DT228/2 Web Development

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" => “DT228");
  echo $stuff["course"] , "\n";
?>
```

```
DT228
```

Dumping an Array

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

```
<?php
  $stuff = array("name" => “Liu",
                 "course" => “DT228");
  print_r($stuff);
?>
```

```
Array
(
[name] => Liu
[course] => DT228
)
```

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";
$za["course"] = "DT228";
print_r($za);
```

```
Array
(
[name] => Liu
[course] => DT228
)
```

Looping Through an Array

```
<?php
  $stuff = array("name" => “Liu",
                 "course" => “DT228");
  Foreach ($stuff as $k => $v ) {
    echo "Key=",$k," Val=",$v,"\n";
  }
?>
```

```
Key=name Val=Liu
Key=course Val=DT228
```

Arrays of Arrays

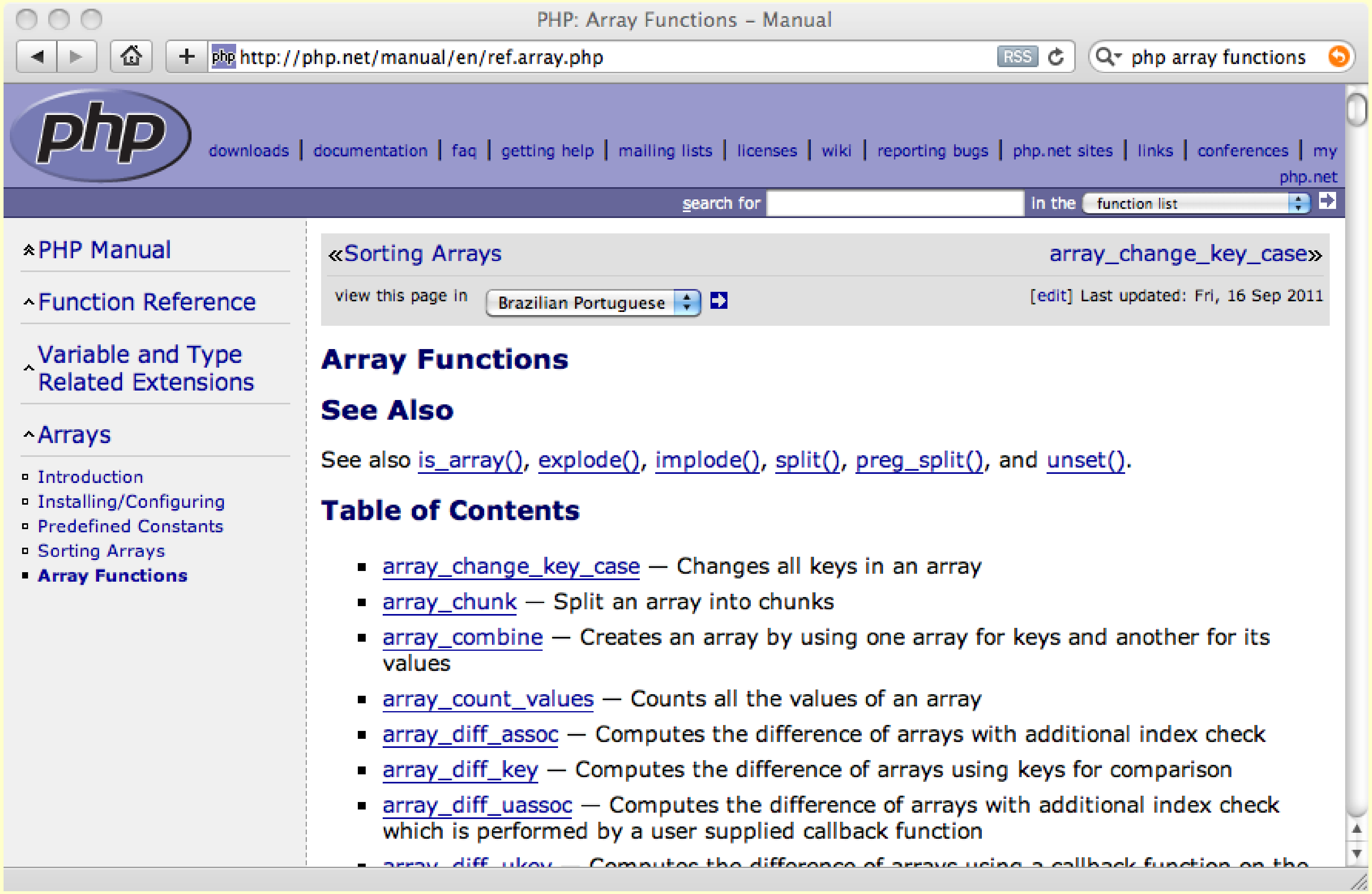
The elements of an array can be many things other than a string or integer. You can even have objects or other arrays.

```
$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
```

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

PHP Manual
Function Reference
Variable and Type Related Extensions
Arrays

- Introduction
- Installing/Configuring
- Predefined Constants
- Sorting Arrays
- **Array Functions**

«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_change_key_case — Changes all keys in an array
- array_chunk — Split an array into chunks
- array_combine — Creates an array by using one array for keys and another for its values
- array_count_values — Counts all the values of an array
- array_diff_assoc — Computes the difference of arrays with additional index check
- array_diff_key — Computes the difference of arrays using keys for comparison
- array_diff_uassoc — Computes the difference of arrays with additional index check which is performed by a user supplied callback function
- array_diff_ukey — Computes the difference of arrays using a callback function on the

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";
$za["course"] = “DT228";
print "Count: " count($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"] = “DT228";
$za["topic"] = "PHP";
print_r($za);
sort($za);
print_r($za);
```

```
Array
(
[name] => Liu
[course] => DT228
[topic] => PHP
)
Array
(
[0] => DT228
[1] => Liu
[2] => PHP
)
```

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

```
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 = b0.33

Five Character wide

String Formatting

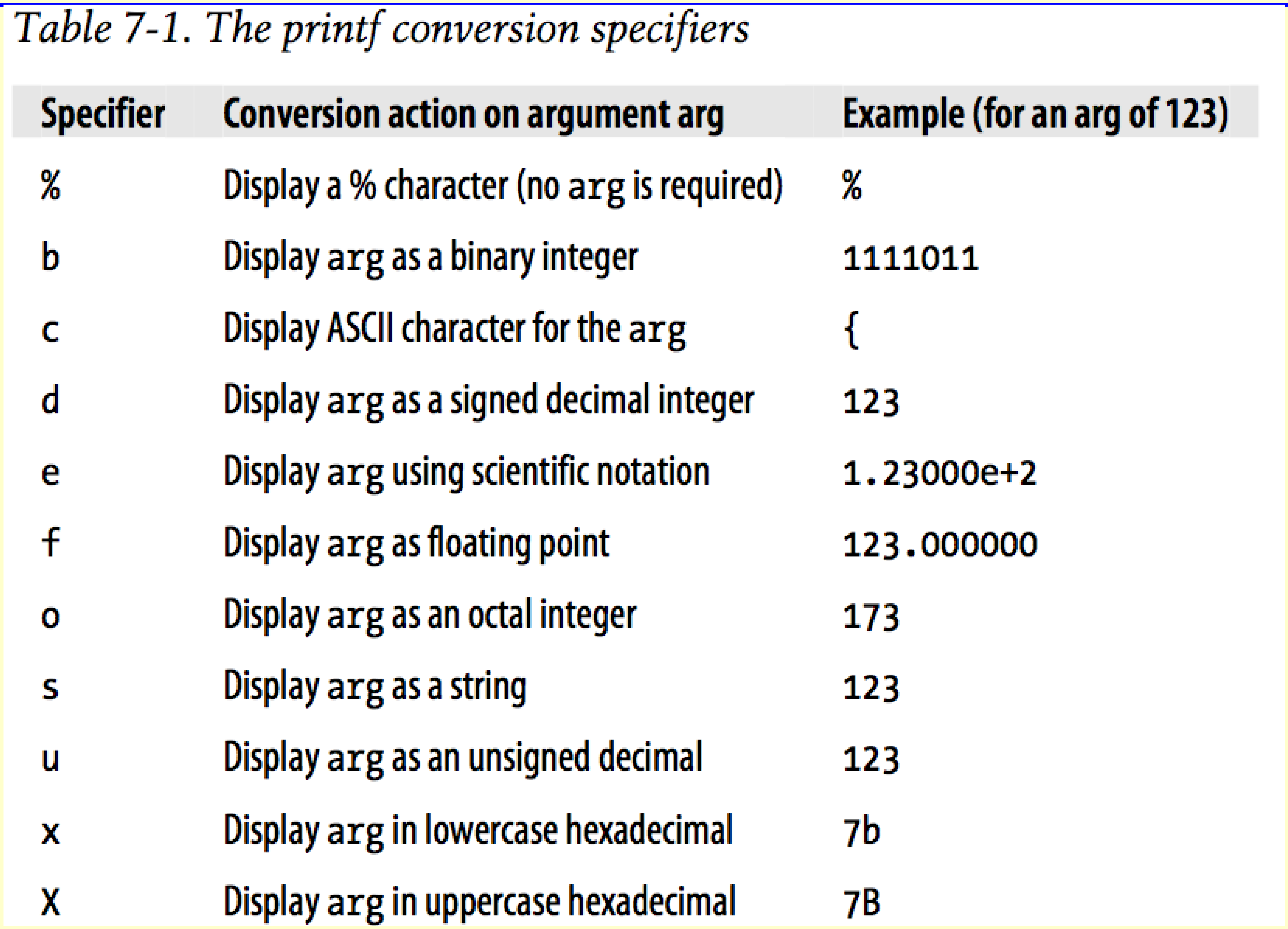


Table 7-1. The printf conversion specifiers

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

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

```
Hex = 417FF5
```

Date and Time

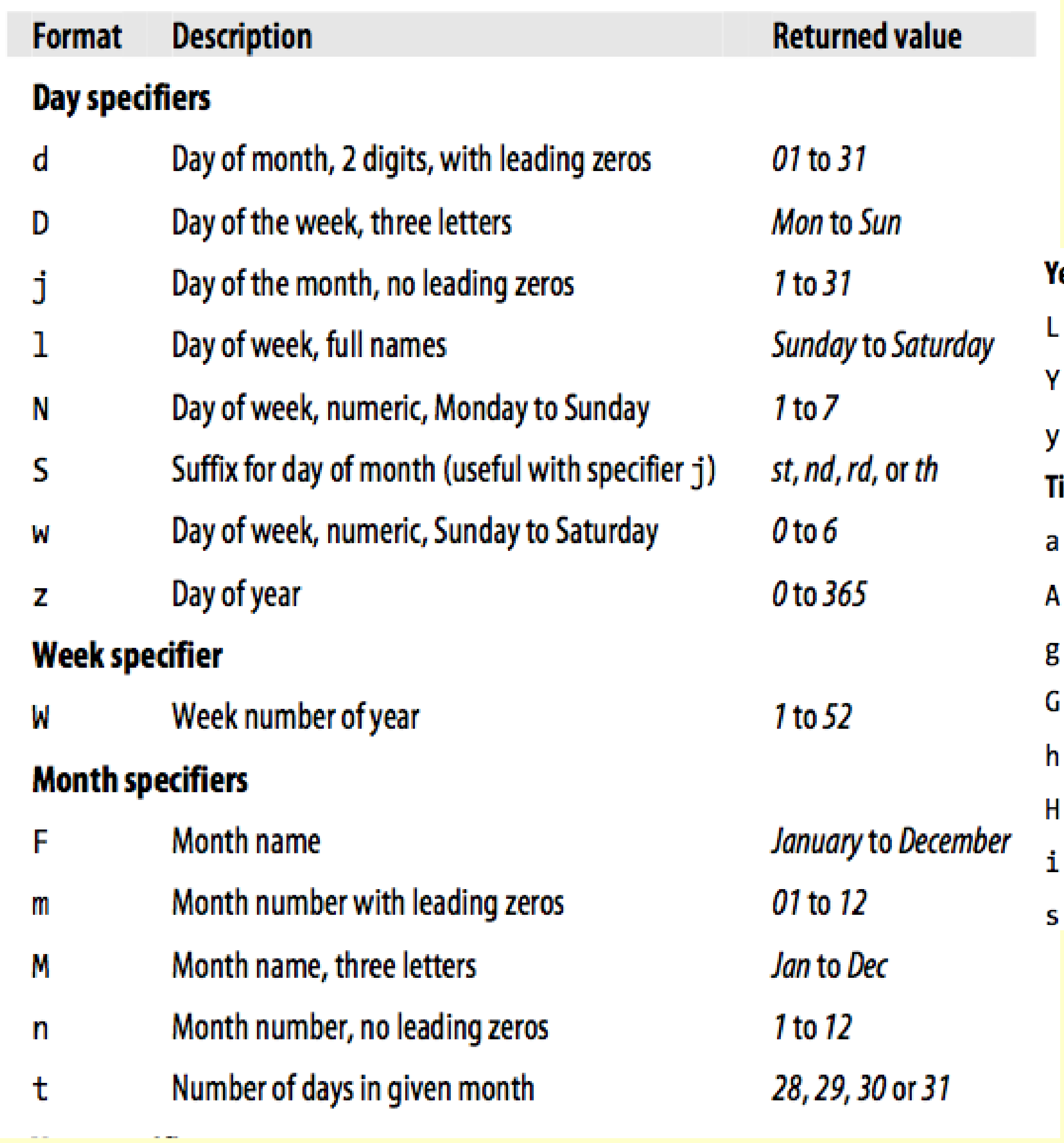
- Time is an integer number of seconds since January 1, 1970
    - Can do relative computations by adding a number of seconds
    - There might be a problem around 2038.....
- The date() function is used to produce various string-formatted representations of the date

Date and Time

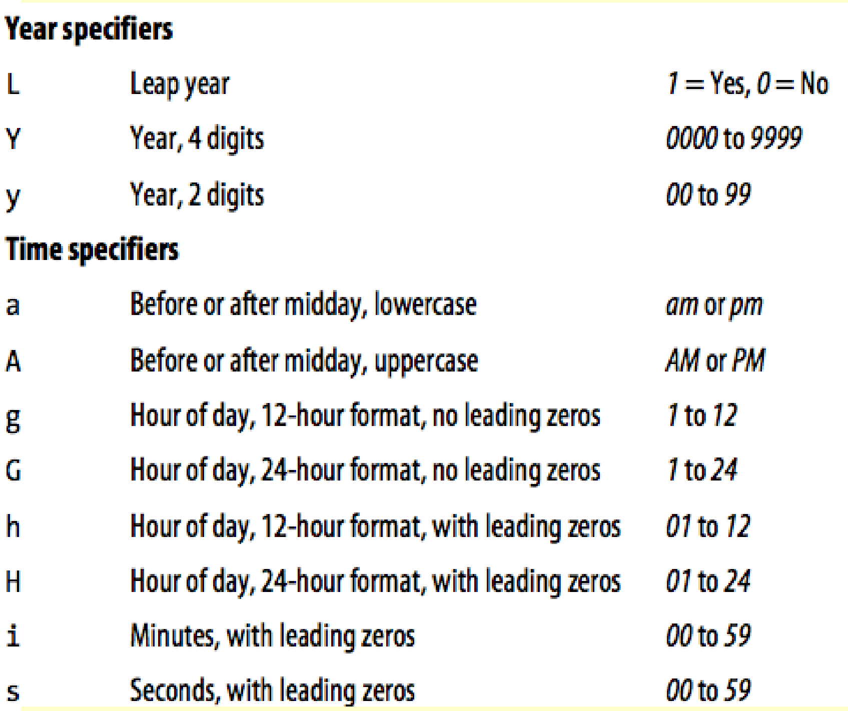
```
echo "Time = " . time() . "\n";
$nextWeek = time() + (7 * 24 * 60 * 60);
// 7 days; 24 hours; 60 mins; 60secs
echo 'Now: '. date('Y-m-d') ."\n";
echo 'Next Week: '. date('Y-m-d', $nextWeek) ."\n";
```

```
Time = 1382530436
Now: 2014-10-23
Next Week: 2014-10-30
```

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 = 2013-10-23T12:51:59Z
```

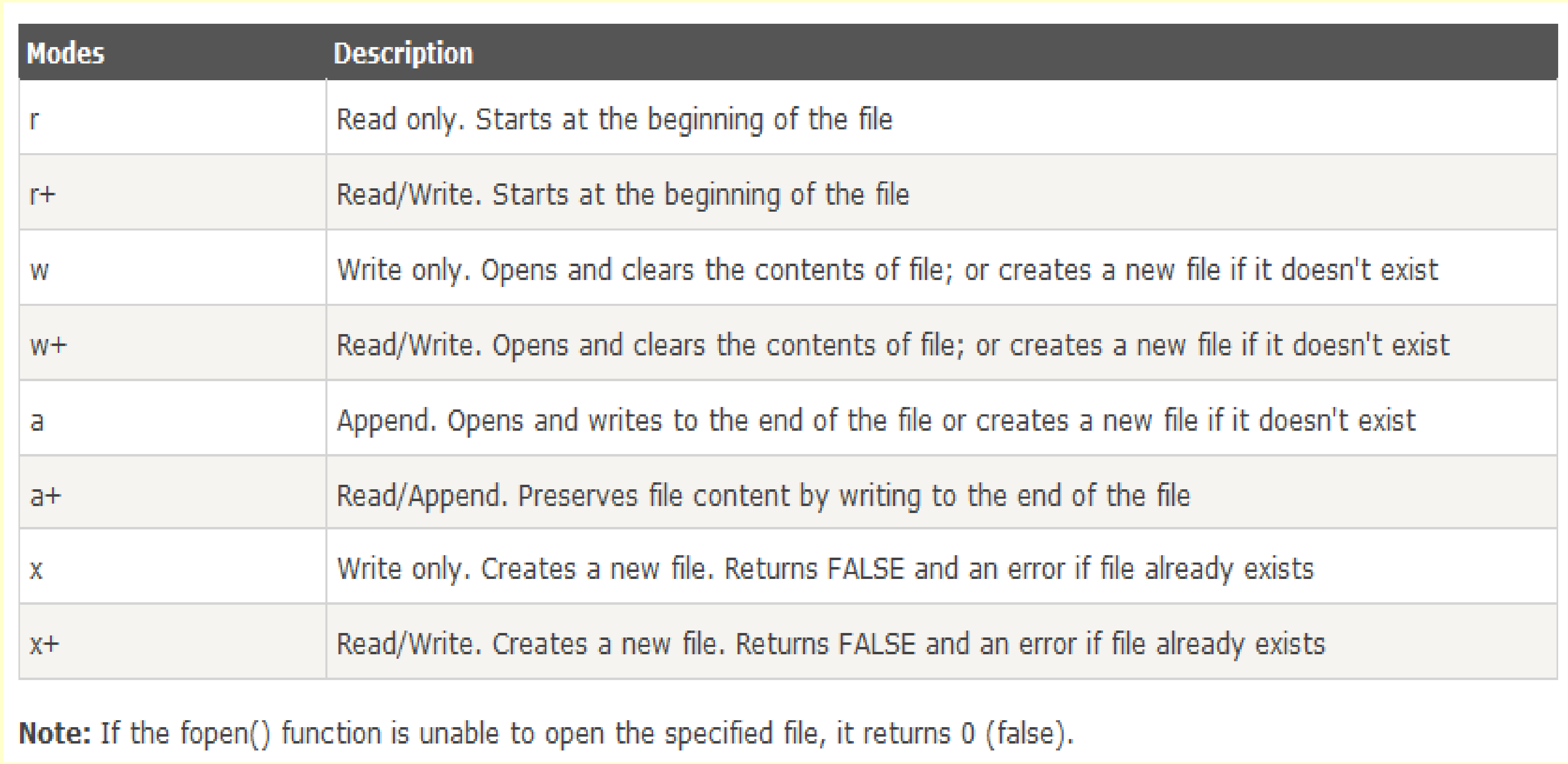
Reading and Writing Files

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 all the lines in a file..

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