ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

IT4552 – Web programming

Chapter 4. Working with Arrays

1

Objectives

- To understand the benefits of using arrays in PHP
- To learn how to create and use sequential arrays and their functions
- To learn how to create and use nonsequential arrays and their functions

2

Content

1. Benefits of arrays
2. Sequential arrays
3. Non-sequential arrays
4. Multidimensional lists

3

Content

1. Benefits of arrays
2. Sequential arrays
3. Non-sequential arrays
4. Multidimensional lists

4

1.1. What is an Array?

- An array is a special type of variable.
  - can hold multiple data values
- A sequential array keeps track of these data items by using sequential numbers
  - (e.g., item 0, item 1, item 2, and so on)
- A nonsequential array or associative array keeps track of these data items by using character strings
  - (e.g., item meat, item poultry, item dairy, and so on)

5

1.2. Why Use Arrays?

- *Include a flexible number of list items.*
- *Examine each item more concisely.*
- *Using Loops to Repeat Statements*
- *Use special array operators and functions.*

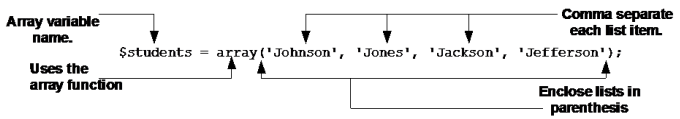
6

Content

7

2.1. Creating Sequential Arrays

- Use the array() function to create an array



- You could also create an array with numerical data
  - `$grades = array(66, 75, 85, 80);`

8

Another way to create an array

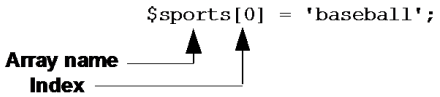
- You can also create an array by making individual value assignments into the array variable name.
- For example, `$students[] = 'Johnson';`

```
$students[] = 'Jones';
$students[] = 'Jackson';
$students[] = 'Jefferson';
```


9

2.2. Referencing Sequential Array Items

- To reference individual array items, use an *array name* and *index pair*



- Indices are referenced sequentially:
  - `$names = array('Denise', 'Christopher', 'Matthew', 'Bryant');`
  - `print ("$names[0], $names[1], $names[2], $names[3]");`
- Outputs names sequentially

10

Warning: Indices starts with 0

- You might think the arrays in the preceding code would be numbered with indices 1 through 4.
  - By default sequential arrays start with index 0,
  - so the indices above are numbered from 0 to 3.
  - Avoid referencing an item past the end of your array (for example, using `$names[20]` in an array that contains only four items).

11

More on Indices ...

- Array indices can be whole numbers or a variable.

```
$i=3;
$classes = array('Math', 'History', 'Science', 'Pottery');
$oneclass = $classes[$i-1];
print "$classes[$i] $oneclass $classes[1] $classes[0]";
```

- This code outputs the following:

  "Pottery Science History Math"

12

2.3. Changing arrays values

- You can change values in an array as follows:

```
$scores = array(75, 65, 85, 90);
$scores[3] = 95;
$average = ($scores[0] + $scores[1] +
                $scores[2] + $scores[3]) / 4;
print "average=$average";
```

- The output of the above PHP segment is "average=80".

13

Explicitly Setting Index Values

- You can explicitly sign values to indices

Assign the value of 65 to the item with index 2.

Assign the value of 85 to the item with index 3.

```
$scores = array(1=>75, 2=>65, 3=>85);
$scores[] = 100;
print "$scores[1] $scores[2] $scores[3]
  $scores[4]";
```

Add item with value 100 to the end of the array.

- The above outputs "`75 65 85 100`".

14

Adding and Deleting Elements

- An element can be added to an array simply by using a key/index that hasn't been used, as shown below:

  $days[5]= "Sat";

- As an alternative to specifying the index, a new element can be added to the end of any array using empty square brackets after the array name, as follows:

  $days[]= "Sun";

- Delete with **unset()**

15

2.4. Using Loops with Sequential Arrays

- Looping statements can be used to iterate through arrays

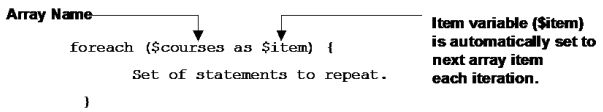
```
$courses = array ('Perl', 'PHP', 'C','Java', 'Pascal', 'Cobol',
   'Visual Basic');
for ($i=0; $i < count($courses); $i++) {
    print ("$courses[$i] ");
}
```

- The above repeats 7 times with $i equal to 0, 1, 2, 3, 4, 5, and 6.
- The above outputs: "`Perl PHP C Java Pascal Cobol Visual Basic`".

16

Using the foreach statement

- PHP supports the foreach statement as another way to iterate through arrays



17

foreach statement - example

- Example of foreach command

```
$courses = array('Perl', 'PHP', 'C', 'Java','Pascal',
   'Cobol', 'Visual Basic');
foreach ($courses as $item){
    print ("$item ");
}
```

- The above outputs "`Perl PHP C Java Pascal Cobol Visual Basic`".

18

Sorting data

- For example the following code segment outputs "1 11 55 91 99 119 911"

```
$courses = array (91, 55, 11, 1, 99, 911, 119);
sort($courses);
foreach ($courses as $item) {
    print "$item ";
}
```


19

Sorting data functions

Effect	Ascending	Descending	User-defined order
Sort array by values, then reassign indices starting with 0	sort( )	rsort( )	usort( )
Sort array by values	asort( )	arsort( )	uasort( )
Sort array by keys	ksort( )	krsort( )	uksort( )

- User-defined ordering requires that you provide a function that takes two values and returns a value that specifies the order of the two values in the sorted array.
  - return 1 if the first value is greater than the second
  - -1 if the first value is less than the second
  - 0 if the values are the same for the purposes of your custom sort order

20

Example

21

A Full Script Example

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

22

A Full Example ...

```
1. <html><head><title> Tuna Cafe </title></head>
2. <body> <font size=4 color="blue">
3. Welcome to the Tuna Cafe Survey! </font>
4. <form action="http://webwizard.aw.com/~phppgm/C5/tunaresults.php" method=post>
5. <?php
6. $menu = array('Tuna Casserole', 'Tuna Sandwich', 'Tuna Pie', 'Grilled Tuna',
                                                    'Tuna Surprise');
7. $bestseller = 2;
8. print 'Please indicate all your favorite dishes.<br>';
9. for ($i=0; $i < count($menu); $i++) {
10.       print "<input type=\"checkbox\" name=\"prefer[]\" value=$i> $menu[$i]";
11.       if ($i == $bestseller) {
12.           print '<font color="red"> Our Best Seller!!!! </font>';
13.      }
14.    print '<br>';
15. }
16. ?>
17. <input type="submit" value="Click To Submit">
18. <input type="reset" value="Erase and Restart">
19. </form></body></html>
```

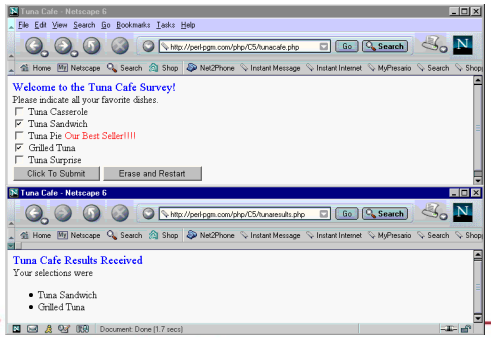
Create a list of menu items.

This array will be available to the receiving script when the form is submitted.

23

The Output ...

The previous code can be executed at



24

Using Arrays to Receive Multiple Form Element Selections

- Suppose you want to receive these multiple items, set as:

```
print "<input type=\"checkbox\" name=\"prefer[]\"
  value=$i> $menu[$i]";
```

- If the user selects the first and third check box items shown then $prefer[] would be an array of two items:
  - $prefer[0], would have a value of 0, and $prefer[1] would be 2.

25

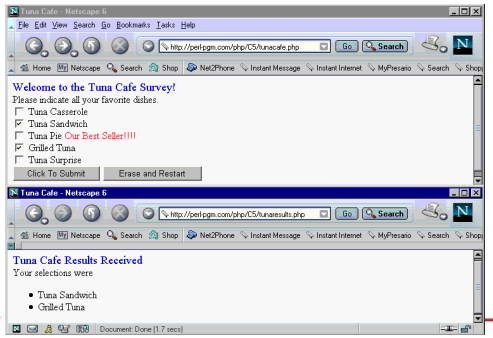
Receiving Code

```
1. <html>
2. <head><title> Tuna Cafe </title></head>
3. <body>
4. <font size=4 color="blue"> Tuna Cafe Results Received </font>
5. <?php
6.      $perfer = $_POST["prefer"];
7.      $menu = array('Tuna Casserole', 'Tuna Sandwich', 'Tuna Pie',
                                        'Grilled Tuna', 'Tuna Surprise');
8.     if (count($prefer) == 0 ) {
9.        print 'Oh no! Please pick something as your favorite! ';
10.    } else {
11.      print '<br>Your selections were <ul>';
12.      foreach ($prefer as $item) {
13.          print "<li>$menu[$item]</li>";
14.    }
15.    print '</ul>';
16. }
17. ?>
18. </body></html>
```


26

The Output ...

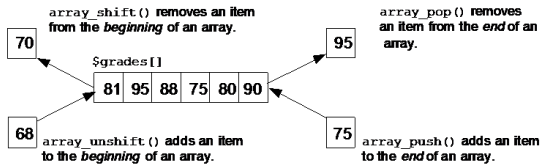
The previous code can be executed at



27

2.5. More Arrays Operations

- Adding and Deleting Items



28

a. The `array_shift()` functions

- array_shift() accepts an array as an argument, removes the first item, and then returns the removed item.
- For example,

```
$work_week = array('Monday','Wednesday', 'Friday');
$day_off = array_shift($work_week);
print "Day off = $day_off Work week = ";
foreach ($work_week as $day) {
    print "$day ";
}
```

The above outputs:

"Day off = Monday Work week = Wednesday Friday"

29

b. The `array_unshift()` functions

- array_unshift() used to *add* an item to the *beginning* of the array.
- It accepts as arguments an array variable and an item to add. For example,

```
$work_week = array('Monday', 'Wednesday', 'Friday');
array_unshift($work_week, 'Sunday');
print 'Work week is now = ';
foreach ($work_week as $day) {
    print "$day ";
}
```

The above outputs:

"Work week is now = Sunday Monday Wednesday Friday".

30

c. The `array_pop()` functions

- array_pop() accepts an array variable as an argument and returns an item it removed from the end of the array.
- For example,

```
$work_week = array('Monday', 'Wednesday', 'Friday');
$day_off = array_pop($work_week);
print "Day off = $day_off Work week = ";
foreach ($work_week as $day) {
    print "$day ";
}
```

The above outputs:

"Day off = Friday Work week = Monday Wednesday"

31

d. The `array_push()` functions

- array_push() accepts an array variable and an item as arguments and adds the item to the end of an array.
- For example, the following code:

```
$work_week = array('Monday', 'Wednesday','Friday');
array_push($work_week, 'Saturday');
print 'Work week is now = ';
foreach ($work_week as $day) {
    print "$day ";
}
```

The above outputs:

"Work week is now = Monday Wednesday Friday Saturday"

32

e. Additional Useful Array Functions

- Use `max()` and `min()` to find the largest and smallest number in an array.

```
$grades = array (99, 100, 55, 91, 65, 22, 16);
$big=max($grades);
$small=min($grades);
print "max=$big small=$small";
```

The above would output:

"max=100 small=16".

33

e. Additional Useful Array Functions (2)

- Use array_sum() to return a sum of all numerical values.
- For example,

```
$grades = array (25, 100, 50, 'N/A');
$total=array_sum($grades);
print "Total=$total";
```

- The above would output:

"Total=175"

34

Mixing Variable Types

- PHP will try to convert character to numerical values when it can. For example,

```
<?php
$grades = array ('2 nights', '3days', 50, '1 more day');
$total=array_sum($grades);
print "total=$total";
?>
```

- Instead of generating an error message, this code outputs "total=56".

35

Splitting/joining strings

- `explode` and `implode` convert between strings and arrays

```
$array = explode(delimiter, string);
$string = implode(delimiter, array);
```

PHP

```
$class = "CS 380 01";
$class1 = explode(" ", $s); # ("CS", "380", "01")
$class2 = implode("...", $a); # "CSE...380...01"
```

PHP

36

Splitting/joining strings

- `explode` example

```
Harry Potter, J.K. Rowling
The Lord of the Rings, J.R.R. Tolkien
Dune, Frank Herbert
contents of input file books.txt
```

```php
<?php foreach (file("books.txt") as $book) {
   list($title, $author) = explode(",", $book);
   ?>
   <p> Book title: <?= $title ?>, Author: <?=
$author ?> </p>
<?php
}
?>
```

PHP

37

More Arrays Operations

- array_keys($someArray)
- array_values($someArray)
- array_rand($someArray, $num=1)
- array_reverse($someArray)
- array_walk($someArray, $callback, $optionalParam)
- in_array($needle, $haystack, $optionalStrict)
- shuffle($someArray)
- array_search(), array_merge(), array_unique(),…

38

Content

1. Benefits of arrays
2. Sequential arrays
3. Non-sequential arrays
4. Multidimensional lists

39

3. Non-sequential arrays

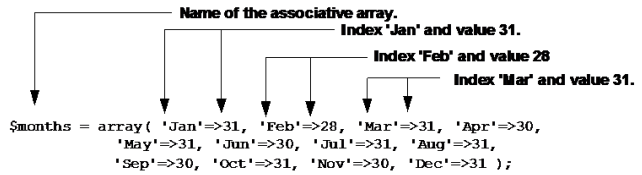
- PHP also supports arrays with string-value indices called non-sequential/associative arrays.
  - String-value index is used to look up or provide a cross-reference to the data value
  - For example, the following code creates an associative array with three items

```
$instructor['Science'] = 'Smith';
$instructor['Math'] = 'Jones';
$instructor['English'] = 'Jackson';
```


40

3.1. Creating Associative Arrays

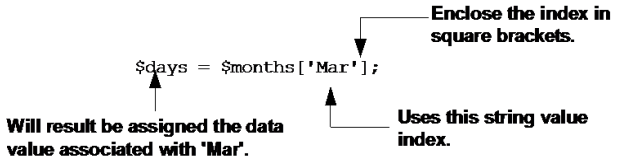
- Use the array() function along with the => operator to create an associative array



41

3.2. Accessing Associative Array Items

- Use a syntax similar to sequential arrays to access items



42

WARNING You Cannot Fetch Indices by Using Data Values

- You might be tempted to use a data item to fetch an index from an associative array, as in the following example:
  - `$mon = $months[28];`
- This syntax is incorrect because associative arrays can fetch data values only by using indices (not the other way around)

43

Consider the following example ...

- Consider an application that reports distance between Chicago and destination cities

```
<select name="destination" size=3>
<option> Boston </option>
<option> Dallas </option>
<option> Las Vegas </option>
<option> Miami </option>
<option> Nashville </option>
<option> Pittsburgh </option>
<option> San Francisco </option>
<option> Toronto </option>
<option> Washington, DC </option>
</select>
```

- When user selects destination city the application reports distance from Chicago

44

Example script source

```
1. <html>
2. <head><title> Distance and Time Calculations </title></head>
3. <body>
4. <?php
5.   $destination = $_POST["destination"];
6.   $cities = array ('Dallas' => 803, 'Toronto' => 435, 'Boston' => 848, 'Nashville' => 406, 'Las Vegas' => 1526, 'San Francisco' =>
       1835, 'Washington, DC'=> 595, 'Miami' => 1189, 'Pittsburgh' => 409);
7. if (isset($cities[$destination])) {
8.     $distance = $cities[$destination];
9.     $time = round( ($distance / 60), 2);
10.    $walktime = round( ($distance / 5), 2);
11.    print "The distance between Chicago and <i>$destination</i> is $distance miles.";
12.    print "<br>Driving at 60 miles per hour it would take $time hours.";
13.    print "<br>Walking at 5 miles per hour it would take $walktime hours.";
14. } else {
15.    print "Sorry, do not have destination information for $destination.";
16. } ?>
17. </body></html>
```

Associative array containing destination city and distance.

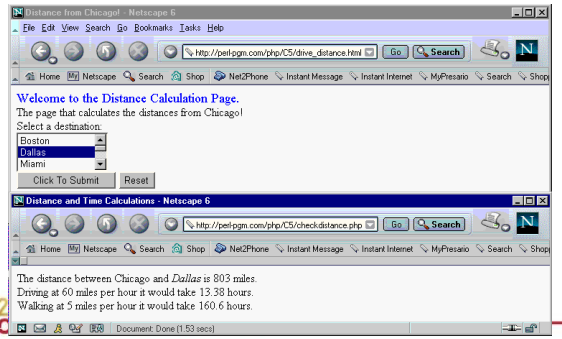
Check if the input destination city has a value in $cities[].

Round results to 2 digits to the right of the decimal point.

45

The Output ...

The previous code can be executed at



46

3.3. Using foreach with associative arrays

- You can use foreach to access items from an associative array

```
foreach ($courses as $index => $item) {
        Set of statements to repeat.
}
```

Array Name

Index variable ($index) is automatically set to next array index each iteration.

Item variable ($item) is automatically set to next array value each iteration.

47

3.3. Using foreach with associative arrays (2)

- Consider the following:

```
$inventory = array('Nuts'=>33, 'Bolts'=>55, 'Screws'=>12);
foreach ($inventory as $index => $item) {
     print "Index is $index, value is $item<br> ";
}
```

- The above outputs:

```
Index is Nuts, value is 33
Index is Bolts, value is 55
Index is Screws, value is 12
```


48

3.4. Changing adding/deleting items

- You can change an item by giving it a new value:

```
$inventory = array('Nuts'=> 33, 'Bolts'=> 55,
  'Screws'=> 12);
$inventory['Nuts'] = 100;
```

- You can add an item as follows:

```
$inventory = array('Nuts'=>33, 'Bolts'=>55, 'Screws'=>12);
$inventory['Nails'] = 23;
```

- You can delete an item as follows:

```
$inventory = array('Nuts'=> 33, 'Bolts'=>55, 'Screws'=> 12);
unset($inventory['Nuts']);
```


49

3.5. Verifying an items existance

- You can use the isset() function to verify if an item exists.

```
$inventory = array('Nuts'=> 33,'Bolts'=>55,'Screws'=> 12);
if (isset($inventory['Nuts'])) {
   print ('Nuts are in the list.');
} else {
   print ('No Nuts in this list.');
}
```


50

Warning indices are case sensitive

- Examine the following lines:

```
$inventory = array( 'Nuts'=> 33,'Bolts'=>55,'Screws'=>12);
$inventory['nuts'] = 32;
```

- Results in items 'Nuts', 'Bolts', 'Screws', and 'nuts'

51

A Full Application

- Consider an application using the following radio buttons:

```
<input type="radio" name="Action" value="Add" > Add
<input type="radio" name="Action" value="Unknown" > Unknown
<br>Enter Index: <input type="text" name="index" size=10>
Enter Value: <input type="text" name="value" size=10>
```

- It "simulates" adding an inventory item

  That is, it adds it to associative array but does not save to a file or database.

52

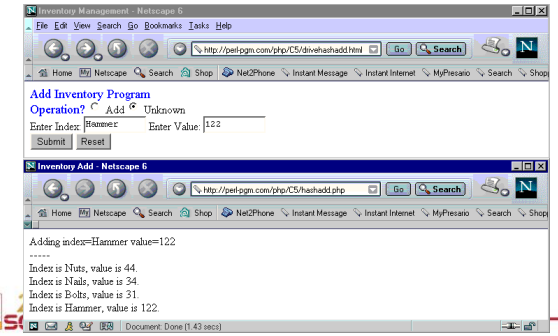
PHP Source ...

```
1.  <html><head><title>Inventory Add </title>
2.  </head><body>
3.  <?php
4.  $invent = array('Nuts'=>44, 'Nails'=>34, 'Bolts'=>31);
5.  if ($Action == 'Add'){
6.     $item=$invent["$index"];
7.     if (isset($invent["$index"])) {
8.        print "Sorry, already exists $index <br>";
9.     } else {
10.   $invent["$index"] = $Value;
11.   print "Adding index=$index value=$Value <br>";
12.   print '-----<br>';
13.   foreach ($invent as $index => $item) {
14.      print "Index is $index, value is $item.<br> ";
15.   }
16.   }
17. } else { print "Sorry, no such action=$Action<br>"; }
18. ?></body></html>
```


53

Would output the following:

The previous code can be executed at



54

3.6. Sorting Associative Arrays

- You can sort associative arrays by values or indices.
- Use asort() to sort by values:

```
$dest = array('Dallas' => 803, 'Toronto' => 435,
              'Boston' => 848, 'Nashville' => 406,
              'Las Vegas' => 1526);
asort($dest);
foreach ($dest as $index => $value) {
       print " $index = $value ";
}
```

- The above would output:

```
Nashville = 406  Toronto = 435  Dallas = 803  Boston = 848
  Las Vegas = 1526
```


55

3.6. Sorting Associative Arrays (2)

- Use ksort() to sort by indices:

```
$dest = array ('Dallas' => 803, 'Toronto' => 435,
               'Boston' => 848, 'Nashville' => 406,
               'Las Vegas' => 1526);
       ksort($dest);
       foreach ($dest as $index => $value) {
              print " $index = $value ";
        }
```

- The above would output:

```
Boston = 848  Dallas = 803  Las Vegas = 1526  Nashville =
  406  Toronto = 435
```


56

Content

1. Benefits of arrays
2. Sequential arrays
3. Non-sequential arrays
4. Multidimensional lists

57

4. Multiple dimensional lists

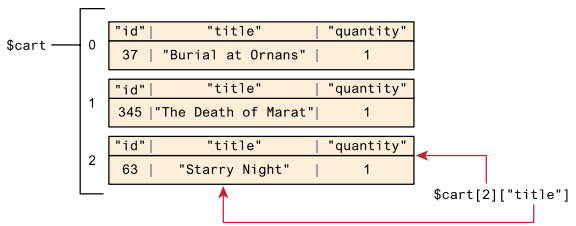
- Some data is best represented using a list of list or a multi-dimensional list.
- For example:

Part Number	Part Name	Count	Price
AC1000	Hammer	122	12.50
AC1001	Wrench	5	5.00
AC1002	Handsaw	10	10.00
AC1003	Screwdriver	222	3.00

58

4. Multiple dimensional lists

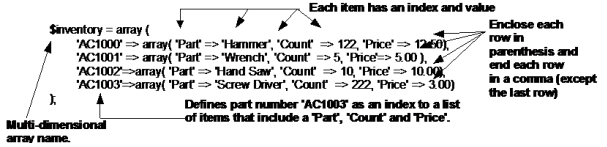
```
$cart = array();
$cart[] = array("id" => 37, "title" => "Burial at Ornans", quantity" => 1);
$cart[] = array("id" => 345, "title" => "The Death of Marat", "quantity" => 1);
$cart[] = array("id" => 63, "title" => "Starry Night", "quantity" => 1);
```



59

4.1. Creating Multidimensional Lists

- You can create multidimensional arrays with the array() function




```
$inventory['AC1000']['Part'] has the value Hammer,
$inventory['AC1001']['Count'] has the value 5,
and $inventory['AC1002']['Price'] has the value
   10.00.
```


60

A Full Application

- Application that receives a part number and then returns information about the part
  - Uses the following HTML form:

```
<input type="radio" name="id" value="AC1000"> AC1000
<input type="radio" name="id" value="AC1001"> AC1001
<input type="radio" name="id" value="AC1002"> AC1002
<input type="radio" name="id" value="AC1003"> AC1003
```


61

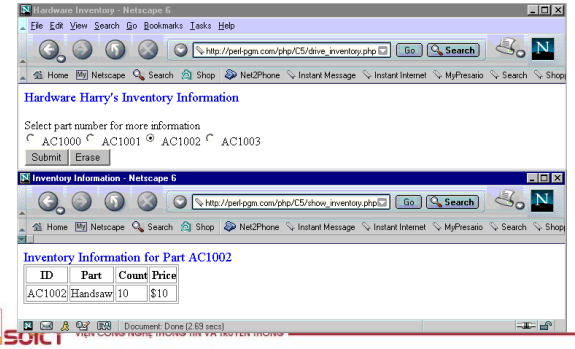
PHP Script Source

```
1.  <html><head><title>Inventory Information</title>
2.  </head><body>
3.  <?php  $id = $_POST["id"];
4.   $inventory = array (
   'AC1000'=>array('Part'=>'Hammer','Count'=>122, 'Price'=> 12.50 ),
   'AC1001' => array('Part' =>'Wrench','Count' =>5, 'Price'=>5.00 ),
   'AC1002'=>array('Part' =>'Handsaw','Count' =>10, 'Price'=>10.00 ),
   'AC1003'=>array('Part' =>'Screwdrivers','Count'=>222, 'Price'=>3.00)
     );
5.  if (isset($inventory[$id])){
6.    print '<font size=4 color="blue"> ';
7.    print "Inventory Information for Part $id </font>";
8.    print '<table border=1> <th> ID <th> Part <th> Count <th> Price ';
9.    print "<tr> <td> $id </td>";
10.  print "<td> {$inventory[$id]['Part']} </td>";
11.  print "<td> {$inventory[$id]['Count']} </td>";
12.  print "<td> \${$inventory[$id]['Price']} </td></tr>";
13. } else {
14.   print "Illegal part ID = $id ";
15. }
16. ?></body></html>
```


62

Would output the following ...

The previous code can be executed at



63

Superglobal Arrays

- PHP uses special predefined associative arrays called **superglobal variables** that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information

64

Superglobal Arrays

- **$GLOBALS** Array for storing data that needs superglobal scope
- **$_COOKIES** Array of cookie data passed to page via HTTP request
- **$_ENV** Array of server environment data
- **$_FILES** Array of file items uploaded to the server
- **$_GET** Array of query string data passed to the server via the URL
- **$_POST** Array of query string data passed to the server via the HTTP header
- **$_REQUEST** Array containing the contents of $_GET, $_POST, and $_COOKIES
- **$_SESSION** Array that contains session data
- **$_SERVER** Array containing information about the request and the server

65

Superglobal Arrays

❖ **$_GET and $_POST**

```
<form action="processLogin.php" method="GET">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

HTML (client) → Browser (client) → HTTP request → PHP (server)

Name ricardo Pass pw01 Submit Query

GET processLogin.php?uname=ricardo&pass=pw01

```
// within processLogin.php
echo $_GET["uname"]; // outputs ricardo
echo $_GET["pass"];  // outputs pw01
```


66

Superglobal Arrays

❖ **$_GET and $_POST**

```
<form action="processLogin.php" method="POST">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

HTML (client) → Browser (client) → HTTP request → PHP (server)

Name ricardo Pass pw01 Submit Query

POST processLogin.php

HTTP POST request body:
uname=ricardo&pass=pw01

```
//File processLogin.php
echo $_POST["uname"]; //outputs "ricardo";
echo $_POST["pass"];  //outputs "pw01";
```


67

Superglobal Arrays

❖ **$_GET and $_POST:** Determining If Any Data Sent

❖ use the **isset()** function in PHP to see if there is any value set for a particular expected key

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if ( isset($_POST["uname"]) && isset($_POST["pass"]) ) {
                // handle the posted data.
```


68

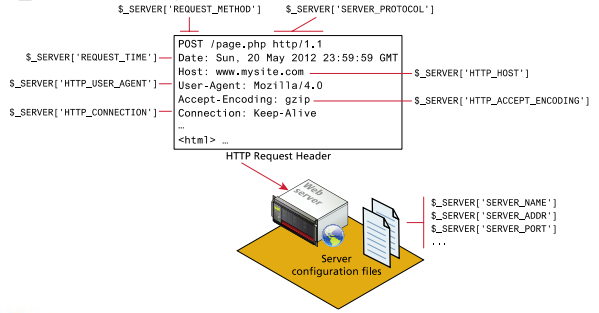
Superglobal Arrays

- **$_GET and $_POST:** Sanitizing Query Strings
- That is, just because you are expecting a proper query string, it doesn't mean that you are going to get one. your program must be able to handle:
  - If query string parameter doesn't exist.
  - If query string parameter doesn't contain a value.
  - If query string parameter value isn't the correct type or is out of acceptable range.
  - If value is required for a database lookup, but provided value doesn't exist in the database table.

69

Superglobal Arrays

- **$_SERVER Array**



70

Superglobal Arrays

- **$_SERVER Array:** Request Header Information

```
<?php
echo $_SERVER['HTTP_USER_AGENT'];
//advanced browser detection
$browser =
get_browser($_SERVER['HTTP_USER_AGENT'], true);
print_r($browser);
?>
```


71

Superglobal Arrays

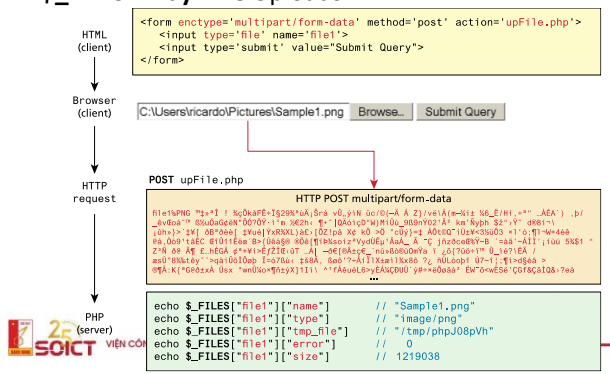
- **$_FILES Array:** File Uploads
- First, you must ensure that the HTML form uses the HTTP POST method
- Second, you must add the enctype="multipart/form-data" attribute to the HTML form that is performing the upload
- Finally you must include an input type of file in your form.

```
<form enctype='multipart/form-data' method='post'>
        <input type='file' name='file1' id='file1'>
        <input type='submit'>
</form>
```


72

Superglobal Arrays

- **$_FILES Array:** File Uploads



73

Superglobal Arrays

- **$_FILES Array:** Checking for Errors

```
foreach ($_FILES as $fileKey => $fileArray) {
        if ($fileArray["error"] != UPLOAD_ERR_OK) { // error
                echo "Error: " . $fileKey . " has error" .
        $fileArray["error"] . "<br>";
        }
        else { // no error
                echo $fileKey . "Uploaded successfully ";
        }
}
```


74

Superglobal Arrays

- **$_FILES Array:** File Size Restrictions
- You can limit in multiple ways
  - HTML form attributes in inputs (browser)
  - JavaScript (browser)
  - Php validation (server)

75

Superglobal Arrays

- **$_FILES Array:** File Size Restrictions

```
$validExt = array("jpg", "png");
$validMime = array("image/jpeg","image/png");
foreach($_FILES as $fileKey => $fileArray ){
        $extension = end(explode(".", $fileArray["name"]));
        if (in_array($fileArray["type"],$validMime) &&
in_array($extension, $validExt)) {
                echo "All is well. Extension and mime types valid";
        }
        else {
                echo $fileKey." has an invalid mime type or extension";
        }
}
```


76

Superglobal Arrays

- **$_FILES Array:** Moving the File

```
$fileToMove = $_FILES['file1']['tmp_name'];
$destination = "./upload/" . $_FILES["file1"]["name"];
if (move_uploaded_file($fileToMove,$destination)) {
        echo "The file was uploaded and moved
successfully!";
}
else {
        echo "There was a problem moving the file.";
}
```


77

Summary

- Using arrays helps you organize data into lists instead of separate variables.
- Sequential arrays use indices numbered with sequential numbers. By default indices start numbering from 0, then 1, 2, 3, and so on.
  - You can use the `for` loop and `foreach` loop to concisely examine the various items within an array..

78

Summary

- Associative arrays use string value indices rather than numerical values. They are useful for cross-referencing an index with a value.
  - You can use the `foreach` loop to concisely examine the various items within an associative array.

79

Question?

80