Introduction to Databases

PHP 1

Key Concepts

- PHP in HTML
- Calling functions
- Form variables
- Identifies and data types
- Operators
- Decisions
- Conditionals
- Arrays
- Multi-dimensional arrays
- Sorting arrays
- Array manipulation

Calling a PHP Script

```html
<form action="processorder.php" method="post">
  <input type="text" name="tireqty">
</form>
```
PHP Tag Styles

- XML style
  ```php
  echo '<p>Order processed.</p>'; ?>
  ```
- Short style
  ```php
  ? echo '<p>Order processed.</p>'; ?>
  ```
- SCRIPT style
  ```php
  <script language='php'>
  echo '<p>Order processed.</p>'; </script>
  ```
- ASP style
  ```php
  <%= echo '<p>Order processed.</p>'; %>
  ```

Comments

- Multi-line comment
  ```
  /* Author: Bob Smith
     Last modified: April 10
     This script processes orders */
  ```
- Single-line comment
  ```
  //Start printing order
  ```
- Single-line shell-script style comment
  ```
  #Start printing order
  ```
- Single-line comment ended by
  ```
  - End of line
    - End of php script tag
  ```

The date Function

- Outputs the current date
- Accepts a format string that specifies the date’s format
- Example:
  ```
  <?php echo date('H:i,jS FY'); ?>
  ```
Accessing Form Variables

- **Short style**
  
  $tireqty
  
  - Requires register_globals configuration
  
  - Security vulnerability
- **Medium style**
  
  $_POST['tireqty']
  
  - Default and recommended style
- **Long style**
  
  $HTTP_POST_VARS['tireqty']
  
  - Will be deprecated
  
  - Use for legacy server compatibility

String Concatenation

- Use period (.) as a string concatenation operator
  
  echo $tireqty.' tires<br/>';
  
  Place string and simple variable inside double quotes
  
  echo "$tireqty tires<br/>";
- Use a heredoc (<<<) for multiple lines
  
  echo <<<theEnd
  
  $tireqty tires
  
  $sparkqty spark plugs

Variables and Data Types

- Weakly typed

- **Cast data type**
  
  $totalqty = 0;
  
  $totalamount = (float)$totalqty;
- **Change data type with settype function**
  
  bool settype(mixed var, string type);
  
  settype($a, 'double');
- **Determine data type with gettype function**
  
  string gettype(mixed var);
  
  echo gettype($a);
Variable Variables

- Allow you to change the name of a variable dynamically
- Useful for processing all fields in a form
- Example

  ```
  $tireqty = 4;
  $varname = 'tireqty';
  $$varname = 5;
  ```

Constants

- Use define function to create a constant

  ```
  define('TIREPRICE', 100);
  ```
- A constant name is not preceded by $

  ```
  echo TIREPRICE;
  ```
- Use `phpinfo();` to obtain a list of predefined PHP constants

Variable Scope

<table>
<thead>
<tr>
<th>Variable</th>
<th>Where visible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in superglobals</td>
<td>Everywhere in script</td>
</tr>
<tr>
<td>Constants</td>
<td>Everywhere after they are declared – inside and outside functions</td>
</tr>
<tr>
<td>Global variables in script</td>
<td>Visible within the script, but not inside functions</td>
</tr>
<tr>
<td>Global variables in function</td>
<td>Refer to the script global variable of the same name</td>
</tr>
<tr>
<td>Static variables in function</td>
<td>Visible only in the function, but keep their value between calls to the function</td>
</tr>
<tr>
<td>Variables inside functions</td>
<td>Visible only within a function</td>
</tr>
</tbody>
</table>
Superglobals

<table>
<thead>
<tr>
<th>Superglobal</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GLOBALS</td>
<td>Array of global variables</td>
</tr>
<tr>
<td>$_SERVER</td>
<td>Array of server environment variables</td>
</tr>
<tr>
<td>$_GET</td>
<td>Array of variables passed via GET method</td>
</tr>
<tr>
<td>$_POST</td>
<td>Array of variables passed via POST method</td>
</tr>
<tr>
<td>$_COOKIE</td>
<td>Array of cookie variables</td>
</tr>
<tr>
<td>$_FILES</td>
<td>Array of variables related to file uploads</td>
</tr>
<tr>
<td>$_ENV</td>
<td>Array of environment variables</td>
</tr>
<tr>
<td>$_REQUEST</td>
<td>Array of all input except $_FILES</td>
</tr>
<tr>
<td>$_SESSION</td>
<td>Array of session variables</td>
</tr>
</tbody>
</table>

Operators

- Arithmetic operators
  - $, -, *, /, %
- Assignment operator
  - =
- Combined operators
  - +=, -=, /=, *=, %=, .=
- Increment and decrement operators
  - --, ++
- Reference operator
  - $
- Comparison operators
  - ==, !=, ===, !==, <>, <, >, <=, >=
- Logical operators
  - !, &&, ||, and, or, xor
- Bitwise operators
  - &\ (AND), |\ (OR), ~\ (NOT), ^\ (XOR), <<\ (left shift), >>

Other Operators

- Instantiate a class
  - new
- Access a class member
  - ->
- Type operator
  - instanceof
- Ternary operator
  - $grade = 80 ? 'Passed' : 'Failed';
- Error suppression operator
  - @
- Command-line execution operator
  - `ls -la`; echo "<pre>\$out\</pre>;"
Variable Functions

- is_array
- is_double, is_float, is_real
- is_long, is_int, is_integer
- is_string
- is_bool
- is_object
- is_resource
- is_null
- is_scalar
- is_numeric
- is_callable
- isset
- unset
- empty
- intval
- floatval
- strval

Decisions

- Single-statement if
  ```php
  if ($totalqty == 0)
  echo 'You did not order anything<br/>';
  ```

- Code blocks
  ```php
  if ($totalqty == 0) {
    echo '<p style="color:red">';
    echo 'You did not order anything</p>';
  }
  ```

Using else

```php
if ($totalqty == 0)
  echo "You did not order anything<br/>";
else {
  echo $tireqty." tires<br/>";
  echo $sparkqty." sparkplugs<br/>";
}
```
Using elseif

```php
if ($tireqty < 10) {
    $discount = 0;
} elseif (($tireqty >= 10) && ($tireqty <=49)) {
    $discount = 5;
} elseif (($tireqty >=50) && ($tireqty <=99)) {
    $discount = 10;
} elseif ($tireqty > 100) {
    $discount = 15;
}
```

Switch Statement

```php
switch($find) {
    case "a":
        echo "<p>Regular customer</p>";
        break;
    case "b":
        echo "<p>Customer referred by TV advertisement</p>";
        break;
    default:
        echo "<p>We do not know how the customer found us.</p>";
}
```

while and do..while

```php
while loop
$num = 1;
while ($num <= 5) {
    echo $num."<br/>
";
    $num++;
}

do..while loop
$num = 100;
do {
    echo $num."<br/>
";
    $num--;
} while ($num > 0);
```
for Loop

```php
for ($distance=50; $distance<=250; $distance+=50) {
    echo "<tr>
    <td align="right">".$distance."</td>
    <td align="right">".(($distance/10)."</td>
    </tr>
};
```

Creating and initializing an array

- Array of discrete values
  ```php
  $products = array('Tires', 'Oil', 'Spark plugs');
  ```
- Array of numbers from 1 to 10
  ```php
  $numbers = range(1, 10);
  ```
- Array of odd numbers from 1 to 10
  ```php
  $oddNumbers = range(1, 10, 2);
  ```

Using Arrays

- Setting a value
  ```php
  $products[0] = 'Fuses';
  ```
- Displaying values
  ```php
  echo "$products[0] $products[1]";
  ```
Loops and Arrays

• for Loop
for ($i=0; $i<3; $i++) {
    echo $products[$i]." ";
}

• foreach Loop
foreach ($products as $current) {
    echo $current." ";
}

Keys and Values

• Create an array of key/value pairs
$prices=array('Tires'=>100, 'Oil'=>10, 'Plugs'=>4);

• Add to array
$prices['WindshieldWipers'] = 8;

• Accessing key/value pairs with foreach loop
foreach ($prices as $key => $value) {
    echo $key." – ".value;  
}

• Accessing key/value pairs with each
while ($element = each($prices)) {
    echo $element['key'];
    echo " – ";
    echo $element['value'];
    echo "<br/>";
}

Array Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Use</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+</code></td>
<td>Union</td>
<td>$a + $b</td>
<td>Returns array containing everything in $a and $b</td>
</tr>
<tr>
<td><code>==</code></td>
<td>Equality</td>
<td>$a == $b</td>
<td>Returns true if $a and $b have the same key and pairs</td>
</tr>
<tr>
<td><code>===</code></td>
<td>Identity</td>
<td>$a === $b</td>
<td>Returns true if $a and $b have the same key and value pairs the same order</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Inequality</td>
<td>$a != $b</td>
<td>Returns true if $a and $b are not equal</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>Inequality</td>
<td>$a &lt;&gt; $b</td>
<td>Returns true if $a and $b are not equal</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Nona-Identity</td>
<td>$a !== $b</td>
<td>Returns true if $a and $b are not identical</td>
</tr>
</tbody>
</table>
Two-dimensional Array

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIR</td>
<td>Tires</td>
<td>100</td>
</tr>
<tr>
<td>OIL</td>
<td>OIL</td>
<td>19</td>
</tr>
<tr>
<td>SPK</td>
<td>Spark Plugs</td>
<td>4</td>
</tr>
</tbody>
</table>

Starting array:

sort()

rsort()

Sorting Arrays

Starting array: 9 3 4 8 2

sort() 2 3 4 8 9

rsort() 9 8 4 3 2

Sorting Key/Value Pairs by Value

Starting array:

asort()

arsort()
### Sorting Key/Value Pairs by Key

Starting array:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oil</td>
<td>9</td>
</tr>
<tr>
<td>Sparkplugs</td>
<td>3</td>
</tr>
<tr>
<td>Wipers</td>
<td>4</td>
</tr>
<tr>
<td>Tires</td>
<td>8</td>
</tr>
</tbody>
</table>

```
ksort()
```

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparkplugs</td>
<td>3</td>
</tr>
<tr>
<td>Tires</td>
<td>8</td>
</tr>
<tr>
<td>Wipers</td>
<td>4</td>
</tr>
<tr>
<td>Oil</td>
<td>9</td>
</tr>
</tbody>
</table>

```
krsort()
```

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wipers</td>
<td>4</td>
</tr>
<tr>
<td>Tires</td>
<td>8</td>
</tr>
<tr>
<td>Sparkplugs</td>
<td>3</td>
</tr>
<tr>
<td>Oil</td>
<td>9</td>
</tr>
</tbody>
</table>

### Other Array Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shuffle()</td>
<td>Randomize values</td>
</tr>
<tr>
<td>array_reverse()</td>
<td>Create a new array in reverse order</td>
</tr>
<tr>
<td>array_push()</td>
<td>Add an item to the end of the array</td>
</tr>
<tr>
<td>array_pop()</td>
<td>Remove an item from the end of the array</td>
</tr>
<tr>
<td>array_walk()</td>
<td>Execute a function on each value in the array</td>
</tr>
<tr>
<td>each(), current(), reset(), end(), next(), prev(), pos()</td>
<td>Navigate through an array</td>
</tr>
<tr>
<td>count(), sizeof(), array_count_values()</td>
<td>Count elements in an array</td>
</tr>
<tr>
<td>extract()</td>
<td>Extract values from key/value array</td>
</tr>
</tbody>
</table>