EXCLUSIVE PHP CHEAT-SHEET:

Your Secrets to Coding Fast

BitDegree

INTRODUCTION

If you're reading this, you probably know what PHP is, and might even be familiar all of the different functions it performs. In this ebook, I would like to share with you some of the most commonly used PHP features. Whether you're familiar with all of them or this is your first time hearing about them, one thing's for sure - if you use PHP, these functions are a must.

TABLE OF CONTENTS

1 PHP 5 SESSIONS

A session is a methods of storing data (using variables) so the browser can use it throughout multiple webpages. In contrast to cookies, the data is not kept on the user's system. Session **variables** contain data about the current user, and all pages contained in a single web application. The session **ends** once the window or tab in which the webpage was loaded, is closed.

Session data is not **permanent**, but you can load permanent user data for particular users using **databases**.

Starting PHP Sessions

To start a session, you must use the function `session_start()`.

To set session variables, you will need to apply a global variable `$_SESSION`.

Example #1:

```
1.  <?php session_start();?> // Start the session
2.  <!DOCTYPE html>
3.  <html>
4.  <body>
5.  <?php
6.  $_SESSION["color"]= "blue"; // Set session variables
7.  $_SESSION["animal"]= "dog";
8.  echo "The session variable are set up.";
9.  ?>
10. </body>
11. </html>
```

Getting PHP Sessions Variable Values

To continue, we create the next example. Using this file we will access the data on the first example.

Notice how session data in form of variables must be individually retrieved (`session_start()`). In addition to that the `$_SESSION` variable holds all of the session data declared in your files.

Example #2:

```
1.  <?php session_start();?>
2.  <!DOCTYPE html>
3.  <html>
4.  <body>
5.  <?php// Echo session variables that were set on previous page
6.  echo "The color of my choice is " . $_SESSION["color"] .
  ".<br>";
7.  echo "The animal of my choice is " . $_SESSION["animal"] . ".";
8.  ?>
9.  </body>
10. </html>
```

In the code example below, it is shown how you can display all of the currently declared session variables.

Example #3:

```
1.  <?php session_start();?>
2.  <!DOCTYPE html>
3.  <html>
4.  <body>
5.  <?php
6.  print_e($_SESSION);
7.  ?>
8.  </body>
9.  </html>
```

Modifying PHP Sessions Variables

You can change sessions variables by simply overwriting.

Example #4:

```
1.  <?php session_start();?>
2.  <!DOCTYPE html>
3.  <html>
4.  <body>
5.  <?php //you can change a session variable by assigning a new
value
6.  $_SESSION[“color”] = “red”;
7.  print_e($_SESSION);
8.  ?>
9.  </body>
10. </html>
```

Destroying PHP Sessions

Using `session_unset()` will remove all of the global variable, while `session_destroy()` will destroy the session entirely (however the effect of both is similar).

Example #5:

```
1.  <?php session_start();?>
2.  <!DOCTYPE html>
3.  <html>
4.  <body>
5.  <?php
6.  session_unset(); //remove all session variables
7.  session_destroy(); // destroy the session
8.  ?>
9.  </body>
10. </html>
```

2 PHP 5 SORTING ARRAYS

PHP 5 offers multiple inbuilt functions to sort a PHP array. You can sort elements in PHP in numerical, alphabetical, descending and ascending orders.

These are the various array sorting functions in PHP:

- **sort()** – sorts a PHP array in an ascending order;
- **rsort()** – sorts a PHP array in a descending order;
- **asort()** – sorts associative arrays in an ascending order, by the value;
- **ksort()** – sorts associative arrays in an ascending order, by the key;
- **arsort()** – sorts associative arrays in a descending order, by the value;
- **krsort()** – sorts associative arrays in a descending order, by the key.

2. 1. sort()

This function sorts a PHP array in an ascending order. Below is an example of the same function sorting an array in an alphabetical order.

Example:

```
1.  <?php
2.  $guitars = array("Fender", "Gibson", "Warvick");
3.  sort($guitars);
4.  ?>
```

Below there's an example of the same function sorting an array in a numerical order:

Example:

```
1.  <?php
2.  $numerals = array(5, 7, 3, 23, 12);
3.  sort($numerals);
4.  ?>
```

rsort()

This function sorts a PHP array in a descending order.

Example (alphabetical order):

```
1.  <?php
2.  $guitars = array("Fender", "Gibson", "Warvick");
3.  rsort($guitars);
4.  ?>
```

Example (numerical order):

```
1.  <?php
2.  $numerals = array(5, 7, 3, 23, 12);
3.  rsort($numerals);
4.  ?>
```

asort()

This function sorts associative arrays in an ascending order, by the value.

Example:

```
1.  <?php
2.  $weight = array("Pete"=>"75", "Benjamin"=>"89",
3.  "Jonathan"=>"101");
4.  asort($weight);
5.  ?>
```

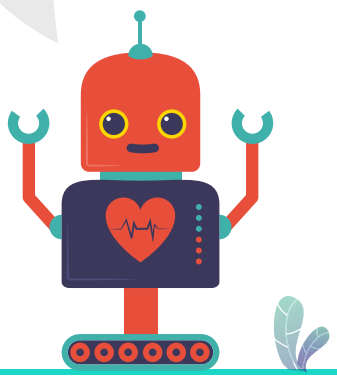
ksort()

This function sorts associative arrays in an ascending order, by the key.

Example:

```
1.  <?php
2.  $weight = array("Pete"=>"75", "Benjamin"=>"89",
3.  "Jonathan"=>"101");
4.  ksort($weight);
5.  ?>
```

The biggest online brands (such as Facebook and Yahoo) are powered by PHP.



arsort()

This function sorts associative arrays in a descending order, by the value.

Example:

```
1.  <?php
2.  $weight = array("Pete"=>"75", "Benjamin"=>"89",
3.  "Jonathan"=>"101");
4.  arsort($weight);
5.  ?>
```

krsort()

This function sorts associative arrays in a descending order, by the key.

Example:

```
1.  <?php
2.  $weight = array("Pete"=>"75", "Benjamin"=>"89",
3.  "Jonathan"=>"101");
4.  krsort($weight);
5.  ?>
```

3 PHP ARRAY PUSH FUNCTION

The `array_push()` function inserts elements into the end of an array. You can as many values as you like. Your added elements will always have numeric keys, even if your array has string keys. It also returns the elements number in the array. It was newly introduced in **PHP4**.

array_push()

This function is used to insert elements into the end of an array.

Example:

```
1.  <?php
2.  $z=array("me","you", "he");
3.  array_push($z,"she","it");
4.  print_r($z);
5.  ?>
```

Syntax

This is an example of how `array_push()` should be written:

```
array_push(array,value1,value2...);
```

Parameter: Description.

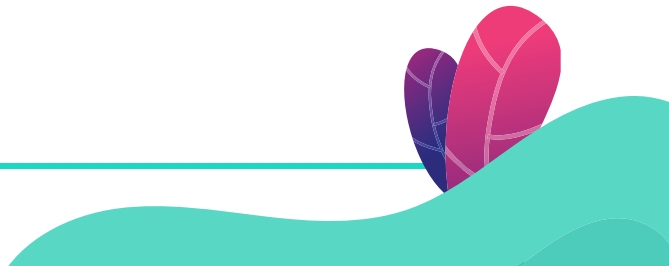
array: Needed. Defines an array.

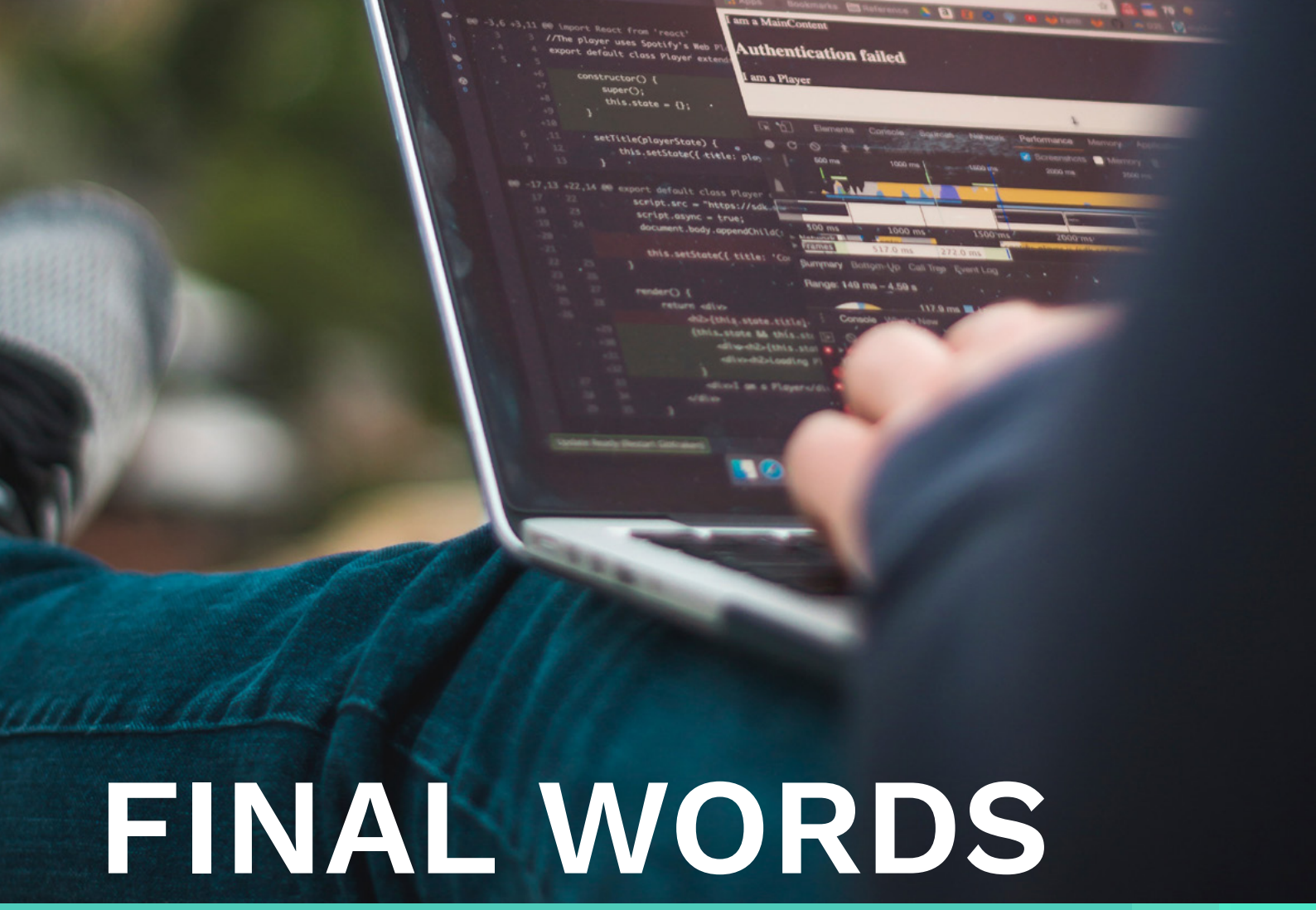
value1: Needed. Defines the value to add.

value2: Optional. Defines the value to add.

Example:

```
1.  <?php
2.  $z=array("Drago"=>"blue","Rex"=>"brown");
3.  array_push($z,"black","purple");
4.  print_r($z);
5.  ?>
```





FINAL WORDS

We've reached the end of the ebook. If you're reading this, congratulations - you have just brushed up your knowledge on PHP! We all know that programming languages can be tough, so it's always a good idea to have a quick cheat sheet - and this ebook is perfect for that! Keep it close and never again encounter the need to google examples for PHP array sortings!