ICT 5 Web Development

Chapter 7. Regular Expressions

More String functions

- int strpos(string *str, string find [, int start])*

```
$numToPower = '20^2';
$caretPos = strpos($numToPower, '^');
$num = substr($numToPower, 0, $caretPos);
$power = substr($numToPower, $caretPos + 1);
echo "You're raising $num to the power of $power.";
```

- string str_replace(string *find, string replace, string str)*

```
$str = 'My dog knows a cat that knows the ferret
                           that stole my keys.';
$find = array('dog', 'cat', 'ferret');
echo str_replace($find, 'mammal', $str);
```

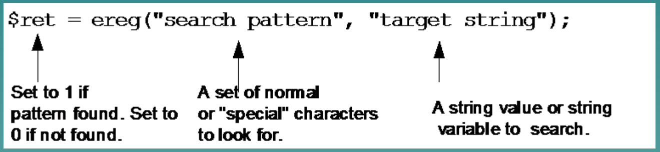
Why regular expressions?

- Scripting problem may require:
  - verification of input from form
    - was input a 7 digit phone number
  - parsing input from a file
    - `FirstName:LastName:Age:Salary`
- PHP supports three pattern matching functions:
  - `ereg()`, `split()`, and `ereg_replace()`
- Regular expressions are used to define very specific match patterns

New version of PHP:
ereg → preg_match **split → preg_split** **ereg_replace →preg_replace**

The ereg() function

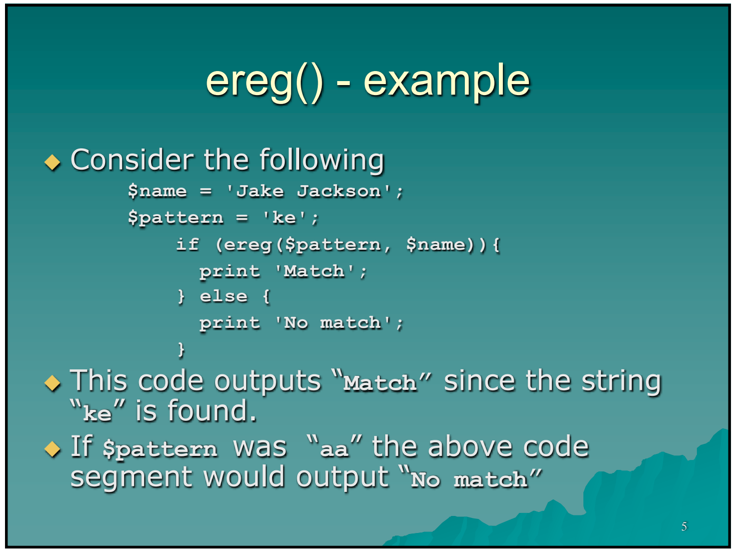
- Use ereg() to check if a string contains a match pattern:



```
$ret = ereg("search pattern", "target string");
```

Set to 1 if pattern found. Set to 0 if not found.
A set of normal or "special" characters to look for.
A string value or string variable to search.

$ret = preg_match('/search pattern/', 'target string')



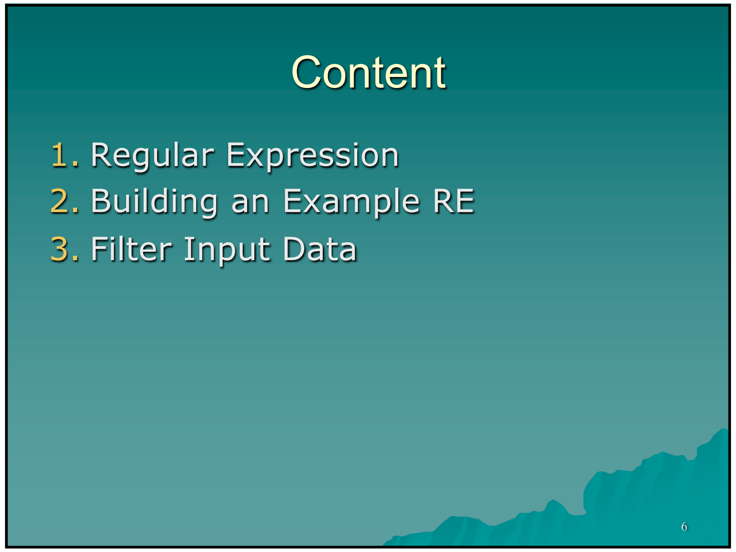
ereg() - example

- Consider the following

```
$name = 'Jake Jackson';
$pattern = 'ke';
    if (ereg($pattern, $name)){
      print 'Match';
    } else {
      print 'No match';
    }
```

- This code outputs "`Match`" since the string "`ke`" is found.
- If `$pattern` was "`aa`" the above code segment would output "`No match`"

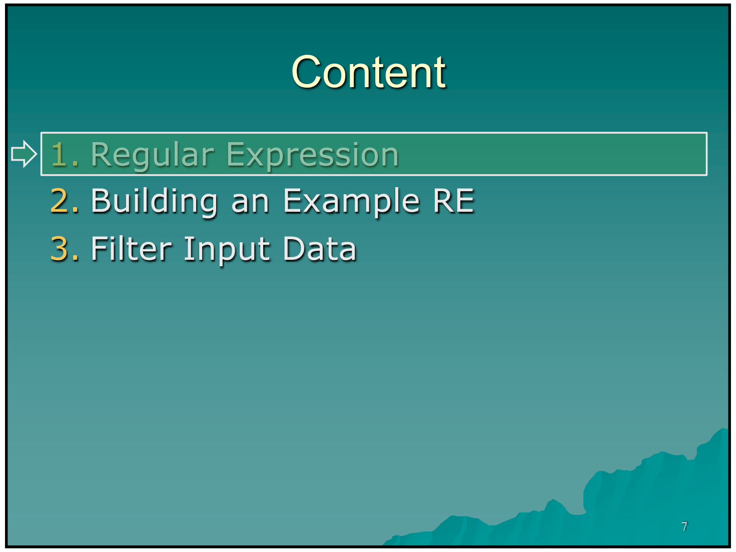
5



Content

1. Regular Expression
2. Building an Example RE
3. Filter Input Data

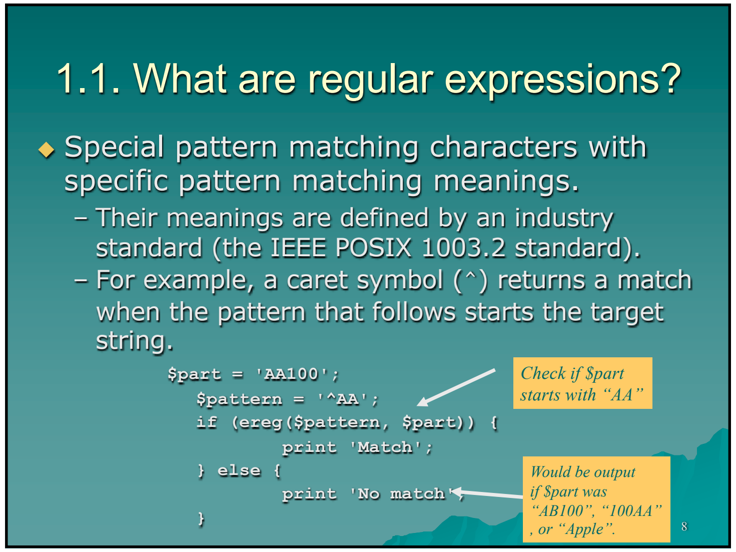
6



Content

1. Regular Expression
2. Building an Example RE
3. Filter Input Data

7



1.1. What are regular expressions?

- Special pattern matching characters with specific pattern matching meanings.
  - Their meanings are defined by an industry standard (the IEEE POSIX 1003.2 standard).
  - For example, a caret symbol (^) returns a match when the pattern that follows starts the target string.

```
$part = 'AA100';
  $pattern = '^AA';
  if (ereg($pattern, $part)) {
        print 'Match';
  } else {
        print 'No match';
  }
```

Check if $part starts with "AA"

Would be output if $part was "AB100", "100AA", or "Apple".

8

1.2. Selected Pattern Matching Characters

Symbol	Description
^	***Matches when the following character starts the string.*** ***E.g*** the following statement is *true* if $name contains "Smith is OK", "Smithsonian", or "Smith, Black". It would be *false* if $name contained only "SMITH" or "Smitty". `if (ereg('^Smith', $name)){`
$	*Matches when the preceding character ends the string.* ***E.g.*** the statement below would is *true* if $name contains "Joe Johnson", "Jackson", or "This is my son". It would be *false* if $name contained only "My son Jake" or "MY SON". `if (ereg('son$', $name )){`

1.2. Selected Pattern Matching Characters (2)

Symbol	Description
+	*Matches one or more occurrences of the preceding character.* For example, the statement below is *true* if $name contains "AB101", "ABB101", or "ABBB101 is the right part". It would be *false* if $name contained only "Part A101". `if(ereg( 'AB+101', $name)){`
*	*Matches zero or more occurrences of the preceding character.* For example, the statement below is *true* if $part starts with "A" and followed by zero or more "B" characters followed by "101", (for example, "AB101", "ABB101", "A101", or "A101 is broke"). It would be *false* if $part contained only "A11". `if (ereg( '^AB*101', $part)){`
?	Matches zero or one occurrences of the preceding character

1.2. Selected Pattern Matching Characters (3)

Symbol	Description
.	*A wildcard symbol that matches any one character.* For example, the statement is *true* if $name contains "Stop", "Soap", "Szxp", or "Soap is good". It would be *false* if $name contained only "Sxp". `if (ereg( '^S..p', $name)){`
|	*An alternation symbol that matches either character pattern.* For example, the statement below would be *true* if $name contains "www.mysite.com", "www.school.edu", "education", or "company". It would be *false* \ if $name contained only "www.site.net". `if (ereg('com|edu', $name)){`

For example ...

- Regular expressions are case sensitive by default

```
<br>Enter product code (Use AB## format):<br>
    <input type="text" size="6" name="code">
<br> Please enter description:<br>
    <input type="text" size="50" name="description">
```

- Asks for a product code and description (not to contain "Boat" or "Plane").

A Full Script Example

- Consider an example script that enables end-user to select multiple items from a checklist.
  - A survey about menu preferences
  - Wil look at how to send multiple items and how to receive them (later)

13

A Full Example ...

```
<html><head><title>Product Information Results </title>
</head><body>
<?php
  $products = array('AB01'=>'25-Pound Sledgehammer',
                    'AB02'=>'Extra Strong Nails',
                    'AB03'=>'Super Adjustable Wrench',
                    'AB04'=>'3-Speed Electric Screwdrive
  if (ereg('boat|plane', $description)){
     print 'Sorry, we do not sell boats or planes anymor
  } elseif  (ereg('^AB', $code)){
     if  (isset($products["$code"])){
        print "Code $code Description: $products[$code]
          } else {
             print 'Sorry, product code not found';
          }
       } else {
          print 'Sorry, all our product codes start with "AB"';

    } ?> </body></html>
```

Create a list of products.

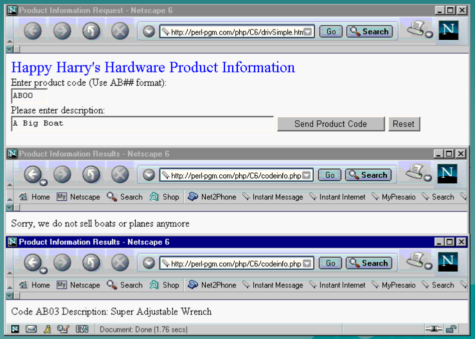
Check if "boat" or "plane".

Check if valid product number

14

The Output ...

The previous code can be executed at
http://webwizard.aw.com/~phppgm/C6/drivSimple.html



Product Information Request - Netscape 6
http://perl-pgm.com/php/C6/drivSimple.htm
Go
Search

Happy Harry's Hardware Product Information

Enter product code (Use AB## format):

AB00

Please enter description:

A Big Boat

Send Product Code
Reset
Product Information Results - Netscape 6
http://perl-pgm.com/php/C6/codeinfo.php
Home
My Netscape
Search
Shop
Net2Phone
Instant Message
Instant Internet
MyPresario

Sorry, we do not sell boats or planes anymore

Product Information Results - Netscape 6
http://perl-pgm.com/php/C6/codeinfo.php

Code AB03 Description: Super Adjustable Wrench

Document: Done (1.76 secs)

15

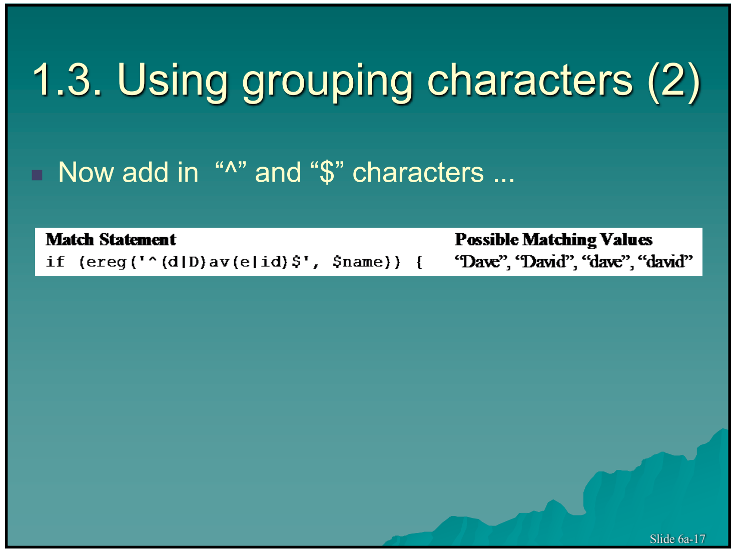
1.3. Using grouping characters

- Use parentheses to specify a group of characters in a regular expression.

Match Statement	Possible Matching Values
`if (ereg('Dav(e|id)', $name)) {`	"Dave", "David", "Dave was here"

- Above uses parentheses with "|" to indicate "Dav" can be followed by "e" or "id".

Slide 6a-16

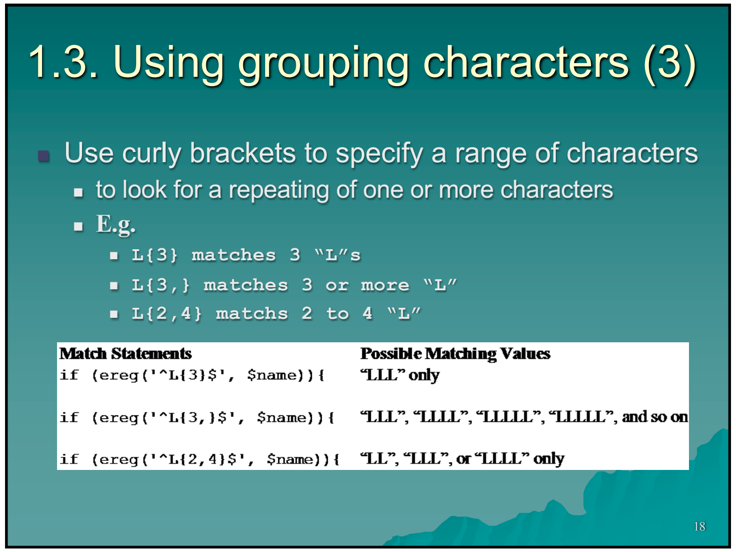


1.3. Using grouping characters (2)

- Now add in "^" and "$" characters ...

Match Statement	Possible Matching Values
`if (ereg('^(d|D)av(e|id)$', $name)) {`	"Dave", "David", "dave", "david"

Slide 6a-17

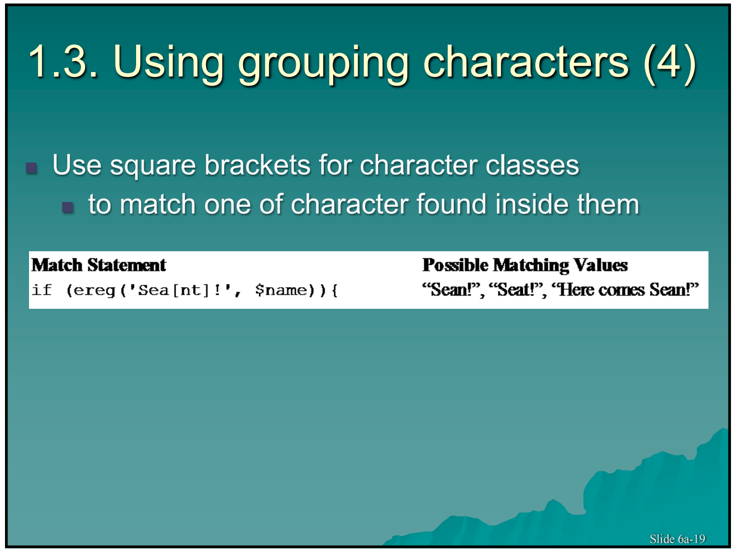


1.3. Using grouping characters (3)

- Use curly brackets to specify a range of characters
  - to look for a repeating of one or more characters
  - E.g.
    - `L{3} matches 3 "L"s`
    - `L{3,} matches 3 or more "L"`
    - `L{2,4} matchs 2 to 4 "L"`

Match Statements	Possible Matching Values
`if (ereg('^L{3}$', $name)){`	"LLL" only
`if (ereg('^L{3,}$', $name)){`	"LLL", "LLLL", "LLLLL", "LLLLLL", and so on
`if (ereg('^L{2,4}$', $name)){`	"LL", "LLL", or "LLLL" only

18

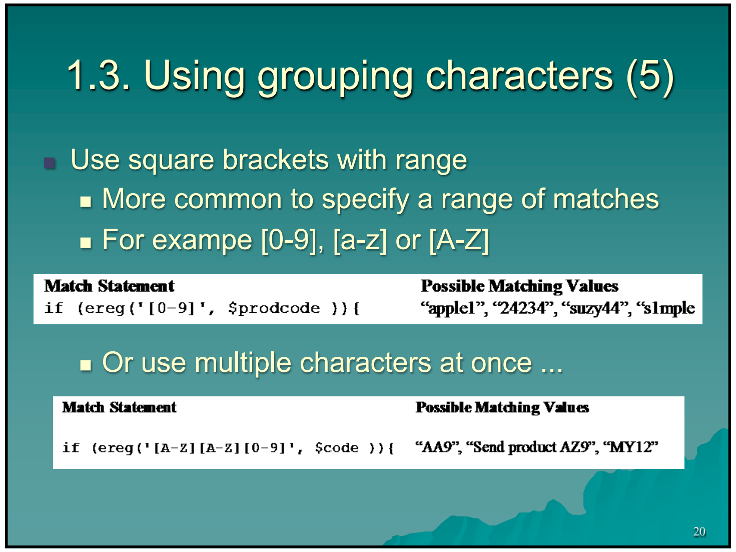


1.3. Using grouping characters (4)

- Use square brackets for character classes
  - to match one of character found inside them

Match Statement	Possible Matching Values
`if (ereg('Sea[nt]!', $name)){`	"Sean!", "Seat!", "Here comes Sean!"

Slide 6a-19



1.3. Using grouping characters (5)

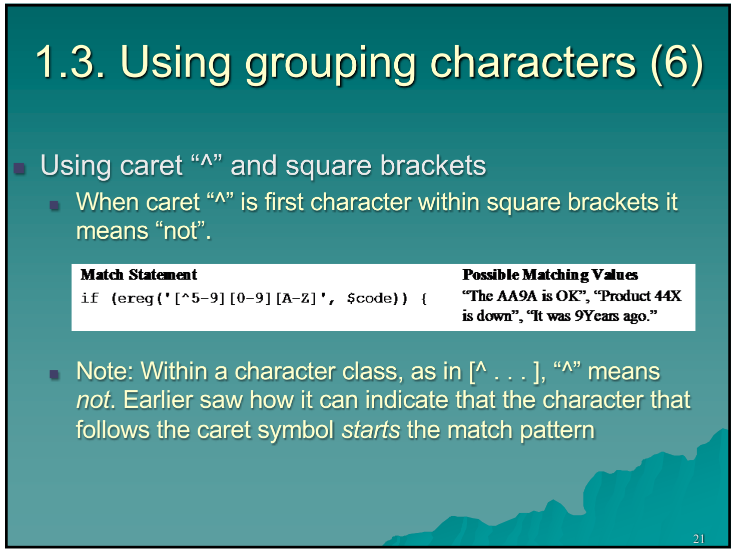
- Use square brackets with range
  - More common to specify a range of matches
  - For exampe [0-9], [a-z] or [A-Z]

Match Statement	Possible Matching Values
`if (ereg('[0-9]', $prodcode )){`	"apple1", "24234", "suzy44", "s1mple

- Or use multiple characters at once ...

Match Statement	Possible Matching Values
`if (ereg('[A-Z][A-Z][0-9]', $code )){`	"AA9", "Send product AZ9", "MY12"

20



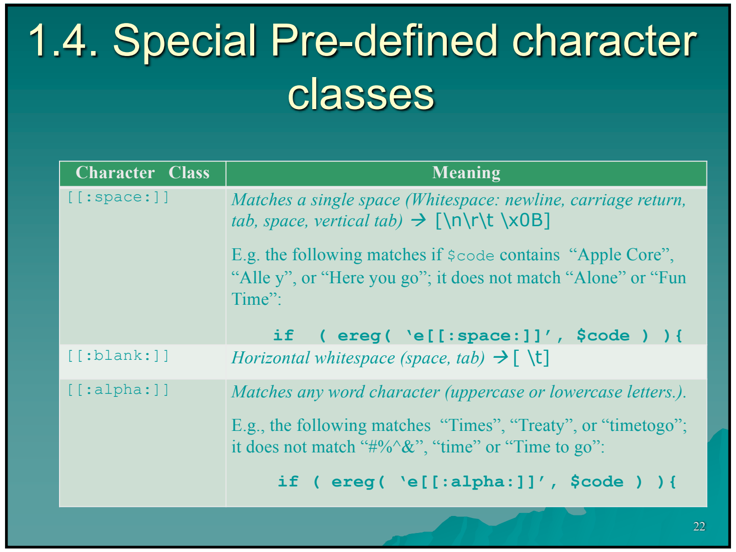
1.3. Using grouping characters (6)

- Using caret “^” and square brackets
  - When caret “^” is first character within square brackets it means “not”.

Match Statement	Possible Matching Values
`if (ereg('[^5-9][0-9][A-Z]', $code)) {`	“The AA9A is OK”, “Product 44X is down”, “It was 9Years ago.”

  - Note: Within a character class, as in [^ . . .], “^” means *not*. Earlier saw how it can indicate that the character that follows the caret symbol *starts* the match pattern

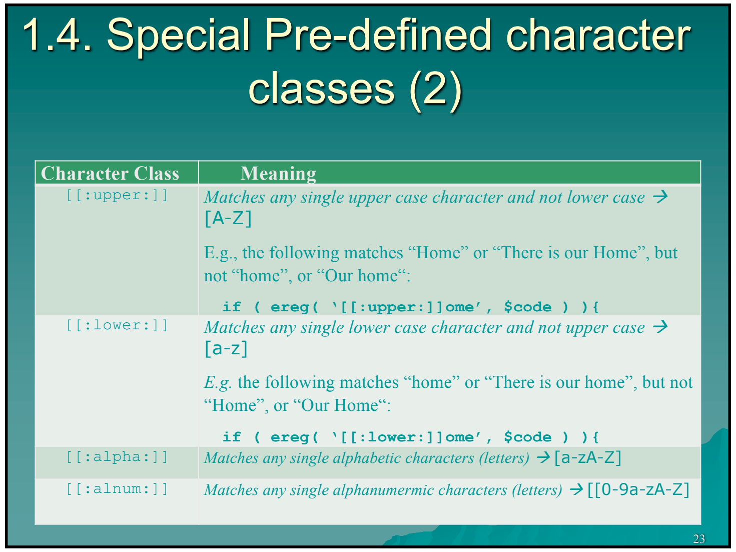
21



1.4. Special Pre-defined character classes

Character Class	Meaning
[[:space:]]	*Matches a single space (Whitespace: newline, carriage return, tab, space, vertical tab)* → [\n\r\t \x0B] E.g. the following matches if `$code` contains “Apple Core”, “Alle y”, or “Here you go”; it does not match “Alone” or “Fun Time”: `if ( ereg( ‘e[[:space:]]’, $code ) ){`
[[:blank:]]	*Horizontal whitespace (space, tab)* → [\t]
[[:alpha:]]	*Matches any word character (uppercase or lowercase letters.).* E.g., the following matches “Times”, “Treaty”, or “timetogo”; it does not match “#%^&”, “time” or “Time to go”: `if ( ereg( ‘e[[:alpha:]]’, $code ) ){`

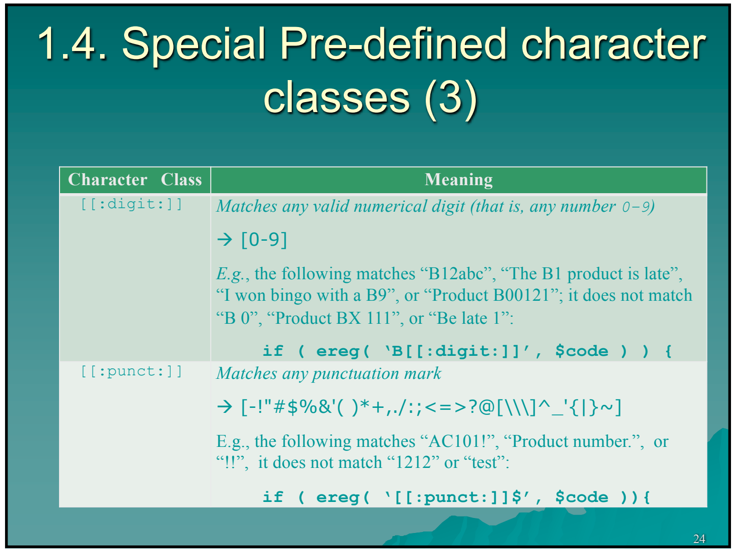
22



1.4. Special Pre-defined character classes (2)

Character Class	Meaning
[[:upper:]]	*Matches any single upper case character and not lower case* → [A-Z] E.g., the following matches “Home” or “There is our Home”, but not “home”, or “Our home“: `if ( ereg( ‘[[:upper:]]ome’, $code ) ){`
[[:lower:]]	*Matches any single lower case character and not upper case* → [a-z] *E.g.* the following matches “home” or “There is our home”, but not “Home”, or “Our Home“: `if ( ereg( ‘[[:lower:]]ome’, $code ) ){`
[[:alpha:]]	*Matches any single alphabetic characters (letters)* →[a-zA-Z]
[[:alnum:]]	*Matches any single alphanumermic characters (letters)* →[[0-9a-zA-Z]

23



1.4. Special Pre-defined character classes (3)

Character Class	Meaning
[[:digit:]]	*Matches any valid numerical digit (that is, any number 0-9)* → [0-9] *E.g.*, the following matches “B12abc”, “The B1 product is late”, “I won bingo with a B9”, or “Product B00121”; it does not match “B 0”, “Product BX 111”, or “Be late 1”: `if ( ereg( ‘B[[:digit:]]’, $code ) ) {`
[[:punct:]]	*Matches any punctuation mark* → [-!"#$%&'( )*+,./:;<=>?@[\\\]^_'{\|}~] E.g., the following matches “AC101!”, “Product number.”, or “!!”, it does not match “1212” or “test”: `if ( ereg( ‘[[:punct:]]$’, $code )){`

24

1.4. Special Pre-defined character classes (4)

Character Class	Meaning
`[[:<:]]`	*Matches when the following word starts the string.*
`[[:>:]]`	*Matches when the preceding word ends the string*

E.g.,

```
// returns false
ereg('[[:<:]]gun[[:>:]]', 'the Burgundy exploded');
// returns true
ereg('gun', 'the Burgundy exploded');
```

Content

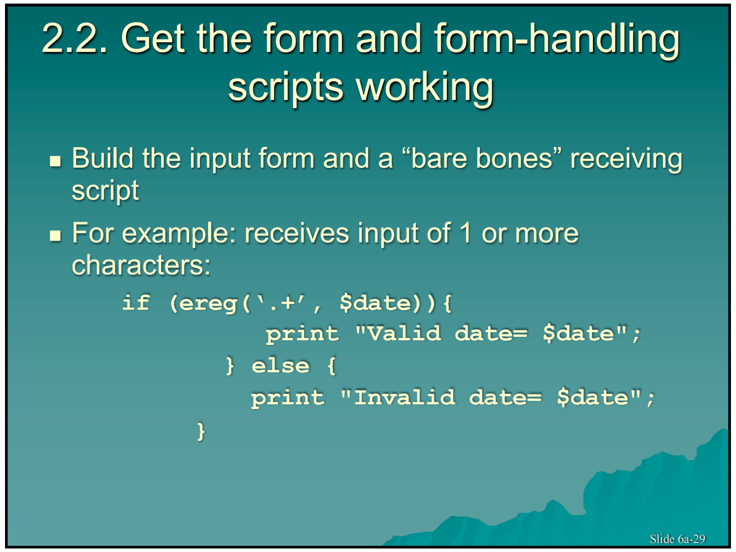
2. Building an example RE

- Building Regular expressions is best done incrementally
- Lets look at a process to build a regular expression to validate a date input field:
  - `mm/dd/yyyy` format (for example, `01/05/2002` but not `1/5/02`).

2.1. Determine the precise field rules

- What is valid input and invalid input
  - You might decide to allow `09/09/2002` but not `9/9/2002` or `Sep/9/2002` as valid date formats.
- Work through several examples as follows:

Rule	Reject These
1. Only accept "`/`" as a separator	`05 05 2002`—Require slash delimiters
2. Use a four-digit year	`05/05/02`—Four-digit year required
3. Only date data	`The date is 05/05/2002`—Only date fields allowed `05/05/2002 is my date`—Only date fields allowed
4.Require two digits for months and days	`5/05/2002`—Two-digit months required `05/5/2002`—Two-digit days required `5/5/2002`—Two-digit days and months required

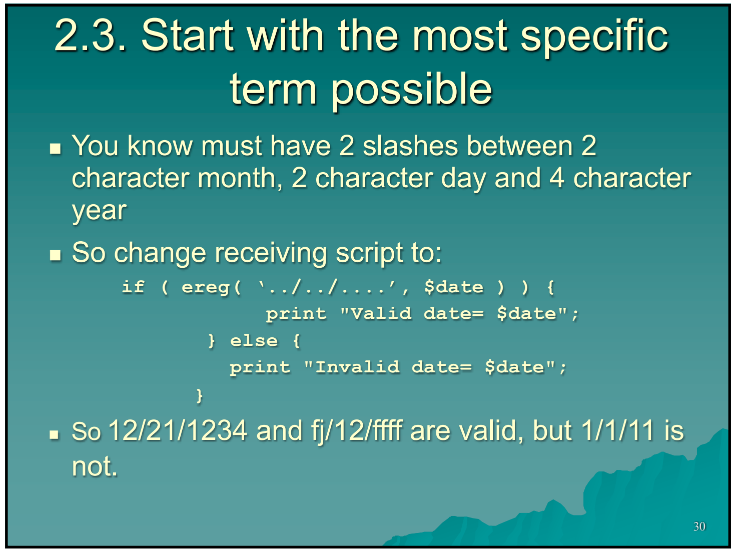


2.2. Get the form and form-handling scripts working

- Build the input form and a “bare bones” receiving script
- For example: receives input of 1 or more characters:

```
if (ereg(‘.+’, $date)){
          print "Valid date= $date";
       } else {
         print "Invalid date= $date";
     }
```

Slide 6a-29



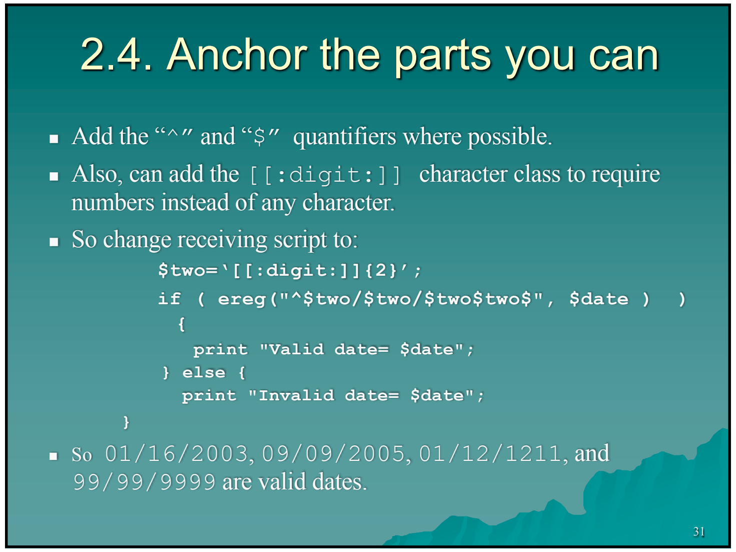
2.3. Start with the most specific term possible

- You know must have 2 slashes between 2 character month, 2 character day and 4 character year
- So change receiving script to:

```
if ( ereg( ‘../../....’, $date ) ) {
            print "Valid date= $date";
       } else {
         print "Invalid date= $date";
      }
```

- So 12/21/1234 and fj/12/ffff are valid, but 1/1/11 is not.

30



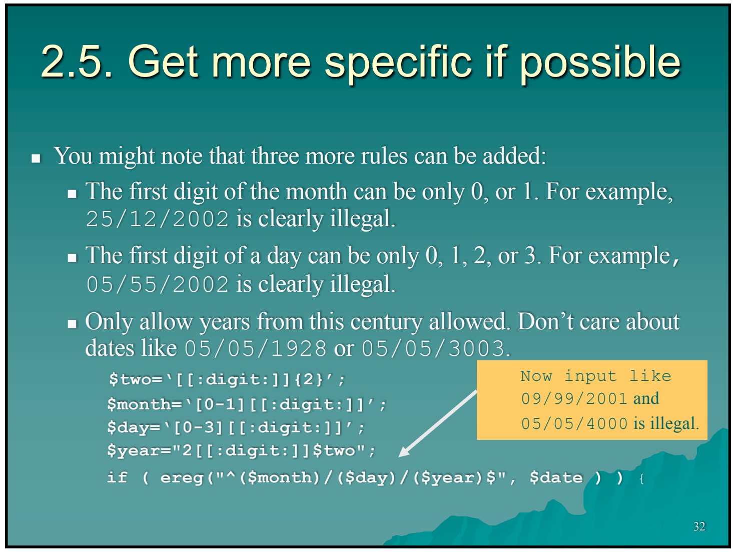
2.4. Anchor the parts you can

- Add the “^” and “$” quantifiers where possible.
- Also, can add the `[[:digit:]]` character class to require numbers instead of any character.
- So change receiving script to:

```
$two=‘[[:digit:]]{2}’;
if ( ereg("^$two/$two/$two$two$", $date )  )
  {
   print "Valid date= $date";
 } else {
  print "Invalid date= $date";
}
```

- So 01/16/2003, 09/09/2005, 01/12/1211, and 99/99/9999 are valid dates.

31



2.5. Get more specific if possible

- You might note that three more rules can be added:
  - The first digit of the month can be only 0, or 1. For example, 25/12/2002 is clearly illegal.
  - The first digit of a day can be only 0, 1, 2, or 3. For example, 05/55/2002 is clearly illegal.
  - Only allow years from this century allowed. Don’t care about dates like 05/05/1928 or 05/05/3003.

```
$two=‘[[:digit:]]{2}’;
$month=‘[0-1][[:digit:]]’;
$day=‘[0-3][[:digit:]]’;
$year="2[[:digit:]]$two";
if ( ereg("^($month)/($day)/($year)$", $date ) ) {
```

Now input like 09/99/2001 and 05/05/4000 is illegal.

32

A Full Script Example

- Consider an example script that asks end-user for a date
  - Use regular expressions to validate
  - Use the following HTML input

```
<input type="text" size="10" maxlength="10" name="date">
```

33

A Full Example ...

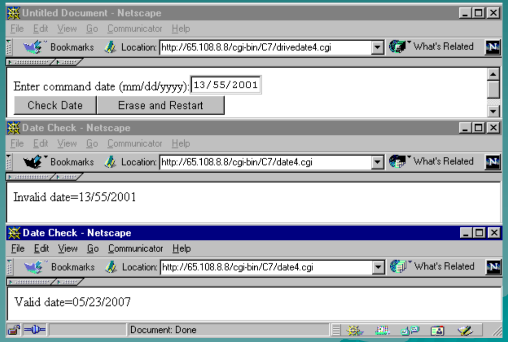
```
<html>
<head><title>Decsions</title></head>
<body>
<?php
        $two='[[:digit:]]{2}';
        $month='[0-3][[:digit:]]';
        $day='[0-3][[:digit:]]';
        $year="2[[:digit:]]$two";
        if ( ereg("^($month)/($day)/($year)$", $date )   ) {
                print "Got valid date=$date <br>";
         } else {
                print "Invalid date=$date";
         }
?> </body></html>
```

Use same regular expression as before

34

The Output ...

The previous code can be executed at
http://webwizard.aw.com/~phppgm/C6/drivedate4.cgi



Location: http://65.108.8.8/cgi-bin/C7/drivedate4.cgi

Enter command date (mm/dd/yyyy): 13/55/2001

Location: http://65.108.8.8/cgi-bin/C7/date4.cgi

Invalid date=13/55/2001

Location: http://65.108.8.8/cgi-bin/C7/date4.cgi

Valid date=05/23/2007

35

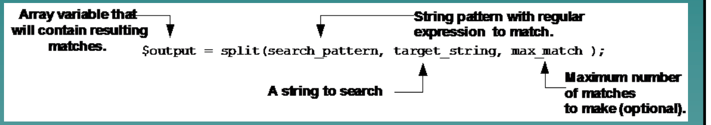
Content

1. Regular Expression
2. Building an Example RE
3. Filter Input Data

36

3.1. Matching Patterns With split()

- Use split() to break a string into different pieces based on the presence of a match pattern.



Array variable that will contain resulting matches.
String pattern with regular expression to match.
A string to search
Maximum number of matches to make (optional).

```
$output = split(search_pattern, target_string, max_match );
```

preg_split (string $pattern , string $subject [, int $limit = -1 [, int $flags = 0]])

3.1. Matching Patterns With split()

- Consider another example:

```
$line = 'Baseball, hot dogs, apple pie';
$item =  split( ',', $line );
print ("0=$item[0] 1=$item[1] 2=$item[2]");
```

- These lines will have the following output:

```
0=Baseball 1= hot dogs 2= apple pie
```

3.1. Matching Patterns With split()

- When you know how many patterns you are interested can use list() along with split():

```
line = 'AA1234:Hammer:122:12';
list($partno, $part, $num, $cost)
                         = split(':', $line, 4);
print "partno=$partno part=$part num=$num
                                           cost=$cost";
```

- The above code would output the following:

```
partno=AA1234 part=Hammer num=122 cost=12
```

Example of split()

- As an example of split() consider the following:

```
$line = 'Please , pass      thepepper';
$result =  split( '[[:space:]]+', $line );
```

- Will results in the following:

```
$result[0] = 'Please';
$result[1] = ','
$result[2] = 'pass';
$result[3] = 'thepepper';
```

A Full Script Example

- Consider an example script that updates the date checker just studied:
  - Uses split() to further refine date validation
  - Uses the same input form:

```
<input type="text" size="10" maxlength="10" name="date">
```

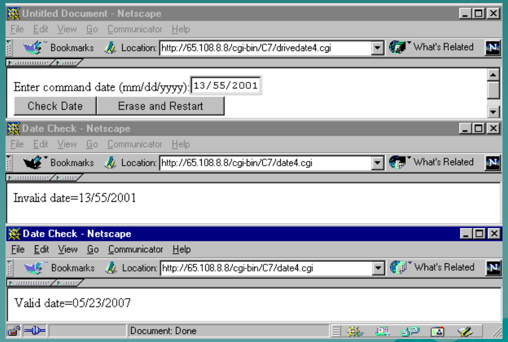
A Full Example ...

```
<html>
<head><title>Date Check</title></head>
<body>
<?php
    $two='[[:digit:]]{2}';
    $month='[0-3][[:digit:]]';
    $day='[0-3][[:digit:]]';
    $year="2[[:digit:]]$two";
    if ( ereg("^($month)/($day)/($year)$", $date )  ) {
        list($mon, $day, $year) = split( '/', $date );
        if ( $mon >= 1 && $mon <= 12 ) {
             if ( $day <= 31 ) {
                print "Valid date mon=$mon day=$day year=$year";
             } else {
                print " Illegal day specifed Day=$day";
             }
        } else {
           print " Illegal month specifed Mon=$mon";
        }
    } else {
        print ("Invalid date format= $date");
    }
 ?></body></html>
```

Use split() and list() to get month, day and year.

The Output ...

The previous code can be executed at
http://webwizard.aw.com/~phppgm/C6/drivedate4.cgi


Untitled Document - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://65.108.8.8/cgi-bin/C7/drivedate4.cgi What's Related
Check Date
Erase and Restart
Date Check - Netscape
Location: http://65.108.8.8/cgi-bin/C7/date4.cgi
Date Check - Netscape
Location: http://65.108.8.8/cgi-bin/C7/date4.cgi
Document: Done

Enter command date (mm/dd/yyyy): 13/55/2001

Invalid date=13/55/2001

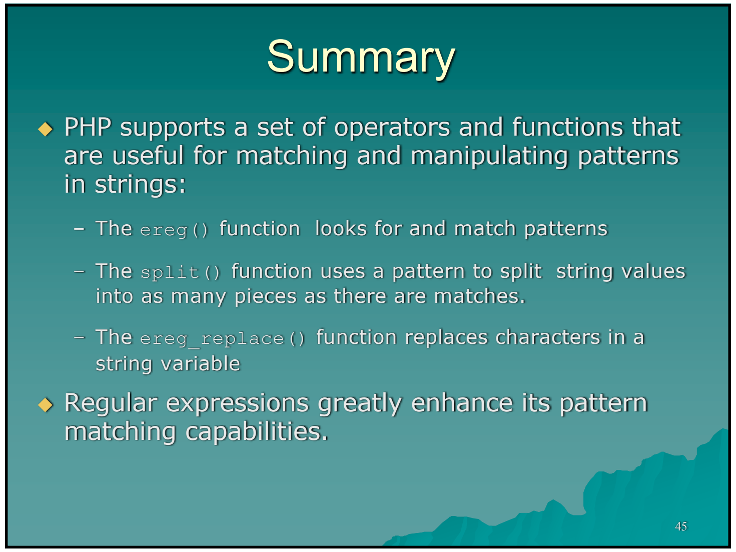
Valid date=05/23/2007

3.2. Using ereg_replace()

- Use `ereg_replace()` when replacing characters in a string variable.
  - It can be used to replace one string pattern for another in a string variable.
  - E.g:

```
$start = 'AC1001:Hammer:15:150';
$end = ereg_replace('Hammer', 'Drill', $start );
print "end=$end";
```

  - The above script segment would output:
    `end=AC1001:Drill:15:150`



Summary

- PHP supports a set of operators and functions that are useful for matching and manipulating patterns in strings:
  - The `ereg()` function looks for and match patterns
  - The `split()` function uses a pattern to split string values into as many pieces as there are matches.
  - The `ereg_replace()` function replaces characters in a string variable
- Regular expressions greatly enhance its pattern matching capabilities.

45