Chapter 1

PHP Crash Course

Order form

- Common application for server-side scripting language to process the form
  - Create form on html page
  - Form action set to name of PHP script that will process the form
  - Use the form field names in the PHP script
  - Orderform.html
- Process the form
  - Processorder.php

orderform.html ---p.15-16

processorder.php—p. 16-17

- Not see raw php code in html page source code rendered in browser when you embed php code—p. 17-18

```
<?php
    echo ‘<p>Order processed.</p>’;
?>
```

PHP tags

- <?php and ?>
- Any text between tags is PHP code
- 4 styles of PHP tags and all equivalent
  - XML
    - <?php echo ‘<p>Order processed.</p>’; ?>
    - One used in this book
  - Short style
    - <? Echo ‘<p>Order processed.</p>’; ?>
  - SCRIPT style
    - <script language-’php’> echo ‘<p>Order processed.</p>’;</script>
  - ASP style
    - <% echo ‘<p>Order processed.</p>’; %>

PHP Statements

- Included between opening and closing PHP tags to tell the interpreter what to do
- Echo prints the string passed to it to the browser
- ; separates statements in PHP

Whitespace

- Spacing characters such as newlines(carriage returns), spaces, and caps

Comments

- Notes for people reading the code
- Explain
  - Purpose of script
  - Who wrote it
  - Why the wrote it the way they did
  - When modified
- Types
  - Multiline
  - Single line

```
/* Author: Bob Smith
   Last modified:  April 10
   This script processes the customer orders.
*/

echo ‘<p>Order processed.</p>’;   //start printing order---C++ style
Echo ‘<p>Order process.</p>’; # Start printing order---shell script style
```

Adding Dynamic Content

- Use built in date function to give current date and time

```
<?php
    echo "<p>Order processed at "
   echo date('H:i, jS FY');
   echo "</p>";
?>
```

Function call

    function name(parameters)

Accessing Form Variables

- All variable names start with $
- 3 ways to access form data
  - Short style
    - $tireqty
    - Requires register_globals configuration settings be turned on
    - Names in script are same as in form
  - Medium style
    - $_POST['tireqty']
    - Used in this book and recommended approach
    - Retrieve variables one of the arrays $_POST, $_GET, $_REQUEST
      - Dependent on how submitted and request has combination
  - Long style
    - $HTTP_POST_VARS['tireqty']
    - Most verbose
    - Deprecated and likely be removed in future

Code to store user entry in variable and echo it

```
<?php
   //create short variable names
   $tireqty=$_POST['tireqty'];
    $oilqty=$_POST['oilqty'];
   $sparkqty=$_POST['sparkqty'];
?>

//print the variables
echo '<p>Your order is as follows: </p>';
echo $tireqty.' tires<br> />';
echo $oilqty.' bottles of oil<br> />';
echo $sparkqty.' spark plugs<br />';
```

String Concatenation

- Add string together
- Use .
- Example
  - echo $tireqty.’ tires<br />’;
    - Places the value of the variable $tireqty in front on word tires
    - 7 tires

Variables and Literals

- Literal
  - Raw piece of data
  - Example
    - ‘ tires<br />’ is literal whereas $tireqty is a variable (store data)
- Two types strings
  - “” –double quotation marks—PHP evaluate
  - ‘ ’ –single—treated as a Literal

Identifiers

- Names of variables, functions, classes
- Rules for identifiers
  - Can be of any length and can consist of letters, numbers, and underscores
  - Cannot begin with a digit
  - Case sensitive in PHP
  - Variable and function can have same name but should be avoided
- Can declare own variables in addition to ones from form
- Not need to declare ahead—created when assigned first value
- Assign using =
- Examples
  - $totalqty=0;
  - $totalqty=$price+$tax;

Variable Types

- Type of data
  - Integer, Float(double), string, Boolean, array, object
- Two special types of data
  - Null
    - Variables have no value or given a specific value of NULL
  - Resource
    - Some built in functions like ones that represent external resources like database connections
- Type Strength
  - PHP is weakly or dynamically typed
    - Determined by value assigned it
    - Could be an int at one point and later a string
- Type casting
  - Pretend variable of different data type
  - Example
    - $totalqty=0;
    - $totalqty=(float)$totalqty;
- Variable variable
  - Variable type that enables you to change the name of a variable dynamically—look at later

Declaring and Using Constants

- Constant
  - Stores a value just like a variable, but its value is set at once and then cannot be changed elsewhere in the script
  - Can store only Boolean, integer, float, string unlike variables
- Example
  - define('TIREPRICE',100);
  - Not have a$ in front as a variable does
  - echo TIREPRICE;   //displays the constant value
- See built in constants and variables to PHP
  - phpinfo();

Understanding Variable Scope

- Scope
  - Places within a script where a particular variable is visible
- Scope rules
  - P. 31-32
- Superglobals
  - Can be seen everywhere both inside and outside functions
  - P. 32 list of them

Using Operators

- Symbols you can use to manipulate values and variables by performing an operation on them
- Arithmetic operators
  - +, -, *, / , % (modulus)
  - Example
    - $a%$b
- - can be used as negative sign in number
- - can be used as uninary operator
  - $b=-$a;
- String operators (.) for concatenation
- Assignment (=)

Combined Assignment Operators

- -=
- +=
- *=
- /=
- %=
- .=

Pre and Post Increment and Decrement

- ++a
  - Preincrement
  - Example
    - $a=4;
    - echo ++$a;
    - 5 displayed
- a++
  - post increment
  - Example
    - $a=4;
    - echo $a++;
    - 4 displayed
- -- is decrement and similar idea

Reference Operator

- &
  - Used in conjunction with =
  - $b = &$a
  - Changing either one now changes the other
  - unset($a) or unset($b) will break the link

More operators

- Comparison
  - Compare two values
  - Table 1.3 on p. 37
- Logical
  - Combine results of logical conditions
  - Table 1.4 on p. 38
- Bitwise
  - Treat an integer as the series of bits used to represent it
  - Table 1.5 p. 38
- Comma (,) used to separate function arguments
- Ternary operator(?:)
  - Condition ? Value if true: value if false
  - ($grade> = 50 ? 'Passed' : 'Failed')
- Error Suppression
  - Suppresses an error like $a=@(57/0) suppresses the divide by zero error
- Execution
  - Pair of backticks (` `)
  - Used to execute commands on Unix operating system
- Array
  - [] access an element in the array
  - Table 1.6 p. 40
- Type
  - Instanceof
    - Allows you to check whether an object is an instance of a particular class

Working out form totals

```
$totalqty=0;
$totalqty=$tireqty+$oilqty+$sparkqty;
echo “Items ordered: “.$totalqty.”<br />”;
$totalamount=0.00;

define(‘TIREPRICE’, 100);
define(‘OILPRICE’, 10);
define(‘SPARKPRICE’,4);

$totalamount=$tireqty*TIREPRICE
             + $oilqty*OILPRICE
            + $sparkqty*SPARKPRICE;
echo “Subtotal: $”.number_format($totalamount,2).”<br />;

$taxrate=0.10;  //local sales tax is 10%
$totalamount=$totalamount*(1+$taxrate);
echo “Total including tax: $”.number_format($totalamount,2).”<br />”;
```

Precedence and Associativity

- Generally left to right
- Order in mathematics
- Table 1.7 p. 43

Conditionals

```
if(conditional){
    statement;
```

- statement; Code is divided into blocks by {}
  - Curly braces always come in pairs, common error
  - Always indent what's inside the curly braces!

- Blocks can be nested

```
{
  //this is one block
  {
                  //this is a block within a
```

- Conditional is something that is either true or false
  - False is 0, empty, Boolean false
  - True is anything else
- If conditional is true, block gets executed
- Otherwise, block is skipped

Making Decisions with Conditionals

- If
  - Used for decisions
  - if($totalqty==0)
  - {
  - echo '< p style="color: red">';
    echo 'You did not order anything on the previous page!";
  - echo '</p>;
  - }

Else statement

- Used to decide between a set of conditions
- Form
  - If condition
    - Then do
  - Else
    - Do this

```
if($a == 4){
            echo “4!”;
}
else{
            echo “Not 4!”;
}
```

Elseif

```
if($a == 4){
    echo “4!”;
}
elseif($a == 5){
    echo “5!”
}
else{
    echo “Not 4 or 5!”;
}
```

Switch—condition take more than 2 values

```
switch($a){
   case “a”:
                  echo ‘It’s an a!’;
                  break;
   case “b”:
                  echo ‘It’s a b!’;
                  break;
   default:
                  echo ‘Hmm.  What is it?’;
                  break;
}
```

While loops

- Executes block repeatedly as long as the condition is true
- while (condition) expression
- Example

```
while ($num<=5) {
   echo $num."<br />";
   $num++;
}
```

freight.html—p. 52-53 --longer
freight.php ---p. 54 --use of while statement

For and for each loops

- For (expression1;condition;expression2)
  - Expression3;
- Example

```
for($x = 50; $x < 250; $x += 50){
   echo x."<br />";
}
//set variable, check conditional, execute, increment
```

Do while

```
$x = 0;
do{
   echo x.”<br />”;
}
while ($x > 3);
//always executes at least once!
```

Control structures-break

- continue;
  - Skip the rest of the loop, but move to the next iteration
- break;
  - Completely exit the loop you're presently in
- exit;
  - End the script at this point