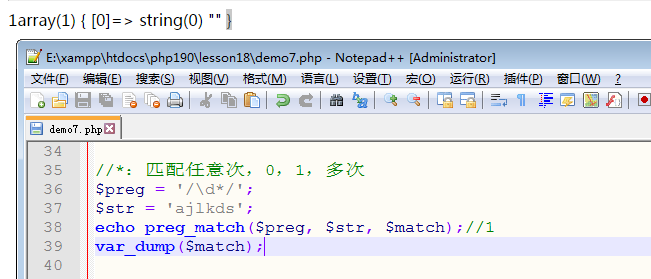
### +：匹配1次或多次

* 最少有1次

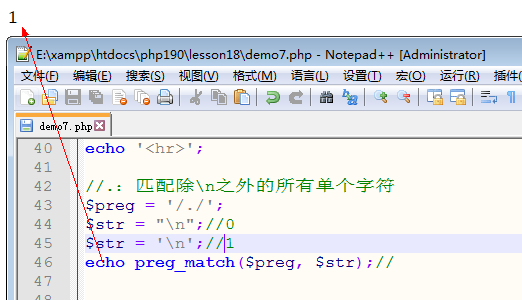


### \*:匹配0次，1次，多次

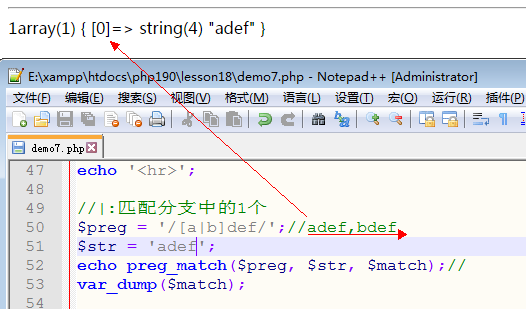
* 结果百分百能匹配到



### .:匹配除\n之外的任意1个字符



### |:匹配多个分支中的任意一个



### {n}:匹配正好n个前面的原子

