

Chapter 18: PHP

CS 80: Internet Programming

Instructor: Mark Edmonds

PHP

- Extremely common
- Open-source
- Platform independent

Preliminaries

- PHP handles client requests
- PHP is embedded into HTML documents, but executes on the server *before* the HTML document is delivered to the client
- PHP files have the extension .php

Preliminaries

- php code resides between `<?php /*PHP code */?>`
 - Single line php comments start with `//`
 - Multiline comments are enclosed with `/* */`
- Statements terminated with a semicolon `;` (required)

Variables

- Declared with `$name`
 - `name` must start with a letter or underscore
 - `name` can only contain `A-z`, `0-9`, and `_`

Variables

- Variables are case-sensitive
- Loosely typed
 - Similar idea as Javascript. Variables have types, but their type can change on the fly
 - In PHP, we have to explicitly change types

PHP Types

Type	Description
int, integer	Whole numbers (i.e., numbers without a decimal point).
float, double, real	Real numbers (i.e., numbers containing a decimal point).
string	Text enclosed in either single (' ') or double (" ") quotes. [<i>Note:</i> Using double quotes allows PHP to recognize more escape sequences.]
bool, boolean	true or false.
array	Group of elements.
object	Group of associated data and methods.
resource	An external source—usually information from a database.
NULL	No value.

Figure 1: PHP Types

Example: first.php

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.1: first.php -->
3 <!-- Simple PHP program. -->
4 <html>
5   <?php
6     $name = "Paul"; // declaration and initialization
7   ?><!-- end PHP script -->
8   <head>
9     <meta charset = "utf-8">
10    <title>Simple PHP document</title>
11  </head>
12  <body>
13    <!-- print variable name's value -->
14    <h1><?php print( "Welcome to PHP, $name!" ); ?></h1>
15  </body>
16 </html>
```

Important Notes

- Line 6 declares a php variable named `name` and sets it equal to Paul
- Line 14 prints text into the `<h1>` tag **before the file is served to the client**
 - Note that the value of `$name` is printed, not the string "`$name`"
 - Double quoted strings will have variables evaluated (called *interpolating* a variable)
 - Single quoted strings will have the entire string taken as a literal value (no interpolation)

Example: data.php

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.3: data.php -->
3 <!-- Data type conversion. -->
4 <html>
5   <head>
6     <meta charset = "utf-8">
7     <title>Data type conversion</title>
8     <style type = "text/css">
9       p
10      {
11        margin: 0;
12      }
13      .head
14      {
15        margin-top: 10px;
16        font-weight: bold;
17      }
18      .space
19      {
20        margin-top: 10px;
21      }
22    </style>
23  </head>
24  <body>
25    <?php
26      // declare a string, double and integer
27      $testString = "3.5 seconds";
28      $testDouble = 79.2;
29      $testInteger = 12;
30    ?><!-- end PHP script -->
31    <!-- print each variable's value and type -->
```

```
32     <p class = "head">Original values:</p>
33     <?php
34         print( "<p>$testString is a(n) " . gettype( $testString ) . "</p>" );
35         print( "<p>$testDouble is a(n) " . gettype( $testDouble ) . "</p>" );
36         print( "<p>$testInteger is a(n) " . gettype( $testInteger ) . "</p>" );
37     ?><!-- end PHP script -->
38     <p class = "head">Converting to other data types:</p>
39     <?php
40         // call function settype to convert variable
41         // testString to different data types
42         print( "<p>$testString " );
43         settype( $testString, "double" );
44         print( " as a double is $testString</p>" );
45         print( "<p>$testString " );
46         settype( $testString, "integer" );
47         print( " as an integer is $testString</p>" );
48         settype( $testString, "string" );
49         print( "<p class = 'space'>Converting back to a string results in $testString</p>" );
50         // use type casting to cast variables to a different type
51         $data = "98.6 degrees";
52         print( "<p class = 'space'>Before casting: $data is a " . gettype( $data ) . "</p>" );
53         print( "<p class = 'space'>Using type casting instead:</p>" );
54         print( "<p>as a double: " . (double) $data . "</p>" );
55         print( "<p>as an integer: " . (integer) $data . "</p>" );
56         print( "<p class = 'space'>After casting: $data is a " . gettype( $data ) . "</p>" );
57     ?><!-- end PHP script -->
58 </body>
59 </html>
```

Types and Conversion

- `gettype` gets the type of the parameter
- `settype` changes the type of first parameter to the second parameter
- Using `settype` can result in data loss: values may be truncated

- For example, converting 3.5 to an int yields 3, and converting the int back to a double yields 3
- Same thing happens with strings "3.5 seconds" as a double becomes 3.5

Types and Conversion

- Casting
 - Creates a temporary copy of the data before converting it
 - Means we won't lose data
 - Useful when a different type is required for a specific operation, but you want to retain the original value

String Concatenation

- Same as with javascript, but the operator is .

Constant values

- Created with `define("NAME", value);`
- Not a variable, a constant
- Used by simply typing `NAME` where you want the value

Conditionals

- Basically the same as Javascript, but `else if` is `elseif` (another keyword)

Arithmetic Operators

- Same as Javascript

Example: operators.php

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.4: operators.php -->
3 <!-- Using arithmetic operators. -->
4 <html>
```

```
5  <head>
6    <meta charset = "utf-8">
7    <style type = "text/css">
8      p { margin: 0; }
9    </style>
10   <title>Using arithmetic operators</title>
11 </head>
12 <body>
13   <?php
14     $a = 5;
15     print( "<p>The value of variable a is $a</p>" );
16
17     // define constant VALUE
18     define( "VALUE", 5 );
19
20     // add constant VALUE to variable $a
21     $a = $a + VALUE;
22     print( "<p>Variable a after adding constant VALUE is $a</p>" );
23
24     // multiply variable $a by 2
25     $a *= 2;
26     print( "<p>Multiplying variable a by 2 yields $a</p>" );
27
28     // test if variable $a is less than 50
29     if ( $a < 50 )
30     {
31       print( "<p>Variable a is less than 50</p>" );
32     }
33
34     // add 40 to variable $a
35     $a += 40;
36     print( "<p>Variable a after adding 40 is $a</p>" );
37
38     // test if variable $a is 50 or less
39     if ( $a < 51 )
40     {
41       print( "<p>Variable a is still 50 or less</p>" );
42     }
43     elseif ( $a < 101 )
44     {
45       // $a >= 51 and <= 100
46       print( "<p>Variable a is now between 50 and 100, inclusive</p>"
47             );
```

```
47     }
48     else // $a > 100
49     {
50         print( "<p>Variable a is now greater than 100</p>" );
51     }
52
53     // print an uninitialized variable
54     print( "<p>Using a variable before initializing: $nothing</p>" );
55     // nothing evaluates to ""
56
57     // add constant VALUE to an uninitialized variable
58     $test = $num + VALUE;
59
60     // num evaluates to 0
61     print( "<p>An uninitialized variable plus constant VALUE yields
62           $test</p>" );
63
64     // add a string to an integer
65     $str = "3 dollars";
66     $a += $str;
67     print( "<p>Adding a string to variable a yields $a</p>" );
68     ?><!-- end PHP script -->
69 </body>
70 </html>
```

Arrays

- PHP also supports arrays
 - Note that if an array does not exist, but is assigned, the array will be created
- PHP arrays may be *associative arrays*, meaning they have non-integer indices
 - E.g. you index an array by a name, or by student ID number (stored as a string)

Arrays

- `reset` resets the internal pointer of the array to the beginning of the array
 - `key` returns the index of the element pointed to by the internal pointer
 - `next` moves the internal pointer down one element of the array
- `foreach` is specifically for iterating through arrays
 - `as` divides the key/value (key is on the left, value is on the right)

Example: array.php

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.7: arrays.php -->
3 <!-- Array manipulation. -->
4 <html>
5 <head>
6 <meta charset = "utf-8">
7 <title>Array manipulation</title>
8 <style type = "text/css">
9 p
10 { margin: 0; }
11 .head { margin-top: 10px; font-weight: bold; }
12 </style>
13 </head>
14 <body>
15 <?php
16     // create array first
17     print( "<p class = 'head'>Creating the first array</p>" );
18     $first[ 0 ] = "zero";
19     $first[ 1 ] = "one";
20     $first[ 2 ] = "two";
21     $first[] = "three";
22     // print each element's index and value
23     for ( $i = 0; $i < count( $first ); ++$i )
24         print( "Element $i is $first[$i]</p>" );
25     print( "<p class = 'head'>Creating the second array</p>" );
26     // call function array to create array second
27     $second = array( "zero", "one", "two", "three" );
28     for ( $i = 0; $i < count( $second ); ++$i )
29         print( "Element $i is $second[$i]</p>" );
30     print( "<p class = 'head'>Creating the third array</p>" );
31     // assign values to entries using nonnumeric indices
32     $third[ "Amy" ] = 21;
33     $third[ "Bob" ] = 18;
34     $third[ "Carol" ] = 23;
35     // iterate through the array elements and print each
36     // element's name and value
37     for ( reset( $third ); $element = key( $third ); next( $third ) )
38         print( "<p>$element is $third[$element]</p>" );
39     print( "<p class = 'head'>Creating the fourth array</p>" );
40     // call function array to create array fourth using
```



```
41 // string indices
42 $fourth = array(
43     "January" => "first",
44     "February" => "second",
45     "March" => "third",
46     "April" => "fourth",
47     "May" => "fifth",
48     "June" => "sixth",
49     "July" => "seventh",
50     "August" => "eighth",
51     "September" => "ninth",
52     "October" => "tenth",
53     "November" => "eleventh",
54     "December" => "twelfth" );
55 // print each element's name and value
56 foreach ( $fourth as $element => $value )
57     print( "<p>$element is the $value month</p>" );
58 ?><!-- end PHP script -->
59 </body>
60 </html>
```

String Comparisons

- `strcmp` compares two strings.
 - returns -1 if the first string alphabetically precedes the second string
 - returns 0 if the two strings are equal
 - returns 1 if the first string alphabetically follows the second string
- Can also use relational operators
 - `==`, `!=`, `<`, `<=`, `>`, `>=`

Example: `compare.php`

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.8: compare.php -->
3 <!-- Using the string-comparison operators. -->
4 <html>
5     <head>
6         <meta charset = "utf-8">
7         <title>String Comparison</title>
8         <style type = "text/css">
```

```
9     p { margin: 0; }
10    </style>
11    </head>
12    <body>
13    <?php
14        // create array fruits
15        $fruits = array( "apple", "orange", "banana" );
16        // iterate through each array element
17        for ( $i = 0; $i < count( $fruits ); ++$i )
18        {
19            // call function strcmp to compare the array element
20            // to string "banana"
21            if (strcmp( $fruits[ $i ], "banana" ) < 0) {
22                print( "<p>" . $fruits[ $i ] . " is less than banana " );
23            } elseif ( strcmp( $fruits[ $i ], "banana" ) > 0 ) {
24                print( "<p>" . $fruits[ $i ] . " is greater than banana " );
25            } else {
26                print( "<p>" . $fruits[ $i ] . " is equal to banana " );
27            }
28            // use relational operators to compare each element
29            // to string "apple"
30            if ( $fruits[ $i ] < "apple" )
31                print( "and less than apple!</p>" );
32            elseif ( $fruits[ $i ] > "apple" )
33                print( "and greater than apple!</p>" );
34            elseif ( $fruits[ $i ] == "apple" )
35                print( "and equal to apple!</p>" );
36        } // end for
37    ?><!-- end PHP script -->
38    </body>
39 </html>
```

Regular Expressions

- There is no escape from regular expressions
- php uses the `preg_match` function to search for a string with the specified pattern

Regular Expressions

- Important regex characters

- ^ means beginning of line
- \$ means end of line
- [] denotes a bracket expression
 - * lists of characters
 - * can specify a range with -
 - * E.g. [a-z] are all characters a through z

Regular Expressions

- Quantifiers
 - specifies a quantity to match with the previous expression
 - * means 'zero or more times'
 - + means 'one or more times'
 - ? means 'zero or one times'
 - {n} means 'exactly n times'
 - {m,n} means 'between m and n times'
 - {n,} means 'n or more times'

Character Classes

Character class	Description
alnum	Alphanumeric characters (i.e., letters [a-zA-Z] or digits [0-9])
alpha	Word characters (i.e., letters [a-zA-Z])
digit	Digits
space	White space
lower	Lowercase letters
upper	Uppercase letters

Figure 2: Regex Character Classes

Example: expression.php

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.9: expression.php -->
3 <!-- Regular expressions. -->
```

```
4 <html>
5   <head>
6     <meta charset = "utf-8">
7     <title>Regular expressions</title>
8     <style type = "text/css">
9       p { margin: 0; }
10    </style>
11  </head>
12  <body>
13    <?php
14      $search = "Now is the time";
15      print( "<p>Test string is: '$search'</p>" );
16      // call preg_match to search for pattern 'Now' in variable search
17      if (
18        preg_match( "/Now/", $search )
19      )
20        print( "<p>'Now' was found.</p>" );
21      // search for pattern 'Now' in the beginning of the string
22      if (
23        preg_match( "/^Now/", $search )
24      )
25        print( "<p>'Now' found at beginning of the line.</p>" );
26      // search for pattern 'Now' at the end of the string
27      if (
28        !preg_match( "/Now$/", $search )
29      )
30        print( "<p>'Now' was not found at the end of the line.</p>" );
31      // search for any word ending in 'ow'
32      if (
33        preg_match( "/\b([a-zA-Z]*ow)\b/i", $search, $match )
34      )
35        print( "<p>Word found ending in 'ow': " .
36          $match[ 1 ]
37          . "</p>" );
38      // search for any words beginning with 't'
39      print( "<p>Words beginning with 't' found: " );
40      while (
41        preg_match( "/\b(t[[:alpha:]]+)\b/", $search, $match )
42      )
43      {
44        print(
45          $match[ 1 ]
46          . " " );
```

```
47     // remove the first occurrence of a word beginning
48     // with 't' to find other instances in the string
49     $search = preg_replace("/" . $match[ 1 ] . "/", "", $search);
50 } // end while
51 print( "</p>" );
52 ?><!-- end PHP script -->
53 </body>
54 </html>
```

Form Processing

Superglobal Arrays

- Special arrays that contain client information
- Client information includes:
 - Client's web browser
 - Data sent to the server by the client `$_GET` and `$_POST`
 - * E.g. if the user submit's a form and it is posted to a script (remember the `action` attribute), then the information is available in the `$_POST` array
 - Cookies

Superglobal Arrays

Variable name	Description
<code>\$_SERVER</code>	Data about the currently running server.
<code>\$_ENV</code>	Data about the client's environment.
<code>\$_GET</code>	Data sent to the server by a get request.
<code>\$_POST</code>	Data sent to the server by a post request.
<code>\$_COOKIE</code>	Data contained in cookies on the client's computer.
<code>\$GLOBALS</code>	Array containing all global variables.

Figure 3: Superglobal arrays

Example: `form.html`

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.13: form.html -->
3 <!-- HTML form for gathering user input. -->
4 <html>
5 <head>
6   <meta charset="utf-8">
7   <title>Sample Form</title>
8   <style type="text/css">
9     label {
10       width: 5em;
11       float: left;
12     }
13   </style>
14 </head>
15 <body>
16   <h1>Registration Form</h1>
17   <p>Please fill in all fields and click Register.</p>
18   <!-- post form data to form.php -->
19   <form method="post" action="form.php">
20     <h2>User Information</h2>
21     <!-- create four text boxes for user input -->
22     <div>
23       <label>First name:</label>
24       <input type="text" name="fname">
25     </div>
26     <div>
27       <label>Last name:</label>
28       <input type="text" name="lname">
29     </div>
30     <div>
31       <label>Email:</label>
32       <input type="text" name="email">
33     </div>
34     <div>
35       <label>Phone:</label>
36       <input type="text" name="phone" placeholder="(555) 555-5555">
37     </div>
38     <h2>Publications</h2>
39     <p>Which book would you like information about?</p>
40     <!-- create drop-down list containing book names -->
41     <select name="book">
42       <option>Internet and WWW How to Program</option>
```

```
43     <option>C++ How to Program</option>
44     <option>Java How to Program</option>
45     <option>Visual Basic How to Program</option>
46 </select>
47 <h2>Operating System</h2>
48 <p>Which operating system do you use?</p>
49 <!-- create five radio buttons -->
50 <p>
51     <input type = "radio" name = "os" value = "Windows" checked>
        Windows
52     <input type = "radio" name = "os" value = "Mac OS X">Mac OS X
53     <input type = "radio" name = "os" value = "Linux">Linux
54     <input type = "radio" name = "os" value = "Other">Other
55 </p>
56 <!-- create a submit button -->
57 <p>
58     <input type = "submit" name = "submit" value = "Register">
59 </p>
60 </form>
61 </body>
62 </html>
```

Form Processing

- Let's break down this form (it's been a while)
 - It uses the **POST** HTTP method to send data to `form.php`
 - It has the following inputs:
 - * `fname` (text)
 - * `lname` (text)
 - * `email` (text)
 - * `phone` (text)
 - * `book` (options)
 - * `os` (radio)

Form Processing

- So when we hit `Register` (the `submit` input), we will send the inputs to `form.php` using the `$_POST` superarray
 - Had we used the **GET** method we'd see values in the `$_GET` superarray

- The input names are the glue; they register an input to an entry in the superarray
 - This is why names mattered in chapter 2/3!!

Example: form.php

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.14: form.php -->
3 <!-- Process information sent from form.html. -->
4 <html>
5   <head>
6     <meta charset = "utf-8">
7     <title>Form Validation</title>
8     <style type = "text/css">
9       p
10      { margin: 0px; }
11      .error
12      { color: red }
13      p.head { font-weight: bold; margin-top: 10px; }
14    </style>
15  </head>
16  <body>
17    <?php
18      // determine whether phone number is valid and print
19      // an error message if not
20      // regex looks for the following pattern "(###) ###-####"
21      if (!preg_match( "/^\([0-9]{3}\) [0-9]{3}-[0-9]{4}$/", $_POST["
22        phone"]))
23      {
24        print( "<p class = 'error'>Invalid phone number</p>
25              <p>A valid phone number must be in the form
26              (555) 555-5555</p><p>Click the Back button,
27              enter a valid phone number and resubmit.</p>
28              <p>Thank You.</p></body></html>" );
29        die(); // terminate script execution
30      }
31    ?><!-- end PHP script -->
32    <p>
33      <!-- Access information from the submission using the $_POST
34        superarray -->
35      Hi <?php print( $_POST["fname"] ); ?>. Thank you for completing
36      the survey. You have been added to the
37      <?php print( $_POST["book"] ); ?>mailing list.
```



```
35     </p>
36     <p class = "head">
37         The following information has been saved in our database:
38     </p>
39     <p>Name: <?php print( $_POST["fname"] ); print( " " . $_POST["lname
        "] ); ?></p>
40     <p>Email: <?php print( $_POST["email"] ); ?></p>
41     <p>Phone: <?php print( $_POST["phone"] ); ?></p>
42     <p>OS: <?php print( $_POST["os"] ); ?></p>
43     <p class = "head">
44         This is only a sample form. You have not been added to a mailing
            list.
45     </p>
46 </body>
47 </html>
```

Form Processing

- This validates the phone number!
 - Very important to validate your form inputs
- `die()` terminates the script, stops processing the form

Example: data.html

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.15: data.html -->
3 <!-- Form to query a MySQL database. -->
4 <html>
5     <head>
6         <meta charset = "utf-8">
7         <title>Sample Database Query</title>
8     </head>
9     <body>
10        <h1>Querying a MySQL database.</h1>
11        <form method = "post" action = "database.php">
12            <p>Select a field to display:
13            <!-- add a select box containing options -->
14            <!-- for SELECT query -->
15            <select name = "select">
16                <option selected>*</option>
```

```
17         <option>ID</option>
18         <option>Title</option>
19         <option>Category</option>
20         <option>ISBN</option>
21     </select>
22 </p>
23 <p>
24     <input type = "submit" value = "Send Query">
25 </p>
26 </form>
27 </body>
28 </html>
```

Example: database.php

```
1 <!DOCTYPE html>
2 <!-- Fig. 19.16: database.php -->
3 <!-- Querying a database and displaying the results. -->
4 <html>
5     <head>
6         <meta charset = "utf-8">
7         <title>Search Results</title>
8         <style type = "text/css">
9             body
10             { font-family: sans-serif;
11               background-color: lightyellow; }
12             table { background-color: lightblue;
13                   border-collapse: collapse;
14                   border: 1px solid gray; }
15             td
16             { padding: 5px; }
17             tr:nth-child(odd) {
18                 background-color: white; }
19         </style>
20     </head>
21     <body>
22         <?php
23             $select = $_POST["select"]; // creates variable $select
24             // build SELECT query
25             $query = "SELECT " . $select . " FROM books";
26             // Connect to MySQL
```

```
27     if ( !( $database = mysqli_connect( "localhost", "iw3htp", "
        password" ) ) )
28         die( "Could not connect to database </body></html>" );
29     // open Products database
30     if ( !mysqli_select_db($database, "products") )
31         die( "Could not open products database </body></html>" );
32     // query Products database
33     if ( !( $result = mysqli_query($database, $query) ) )
34     {
35         print( "<p>Could not execute query!</p>" );
36         die( mysqli_error() . "</body></html>" );
37     } // end if
38     mysqli_close( $database );
39 ?><!-- end PHP script -->
40 <table>
41     <caption>Results of "SELECT <?php print( "$select" ) ?>
42     FROM books"</caption>
43     <?php
44         // fetch each record in result set
45         while (
46             $row = mysqli_fetch_row( $result )
47         )
48         {
49             // build table to display results
50             print( "<tr>" );
51             foreach ( $row as $key => $value )
52                 print( "<td>$value</td>" );
53             print( "</tr>" );
54         } // end while
55     ?><!-- end PHP script -->
56 </table>
57 <p>
58     Your search yielded <?php print( mysqli_num_rows( $result ) ) ?>
59     results.
60 </p>
61     Please email comments to <a href = "mailto:deitel@deitel.com">
62     Deitel and Associates, Inc.</a>
63 </p>
64 </body>
</html>
```

Database Processing

- This assumed we followed the Chapter 18 instructions for setting up MySQL
 - Includes source-ing the `products.sql` file
- `mysqli_connect` connects to the database
- `mysqli_select_db` opens the `products` database
- `mysqli_query` executes a MySQL query (what we learned about in chapter 18)
- `mysqli_close` closes the database

Database Processing

- `mysqli_fetch_row` returns an associative array containing the column of the current row from the query result
 - The `key` is a unique column ID for the query
- `mysqli_fetch_assoc` returns an associative array where the column names are the keys storing the corresponding values
- `mysqli_num_rows` stores the number of rows in the query result

Cookies

- What is a cookie?
 - A piece of information from the server that resides on the client's computer
 - Just a text file
 - Maintains information about the client in between browsing sessions
 - * Cookies mean you don't have to login everytime you visit a website
 - * The cookie stores your login session (not password), basically meaning the website assumes you are the same user
 - You can disable cookies if you want, but it makes browsing significantly more annoying!
 - Can also track other client activity

Cookies

- Cookies are text files
 - Should never store passwords, credit card info, etc
- Cookies are only accessible by the website that placed the cookie on the client's computer
- Cookies have an expiration date - at which point the browser will delete the cookie off of the client's computer

- Cookies are sent back to the originating server when the user connects to that server

Example: cookies.html

```
1 <!DOCTYPE html>
2
3 <!-- Fig. 19.17: cookies.html -->
4 <!-- Gathering data to be written as a cookie. -->
5 <html>
6   <head>
7     <meta charset = "utf-8">
8     <title>Writing a cookie to the client computer</title>
9     <style type = "text/css">
10       label { width: 7em; float: left; }
11     </style>
12   </head>
13   <body>
14     <h2>Click Write Cookie to save your cookie data.</h2>
15     <form method = "post" action = "cookies.php">
16       <div><label>Name:</label>
17         <input type = "text" name = "name"></div>
18       <div><label>Height:</label>
19         <input type = "text" name = "height"></div>
20       <div><label>Favorite Color:</label>
21         <input type = "text" name = "color"></div>
22       <p><input type = "submit" value = "Write Cookie">
23     </form>
24   </body>
25 </html>
26
27 <!--
28 *****
29
30 * (C) Copyright 1992-2012 by Deitel & Associates, Inc. and
31   *
32   * Pearson Education, Inc. All Rights Reserved.
33   *
34   *
35   *
36   *
37   *
38   *
39   *
40   *
41   *
42   *
43   *
44   *
45   *
46   *
47   *
48   *
49   *
50   *
51   *
52   *
53   *
54   *
55   *
56   *
57   *
58   *
59   *
60   *
61   *
62   *
63   *
64   *
65   *
66   *
67   *
68   *
69   *
70   *
71   *
72   *
73   *
74   *
75   *
76   *
77   *
78   *
79   *
80   *
81   *
82   *
83   *
84   *
85   *
86   *
87   *
88   *
89   *
90   *
91   *
92   *
93   *
94   *
95   *
96   *
97   *
98   *
99   *
100  *
101  *
102  *
103  *
104  *
105  *
106  *
107  *
108  *
109  *
110  *
111  *
112  *
113  *
114  *
115  *
116  *
117  *
118  *
119  *
120  *
121  *
122  *
123  *
124  *
125  *
126  *
127  *
128  *
129  *
130  *
131  *
132  *
133  *
134  *
135  *
136  *
137  *
138  *
139  *
140  *
141  *
142  *
143  *
144  *
145  *
146  *
147  *
148  *
149  *
150  *
151  *
152  *
153  *
154  *
155  *
156  *
157  *
158  *
159  *
160  *
161  *
162  *
163  *
164  *
165  *
166  *
167  *
168  *
169  *
170  *
171  *
172  *
173  *
174  *
175  *
176  *
177  *
178  *
179  *
180  *
181  *
182  *
183  *
184  *
185  *
186  *
187  *
188  *
189  *
190  *
191  *
192  *
193  *
194  *
195  *
196  *
197  *
198  *
199  *
200  *
201  *
202  *
203  *
204  *
205  *
206  *
207  *
208  *
209  *
210  *
211  *
212  *
213  *
214  *
215  *
216  *
217  *
218  *
219  *
220  *
221  *
222  *
223  *
224  *
225  *
226  *
227  *
228  *
229  *
230  *
231  *
232  *
233  *
234  *
235  *
236  *
237  *
238  *
239  *
240  *
241  *
242  *
243  *
244  *
245  *
246  *
247  *
248  *
249  *
250  *
251  *
252  *
253  *
254  *
255  *
256  *
257  *
258  *
259  *
260  *
261  *
262  *
263  *
264  *
265  *
266  *
267  *
268  *
269  *
270  *
271  *
272  *
273  *
274  *
275  *
276  *
277  *
278  *
279  *
280  *
281  *
282  *
283  *
284  *
285  *
286  *
287  *
288  *
289  *
290  *
291  *
292  *
293  *
294  *
295  *
296  *
297  *
298  *
299  *
300  *
301  *
302  *
303  *
304  *
305  *
306  *
307  *
308  *
309  *
310  *
311  *
312  *
313  *
314  *
315  *
316  *
317  *
318  *
319  *
320  *
321  *
322  *
323  *
324  *
325  *
326  *
327  *
328  *
329  *
330  *
331  *
332  *
333  *
334  *
335  *
336  *
337  *
338  *
339  *
340  *
341  *
342  *
343  *
344  *
345  *
346  *
347  *
348  *
349  *
350  *
351  *
352  *
353  *
354  *
355  *
356  *
357  *
358  *
359  *
360  *
361  *
362  *
363  *
364  *
365  *
366  *
367  *
368  *
369  *
370  *
371  *
372  *
373  *
374  *
375  *
376  *
377  *
378  *
379  *
380  *
381  *
382  *
383  *
384  *
385  *
386  *
387  *
388  *
389  *
390  *
391  *
392  *
393  *
394  *
395  *
396  *
397  *
398  *
399  *
400  *
401  *
402  *
403  *
404  *
405  *
406  *
407  *
408  *
409  *
410  *
411  *
412  *
413  *
414  *
415  *
416  *
417  *
418  *
419  *
420  *
421  *
422  *
423  *
424  *
425  *
426  *
427  *
428  *
429  *
430  *
431  *
432  *
433  *
434  *
435  *
436  *
437  *
438  *
439  *
440  *
441  *
442  *
443  *
444  *
445  *
446  *
447  *
448  *
449  *
450  *
451  *
452  *
453  *
454  *
455  *
456  *
457  *
458  *
459  *
460  *
461  *
462  *
463  *
464  *
465  *
466  *
467  *
468  *
469  *
470  *
471  *
472  *
473  *
474  *
475  *
476  *
477  *
478  *
479  *
480  *
481  *
482  *
483  *
484  *
485  *
486  *
487  *
488  *
489  *
490  *
491  *
492  *
493  *
494  *
495  *
496  *
497  *
498  *
499  *
500  *
501  *
502  *
503  *
504  *
505  *
506  *
507  *
508  *
509  *
510  *
511  *
512  *
513  *
514  *
515  *
516  *
517  *
518  *
519  *
520  *
521  *
522  *
523  *
524  *
525  *
526  *
527  *
528  *
529  *
530  *
531  *
532  *
533  *
534  *
535  *
536  *
537  *
538  *
539  *
540  *
541  *
542  *
543  *
544  *
545  *
546  *
547  *
548  *
549  *
550  *
551  *
552  *
553  *
554  *
555  *
556  *
557  *
558  *
559  *
560  *
561  *
562  *
563  *
564  *
565  *
566  *
567  *
568  *
569  *
570  *
571  *
572  *
573  *
574  *
575  *
576  *
577  *
578  *
579  *
580  *
581  *
582  *
583  *
584  *
585  *
586  *
587  *
588  *
589  *
590  *
591  *
592  *
593  *
594  *
595  *
596  *
597  *
598  *
599  *
600  *
601  *
602  *
603  *
604  *
605  *
606  *
607  *
608  *
609  *
610  *
611  *
612  *
613  *
614  *
615  *
616  *
617  *
618  *
619  *
620  *
621  *
622  *
623  *
624  *
625  *
626  *
627  *
628  *
629  *
630  *
631  *
632  *
633  *
634  *
635  *
636  *
637  *
638  *
639  *
640  *
641  *
642  *
643  *
644  *
645  *
646  *
647  *
648  *
649  *
650  *
651  *
652  *
653  *
654  *
655  *
656  *
657  *
658  *
659  *
660  *
661  *
662  *
663  *
664  *
665  *
666  *
667  *
668  *
669  *
670  *
671  *
672  *
673  *
674  *
675  *
676  *
677  *
678  *
679  *
680  *
681  *
682  *
683  *
684  *
685  *
686  *
687  *
688  *
689  *
690  *
691  *
692  *
693  *
694  *
695  *
696  *
697  *
698  *
699  *
700  *
701  *
702  *
703  *
704  *
705  *
706  *
707  *
708  *
709  *
710  *
711  *
712  *
713  *
714  *
715  *
716  *
717  *
718  *
719  *
720  *
721  *
722  *
723  *
724  *
725  *
726  *
727  *
728  *
729  *
730  *
731  *
732  *
733  *
734  *
735  *
736  *
737  *
738  *
739  *
740  *
741  *
742  *
743  *
744  *
745  *
746  *
747  *
748  *
749  *
750  *
751  *
752  *
753  *
754  *
755  *
756  *
757  *
758  *
759  *
760  *
761  *
762  *
763  *
764  *
765  *
766  *
767  *
768  *
769  *
770  *
771  *
772  *
773  *
774  *
775  *
776  *
777  *
778  *
779  *
780  *
781  *
782  *
783  *
784  *
785  *
786  *
787  *
788  *
789  *
790  *
791  *
792  *
793  *
794  *
795  *
796  *
797  *
798  *
799  *
800  *
801  *
802  *
803  *
804  *
805  *
806  *
807  *
808  *
809  *
810  *
811  *
812  *
813  *
814  *
815  *
816  *
817  *
818  *
819  *
820  *
821  *
822  *
823  *
824  *
825  *
826  *
827  *
828  *
829  *
830  *
831  *
832  *
833  *
834  *
835  *
836  *
837  *
838  *
839  *
840  *
841  *
842  *
843  *
844  *
845  *
846  *
847  *
848  *
849  *
850  *
851  *
852  *
853  *
854  *
855  *
856  *
857  *
858  *
859  *
860  *
861  *
862  *
863  *
864  *
865  *
866  *
867  *
868  *
869  *
870  *
871  *
872  *
873  *
874  *
875  *
876  *
877  *
878  *
879  *
880  *
881  *
882  *
883  *
884  *
885  *
886  *
887  *
888  *
889  *
890  *
891  *
892  *
893  *
894  *
895  *
896  *
897  *
898  *
899  *
900  *
901  *
902  *
903  *
904  *
905  *
906  *
907  *
908  *
909  *
910  *
911  *
912  *
913  *
914  *
915  *
916  *
917  *
918  *
919  *
920  *
921  *
922  *
923  *
924  *
925  *
926  *
927  *
928  *
929  *
930  *
931  *
932  *
933  *
934  *
935  *
936  *
937  *
938  *
939  *
940  *
941  *
942  *
943  *
944  *
945  *
946  *
947  *
948  *
949  *
950  *
951  *
952  *
953  *
954  *
955  *
956  *
957  *
958  *
959  *
960  *
961  *
962  *
963  *
964  *
965  *
966  *
967  *
968  *
969  *
970  *
971  *
972  *
973  *
974  *
975  *
976  *
977  *
978  *
979  *
980  *
981  *
982  *
983  *
984  *
985  *
986  *
987  *
988  *
989  *
990  *
991  *
992  *
993  *
994  *
995  *
996  *
997  *
998  *
999  *
1000 *
```

```
33 * best efforts in preparing the book. These efforts include the
      *
34 * development, research, and testing of the theories and programs
      *
35 * to determine their effectiveness. The authors and publisher make
      *
36 * no warranty of any kind, expressed or implied, with regard to these
      *
37 * programs or to the documentation contained in these books. The
  authors *
38 * and publisher shall not be liable in any event for incidental or
      *
39 * consequential damages in connection with, or arising out of, the
      *
40 * furnishing, performance, or use of these programs.
      *
41 *****
42 -->
```

Example: cookies.php

```
1 <!-- Fig. 19.18: cookies.php -->
2 <!-- Writing a cookie to the client. -->
3 <?php
4     define( "FIVE_DAYS", 60 * 60 * 24 * 5 ); // define constant
5
6     // write each form fields value to a cookie and set the
7     // cookies expiration date
8     setcookie( "name", $_POST["name"], time() + FIVE_DAYS );
9     setcookie( "height", $_POST["height"], time() + FIVE_DAYS );
10    setcookie( "color", $_POST["color"], time() + FIVE_DAYS );
11 ?><!-- end PHP script -->
12
13 <!DOCTYPE html>
14
15 <html>
16     <head>
17         <meta charset = "utf-8">
18         <title>Cookie Saved</title>
19         <style type = "text/css">
```

```
20     p { margin: 0px; }
21     </style>
22 </head>
23 <body>
24     <p>The cookie has been set with the following data:</p>
25
26     <!-- print each form field's value -->
27     <p>Name: <?php print( $_COOKIE["name"] ) ?></p>
28     <p>Height: <?php print( $_COOKIE["height"] ) ?></p>
29     <p>Favorite Color:
30         <span style = "color: <?php print( $_COOKIE["color"] ) ?> ">
31         <?php print( $_COOKIE["color"] ) ?></span></p>
32     <p>Click <a href = "readCookies.php">here</a>
33         to read the saved cookie.</p>
34 </body>
35 </html>
36
37 <!--
38 *****
39
40 * (C) Copyright 1992–2012 by Deitel & Associates, Inc. and
41 *
42 * Pearson Education, Inc. All Rights Reserved.
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
```

```
50 * furnishing, performance, or use of these programs.  
    *  
51 *****  
  
52 -->
```

Cookies

- `setcookie` creates a cookie
 - First parameter is the name
 - Second parameter is the data
 - Third parameter is the expiration date
 - * If there is no expiration date, the cookie is a *session cookie*, which means it only exists during the current browsing session (when the user closes the browser, the session ends)
 - * If a expiration date is specified, we call this cookie a *persistent cookie*
- Cookies are then accessible through the `$_COOKIE` superarray

Example: readCookies.php

```
1 <!DOCTYPE html>  
2  
3 <!-- Fig. 19.19: readCookies.php -->  
4 <!-- Displaying the cookies contents. -->  
5 <html>  
6     <head>  
7         <meta charset = "utf-8">  
8         <title>Read Cookies</title>  
9         <style type = "text/css">  
10             p { margin: 0px; }  
11         </style>  
12     </head>  
13     <body>  
14         <p>The following data is saved in a cookie on your computer.</p>  
15         <?php  
16             // iterate through array $_COOKIE and print  
17             // name and value of each cookie  
18             foreach ( $_COOKIE as $key => $value )  
19                 print( "<p>$key: $value</p>" );
```



```
20      ?><!-- end PHP script -->
21      </body>
22 </html>
23
24 <!--
25 *****
26
27 * (C) Copyright 1992-2012 by Deitel & Associates, Inc. and
28 *
29 * Pearson Education, Inc. All Rights Reserved.
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *****
39 -->
```