



Scripting Facil con PHP

Elminson De Oleo Baez

Tabla de Contenido

Copyright	6
Dedicatoria	7
Sobre este material	8
Que es PHP?	9
Características de PHP	11
¿Qué puede hacer PHP?	13
Lo que necesitas para empezar	15
Empezando con PHP (Todavía no)	18
Empezando con PHP (Ahora si)	23
Comentarios (Documentando el código)	26
Variables	30
Visualizando el valor de Variables	31
Control de flujo	36

Tipos de Datos	37
Cadena de Texto	38
Operadores	39
Operadores Lógicos	40
Manejo de cadenas	41
Condicionales	42
Iteraciones	43
Funciones	44
Clases	45
Manejo de Excepciones	46
Composer	47
Laravel-Zero	48
Nuestro primer script	49
Scripts Avanzados	50
Referencias Bibliográficas	51

Biografia 52

Scripting Facil con PHP

Tu Guia Facil y Rapida para aprender Scripting con PHP

ELMINSON DE OLEO BAEZ

Copyright

Copyright © 2022 by Elminson De Oleo Baez

All rights reserved. No part of this publication may be reproduced, stored or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise without written permission from the publisher. It is illegal to copy this book, post it to a website, or distribute it by any other means without permission. First edition

Dedicatoria

A mi esposa

Mis hijos: Emil, Ethan, Enzo y Emmi

Mis hermanos: Jose y Jenny

Mis padres: Elio Y Kilcia

Mi Familia Los Baez(Lo bai)

***Mis hermanos de otras madres
(ustedes saben quienes son)***

Mis Amigos

A mi Mentor: Antonio Perpnan

Sobre este material

Sobre este material Aunque hay mucho material en internet elige un tema, dedícale tiempo y esfuerzo. Elige algo que te guste y que apasione porque de esta forma disfrutaras cada minuto que le dediques, nunca es tarde para aprender y como dice mi madre: “El conocimiento no pesa”.

Elminson

Que es PHP?



PHP (acrónimo recursivo de PHP: Hypertext Preprocessor*) fue creado por Rasmus Lerdorf y apareció en 1995, es un lenguaje de programación de propósito general de código abierto ampliamente utilizado, es especialmente adecuado para el desarrollo web y se puede incrustar en HTML. Bien, pero ¿qué significa realmente? Un ejemplo nos puede aclarar las cosas:

Ejemplo #1 Un ejemplo introductorio

```
<!DOCTYPE html>
<html lang="es" dir="ltr">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
  <title>Ejemplo básico PHP</title>
</head>

<body>
  <?php
    echo 'Hola mundo'; // Esto es PHP!!
  ?>
</body>

</html>
```

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace “algo” (en este caso, mostrar “¡Hola, soy un script de PHP!”).

El código de PHP está encerrado entre las etiquetas especiales de comienzo y final **<?php** y **?>** que permiten entrar y salir del “modo PHP”.

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. No sienta miedo de leer la larga lista de características de PHP. En unas pocas horas podrá empezar a escribir sus primeros scripts.

Ejemplo #2

```
class Coche {  
  
    public $color = 'rojo';  
    public $potencia;  
    public $marca;  
  
    public function getColor()  
    {  
        return $this->color;  
    }  
}  
  
$miCoche = new Coche();  
$miCoche->color = 'verde';  
$miCoche->potencia = 120;  
$miCoche->marca = 'audi';
```

No te preocupes, aprenderás que significa cada elemento de este programa

Características de PHP



Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.

- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de php arreglos(array).
- El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando módulos (llamados extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).

- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.
- Debido a su flexibilidad, ha tenido una gran acogida como lenguaje base para las aplicaciones WEB de manejo de contenido, y es su uso principal.

Separación de instrucciones

Como en *C* o en *Perl*, *PHP* requiere que las instrucciones terminen en punto y coma al final de cada sentencia. La etiqueta de cierre de un bloque de código de PHP automáticamente implica un punto y coma; no es necesario usar un punto y coma para cerrar la última línea de un bloque de PHP. La etiqueta de cierre del bloque incluirá la nueva línea final inmediata si está presente. Cada script de PHP se compone de instrucciones, tales como funciones, asignaciones de variables, salida de datos, directivas, etc. Excepto en algunos casos, cada uno de estas instrucciones deben terminar (tal como en *C*, *Perl* y *JS*) con un punto y coma. Estas tres líneas de código PHP son válidas.

```
<?php
    echo 'Esto es una prueba';
?>

<?php echo 'Esto es una prueba' ?>

<?php echo 'Hemos omitido la última etiqueta de cierre';
```

¿Qué puede hacer PHP?



Cualquier cosa. PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI*, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Aunque PHP puede hacer mucho más.

Existen tres campos principales donde se usan scripts de PHP.

Scripts del lado del servidor.

Este es el campo más tradicional y el foco principal. Son necesarias tres cosas para que esto funcione: el analizador de PHP (módulo CGI* o servidor), un servidor web y un navegador web. Es necesario ejecutar el servidor con una instalación de PHP conectada. Se puede acceder al resultado del programa de PHP con un navegador, viendo la página de PHP a través del servidor. Todo esto se puede ejecutar en su máquina si está experimentado con la programación de PHP.

(*Common Gateway Interface es una tecnología web y un protocolo que define una forma para que un servidor web (serv

Scripts desde la línea de comandos.

Se puede crear un script de PHP y ejecutarlo sin necesidad de un servidor o navegador. Solamente es necesario el analizador de PHP para utilizarlo de esta manera. Este tipo de uso es ideal para scripts que se ejecuten con regularidad empleando cron (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden usarse para tareas simples de procesamiento de texto.

Escribir aplicaciones de escritorio. (aunque suene extraño)

Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisiera utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar PHP-

GTK para escribir dichos programas. También es posible de esta manera escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal. Si está interesado en PHP-GTK, puede visitar su sitio web: <http://gtk.php.net/>

Lo que necesitas para empezar



Hoy en día tenemos muchas opciones para poder ejecutar código PHP. PHP es multiplataforma, por lo tanto te permite operar en varios sistemas operativos. Funciona excelente en LINUX, UNIX, MacOS y Windows. También funciona sin esfuerzos con Apache/MySQL.

PHP es conocido como un lenguaje basado en servidores. Esto es porque el PHP no se ejecuta en tu computadora, sino en la computadora que visita la página (servidor).

Diferentes formas de ejecutar archivos PHP

Hay dos maneras de ejecutar archivos PHP. La manera preferida de ejecutar archivos PHP es dentro de un servidor web como Apache, Nginx o IIS; esto te permite ejecutar scripts de PHP desde tu navegador. ¡Así es como funcionan todos los sitios web en PHP! La otra manera es ejecutar scripts de PHP en la línea de comandos, y esto no requiere que establezcas un servidor web.

Desde luego, si quieres publicar tus páginas de PHP en línea, tendrás que elegir la opción de establecer un servidor web. Por otro lado, ejecutar scripts de PHP desde la línea de comandos es útil para realizar tareas de rutina.

Estas generalmente están configuradas para ejecutarse en segundo plano como trabajos y se ejecutan mediante el comando `php` sin un servidor web. De hecho, hoy en día muchos scripts o aplicaciones en PHP vienen con comandos integrados que se usan para realizar diversas operaciones, como instalar software, exportar e importar entidades de bases de datos, limpiar la memoria caché y más. Quizá hayas oído hablar de Composer; es un administrador de dependencias para PHP y es una de las herramientas más populares desarrolladas para PHP destinado a la línea de comandos.

Ejecuta un archivo PHP en un servidor web

Si quieres ejecutar scripts de PHP desde un servidor web, necesitas configurar con uno de los servidores web que son compatibles con él. Para Windows, el servidor web IIS es uno de los más populares de entre los demás. Por otro lado, Apache y Nginx son servidores web

ampliamente utilizados para el resto de los sistemas operativos.

La buena noticia es que la mayoría de los proveedores de alojamiento tendrán un servidor web con PHP ya configurado para ti cuando inicies sesión en tu nuevo servidor.

Ejecuta un archivo PHP en el navegador para el desarrollo con XAMPP

Si quieres ejecutar un archivo PHP en el navegador en tu propia computadora, necesitarás establecer una pila de desarrollo PHP. Necesitarás al menos PHP, MySQL y un servidor como Apache o Nginx. MySQL se usa para configurar bases de datos con las que tus aplicaciones de PHP puedan trabajar. Desde luego, puedes elegir trabajar con otros motores de bases de datos, pero MySQL es uno de los motores de bases de datos más populares usados junto con PHP.

En vez de descargar todo este software por separado y configurarlo para que funcione en conjunto, te recomiendo que solamente descargues e instales un programa como XAMPP. Este incluirá todo el software necesario y te preparará para ejecutar PHP en muy poco tiempo. Y sí, es compatible con Windows, Linux y MacOS.

XAMPP contiene todo lo que es necesario para compilar tus páginas web localmente. Es cómodo y te permite comenzar a desarrollar con PHP de inmediato.

Una vez que hayas instalado el software XAMPP localmente y que se ejecute con éxito, deberías poder ver la página predeterminada de XAMPP en <https://localhost> en tu navegador.

Ejecuta tu archivo PHP en XAMPP

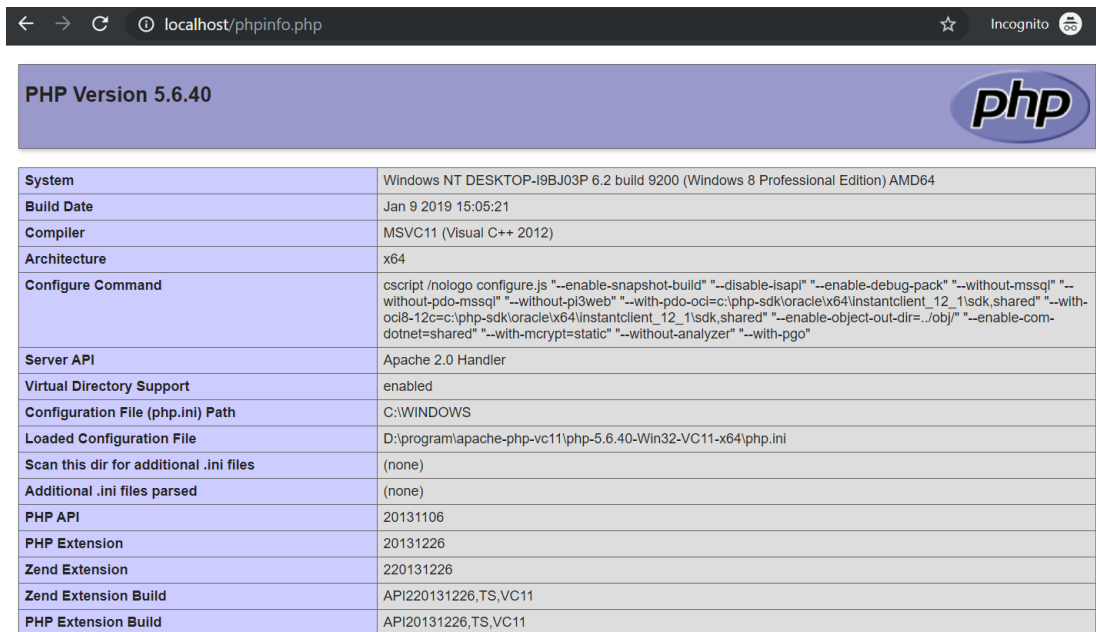
Lo primero que necesitarás saber después de instalar XAMPP es la ubicación en donde colocarás tus archivos PHP. Cuando instalas el software XAMPP, este crea el directorio `htdocs`, que es la raíz de los documentos de tu dominio de servidor web predeterminado: `localhost`. Entonces, si vas a <http://localhost/ejemplo.php>, el servidor intentará encontrar el archivo `ejemplo.php` dentro del directorio `htdocs`.

La ubicación del directorio `htdocs` varía dependiendo del sistema operativo que estés usando. Para Windows, este debería encontrarse en `C:\xampp\htdocs`. De otro modo, este debería encontrarse en `/opt/lampp/htdocs` para los usuarios de Linux. Una vez que hayas encontrado la ubicación del directorio `htdocs`, ¡puedes comenzar a crear archivos PHP de inmediato y ejecutarlos en tu navegador!

`phpinfo()` es una función muy útil que te brinda información sobre tu servidor y la configuración de PHP. Generemos un archivo `phpinfo.php` dentro del directorio `htdocs` con el siguiente contenido:


```
<?php
// Show all information, defaults to INFO_ALL
phpinfo();
?>
```

Ahora sigue adelante y ejecútalo en tu navegador en <http://localhost/phpinfo.php>, y deberás ver una salida como esta:



PHP Version 5.6.40	
System	Windows NT DESKTOP-I9BJ03P 6.2 build 9200 (Windows 8 Professional Edition) AMD64
Build Date	Jan 9 2019 15:05:21
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x64
Configure Command	cmdscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-p3web" "--with-pdo-oci=c:\php-sdk\oracle\x64\instantclient_12_1sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x64\instantclient_12_1sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	D:\program\apache-php-vc11\php-5.6.40-Win32-VC11-x64\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,TS,VC11
PHP Extension Build	API20131226,TS,VC11

Ya ahora sabes lo necesario para empezar! (si te encuentras con algun problema puedes bueskar en internet el problema)

Empezando con PHP

(Todavía no)



Para comenzar a entender PHP debes saber algunos conceptos básico de programación. Sin estos conceptos básicos, vas a creer que es difícil aprender PHP, también a este puntos si te interesa aprender PHP deberías saber estos conceptos, de igual manera aquí un pequeño resumen de algunos conceptos. Si ya tienes estos conceptos básicos puedes ir al directamente próximo capítulo.

Todos los lenguajes de programación comparten algunos elementos básicos que funcionan y se usan de forma diferente en cada lenguaje, pero que cumplen el mismo objetivo. Esos elementos son:

- Tipos de datos
- Variables
- Control de flujo
- Ciclos
- Estructuras de datos
- Funciones

Tipos de datos

Algunos lenguajes de programación tienen más tipos de datos que otros, pero, en general, todos incluyen al menos los siguientes: caracteres (char), cadenas de caracteres (string), enteros (integer), decimales (decimal, float), y booleanos (true y false). Por ejemplo:

```
"Hola Mundo" #Cadenas de caracteres
34 #Enteros
23.45 #Decimales
false #Booleano
```

Variables

Las variables nos permiten almacenar información temporalmente. Por ejemplo, podemos almacenar la cadena "Hola Mundo" en una variable.

```
$miPrimeraVariable = "Hola Mundo";
```

Mas adelante hablaremos mas de Variables

Un mecanismo para tomar de decisiones

Un elemento muy importante en todo lenguaje de programación es cómo tomar una decisión según la información que se tenga en las variables. Por ejemplo:

```
<?php

$puntos = 20;

if ($puntos > 15){
    print("Felicidades, tienes mas de 15 puntos!");
} else {
    print("Muy Regular, tienes menos de 15 puntos!");
}
```

A esto se le llama **control de flujo** porque cuando el programa está corriendo, los **if** deciden qué código se ejecuta.

Un mecanismo para iterar (ciclos)

Por ejemplo, cuando queremos imprimir los números de 0 hasta 9.

```
<?php

$numeroMaximo = 10;

for ($indice = 0; $indice < $numeroMaximo; $indice++){
    print($indice);
}
```

Estructuras de datos (Array)

Las estructuras de datos existen porque programar únicamente con los tipos básicos (char, integer, boolean, etc) sería muy difícil. A veces necesitamos almacenar colecciones de datos e información más estructurada.

Los arreglos son los más conocidos, nos permiten almacenar varios datos en una sola variable.

```
<?php

$arreglo = [1, 2, 3, 4, 5];
print($arreglo[2]); // 3
```

El primer elemento de los arreglos empieza en **0**
Esto quiero decir que el valor de:
\$arreglo[0] es **1**
\$arreglo[1] es **2**
\$arreglo[2] es **3**
\$arreglo[3] es **4**
\$arreglo[4] es **5**

A veces se necesita guardar información más estructurada. Por ejemplo, los datos de una persona.

```
<?php

$person = [];
$person["name"] = "Juan Torres";
$person["gender"] = "masculino";
$person["age"] = 25;
print($person["gender"]); //imprimira masculino
```

Funciones

Las funciones, también llamadas procedimientos o métodos, encapsulan algunas instrucciones y se pueden llamar utilizando el nombre de la función.

```
<?php

/**
 * Que lenguaje rokea!
 *
 * @param string $lenguaje El language de programacion
 *
 * @return void
 */
function rockear($lenguaje)
{

    print($lenguaje . " rocks!");
    $ejemplo = "asdf";
    echo $Ejemplo;
}

rockear("PHP");
```

```
$php rockear.php  
$PHP Rocks! // Esta sera la salida
```

Estos son los elementos (super simplificado) que se necesitan para aprender a programar y empezar a crear tus primeros programas en PHP.

Empezando con PHP (Ahora si)



Cada variable consta del símbolo del dólar(\$), seguido del nombre de la variable. Las variables se utilizan en los scripts de PHP para integrar datos externos en páginas web. En este sentido se puede hablar de valores muy variados que van desde números simples y cadenas de caracteres hasta textos completos o estructuras de documentos HTML.

La administración central de los contenidos tiene lugar, en general, con ayuda de sistemas de bases de datos. Los valores para las variables pueden definirse directamente en el script. Este tipo de clasificación se realiza según el esquema siguiente:

```
<?php
$ejemplo = "Valor";
```

Atencion Los valores para la variables del tipo entero (integer) y flotante (float) no se escriben en comillas.

Ejemplo:

```
<?php

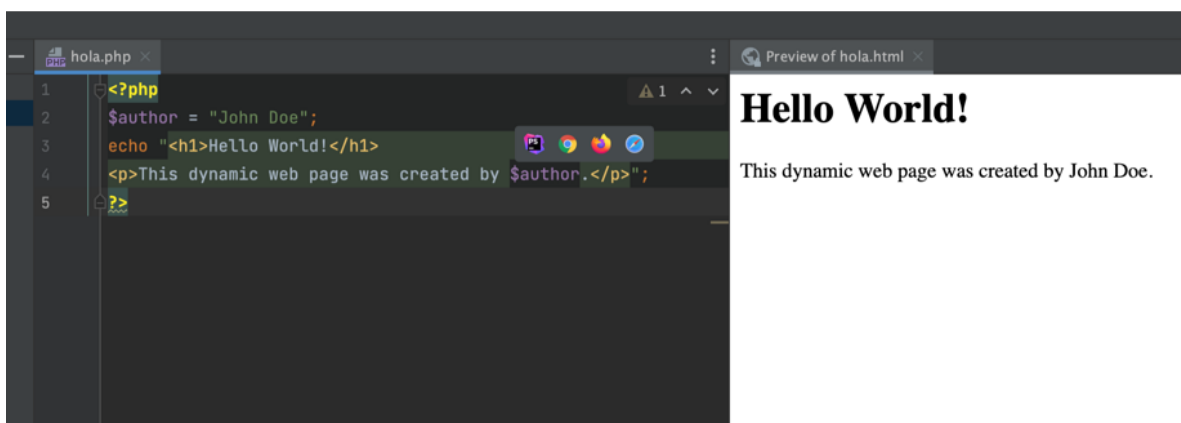
$entero = 24;
$flotante = 2.4;
```

PHP te da la libertad de designar variables según tu creas conveniente, pero surgen aquí ciertas limitaciones:

- Cada variable comienza con el símbolo del dólar (\$).
- El nombre de las variables es una secuencia de caracteres formada por letras, números y guiones (p.ej., \$ejemplo_1).
- Un nombre de variable válido siempre comienza con una letra o con un guion bajo (\$ejemplo1 o \$_ejemplo), pero nunca con un número (incorrecto: \$1ejemplo).
- PHP distingue entre mayúsculas y minúsculas (\$ejemplo ≠ \$Ejemplo).
- Los nombres de las variables no pueden contener espacios o saltos de línea (incorrecto: \$ejemplo 1)
- El usuario no puede hacer uso libre de las secuencias de caracteres reservadas por PHP para otros propósitos (p. ej., \$this)

Veámoslo en un ejemplo:

```
<?php
$author = "John Doe";
echo "<h1>Hello World!</h1>
<p>This dynamic web page was created by $author.</p>";
?>
```



La palabra reservada **echo** resulta de una utilidad mayor que la propia emisión de texto, la cual puede implementarse también sin PHP y tomando HTML como base. La verdadera plusvalía de **echo** está basada en el hecho de que la instrucción permite generar textos de

manera dinámica con ayuda de variables.

La etiqueta de apertura de PHP va seguida de la definición de la variable: en el caso de \$author se utilizaría el valor John Doe. A la hora de ejecutar el script, la variable \$author se sustituye por el valor John Doe cada vez que se haga mención a ella en el entorno del script. El siguiente gráfico muestra cómo se refleja esto en el navegador web.

Comentarios (Documentando el código)



Comentarios de una sola linea

Los comentarios de una sola linea comienzan con `//` o `#`. Cualquier texto que este a la derecha de estos dos elementos sera ignorado por el interprete de PHP. La intencion de los cometarios es documentar el codigo y asi dejarle saber a otro ingeniero que hace el codigo o como funciona.

```
// Esto es un comentario y no es interpretado por PHP  
# Esto tambien es un cometario
```

Otros Ejemplos

```
echo "Hello World!"; // Esto es un comentario, y empieza con "//"
```

Comentarios en multiples lineas

En algunas ocasiones necesitaremos comentar un block de codigo completo. Para esto empezamos como `/*` y terminamos con `*/`

```
/* Esto es un comentario de multiples lineas.  
Aqui puede ir otro comentario.  
Esto tambien es parte del comentario  
*/
```

Docblocks (Bloques de documentacion)

Un Docblock no es más que un bloque de documentación estilo **C** o **C-stlye** para documentar un bloque de código. Comienza con **/**** y tiene un asterisco al principio de cada línea. He aquí un ejemplo:

```
/**  
 * Calcula la suma del cuadrado de un arreglo  
 *  
 * Ciclo que recorre el arreglo, obtiene el valor lo  
 * eleva al cuadrado y se lo suma y retorna el total  
 *  
 * @param array $arr Arreglo de valores  
 * @return int  
 * @throws Exception Si el elemento no es un entero  
 */  
function sumaDeCuadrados($arr)  
{  
  
    $total = 0;  
    foreach ($arr as $val) {  
        if (!is_int($val)) {  
            throw new Exception("Elemento no es un entero !");  
        }  
        $total = $val * $val;  
    }  
  
    return $total;  
}
```

El DocBlocks consiste en tres partes todas tres opcionales, la descripción corta, la descripción larga y los tags. La descripción corta, la cual resumen la funcionalidad, la descripción larga la cual explica de manera específica el funcionamiento el cual puede ser tan largo como desee.

Los tags de los DocBlocks contiene una serie de etiquetas especiales indicados por el símbolo @

Los tags se utilizan para especificar información adicional, tales como los parámetros esperados y su tipo.

No todo el código se puede documentar con DocBlocks, acá están la lista de elemento que deberían ser documentados

- Archivos
- Clases
- Funciones y métodos
- Propiedades de las clases
- Variables Globales
- include()/require()
- define()

Archivos / Files

En los Docblocks se documentan el contenido de un archivo, la mayoría de elementos están documentados mediante la colocación del Docblocks antes del elemento, pero los archivos son una excepción (ya que no puede escribir nada antes de un archivo). A nivel de archivo docblocks se colocan en la primera línea del archivo.

Para este tipo de docblocks es recomendado utilizar las siguientes tags: **@package**, **@subpackage**, **@license**, y **@author**. De los tags hablaremos mas adelante.

Aca un ejemplo de documentación de un file:

```
/**
 * Este Archivo contiene funciones utilizadas dentro del programa
 *
 * @package      MiProyecto
 * @subpackage   Comun
 * @license      http://opensource.org/licenses/gpl-license.php  GNU Public
 License
 * @author       Emmi V. De Oleo <email@gmail.com>
 */
```

Clases

Un DocBlock de clase es colocado siempre antes de la clase, en esta va la descripción de la clase, Generalmente utiliza los tags `@package`, `@subpackage`, and `@author`. Por Ejemplo :

```
/**
 * Un ejemplo de clase
 *
 * Colocaremos la clase vacia para el ejemplo
 *
 * @package    MiProyecto
 * @subpackage Comun
 * @author     Emmi V. <pedropicapiedra@yabadabado.com>
 */
class Example {

}
```

Métodos o funciones

Los métodos y las funciones se documentan igual, para los que no saben que es un método, el método es una función, pero esta, se encuentra dentro de una clase. Funciones y métodos generalmente contiene los tags `@param` y `@return`, estos tags nos indicaran el tipo de datos de entrada y el tipo de datos de salida. Ejemplo :

```
/**
 * La división de dos numeros
 *
 * @param int $a primero numero ingresado
 * @param int $b Segundo numero entrado
 * @return int     Retorna
 */
function division($a, $b){
    if ($b > 0) {
        return $a/$b;
    }
    return 0;
}
```

Variables



¿Qué es una Variable en PHP?

Las variables se utilizan para almacenar datos, como cadenas de texto, números, etc. Los valores de las variables pueden cambiar en el transcurso de un script. Aquí hay algunas cosas importantes que debe saber sobre las variables:

En PHP, no es necesario declarar una variable antes de agregarle un valor. PHP convierte automáticamente la variable al tipo de datos correcto, según su valor. Después de declarar una variable, se puede reutilizar en todo el código. El operador de asignación (=) utilizado para asignar valor a una variable. En PHP, la variable se puede declarar como:

```
#Variable estilo snake
$var_name = 'valor';

#Variable estilo Camelcase
$varName = 'valor';

echo $varName; # retorna: valor

function add($a, $b) {
    return $a + $b;
}

$suma = add(1, 2); // ahora la variable $suma tiene el valor 3
```

Visualizando el valor de Variables



echo y print

El lenguaje PHP nos permite varias formas de imprimir (visualizar) el valor de una variable.

`echo` y `print` son construcciones del lenguaje, no funciones. Esto significa que no requieren paréntesis alrededor del argumento como lo hace una función (aunque uno siempre puede agregar paréntesis alrededor de casi cualquier expresión de PHP y, por lo tanto, `echo ("prueba");` tampoco hará ningún daño). Producen la representación de cadena de una variable, constante o expresión. No se pueden usar para imprimir matrices u objetos.

Asigne la cadena Joel a la variable `$nombre`

```
$nombre = "Joel";
```

Muestra el valor de `$name` usando `echo` & `print`

```
echo $name; #> Joel  
print $name; #> Joel
```

Los paréntesis no son necesarios, pero se pueden utilizar

```
echo($name); #> Joel  
print($name); #> Joel
```

Usando múltiples parámetros (solo eco)

```
echo $name, "Smith"; #> JoelSmith
echo($name, " ", "Smith"); #> Joel Smith
```

`print`, a diferencia de `echo`, es una expresión (devuelve 1) y, por lo tanto, se puede usar en más lugares:

```
print("oye") && print(" ") && print("tu"); #> tu11
```

Lo anterior es equivalente a:

```
print ("hola" && (print (" " && print "tu"))); #> tu11
```

Salida de una vista estructurada de matrices y objetos

`print_r()`: generar matrices y objetos para depurar `print_r` generará un formato legible por humanos de una matriz u objeto. Puede tener una variable que sea una matriz u objeto. Intentar generarlo con un eco generará el error: Aviso: conversión de matriz a cadena. En su lugar, puede usar la función `print_r` para volcar un formato legible por humanos de esta variable. Puede pasar `true` como segundo parámetro para devolver el contenido como una cadena.

```
$myobject = new stdClass();
$myobject->myvalue = 'Hello World';
$myarray = [ "Hello", "World" ];
$mystring = "Hello World";
$myint = 42;

// Usando print_r podemos ver los datos que contiene la matriz.
print_r($myobject);
print_r($myarray);
print_r($mystring);
print_r($myint);
//Esto genera lo siguiente:
```



```
stdClass Object
(
    [myvalue] => Hello World
)
Array
(
    [0] => Hello
    [1] => World )
Hello World
42
```

var_dump(): genera información de depuración legible por humanos sobre el contenido de los argumentos, incluido su tipo y valor. La salida es más detallada en comparación con `print_r` porque también genera el tipo de variable junto con su valor y otra información como ID de objeto, tamaños de matriz, longitudes de cadena, marcadores de referencia, etc. Puede usar `var_dump` para generar una versión más detallada para la depuración.

```
var_dump($myobject, $myarray, $mystring, $myint);
```

La salida es más detallada:

```
object(stdClass)#12 (1) { ["myvalue"]=>
    string(11) "Hello World"
}
array(2) {
    [0]=>
    string(5) "Hello"
    [1]=>
    string(5) "World"
}
string(11) "Hello World"
int(42)
```

Concatenación de cadenas con **eco**

Puede usar la concatenación para unir cadenas "de extremo a extremo" mientras las genera (con `echo` o impresión, por ejemplo). Puede concatenar variables usando un `.` (punto).

```
// String variable
$name = 'Joel';
// Concatenate multiple strings (3 in this example) into one and echo it
once done. // 1. ↓ 2. ↓ 3. ↓ - Three Individual string items
echo '<p>Hello ' . $name . ', Nice to see you.</p>';
// ↑ ↑ - Concatenation Operators
#> "<p>Hello Joel, Nice to see you.</p>"
```

Genere una matriz multidimensional con índice y valor e imprima en una tabla

```
$array = [ 0 => [
    [id] => 13
    [category_id] => 7
    [name] => Leaving Of Liverpool
    [description] => Leaving Of Liverpool
    [price] => 1.00
    [virtual] => 1
    [active] => 1
    [sort_order] => 13
    [created] => 2007-06-24 14:08:03
    [modified] => 2007-06-24 14:08:03
    [image] => NONE
],
  1 => [
    [id] => 16
    [category_id] => 7
    [name] => Yellow Submarine
    [description] => Yellow Submarine
    [price] => 1.00
    [virtual] => 1
    [active] => 1
    [sort_order] => 16
    [created] => 2007-06-24 14:10:02
    [modified] => 2007-06-24 14:10:02
    [image] => NONE
  ]
];
```

```
<table>
<?php
foreach ($products as $key => $value) {
    foreach ($value as $k => $v) {
        echo "<tr>";
        echo "<td>$k</td>"; // Get index. echo "<td>$v</td>"; // Get
value. echo "</tr>";
    }
}
?>
</table>
```

Control de flujo



Tipos de Datos



Cadena de Texto



Operadores



Operadores Lógicos



Manejo de cadenas



Condicionales



Iteraciones



Funciones



Classes



Manejo de Excepciones



Composer



Laravel-Zero



Nuestro primer script



Scripts Avanzados



Referencias Bibliográficas



<https://www.php.net> https://es.wikipedia.org/wiki/PHP#Caracter%C3%ADsticas_de_PHP
<https://www.web-matter.com.ar/servicios-desarrollo-programacion-php-7.asp>
<https://blog.makeitreal.camp/conceptos-basicos-de-programacion/>
<https://diego.com.es/programacion-orientada-a-objetos-en-php>

Biografia



Elminson De Oleo Baez es un dominicano desarrollador Senior de software radicado en la ciudad de New York. Forma parte del Grupo PHP Dominicana y de la Fundación Código Libre Dominicana Fundada por Antonio Perpinan. Ha trabajado en proyectos de software en la República Dominicana para empresas como Claro, Seguros Universal, Seguros Banreservas, Banco del Progreso, Cardnet, Ferretería Americana, Centros Tecnológicos Comunitarios, etc.

Ha trabajado para empresas en Estados Unidos como: WWE, Choice Money Transfer, One Rockwell y Refersion en el desarrollo software y APIs de alto tráfico.

Elminson ha sido seleccionado 2 veces como finalista de los premios PHP Innovation Award (2014 y 2018). Fundador y creador de 2 proyectos www.logjs.net y de www.rlist.net y participante activo de varios proyectos en la República Dominicana.

Elminson también ha ganado varios premios en República Dominicana y a nivel internacional dentro de los cuales se pueden destacar: Primer lugar en la competencia HACK IT, The Most Innovative App and Most Disruptive: INTERNATIONAL SPACE APP CHALLENGE de la NASA, Primer lugar RANDOM HACKS OF KINDNESS de Canada, Finalista App Genius Orange Dominicana. También ha aportado en comunidades de software libre en GitHub, StackOverflow entre otros.

Actualmente es mentor en Lambda School en la ciudad de New York y de PHP Dominicana. Se desempeña en la actualidad como asesor para varias empresas y como Senior Software Engineer en Refersion en la ciudad de New York.