Basic PHP 3

Associative Arrays

- Like Python Dictionaries - but more powerful
- PHP Arrays have all the benefits of Python Dictionaries but they can also maintain the order of the items in the array
- Can be key => value or simply indexed by numbers
- Ignore two-dimensional arrays for now..

Integer Indices

```
<?php
$stuff = array("Hi","There");
echo $stuff[1] , "\n";
?>
```

```
There
```

Key / Value

```
<?php
$stuff = array("name" => "Liu",
               "course" => "TU856");
echo $stuff["course"] , "\n";
?>
```

```
TU856
```

Dumping an Array

- The function print_r() dumps out PHP data - it is used mostly for debugging

```
<?php
$stuff = array("name" => “Liu",
               "course" => “TU856");
print_r($stuff);
?>
```

```
Array
(
[name] => Liu
[course] => TU856
)
```

Building up an Array

- You can allocate a new item in the array and add a value at the same time using empty square braces [] on the right hand side of an assignment statement

```
$va = array();
$va[] = "Hello";
$va[] = "World";
print_r($va);
```

```
Array
(
[0] => Hello
[1] => World
)
```

Building up an Array

- You can also add new items in an array using a key as well

```
$za = array();

$za["name"] = “Liu";

$za["course"] = “TU856";

print_r($za);$va = array();
```

```
Array
(
[name] => Liu
[course] => TU856
)
```

Array Type Casting and Overwriting

- The key can either be an integer or a string. The value can be of any type.
- Additionally the following key casts will occur:
  - Strings containing valid integers will be cast to the integer type. E.g. the key "8" will actually be stored under 8. On the other hand "08" will not be cast, as it isn't a valid decimal integer.
  - Floats are also cast to integers, which means that the fractional part will be truncated. E.g. the key 8.7 will actually be stored under 8.
  - Bools are cast to integers, too, i.e. the key true will actually be stored under 1 and the key false under 0.
  - Null will be cast to the empty string, i.e. the key null will actually be stored under "".
  - Arrays and objects can not be used as keys. Doing so will result in a warning: Illegal offset type.

Array Type Casting and Overwriting

```
<?php
$array = array(
1     => "a",
"1"   => "b",
1.5   => "c",
true  => "d",);
var_dump($array);
?>
```

```
array(1) {
[1]=>
string(1) "d"
}
```

Array Type Casting and Overwriting

- PHP arrays can contain integer and string keys at the same time as PHP does not distinguish between indexed and associative arrays.

```
<?php
$array = array(
"foo" => "bar",
"bar" => "foo",
100   => -100,
-100  => 100,);
var_dump($array);
?>
```

```
array(4) {
["foo"]=> string(3)
"bar" ["bar"]=>
string(3) "foo"
[100]=> int(-100)
[-100]=> int(100) }
```

Looping Through an Array

```
<?php
$stuff = array("name" => "Liu",
               "course" => "TU856");
Foreach ($stuff as $k => $v ) {
  echo "Key=",$k," Val=",$v,"\n";
}
?>
```

```
Key=name Val=Liu
Key=course Val=TU856
```

Creating/modifying with square bracket syntax

- Assign values to the array, specifying the key in brackets. The key can also be omitted, resulting in an empty pair of brackets ([]).

```
$arr[key] = value;
$arr[] = value;
// key may be an integer or string
// value may be any value of any type
```

Creating/modifying with square bracket syntax

- To change a certain value, assign a new value to that element using its key. To remove a key/value pair, call the unset() function on it.

```php
<?php
$arr = array(5 => 1, 12 => 2);
$arr[] = 56;    // This is the same as $arr[13] = 56;
                // at this point of the script
$arr["x"] = 42; // This adds a new element to
                // the array with key "x"
unset($arr[5]); // This removes the element from the array
unset($arr);    // This deletes the whole array
?>
```

Creating/modifying with square bracket syntax

- An example:

```
<?php
// Create a simple array.
$array = array(1, 2, 3, 4, 5);
print_r($array);
// Now delete every item, but leave the array itself intact:
foreach ($array as $i => $value) {
    unset($array[$i]);
}
print_r($array);
// Append an item (note that the new key is 5, instead of 0).
$array[] = 6;
print_r($array);
// Re-index:
$array = array_values($array);
$array[] = 7;
print_r($array);
?>
```

Creating/modifying with square bracket syntax

- The result:

```
Array
(
        [0] => 1
        [1] => 2
        [2] => 3
        [3] => 4
        [4] => 5
)
Array
(
)
Array
(
        [5] => 6
)
Array
(
        [0] => 6
        [1] => 7
)
```

Arrays of Arrays

- The elements of an array can be many things other than a string or integer. You can even have objects or other arrays.

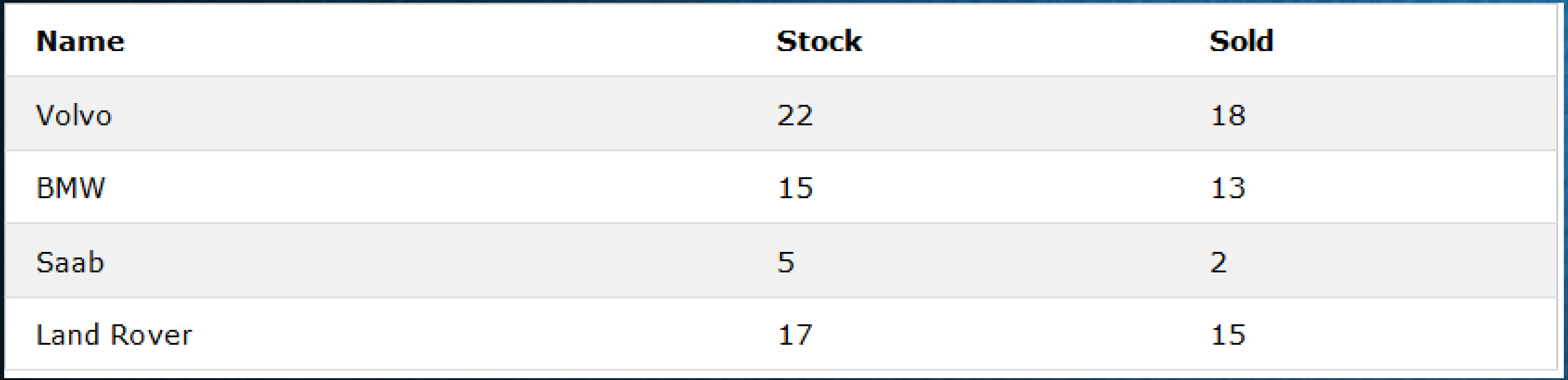
```
$products = array(
'paper' => array(
'copier' => "Copier & Multipurpose",
'inkjet' => "Inkjet Printer",
'laser' => "Laser Printer",
'photo' => "Photographic Paper"),
'pens' => array(
'ball' => "Ball Point",
'hilite' => "Highlighters",
'marker' => "Markers"),
'misc' => array(
'tape' => "Sticky Tape",
'glue' => "Adhesives",
'clips' => "Paperclips")
);
```

```
echo $products["paper"]["copier"];

Copier & Multipurpose
```

Two-dimensional Arrays

- A two-dimensional array is an array of arrays
- First, take a look at the following table:If there is a match, the code associated with that case is executed



Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

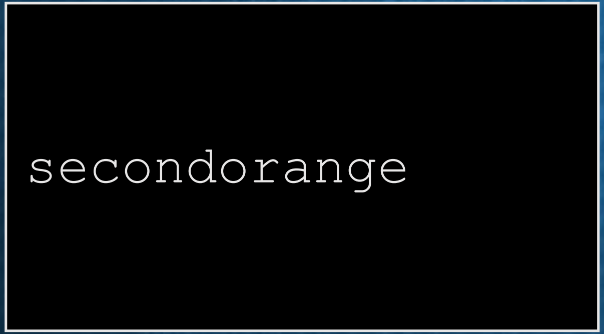
Two-dimensional Arrays

- We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array
(
array("Volvo",22,18),
array("BMW",15,13),
array("Saab",5,2),
array("Land Rover",17,15)
);
```

Two-dimensional Arrays

```
<?php
$fruits = array ( "fruits"  => array ( "a" => "orange",
                                      "b" => "banana",
                                      "c" => "apple"
                                    ),
                  "numbers" => array ( 1,
                                      2,
                                      3,
                                      4,
                                      5,
                                      6
                                    ),
                  "holes"   => array (     "first",
                                      5 => "second",
                                           "third"
                                    )
                );
```

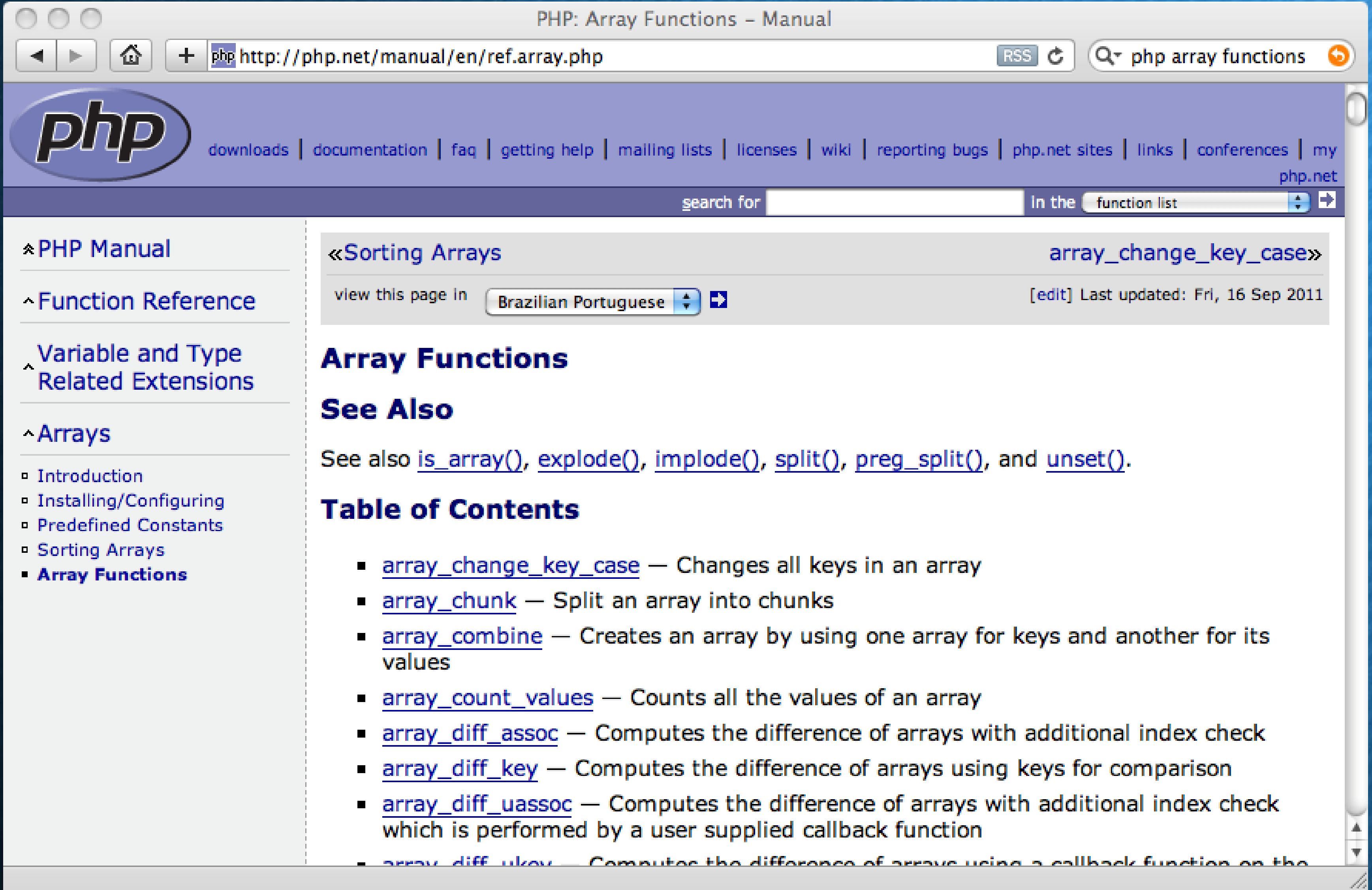


```
secondorange
```

```
// Some examples to address values in the array above
echo $fruits["holes"][5];    // prints "second"
echo $fruits["fruits"]["a"]; // prints "orange"
unset($fruits["holes"][0]);  // remove "first"

// Create a new multi-dimensional array
$juices["apple"]["green"] = "good";
```

Array Functions



PHP: Array Functions - Manual

http://php.net/manual/en/ref.array.php

php array functions
downloads | documentation | faq | getting help | mailing lists | licenses | wiki | reporting bugs | php.net sites | links | conferences | my php.net
search for
in the function list

«Sorting Arrays   array_change_key_case»

view this page in Brazilian Portuguese   [edit] Last updated: Fri, 16 Sep 2011

Array Functions

See Also

See also is_array(), explode(), implode(), split(), preg_split(), and unset().

Table of Contents

Array Functions

- count($ar) - How many elements in an array
- is_array($ar) - Returns TRUE if a variable is an array
- sort($ar) - Sorts the array values (loses key)
- ksort($ar) - Sorts the array by key
- asort($ar) - Sorts array by value, keeping key association
- shuffle($ar) - Shuffles the array into random order

Array Functions

```
$za = array();
$za["name"] = "Liu";
$za["course"] = "TU856";
print "Count: " count($za)"\n";
if ( is_array($za) ) {
      echo '$za Is an array' . "\n";
      } else {
      echo '$za Is not an array' . "\n";}
$zb = "123";
echo is_array($zb) ? '$zb Is an array' : '$zb Is not an
array';
echo "\n";
```

```
Count: 2
$za Is an array
$zb Is not an array
```

Array Functions

```
$za = array();
$za["name"] = “Liu";
$za["course"] = “TU856";
$za["topic"] = "PHP";
print_r($za);
sort($za);
print_r($za);
```

```
Array
(
[name] => Liu
[course] => TU856
[topic] => PHP
)
Array
(
[0] => Liu
[1] => PHP
[2] => TU856
)
```

Arrays and Strings

```
$inp = "This is a sentence with seven words";
$temp = explode(' ', $inp);
print_r($temp);
```

```
Array
(
[0] => This
[1] => is
[2] => a
[3] => sentence
[4] => with
[5] => seven
[6] => words
)
```

Summery

- PHP arrays are a very powerful associative array as they can be indexed by integers like a list, or use keys to look values up like a hash map or dictionary
- There are many options for sorting
- We can use explode() to split a string into an array of strings

Miscellaneous Useful Stuff

- String formatting
- Date Functions
- File Handling

String Formatting

- Most languages inspired by C have a feature similar to C's printf() function that gives a high level of control over formatted output when variables are converted to strings

```
$x = 1.0 / 3.0;
echo "x = $x\n";
printf ("x = %5.2f\n",$x);
```

```
x = 0.333333333333
x = 0.33
```

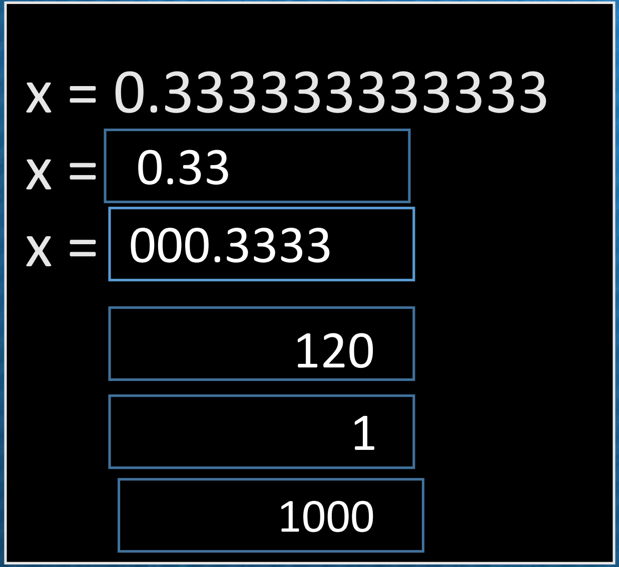
String Formatting

```
$x = 1.0 / 3.0;
echo "x = $x\n";
printf ("x = %5.2f\n",$x);
printf ("x = %08.4f\n",$x);
$y = 120;
$z = 1;
$a = 1000;
printf("%8d\n",$y);
printf("%8d\n",$z);
printf("%8d\n",$a);
```

```
x = 0.333333333333
x = 0.33
x = 000.3333
120
1
1000
```

String Formatting

```
$x = 1.0 / 3.0;
echo "x = $x\n";
printf ("x = %5.2f\n",$x);
printf ("x = %08.4f\n",$x);
$y = 120;
$z = 1;
$a = 1000;
printf("%8d\n",$y);
printf("%8d\n",$z);
printf("%8d\n",$a);
```



```
x = 0.333333333333
x =  0.33
x = 000.3333
         120
           1
        1000
```

String Formatting

```
printf  ("x = % 5.2f \n", $x);
```

```
X = 0.33
```

String Formatting

Five Character wide

Floating point number

```
printf  ("x = % 5.2f \n", $x);
```

Begin format code

Two digits after the decimal place

X = 0.33

Five Character wide

String Formatting

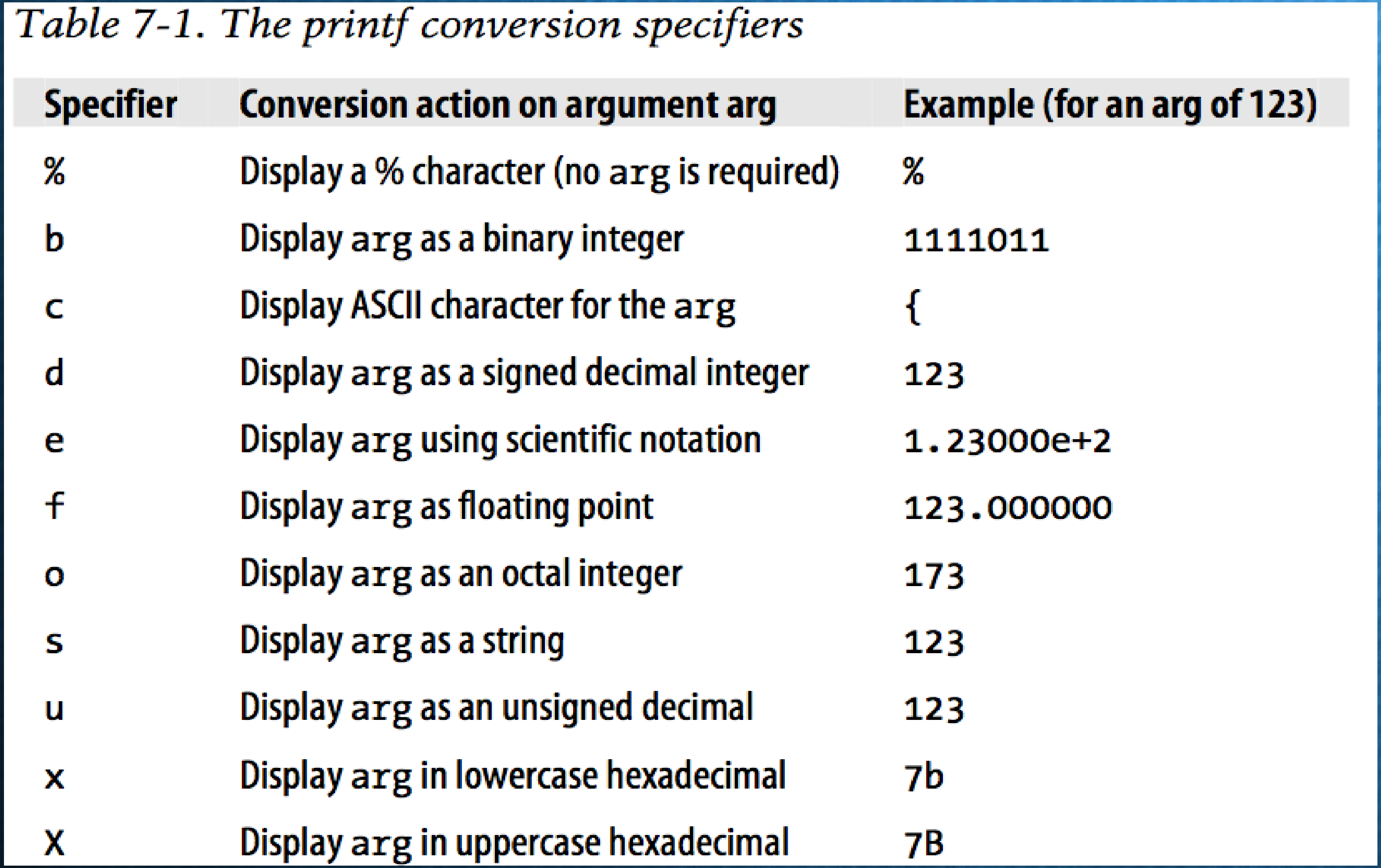


Table 7-1. The printf conversion specifiers

Specifier	Conversion action on argument arg	Example (for an arg of 123)
%	Display a % character (no arg is required)	%
b	Display arg as a binary integer	1111011
c	Display ASCII character for the arg	{
d	Display arg as a signed decimal integer	123
e	Display arg using scientific notation	1.23000e+2
f	Display arg as floating point	123.000000
o	Display arg as an octal integer	173
s	Display arg as a string	123
u	Display arg as an unsigned decimal	123
x	Display arg in lowercase hexadecimal	7b
X	Display arg in uppercase hexadecimal	7B

Multiple Format Codes

- The string can have multiple format codes and needs one argument after the format string for each of the codes

```c
printf( "My name is %s. I'm %d years old, which is
%X in hexadecimal\n", 'Simon', 33, 33 );
```

```
My name is Simon. I'm 33 years old, which is 21 in hexadecimal
```

Formatted Print to a String

- Often we want to format a string printf() style but instead, have the formatted result in a variable to put in a database field or send across a networks, etc.

```
$hexstring = printf("%X%X%X", 65, 127, 245);
echo "Hex = " . $hexstring . "\n";
```

```
Hex = 417FF5
```

Date and Time

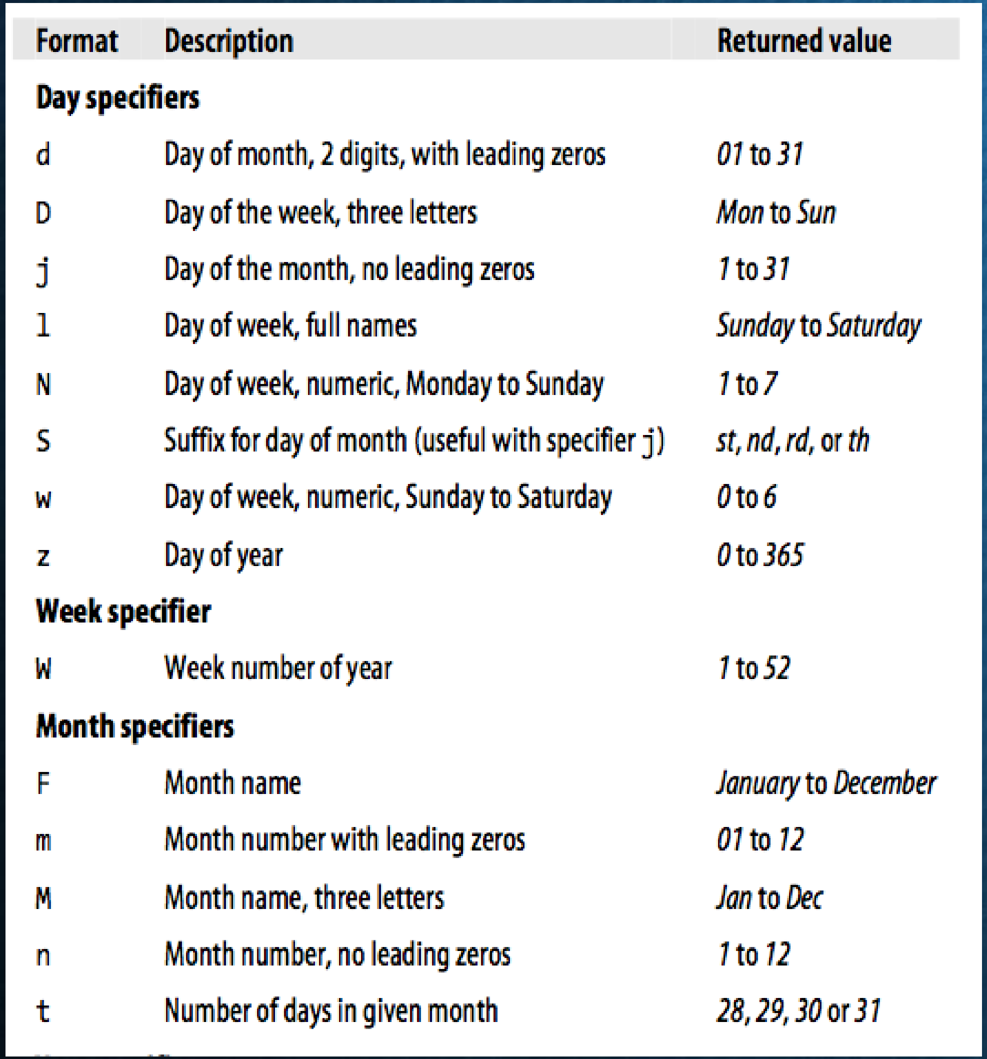
- Time is an integer number of seconds since January 1, 1970
- Can do relative computations by adding a number of seconds
- There might be a problem around 2038.....
- The date() function is used to produce various string-formatted representations of the date

Date and Time

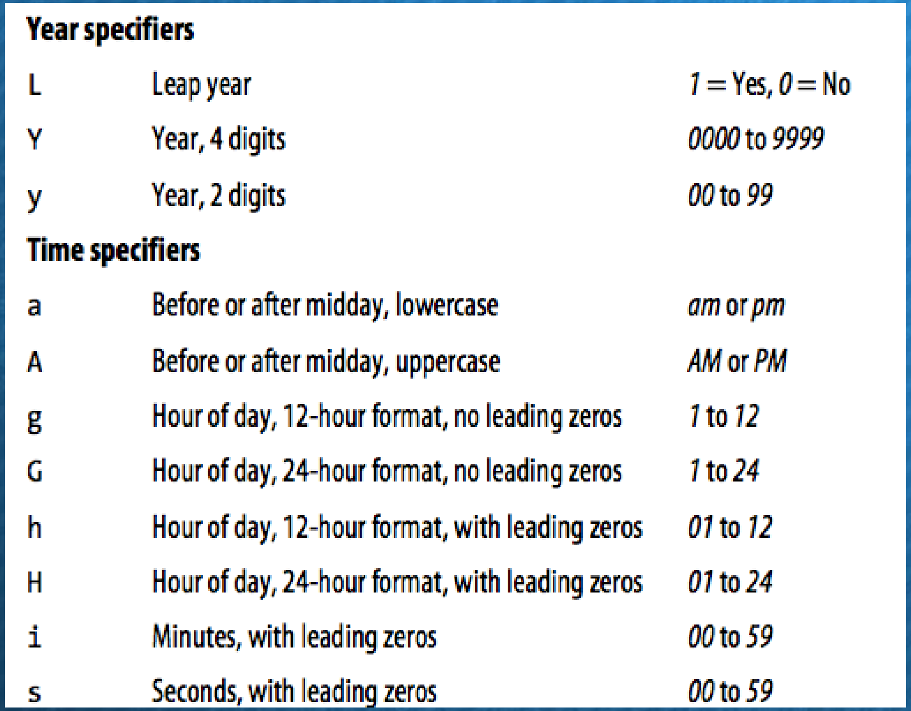
```
echo "Time = " . time() . "\n";
$nextWeek = time() + (7 * 24 * 60 * 60);
// 7 days; 24 hours; 60 mins; 60secs
echo 'Now: '. date('Y-m-d') ."\n";
echo 'Next Week: '. date('Y-m-d', $nextWeek) ."\n";
```

```
Time = 1635974620
Now: 2021-11-03
Next Week: 2021-11-10
```

Date and Time



Format	Description	Returned value
Day specifiers		
d	Day of month, 2 digits, with leading zeros	*01* to *31*
D	Day of the week, three letters	*Mon* to *Sun*
j	Day of the month, no leading zeros	*1* to *31*
l	Day of week, full names	*Sunday* to *Saturday*
N	Day of week, numeric, Monday to Sunday	*1* to *7*
S	Suffix for day of month (useful with specifier j)	*st*, *nd*, *rd*, or *th*
w	Day of week, numeric, Sunday to Saturday	*0* to *6*
z	Day of year	*0* to *365*
Week specifier		
W	Week number of year	*1* to *52*
Month specifiers		
F	Month name	*January* to *December*
m	Month number with leading zeros	*01* to *12*
M	Month name, three letters	*Jan* to *Dec*
n	Month number, no leading zeros	*1* to *12*
t	Number of days in given month	*28*, *29*, *30* or *31*



Year specifiers		
L	Leap year	*1* = Yes, *0* = No
Y	Year, 4 digits	*0000* to *9999*
y	Year, 2 digits	*00* to *99*
Time specifiers		
a	Before or after midday, lowercase	*am* or *pm*
A	Before or after midday, uppercase	*AM* or *PM*
g	Hour of day, 12-hour format, no leading zeros	*1* to *12*
G	Hour of day, 24-hour format, no leading zeros	*1* to *24*
h	Hour of day, 12-hour format, with leading zeros	*01* to *12*
H	Hour of day, 24-hour format, with leading zeros	*01* to *24*
i	Minutes, with leading zeros	*00* to *59*
s	Seconds, with leading zeros	*00* to *59*

Date Formats

- Different Web protocols need different date formats
- ISO8601 is a popular format becuase it is simple and in UTC / GMT

```
echo "ISO 8601 = " . gmDate("Y-m-d\TH:i:s\Z") . "\n";
```

```
ISO 8601 = 2021-10-28T10:55:25Z
```

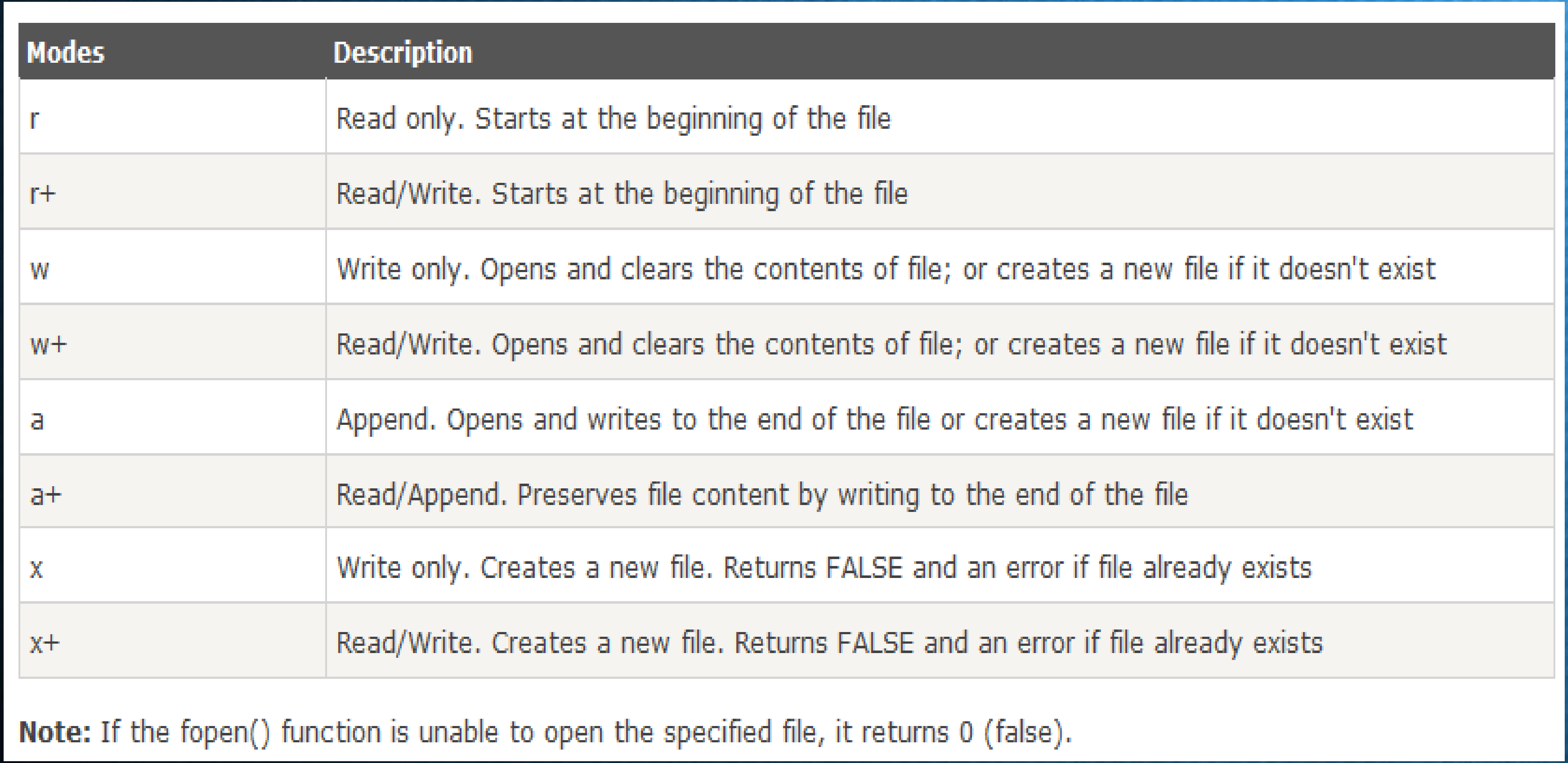
Reading and Writing Files

- Checking for Existence

```
if (file_exists("names.txt")) echo "names.txt
exists\n";
```

```
names.txt exists
```

Reading and Writing Files



Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

Note: If the fopen() function is unable to open the specified file, it returns 0 (false).

Reading and Writing Files

```
$file = fopen("names.txt", "r") or exit("Unable to
open file!");
//Output a line of the file until the end is reached
while(!feof($file))
  {
  echo fgets($file). "<br>";
  }
fclose($file);
```

```
# First, last, email, age
Granny,Smith,gsmith@dit.ie,29
Mariela,Bischoff,mb@dit.ie,29
Harry,Spitz,hs@dit.ie,29
Roni Callaghan,rc@dit.ie,29
Latanya,Hosmer,lh@dit.ie,29
Tyson,Bortz,tb@dit.ie,29
Charity,Sato,cs@dit.ie,29
Jaymie,Valencia,jv@dit.ie,29
Una,Mcalister,um@dit.ie,29
Adella,Gries,ag@dit.ie,29
Cathleen,Mclaughlin,cm@dit.ie,29
```

Reading a File Character by Character

```
$file=fopen("names.txt","r") or exit("Unable to
open file!");
while (!feof($file))
  {
  echo fgetc($file);
  }
fclose($file);
```

Creates a New File

- Opens and clears the contents of file; or creates a new file if it doesn't exist

```
<html>
<body>

<?php
$file=fopen("welcome.txt","w");
?>

</body>
</html>
```

Write to a File

- To insert text without over-writing the beginning of the file, you'll have to open it for appending (a+ rather than r+)

```
$file=fopen(“welcome.txt","a+") or exit("Unable to
open file!");

if ($_POST["lastname"] <> "")
{
    fwrite($file,$_POST["lastname"]."\n");
}

fclose($file);
```