

## PHP

### VIDEO 2, INSTALAR LARAGON

Página para descarga: [laragon.org](http://laragon.org)

Los iconos para iniciar al prender el computador y el de readme se desmarcan y se instala

No instale laragon porque toca pagar así que trabaje con XAMMP

Todo se guarda en htdoc de XAMMP en la carpeta que cree PHP, y trabajamos con VISUAL STUDIO CODE

Para iniciar un programa php, se debe crear con extensión php: prueba.php

Lo primero es los símbolos que encierran todo programa de php:

```
<?php  
... programa php  
...  
>
```

Toda linea de código se debe terminar con (;) punto y coma

Si vamos a trabajar solo con código php no es obligatorio la etiqueta de cierre

```
<?php  
... programa;
```

Si vamos a trabajar PHP en conjunto con Html si es obligatorio las etiquetas de apertura y cierre de código para indicar donde inicia y donde acaba el código PHP.

Ejemplo

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>
```

```
</head>

<body>

    <h1>Hola Mundo</h1>

    <?php

        echo "Hola mundo desde Php";

    ?>

</body>

</html>
```

### Video 3

Donde se guarda la información y los proyectos como trabaje con XAMMP todo se guarda en la capeta XAMMP htdoc

### Video 4

#### Comentarios

```
<?php

    echo "Hola mundo";

    //este es un comentario de linea

    /*

        Un comentario mas largo

        para bloque de comentario

    */

# comentario con otra forma para linea
```

```
?>
```

Los comentarios se utilizan para explicar el código y sea más fácil realizar cambio o revisar fallas

## Video 5 Tipos de datos

### Tipos de datos en PHP

1. **Booleanos (boolean)**: TRUE (1), FALSE (0)
2. **Enteros (integer)**: 10,7, -20, -9
3. **Flotantes (double)**: 7.90, 0.20, -12.75, 3.1416
4. **Cadenas (string)**: "Cadenas de texto", 'Cadena de texto 2'



Para saber de que tipo es un dato se utiliza

```
var_dump(dato);  
echo -100;// muestra -100  
var_dump(-100); //muestra int(-100) int es la sigla de entero  
var_dump('hola mundo');//string(10) "hola mundo", nos dice que es un string y muestra la  
cantidad de caracteres
```

## VIDEO 6

### Variables

#### ¿Qué es una variable?

Una variable es un espacio de memoria (con un nombre) reservado para almacenar un valor, correspondiente a un tipo de dato (texto, numérico, booleano etc.)

Las variables en php se declaran con el símbolo \$

## Reglas para definir una variable en PHP

1. En PHP las variables se representan con un **signo de dólar** seguido por el nombre de la variable.
2. Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado (guion bajo), seguido de cualquier número de letras, números y caracteres de subrayado. **No puede empezar con un numero.**
3. El nombre de la variable es **sensible a minúsculas y mayúsculas**. Se puede usar el estándar de nomenclatura **Camel Case**.

```
$nombre='Carlos';  
$Nombre='Juan';  
  
echo $nombre;  
echo $Nombre;// son variables diferentes
```

## Estándares de nomenclatura

Es un conjunto de normas (un estándar de nomenclatura) para un lenguaje de programación que se recomienda usar como buenas prácticas para facilitar la lectura del código y este sea fácilmente más entendible y mantenible.

1. **Camel Case:** El nombre viene porque se asemeja a las dos jorobas de un camello, y se puede dividir en dos tipos:

– **Upper Camel Case:** Cuando la primera letra de cada una de las palabras es mayúscula. También denominado **Pascal Case**. Ejemplo: EjemploDeNomenclatura.

– **Lower Camel Case:** Igual que la anterior con la excepción de que la primera letra es minúscula. Ejemplo: ejemploDeNomenclatura.

**2. Upper Case:** Todas las letras del identificador se encuentran en mayúsculas, por ejemplo: EJEMPLDENOMENCLATURA. Es usado, por ejemplo, para las constantes definidas en PHP.

**3. Snake Case:** Cuando cada una de las palabras, se separa por un guión bajo (\_). Es común en los nombres de variables y funciones de lenguajes como C, aunque también Ruby y Python lo han adoptado. Como el Camel Case existen variedades, como todas las letras en mayúsculas, denominado **SCREAMING\_SNAKE\_CASE**, que se utiliza para definir constantes.

Ejemplo: ejemplo\_de\_nomenclatura

## Variables predefinidas

1. **\$GLOBALS** – Hace referencia a todas las variables disponibles en el ámbito global
2. **\$\_SERVER** – Información del entorno del servidor y de ejecución
3. **\$\_GET** – Variables HTTP GET
4. **\$\_POST** – Variables POST de HTTP
5. **\$\_FILES** – Variables de subida de ficheros HTTP
6. **\$\_REQUEST** – Variables HTTP Request
7. **\$\_SESSION** – Variables de sesión
8. **\$\_ENV** – Variables de entorno
9. **\$\_COOKIE** – Cookies HTTP
10. **\$php\_errormsg** – El mensaje de error anterior
11. **\$\_HTTP\_RAW\_POST\_DATA** – Datos POST sin tratar
12. **\$http\_response\_header** – Encabezados de respuesta HTTP
13. **\$argc** – El número de argumentos pasados a un script
14. **\$argv** – Array de argumentos pasados a un script
15. **\$this** – Palabra reservada utilizada en programación orientada a objetos

Estas variables predefinidas no se pueden utilizar.

En este curso trabajaremos con snake case

Ejemplo

```
$nombre_completo = 'Fernando Ortiz';
```

Las variables en php pueden cambiar de tipo de dato esto porque no es un lenguaje tipado como java.

## VIDEO 7

### CONSTANTES

Es un espacio de memoria donde se almacena un valor, cuando se define ese valor no puede volver a cambiarse en la ejecución del programa

### Reglas para definir una constante en PHP

1. El nombre de una constante sigue las mismas reglas que cualquier variable de PHP. Un nombre de constante válido empieza por una letra o guion bajo, seguido por cualquier número de letras, números o guiones bajos.
2. Por defecto, una constante distingue mayúsculas y minúsculas.
3. Por convención, los identificadores de constantes siempre se declaran en mayúsculas.

Las constantes no llevan el símbolo de dólar (\$) antes del nombre

Sintaxis

Forma antigua: define("Pi", 3.141623);

echo Pi;

Si se intenta modificar

```
<?php
```

```
define("Pi", 3.141623);
```

```
define("pi", 3.1415);
```

```
echo Pi;
```

El resultado es un error

**Warning:** Constant Pi already defined in C:\xampp\htdocs\PHP\constantes.php on line 3

Nos dice que la constante ya está definida

```
//sintaxis nueva
```

```
echo "<br>";//salto de linea.
```

```
const CC = 88205852;
```

```
echo CC;
```

# Constantes predefinidas

Varias constantes "mágicas" de PHP	
Nombre	Descripción
<code>__LINE__</code>	El número de línea actual en el fichero.
<code>__FILE__</code>	Ruta completa y nombre del fichero con enlaces simbólicos resueltos. Si se usa dentro de un include, devolverá el nombre del fichero incluido.
<code>__DIR__</code>	Directorio del fichero. Si se utiliza dentro de un include, devolverá el directorio del fichero incluido. Esta constante es igual que <code>dirname(__FILE__)</code> . El nombre del directorio no lleva la barra final a no ser que esté en el directorio root.
<code>__FUNCTION__</code>	Nombre de la función.
<code>__CLASS__</code>	Nombre de la clase. El nombre de la clase incluye el namespace declarado en (p.ej. <code>Foo\Bar</code> ). Tenga en cuenta que a partir de PHP 5.4 <code>__CLASS__</code> también funciona con traits. Cuando es usado en un método trait, <code>__CLASS__</code> es el nombre de la clase del trait que está siendo utilizado.
<code>__TRAIT__</code>	El nombre del trait. El nombre del trait incluye el espacio de nombres en el que fue declarado (p.ej. <code>Foo\Bar</code> ).
<code>__METHOD__</code>	Nombre del método de la clase.
<code>__NAMESPACE__</code>	Nombre del espacio de nombres actual.
<code>className::class</code>	El nombre de clase completamente cualificado. Véase también <code>::class</code> .

Estos nombres no se pueden utilizar

A partir de php 7 se pueden asignar arrays a una constante

```
define('ANIMALES', array(  
    'perro',  
    'gato',  
    'pájaro'  
));  
  
echo ANIMALES[1];
```

## VIDEO 8

### Arrays en PHP

Un array en PHP es un **tipo especial de datos** que representa los llamados **mapas ordenados de datos**. Un mapa es un tipo de datos que asocia **valores con claves**, también llamado '**array asociativo**'.

Los valores que almacenamos en un array **no tienen por qué ser del mismo tipo** como ocurre en otros lenguajes de programación.

La **posibilidad de utilizar** como valores de un array otros arrays, árboles, les permite la **definición de arrays multidimensionales**.



PHP soporta tanto **arrays escalares** (índice numérico), **arrays asociativos** (índice por clave) y **arrays multidimensionales**.

Para acceder a los elementos de un array se utilizan los corchetes **[]**, dentro de los cuales se indicará un **índice o clave** de localización.

A diferencia de otros lenguajes en PHP no hace falta definir el array antes de utilizarlo. Cuando se definen elementos de un array, PHP reconoce automáticamente que se trata de un array sin necesidad de declaración previa.

## Tipos de Arrays

### Escalares

Los arrays escalares son aquellos en los que para acceder a los elementos utilizamos un índice que representa la posición del valor dentro del array comenzando desde el índice 0.

### Asociativos

Para acceder a los elementos del array utilizamos la clave asociada con el elemento, donde este toma un cierto número de parejas utilizando la sintaxis clave => valor como argumentos.

### Multidimensionales

Un array multidimensional es aquel cuyos valores son otros arrays. Para acceder a sus elementos se tienen que indicar los índices de cada una de sus dimensiones, utilizando tantos pares de corchetes como dimensiones se definan en el array: [][] -> 2 dimensiones, [][] -> 3 dimensiones, etc...

## Definir un Array

En PHP podemos definir arrays utilizando el constructor del lenguaje **array()** o utilizando la notación de **corchetes []** (a partir de PHP 5.4) también llamada sintaxis corta.

Si a la hora de definir el array no se **indican los valores**, estaremos creando un **array vacío**.

```
$estudiantes = array("Carlos", "Jose", "Vane", "Carla");//array escalar  
echo $estudiantes[3];//acceder a un dato del array  
  
$estudiantes[3]="Josefina";//modificar un dato
```

el array escalar no tiene llave solo el valor, la key o llave la asigna automáticamente php empezando desde cero (0) para el primer valor o primer dato del array.

Key	valor
-----	-------

0	carlos
1	Jose

2	Vane
3	Josefina

## Notación ( ) (corchetes a partir de PHP 5.4)

```
[Valor1, Valor2, Valor3, ...];
["Clave1"=>"Valor1", "Clave2"=>"Valor2" ...];
[];
```

```
$numeros = [10, 20, 30, 40, 50];
$nombres = ["Nombre1"=>"Luis", "Nombre2"=>"Juan"];
$vacio = [];
```

Ejemplo:

```
$estudiantes2 = ["Carlos", "Jose", "Vane", "Carla"];
$estudiantes2 = ["Carlos", "Jose", "Vane", "Carla", 5, 2.6];//pueden tener diferentes tipos de
datos

echo var_dump($estudiantes2[5]); //nos muestra float(2.6)

echo var_dump($estudiantes2[4]); //nos muestra int(5)
echo var_dump($estudiantes2[3]); // nos muestra string(5) "Carla"
```

## Asignar valores

Para asignar valores a los elementos de un array, utilizaremos la sintaxis:

```
$NombreArray[IndiceClave]=ValorAsignado;
```

```
$elementos[11]=50;
$elementos['Precio']=9.99;
```

PHP admite también otra posibilidad que no admiten otros lenguajes: **se pueden asignar valores a un array sin especificar el índice**, ya que PHP los coloca automáticamente al final de los elementos existentes.

```
$NombreArray[] = ValorAsignado;
```

```
$elementos[] = 100.25;
$elementos[] = 50;
```

## array vacío

```
$vacio = [];//array vacio
```

## ARRAY ASOCIATIVO

Se crean con clave y valor para cada dato, la clave y el valor pueden ser de cualquier tipo y las parejas claves y valor se asignan mediante ( $=>$ ) dentro del array

Ejemplo:

```
# ARRAY ASOCIATIVO (cada valor tiene asociada una clave)
```

```
//$tutor = ["nombre"=>"Juan", "apellido"=>"Perez", "Edad"=>28];
```

```
//normalmente los array los muestran asi:
```

```
$tutor = [  
    "nombre"=>"Juan",  
    "apellido"=>"Perez",  
    "Edad"=>28  
];
```

```
//para mostrar los datos se utiliza la clave
```

```
echo $tutor["nombre"];//muestra Juan  
echo "<br>";  
echo $tutor["Edad"];//muestra 28
```

```
//modificar un valor, se hace mediante la clave
```

```
$tutor["Edad"]=19;  
echo "<br>";  
echo $tutor["Edad"];//ahora muestra 19
```

## ARRAY MULTIDIMENSIONALES

```
//Array que llevan arrays dentro de ellos
```

```
$tutor_2=[  
    "nombre"=>"Vanessa",  
    "apellido"=>"Pinto",  
    "edad"=>21,
```

```
"cursos"=>["PHP","Python","CSS"]// este es un array escalar dentro de un array asociativo  
];
```

Acceder a los datos contenidos en la clave cursos:

```
echo $tutor_2["cursos"][0];  
  
echo "<br>";  
  
echo $tutor_2["cursos"][1];  
  
echo "<br>";  
  
echo $tutor_2["cursos"][2];  
  
echo "<br>";
```

## OTRO EJEMPLO

```
$tutor_2=[  
"nombre"=>"Vanessa",  
"apellido"=>"Pinto",  
"edad"=>21,  
"horas"=>["PHP"=>40,"Python"=>22,"CSS"=>18]//ARRAY ASOCIATIVO  
];  
  
//mostrar datos  
  
echo $tutor_2["horas"]["PHP"];  
  
echo "<br>";  
  
echo $tutor_2["horas"]["Python"];  
  
echo "<br>";  
  
echo $tutor_2["horas"]["CSS"];  
  
echo "<br>";
```

## Modificar un dato de un array multidimensional

//Modificar VALOR dentro del array multidimensional ¿Se podrán cambiar las key?

```

$tutor_2["horas"]["PHP"]=50;
echo $tutor_2["horas"]["PHP"];

//agregar elemento del array multidimensional
$tutor_2["pais"]="El Salvador";//primero la nueva clave y después el nuevo valor

echo $tutor_2["pais"];echo "<br>";

```

## Contar los elementos de un array: count()

Para contar los elementos que componen un array utilizamos la función `count()` indicando como argumento la variable de tipo array.

```

<?php
$numeros=[10,20,30,40];
echo count($numeros); // 4
?>

```

Si deseamos contar todos los elementos de un **array multidimensional**, tendremos que indicar la constante `COUNT_RECURSIVE` como segundo argumento de la función `count()`.

```

<?php
$matriz_numeros=[[2,4,6,8],[5,10,15,20],[10,20,30,40]];
// 15 (ya que se tienen en cuenta los 3 elementos de la primera dimensión
os echo count($matriz_numeros,COUNT_RECURSIVE);
?>

```

```

echo count($estudiantes)."<br>";
echo count($tutor)."<br>";
echo count($tutor_2)."<br>";
echo count($tutor_2,COUNT_RECURSIVE)."<br>"; //COUNT_RECURSIVE Realiza el conteo
del array multidimensional

```

Array ( [nombre] => Vanessa [apellido] => Pinto [edad] => 21 [horas] => Array ( [PHP] => 50
[Python] => 22 [CSS] => 18 [HTML] => 80 ) [pais] => El Salvador )//9 elementos

Video 9;

## CONCATENACIÓN PHP

```
$nombre = "Fernando ";  
$pais = "Colombia";  
$resultado = $nombre.$pais;  
echo $resultado;  
el punto (.) es el símbolo para concatenar en php
```

## Interpolación de variables

Mostrar el valor de una variable dentro de un String, solo se puede hacer en String que estén entre comillas dobles

```
echo "Mi nombre es: $nombre, mi pais de residencia es: $pais";
```

video 10:

## Operadores aritméticos

### Operadores aritméticos

Nombre	Símbolo en PHP
Suma	+
Resta	-
División	/
Resto de (modulo o residuo)	%
Multiplicación	*
Exponenciación	**

La jerarquía de operadores determina el orden en el que se resuelven las expresiones cuando se involucran operaciones aritméticas como la suma, resta, multiplicación, división, potencia, módulo de la división.

Jerarquía	Operador	Comentario
1 ↓	( )	Se ejecuta primero
2	**	Después de ( )
3	* , /	Después de los anteriores
4	+ , -	Después de los anteriores
5	%	Después de los anteriores

Cuando dos operadores tienen el mismo nivel de prioridad, dentro de una expresión se evalúan de izquierda a derecha.

```
$numero_1=4;
```

```
$numero_2=3;
```

```
echo $numero_1+$numero_2;//muestra 7
```

```
echo $numero_1+10;//muestra 14
```

6:23

```
echo ($numero_1+10)."  
"; //suma  
echo ($numero_1-$numero_2)."  
"; //resta  
echo ($numero_1/2)."  
"; //division  
echo ($numero_1**10)."  
"; //potencia  
echo ($numero_2**$numero_1)."  
";  
echo ($numero_1%$numero_2)."  
"; //modulo
```

## Video 11

### Operadores de asignación

## Operadores de asignación

Nombre	Símbolo en PHP	Ejemplo
Asignar	=	\$a=10;
Sumar y asignar	+=	\$a+=5
Restar y asignar	-=	\$a-=5
Multiplicar y asignar	*=	\$a*=5
Dividir y asignar	/=	\$a/=5
Concatenar y asignar	.=	\$a.=" es el valor de A"

Los operadores de asignación nos permiten realizar operaciones de una forma mas corta por ejemplo:  
**\$a+=5** es lo mismo que **\$a=\$a+5**

### Asignacion

```
$numero = 5;
```

### Sumar y asignar

```
$numero+=5;
```

```
$numero = 5;
```

```
$numero_2=3;
```

```
$numero_2+=$numero; // $numero_2 = $numero_2 + $numero
```

```
$numero+=4; // $numero = 9
```

```
echo $numero."<br>";
```

```
$numero-=4; // $numero = 5
```

```
echo $numero."<br>";
```

```
$numero*=4; // $numero = 20
```

```
echo $numero."<br>";
```

```
$numero/=5; // $numero = 4
```

```
echo $numero."<br>";
```

### //concatenar y asignar

```
$texto = "Texto de Prueba ";
```

```
$texto .= 2;
```

```
echo $texto;
```

resultado

Texto de Prueba 2

```
$texto .= " otro texto";
```

```
echo $texto."<br>";
```

→Texto de Prueba 2 otro texto

Video 12

asignación por Referencia

ambas variables terminan apuntando a los mismos datos y nada es copiado en ninguna parte.

Ejemplo:

```
$texto = "Colombia";
```

```
$variable_1 = $texto;//asignación normal
```

```
$variable_2 = &$texto;//asignación por referencia
```

```
echo "la variable texto contiene: ".$texto."<br>";
```

```
echo "la variable variable_1 contiene: ".$variable_1."<br>";
```

```
echo "la variable variable_2 contiene: ".$variable_2."<br>";
```

la variable texto contiene: Colombia

la variable variable\_1 contiene: Colombia

la variable variable\_2 contiene: Colombia

```
$texto = "Venezuela";
```

```
echo "la variable texto contiene: ".$texto."<br>";
```

```
echo "la variable variable_1 contiene: ".$variable_1."<br>";
```

```
echo "la variable variable_2 contiene: ".$variable_2."<br>";
```

la variable texto contiene: Venezuela  
la variable variable\_1 contiene: Colombia  
la variable variable\_2 contiene: Venezuela

explicación: la variable\_2 esta referenciada a la variable texto cuando texto cambia ella tambien

video 13

Operadores de comparación

## Operadores de comparación

Los operadores de comparación comparan dos expresiones y devuelven un valor Booleano que puede ser TRUE (1) o FALSE (0).

Ejemplo	Símbolo (Nombre)	Resultado
1 == "1"	== (Igual)	TRUE
10 === "10"	=== (Idéntico)	FALSE
7 != "7"	!= (Diferente)	FALSE
21 <> "21"	<> (Diferente)	FALSE
19 !== "19"	!== (No idéntico)	TRUE
7 < 4	< (Menor que)	FALSE
7 > 4	> (Mayor que)	TRUE
2 <= 2	<= (Menor o igual que)	TRUE
3 >= 7	>= (Mayor o igual que)	FALSE

```
var_dump(1=="1");//true
```

```
var_dump(2=="1");//bool(false)
```

var\_dump(1==="1");//bool(false) porque el numero 1 pero uno es un numero y el otro un string el comparador idéntico tiene en cuenta el tipo de dato.

- Utilizando variables

```
$variable=2;
```

```
var_dump($variable==="1");//bool(false)
```

```
$variable=7;
```

var\_dump(\$variable!="7");//(diferente) =>> bool(false) porque no tiene en cuenta el tipo de dato.

```
<?php
```

```
$variable=9;  
var_dump($variable!="7");//bool(true)  
  
$variable="hola";  
var_dump($variable!="Hola");//bool(true) pues son diferente por la h minúscula
```

ahora:

```
$variable="Hola";  
var_dump($variable!="Hola");//bool(false) pues son iguales
```

No idéntico

```
$variable=2;  
var_dump($variable!="2");//bool(true) son diferentes pues uno es String el otro entero
```

otros ejemplos de comparaciones igual que en todos los lenguajes

Video 14 Operadores Lógicos

## Operadores lógicos

Los operadores lógicos permiten combinar expresiones simples en expresiones más complejas, estas expresiones se evalúan y devuelven un valor booleano.

Símbolo	Nombre
and	And (y)
or	Or (o)
!	Not (no)
&&	And (y)
	Or (o)

```
$valor_1=7;
```

```
$valor_2=2;
```

var\_dump(\$valor\_1==7 && \$valor\_1<\$valor\_2);&& es equivalente a escribir (and)

el resultado es (false) pues la primera condición se cumple, pero la segunda no y pues la Lógica And es que ambas se deben cumplir para ser verdadera

Valor 1	Comparador lógico	Valor 2	Resultado
0	And	0	False
0	&&	1	False
1	&&	0	false
1	&&	1	true

Valor 1	Comparador lógico	Valor 2	Resultado
0	Or	0	False
0		1	True
1		0	True
1		1	true

## Tabla de operador NOT

EXPRESIÓN	RESULTADO
!FALSE	TRUE
!TRUE	FALSE

**Conclusión:** Devuelve el valor contrario de la expresión

```
$valor_1=7;
```

```
$valor_2=2;
```

var\_dump(!(\$valor\_1==\$valor\_2));//bool(true)

son valores diferentes pero como esta con el operador Not (!) da verdadero

```

$valor_1=7;
$valor_2=2;

$resultado=!(valor_1==$valor_2);
var_dump($resultado);//bool(true)

```

video 15

OPERADORES DE INCREMENTO Y DECREMENTO

## Operadores de incremento y decremento

Símbolo	Nombre
++	Incremento
--	Decremento

### EJEMPLOS

++\$variable	Pre-Incremento
\$variable++	Post-Incremento
--\$variable	Pre-Decremento
\$variable--	Post-Decremento

```

$numero=10;

//pre incremento
echo ++$numero;//primero incrementa y luego muestra en pantalla osea
// que muestra el numero 11

Post incremento
echo $numero++."<br>";//primero muestra en panta y luego incrementa
echo $numero;//primero muestra 10 y despues 11

```

10

11

Pre decremtno

```
echo --$numero."<br>";//pasa de 11 a 10
```

post decremento

```
echo $numero--."<br>";//muestra el 10 y despues decrementa
```

```
echo $numero."<br>";//muestra el 9
```

cargamos el dato a otra variable

```
$resultado = ++$numero;
```

```
echo $resultado;//10
```

```
$otro = $resultado++;//de esta forma carga en otro el valor de resultado antes
```

```
//de incrementar resultado
```

```
echo $otro."<br>";//muestra 10
```

```
echo $resultado;//muestra 11
```

video 16

Estructuras

If

#### ESTRUCTURA CONDICIONAL SIMPLE (IF)

Permite la ejecución fragmentos de código, la expresión es evaluada a su valor booleano. Si la expresión se evalúa como **TRUE**, PHP ejecutará la sentencia y si se evalúa como **FALSE** la ignorará.

<pre>if(expresión){     Código a ejecutar }</pre>	<pre>if(expresión):     Código a ejecutar endif;</pre>
---	--

## Ejemplos

```
if(1>=0){  
    echo "Expresión Verdadera";  
  
}
```

- Sintaxis alternativa

```
• if(1>=0):  
•     echo "Expresión Verdadera";  
•  
•     endif;
```

php incrustado en html

```
<body>  
  
<?php if(9>=8):?>  
  
<h2>Expresión verdadera</h2>  
  
<h2>otra linea</h2>  
  
<h3>todo el bloque</h3>  
  
<?php endif ?>
```

```
</body>
```

Cuando la expresión es verdadera se ejecuta el bloque de líneas hasta el endif si no es verdadera no ejecuta ninguna linea que este dentro del if y el endif

Tambien se puede utilizar la llaves cuando se incrusta php en html pero no es tan legible

```
<?php if(1>=8){?>  
  
<h2>Expresión verdadera</h2>  
  
<h2>otra linea</h2>  
  
<h3>todo el bloque</h3>  
  
<?php } ?>
```

Video 17

## EJERCICIOS (IF)

1 - Realizar un programa en donde se pide la edad del usuario; si es mayor de edad debe aparecer un mensaje indicándolo.

2 - En un almacén se hace un 20% de descuento a los clientes cuya compra supere los \$100 ¿Cuál será la cantidad que pagara una persona por su compra?

```
$edad=16;
```

```
if($edad>=18){  
    echo "Es mayor de edad";  
}  
  
if($edad>=18){  
    echo "Es mayor de edad";  
}*/  
  
if($edad>=18): echo "Es mayor de edad"; endif;  
  
  
  
echo "<br>";  
echo "Segundo ejercicio <br>";  
$compra=200;  
/*  
if($compra>100){  
    $compra-=$compra*20/100;  
    echo "tiene descuento<br>";  
}*/  
if($compra>100):$compra-=$compra*20/100;endif;
```

```
echo "Total a pagar es: ".$compra;
```

## VIDEO 18 IF ELSE

### ESTRUCTURA CONDICIONAL DOBLE (IF - ELSE)

Con frecuencia se desea ejecutar una sentencia si una determinada condición se cumple y una sentencia diferente si la condición no se cumple. Esto es para lo que sirve **else**. El **else** extiende una sentencia **if** para ejecutar una sentencia en caso que la expresión en la sentencia **if** se evalúe como **FALSE**.

```
if(expresión){  
    Código a ejecutar si la expresión  
    es verdadera  
}else{  
    Código a ejecutar si la expresión  
    es falsa  
}
```

```
if(expresión):  
    Código a ejecutar si la expresión  
    es verdadera  
else:  
    Código a ejecutar si la expresión  
    es falsa  
endif;
```

Ejemplo:

```
if(9>7){  
    echo "condición verdadera";  
}  
else{  
    echo "condición falsa";  
}
```

Ejemplo:

```
if(5<5): echo "condición verdadera";else:echo "condición falsa";endif;
```

video 19

ejemplos if else

## EJERCICIOS (IF - ELSE)

1 - Calcular el total que una persona debe pagar en una llantera, el precio de cada llanta es de \$800 si se compran menos de 5 llantas y de \$700 si se compran 5 o mas.

2- Determinar si un alumno aprueba o repreuba un curso, sabiendo que aprobara si su promedio de tres calificaciones es mayor o igual a 7.0; repreuba en caso contrario

Solución 1:

```
$llantas_cantidad=6;  
$total_a_pagar=0;  
  
if($llantas_cantidad<5){  
    $total_a_pagar= 800*$llantas_cantidad;  
}  
else{  
    $total_a_pagar=700*$llantas_cantidad;  
}  
echo "total a pagara: ".$total_a_pagar;
```

solución 2:

```
$nota_1=7;  
$nota_2=8;  
$nota_3=5;  
$promedio=($nota_1+$nota_2+$nota_3)/3;  
if($promedio>=7){  
    echo "Aprobo el curso con calificacion de: ".$promedio;  
}  
else{  
    echo "Reprobo el curso con calificacion de: ".$promedio;  
}
```

Video 20 operador ternario

## OPERADOR TERNARIO

El operador ternario puede considerarse como una instrucción if de una sola línea. Se compone de tres partes. **El operador , y dos resultados**. La sintaxis es la siguiente:

<operator> ? <true value> : <false value>  
I

La instrucción (echo) no se puede utilizar pues marca error, se pueden hacer operaciones

(0>5)? \$boolean=true: \$boolean=false;

var\_dump(\$boolean);

(8<7)? \$total=10\*5:\$total=10\*6;

echo \$total;//muestra 60, la operación que se realiza si la condición es falsa.

\$edad=15;

\$resultado = (\$edad>18)? "Mayor de Edad":"Menor de edad";

echo \$resultado;

el operador ternario nos permite hacer condicionales en una sola linea, para operaciones que sean sencillas.

Video 21 Ejercicios operador ternario

## EJERCICIO (OPERADOR TERNARIO)

1 - Hacer un programa que calcule el total a pagar por la compra de camisas. Si se compran tres camisas o mas se aplica un descuento del 20% sobre el total de la compra y si son menos de tres camisas un descuento del 10%.

\$valor=50;

\$camisas=6;

\$total=(\$camisas<3)? (\$valor-(\$valor\*0.1))\*\$camisas: (\$valor-(\$valor\*0.2))\*\$camisas;

```
echo "Total a pagar es: ".$total;
```

Video 22 Estructura condicional (if elseif)

### ESTRUCTURA CONDICIONAL MULTIPLE (IF – ELSEIF)

**elseif**, como su nombre lo sugiere, es una combinación de if y else. Del mismo modo que else, extiende una sentencia if para ejecutar una sentencia diferente en caso que la expresión if original se evalúe como **FALSE**. Sin embargo, a diferencia de else, esa expresión alternativa sólo se ejecutará si la expresión condicional del elseif se evalúa como **TRUE**.

```
if ($a > $b) {  
    echo "a es mayor que b";  
} elseif ($a == $b) {  
    echo "a es igual que b";  
} else {  
    echo "a es menor que b";  
}
```

```
if ($a > $b):  
    echo "a es mayor que b";  
elseif ($a == $b):  
    echo "a es igual que b";  
else:  
    echo "a es menor que b";  
endif;
```

**Nota:** Tenga en cuenta que elseif y else if serán considerados exactamente **iguales sólamente cuando se utilizan llaves** como en el ejemplo anterior. Al utilizar los **dos puntos** para definir las condiciones if/elseif, **no debe separarse else if** en dos palabras o PHP fallará con un error del interprete.

```
if ($a > $b):  
    echo "a es mayor que b";  
elseif ($a == $b):  
    echo "a es igual que b";  
else:  
    echo "a es menor que b";  
endif;
```

```
if ($a > $b):  
    echo "a es mayor que b";  
else if ($a == $b):  
    echo "a es igual que b";  
else:  
    echo "a es menor que b";  
endif;
```

## EJERCICIOS (IF – ELSEIF)

1- Se desea diseñar un programa que **escriba los nombres de los días** de la semana en función del **valor** de una **variable DIA**.

2 - En una fábrica de computadoras se planea ofrecer a los clientes un descuento que dependerá del número de computadoras que compre. Si las computadoras son menos de cinco se les dará un 10% de descuento sobre el total de la compra; si el número de computadoras es mayor o igual a cinco pero menos de diez se le otorga un 20% de descuento; y si son 10 o más se les da un 40% de descuento. El precio de cada computadora es de \$700.

```
$dia=4;  
  
if($dia==1){  
    echo "lunes";  
  
}else if($dia==2){//en esta linea esta la instrucción (else if) con espacio y no genera error  
    echo "martes";  
  
}elseif($dia==3){  
    echo "miercoles";  
  
}elseif($dia==4){  
    echo "jueves";  
  
}elseif($dia==5){  
    echo "viernes";  
  
}elseif($dia==6){  
    echo "sabado";  
  
}elseif($dia==7){  
    echo "domingo";  
  
}else{  
    echo "Debes digitar un numero entre 1 y 7";  
}
```

En la notación con dos puntos si genera error al dejar espacio en la misma linea

```
$dia=4;
```

```

if($dia==1):
    echo "lunes";
else if($dia==2):
    echo "martes";
Parse error: syntax error, unexpected token "if", expecting ":"  

in C:\xampp\htdocs\PHP\else_if.php on line 25

```

la notación con dos puntos no permite espacio en la instrucción elseif

video 23 desarrollo ejercicio 2 de else if

```

$cantidad=3;

$total=$cantidad*700;

if($cantidad<5){
    $total=$total-($total*0.10);
}elseif($cantidad>=5 && $cantidad<10){
    $total=$total-($total*0.20);
}elseif($cantidad>=10){
    $total=$total-($total*0.40);
}

echo "El total a pagar es ".$total;

```

En el video solucionaron así, me parece que mi solución es más lógica

```

$compu=10;
$precio=700;

$total = $compu * $precio;

if($compu<5){
    $total=($total-$total*10/100);
}elseif($compu<10){
    $total=($total-$total*20/100);
}

```

```
 }else{
    $total=($total-$total*40/100);
}

echo "compro ".$compu." computadoras, debe pagar un total de ".$total;
```

video 24 estructuras condicionales anidadas en php

### ESTRUCTURAS CONDICIONALES ANIDADAS

```
if (condicion) {
    if(condicion){
        Bloque de código a ejecutar si la condición es TRUE
    }else{
        Bloque de código a ejecutar si la condición es FALSE
    }
} elseif (condicion) {
    Bloque de código a ejecutar si la condición es TRUE
} else {
    Bloque de código a ejecutar si la condición es FALSE
}
```

Ejercicio

### EJERCICIOS (CONDICIONALES ANIDADAS)

1- Pida a usuario la edad y el genero, para que la computadora le indique si ya puede jubilarse. Tome en cuenta que un Hombre se puede jubilar cuando tenga 60 años o más, en cambio, una mujer mayor se jubilara si tiene más de 54 años.

```
$edad=51;
```

```
$genero="femenino";
```

```
if ($genero=="masculino"){
    if($edad>60){
        echo "su genero es: ".$genero." su edad es: ".$edad."ya puede jubilarce";
    }
}
```

```

}else{
    echo "su genero es: ".$genero." pero no llega a la edad de jubilacion";
}

}else if($genero == "femenino"){

    if($edad>54){

        echo "su genero es: ".$genero." su edad es: ".$edad."ya puede jubilarce";

    }else{

        echo "su genero es: ".$genero." pero no llega a la edad de jubilacion";

    }

}

echo "digito mal el genero";
}

```

Solución del video

```

$edad=60;
$genero="M";

if($genero=="M"){
    if($edad>=60){
        echo "Puede jubilarse";
    }else{
        echo "No puede jubilarse";
    }
}elseif($genero=="F"){
    if($edad>54){
        echo "Puede jubilarse";
    }else{
        echo "No puede jubilarse";
    }
}else{
    echo "Coloque una opcion valida";
}

```

## Video 25 Estructura de Selección múltiple

### SWITCH

La sentencia switch **es similar** a una **serie de sentencias IF** en la misma expresión.

En muchas ocasiones, es posible que se quiera **comparar la misma variable (o expresión)** con muchos **valores diferentes**, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para esto es exactamente la expresión switch.

La comparación es una comprobación de **igualdad débil (==)**

```
switch (variable) {  
    case valor1:  
        Código a ejecutar  
        break;  
  
    case valor2:  
        Código a ejecutar  
        break;  
  
    default:  
        Código a ejecutar  
}
```

```
$fruta="pera";  
  
switch($fruta){  
    case "fresa":  
        echo "eres una fresa";  
        break;  
    case "uva":  
        echo "eres una uva";  
        break;  
    case "limon":  
        echo "eres un limon";  
        break;  
    case "pera":  
        echo "eres una pera";  
        break;  
    default:  
        echo "no estas en las opciones";  
        break;  
}
```

## EJERCICIOS (SWITCH)

Se desea diseñar un programa que escriba los nombres de los días de la semana en función del valor de una variable DIA.

Los días de la semana son 7; por consiguiente, el rango de valores de DIA será 1..7, y caso de que DIA tome un valor fuera de este rango se deberá producir un mensaje de error advirtiendo la situación anómala.

```
$dia=7;

switch($dia){
    case 1:
        echo "Lunes";
        break;
    case 2:
        echo "Martes";
        break;
    case 3:
        echo "Miercoles";
        break;
    case 4:
        echo "Jueves";
        break;
    case 5:
        echo "Viernes";
        break;
    case 6:
        echo "Sabado";
        break;
    case 7:
        echo "Domingo";
        break;
    default:
        echo "debes escoger una opcion entre 1 y 7";
        break;
}
```

Switch hace una comparación de igualdad que no es estricta osea en el ejercicio anterior podríamos haber introducido el día como un String y funciona de la misma manera

## MATCH

La expresión match ramifica la evaluación basada en una comprobación de identidad de un valor. De forma similar a una **sentencia switch**, una expresión match tiene una expresión de sujeto que se compara con múltiples alternativas.

A diferencia de switch, la comparación es una comprobación de **identidad (==)** en lugar de una comprobación de igualdad débil (==). Las expresiones match están disponibles a partir de

s **PHP 8.0.0.**

**match (variable) {**  
    \$variable => Código a ejecutar,  
    default => Código a ejecutar

**}**

```
$a=7;  
$x=10;  
$y=9;  
$z=7;
```

```
$resultado=match($a){//comparación parecida a switch pero es una comparación estricta  
(==)}
```

```
    $x => "Valor igual a x",  
    $y => "Valor igual a y",  
    $z => "Valor igual a z",  
    default =>"No es igual a ningun valor"
```

```
};
```

```
echo $resultado;
```

```

$edad=18;

$resultado = match (true) {//se coloca el true para hacer un tipo de comparación
    //que no es de igualdad
    $edad>=60 => "Eres de la tercera edad",
    $edad>= 30=> "Eres un adulto",
    $edad>=18=>"Eres un adulto joven",
    default => "Eres un niño"

};

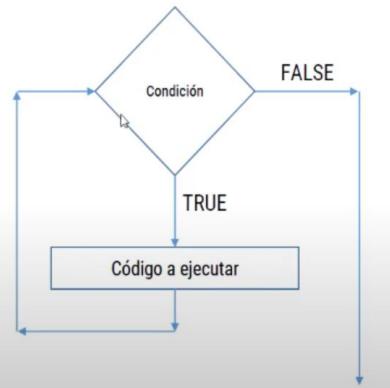

```

Video 27 Ciclos

## ESTRUCTURAS REPETITIVAS (CICLOS | BUCLES | LOOPS)

Las estructuras repetitivas son aquellas que reiteran una o un grupo de instrucciones "n" veces y dependen de una variable de control del ciclo llamada **contador**. Es decir, **ejecutan una o varias instrucciones un número de veces definido**.

Los cuatro bucles en PHP son el **bucle while**, el **bucle for**, el **bucle do-while** y el **bucle foreach**.



### CICLO WHILE

La instrucción while (mientras) ejecuta una porción de programa mientras se cumpla una cierta condición. Mientras la condición sea **verdadera**, se **ejecutan las instrucciones contenidas** en el while. Cuando **deja de cumplirse la condición**, se **sale del ciclo** y se continúa ejecutando el resto del programa.

```

while(Condicion){
    Código a ejecutar
}

```

```

while(Condicion):
    Código a ejecutar
endwhile;

```

## EJEMPLOS

1. Diseñe un programa que imprima los números del 1 hasta el 20. (Incremento y decremento).  
↳
2. Diseñe un programa que imprima la tabla de multiplicar de un numero dado, desde el factor 1 hasta el 12. (Incremento y decremento).

Desarrollo

```
1. $cont=1;
2.
3. while($cont<=20){
4.     echo $cont."<br>";
5.     $cont++;
6. }
7.
8. $cont=20;
9. while($cont>=1){
10.    echo $cont."<br>";
11.    $cont--;
12. }
```

El primero es ascendente y el segundo descendente

2. ejercicio

\$num=73;//cualquier numero que se quiera genera la tabla de multiplicar hasta el 12

```
$cont=1;
while($cont<=12){
    echo $num."x".$cont."=".$num*$cont."<br>";
    $cont++;
}
```

```
$cont=12;
while($cont>=1){
    echo $num."x".$cont."=".$num*$cont."<br>";
    $cont--;
}
```

## CICLO DO-WHILE

La estructura repetitiva do-while (hacer-mientras) es muy similar a la estructura while, excepto que **la expresión verdadera es verificada al final** de cada iteración en lugar de al principio. **Hay una sola sintaxis para bucles do-while.**

```
do{  
    Código a ejecutar  
}while(Condicion);
```

## EJEMPLOS

1. Diseñe un programa que imprima los números del 1 hasta el 20. (Incremento y decremento).
2. Diseñe un programa que imprima la tabla de multiplicar de un numero dado, desde el factor 1 hasta el 12. (Incremento y decremento).

---

Desarrollo 1. Ejemplo:

```
$cont=1;
```

```
do{  
    echo $cont."<br>";  
    $cont++;  
}while($cont<=20);
```

```
echo "<br>";
```

```
$cont=20;
```

```
do{  
    echo $cont."<br>";
```

```
$cont--;
}while($cont>=1);
```

Ejemplo 2

```
$num=3;
```

```
$cont=1;
```

```
do{
```

```
echo $cont." X ".$num." = ".$cont*$num." /// ";
```

```
$cont++;
```

```
}while($cont<=12);
```

```
echo "<br>";
```

```
$cont=12;
```

```
do{
```

```
echo $cont." X ".$num." = ".$cont*$num." /// ";
```

```
$cont--;
```

```
}while($cont>=1);
```

```
echo "<br>";
```

Video 29 ciclo for

## CICLO FOR

La estructura repetitiva for (para) se utiliza generalmente cuando tenemos bien determinada la cantidad de repeticiones a realizar. Se **diferencia** de las anteriores en que **se debe incluir** en la propia instrucción una **variable de control**, la cual se **incrementa** o **decrementa** de forma automática

```
for (var; Condicion; incre | decre){    for(var; Condicion; incre | decre):  
    Código a ejecutar                      Código a ejecutar  
}  
                                         endfor;
```

Los mismo ejercicios anteriores resueltos con el ciclo for:

```
<?php  
  
for($i=1;$i<=20;$i++){  
    echo $i."/";  
}  
echo "<br>";  
for($i=20;$i>=1;$i--){  
    echo $i.*";  
}  
echo "<br>";  
$num=5;  
for ($i=1; $i <= 12; $i++) {  
    echo $i." X ".$num." = ".$i*$num." / ";  
}  
echo "<br>";  
$num=5;  
for ($i=12; $i >= 1; $i--) {  
    echo $i." X ".$num." = ".$i*$num." / ";  
}
```

```
echo "<br>utilizando notacion alternativa <br>";
for ($i=1; $i <=20 ; $i++):echo $i."/";endfor;
```

Video 30 ciclo foreach

## CICLO FOREACH

El ciclo foreach proporciona un modo sencillo de **iterar sobre arrays**. foreach funciona **sólo sobre arrays** y emitirá un error al intentar usarlo con una variable de un tipo diferente de datos o una variable no inicializada. Existen dos sintaxis:

<pre>foreach(\$array as \$valor){     \$valor tendrá en cada iteración     un valor del array }  :  }</pre>	<pre>foreach(\$array as \$clave =&gt; \$valor){     \$clave tendrá en cada iteración     una clave del array      \$valor tendrá en cada iteración     un valor del array }</pre>
---	---

```
$laptop=["Acer Nitro 5","Windows 11", "AMD Ryzen 5 4600Hz", "SSD 256GB",
"RAM 24GB"];
```

```
$frutas=[  
    "Fresas"=>100,  
    "Peras"=>30,  
    "Sandias"=>10,  
    "Melocotones"=>17,  
    "Manzanas"=>9  
];  
  
foreach($laptop as $valor){  
    echo $valor."<br>";  
}
```

\$valor se puede cambiar por cualquier nombre de variable que nosotros le queramos asignar.

```
foreach($laptop as $clave =>$valor){  
    echo $clave." = ".$valor."<br>";//muestra posicion y valor  
}
```

Tanto \$clave como \$valor pueden tomar cualquier nombre de variable que se quiera asignar.

```
foreach($frutas as $valor){
```

```
    echo $valor."<br>";//muestra solo los valores
}
```

100  
30  
10  
17  
9

```
foreach($frutas as $clave =>$valor){
    echo "Hay ".$valor." ".$clave."En el Inventario" . "<br>";//muestra
posicion y valor
}
```

Hay 100 Fresas En el Inventario  
Hay 30 Peras En el Inventario  
Hay 10 Sandias En el Inventario  
Hay 17 Melocotones En el Inventario  
Hay 9 Manzanas En el Inventario

Video 30 10:32

```
$productos =[  
    ["codigo"=>"A0001", "descripcion"=>"Mouse"],  
    ["codigo"=>"A0002", "descripcion"=>"Teclado"],  
    ["codigo"=>"A0003", "descripcion"=>"Monitor"],  
    ["codigo"=>"A0004", "descripcion"=>"Impresora"]  
];  
  
foreach($productos as $produ){  
    echo $produ["codigo"]." - ".$produ["descripcion"]."<br>";  
}
```

Video 31 (salir o detener un ciclo con Break & Continue)

```
$cont=1;  
  
while($cont<=20){  
    echo $cont."<br>";  
    if($cont==10){  
        break;  
    }  
    $cont++;  
}
```

La instrucción break termina el ciclo cuando se cumple la condición

```
$pc=["SO","SSD","GPU","RAM","CPU"];  
  
foreach($pc as $p){  
    echo $p."<br>";  
    if ($p=="RAM"){  
        break;  
    }  
}
```

Muestra en pantalla:

```
SO  
SSD  
GPU  
RAM
```

Continue hace que el ciclo se siga ejecutando, pero no realiza las instrucciones que están después de (continue) si no que envia el flujo del programa a el inicio del ciclo para realizar la siguiente iteración.

```
$pc=["SO","SSD","GPU","RAM","CPU"];  
  
foreach($pc as $p){  
    if ($p=="RAM"){
```

```
        continue;  
    }  
    echo $p."<br>";  
}
```

Lo unico que no muestra es RAM porque cuando lo detecta continua con el ciclo per sin ejecutar el echo para mostrar la información.

SO  
SSD  
GPU  
CPU

Ejemplo:

```
for ($i=1; $i <= 10; $i++) {  
    if($i==5){  
        continue;  
    }  
    echo $i."<br>";  
}
```

Muestra los numero sel 1 a 10 excluyendo el 5:

Ejemplo con while: a diferencia del ciclo (for) hay que hacer incremento de la variable de control antes de la instrucción (continue) porque si no seguiría evaluando el mismo numero y el ciclo sería un bucle infinito

```
$cont=0;  
  
while($cont<=10){  
    if($cont==5){  
        $cont++; //si no se hace esto se quedaría en un ciclo infinito mostrando el 5  
        continue;  
    }  
    echo $cont."<br>";  
    $cont++;  
}
```

## INCLUDE & REQUIRE

Ambas funciones sirven para añadir otros ficheros a nuestros scripts en PHP.

**include:** inserta en nuestro script un código procedente de otro archivo, si no existe dicho archivo o si contiene algún tipo de error nos mostrará un '**warning**' por pantalla y el **script seguirá ejecutándose**.

**require:** hace la misma operación que include, pero en caso de no existir el archivo o error en el mismo mostrará un '**fatal error**' y el script **no se sigue ejecutando**.

## INCLUDE, INCLUDE\_ONCE, REQUIRE & REQUIRE\_ONCE

```
include("ruta_archivo.php");
include "ruta_archivo.php";
```

```
require("ruta_archivo.php");
require "ruta_archivo.php";
```

```
include_once("ruta_archivo.php");
include_once "ruta_archivo.php";
```

```
require_once("ruta_archivo.php");
require_once "ruta_archivo.php";
```

```
<?php
```

```
include("forz.php");//la ruta o si esta en la misma carpeta el nombre del archivo
```

```
include("for.php");//si se incluye dos veces se ejecuta de nuevo
```

el primer archivo esta mal escrito al no encontrar la ruta genera un warning pero se sigue ejecutando el programa y muestra el segundo llamado al archivo for.php que si esta bien escrito.

**Warning:** include(forz.php): Failed to open stream: No such file or directory  
in C:\xampp\htdocs\PHP\include.php on line 3

**Warning:** include(): Failed opening 'forz.php' for inclusion

```
(include_path='C:\xampp\php\PEAR')
in C:\xampp\htdocs\PHP\include.php on line 3
1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/18/19/20/
20--19--18--17--16--15--14--13--12--11--10--9--8--7--6--5--4--3--2--1--
1 X 5 = 5 / 2 X 5 = 10 / 3 X 5 = 15 / 4 X 5 = 20 / 5 X 5 = 25 / 6 X 5 = 30 / 7 X 5 = 35
/ 8 X 5 = 40 / 9 X 5 = 45 / 10 X 5 = 50 / 11 X 5 = 55 / 12 X 5 = 60 /
12 X 10 = 120 / 11 X 10 = 110 / 10 X 10 = 100 / 9 X 10 = 90 / 8 X 10 = 80 / 7 X 10
= 70 / 6 X 10 = 60 / 5 X 10 = 50 / 4 X 10 = 40 / 3 X 10 = 30 / 2 X 10 = 20 / 1 X 10 =
10 /
utilizando notacion alternativa
1/2/3/4/5/6/7/8/9/10/
```

Con (require)

```
require("for.php");//la ruta o si esta en la misma carpeta el nombre del
archivo
require("for.php");//si se incluye dos veces se ejecuta de nuevo
```

```
12 X 10 = 120 / 11 X 10 = 110 / 10 X 10 = 100 / 9 X 10 = 90 / 8 X 10 = 80 / 7 X 10
= 70 / 6 X 10 = 60 / 5 X 10 = 50 / 4 X 10 = 40 / 3 X 10 = 30 / 2 X 10 = 20 / 1 X 10 =
10 /
12 X 10 = 120 / 11 X 10 = 110 / 10 X 10 = 100 / 9 X 10 = 90 / 8 X 10 = 80 / 7 X 10
= 70 / 6 X 10 = 60 / 5 X 10 = 50 / 4 X 10 = 40 / 3 X 10 = 30 / 2 X 10 = 20 / 1 X 10 =
10 /
```

Si hay un error en el archivo o no existe

```
require("forz.php");//la ruta o si esta en la misma carpeta el nombre del
archivo
require("for.php");//si se incluye dos veces se ejecuta de nuevo
```

**Warning:** require(forz.php): Failed to open stream: No such file or directory  
in C:\xampp\htdocs\PHP\include.php on line 3

**Fatal error:** Uncaught Error: Failed opening required 'forz.php'  
(include\_path='C:\xampp\php\PEAR') in

C:\xampp\htdocs\PHP\include.php:3 Stack trace: #0 {main} thrown  
in **C:\xampp\htdocs\PHP\include.php** on line **3**

Nos muestra fatal error.

```
require_once("for.php");//la ruta o si esta en la misma carpeta el nombre del archivo  
require_once("for.php");//si se incluye dos veces se ejecuta solo una
```

```
include_once("for.php");//la ruta o si esta en la misma carpeta el nombre del archivo  
include_once("for.php");//si se incluye dos veces se ejecuta solo una
```

ejemplo de llamar a menú y a footer en una pagina, de esta forma todas las pagina tienen el mismo menú y el mismo footer, si quiero agregar items al menú simplemente se cambia una sola vez y automáticamente se mostrara en todas las pagina, de igual manera se haría con el footer

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Include</title>  
</head>  
<body>  
    <?php include("./inc/nav.php") ?>  
    <h1>Pagina Principal</h1>  
    <?php include "inc/footer.php" ?>  
    <!-- tambien se puede llamar la carpeta de las dos formas, vemos  
    de esta forma reutilizamos codigo y son mas legibles los programas -->  
</body>  
</html>
```

Nav.php

```
<nav>  
    <ul>  
        <li><a href="index.php">Inicio</a></li>  
        <li><a href="index2.php">Segundo</a></li>  
        <li>menu modificado agregar enlaces...</a></li>
```

```
</ul>
</nav>
```

Footer.php

```
<footer>Pie de pagina modificado y aplicaria para todas las paginas</footer>
```

Nos muestra la estructura

- [Inicio](#)
- [Segundo](#)
- menu modificado agregar enlaces...

# Pagina Principal

Pie de pagina modificado y aplicaria para todas las paginas

Video 33 Funciones

## FUNCIONES (PROPIAS)

Una función es un **conjunto de instrucciones** a la que podemos **recurrir siempre que queramos**. Éstas **pueden recibir parámetros** y realizar todo tipo de tareas, ya sean complejas o sencillas.

Un **nombre de función válido** comienza con una **letra o guión bajo**, seguido de cualquier **número de letras, números o guiones bajos**.

```
function nombre_de_función($parametros){
    Código de la función
}
```

```
function saludo(){
    echo ("hola, soy una función.<br>");
}
```

```
saludo();
saludo();
```

```
saludo();
```

muestra:

hola, soy una función.

hola, soy una función.

hola, soy una función.

De esta forma la función no devuelve ningún valor si quisiéramos cargar en una variable lo que está en la función se debe utilizar (return).

```
function saludo(){
    return ("hola, soy una función.<br>");
}

$saludar =saludo();
echo $saludar;//de esta forma la función retorna un valor
```

tambien se puede mostrar directamente lo que retorna la función.

```
function saludo(){
    return ("hola, soy una función.<br>");
}

echo saludo();//mostramos directamente lo que devuelve la función.
```

Función con parámetros

```
function saludo($nombre){
    return ("hola, soy: $nombre <br>");
}

echo saludo("Nelson");
$user = "Fernando";

echo saludo($user);
```

podemos enviar directamente el String que sirve de parámetro o incluido en otra variable que puede ser un nombre diferente al que tiene la función

muestra:

hola, soy: Nelson

hola, soy: Fernando

otra forma de enviar el parámetro

```
echo saludo($nombre="Carlos");
```

ejemplo:

```
unction promedio_alumno($nota_1, $nota_2, $nota_3){  
    return ($nota_1+$nota_2+$nota_3)/3;  
}  
  
echo "su promedio es: ".promedio_alumno(10,10,10);
```

para utilizar un parámetro dentro de la función debe coincidir exactamente como se definió en los parámetros

video 34 incluir y llamar funciones

el ejemplo esta en el archivo incluir\_funciones.php, para poder utilizar funciones que están en un archivo diferente se debe utilizar include o require para poder hacer el llamado a la función

si están en la misma carpeta no hay que dar ruta si esta en otra carpeta debe hacer la ruta donde está el archivo.

```
<?php
```

include "funciones.php";//de esta forma se hace cuando esta en la misma carpeta el archivo que hace el llamado y el archivo donde esta la función.

```
echo promedio_alumno(4.5,6.5,8.9);
```

de esta forma se utiliza la función que esta en otro archivo

ejemplo con require y con el archivo en otra carpeta

```
require "funciones/funciones.php";// el archivo esta en la carpeta funciones
```

```
echo promedio_alumno(4.5,6.5,8.9);
```

para entrar en una carpeta simplemente con el nombre de la carpeta y (/) el símbolo dividir, o tambien ( ./nombre de carpeta )

ahora si el archivo esta en otra carpeta, debemos salir de la carpeta actual eso con (../) después nombre de la carpeta que contiene el archivo y después el archivo

“../nombre\_carpeta/archivo.php”

## Video 35 FUNCIONES PARA FORMATEAR STRING A MAYUSCULAS Y MINUSCULAS

Son funciones predefinidas de PHP

Strtolower: mayúsculas a minusculas

```
$cadena_texto="Hola Mundo";
```

echo strtolower(\$cadena\_texto); convierte mayúsculas en minúsculas

ejemplo con variable

```
$cadena_texto="Hola Mundo";
$cadena_texto= strtolower($cadena_texto);
echo $cadena_texto;
```

hola mundo

strtoupper: minúsculas a mayúsculas.

Ejemplo:

```
$cadena_texto="Hola Mundo";
$cadena_texto= strtoupper($cadena_texto);
echo $cadena_texto;
```

resultado:

HOLA MUNDO

ucfirst: convierte la primera letra del String a mayuscula

```
$cadena_texto="hola mundo";
$cadena_texto= ucfirst($cadena_texto);
echo $cadena_texto;
```

resultado:

Hola mundo

ucwords: convierte la primera letra de cada palabra en mayúscula.

```
$cadena_texto="hola mundo";
$cadena_texto= ucwords($cadena_texto);
echo $cadena_texto;
```

resultado

Hola Mundo

Video 36 funciones para contar caracteres y palabras de un string

- Strlen()

```
$cadena_texto="hola mundo";
$longitud=strlen($cadena_texto);//strlen: para contar caracteres, el espacio es un caracter
echo $cadena_texto." tiene ".$longitud." caracteres";
```

- str\_word\_count()

```
$palabras=str_word_count($cadena_texto);//str_word_count contar palabras
echo $cadena_texto.", tiene ".$palabras." palabras<br>";
```

video 37 como convertir un String en un Array

explode(delimitador, string, limitador):nos permite convertir string en array

```
$fecha_1 = "2021/11/29";

$array_fecha= explode("/", $fecha_1);
var_dump($array_fecha);
echo "<br> año $array_fecha[0]";
echo "<br> mes $array_fecha[1]";
echo "<br> dia $array_fecha[2]";
muestra
```

```
array(3) { [0]=> string(4) "2021" [1]=> string(2) "11" [2]=> string(2) "29" }
a o 2021
mes 11
dia 29
```

```
$fecha_2 = "2021-11-30";
```

```
$array_fecha= explode("-",$fecha_2);
var_dump($array_fecha);
echo "<br> a o $array_fecha[0]";
echo "<br> mes $array_fecha[1]";
echo "<br> dia $array_fecha[2]";
muestra
```

```
array(3) { [0]=> string(4) "2021" [1]=> string(2) "11" [2]=> string(2) "30" }
a o 2021
mes 11
dia 30
```

```
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";
```

```
$array_numeros= explode(" ",$numeros);
var_dump($array_numeros);
echo "<br>";
foreach($array_numeros as $num){
    echo "$num<br>";
}
```

```
Muestra
```

```
array(7) { [0]=> string(3) "Uno" [1]=> string(3) "Dos" [2]=> string(4) "Tres" [3]=> string(6)
"Cuatro" [4]=> string(5) "Cinco" [5]=> string(4) "Seis" [6]=> string(5) "Siete" }
Uno
Dos
Tres
Cuatro
Cinco
Seis
Siete
```

uso del tercer parámetro

explode (delimitador, string, **limitador**):

```
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";  
  
$array_numeros= explode(" ",$numeros,2); //lo dividirá solo en 2  
var_dump($array_numeros);  
echo "<br>";  
foreach($array_numeros as $pos=>$num){  
    echo "en la posicion $pos esta: $num<br>";  
}
```

Muestra

```
array(2) { [0]=> string(3) "Uno" [1]=> string(32) "Dos Tres Cuatro Cinco Seis Siete" }  
en la posicion 0 esta: Uno  
en la posicion 1 esta: Dos Tres Cuatro Cinco Seis Siete
```

divide de izquierda a derecha según el limitador que tenga en las partes que dice el limitador, ahora vamos a motrar con el limitador en 3:

```
$array_numeros= explode(" ",$numeros,3);
```

```
array(3) { [0]=> string(3) "Uno" [1]=> string(3) "Dos" [2]=> string(28) "Tres  
Cuatro Cinco Seis Siete" }  
en la posicion 0 esta: Uno  
en la posicion 1 esta: Dos  
en la posicion 2 esta: Tres Cuatro Cinco Seis Siete
```

el limitador con números negativos nos excluye de derecha a izquierda la cantidad que designemos si colocamos -1 nos excluye el ultimo -2 el ultimo y penúltimo y asi sucesivamente.

```
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";  
$array_numeros= explode(" ",$numeros,-1)  
Muestra:
```

```
array(6) { [0]=> string(3) "Uno" [1]=> string(3) "Dos" [2]=> string(4) "Tres" [3]=>  
string(6) "Cuatro" [4]=> string(5) "Cinco" [5]=> string(4) "Seis" }  
en la posicion 0 esta: Uno
```

en la posicion 1 esta: Dos  
en la posicion 2 esta: Tres  
en la posicion 3 esta: Cuatro  
en la posicion 4 esta: Cinco  
en la posicion 5 esta: Seis

el array que formo no incluyo el ultimo elemento

```
$array_numeros= explode(" ",$numeros,-4);
```

No incluyo los últimos 4 elementos

array(3) { [0]=> string(3) "Uno" [1]=> string(3) "Dos" [2]=> string(4) "Tres" }  
en la posicion 0 esta: Uno  
en la posicion 1 esta: Dos  
en la posicion 2 esta: Tres

video 38

## Funciones matemáticas

Funciones matemáticas predefinidas: [PHP: Math - Manual](#)

- Pow: potencia

echo pow(5,3); // equivale a 5\*5\*5, osea 5 elevado a la 3

- sqrt: raíz cuadrada de un numero

echo sqrt(25); // raiz cuadrada = 5

- rand: numero random entre dos limites definidos

echo rand(1,100); //genera un numero random entre 1 y 100

- pi: numero pi

echo pi(); //3.1415926535898

- floor():
- ceil():

- echo floor(4.3); //redondea hacia abajo =4

- echo "<br>";

- echo ceil(4.3); //redondea hacia arriba = 5

con numero negativos

echo floor(-3.14); //redondea hacia abajo = -4

echo "<br>";

echo ceil(-3.14); //redondea hacia arriba = -3

- round
- echo round(3.4). "<br>"; //redondea si pasa el decimal de 5 hacia arriba  
si es menor de 5 hacia abajo
- echo round(3.5);

3

4

Recibe otro parámetro que me permite hacer un redondeo de la cantidad de decimales que necesite, esto para mejorar cálculos.

echo round(3.44598522, 2). "<br>"; //redondea con dos decimales

echo round(3.4445999, 2);

3.45

3.44

Video 39

Formatear números o cantidades de dinero

 CURSO de PHP desde CERO - 39 FORMATEAR NUMERO o CANTIDADES de DINERO en PHP

Se utiliza la función “number\_format(cantidad,decimales,sep\_decimal,sep\_millar);”

Esta función se puede utilizar con un parámetro con dos o con cuatro

number\_format( float \$number , int \$decimals = 0 , string \$dec\_point = ":" , string \$thousands\_sep = "," ): string

con un solo parámetro que es el float elimina los decimales haciendo redondeo y separa los miles.

```
$cantidad_1 = 12732.5;  
$cantidad_2 = 1931.81;  
  
echo number_format($cantidad_1,2,",",".");
```

12.732,50

Si queremos que no tenga separador de millares el ultimo parámetro se establece pero no se coloca nada entre las comillas

```
echo number_format($cantidad_1,2,",","","");
```

12732,50

Separamos millares con espacio

```
$cantidad_1 = 123452732.5;  
echo number_format($cantidad_1,2,","," " );
```

123 452 732,50

Para guardar en base de datos normalmente no se utilizan separador de millares y se utiliza el punto para separar decimales.

Video 40 función DATE

FECHA Y HORA

Con date obtenemos las fechas en inglés

[PHP: date - Manual](#)

Zona horaria:

[PHP: date\\_default\\_timezone\\_set - Manual](#)

```
date_default_timezone_set("America/Bogota");//para establecer la zona  
horaria  
echo date_default_timezone_get();//para leer zona horaria
```

`date( string $format [, int $timestamp = time() ]): string`

Devuelve una cadena formateada según el formato indicado usando el integer **timestamp** (Unix timestamp) dado, o el momento actual si no se da una marca de tiempo. En otras palabras, **timestamp** es opcional y por defecto es el valor de [time\(\)](#).

Antes de usar (date) se debe definir la zona horaria con la siguiente función:

`date_default_timezone_set:`

`date_default_timezone_set` — Establece la zona horaria predeterminada usada por todas las funciones de fecha/hora en un script

`date_default_timezone_set(string $timezone_identifier): bool`

`date_default_timezone_get` — Obtiene la zona horaria predeterminada usada por todas las funciones de fecha/hora en un script

`date_default_timezone_get(): string`

format para date()

Carácter de formato	Descripción	Ejemplo de valores devueltos
Día	---	---
d	Día del mes, 2 dígitos con ceros iniciales	01 a 31
D	Una representación textual de un día, tres letras	Mon hasta Sun
j	Día del mes sin ceros iniciales	1 a 31
l ('L' minúscula)	Una representación textual completa del día de la semana	Sunday hasta Saturday
N	Representación numérica ISO-8601 del día de la semana (añadido en PHP 5.1.0) ↳	1 (para lunes) hasta 7 (para domingo)
S	Sufijo ordinal inglés para el día del mes, 2 caracteres	st, nd, rd o th. Funciona bien con j
w	Representación numérica del día de la semana	0 (para domingo) hasta 6 (para sábado)
z	El día del año (comenzando por 0)	0 hasta 365
Semana	---	---
W	Número de la semana del año ISO-8601, las semanas comienzan en lunes	Ejemplo: 42 (la 42 <sup>a</sup> semana del año)
Mes	---	---
F	Una representación textual completa de un mes, como January o March	January hasta December

Mes		
F	Una representación textual completa de un mes, como January o March	January hasta December
m	Representación numérica de un mes, con ceros iniciales	01 hasta 12
M	Una representación textual corta de un mes, tres letras	Jan hasta Dec
n	Representación numérica de un mes, sin ceros iniciales	1 hasta 12
t	Número de días del mes dado	28 hasta 31
Año		
L	Si es un año bisiesto	1 si es bisiesto, 0 si no.
o	Año según el número de la semana ISO-8601. Esto tiene el mismo valor que Y, excepto que si el número de la semana ISO (w) pertenece al año anterior o siguiente, se usa ese año en su lugar. (añadido en PHP 5.1.0)	Ejemplos: 1999 o 2003
Y	Una representación numérica completa de un año, 4 dígitos	Ejemplos: 1999 o 2003
y	Una representación de dos dígitos de un año	Ejemplos: 99 o 93

Hora		
a	Ante meridiem y Post meridiem en minúsculas	am o pm
A	Ante meridiem y Post meridiem en mayúsculas	AM o PM
B	Hora Internet	000 hasta 999
g	Formato de 12 horas de una hora sin ceros iniciales	1 hasta 12
G	Formato de 24 horas de una hora sin ceros iniciales	0 hasta 23
h	Formato de 12 horas de una hora con ceros iniciales	01 hasta 12
H	Formato de 24 horas de una hora con ceros iniciales	00 hasta 23
i	Minutos con ceros iniciales	00 hasta 59
s	Segundos con ceros iniciales	00 hasta 59
u	Microsegundos (añadido en PHP 5.2.2). Observe que <b>date()</b> siempre generará 000000 ya que toma un parámetro de tipo integer, mientras que <b>DateTime::format()</b> admite microsegundos si <b>DateTime</b> fue creado con microsegundos.	Ejemplo: 654321
v	Milisegundos (añadido en PHP 7.0.0). La misma observación se aplica para u.	Example: 654

8:34

```
$fecha_us=date("Y-m-d");
echo $fecha_us;
```

```
echo "<br>";
$fecha_us=date("Y/m/d");
echo $fecha_us;
echo "<br>";
$fecha_es = date("d-m-Y");
echo $fecha_es;
```

2025-02-26

2025/02/26

26-02-2025

### Fecha completa

```
$fecha_us=date("l d F Y");
echo $fecha_us;
```

Wednesday 26 February 2025

### HORA

\$hora\_12 = date("h:i:s a");

echo \$hora\_12;

me muestra la hora actual

06:37:10 am

\$hora\_12 = date("h:i:s A");=> 06:37:51 AM

### Fecha con hora:

```
$fecha_completa = date("d-m-Y h:i A");
```

```
echo $fecha_completa;
```

26-02-2025 06:41 AM

```
$fecha_completa = date("l d F Y H:i:s ");
```

Wednesday 26 February 2025 06:44:17

Video 41 como obtener la fecha en español

En el ejemplo muestra el detalle del programa

fecha\_espanol.php

10:43

//fecha corta ingresando datos a la función

function fecha\_corta(\$fecha=""), esta función recibe un parámetro una fecha que es la que va a mostrar en forma de fecha corta pero como está con un valor predeterminado de vacío el parámetro si no recibe nada muestra la fecha actual.

```
if($fecha==""){
    $fecha=date("d-m-Y");
} else{
    $fecha=date("d-m-Y",strtotime($fecha));
}
```

Si no se envía el parámetro fecha carga la fecha actual si se envía una fecha entonces por el lado del else se convierte la fecha recibida por medio de strtotime a formato de fecha.

strtotime(): convierte un string en un formato de fecha especificado por medio del date

- echo fecha\_corta("01-12-2022");

01 de Diciembre del año 2022

- echo fecha\_corta("10/01/2022");

01 de Octubre del año 2022 se esperaba el 10 de enero de 2022 no es lo mismo cambiar el separador de fecha

- echo fecha\_corta("2022/10/01");

01 de Octubre del año 2022

- echo fecha\_corta(date("d-m-Y"));

28 de Febrero del año 2025//muestra fecha actual

- \$fecha\_prueba=date("d-m-Y");

```
• echo fecha_corta($fecha_prueba);  
•
```

28 de Febrero del año 2025

## VIDEO 42 COMO ENcriptar una CONTRASEÑA EN PHP

### ¿Qué Es Un Hash Y Cómo Funciona?

Una función criptográfica hash- usualmente conocida como “hash”- es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una **nueva serie de caracteres con una longitud fija.**

Independientemente de la **longitud de los datos de entrada**, el valor hash de salida tendrá **siempre la misma longitud**.

Antes se utilizaba

```
echo md5($clave);  
md5  
o
```

```
573c2beb476f6682d069c260f9395b5d =>md5  
echo sha1($clave); =>fdc94fba83e2f3dba11546ad65e8284ac8352c00
```

```
echo hash("md5",$clave);
```

con esta función se pueden utilizar múltiples algoritmos

para ver todos los algoritmos que soporta :

```
foreach(hash_algos() as $algo){  
    echo $algo."<br>";  
}
```

Resulta una lista de algoritmos para encriptar

Aplicado a la clave

```
foreach(hash_algos() as $algo){  
    echo $algo." - ".hash("$algo",$clave)."<br>" ;  
}
```

Y nos muestra la clave en cada uno de los algoritmos

Mas informacion [PHP: password\\_hash - Manual](#)

Lo mas recomendable es usar lo siguiente:

8:20

```
echo password_hash($clave,PASSWORD_DEFAULT);
```

```
echo password_hash($clave,PASSWORD_BCRYPT,[ "cost"=>11]);
```

**PASSWORD\_BCRYPT** - Usar el algoritmo **CRIPT\_BLOWFISH** para crear el hash. Producirá un hash estándar compatible con [crypt\(\)](#) utilizando el identificador "\$2y\$". El resultado siempre será un string de 60 caracteres, o [false](#) en caso de error.

Opciones admitidas:

Salt: pero ya no se usa

- o cost - denota el coste del algoritmo que debería usarse. Se pueden encontrar ejemplo de estos valores en la página de [crypt\(\)](#).

Si se omite, se usará el valor predeterminado 10. Este es un buen coste de referencia, pero se podría considerar aumentarlo dependiendo del hardware.

Para verificar una clave se utiliza

password\_verify

[PHP: password\\_verify - Manual](#)

Sintaxis

```
password_verify(string $password, string $hash): bool
```

compara el string original, con el string del password y resultado false o true (1 o 0)

## video 43 enviar un formulario

Para recibir después de enviar el formulario

```
$_POST['nombre'];
```

Se puede mostrar o almacenar en una variable

```
echo $_POST['nombre'];
```

```
$nombre= $_POST['nombre'];
```

```
echo $nombre;
```

para el método get hay que cambiar en el formulario el parámetro method="GET"

Y en el archivo que recibe con la variable global \$\_GET

```
$nombre= $_GET['nombre'];
```

```
$asignatura=$_GET['asignatura'];
```

```
$frutas=$_GET['frutas'];
```

```
echo $nombre." - ".$asignatura." - ".$frutas;
```

LA diferencia es que en el método Get la información se ve en la línea de la URL

[localhost/PHP/index.php?nombre=PEDIPT&asignatura=Matematicas&frutas=Manzana](http://localhost/PHP/index.php?nombre=PEDIPT&asignatura=Matematicas&frutas=Manzana)

modifico la URL:

[localhost/PHP/index.php?nombre=PEDIPT&asignatura=Matematicas&frutas=Manzana&apellido=pollito](http://localhost/PHP/index.php?nombre=PEDIPT&asignatura=Matematicas&frutas=Manzana&apellido=pollito)

Agregue esta linea al PHP : \$apellido=\$\_GET['apellido'];

Modifique esta: echo \$nombre." - ".\$asignatura." - ".\$frutas." - ".\$apellido;

Y me muestra en la pagina :

PEDIPT - Matematicas - Manzana – pollito

La ultima información sin haber sido enviada del formulario

Por eso para información que sea sensible se utiliza el método post

## Video 44 select y checkbox

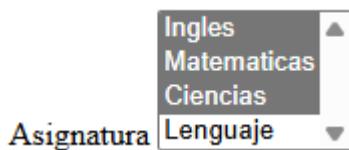
Para seleccionar múltiples valores en un formulario y recuperarlos en php se hace lo siguiente en el archivo html

<select name="asignatura[]" id="asignatura" multiple> ahora se agrego el atributo multiple para recibir varios valores y además en el atributo name se va a recibir un array

Html:

```
<select name="asignatura[]" id="asignatura" multiple>
    <option value="Ingles">Ingles</option>
    <option value="Matematicas">Matematicas</option>
    <option value="Ciencias">Ciencias</option>
    <option value="Lenguaje">Lenguaje</option>
</select>
```

Se selecciono lo siguiente:



Y en php:

```
var_dump($_POST['asignatura']);
```

resultado:

```
array(3) { [0]=> string(6) "Ingles" [1]=> string(11) "Matematicas" [2]=> string(8) "Ciencias" }
```

se pueden mostrar los valores por medio de un foreach

```
foreach($_POST['asignatura'] as $asignatura){
    echo $asignatura."<br>";
}
```

Se puede cargar el valor del select multiple en una variable

```
$materias = $_POST['asignatura'];
foreach($materias as $asignatura){
```

```
    echo $asignatura."<br>";
}
```

## Checkbox multiple

En html se hace lo siguiente

```
<label for="opcion-1">
    <input type="checkbox" value="Manzana" id="opcion-1" name="frutas[]">
    Manzana
</label>
<label for="opcion-2">
    <input type="checkbox" value="pera" id="opcion-2" name="frutas[]">
    Pera
</label>
<label for="opcion-3">
    <input type="checkbox" value="uva" id="opcion-3" name="frutas[]">
    Uva
</label>
```

La diferencia es que en name a parte de que debe tener el mismo nombre tiene que ser un array, que es la información que queremos enviar

En php

```
$frutas = $_POST['frutas'];
foreach($frutas as $fruta){
    echo $fruta."<br>";
}
```

Video 45 saber si una variable es null, vacia o definida

Is\_null

Comprueba si una variable es null devuelve verdadero si es nula y falso si no lo es

Una variable es Null si se define como null \$variable=NULL, cuando no se define un valor a la variable \$variable o cuando destruimos la variable por medio de unset()

```
$numero="9";
```

```
unset($numero);
if(is_null($numero)){
    echo "Es null";
}else{
    echo "No es null";
}
```

Resultado:

**Warning:** Undefined variable \$numero  
in C:\xampp\htdocs\PHP\isset.php on line 5  
Es null

Si comentamos la linea de unset:

No es null

Y no genera ningún warning

Empty: determina si una variable esta vacía

Los siguientes valores son considerados como vacíos:

- "" (una cadena vacía)
- 0 (0 como un integer)
- 0.0 (0 como un float)
- "0" (0 como un string)
- null
- false
- array() (un array vacío)

devuelve true si esta vacía y false si no lo esta

```
$numero="9";
if(empty($numero)){
    echo "esta vacia";
}else{
    echo "no esta vacia";
}
```

Rta: no está vacía

\$numero=""; vacía

```
$numero=0; vacía
```

En formularios se utiliza para verificar que la información recibida tenga un dato

Si modificamos la url así; (<http://localhost/PHP/isset.php?numero=9>)

Y en nuestro archivo php \$numero=\$\_GET['numero']; lo verificamos con empty recargando la página, me dice que no está vacía pero si hacemos la url así:

<http://localhost/PHP/isset.php?numero=>, me dice que está vacía la variable

Isset: determina si una variable está definida y no es null

```
$numero=7;  
unset($numero);  
if(isset($numero)){  
    echo "esta definida";  
}  
else{  
    echo "no esta definida";  
}
```

Rta: no está definida; (no genera warning)

Ejemplo isset

```
<form action="ejeisset.php" method="POST">  
    <input type="text" name="numero">  
    <button type="submit">Enviar</button>  
</form>
```

Recibimos en php y verificamos

```
if(isset($_POST['numero']) && $_POST['numero']!=""){  
    $numero=$_POST['numero'];  
    for($i=12; $i>=1;$i--){  
        echo $numero. " X ".$i. " = ".$i*$numero."<br>";  
    }  
}  
else{  
    echo "no esta definida";  
}
```

Ejemplo uso en el formulario directamente

```
<form action="" method="POST">
    <!-- si se deja el action vacio se envia a la misma pagina -->
    <input type="text" name="numero">
    <button type="submit">Enviar</button>
</form>
<?php
    if(isset($_POST['numero'])&& $_POST['numero']!=""){
        include "eje_isset.php";
    }
?>
```

Y en php;

```
$numero=$_POST['numero'];
for($i=12; $i>=1;$i--){
    echo $numero. " X ".$i. " = ".$i*$numero."<br>";
}
```

19:55

Video 46

Subir o enviar archivos a servidor php con formularios

Creamos un fichero llamado archivo.php donde va estar mi formulario

Para enviar archivos al servidor se agregan otros atributos al formulario y se utiliza un input type file

El atributo para enviar archivos debe ser POST en method y tambien debe tener este atributo enctype="multipart/form-data"

Html:

```
<form action="carga.php" method="POST" enctype="multipart/form-data">
    <input type="file" name="fichero">
    <br><br>
    <button type="submit">Enviar</button>
</form>
```

Y en php

Se utiliza la variable global \$\_FILES['name que tiene en el formulario'][‘atributos que se pueden consultar’];

Los atributos que se pueden consultar son :

name: nombre real del archivo

tmp\_name: ruta temporal donde se almacena el archivo

type: tipo de archivo

error: si no hay errores 0 si hay 1

size: tamaño del archivo en byte

```
echo $_FILES['fichero']['name']."<br>";  
echo $_FILES['fichero']['tmp_name']."<br>";  
echo $_FILES['fichero']['type']."<br>";  
echo $_FILES['fichero']['error']."<br>";  
echo $_FILES['fichero']['size']."<br>";
```

11B387A12B.jpg

C:\xampp\tmp\phpB921.tmp

image/jpeg

0

215351

Ahora vamos a cargar un archivo y lo vamos a mover a una carpeta dentro de nuestro proyecto

Creamos la carpeta en donde tenemos todos los proyectos con el nombre de archivos, allí vamos a llevar los archivos que recibamos por el formulario

Comprobamos si el directorio existe por medio de file\_exists, si existe no hacemos nada, si no existe lo creamos

```
if(!mkdir("archivos",0777)){
```

esta linea me permite crear la carpeta archivos y con el segundo parámetro se da a esa carpeta todos los permisos (escritura, lectura y demás)el if me permite preguntar si la carpeta fue creada correctamente nos genera true y entraría al if pero como queremos que continue el programa le colocamos la pregunta con negación para que continue el programa si todo fue correcto

```
if(!file_exists("archivos")){  
    if(!mkdir("archivos",0777)){  
        echo "Error al crear el directorio";  
        exit();  
    }  
}
```

Con el anterior bloque se asegura que la carpeta donde queramos enviar archivos exista y si no existe se cree, y en caso de que no se pueda crear sale un error y se sale del programa por medio de exit.

Chmod: se utiliza para dar permisos a una carpeta o archivo

chmod("archivos",0777); //se generan todos los permisos a la carpeta archivos.

Mover Archivo

Para mover el archivo a una ubicación definida por nosotros se utiliza la instrucción

move\_uploaded\_files(parámetro 1, parámetro 2), si la carga es correcta genera true si no lo es false

parámetro 1= nombre de la variable que contiene el archivo con su ubicación temporal  
\$\_FILE['nombre variable']['tmp\_name']

parámetro 2 = carpeta destino concatenado con el nombre real del archivo \$\_FILE['nombre variable']['name']

codigo:

```
<?php

if(!file_exists("archivos")){//preguntamos si existe la carpeta
    //si no existe entra al if porque tenemos negado el file_exists
    //Creamos la carpeta
    if(!mkdir("archivos",0777)){
        echo "Error al crear el directorio";
        exit();
    }
}

chmod("archivos",0777);
if(move_uploaded_file($_FILES['fichero']['tmp_name'], "archivos/".$_FILES['fichero']['name'])){
    echo "Archivo Cargado correctamente";
}else{
    echo "ocurrio un error al cargar el archivo";
}
```

Se probó eliminando la carpeta destino y la creo perfectamente

## Como limitar el formato de archivos

Se hace de dos formas dentro del formulario html por medio del atributo accept que se agrega al input

```
<input type="file" name="fichero" accept=".jpg, .png, .jpeg">
```

Se colocan todos los formatos que se acepten, al intentar cargar los archivos solo aparecen los que tenga la extensión que se especificó en el input.

Pero de esta forma solo se ocultan los archivos pues si en la venta se elige ver todos los archivos se podrá cargar cualquier archivo por eso se debe limitar también desde php

Se utiliza la función: mime\_content\_type(\$\_FILE['nombre de variable que contiene el archivo']['tmp\_name'])

Esta función me da el mime type del archivo que se recibe estos mime type se pueden consultar aca [Lista completa de tipos MIME - HTTP | MDN](#) por ejemplo para jpg y jpeg

.jpeg	Imágenes JPEG	image/jpeg
.jpg		

el mime type es image/jpeg y para png

[Media Types](#) busque el de png en esta pagina

png image/png [W3C][[PNG WG](#)]

El mime es image/png el código quedaría así

```
if(mime_content_type($_FILES['fichero']['tmp_name'])!="image/jpeg"
&& mime_content_type($_FILES['fichero']['tmp_name'])!="image/png"){
    echo "tipo de archivo no admitido";
    exit();
}
```

De esta forma solo admitimos esos tipos de archivos si son diferentes a jpeg o a jpg o a png entra al if muestra la información y sale del script o programa no permitiendo la carga del archivo, si es igual a alguna de las extensiones permitidas continua el programa permitiendo que el archivo se guarde

## Limitar el peso de un archivo

```
if($_FILES['fichero']['size']/1024>3072){//El size me da el peso del archivo en byte se pasa a kilobyte y queremos limitar a 3MB pasamos también a kilobyte 3 * 1024
```

```
echo "El archivo supera el peso permitido";
exit();}
```

## video 47 subir o enviar archivos con PHP y AJAX (API FETCH)

Se utilizan para ejemplo el video archivo.php y carga.php que se utilizaron en la clase de enviar archivos pero ahora vamos a utilizar AJAX

El código Ajax.js tiene errores de sincronismo se debe mejorar. Se mejoró con la ayuda de la IA Quedó así:

Js:

```
const formularios_ajax = document.querySelectorAll(".FormularioAjax");

async function enviar_formulario_ajax(e){
    e.preventDefault();
    let enviar = confirm("Quieres enviar el formulario");
    if(enviar){
        let data= new FormData(this);
        let method = this.getAttribute("method");
        let action = this.getAttribute("action");

        let encabezados = new Headers();

        let config = {
            method: method,
            headers: new Headers(),
            mode: 'cors',
            cache: 'no-cache',
            body: data
        };
        /*fetch(action,config)
        .then(respuesta => respuesta.text())
        .then(respuesta =>{
            alert(respuesta);
        });la siguiente es la corrección por IA pues no funcionaba*/
        try{
            let respuesta = await fetch(action, config);
            let texto = await respuesta.text();
            alert (texto);

        }catch(error){
            console.error("Error en la solicitud", error);

        }
    }
}
```

```
formularios_ajax.forEach(formularios =>{
    formularios.addEventListener("submit", enviar_formulario_ajax);
});
```

Html:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cargar Archivo</title>
</head>
<body>
    <h3>Subir archivo con PHP</h3>
    <form action="carga.php" method="POST" enctype="multipart/form-data"
class="FormularioAjax">
        <input type="file" name="fichero" accept=".jpg, .png, .jpeg">
        <br><br>
        <button type="submit">Enviar</button>
    </form>
    <form action="carga.php" method="POST" enctype="multipart/form-data"
class="FormularioAjax">
        <input type="file" name="fichero" accept=".jpg, .png, .jpeg">
        <br><br>
        <button type="submit">Enviar</button>
    </form>
    <Script src="ajax.js"></Script>
</body>
</html>
```

Php:

```
<?php
if(mime_content_type($_FILES['fichero']['tmp_name'])!="image/jpeg"
&& mime_content_type($_FILES['fichero']['tmp_name'])!="image/png"){
    echo "tipo de archivo no admitido";
    exit();
}

if(($_FILES['fichero']['size']/1024)>2048){//El syze me da el peso del
archivo en byte se pasa a kilobyte y queremos limitar a 3MB pasamos tambien
a kilobyte 3 * 1024
    echo "El archivo supera el peso permitido";
    exit();
}
```

```
}

if(!file_exists("archivos")){//preguntamos si existe la carpeta
    //si no existe entra al if porque tenemos negado el file_exists
    //creamos la carpeta
    if(!mkdir("archivos",0777)){
        echo "Error al crear el directorio";
        exit();
    }
}

chmod("archivos",0777);
if(move_uploaded_file($_FILES['fichero']['tmp_name'],"archivos/".$_FILES['fichero'][ 'name'])) {
    echo "Archivo Cargado correctamente";
}else{
    echo "ocurrio un error al cargar el archivo";
}
```

Video 48 cookies

PHP de 0 a 100

PHP

## Cookies en PHP

Las cookies son un mecanismo por el que se **almacenan datos** en el navegador para monitorizar o identificar a los usuarios que vuelvan al sitio web. En otras palabras podemos decir que **las cookies son pequeños archivos** donde almacenamos un datos, estos archivos se **almacen en el navegador del cliente**.

Las cookies se deben de crear **antes del Doctype**, ya que han de ser generadas antes de que el **navegador procese el código HTML**.

**Ejemplo de uso de cookies:** preferencias de idioma, seguimiento de anuncios etc.

Para crear

```
setcookie("Nombre", valor, expiracion, dir, dominio, secure, httponly);
```

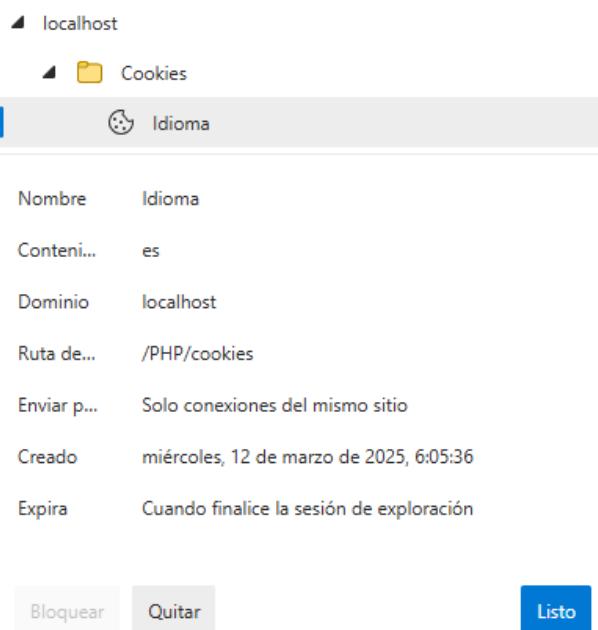
en la parte de arriba del codigo html se inserta el codigo php con la cookie

```
<?php  
    //setcookie("Nombre", valor, expiration, dir, dominio, secure,  
    httponly);  
    setcookie("Idioma", "es");  
  
?>
```

## Cookies en uso

Permitido    Bloqueado

Las siguientes cookies se establecieron cuando visualizó esta página



A screenshot of a browser's cookie settings dialog. At the top, there are two buttons: 'Permitido' (underlined) and 'Bloqueado'. Below them, it says 'Las siguientes cookies se establecieron cuando visualizó esta página'. A tree view shows 'localhost' expanded, with 'Cookies' under it, and 'Idioma' selected. The details for the 'Idioma' cookie are listed below:

Nombre	Idioma
Conten...	es
Dominio	localhost
Ruta de...	/PHP/cookies
Enviar p...	Solo conexiones del mismo sitio
Creado	miércoles, 12 de marzo de 2025, 6:05:36
Expira	Cuando finalice la sesión de exploración

At the bottom are three buttons: 'Bloquear', 'Quitar', and a blue 'Listo' button.

Obtener el valor de la cookie

Dentro de html

```
<h1><?php echo $_COOKIE['Idioma'];?></h1>
```

Definir expiración de la cookie

```
setcookie("Idioma", "es", time() + 60 * 60 * 24 * 30);
```

el parámetro time me da la fecha de hoy + 60 segundos \* 60 minutos \* 24 horas (hasta aquí es un día)\*30 (sería un mes) si quisieramos un año se multiplica por 365

definir el directorio

```
setcookie("Idioma", "es", time() + 60 * 60 * 24 * 365, "/");
```

"/" significa que estará disponible en cualquier parte de nuestro servidor, como esta en el local host, sería solo en esta servidor

```
setcookie("Idioma", "es", time() + 60 * 60 * 24 * 365, "/archivos/"); si quisieramos que estuviera disponible solo en la carpeta archivos de nuestro localhost
```

el parámetro (secure) si es true, solo se crea la cookie si el protocolo es seguro (HTTPS) si es false se puede crear así el protocolo no sea seguro

el parámetro (httponly) si es false, quiere decir que la cookie se puede acceder desde cualquier protocolo y true solo por el protocolo http

```
setcookie("Idioma", "es", time() + 60 * 60 * 24 * 365, "/", "localhost", false, false);
```

eliminar una cookie

en el archivo cookie\_del.php

se crea la cookie con el time negativo

```
setcookie("Idioma", "es", time() - 60, "/", "localhost", false, false); al ejecutar la página elimina la cookie
```

video 49

secciones en PHP

## Sesiones en PHP

Las **sesiones** en aplicaciones web desarrolladas con **PHP** nos sirven para **almacenar información** durante toda la visita de un usuario a un **sitio web**. Dicha información se almacena en el **servidor**.

**Ejemplo de uso de sesiones:** inicio de sesión (login), buscadores (que recuerde término de búsqueda) etc.

Las sesiones se empiezan al inicio de nuestro código antes de cualquier otro código

- Para crear sesión: session\_start();

La sesión se almacena en el servidor, como estamos utilizando localhost vemos que se almacena en las cookies con el nombre PHPSESSID

Cookies en uso

Permitido Bloqueado

Las siguientes cookies se establecieron cuando visualizó esta página

localhost  
Cookies  
PHPSESSID

Nombre	Contenido	Dominio	Ruta de	Enviar p...	Creado	Expira
no se seleccionó ninguna cookie						

Bloquear Quitar Listo

- Para cambiar nombre de sesión

Sesión\_name("nombre\_session");

Eje: sesión\_name("CON");// nos muestra en consola, almacenamiento en cookies

damos la bienvenida </> Elementos Consola Fuentes Red Rendimiento Memoria Aplicación Recursos para desarrolladores Consola de red +

Filtro

Nombre	Valor	Domain	Path	Expires / Max-Age	Tamaño	HttpOnly
CON	1odccpvkb3ocrlld86...	localhost	/	Sesión	29	

- modificar ID

session\_id("PHP");

Nombre	Valor	Domain	Path	Expires / Max-Age
CON	PHP	localhost	/	Sesión

- Obtener id de la sesión

Dentro del código html del archivo contador.php

<?php echo session\_id(); ?>

Muestra en la página: PHP

- Variables de sesión: contador.php

```

• <?php
•     session_name("CON");
•     session_start();
•
•     if(isset($_SESSION['contador'])){
•         $_SESSION['contador']++;
•     }else{
•         $_SESSION['contador']=1;
•     }
•     ?>
•     <!DOCTYPE html>
•     <html lang="es">
•     <head>
•         <meta charset="UTF-8">
•         <meta name="viewport" content="width=device-width, initial-
scale=1.0">
•         <title>Sesiones en PHP</title>
•     </head>
•     <body>
•         <?php echo "Har recargado esta pagina ".$_SESSION['contador'].
•             " veces"; ?>
•         <br><br>
•         <a href="index.php">Inicio</a>
•         <br>
•         <a href="cerrar.php">Eliminar Sesion</a>
•
•     </body>
•     </html>
•

```

Para leer la variable de sesión:

```
<?php echo "Har recargado esta pagina ".$_SESSION['contador'].
    " veces"; ?>
```

En el index.php se incluyo

```
<?php
    session_name("CON");
    session_start();
```

```
?>
```

Antes del formulario se incluyó código PHP para mostrar la sesión:

```
<body>

<?php echo "Has recargado esta página ".$_SESSION['contador']. "
    veces"; ?>
<form action="login.php">
    <label>Usua. . .
```

11:19

Como las variables de sesión se pueden utilizar en cualquier archivo que tenga la sesión iniciada por eso estas se utilizan para realizar login o identificación con contraseña en los sitios que lo requieran

Vamos a cerrar el contador.php y vamos a realizar un login desde el index

Index.php

```
<?php
    session_name("LOGIN");
    session_start();
?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sesiones en PHP</title>
</head>
<body>
    <form action="login.php">
        <label>Usuario</label>
        <input type="text" name="usuario">
        <br>
        <label>Clave</label>
        <input type="password" name="clave">
        <br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

Login.php

```
<?php
    if($_POST['usuario']=="Carlos" && $_POST['clave']=="1234"){
        session_name("LOGIN");
        session_start();
        $_SESSION['Nombre']="Carlos";
        $_SESSION['Apellido']="Carlos";
        $_SESSION['Pais']="El salvador";

        echo "Sesión Iniciada";
```

```

        }else{
            echo "datos incorrectos";
        }
    ?>

```

### Contador.php

```

<?php
    session_name("LOGIN");
    session_start();
?>

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sesiones en PHP</title>
</head>
<body>
    <?php echo "Hola".$_SESSION['nombre']."'"; ?>
    <br><br>
    <a href="cerrar.php">Eliminar Sesion</a>
</body>
</html>

```

Para eliminar la sesión

Desde el archivo cerrar.php

```

<?php
session_name("LOGIN");
session_start();
session_destroy();

```

sesión\_destroy(): elimina todos los datos de sesión que se crean incluidas las variables de sesión

//session\_unset()//elimina solo las variables de Sesión

Después ya no servirán las variables en ningún archivo hasta que se entre de nuevo por el index.php y se cree una nueva sesión en el login.php

//unset(\$\_SESSION['Nombre'])//se elimina solo la Variable de sesion

### VIDEO 50 DIFERENCIA ENTRE COOKIES Y SESSIONES

#### Diferencia entre cookies y sesiones en PHP

COOKIES

SESIONES

Se almacena la **información en local**, es decir en el navegador.

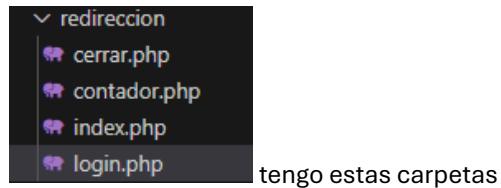
Se almacena la **información en el servidor**.

Se pueden **editar o borrar**

Es mas seguro que las cookies.

## VIDEO 51

### REDIRECCIONAR PAGINAS PHP Y JAVA SRIP



Ahora desde login voy a enviar a contador

```
<?php
    if($_POST['usuario']=="Carlos" && $_POST['clave']=="1234"){
        session_name("LOGIN");
        session_start();
        $_SESSION['Nombre']="Carlos";
        $_SESSION['Apellido']="Alfaro";
        $_SESSION['Pais']="El salvador";
        header("Location: contador.php");//funciona si cumple con:
    }else{
        echo "datos incorrectos";
    }
?>
```

Header: se puede utilizar y no genera error si dentro de nuestro código no existe ningún fragmento html

Si existiera un (echo “texto”) generaría error (hice la prueba con un (echo) pero igual siguió funcionando)

Cuando entramos a eliminar sesión vamos a hacer que redireccione por medio de java script a la ubicación del index.

```
<?php
session_name("LOGIN");
session_start();
session_destroy();
echo "<script>window.location.href='index.php'; </script>";
```

para mejorar el código se utilizó la siguiente condicional

```
if(headers_sent()){
    echo "<script>window.location.href='contador.php'; </script>";
}else{
    header("Location: contador.php");
}
```

Si en el programa ha habido un encabezado (headers\_sent()): me dice si se han utilizado etiquetas html) hace el reenvío por JavaScript si no lo hace por la función header

## Video 52

### Expresiones regulares

Validar formularios tanto en fronted como el backend

#### ¿Qué son las expresiones regulares en PHP?

- Son “una serie de caractéres que definen un patrón de búsqueda”.

- Las expresiones regulares son patrones que usaremos para encontrar una o varias combinaciones de caracteres en un texto.

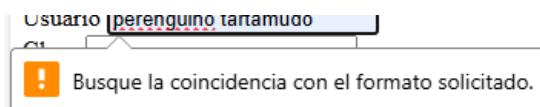
- En el área de **la programación**, las **expresiones regulares son un método** por medio del cual se pueden **realizar búsquedas** dentro de **cadenas de caracteres**. Sin importar la amplitud de la búsqueda requerida de un patrón definido de caracteres, las expresiones regulares proporcionan una solución práctica al problema.

### Validacion en fronted

Para la validación en html a los input les agregamos la expresión regular modificamos el index.php de la carpeta validacion

```
<input type="text" name="usuario" pattern="[a-zA-Z]{3,10}">
```

El pattern o patron nos indica en los corchetes que solo pueden haber letras minúsculas y mayúsculas y en las llaves el tamaño que va de 3 a 10



Si no cumple con eso muestra lo siguiente

```
<input type="text" name="usuario" pattern="[a-zA-Z]{3,10}" maxlength="10">
```

Con el (maxlength) limitamos el máximo de caracteres permitidos a 10 o al numero que se establezca en el pattern

### Validación en backend

En el archivo login.php de la carpeta de validación

```
if(!preg_match("/^a-zA-Z){3,10}$/ ",$_POST['usuario'])){
    echo "El usuario no coincide con el formato solicitado";
    exit();
}
```

La función preg\_match compara el formato de la expresión regular con la que llego en la variable usuario recibida por el method POST, si coincide arroja true pero como queremos que continue con el programa en caso de coincidir la negamos para que en caso de que no coincidan arroje true y entre a enviar el mensaje y terminar el programa

En php la expresión regular se encierra entre comillas y barras oblicuas y el inicia de la expresión lo define el símbolo de elevación y lo termina el símbolo de pesos ("/^expresión...\$/")

Para probar entramos en el index y modificamos por medio de inspeccionar el código html eliminando el pattern y maxlength

Me salió a mi este mensaje:

El usuario no coincide con el formato solicitado

11:22

me puedes explicar esta expresión regular: ("/^@[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\$/", \$email))

¡Claro! Vamos a desglosar esta expresión regular que parece estar validando direcciones de correo electrónico:

/^@[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\$/

1. **/ al principio y al final:** Estas barras diagonales delimitan la expresión regular en lenguajes como PHP o JavaScript. No son parte de la lógica en sí.
2. **^ al principio:** Indica el inicio de la cadena que se va a evaluar. Todo debe coincidir desde el principio.
3. **[a-zA-Z0-9.\_-]+:**
  - Esto define los caracteres permitidos para la parte inicial del correo (antes del @).
  - a-zA-Z permite letras en mayúsculas y minúsculas.
  - 0-9 permite números.
  - .\_- permite los caracteres punto (.), guion bajo (\_) y guion medio (\-) **EL GUION MEDIO SOLO NO SE PUEDE COLOCAR TIENE QUE SER CON LA BARRA OBLICUA PARA QUE ENTIENDA QUE ES UN CARÁCTER PERMITIDO**
  - El signo + indica que debe haber al menos uno de estos caracteres, pero puede haber más.
4. **@:** Representa el carácter obligatorio @ que separa el nombre del usuario de su dominio en una dirección de correo.
5. **[a-zA-Z0-9.-]+:**
  - Similar al grupo anterior, pero aquí se define la parte del dominio (después del @).
  - Permite letras, números, puntos (.) y guiones (-).
  - Al igual que antes, el + requiere al menos un carácter, pero permite múltiples.

6. \.: Representa un punto literal (.). Se escapa con una barra invertida (\) porque, en expresiones regulares, el punto sin escapar significa "cualquier carácter".
7. [a-zA-Z]{2,4}:
  - o Define la extensión del dominio (como com, org, net, etc.).
  - o a-zA-Z permite letras en mayúsculas y minúsculas.
  - o {2,4} especifica que debe tener entre 2 y 4 caracteres.
8. \$ al final: Indica el final de la cadena. Todo lo que precede debe coincidir completamente para que la validación sea correcta.

#### **¿Qué hace en general esta expresión?**

Esta expresión regular verifica si una cadena de texto es una dirección de correo electrónico válida con las siguientes características:

- Comienza con letras, números o ciertos caracteres especiales (.\_-).
- Tiene un carácter @.
- Sigue con un dominio compuesto de letras, números, puntos y guiones.
- Termina con una extensión de dominio entre 2 y 4 letras (como .com, .org, .net).

**Ejemplo de correo válido:** [ejemplo.usuario-123@dominio.co](mailto:ejemplo.usuario-123@dominio.co)

