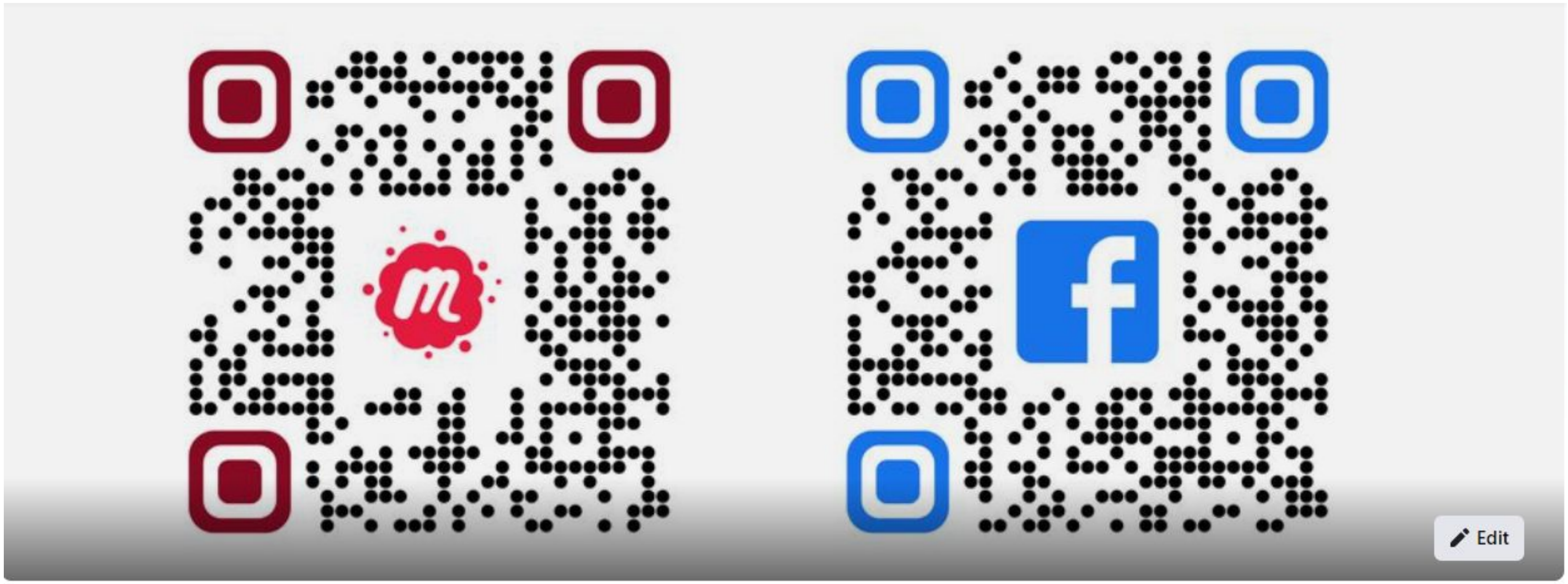
GROUND GURUS

"Effective center for teaching, learning, creating and development"

- **Darell Duma**
- **Software Engineer**
- **8 years in the IT industry**
- **darellduma.com**
- **mailme@darellduma.com**

GROUND GURUS



WordPress San Pablo

Public group · 23 members

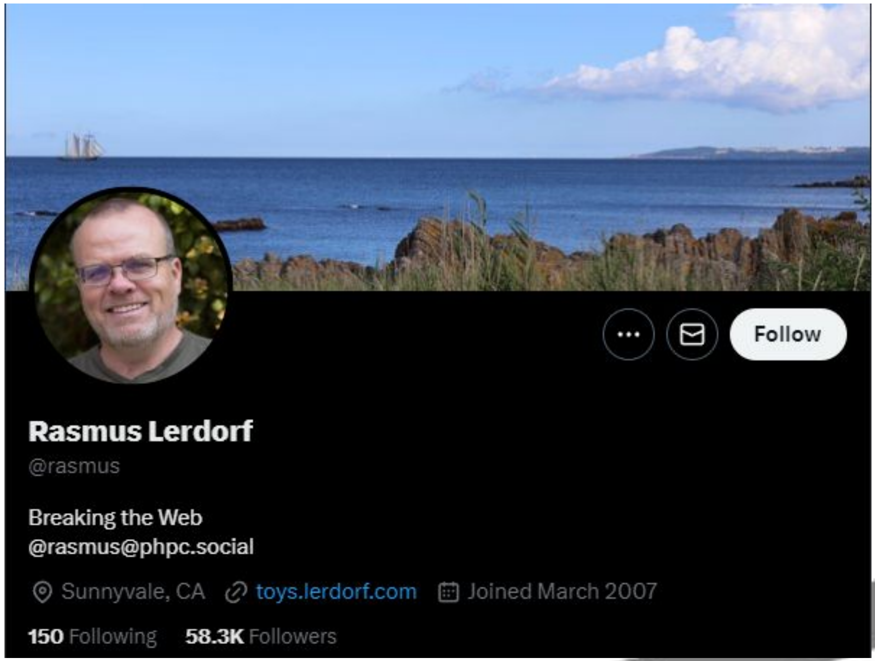
WordPress San Pablo

23 likes • 194 followers

PHP Live Class

PHP

- A popular general-purpose scripting language
- Acronym for PHP: Hypertext Preprocessor
- Widely-used, free, and efficient alternative to c#, java, and asp.
- Based on C language
- Released in 1995 by Rasmus Lerdorf
- Current versions: 8.2, 8.3



Rasmus Lerdorf
@rasmus

Breaking the Web
@rasmus@phpc.social

Sunnyvale, CA toys.lerdorf.com Joined March 2007

150 Following **58.3K** Followers

PHP Installation

Ways to Install PHP

- php.net
- XAMPP/WAMP/MAMP/LAMP
- Laragon
- Docker Images

AMP - Apache, MySQL/MariaDB, PHP

W - Windows

M - MACOS

L - Linux

X - Cross-operation

P - Perl

PHP Fiddle/Playground

- W3Schools Try It Yourself
- https://3v4l.org/

PHP Syntax

```
<?php
    // your PHP code goes here
?>
```

- PHP scripts starts with <?php
- And ends with ?>
- PHP files has the .php default extension
- PHP statements end with a semicolon (;)
- PHP keywords are not case-sensitive
- However, variable names are case-sensitive

PHP Comments

```
<?php
    // this is a comment
?>
```

- Ignored/skipped/not executed as part of the program
- It’s only purpose is to be read by someone who is looking at the code
- Let others understand your code
- Remind yourself what you did
- Leave out some parts of the code

PHP Single Line Comments

```
<?php
   // this is a comment
?>
```

- Starts with double forward slash (//)
- Any texts between // and the end of the line is ignored
- You can also use hash (#)

PHP Multiple Line Comments

```
<?php
   /*
      this is a comment
      this is also a comment
      this is the comment
   */
?>
```

- Starts with /* and ends with */
- Any texts between /* and */ are ignored
- Can also be used inside a code line

PHP Variables

- Variables are “containers” for storing values/information
- Starts with $ sign, followed by the name of the variable

```php
<?php
    $x = 5;
    $y = 90.7;
    $name = “Darell”;
    $is_allowed = true;
?>
```

Rules for Naming Variables

- Must start with $ sign, followed by the name of the variable
- Must start with a letter or an underscore character
- Must not start with a number
- Must only contain alphanumeric and underscore characters (A-z, 0-9, _)
- They are case-sensitive ($age and $AGE are two different variables)

```
<?php
    $name = “Darell”;
    $temperature = 32.7;
    $_context = context();
?>
```

Displaying Output (echo/print)

- echo and print are almost the same
- echo returns multiple values
- print returns 1
- echo can take multiple parameters
- print can take only 1 parameter
- echo is marginally faster than print

PHP echo

- without parenthesis

```
echo "Hello";
```

- with parenthesis

```
echo("Hello");
```

PHP echo (displaying text)

```
    echo "<h2>PHP is Fun!</h2>";

    echo "Hello world!<br />";

    echo "I'm about to learn PHP!<br>";

    echo "This ", "string ", "was ", "made ", "with multiple
 parameters.";
```

PHP echo (displaying variables)

```
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

echo "<h2>$txt1</h2>";
echo "<p>Study PHP at $txt2</p>";
```

PHP echo (using single quotes)

```php
$txt1 = "Learn PHP";
$txt2 = "groundgurus.net";


echo '<h2>' . $txt1 . '</h2>';
echo '<p>Study PHP at ' . $txt2 . '</p>';
```

PHP Data Types

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP Getting the Data Type

- Get the data type using the var_dump() function

```
$x = 5;

var_dump($x);

//returns int(5)



$name = “Darell”;

var_dump($name);

//returns string(6) “Darell”
```

PHP String Data Type

- A series of characters, like “Hello World”
- Can be any texts inside quotes
- Can use single or double quotes

```
$x = "Hello world!";

$y = 'Hello world!';

var_dump($x);

var_dump($y);
```

// both returns string(12) “Hello world!”

PHP Integer Data Type

- A non-decimal number
- Rules:
  - Must have at least 1 digit (0, 11, 222, …)
  - Must not have a decimal point (10, ~~12.5~~, 50, ~~99.99~~, 100)
  - Can either be positive or negative (10, -5, 200, -999)
  - Can be specified in Decimal (10), hexadecimal (16), octal (8), or binary (2) notation

PHP Boolean Data Type

- Represents 2 possible values
    - true
    - false

PHP Boolean Data Type

- Represents 2 possible values
  - true
  - false

PHP Array Data Type

- Can store multiple values in one (1) variable

```
$cars = array("Volvo","BMW","Toyota");

$grades = [90, 95, 95, 92, 97];

$settings = array("top", 0, true);
```

PHP Object Data Type

- Classes and objects are two (2) main aspects of object-oriented programming
- A class is a template for objects
- An object is an instance of a class

```
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("red", "Volvo");

var_dump($myCar);
```

PHP NULL Data Type

- Special data type that can only have one value: null

```
$x = "Hello world!";

$x = null;

var_dump($x);
```

PHP Type Casting

- Used to change a variable’s data type to another

`(string)` - Converts to data type String

`(int)` - Converts to data type Integer

`(float)` - Converts to data type Float

`(bool)` - Converts to data type Boolean

`(array)` - Converts to data type Array

`(object)` - Converts to data type Object

~~`(unset)` - Converts to data type NULL~~

```
$a = 5;        // Integer

$a = (string)$a;

var_dump($a); //returns “5”
```

PHP Math Functions

- Used to perform mathematical tasks on numbers

  `pi()` - returns the value of pi

  `min()` - returns the lowest value in a list of arguments

  `max()` - returns the highest value in a list of arguments

  `abs()` - returns the absolute (positive) value of a number

  `sqrt()` - returns the square root of a number

  `round()` - rounds a floating-point number to its nearest integer

  `rand()` - generate a random number

  `floor()` - rounds a number down to its nearest integer

  `ceil()` - rounds a number up to its nearest integer

PHP Constants (define())

- Like variables, except that once they are defined they cannot be modified
- Create a constant using define()
- Parameters:
  - *name*: Specifies the name of the constant
  - *value*: Specifies the value of the constant
  - *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false. Note: Defining case-insensitive constants was deprecated in PHP 7.3. PHP 8.0 accepts only false, the value true will produce a warning.

```
define("GREETING", "Welcome to W3Schools.com!");

echo GREETING;
```

PHP Constants (const)

- Create a constant using const

```
const MYCAR = "Volvo";

echo MYCAR;
```

PHP Constants (const)

- `const` vs. `define()`
    - `const` are always case-sensitive
    - `define()` has a case-insensitive option.
    - `const` cannot be created inside another block scope, like inside a function or inside an `if` statement.
    - `define` can be created inside another block scope.

PHP Magic Constants

- Constants that changes value depending on where they are used
- Written with two underscore at the start and at the end except for the ClassName:class constant

PHP Magic Constants Examples

__CLASS__	If used inside a class, the class name is returned.
__DIR__	The directory of the file.
__FILE__	The file name including the full path.
__FUNCTION__	If inside a function, the function name is returned.
__LINE__	The current line number.
__METHOD__	If used inside a function that belongs to a class, both class and function name is returned.
__NAMESPACE__	If used inside a namespace, the name of the namespace is returned.
__TRAIT__	If used inside a trait, the trait name is returned.
ClassName::class	Returns the name of the specified class and the name of the namespace, if any.

PHP Operations

- used to perform operations on variables and values.
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Array operators
  - Conditional assignment operators

PHP Arithmetic Operators

- used with numeric values to perform common arithmetical operations

Operator	**Name**	**Example**	**Result**
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators

- used with numeric values to write a value to a variable

Assignment	**Same as…**	**Description**
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

- used with numeric values to write a value to a variable

Operator	**Name**	**Example**	**Result**
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y

PHP Comparison Operators (contd.)

- used with numeric values to write a value to a variable

Operator	**Name**	**Example**	**Result**
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Comparison Operators (contd.)

- used to compare two values (number or string)

Operator	Name	Example	Result
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment/Decrement Operators

- used increment/decrement a value of a variable

Operator	Same as	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

- used to combine conditional statements

Operator	**Name**	**Example**	**Result**
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

- designed for strings

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators

- used to compare arrays

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP Conditional Assignment Operators

- used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	$x = expr1 ? expr2 : expr3	Returns the value of $x. The value of $x is expr2 if expr1 = TRUE. The value of $x is expr3 if expr1 = FALSE
??	Null coalescing	$x = expr1 ?? expr2	Returns the value of $x. The value of $x is expr1 if expr1 exists, and is not NULL. If expr1 does not exist, or is NULL, the value of $x is expr2. Introduced in PHP 7

PHP Conditional Assignment Operators

- used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	$x = expr1 ? expr2 : expr3	Returns the value of $x. The value of $x is expr2 if expr1 = TRUE. The value of $x is expr3 if expr1 = FALSE
??	Null coalescing	$x = expr1 ?? expr2	Returns the value of $x. The value of $x is expr1 if expr1 exists, and is not NULL. If expr1 does not exist, or is NULL, the value of $x is expr2. Introduced in PHP 7

GROUND GURUS

PHP Conditional Statements

- used to perform different actions depending on the conditions
  - if statement - executes some code if one condition is true
  - if...else statement - executes some code if a condition is true and another code if that condition is false
  - if...elseif...else statement - executes different codes for more than two conditions
  - switch statement - selects one of many blocks of code to be executed

PHP Shorthand If

- You can write shorter if on one line

```
$a = 5;

if ($a < 10) $b = "Hello";

echo $b
```

PHP Nested If

- You can write shorter if on one line

```
$grade = 90;

$subject = “Math”

if ($subject === “Math”) {

   echo “Subject: $subject: “;

   if ($grade >= 75) {

    echo “Passed";

  } else {

    echo "Failed";

  }

}
```

PHP Loops

- used to repeat a sequence of codes
    - `while` - loops through a block of code as long as the specified condition is true
    - `do...while` - loops through a block of code once, and then repeats the loop as long as the specified condition is true
    - `for` - loops through a block of code a specified number of times
    - `foreach` - loops through a block of code for each element in an array

PHP Break

- break statement can be used to jump out of a for loop.

```
for ($x = 0; $x < 10; $x++) {

     if ($x == 4) {

       break;

     }

     echo "The number is: $x <br>";

}
```

PHP Continue

- continue stops the current iteration in the for loop and continue with the next.

```
for ($x = 0; $x < 10; $x++) {

     if ($x == 4) {

       continue;

     }

     echo "The number is: $x <br>";

}
```

PHP Functions

- PHP has over 1000 built-in functions
- It is also possible to create your own custom function

```
function myMessage() {

  echo "Hello world!";

}



myMessage();
```

PHP Function Arguments

- Information can be passed through arguments
- Specified after the function name, inside parenthesis
- You can add as many arguments as you want, just separate them with a comma

```
function familyName($fname) {

  echo "$fname Zoldyck.\n";

}

familyName('Zeno');

familyName('Silva');

familyName('Killua');
```

PHP Function Arguments (2 arguments)

```
function familyName($fname,$birth_year) {

  echo "$fname Zoldyck - $birth_year\n";

}

familyName(‘Zeno’, ‘1932’);

familyName(‘Silva’, ‘1953’);

familyName(‘Killua’, ‘1999’);
```

PHP Default Arguments

```php
function set_height($height=50) {
    echo "The height is : $height\n";
}
set_height(30);
set_height();
set_height(150);
```

PHP Functions Returning Values

- To let a function return a value, use the return statement:

```
function sum($x, $y) {
  return $x + $y;
}


$sum = sum(5,3);
echo “The sum of 5 and 3 is $sum”;
echo “The sum of 4 and 2 is” . sum(4,2);
```

PHP Arrays

- stores multiple values in one single variable

```
$cars = array("Volvo", "BMW", "Toyota");
```

- Types of arrays:
    - Indexed arrays - Arrays with a numeric index
    - Associative arrays - Arrays with named keys
    - Multidimensional arrays - Arrays containing one or more arrays

PHP Arrays

- Index Arrays

```
$cars = array("Volvo", "BMW", "Toyota");

echo $cars[1]; //outputs BMW
```

- Associative arrays

```
$person = array("first_name"=>"Darell", "last_name", "Duma");

echo $person["first_name"]; //outputs Darell
```

PHP Arrays

- Multidimensional Arrays

```
$cars = array (
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
     array("Land Rover",17,15)
);
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2];
//Saab: Instock: 5, sold: 2
```

PHP Updating Arrays Items

- Indexed array

```
$cars = array("Volvo", "BMW", "Toyota");

$cars[0] = "Toyota";
```

- Associative array

```
$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);

$cars["year"] = 2024;
```

PHP Adding Arrays Items (Single)

- Indexed array

```
$fruits = array("Apple", "Banana", "Cherry");

$fruits[] = "Orange";
```

- Associative array

```
$cars = array("brand" => "Ford", "model" => "Mustang");

$cars["color"] = "Red";
```

PHP Adding Arrays Items (Multiple)

- Indexed array

```
$fruits = array("Apple", "Banana", "Cherry");

array_push($fruits, "Orange", "Kiwi", "Lemon");
```

- Associative array

```
$cars = array("brand" => "Ford", "model" => "Mustang");

$cars += ["color" => "red", "year" => 1964];
```

PHP Superglobals

- Predefined variables
- Always accessible, regardless of scope
    - `$GLOBALS`
    - `$_SERVER`
    - `$_REQUEST`
    - `$_POST`
    - `$_GET`
    - `$_FILES`
    - `$_ENV`
    - `$_COOKIE`
    - `$_SESSION`

PHP Regular Expression (RegEx)

- a sequence of characters that forms a search pattern

```
$exp = "/David/i";
```

PHP preg_match()

- tells you whether a string contains matches of a pattern

```
$str = "My name is David";

$pattern = "/David/i";

echo preg_match($pattern, $str);

//outputs 1
```

PHP preg_match_all()

- tell you how many matches were found for a pattern in a string

```php
$str = "The rain in SPAIN falls mainly on the plains.";

$pattern = "/ain/i";

echo preg_match_all($pattern, $str);

//outputs 4
```

PHP preg_replace()

- will replace all of the matches of the pattern in a string with another string

```
$str = "Visit Microsoft!";

$pattern = "/microsoft/i";

echo preg_replace($pattern, "Ground Gurus", $str);

//outputs Visit Ground Gurus!
```

Thank You!

Contact us

www.groundgurus.com
09171100312 | 09771673162
Facebook: @groundgurus
Instagram: @ggurus2016

GROUND GURUS