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. [*Note:* 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`

```
<!DOCTYPE html>
<!-- Fig. 19.1: first.php -->
<!-- Simple PHP program. -->
<html>
  <?php
    $name = "Paul"; // declaration and initialization
  ?><!-- end PHP script -->
  <head>
    <meta charset = "utf-8">
    <title>Simple PHP document</title>
  </head>
  <body>
    <!-- print variable name's value -->
    <h1><?php print( "Welcome to PHP, $name!" ); ?></h1>
  </body>
</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          <p>as a double: " . (double) $data . "</p>" .
55          "<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 oepration, 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 Javscript, but **`else if`** is `elseif` (another keyword)

Arithmetic Operators

- Same as Javascript

Example: `operators.php`

```
<!DOCTYPE html>
<!-- Fig. 19.4: operators.php -->
<!-- Using arithmetic operators. -->
<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        }
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>" );
              // nothing evaluates to ""
55
56        // add constant VALUE to an uninitialized variable
57        $test = $num + VALUE;
58
59        // num evaluates to 0
60        print( "<p>An uninitialized variable plus constant VALUE yields
             $test</p>" );
61
62        // add a string to an integer
63        $str = "3 dollars";
64        $a += $str;
65        print( "<p>Adding a string to variable a yields $a</p>" );
66      ?><!-- end PHP script -->
67    </body>
68  </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 indicies
  - 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`

```
<!DOCTYPE html>
<!-- Fig. 19.7: arrays.php -->
<!-- Array manipulation. -->
<html>
<head>
<meta charset = "utf-8">
<title>Array manipulation</title>
<style type = "text/css">
p
{ margin: 0; }
.head { margin-top: 10px; font-weight: bold; }
</style>
</head>
<body>
<?php
  // create array first
  print( "<p class = 'head'>Creating the first array</p>" );
  $first[ 0 ] = "zero";
  $first[ 1 ] = "one";
  $first[ 2 ] = "two";
  $first[] = "three";
  // print each element's index and value
  for ( $i = 0; $i < count( $first ); ++$i )
    print( "<p>Element $i is $first[$i]</p>" );
  print( "<p class = 'head'>Creating the second array</p>" );
  // call function array to create array second
  $second = array( "zero", "one", "two", "three" );
  for ( $i = 0; $i < count( $second ); ++$i )
    print( "<p>Element $i is $second[$i]</p>" );
  print( "<p class = 'head'>Creating the third array</p>" );
  // assign values to entries using nonnumeric indices
  $third[ "Amy" ] = 21;
  $third[ "Bob" ] = 18;
  $third[ "Carol" ] = 23;
  // iterate through the array elements and print each
  // element's name and value
  for ( reset( $third ); $element = key( $third ); next( $third ) )
    print( "<p>$element is $third[$element]</p>" );
  print( "<p class = 'head'>Creating the fourth array</p>" );
  // 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 quanity 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. -->
```

```
<html>
  <head>
    <meta charset = "utf-8">
    <title>Regular expressions</title>
    <style type = "text/css">
      p { margin: 0; }
    </style>
  </head>
  <body>
    <?php
      $search = "Now is the time";
      print( "<p>Test string is: '$search'</p>" );
      // call preg_match to search for pattern 'Now' in variable search
      if (
        preg_match( "/Now/", $search )
      )
      print( "<p>'Now' was found.</p>" );
      // search for pattern 'Now' in the beginning of the string
      if (
        preg_match( "/^Now/", $search )
      )
      print( "<p>'Now' found at beginning of the line.</p>" );
      // search for pattern 'Now' at the end of the string
      if (
        !preg_match( "/Now$/", $search )
      )
      print( "<p>'Now' was not found at the end of the line.</p>" );
      // search for any word ending in 'ow'
      if (
        preg_match( "/\b([a-zA-Z]*ow)\b/i", $search, $match )
      )
      print( "<p>Word found ending in 'ow': " .
      $match[ 1 ]
      . "</p>" );
      // search for any words beginning with 't'
      print( "<p>Words beginning with 't' found: " );
      while (
        preg_match( "/\b(t[[:alpha:]]+)\b/", $search, $match )
      )
      {
        print(
          $match[ 1 ]
          . " " );
```

```
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
$_SERVER	Data about the currently running server.
$_ENV	Data about the client's environment.
$_GET	Data sent to the server by a get request.
$_POST	Data sent to the server by a post request.
$_COOKIE	Data contained in cookies on the client's computer.
$GLOBALS	Array containing all global variables.

Figure 3: Superglobal arrays

Example: `form.html`

```
<!DOCTYPE html>
<!-- Fig. 19.13: form.html -->
<!-- HTML form for gathering user input. -->
<html>
<head>
  <meta charset="utf-8">
  <title>Sample Form</title>
  <style type="text/css">
    label {
      width: 5em;
      float: left;
    }
  </style>
</head>
<body>
  <h1>Registration Form</h1>
  <p>Please fill in all fields and click Register.</p>
  <!-- post form data to form.php -->
  <form method="post" action="form.php">
    <h2>User Information</h2>
    <!-- create four text boxes for user input -->
    <div>
      <label>First name:</label>
      <input type="text" name="fname">
    </div>
    <div>
      <label>Last name:</label>
      <input type="text" name="lname">
    </div>
    <div>
      <label>Email:</label>
      <input type="text" name="email">
    </div>
    <div>
      <label>Phone:</label>
      <input type="text" name="phone" placeholder="(555) 555-5555">
    </div>
    <h2>Publications</h2>
    <p>Which book would you like information about?</p>
    <!-- create drop-down list containing book names -->
    <select name="book">
      <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; the register an input to an entry in the superarray
  - This is why names mattered in chapter 2/3!!

Example: `form.php`

```
<!DOCTYPE html>
<!-- Fig. 19.14: form.php -->
<!-- Process information sent from form.html. -->
<html>
  <head>
    <meta charset = "utf-8">
    <title>Form Validation</title>
    <style type = "text/css">
      p
      { margin: 0px; }
      .error
      { color: red }
      p.head { font-weight: bold; margin-top: 10px; }
    </style>
  </head>
  <body>
    <?php
    // determine whether phone number is valid and print
    // an error message if not
    // regex looks for the following pattern "(###) ###-####"
    if (!preg_match( "/^\([0-9]{3}\) [0-9]{3}-[0-9]{4}$/", $_POST["
       phone"]))
    {
      print( "<p class = 'error'>Invalid phone number</p>
        <p>A valid phone number must be in the form
        (555) 555-5555</p><p>Click the Back button,
        enter a valid phone number and resubmit.</p>
        <p>Thank You.</p></body></html>" );
      die(); // terminate script execution
    }
    ?><!-- end PHP script -->
    <p>
      <!-- Access information from the submission using the $_POST
         superarray -->
      Hi <?php print( $_POST["fname"] ); ?>. Thank you for completing
         the survey. You have been added to the
      <?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 )) ?>
                results.
59        </p>
60      <p>
61        Please email comments to <a href = "mailto:deitel@deitel.com">
              Deitel and Associates, Inc.</a>
62      </p>
63    </body>
64  </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 chatper 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_fecth_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`

```
<!DOCTYPE html>

<!-- Fig. 19.17: cookies.html -->
<!-- Gathering data to be written as a cookie. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>Writing a cookie to the client computer</title>
      <style type = "text/css">
         label { width: 7em; float: left; }
      </style>
   </head>
   <body>
      <h2>Click Write Cookie to save your cookie data.</h2>
      <form method = "post" action = "cookies.php">
         <div><label>Name:</label>
            <input type = "text" name = "name"><div>
         <div><label>Height:</label>
            <input type = "text" name = "height"></div>
         <div><label>Favorite Color:</label>
            <input type = "text" name = "color"></div>
         <p><input type = "submit" value = "Write Cookie">
      </form>
   </body>
</html>

<!--
**************************************************************************
* (C) Copyright 1992-2012 by Deitel & Associates, Inc. and
                    *
* Pearson Education, Inc. All Rights Reserved.
                                  *
*

    *
* DISCLAIMER: The authors and publisher of this book have used their
        *
```

```
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 * (C) Copyright 1992-2012 by Deitel & Associates, Inc. and               *
40 * Pearson Education, Inc. All Rights Reserved.                           *
41 *
42 * DISCLAIMER: The authors and publisher of this book have used their     *
43 * best efforts in preparing the book. These efforts include the          *
44 * development, research, and testing of the theories and programs        *
45 * to determine their effectiveness. The authors and publisher make       *
46 * no warranty of any kind, expressed or implied, with regard to these    *
47 * programs or to the documentation contained in these books. The         *
48 * and publisher shall not be liable in any event for incidental or       *
49 * consequential damages in connection with, or arising out of, the       *
```

```
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  * (C) Copyright 1992-2012 by Deitel & Associates, Inc. and
                    *
27  * Pearson Education, Inc. All Rights Reserved.
                                  *
28  *


       *
29  * DISCLAIMER: The authors and publisher of this book have used their
           *
30  * best efforts in preparing the book. These efforts include the
                *
31  * development, research, and testing of the theories and programs
              *
32  * to determine their effectiveness. The authors and publisher make
             *
33  * no warranty of any kind, expressed or implied, with regard to these
           *
34  * programs or to the documentation contained in these books. The
       authors *
35  * and publisher shall not be liable in any event for incidental or
             *
36  * consequential damages in connection with, or arising out of, the
             *
37  * furnishing, performance, or use of these programs.
                            *
38  ************************************************************************

39  -->
```