IAPWEB

TEMA 4 - NOCIONES AVANZADAS SOBRE EL LENGUAJE PHP



EMPEZAR >

Powered by genially EDUCATION

< 00 > Funciones

< 01 > Arrays

< 02 > Strings

< 03 > Cifrado

< 04 > Expresiones regulares

< 05 > Ejercicios



Powered by genially EDUCATION

INTRODUCCIÓN. PROGRAMACIÓN MODULAR

Cuando necesitamos escribir muchas líneas de código para crear una aplicación, nos damos cuenta de que sin más elementos en el lenguaje que los vistos en la unidad anterior, el problema se puede volver tan complejo que puede ser inabordable.

Para mitigar ese problema apareció en los años 60 la **programación modular**. En ella, el código se divide en módulos de tamaño manejable. Cada módulo realiza una función muy concreta y así, el programador se centra en esa tarea y evita la complejidad de tener que resolver el problema completo.

En definitiva, se trata de concentrar los esfuerzos en resolver problemas sencillos y luego ensamblar todos los módulos de forma apropiada para resolver el problema completo (paradigma **divide y vencerás**).

En PHP la programación modular se implementa mediante **funciones**. Una función es código que recibe unos datos y, tras operar con ellos, puede o no devolver un resultado. Los datos que necesita la función para realizar su tarea se llaman **parámetros**. Una función no siempre requiere parámetros.

Una función se puede utilizar las veces que haga falta sin necesidad de volver a programarla, incluso desde otras aplicaciones. Un conjunto de funciones útiles creadas por un programador y que se utilizan en diferentes aplicaciones, forman lo que se conoce como **librería**.

Powered by genially EDUCATION

FUNCIONES EN PHP

En PHP, las funciones tienen esta forma de trabajar:

1. Las funciones poseen un nombre o **identificador** que cumple las mismas reglas que utilizamos para el nombre de las variables, pero sin utilizar el símbolo **$**. Se aconseja que el nombre de las funciones se escriba en minúsculas. El identificador siempre va seguido de paréntesis.

2. Al definir una función, entre paréntesis se indican los parámetros que esta necesita para funcionar.

3. Tras la declaración de los parámetros se escribe el código de la función, en el que se implementa lo que la función debe ser capaz de realizar.

4. Las funciones pueden devolver un valor, resultado del trabajo de la misma, mediante el comando **return**.

Php incorpora muchas funciones ya creadas para trabajar (como **print** o **echo**).
Por ejemplo, este código:

echo rand(1,10);

escribe un número aleatorio entre 1 y 10.
En el ejemplo anterior, **rand** es el identificador y **1** y **10** los parámetros.

Powered by genially EDUCATION

DECLARACIÓN Y USO DE FUNCIONES PERSONALES

Para crear nuestras propias funciones tenemos que seguir esta sintaxis:

```
function nombreDeLaFuncion(listaDeParámetros){
    código de la función
}
```

Powered by genially EDUCATION

ALCANCE DE LAS VARIABLES

Las variables definidas en una función, al finalizar la función se eliminan. Es decir, su ámbito es **local** a la función. Ejemplo:

```
function f1(){
      $h=9;
}
echo $h;
```

¿Qué crees que se muestra por pantalla?

Powered by genially EDUCATION

PASO DE PARÁMETROS POR REFERENCIA

¿Qué crees que se muestra por pantalla?

```
function f1($x){
        $x=9;
}
$x = 7;
f1($x);
echo $x;
```

Powered by genially EDUCATION

PASO DE PARÁMETROS POR REFERENCIA

NO FUNCIONA

¿Por qué?

Powered by genially EDUCATION

PARÁMETROS PREDEFINIDOS

Se puede indicar un valor por defecto a los parámetros a fin de permitir que dicho parámetro sea opcional, de modo que, si se pasa el parámetro se toma el valor que se pasa y si no, se toma el valor por defecto.

Ejemplo:

```
function potencia($base, $exponente = 2){
    $valor = 1;
    for ($i = 1; $i <= $exponente; $i++){
        $valor *= $base;
    }
    return $valor;
}
echo potencia(2,3); //Escribe 8, es decir 2^3
echo potencia(4);  //Escribe 16, es decir 4^2. La función toma el valor por defecto 2
```

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

VARIABLES GLOBALES

Las variables globales permiten ser utilizadas en todo el código, aunque se hayan definido dentro de una función. Ejemplo:

```
function encadenar($caracter){
      global $texto;
      $texto .= $caracter;
}
$texto = "Hola";
encadenar("!");
echo $texto; //Escribe: Hola!
encadenar ("?");
echo $texto; //Escribe: Hola!?
```

Powered by genially EDUCATION

VARIABLES ESTÁTICAS

Son variables locales a la función, pero que son capaces de recordar su valor entre cada llamada a la función. Ejemplo:

```
function estatica( ){
        static $contador = 0;
        $contador++;
        echo "Llamada número $contador </br>";
}
for ($i=1; $i<=5; $i++){
        estatica();
}
```



Powered by genially EDUCATION

RECURSIVIDAD

La recursividad es una técnica de creación de funciones pensada para solventar problemas complejos. Se basa en que, dentro del código de la función, ésta se invoca a sí misma. En cada llamada a la función, ésta resuelve parte del problema, de manera que, con cada nueva llamada, el problema se va volviendo más sencillo, hasta llegar a una llamada en la que la función devuelve un único valor. Tras esa última llamada, los resultados se encadenan hasta conseguir devolver el resultado final.

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION



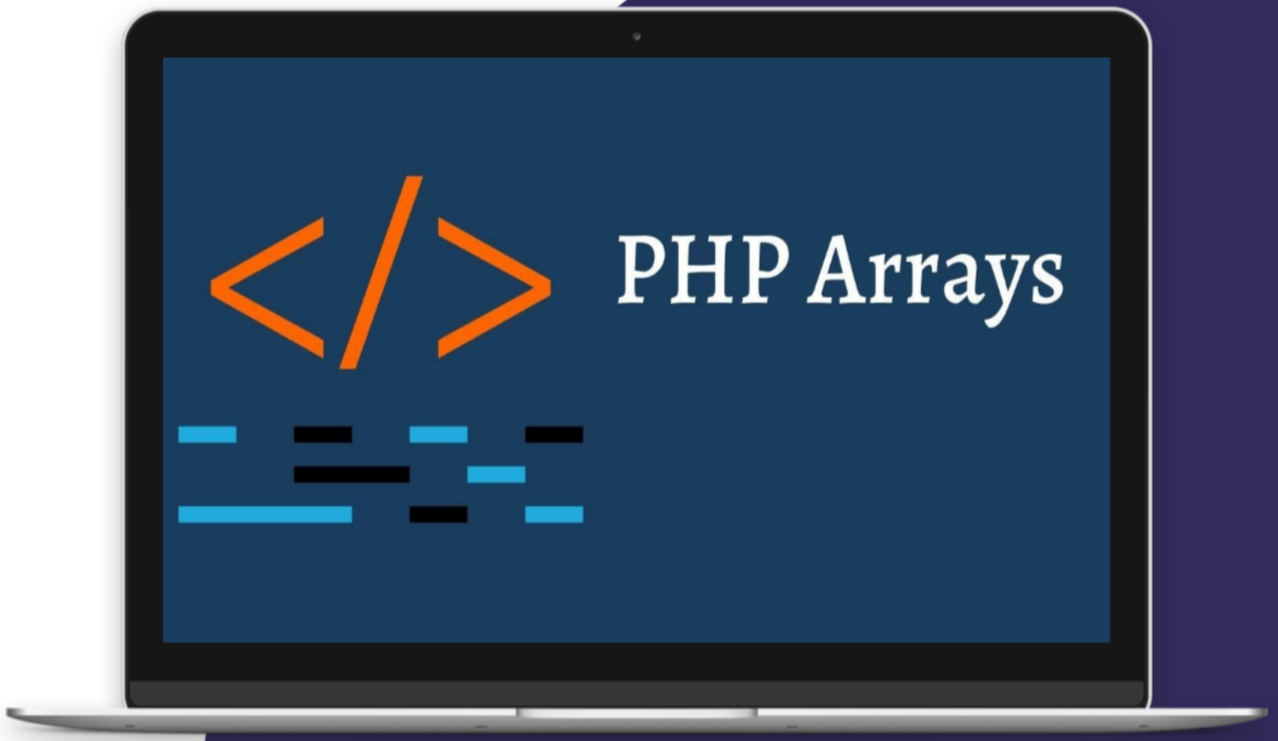
INCLUSIÓN DE FICHEROS

Dentro del código PHP se puede hacer uso de las instrucciones **include** y/o **require** para incluir el código de otro archivo. El archivo puede ser de cualquier tipo: **html**, **php** u otras extensiones. Esto permite la creación de archivos que contengan librerías de funciones o código reutilizable en múltiples páginas.

Powered by genially EDUCATION

<01>
ARRAYS

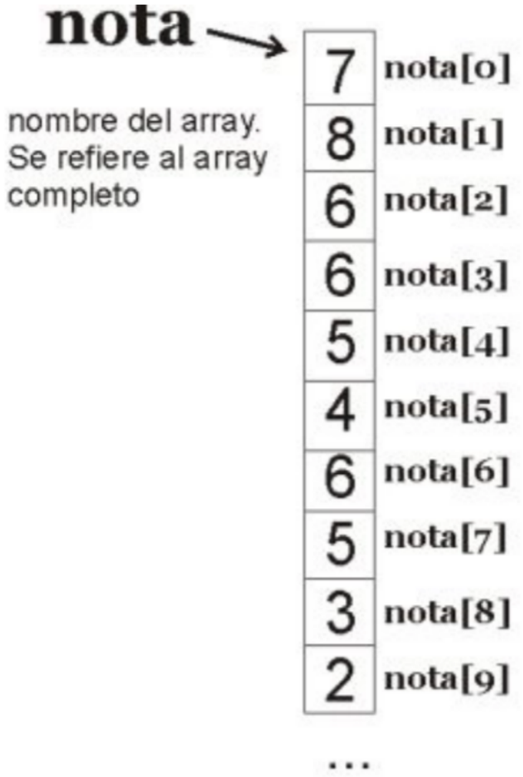
EMPEZAR >



Powered by genially EDUCATION

INTRODUCCIÓN

Imagina que queremos guardar las notas de 100 alumnos. Con lo que sabemos hasta ahora, necesitaríamos 100 variables, lo cual es tremendamente pesado de programar.



Por ello, existe una estructura llamada **array** que permite agrupar una serie de variables del mismo tipo. Con ella podemos referirnos a todos los elementos, pero también nos permite acceder individualmente a cada uno.

Un **array** es una colección de datos a la que se le asigna un identificador (por ejemplo **nota**). Para acceder a un dato de esa colección hay que utilizar su posición dentro del array. La posición es un número entero llamado **índice**, que empieza a contar desde el 0. Así, para acceder al tercer elemento del array, habría que utilizar la expresión **$nota[2]**.

Podemos distinguir entre dos tipos de arrays. Los **escalares** y los **asociativos**.

Powered by genially EDUCATION

ARRAYS ESCALARES

El array de la página anterior es un ejemplo de **array escalar**: un conjunto de valores a los que se accede a través de un índice que empieza a contar en 0. Así, por ejemplo, las notas se podrían almacenar de esta forma:

```
$nota[0] = 7;
$nota[1] = 8;
$nota[2] = 6;
...
```

Powered by genially EDUCATION

ASIGNACIÓN DE VALORES

En la página anterior vimos una manera de asignar valores a un array. Otra forma, más cómoda, es mediante la función **array()**. Dentro de esta función se colocan los valores que tendrá el array:

```php
$notas = array(5, 7, 2, 9);
//Es equivalente a
$notas[0] = 5;
$notas[1] = 7;
$notas[2] = 2;
$notas[3] = 9;
```

Powered by genially EDUCATION

ARRAYS ASOCIATIVOS

En este caso el índice es un texto que se comporta como una clave que permite acceder al dato. Los arrays así construidos forman estructuras de datos de tipo clave/valor, como ocurre con los arrays **$_GET** y **$_POST**. Ejemplo:

```
$nota["Antonio"]=5;
$nota["Luis"]=9;
$nota["Ana"]=8;
```

Powered by genially EDUCATION

FUNCIÓN COUNT

La función **count** sirve para saber el tamaño de un array. Es útil para usar un array del que previamente no sabemos su tamaño:

```
for ($i=0; $i<count($valor); $i++){
      echo $valor[$i]."</br>";
}
```

Powered by genially EDUCATION



Para evitar esto, podemos utilizar la función **isset**, que devuelve verdadero cuando a la variable que se le pasa como parámetro se le ha asignado ya un valor.
El código anterior quedaría así:

Powered by genially EDUCATION

BUCLE FOREACH

Recorrer arrays con elementos sin inicializar resulta muy complejo y pesado con los bucles **while** o **for**. Por ello, PHP dispone de un bucle muy potente que permite recorrer todos los elementos de un array sin importar si hay elementos indefinidos, ya que es capaz de ignorarlos; lo que lo hace idóneo para recorrer arrays asociativos.

Powered by genially EDUCATION

ARRAYS MULTIDIMENSIONALES

Un array multidimensional sirve para representar datos agrupados en una estructura que utiliza dos índices para representar cada elemento del array.

```
$nota[0][0]=9;
$nota[0][1]=7;
$nota[1][0]=6;
```

Powered by genially EDUCATION

RECORRIDO DE ARRAYS MULTIDIMENSIONALES

Volviendo al ejemplo anterior:

```php
//Es equivalente a
$nota=array(
    "Carlos"=>array(
        "Matemáticas"=>9,
        "Inglés"=>7,
        "Lengua"=>6
    ),
    "María"=>array(
        "Lengua"=>8,
        "Historia"=>6
    )
);
```

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

INSPECCIÓN DE ARRAYS MEDIANTE FUNCIONES DE RECORRIDO

Se trata de funciones basadas en el manejo de punteros de los lenguajes tradicionales. Utilizan un puntero virtual, que sería un objeto que señala a uno de los elementos del array y que al moverlo permite acceder al resto de elementos.

Esta técnica se basa en el manejo de las siguientes funciones:

FUNCIÓN	USO
current(array)	Devuelve el valor del elemento al que actualmente señala el puntero. Si no hay ninguno, devuelve false
key(array)	Devuelve la clave a la que señala el puntero
next(array)	Mueve el puntero al siguiente elemento del array y devuelve su valor. Si el elemento actual es el último, devuelve false
prev(array)	Mueve el puntero al elemento anterior del array y devuelve su valor. Si el elemento actual es el primero, devuelve false
reset(array)	Coloca el puntero en el primer elemento del array y devuelve su valor. Si no hay ningún elemento en el array, devuelve false
end(array)	Coloca el puntero en el último elemento y devuelve su valor

Powered by genially EDUCATION

INSPECCIÓN DE ARRAYS MEDIANTE FUNCIONES DE RECORRIDO

Ejemplo de recorrido de un array mediante estas funciones:

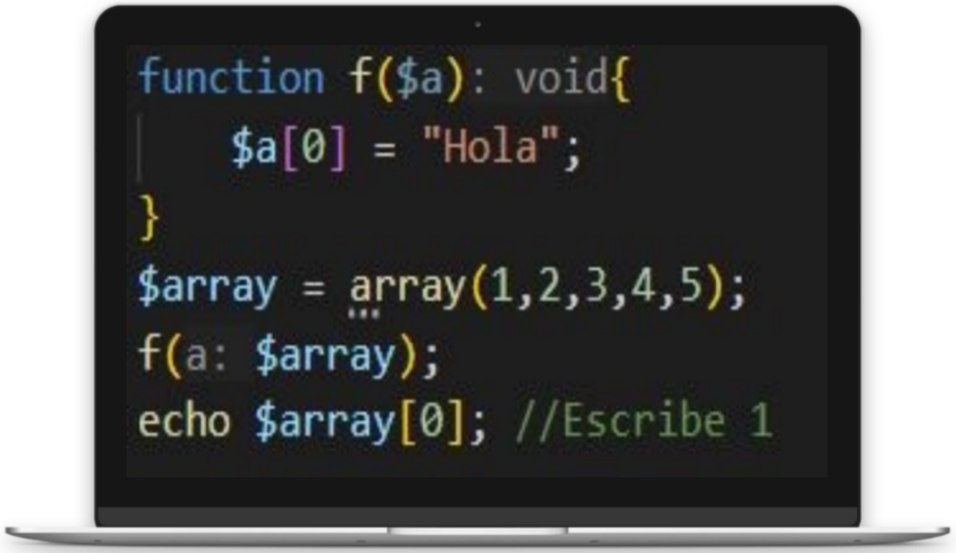
```php
$capital = array (
    "Castilla y León"=>"Valladolid",
    "Asturias"=>"Oviedo",
    "Aragón"=>"Zaragoza"
);
while (current(array: $capital)){
    echo "<strong>" . current(array: $capital) . "</strong></br>";
    next(array: &$capital);
}
```

¿Y para recorrerlo al revés?

Powered by genially EDUCATION

FUNCIONES Y ARRAYS

Las funciones pueden recibir y devolver arrays al igual que cualquier otro tipo de variables. En PHP, a diferencia de la mayoría de lenguajes estructurados, los arrays se pasan **por valor** a las funciones. Es decir, cuando se pasa como parámetro un array a una función, el parámetro genera una copia del mismo. Así, las modificaciones al array que se hagan dentro de la función **no afectan al array original**.
Ejemplo:



Powered by genially EDUCATION

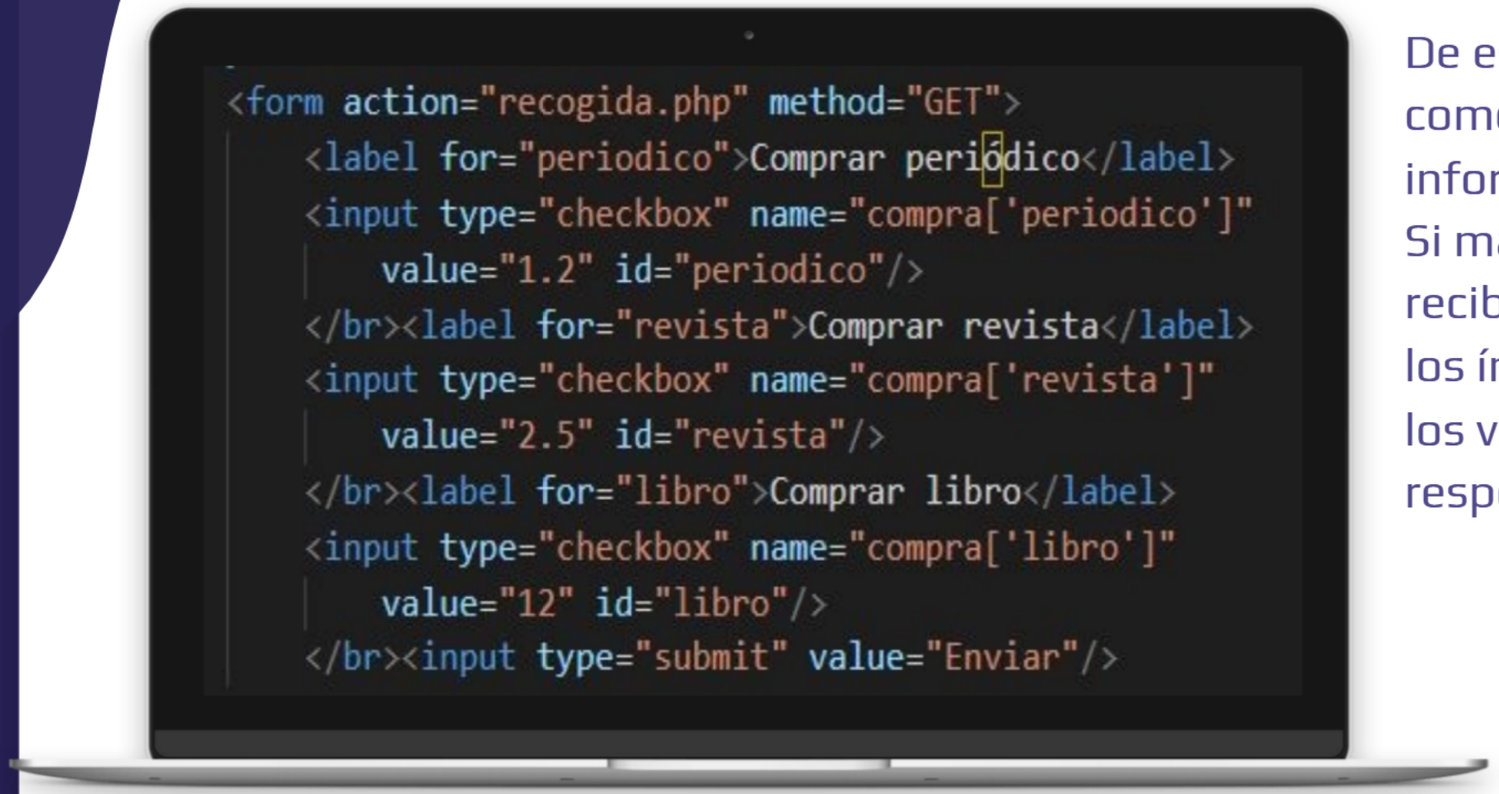
USO DE ARRAYS EN FORMULARIOS

Una de las virtudes de un array es su capacidad de recoger en una sola variable valores procedentes de diferentes controles en un formulario. Para ello, a todos los elementos del formulario tenemos que asignarles el mismo nombre en su atributo **name**, seguido de los símbolos [ ] para indicar que se trata de un array.

Powered by genially EDUCATION

USO DE ARRAYS EN FORMULARIOS

Es posible utilizar índices concretos en el array al recoger valores:



```
<form action="recogida.php" method="GET">
    <label for="periodico">Comprar periódico</label>
    <input type="checkbox" name="compra['periodico']"
        value="1.2" id="periodico"/>
    </br><label for="revista">Comprar revista</label>
    <input type="checkbox" name="compra['revista']"
        value="2.5" id="revista"/>
    </br><label for="libro">Comprar libro</label>
    <input type="checkbox" name="compra['libro']"
        value="12" id="libro"/>
    </br><input type="submit" value="Enviar"/>
```

De esa forma, tanto el índice del array como el valor nos proporcionan información.
Si marcamos todas las opciones, se recibirá un array llamado **compra** con los índices **periodico**, **revista** y **libro** y los valores **1.2**, **2.5** y **12** respectivamente.

```
Array
(
    ['periodico'] => 1.2
    ['revista'] => 2.5
    ['libro'] => 12
)
```

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

FUNCIONES DE USO CON ARRAYS

Existen muchas funciones que nos permiten trabajar con arrays (para ordenarlos, buscar en ellos, manipular su contenido, etcétera). Algunas ya las conocemos, como **print_r(array)**, que sirve para escribir todo el contenido del array.
Pero hay muchas más, y algunas de ellas pueden ser muy útiles, por lo que conviene conocerlas.

Aquí tienes una muestra de varias funciones útiles

Powered by genially EDUCATION

<02> STRINGS

Algunos conceptos básicos

EMPEZAR >



Powered by genially EDUCATION

STRINGS

La palabra String en inglés significa **cadena**. Sin embargo, en realidad, por String se entiende una serie de caracteres, lo que normalmente llamamos simplemente **texto**. En toda aplicación informática los strings son el tipo de dato fundamental.
Vamos a desglosar todo el funcionamiento de los strings que permite PHP.

- ASIGNACIÓN DE STRINGS
- MANEJO DE STRINGS COMO ARRAYS DE CARACTERES
- CADENAS HEREDOC
- CONCATENACIÓN DE TEXTOS
- USO DE LLAVES
- CADENAS NOWDOC

Powered by genially EDUCATION

FUNCIONES DE USO CON STRINGS

Al igual que con los arrays, existen muchas funciones que nos permiten trabajar con cadenas (para compararlas, buscar en ellas, extraer texto, etcétera). Algunas ya las conocemos, como **strtolower(texto)**, que sirve para convertir todo el texto a minúsculas.
Pero hay muchas más, y algunas de ellas pueden ser muy útiles, por lo que conviene conocerlas.

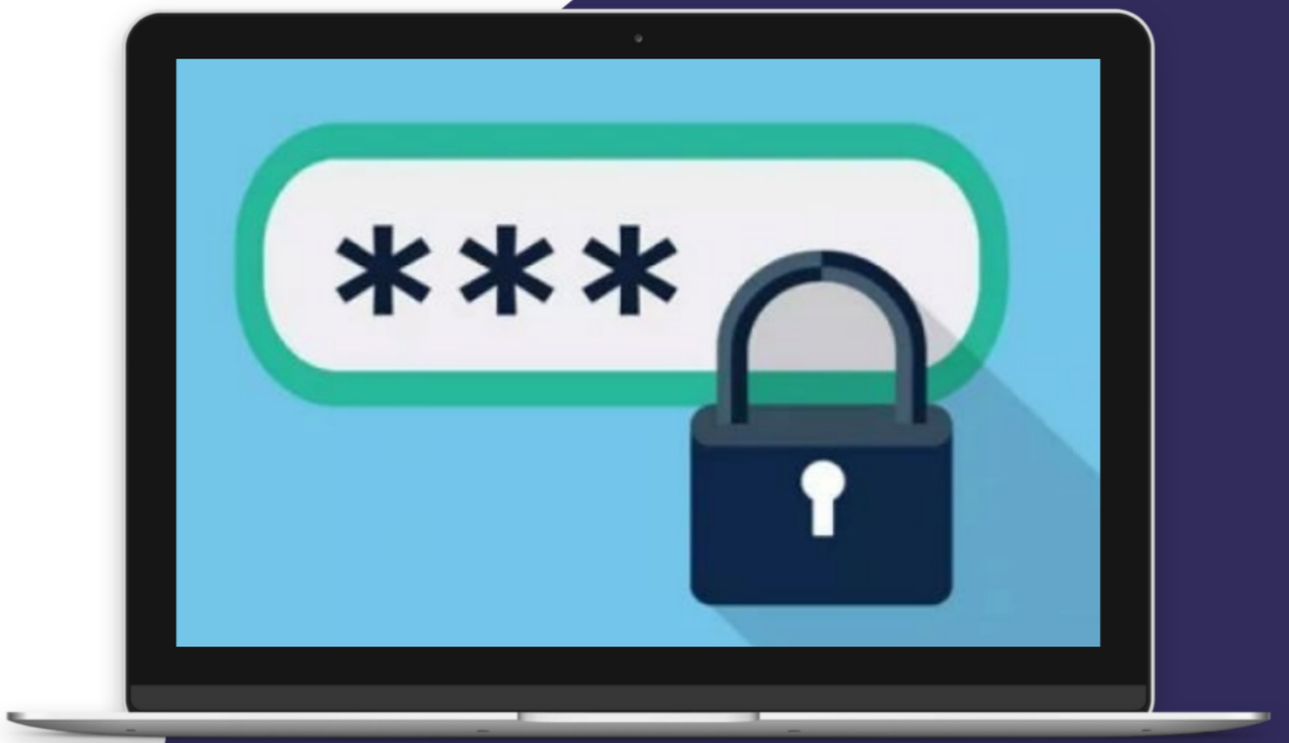
Aquí tienes una muestra de varias funciones útiles

Powered by genially EDUCATION

<03>
CIFRADO

Aprendamos a guardar contraseñas de forma segura

EMPEZAR >



Powered by genially EDUCATION

CIFRADO EN PHP

Cuando escribimos datos sensibles, como una contraseña, en un formulario, deseamos que esa información quede oculta a cualquiera que quiera espiar la comunicación entre nuestro servidor y el cliente.

Para ello, se deben utilizar funciones que convierten la contraseña en un texto **hash** obtenido a través de un algoritmo de cifrado.

Powered by genially EDUCATION

FUNCIÓN PASSWORD_HASH

Se trata de la función recomendada para el cifrado. Su sintaxis es:

password_hash (texto , algoritmo [,opciones])

Powered by genially EDUCATION

FUNCIÓN PASSWORD_HASH

La función retorna el texto hash resultante de aplicar las opciones.
Ejemplo:

```php
$hash = password_hash(password: "prueba", algo: PASSWORD_DEFAULT);
```



$2y$10$Mg7fugB3Yv5/R9Ab6zDDTODq01D4dxMoXi7VLpUaPLMX4twG7S.dC

Powered by genially EDUCATION

VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

<04> EXPRESIONES REGULARES

EMPEZAR >



Powered by genially EDUCATION

EXPRESIONES REGULARES

Se utilizan para establecer un patrón que permita validar el contenido de un texto. Podríamos usarlas, por ejemplo, para comprobar que el número de teléfono introducido en un formulario tiene 9 dígitos, o que el nombre empieza por mayúscula.

Powered by genially EDUCATION

EXPRESIONES REGULARES

SÍMBOLO	SIGNIFICADO	EJEMPLO	EJEMPLO DE TEXTO QUE LA CUMPLE	EJEMPLO DE TEXTO QUE NO LA CUMPLE
c	Un carácter cualquiera. Dentro de una expresión regular, obligaría a que aparezca ese carácter en la posición en la que se indica.	"/e/"	perro, jueves	gato, sol
cde	Siendo c, d y e caracteres, indica que esos caracteres deben aparecer en ese orden en la expresión.	"/mar/"	martes, amar	mamá, lunes
.	Cualquier carácter.	"/amig./"	amigo, amig3	amig
^x	Indica que el string debe comenzar por la expresión x	"/^a/"	ala, armario	perro
x$	Indica que el string debe terminar con la expresión x	"/a$/"	mosca, lámpara	perro, gato
x+	La expresión a la izquierda del + se puede repetir una o más veces	"/a+/"	a, aaa	" ", b

Powered by genially EDUCATION

EXPRESIONES REGULARES

SÍMBOLO	SIGNIFICADO	EJEMPLO	EJEMPLO DE TEXTO QUE LA CUMPLE	EJEMPLO DE TEXTO QUE NO LA CUMPLE
x*	La expresión a la izquierda del * se puede repetir cero o más veces	"/a*/"	" ", a, aaaaa	b
x?	La expresión a la izquierda del ? se puede repetir cero o una veces	"/a?/"	" ", a	aaa, b
x{n}	La expresión x debe aparecer n veces	"/a{3}/"	aaa	aa, aaaa
x{n,}	La expresión x debe aparecer n o más veces	"/a{3,}/"	aaa, aaaa, aaaaa	a, aa
x{m,n}	La expresión x debe aparecer entre m y n veces	"/a{2,4}/"	aa, aaa, aaaa	a, aaaaa
x|y	Las expresiones x e y son opcionales, se puede cumplir una u otra	"/a|b/"	abeto, gato, bebé	sol, perro

Powered by genially EDUCATION

EXPRESIONES REGULARES

SÍMBOLO	SIGNIFICADO	EJEMPLO	EJEMPLO DE TEXTO QUE LA CUMPLE	EJEMPLO DE TEXTO QUE NO LA CUMPLE
c-d	Cumplen esa expresión los caracteres que, en orden ASCII, vayan del carácter c al carácter d	"/a-m/"	a, c, m	o, r, s
[^x]	No es válido ninguno de los caracteres que cumplan la expresión x	"/[^abc]/"	sol, perro	gato, bebé, cama
\d	Vale cualquier dígito	"/\d/"	1, 2, 6	e, r
\D	Todo menos dígito	"/\D/"	a, $, !	3, 7
\w	Word, carácter válido dentro de los que PHP considera para identificar variables. Es decir, letras, números o guion bajo	"/\w/"	a, _, 2	!, %
\W	Todo lo que no sea un carácter de tipo Word	"/\W/"	!, %	a, _, 2

Powered by genially EDUCATION

EXPRESIONES REGULARES

SÍMBOLO	SIGNIFICADO	EJEMPLO	EJEMPLO DE TEXTO QUE LA CUMPLE	EJEMPLO DE TEXTO QUE NO LA CUMPLE
\s	Espacio en blanco	"/\s/"	" "	a
\S	Cualquier caracter menos espacio en blanco	"/\S/"	a, 5, !	" "
\n	Nueva línea			
\t	Tabulador			
\c	Permite representar el carácter c cuando este sea un carácter que no es representable de otra forma (como [,], /, \...).	"/\\/"	\	a
i	La letra i al final de los delimitadores permite ignorar mayúsculas y minúsculas	"/hola/i"	HOLA, hola, hOlA	

Powered by genially EDUCATION

EXPRESIONES REGULARES

SÍMBOLO	SIGNIFICADO	EJEMPLO	EJEMPLO DE TEXTO QUE LA CUMPLE	EJEMPLO DE TEXTO QUE NO LA CUMPLE
\p{xx}	Indica un carácter que cumpla la propiedad Unicode indicada con los códigos xx. Algunos códigos interesantes son los de las siguientes filas:			
L	Letra (incluye Ll y Lu)	"/[\p{L}]+/"	Hola, hola	Hol@, 123
Lu	Letra mayúscula	"/\p{Lu}/"	A, B, C	a, b, 4, !
Ll	Letra minúscula	"/\p{Ll}/"	a, b, c	A, 5, !
S	Símbolo	"/\p{S}/"	+, *	a, 1
alfabeto	Permite indicar un código de alfabeto	"/\p{Greek}/"	Ψ, Ω	a, b

Powered by genially EDUCATION

PROBLEMAS CON UNICODE

Observa este código:

```
$exp = "/^[\p{L}]+$/";
echo preg_match($exp, "España");
```

Debería dar verdadero, ¿no?

Powered by genially EDUCATION

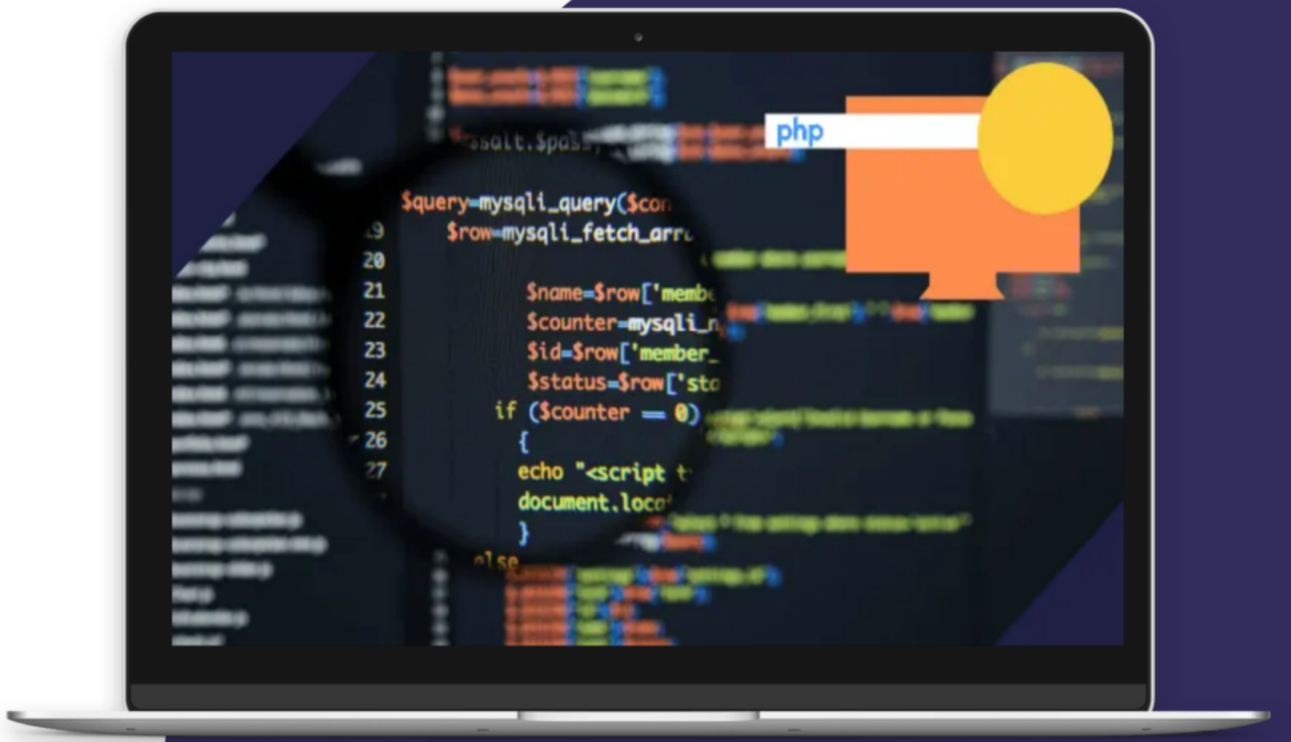
VAMOS A PONER EN PRÁCTICA LO APRENDIDO

EJERCICIOS >

Powered by genially EDUCATION

<05>
EJERCICIOS

EMPEZAR >



Powered by genially EDUCATION

EJERCICIOS

ENUNCIADOS

EJERCICIOS RESUELTOS

Powered by genially EDUCATION

BIBLIOGRAFÍA

Para la elaboración de esta presentación me he basado en contenidos de las siguientes fuentes:

- Jorge Sánchez Asenjo (2015). Implantación de Aplicaciones Web. Garceta
- mclibre.org. Programación web en PHP. https://www.mclibre.org/consultar/php/
- PHP de Venezuela. Manipulación de cadenas. https://micaelianomerida.wordpress.com/wp-content/uploads/2011/11/funciones-de-cadenas-php.pdf

Powered by genially EDUCATION

Tema 4 finalizado



Powered by genially EDUCATION