

CSCI 241

Arrays in PHP

*Department of CS & QM
Winthrop University
Fall 2011*

Prepared by X. Wang

Arrays

- Arrays in PHP are sophisticated and more flexible than in many other high-level languages
- A PHP array is an ordered set of *variables*, in which each variable – called an *element* – has an associated *key*
- PHP allows elements to be accessed using either integer or string keys
- PHP automatically assigns integer key values if keys are not specified when arrays are constructed

Prepared by X. Wang 2

Arrays

- Arrays can hold scalar values (integers, floats, booleans and strings) or compound values (objects and even other arrays)
- The same array can even hold elements of different types

Prepared by X. Wang 3

Creating Arrays

- PHP provides the `array()` function to construct an array
- `$number = array(50, 40, 30, 20, 10);` creates an array as the following left one
- `$words = array("Web", "Database", "Applications");` creates an array as the following right one

Keys	Values
0	50
1	40
2	30
3	20
4	10

Keys	Values
0	Web
1	Database
2	Applications

Prepared by X. Wang 4

Creating Arrays

- We can create an array by directly assigning *values* to a new *variable* with indices (*keys*)
- `$language[2] = "Java";`
`$language[1] = "PHP";`
`$language[5] = "C++";`
creates the following array `$language`

Keys	Values
2	Java
1	PHP
5	C++

Prepared by X. Wang 5

Creating Arrays

- An empty array can be created by assigning `array()` to a variable
- Values can be assigned to the element with or without keys
- `$shopping = array();`
`$shopping[] = "Milk";`
`$shopping[] = "Coffee";`
`$shopping[6] = "Sugar";`

Keys	Values
0	Milk
1	Coffee
6	Sugar

Prepared by X. Wang 6

Printing Array Elements

- To print elements in an array, we can use the keys

```
print $shopping[0]; // print "Milk"
print $shopping[1]; // print "Coffee"
print $shopping[6]; // print "Sugar"
```
- The function `print_r()` can be used to print all values with their keys in an array
- `print_r($shopping);` will print the following result

```
Array ( [0] => Milk [1] => Coffee [6] => Sugar )
```

Prepared by X. Wang

7

Printing Array Elements

- Using `print_r` with `<pre>` tag can generate the following output format
- ```
print "<pre>";
print_r($shopping);
print "</pre>";
```

 will print the following result  

```
Array
(
 [0] => Milk
 [1] => Coffee
 [6] => Sugar
)
```

Prepared by X. Wang

8

### Printing Array Elements

- The function `var_dump()` can print an array by specifying the data type of every element
- `var_dump($shopping);` will print the following result  

```
array(3) { [0]=> string(4) "Milk" [1]=>
string(6) "Coffee" [6]=> string(5) "Sugar"
}
```
- The `<pre>` can also be used to create a vertical list

Prepared by X. Wang

9

### Associative Arrays

- An associative array uses string keys to access values stored in an array
- `$array = array("first"=>1,"second"=>2,"third"=>3);` creates an array as the right  

| Keys   | Values |
|--------|--------|
| first  | 1      |
| second | 2      |
| third  | 3      |
- `print_r($array);` will print the result:  

```
Array ([first] => 1 [second] => 2 [third] => 3)
```

Prepared by X. Wang

10

### Removing elements of an Array

- The function `unset()` can remove an element or all elements from an array
- ```
$langs = array("PHP","Ace","COBOL","Java","C++");
unset($langs[2]);
print_r($langs);
```

 will print:

```
Array ( [0]=>PHP [1]=>Ace [3]=>Java [4]=>C++ )
```
- `unset($langs);` will destroy the whole array `$langs`

Prepared by X. Wang

11

Heterogeneous Arrays

- The values stored in a single array don't have to be the same type
- This means that PHP arrays can contain heterogeneous values
- ```
$mixedBag = array("cat", 42, 8.5, false);
var_dump($mixedBag);
```

 will print the result:  

```
array(4) { [0]=> string(3) "cat" [1]=> int(42)
[2]=> float(8.5) [3]=> bool(false) }
```

Prepared by X. Wang

12

## Multidimensional Arrays

- > If an element of an array is another array, then multidimensional array is created
- > 

```
$planets = array(array("Mercury", 0.39, 0.38),
array("Venus", 0.72, 0.95),
array("Earth", 1.0, 1.0));
```

creates a two dimensional array

- > 

```
print $planets[2][0];
```

will print "Earth"

|   |   |         |
|---|---|---------|
| 0 | 0 | Mercury |
|   | 1 | 0.39    |
|   | 2 | 0.38    |
| 1 | 0 | Venus   |
|   | 1 | 0.72    |
|   | 2 | 0.95    |
| 2 | 0 | Earth   |
|   | 1 | 1.0     |
|   | 2 | 1.0     |

Prepared by X. Wang

13

## foreach Loops with Arrays

- > The easiest way to traverse or iterate through an array is using the `foreach` statement
- > The foreach statement has two forms:
- > 

```
foreach(array as $value) {
// body of loop
}
```
- > 

```
foreach(array as $key => $value) {
// body of loop
}
```

Prepared by X. Wang

14

## foreach Loops with Arrays

- > 

```
$lengths = array(0,107,202,400,475);
foreach ($lengths as $cm) {
$inch = $cm/2.54;
print "$cm centimeters = $inch inches
";
}
```

will print the result as

```
0 centimeters = 0 inches
107 centimeters = 42.125984252 inches
202 centimeters = 79.5275590551 inches
400 centimeters = 157.480314961 inches
475 centimeters = 187.007874016 inches
```

Prepared by X. Wang

15

## foreach Loops with Arrays

- > 

```
$sounds = array("Cow"=>"moo", "Dog"=>"woof",
"Pig"=>"oink","Duck"=>"quack");
foreach ($sounds as $animal => $sound) {
print "$animal sounds $sound.
";
}
```

will print the result as

```
Cow sounds moo.
Dog sounds woof.
Pig sounds oink.
Duck sounds quack.
```

Prepared by X. Wang

16

## while Loops with Arrays

- > We can also use a `while` loop with `each()` and `list()` functions to traverse the elements in an array
- > 

```
while (list($animal, $sound) = each($sounds)) {
print "$animal sounds $sound.
";
}
```

will print the same result as the previous slide

- > The `each()` function returns the `key` and `value` of current element as a new array
- > The `list()` function assigns the `values` of current array to the `variables`

Prepared by X. Wang

17

## Traverse Arrays with Functions

- > The following functions can be used to traverse the elements in an array
- > `current($arr)`, returns the current value in the array `$arr`, if it has
- > `key($arr)`, return the key in the array `$arr`
- > `next($arr)`, moves the pointer to the next element and returns the next value in the array `$arr`, if it exists
- > `prev($arr)`, moves the pointer to the previous element and return the previous value in the array `$arr`, if it exists
- > `reset($arr)`, moves the pointer to the first element and return the first value in the array `$arr`, if it exists
- > `end($arr)`, moves the pointer to the end element and return the end value in the array `$arr`, if it exist

Prepared by X. Wang

18

## Basic Array Functions

- Counting elements in an array
- The `count()` function returns the number of elements in the array `var`:  
`integer count(mixed var)`
- To make sure `var` is an array, the `is_array()` function may be called first
- ```
$days = array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun");
if (is_array($days))
    print "A week has " . count($days) . " days";
```

Prepared by X. Wang

19

Basic Array Functions

- Counting values in an array
- The `array_count_values()` function counts the instances of values in an array `var` and creates a new array
`array array_count_values(mixed var)`
- ```
$pets = array("Beth"=>"Dog", "Arabella"=>"rabbit", "Meg"
=>"Cat", "Louise"=>"Chicken", "Ben"=>"Dog", "Neca"=>"Cat");
$petFreq = array_count_values($pets);
print_r($petFreq);
```

  
will show a new array with the number of different values  
Array ([Dog] => 2 [rabbit] => 1 [Cat] => 2 [Chicken] => 1)

Prepared by X. Wang

20

## Basic Array Functions

- Functions that create arrays
- PHP provides two functions that can create new arrays with pre-filled values  
`array array_fill(integer start, integer count, mixed value)`  
`array range(mixed low, mixed high [, integer step])`
- `array_fill()` creates an array with `count` elements, the keys starting at `start`, all set to the same `value`
- `range()` returns a new array filled with a sequence of element starting from the value `low` to the value `high` by using the optional `step` (default step is 1)

Prepared by X. Wang

21

## Basic Array Functions

- Functions that create arrays
- ```
$unity = array_fill(2, 5, "one");
print_r($unity);
```

will print the new array \$unity as
Array ([2] => one [3] => one [4] => one [5] => one [6] => one)
- ```
$letters = range("A", "F");
print_r($letters);
```

will print the new array \$unity as  
Array ( [0] => A [1] => B [2] => C [3] => D [4] => E [5] => F )

Prepared by X. Wang

22

## Basic Array Functions

- Exploding and imploding (joining) strings
- PHP provides three functions to convert strings to arrays and back to strings  
`array explode(string separator, string subject [, integer limit])`  
`string implode(string glue, array pieces)`  
`string join(string glue, array pieces)`

Prepared by X. Wang

23

## Basic Array Functions

- Exploding and imploding (joining) strings
- `explode()` returns an array of strings by breaking the `subject` string at each occurrence of the `separator` string  

```
$words = explode(" ", "Now is the time.");
print_r($words);
```

  
The output is:  
Array ( [0] => Now [1] => is [2] => the [3] => time. )
- The optional integer `limit` determines the maximum number of elements in the resulting array  

```
$words = explode(" ", "Now is the time.", 2);
```

Prepared by X. Wang

24

## Basic Array Functions

- ❑ Exploding and imploding (joining) strings
  - Both `implode()` and `join()` return a string by joining each element in the array pieces, inserting the string glue between each piece
- ```
$animals = array("kangaroo","wombat","dingo","echidna");
print "Animals include: " . implode(" ", $animals);
```
- The output is:
- ```
Animals include: kangaroo, wombat, dingo, echidna
```

Prepared by X. Wang

25

## Basic Array Functions

- ❑ Finding the maximum and minimum values in an array
  - PHP provides the `max()` and `min()` functions to find the maximum and minimum values from an array
- ```
number max(array var)
number min(array var)
```
- `$letters = range("a", "g");`
`print "Maximum value is: " . max($letters) . "
;`
`print "Minimum value is: " . min($letters) . "
;`
- print the output:
- ```
Maximum value is: g
Minimum value is: a
```

Prepared by X. Wang

26

## Basic Array Functions

- ❑ Finding a value in an array
  - The `in_array()` function returns `true` if the array `var` contains the specific value `needle`
- ```
boolean in_array(mixed needle,array var [,boolean strict])
```
- The `array_search()` function works the same way as the `in_array`, except the `key` of the matching value `needle` is returned rather than the boolean value `true`
- ```
mixed array_search(mixed needle,array var [,boolean strict])
```
- If the value `needle` isn't found, `array_search()` returns `false`

Prepared by X. Wang

27

## Basic Array Functions

- ❑ Finding a value in an array
  - `$measure=array("inch"=>1,"foot"=>12,"yard"=>36,"cent"=>12;`  
`$var = 12;`  
`if (in_array($var,$measure))`  
`print "{$var} is in the array<br />;`  
`print "The key is: ".array_search($var,$measure)."<br />;`
  - The output is:
- ```
12 is in the array
The key is: foot
```

Prepared by X. Wang

28

Basic Array Functions

- ❑ Keys and Values of an array
 - The `array_key_exists()` function tests whether an element in the `source` array associated with the `key`
- ```
boolean array_key_exists(mixed key, array source)
```
- The `array_keys()` function creates a new array with the `keys` of the `input` array. If the `search_value` is specified, the function can also be used to find the key
- ```
array array_keys(array input [,mixed search_value])
```

Prepared by X. Wang

29

Basic Array Functions

- ❑ Keys and Values of an array
 - The `array_values()` function creates a new array with the values of the `input` array and adds numeric keys
- ```
array array_values(array input)
```
- The `array_unique()` function creates a new array with the unique values from the `input` array
- ```
array array_unique(array input)
```

Prepared by X. Wang

30

Basic Array Functions

Keys and Values of an array

An example:

```
$pets = array("Beth"=>"Dog", "Arabella"=>"Rabbit", "Meg"=>"Cat", "Louise"=>"Chicken", "Ben"=>"Dog", "Neda"=>"Cat");
$owner = "Eddie";
if (array_key_exists($owner, $pets))
    print "{$owner} has a {$pets[$owner]} as a pet<br />";
else {
    print "{$owner} doesn't have a pet.<br />";
    print "Pet owners are: ".join(" ", array_keys($pets))."<br />";
}
$dogOwners = array_keys($pets, "Dog");
print_r($dogOwners);
print "<br />";
$petTypes = array_values($pets);
print_r($petTypes);
print "<br />";
$uniquePetTypes = array_unique($petTypes);
print_r($uniquePetTypes);
```

Prepared by X. Wang

31

Basic Array Functions

Keys and Values of an array

The output is:

```
Eddie doesn't have a pet.
Pet owners are: Beth, Arabella, Meg, Louise, Ben, Neda
Array ( [0] => Beth [1] => Ben )
Array ( [0] => Dog [1] => Rabbit [2] => Cat [3] => Chicken
[4] => Dog [5] => Cat )
Array ( [0] => Dog [1] => Rabbit [2] => Cat [3] => Chicken )
```

Prepared by X. Wang

32

Basic Array Functions

Merging arrays

- Arrays can be merged using the + operator. However, values with the same key are overwritten
- PHP calculates array addition (+) from right to left
- In contrast, the `array_merge()` function appends two or more arrays together without overwriting values

```
array array_merge(array array1, array array2
[, array...])
```

Prepared by X. Wang

33

Basic Array Functions

Merging arrays

An example

```
$clothing = array("silk", "satin", "cotton", "rags");
$dwelling = array("house", "tree", "palace");
$scintoss = array("heads", "tails");

$added = $scintoss + $dwelling + $clothing;
print_r($added);
print "<br />";

$mmerged = array_merge($scintoss, $dwelling, $clothing);
print_r($mmerged);
print "<br />";
```

The output is:

```
Array ( [0] => heads [1] => tails [2] => palace [3] => rags )
Array ( [0] => heads [1] => tails [2] => house [3] => tree [4] => palace [5] => silk [6] => satin [7] => cotton [8] => rags )
```

Prepared by X. Wang

34

Basic Array Functions

Reverse elements (values) of an array

- The `array_reverse()` function returns a new array by reversing the elements from a source array

```
array array_reverse(array source
[, boolean preserve_keys])
```

- If `preserve_keys` is true, the elements are reversed, and the association between elements and keys (indices) is preserved

Prepared by X. Wang

35

Basic Array Functions

Reverse elements (values) of an array

An example

```
$count = array("zero", "one", "two", "three", "four");
print_r($count);
print "<br />";
$countdown = array_reverse($count);
print_r($countdown);
print "<br />";
$countdown = array_reverse($count, true);
print_r($countdown);
print "<br />";
```

The output is:

```
Array ( [0] => zero [1] => one [2] => two [3] => three [4] => four )
Array ( [0] => four [1] => three [2] => two [3] => one [4] => zero )
Array ( [4] => four [3] => three [2] => two [1] => one [0] => zero )
```

Prepared by X. Wang

36

Sorting Arrays

- Sorting non-associative (indexed) arrays (p. 69 – 70)
- Two simple array sorting function are `sort()` and `rsort()`, which rearrange the *values* of the `subject` array in ascending and descending order, respectively


```
sort(array subject [, integer sort_flag])
rsort(array subject [, integer sort_flag])
```
- The optional `sort_flag` can be:

SORT_STRING, SORT_NUMERIC, SORT_REGULAR

Prepared by X. Wang

37

Sorting Arrays

- Sorting associative arrays
- If we want to keep the key/value associations of an array, we can use the `asort()` and `arsort()` functions


```
asort(array subject [, integer sort_flag])
arsort(array subject [, integer sort_flag])
```
- The sort order reflects the element values in the array, not the keys
- When `asort()` and `arsort()` are used on non-associative arrays, the indexes (keys) to the values are not changed

Prepared by X. Wang

38

Sorting Arrays

- Sorting on keys
- Rather than sort on element values, the `ksort()` and `krsort()` functions rearrange elements by sorting on the keys (indexes)


```
ksort(array subject [, integer sort_flag])
krsort(array subject [, integer sort_flag])
```
- The values associated with the keys are not changed

Prepared by X. Wang

39