PHP BASICS

Including PHP in a File

```
<?php // place PHP code here ?>
```

Writing Comments

```
//
```

Denotes comments that only span on one line

```
/*...*/
```

Everything between /* and */ is not executed, also works across several lines

Outputting Data

```
<?php echo "<h1>PHP Cheat Sheet</h1>"; ?>
```

Writing PHP Functions

```
function NameOfTheFunction() {
      //place PHP code here
}
```

VARIABLES AND CONSTANTS

Defining Variables

```
<?php
      $BlogPostTitle = "PHP Cheat Sheet";
?>
```

Types of Data

Integers

Integers are non-decimal numbers between -2,147,483,648 and 2,147,483,647. They must have at least one digit and no decimal point. Can be in decimal, hexadecimal or octal.

Floats

This is the name for numbers with a decimal point or in exponential form.

Strings

This simply means text, we will talk about it in detail further below.

Boolean values

Meaning true/false statements.

Arrays

Arrays are variables that store several values. We will talk about them in detail further below.

NULL

A variable that is NULL doesn't have any value.

Variable Scope

```
function myFunction() {
      global $a, $b;
      $b = $a - $b;
}
```

Predefined Variables

$GLOBALS

Used to access global variables from anywhere inside a PHP script.

$_SERVER

Contains information about the locations of headers, paths and scripts.

$_GET

Can collect data that was sent in the URL or submitted in an HTML form.

$_POST

Used to gather data from an HTML form and to pass variables.

$_REQUEST

Also collects data after submitting an HTML form

Variable-handling Functions

empty

Checks whether a variable is empty or not

gettype

Retrieves the variable type

is_array

Checks whether a variable is an array

is_bool

Finds out if a variable is a boolean

is_float

Find out if the type of a variable is float, alternatives: is_double and is_real

is_int

Check if the type of a variable is an integer, is_integer and is_long also works

is_null

Checks whether a variable's value is NULL

is_string

Find out whether the type of a variable is a string

isset

Determine if a variable has been set and is not NULL

print_r

Provides human-readable information about a variable

unset

Unsets a variable

var_dump

Dumps information about a variable

Constants

define(name, value, true/false)

PHP ARRAYS – GROUPED VALUES

Indexed arrays

Arrays that have a numeric index

Associative arrays

Arrays where the keys are named

Multidimensional arrays

Arrays that contain one or more other arrays

Declaring an Array in PHP

```
<?php
      $cms = array("WordPress", "Joomla", "Drupal");
      echo "What is your favorite CMS? Is it " . $cms[0] . ", " .
      $cms[1] . " or " . $cms[2] . "?";
?>
```

Array Functions

arsort

Sorts an associative array in descending order according to the value

asort

Sorts an associative array in ascending order according to the value

count

Count all elements in an array, alternatively use sizeof

krsort

Sorts an associative array by key in reverse order

ksort

Sorts an associative array by key

rsort

Sort an array in reverse order

sort

Sorts an indexed array in ascending order

PHP STRINGS

Defining Strings

Single quotes

This is the simplest way. Just wrap your text in ' markers and PHP will handle it as a string.

Double quotes

As an alternative you can use ". When you do, it's possible to use the escape characters below to display special characters.

Escape Characters

\n — Line feed
\r — Carriage return
\t — Horizontal tab
\v — Vertical tab
\e — Escape
\f — Form feed
\\ — Backslash
\$ — Dollar sign

\' — Single quote
\" — Double quote
\[0-7]{1,3} — Character in octal notation
\x[0-9A-Fa-f]{1,2} — Character in hexadecimal notation
\u{[0-9A-Fa-f]+} — String as UTF-8 representation

PHP OPERATORS

Arithmetic Operators

```
+     — Addition
-     — Subtraction
*     — Multiplication
/     — Division
%     — Modulo (the remainder of value divided by another)
**    — Exponentiation
```

Assignment Operators

```
+=    — a += b is the same as a = a + b
-=    — a -= b is the same as a = a - b
*=    — a *= b is the same as a = a * b
/=    — a /= b is the same as a = a / b
%=    — a %= b is the same as a = a % b
```

Comparison Operators

```
==    — Equal
===   — Identical
!=    — Not equal
<>    — Not equal
!==   — Not identical
<     — Less than
>     — Greater than
<=    — Less than or equal to
>=    — Greater than or equal to
<=>   — Less than, equal to, or greater than
```

Logical Operators

```
and   — And
or    — Or
xor   — Exclusive or
!     — Not
&&    — And
||    — Or
```

Bitwise Operators

```
&     — And
|     — Or (inclusive or)
^     — Xor (exclusive or)
~     — Not
<<    — Shift left
>>    — Shift right
```

Increment/Decrement Operators

`++$v`

Increments a variable by one, then returns it

`$v++`

Returns a variable, then increments it by one

`--$v`

Decrements the variable by one, returns it afterward

`$v--`

Returns the variable then decrements it by one

String Operators

`.`

Used to concatenate (mean combine) arguments

`.=`

Used to append the argument on the right to the left-side argument

LOOPS IN PHP

For Loop

```
for (starting counter value; ending counter value; increment by which
to increase) {
// code to execute goes here
}
```

Foreach Loop

```
foreach ($InsertYourArrayName as $value) {
      // code to execute goes here
}
```

While Loop

```
while (condition that must apply) {
      // code to execute goes here
}
```

Do..While Loop

```
do {
      // code to execute goes here;
} while (condition that must apply);
```

CONDITIONAL STATEMENTS

If Statement

```
if (condition) {
      // code to execute if condition is met
}
```

If..Else

```
if (condition) {
      // code to execute if condition is met
} else {
      // code to execute if condition is not met
}
```

If..Elseif..Else

```
if (condition) {
      // code to execute if condition is met
} elseif (condition) {
      // code to execute if this condition is met
} else {
      // code to execute if none of the conditions are met
}
```

Switch Statement

```
switch (n) {
      case x:
            code to execute if n=x;
            break;
      case y:
            code to execute if n=y;
            break;
      case z:
            code to execute if n=z;
            break;
      // add more cases as needed
      default:
            code to execute if n is neither of the above;
}
```

WORKING WITH FORMS IN PHP

Using GET vs POST

GET collects data via URL parameters. That means all variable names and their values are contained in the page address.

The advantage of this is that you're able to bookmark the information. Keep in mind that it also means that the information is visible to everyone. For that reason, GET is not suitable for sensitive information such as passwords. It also limits the amount of data that can be sent in ca 2000 characters.

POST, on the other hand, uses the HTTP POST method to pass on variables. This makes the data invisible to third parties, as it is sent in the HTTP body. You are not able to bookmark it.

With POST, there are no limits to the amount of information you can send. Aside from that, it also has advanced functionality and is therefore preferred by developers.

PHP Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

Syntax

```
function functionName() {
  code to be executed;
}
functionName();
```

Function Arguments

```
<?php
function familyName($fname, $year) {
  echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

HTTP FUNCTIONS IN PHP

HTTP Functions

`header()`

Sends a raw HTTP header to the browser

WORKING WITH MYSQL

MySQL Functions

`mysqli_close()`

Closes an open database connection

`mysqli_connect_errno()`

The error code from the last connection error

`mysqli_connect_error()`

The error description from the last connection error

`mysqli_connect()`

Opens a new connection to the MySQL server

`mysqli_errno()`

The last error code for the most recent function call

`mysqli_error()`

The last error description for the most recent function call

`mysqli_fetch_all()`

Fetches all result rows as an array

`mysqli_fetch_array()`

Fetches a result row as an associative, a numeric array, or both

mysqli_fetch_assoc()

Fetches a result row as an associative array

mysqli_fetch_row()

Fetches one row from a result set and returns it as an enumerated array

mysqli_num_rows()

The number of rows in a result set

mysqli_query()

Performs a query against the database

mysqli_real_escape_string()

Escapes special characters in a string for use in an SQL statement