PHP

PHP File Handling

Introduction [1]

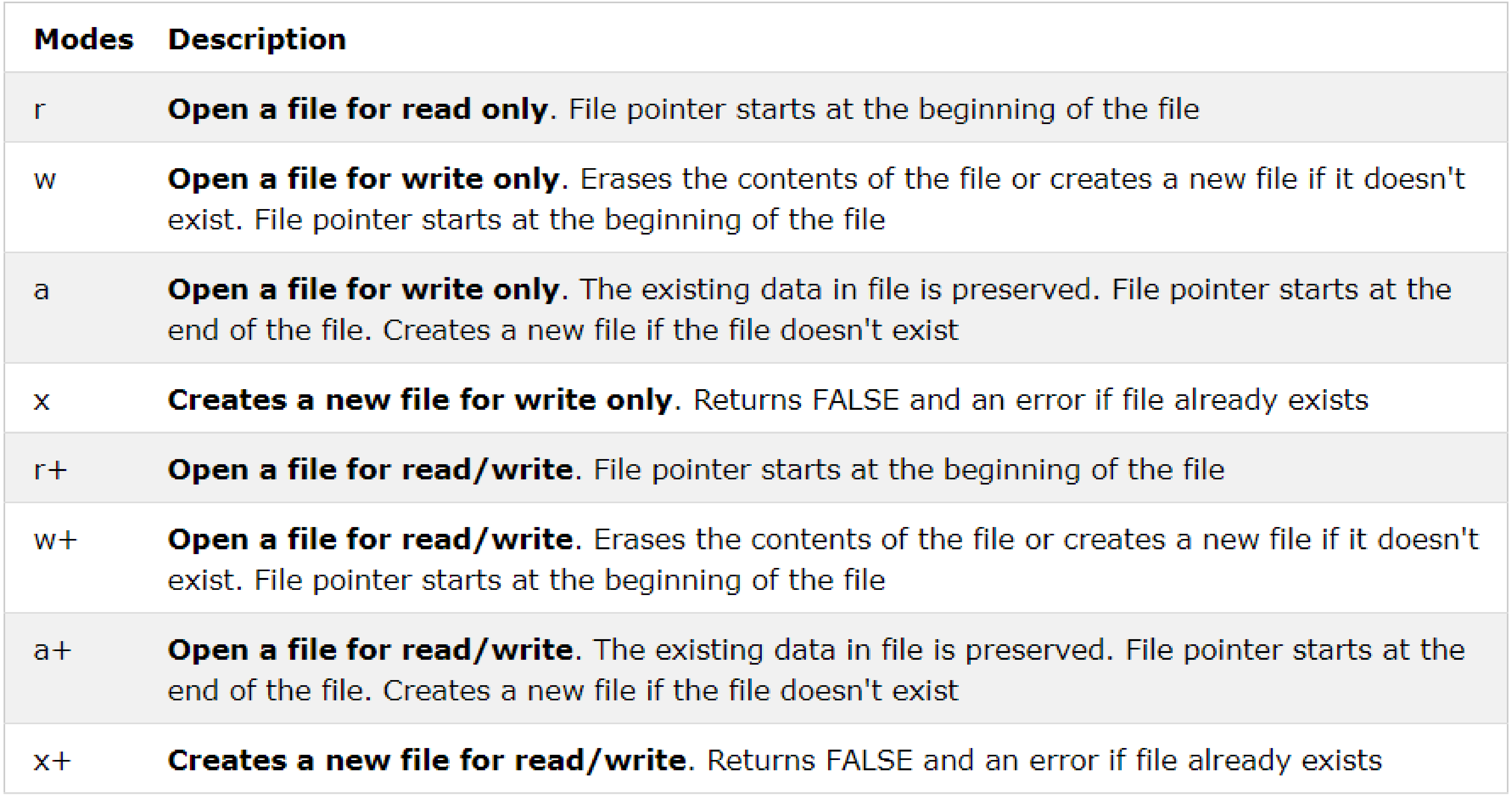
- File handling is an important part of any web application. You often need to open and process a file for different tasks.

- PHP Manipulating Files
  - PHP has several functions for creating, reading, uploading, and editing files.
  - **Be careful when manipulating files!**
    - When you are manipulating files you must be very careful. You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

File Operations [1] [2]

- Opening a file
- Reading a file
- Writing a file
- Closing a file

File modes [1][2]



Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

readfile() Function [1]

- The readfile() function reads a file and writes it to the output buffer.
- The readfile() function is useful if all you want to do is open up a file and read its contents.

readfile() Function [1]

The readfile() function reads a file and writes it to the output buffer.

This function returns the number of bytes read on success, or FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.

Syntax

```
readfile(filename,include_path,context)
```

Parameter	Description
filename	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

File Open/Read/Close

Open File - fopen()

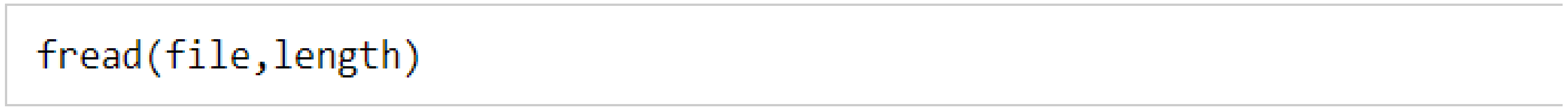
- A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.
- The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.
- The following example also generates a message if the fopen() function is unable to open the specified file:

```
fopen(filename,mode,include_path,context)
```

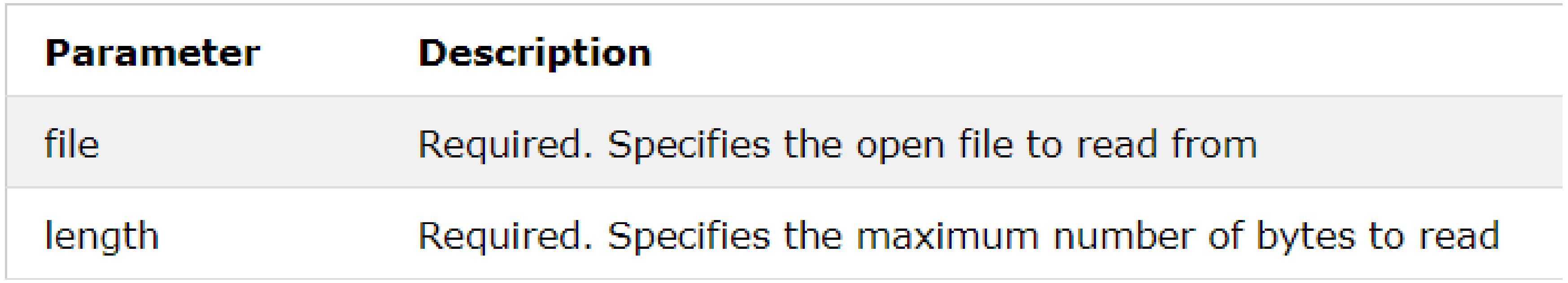
- Example 2

Read File - fread() [1]

- The fread() function reads from an open file.
- The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
- Syntax:



```
fread(file,length)
```



Parameter	Description
file	Required. Specifies the open file to read from
length	Required. Specifies the maximum number of bytes to read

Read File - fread() [1]

- The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

- Example 2

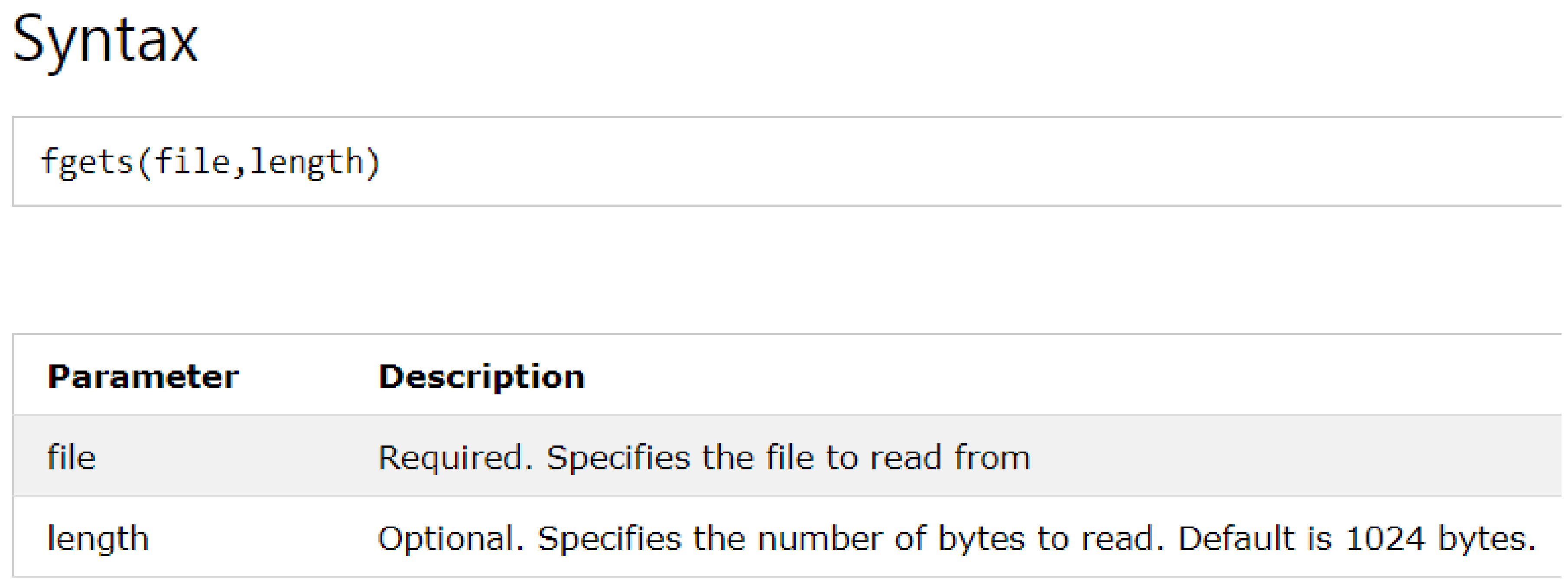
Close File - fclose() [1]

- The fclose() function is used to close an open file.
- It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!
- The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

- Example 2

Read Single Line - fgets() [1]

- The fgets() function returns a line from an open file.
- The fgets() function stops returning on a new line, at the specified length, or at EOF, whichever comes first.
- This function returns FALSE on failure.



Syntax

```
fgets(file,length)
```

Parameter	Description
file	Required. Specifies the file to read from
length	Optional. Specifies the number of bytes to read. Default is 1024 bytes.

- Example 3

Check End-Of-File - feof() [1]

- The feof() function checks if the "end-of-file" (EOF) has been reached.
- The feof() function is useful for looping through data of unknown length.

- Example 3

Read Single Character - fgetc() [1]

- The fgetc() function is used to read a single character from a file.

- **Note:** After a call to the fgetc() function, the file pointer moves to the next character.
- **Note:** This function is **slow** and should not be used on large files. If you need to read one character at a time from a large file, use fgets() to read data one line at a time and then process the line one character at a time with fgetc().

- Example 4

File Create/Write [1]

Create File - fopen():

- The `fopen()` function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
- If you use `fopen()` on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

File Create/Write [1]

Write to File - fwrite():

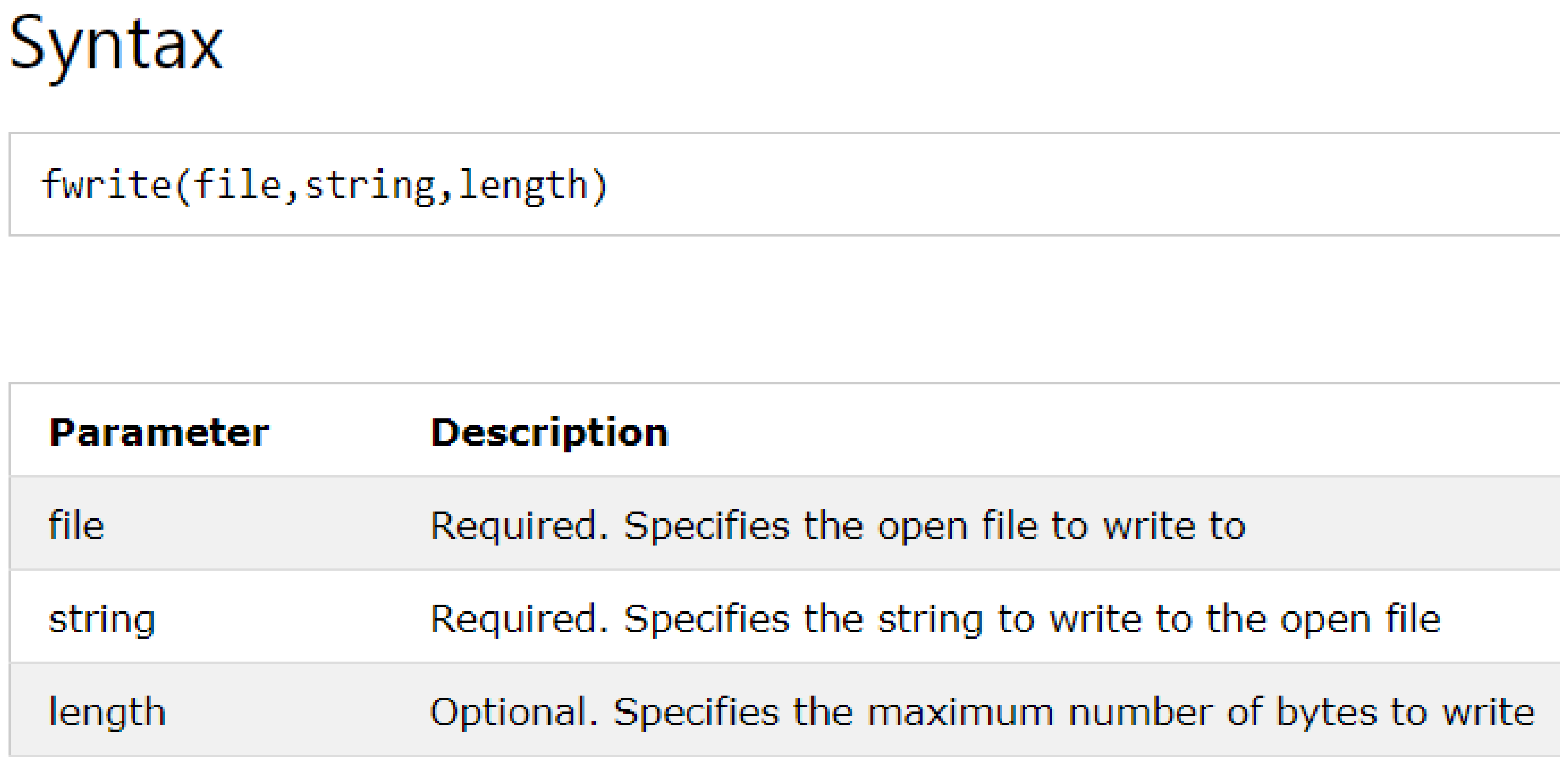
- The `fwrite()` function is used to write to a file.
- The first parameter of `fwrite()` contains the name of the file to write to and the second parameter is the string to be written.

- The function will stop at the end of the file or when it reaches the specified length, whichever comes first.

- This function returns the number of bytes written, or FALSE on failure.

File Create/Write [1]

Write to File - fwrite():

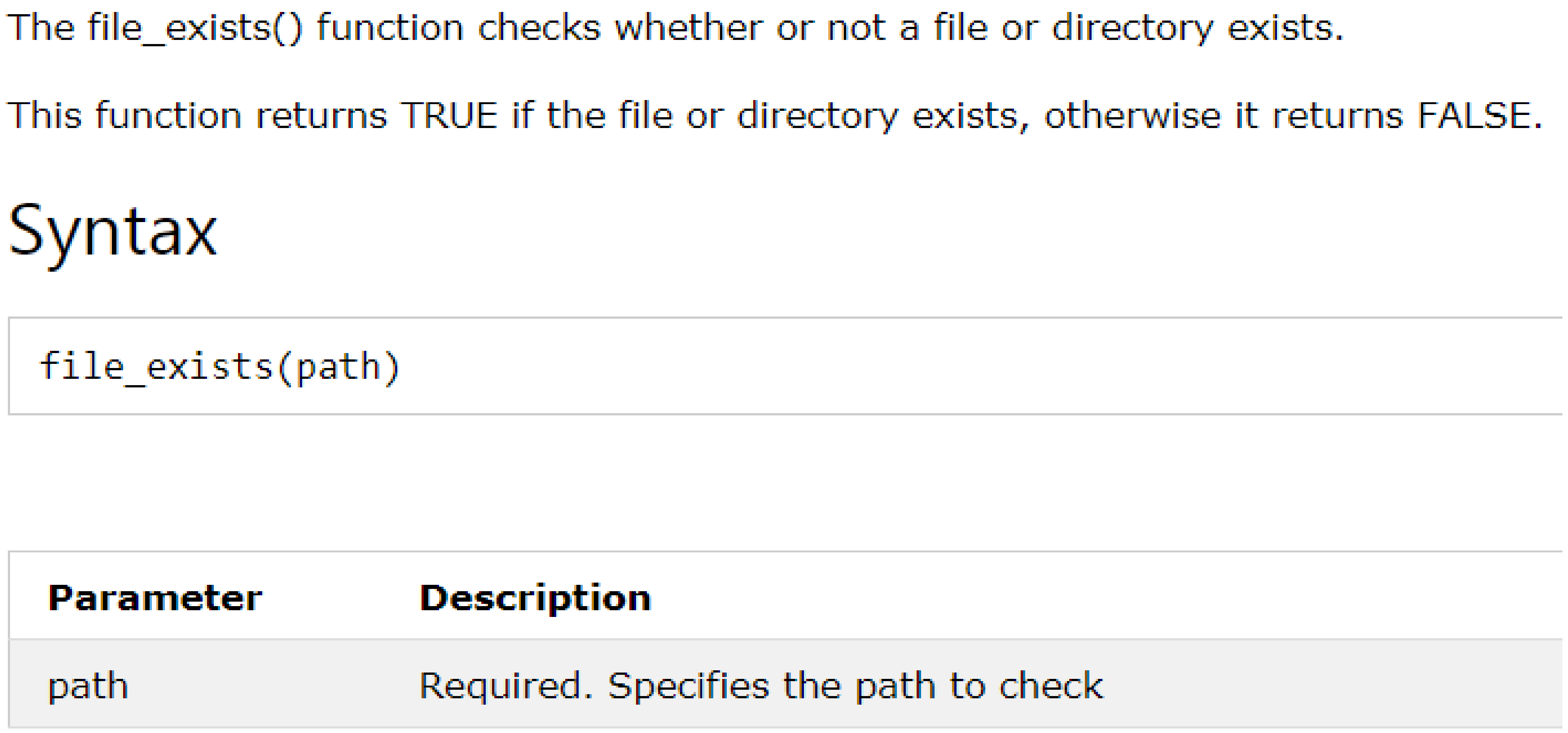


Syntax

```
fwrite(file,string,length)
```

Parameter	Description
file	Required. Specifies the open file to write to
string	Required. Specifies the string to write to the open file
length	Optional. Specifies the maximum number of bytes to write

file_exists() Function [1]



The file_exists() function checks whether or not a file or directory exists.

This function returns TRUE if the file or directory exists, otherwise it returns FALSE.

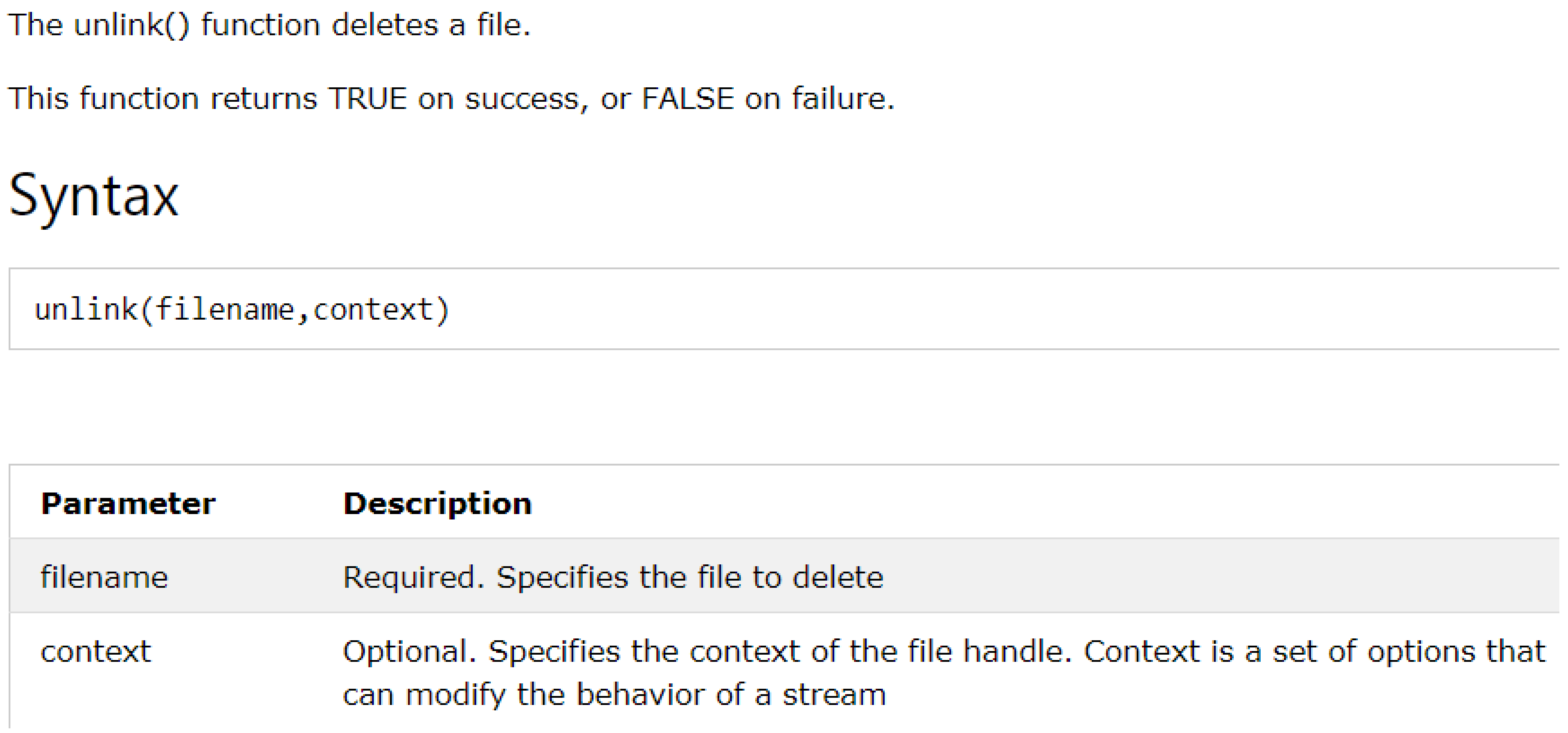
Syntax

```
file_exists(path)
```

Parameter	**Description**
path	Required. Specifies the path to check

- Example 6

unlink() Function [1]



The unlink() function deletes a file.

This function returns TRUE on success, or FALSE on failure.

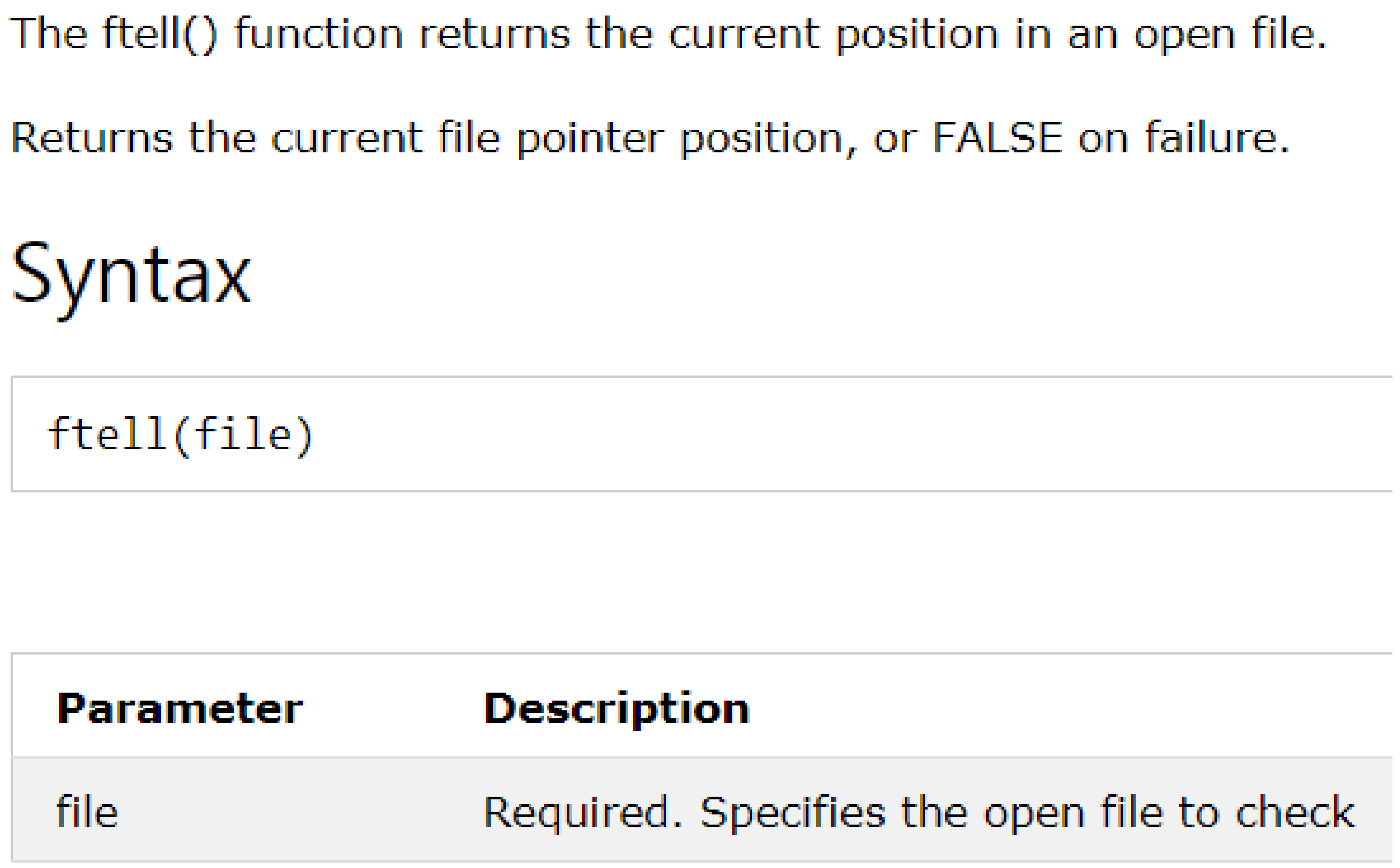
Syntax

```
unlink(filename,context)
```

Parameter	**Description**
filename	Required. Specifies the file to delete
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

- Example 7

ftell() Function [1]



The ftell() function returns the current position in an open file.

Returns the current file pointer position, or FALSE on failure.

Syntax

```
ftell(file)
```

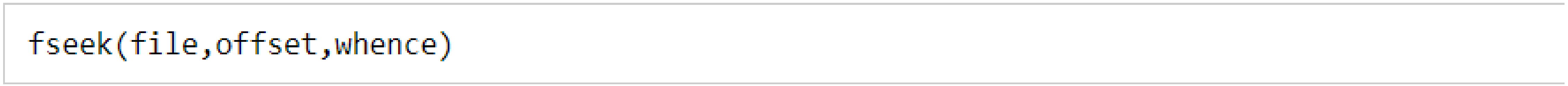
Parameter	Description
file	Required. Specifies the open file to check

- Example 8

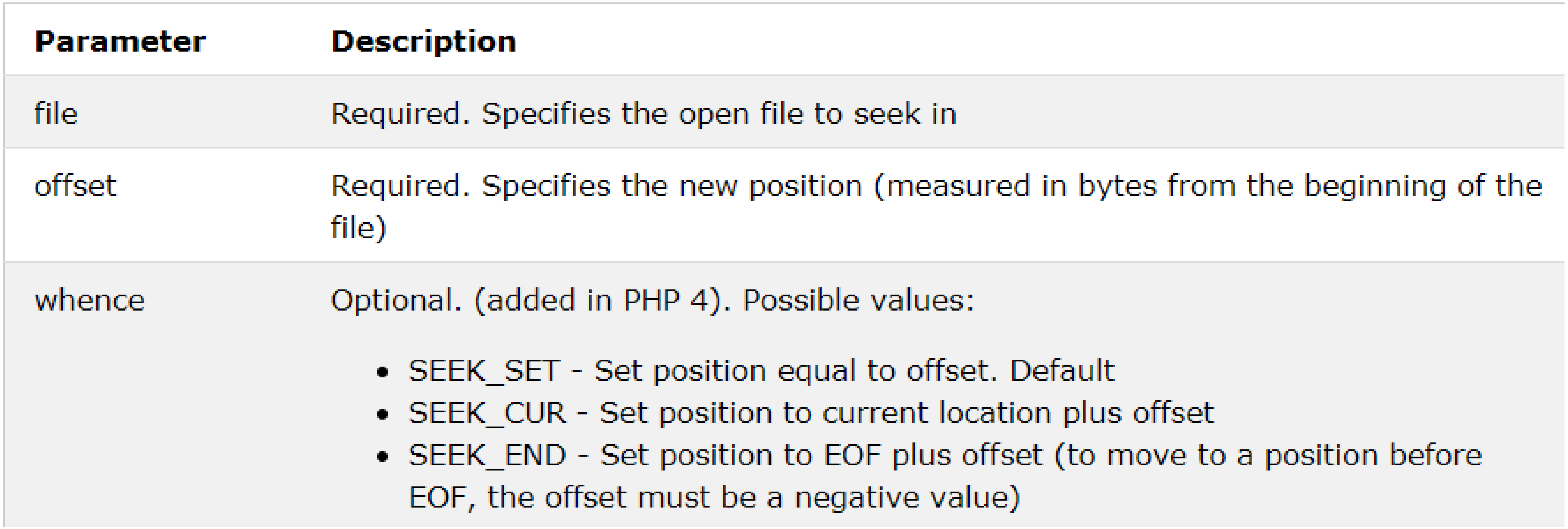
fseek() Function [1]

- The fseek() function seeks in an open file.
- This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.
- **This function returns 0 on success, or -1 on failure. Seeking past EOF will not generate an error.**
- Example 8

fseek() Function [1]

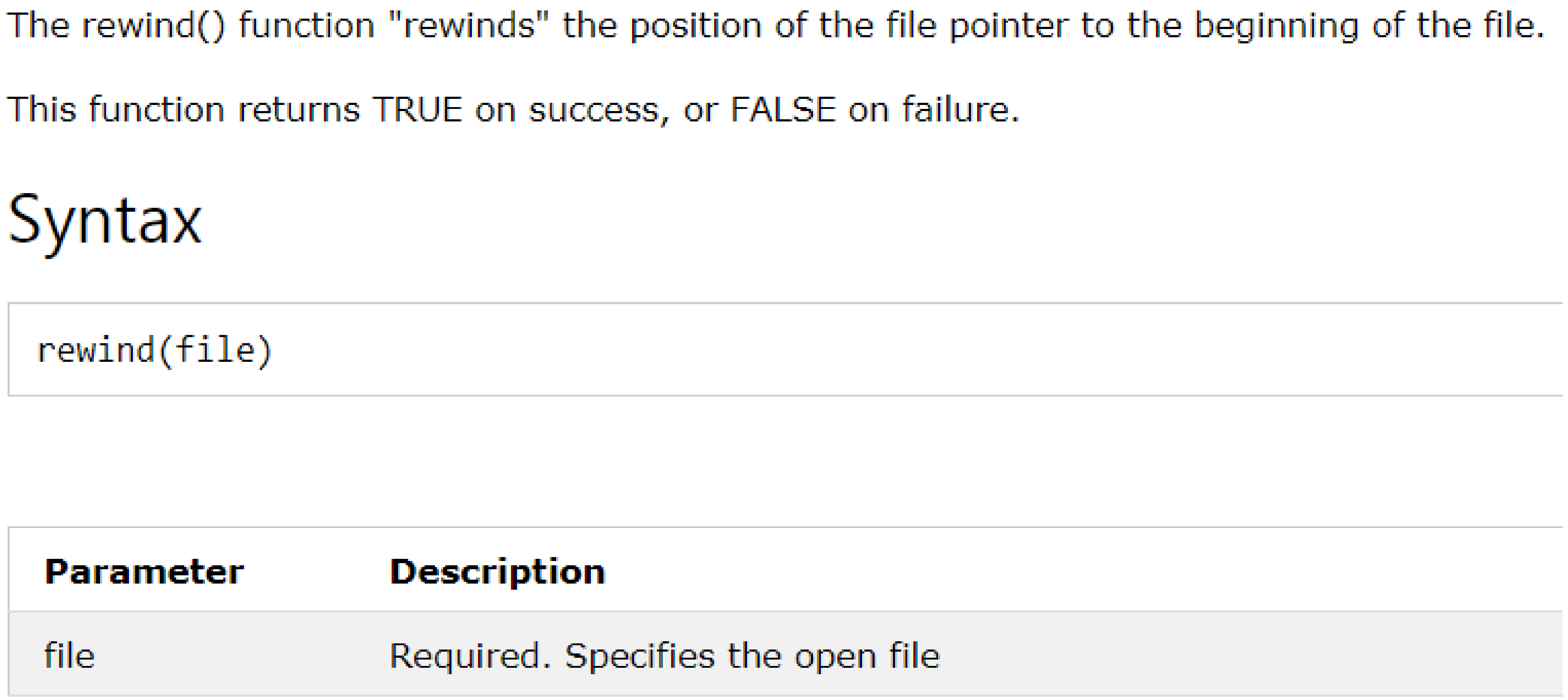


```
fseek(file,offset,whence)
```



Parameter	**Description**
file	Required. Specifies the open file to seek in
offset	Required. Specifies the new position (measured in bytes from the beginning of the file)
whence	Optional. (added in PHP 4). Possible values: • SEEK_SET - Set position equal to offset. Default • SEEK_CUR - Set position to current location plus offset • SEEK_END - Set position to EOF plus offset (to move to a position before EOF, the offset must be a negative value)

rewind() Function [1]



The rewind() function "rewinds" the position of the file pointer to the beginning of the file.

This function returns TRUE on success, or FALSE on failure.

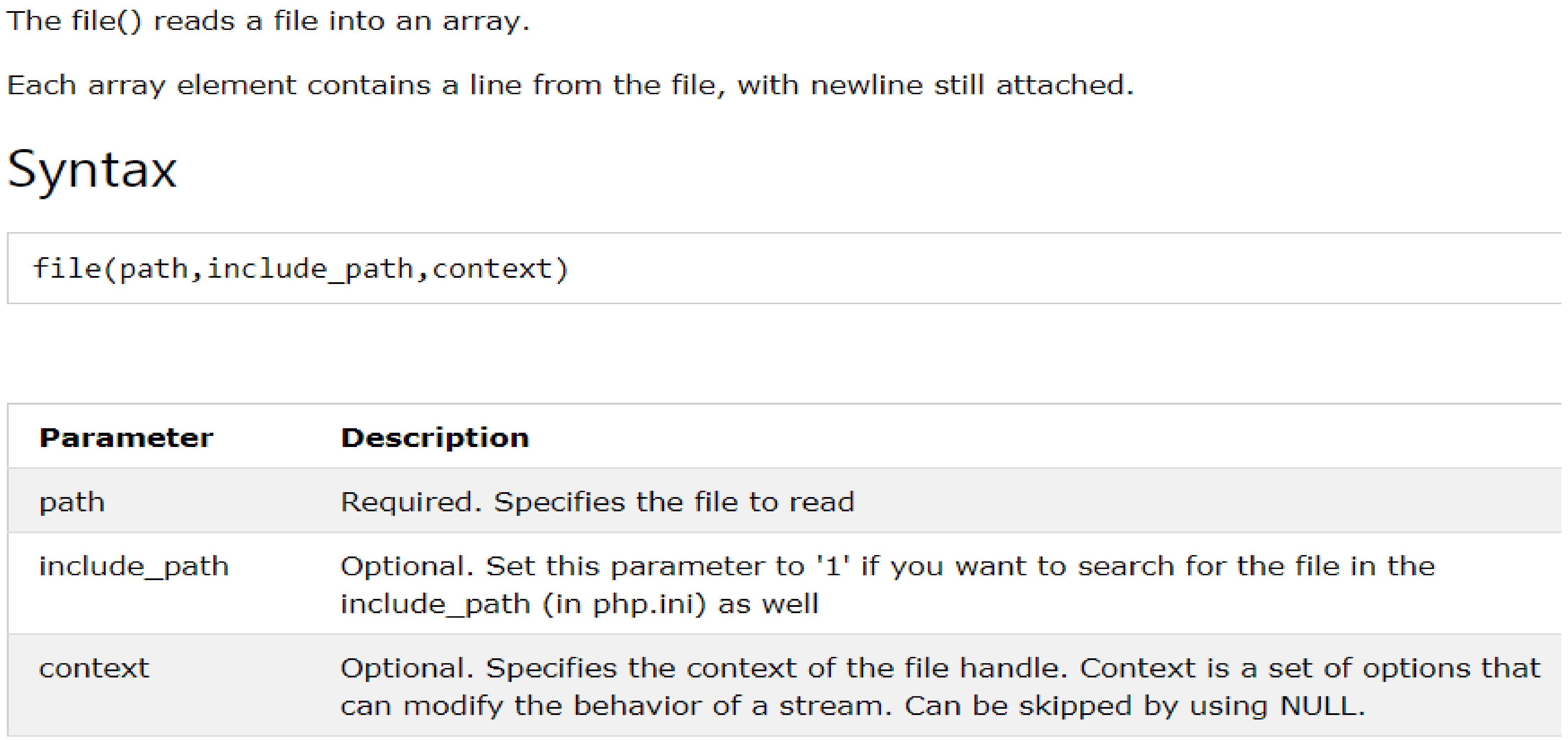
Syntax

```
rewind(file)
```

Parameter	Description
file	Required. Specifies the open file

- Example 8

file() Function [1]



The file() reads a file into an array.

Each array element contains a line from the file, with newline still attached.

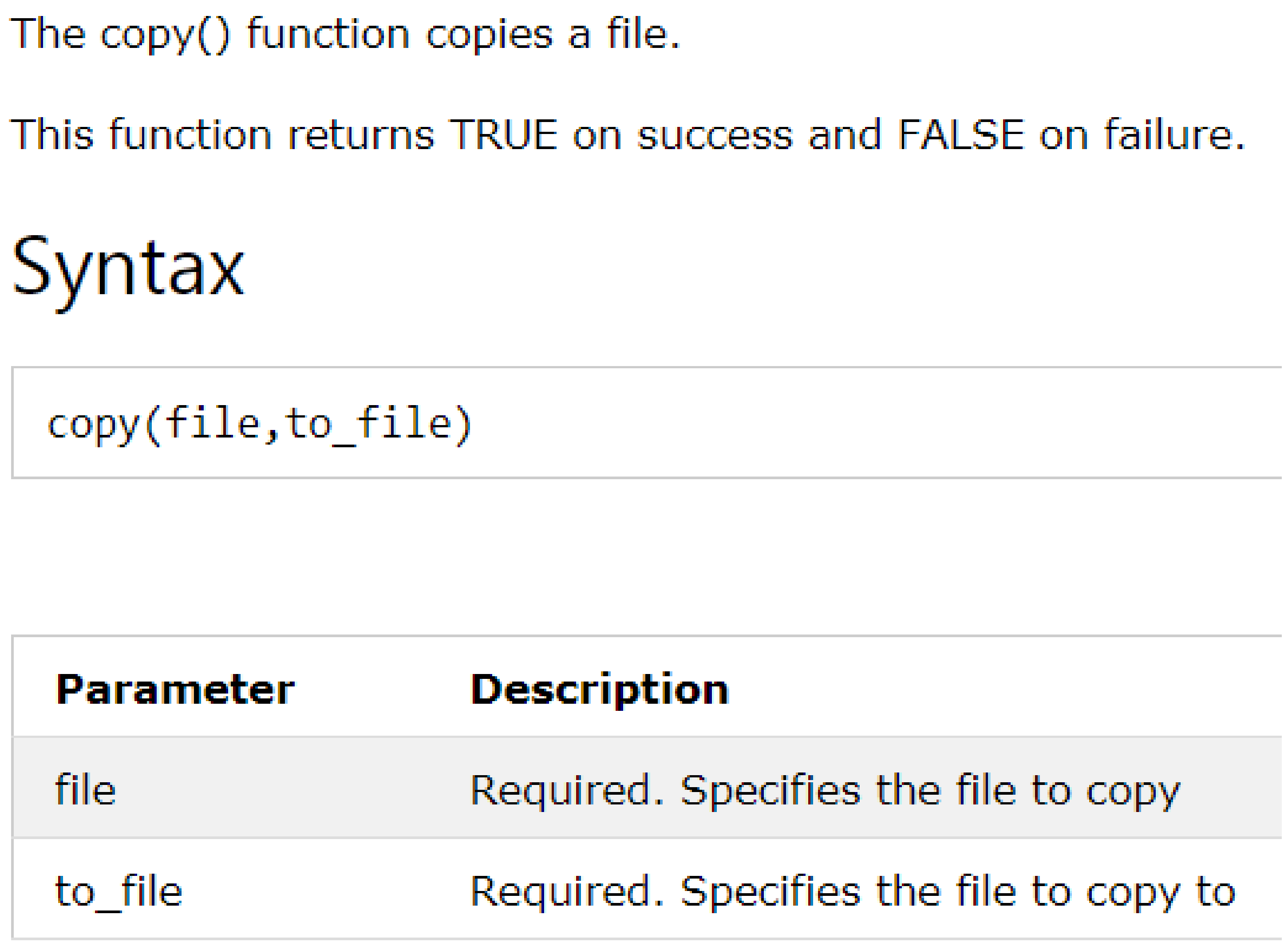
Syntax

```
file(path,include_path,context)
```

Parameter	Description
path	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream. Can be skipped by using NULL.

- Example 8

copy() Function [1]



The copy() function copies a file.

This function returns TRUE on success and FALSE on failure.

Syntax

```
copy(file,to_file)
```

Parameter	**Description**
file	Required. Specifies the file to copy
to_file	Required. Specifies the file to copy to

- Example 8

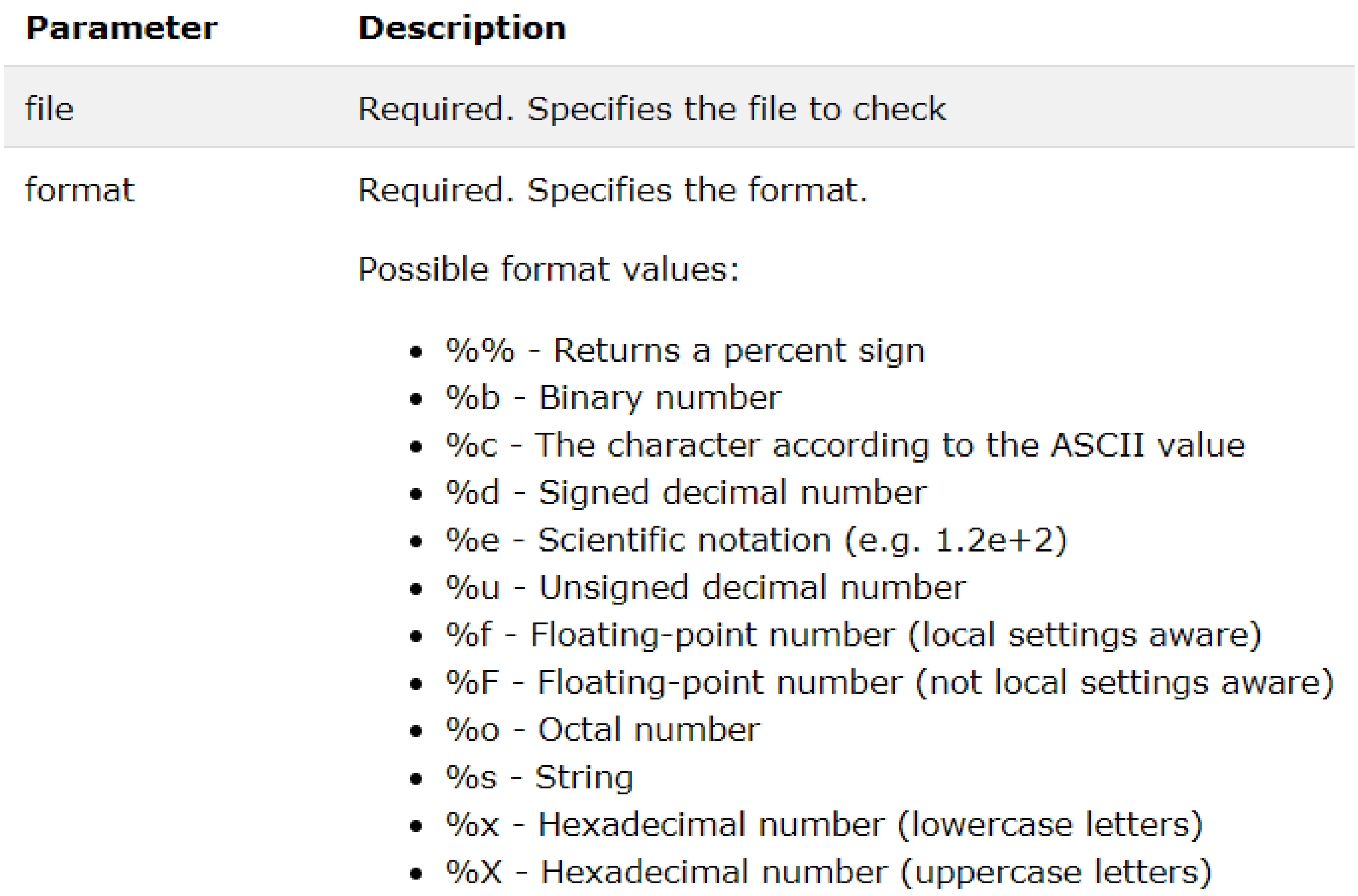
fscanf() Function [1]

- The fscanf() function parses the input from an open file according to the specified format.

```
fscanf(file,format,mixed)
```

- Example 9

fscanf() Function [1]



Parameter	Description
file	Required. Specifies the file to check
format	Required. Specifies the format. Possible format values: • %% - Returns a percent sign • %b - Binary number • %c - The character according to the ASCII value • %d - Signed decimal number • %e - Scientific notation (e.g. 1.2e+2) • %u - Unsigned decimal number • %f - Floating-point number (local settings aware) • %F - Floating-point number (not local settings aware) • %o - Octal number • %s - String • %x - Hexadecimal number (lowercase letters) • %X - Hexadecimal number (uppercase letters)

filemtime() Function [1]

- The filemtime() function returns the last time the file content was modified.
- This function returns the last change time as a Unix timestamp on success, FALSE on failure.

```
filemtime(filename)
```

- Example 12.php

filectime() Function [1]

- The filectime() function returns the last time the specified file was changed.
- This function checks for the inode changes as well as regular changes. Inode changes is when permissions, owner, group or other metadata is changed.
- This function returns the last change time as a Unix timestamp on success, FALSE on failure.

```
filectime(filename)
```

- Example 12.php

fileatime() Function [1]

- The fileatime() function returns the last access time of the specified file.
- This function returns the last access time as a Unix timestamp on success, FALSE on failure.

```
        fileatime(filename)
```

- Example 12.php

stat() Function [1]

- The stat() function returns information about a file.
- This function returns an array with the following elements:
  - [0] or [dev] - Device number
  - [1] or [ino] - Inode number
  - [2] or [mode] - Inode protection mode
  - [3] or [nlink] - Number of links
  - [4] or [uid] - User ID of owner
  - [5] or [gid] - Group ID of owner
  - [6] or [rdev] - Inode device type
  - [7] or [size] - Size in bytes
  - [8] or [atime] - Last access (as Unix timestamp)
  - [9] or [mtime] - Last modified (as Unix timestamp)
  - [10] or [ctime] - Last inode change (as Unix timestamp)
  - [11] or [blksize] - Blocksize of filesystem IO (if supported)
  - [12] or [blocks] - Number of blocks allocated
- Syntax: stat(*filename*)
- Example 12.php

PHP File upload

File Upload [1]

- **Configure The "php.ini" File**
  - First, ensure that PHP is configured to allow file uploads.
  - In your "php.ini" file, search for the **file_uploads** directive, and set it to On

- **Example:** fileupload.html, 10.php

File Upload [1]

- There is one global PHP variable called **$_FILES**. This variable is an associate double dimension array and keeps all the information related to uploaded file. So if the value assigned to the input's name attribute in uploading form was **file**, then PHP would create following five variables −
  - **$_FILES['file']['tmp_name']** − the uploaded file in the temporary directory on the web server.
  - **$_FILES['file']['name']** − the actual name of the uploaded file.
  - **$_FILES['file']['size']** − the size in bytes of the uploaded file.
  - **$_FILES['file']['type']** − the MIME type of the uploaded file.
  - **$_FILES['file']['error']** − the error code associated with this file upload.

File Upload [1]

- The basename() function **returns** the **filename** from a path.
- The pathinfo() Function
  - The pathinfo() function returns an array that contains information about a path.
  - The following array elements are returned:
    - [dirname]
    - [basename]
    - [extension]
  - pathinfo(path,options)
    - Options:
      - PATHINFO_DIRNAME - return only dirname
      - PATHINFO_BASENAME - return only basename
      - PATHINFO_EXTENSION - return only extension

mkdir() function [1]

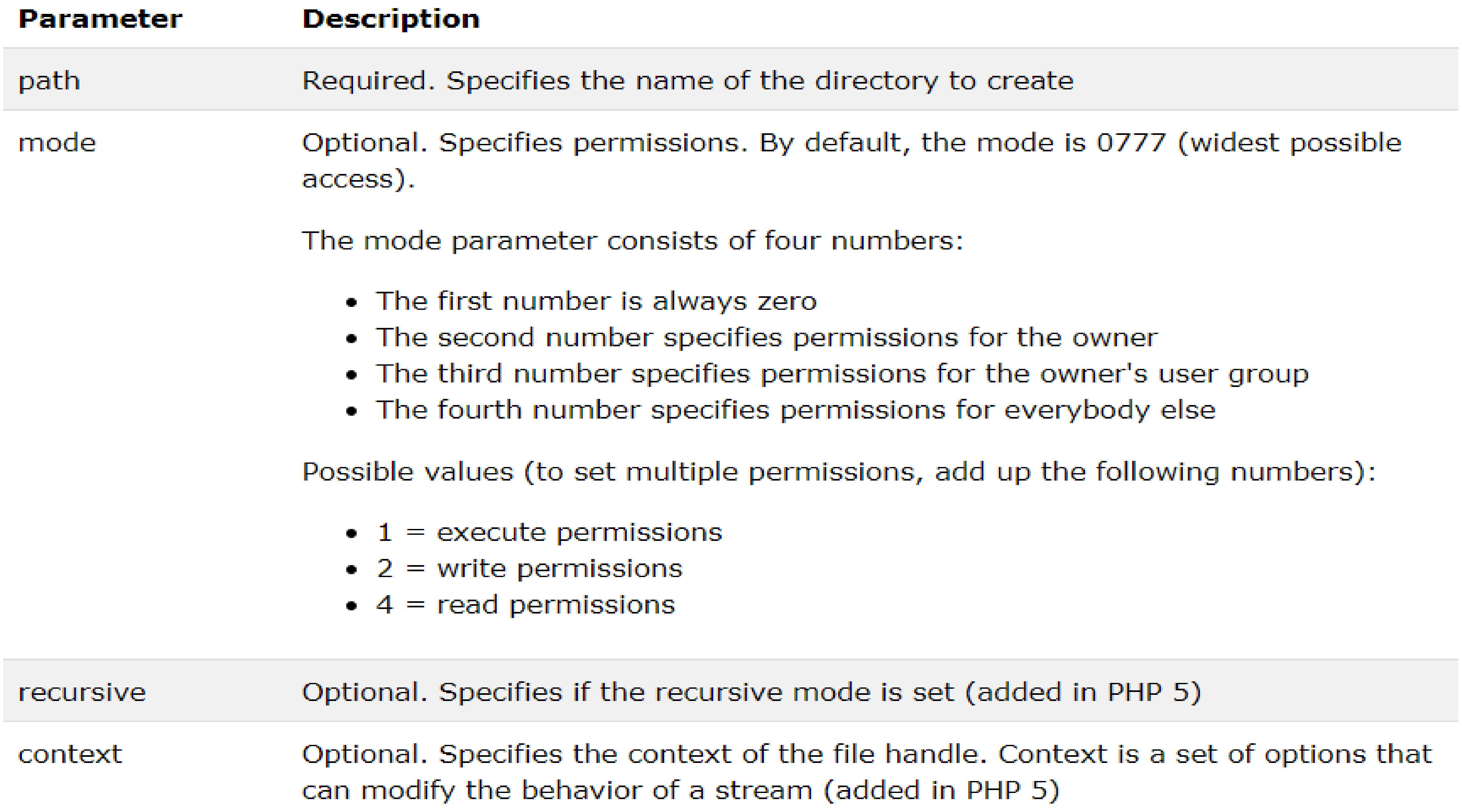
- The mkdir() function creates a directory.
- This function returns TRUE on success, or FALSE on failure.
- Syntax:

```
mkdir(path,mode,recursive,context)
```

- **Note:** The mode parameters is ignored on Windows platforms.

- **Example:** dirfileupload.html, 11.php
-

mkdir() function [1]



Parameter	Description
path	Required. Specifies the name of the directory to create
mode	Optional. Specifies permissions. By default, the mode is 0777 (widest possible access). The mode parameter consists of four numbers: • The first number is always zero • The second number specifies permissions for the owner • The third number specifies permissions for the owner's user group • The fourth number specifies permissions for everybody else Possible values (to set multiple permissions, add up the following numbers): • 1 = execute permissions • 2 = write permissions • 4 = read permissions
recursive	Optional. Specifies if the recursive mode is set (added in PHP 5)
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream (added in PHP 5)

rmdir() function [1] [2]

- The rmdir() function removes an empty directory.
- This function returns TRUE on success, or FALSE on failure.

```
rmdir(dir,context)
```

is_dir() function [1]

- The is_dir() function checks whether the specified file is a directory.
- This function returns TRUE if the directory exists.

```
is_dir(file)
```

is_file() function [1]

References

1. https://www.w3schools.com/php/
2. https://andy-carter.com/blog/recursively-remove-a-directory-in-php

Thank you....