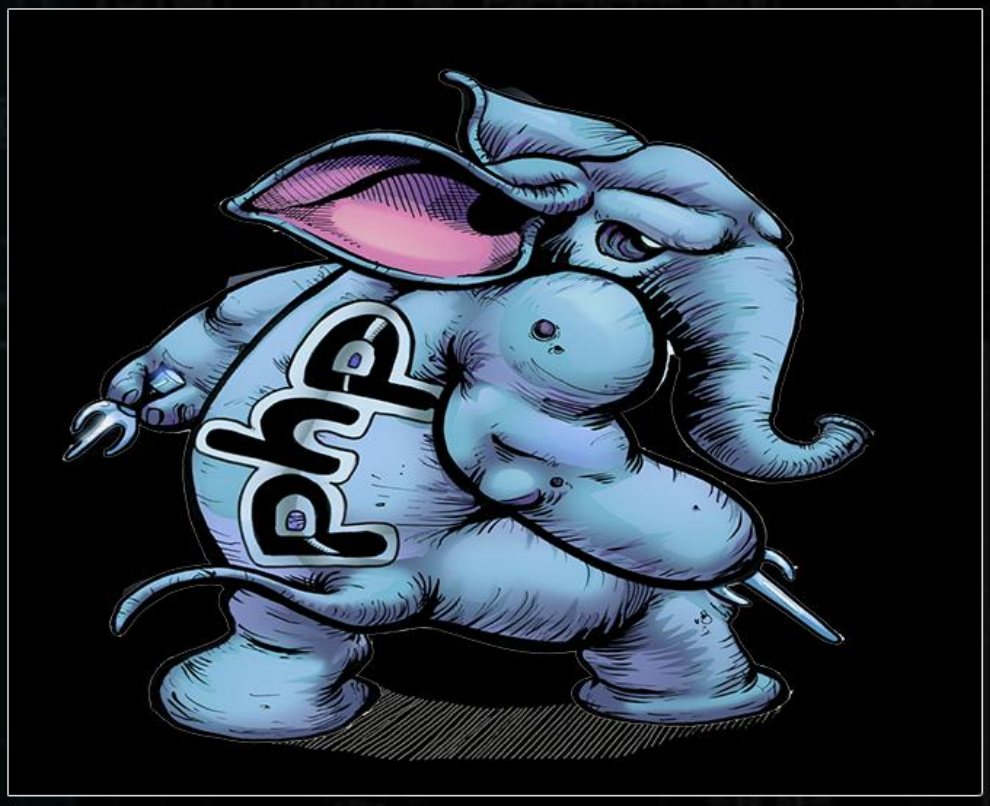
#2

UNDERCODE

TALLER DE PHP



TEMAS

FUNCIONES
PARAMETROS
ARRAYS
PASO DE VARIABLES
POST Y GET
SESIONES
Y MÁS!

TUTOR

DESTRUCTOR.PHP

FUNCIONES EN PHP

En la primera entrega de este taller se pudo apreciar la gran utilización de la función **echo**, esta función permitía imprimir un texto en pantalla y es la más útil cuando empezamos a programar PHP. Aún así, PHP nos ofrece un sin fin de funciones las cuales te permiten desde manejar y modificar archivos, hasta ingresar datos en una base de datos.

Este tipo de funciones se llaman de "caja negra" ya que sabemos lo que tenemos que ingresar y los que nos devuelve, pero no sabemos el procedimiento que hace.

Ahora, ¿es necesario aprenderse todas las funciones? La respuestas se un rotundo NO, es imposible, aún así contamos con una gran herramienta como lo es www.php.net, allí tendremos todas las funciones de PHP con su explicación, esto nos servirá por ejemplo para interpretar códigos en los cuales no sabemos alguna función, es solo buscar y listo.

Para los usuarios que no conocen la forma de trabajar con el lenguaje C, se hace necesario explicar cómo aparecen las funciones descritas en el manual de www.php.net.
El formato es el siguiente:

tipo_devuelto nombre_función (tipol argl, tipo2 arg2...) ,-

El tipo devuelto es el tipo de valor que la función dará como salida. Puede ser un integer, double, array, etcétera. El nombre de la función es la forma correcta de ejecutar el código. Los parámetros de entrada (argumentos) deben pertenecer al tipo que define la función. Un ejemplo de definición es:

string substr (string string, int start[, int length]) ;

Puede ver que la función substr () devolverá como resultado una cadena de caracteres y que tiene 2 argumentos de entrada como mínimo. El parámetro que se encuentra entre dos corchetes es opcional y puede decidir usarlo o no.

FUNCIONES CREADAS POR EL PROGRAMADOR

Ahora, **¿Por qué no poder crear nuestras propias funciones?**
Y es que PHP, como la mayoría de los lenguajes de programación lo permiten

Muchos se preguntaran **¿Para qué?**
Es simple, imaginemos que queremos ejecutar un mismo código muchas veces en nuestra página PHP, simplemente deberíamos llamar a la función y listo! Veamos un ejemplo:

Nuestra intención será ejecutar 3 echo seguidos en la página unas 3 veces, si no trabajáramos con funciones sería algo así:

```php
<?php
	echo 'hola';
	echo '¿Como';
	echo 'estas?';

	echo 'hola';
	echo '¿Como';
	echo 'estas?';

	echo 'hola';
	echo '¿Como';
	echo 'estas?';
?>
```

Ahora al trabajar con funciones quedaría algo así:

```php
<?php
	function saludar(){
		echo 'hola';
		echo '¿Como';
		echo 'estas? ';
	}
	saludar();
	saludar();
	saludar();
?>
```

Verán como nuestro código se ha reducido, ahora imaginemos que tengamos que imprimir lo mismo pero 8 veces, o hasta 15, la reducción del código será mucho mayor!

La estructura de una función es muy simple y la pueden apreciar muy fácilmente:

```
function nombre_funcion($argumentol, $argumento2, ..){
	instruccionl;
	instruccion2;
	instruccion3;
}
```

La palabra function va primero, luego el nombre de la función y entre paréntesis los argumentos, y entre las lleves que abren y cierran la función todas las instrucciones.
A la hora de llamar a la función simplemente se coloca el nombre y se paréntesis.

NOTA: Recuerde que PHP distingue entre mayúsculas y minúsculas

FUNCIONES CON PARÁMETROS

Más de uno podrá apreciar lo que yo llamé “parámetros”, estos parámetros son los que hacen versátil a la función y algo tan útil, imaginemos que queremos sumar dos números y multiplicarlos por diez un par de veces en el código, el proceso es el mismo, pero los números pueden cambiar, allí es cuando toman partido los parámetros!
Veamos un ejemplo y aclaremos:

```
<?php
	function suma($numero1, $numero2){
		$suma = $numero1 + $numero2;
		$suma = $suma * 10;
		return $suma
	}
	$suma1 = suma(2, 4);
	$suma2 = suma(8, 10);
?>
```

El código continúa siendo muy simple, entre los paréntesis de suma declaramos los parámetros, estos tienen que ser declarados como una variable y serán usados como ella dentro del código.

Este código no imprimirá nada en pantalla, ya que no contiene un echo, lo que si tiene es un return, esto hará que la función retorne el resultado, haciendo que la variable adquiera ese valor.

Siendo más claros, a la variable suma1 se le asignará como valor, la suma de 2 + 4, multiplicada por 10, en otras palabras, suma1 valdrá 60, mientras que suma2 valdrá 180

IMPORTANTE: Las funciones cuando se llaman pueden recibir una variable como parámetro, podríamos cambiar $suma1 = suma(2,4); por algo así:

```
$parametro1 = 2;
$parametro2 = 4;
$suma1 = suma($parametro1, $parametro2);
```

El resultado sería exactamente el mismo!
Además, es recomendable saber que cuando se quiere pasar un string por parámetro se tiene que pasar entre comillas

PARAMETROS CON VALOR POR DEFECTO

Es muy importante saber que si declaraste una función con dos parámetros y le has enviado uno solo te dará un error, es para eso que PHP posee un método muy simple de evitar este tipo de errores: Parámetros con valor por defecto

Esto funciona muy fácil, al parámetro de la función se le agregara un valor por defecto, y en cambio de no recibir parámetro lo utilizará a este. Veamos un pequeño ejemplo con la función anterior:

```php
<?php
	function suma($numero1, $numero2 = 0){
		$suma = $numero1 + $numero2;
		$suma = $suma * 10;
		return $suma
	}
	$suma1 = suma(2);
?>
```

Al parámetro $numero 2 se le agregó 0 como valor por defecto. En este caso la variable suma1 tendrá un valor de 20, ya que no ingresó un segundo parámetro, entonces se le asignará la suma de 2 + 0 multiplicado por 10, que da como resultado 20

NOTA: Es muy importante saber que los valores por defecto se le tienen que ir agregando de derecha a izquierda, de modo que en ese caso $numero1 no podría tener un valor por defecto si $numero2 no lo tiene!

ARRAYS EN PHP

Un array es una colección de valores con un único nombre. Para acceder a los distintos valores de la variable se utiliza un índice numérico o alfanumérico. Podríamos ver un ejemplo así:

```
<?php
        $mi_array[0] = 23;
        $mi_array[1] = "Este valor es un string";
        $mi_array["ejemplo"] = "Esto es un array asociativo";
?>
```

De esta manera podríamos crear un array llamado estudiantes el cual contendría todos los nombres de los estudiantes de un liceo y que tendrían un indice que lo identificaría

NOTA: Es importante saber que los arrays numéricos comienzan en 0

CREACIÓN DE ARRAYS

El camino más simple y, por otro lado lógico, es asignar valores cuando se necesiten. La primera vez que asignemos un valor, el array se creará en el entorno:

```
<?php
        $mi_array[l] = 23; // Asignación directa
?>
```

De esta forma tenemos un valor asignado al índice 1 del array. Puede asignar cualquier índice en la creación de este tipo de dato, e incluso no asignar ninguno, de forma que PHP se encarga de asociar un índice distinto para cada valor.

```
<?php
        $mi_array[] = 23; // Empieza en el índice 0
        $mi_array[] = 54; // índice 1
?>
```

A parte de esto contamos con la función array la cual crea un array numérico con los datos que se le pasen por parámetro, veamos un ejemplo:

```
<?php
        $estudiantes = array("Marcelo", "Pablo", "Pedro");
?>
```

De esta manera tendríamos un array estudiantes, Marcelo tendría como índice 0, Pablo como índice 1 y Pedro como índice 2. Esta función también permite darle un índice a cada valor gracias al operador “=>”

```
<?php
        $estudiantes = array( 1 => "Pablo",  0 => "Marcelo", 3 => "Pedro");
?>
```

Esto tendría el mismo resultado que el anterior. Además, esto permite crear arrays con indices alfanuméricos como este caso:

```
<?php
	$estudiantes = array( "bueno" => "Pedro", "medio" => "Marcelo", "malo" => "Pablo");
?>
```

De esta manera calificaríamos a los alumnos según su rendimiento también.

IMPORTANTE: Para obtener el valor de un array se utiliza el indice dentro de los corchetes. Ej:

```
<?php
	$estudiantes = array( "bueno" => "Pedro", "medio" => "Marcelo", "malo" => "Pablo");
	echo $estudiantes["bueno"];  //imprime: Pedro
?>
```

INTERACTUAR CON ARRAYS

En la entrega anterior vieron estructuras de control simples pero no llegaron a ver todas, para interactuar con arrays hay una muy útil llamada “foreach”, el uso es así:

```
<?php
	$ciudades = array ("Badajoz","Mérida","Cáceres","Plasencia") ;
	foreach ($ciudades as $valor) {
		echo ("El valor es $valor<br>");
	}
?>
```

El resultado de este script sería algo así:

Badajoz
Mérida
Cáceres
Plasencia

La construcción anterior recorre el array desde el principio. En el ejemplo se puede ver que foreach toma el array a recorrer y sus valores los va almacenando en la variable $valor a medida que el bucle se ejecuta.

Existe una segunda construcción que permite recuperar el índice y el valor. Veamos un ejemplo de esto:

```
<?php
	$ciudades = array ("Badajoz","Mérida","Cáceres","Plasencia");
	foreach ($ciudades as $indice => $valor) {
		echo ("El índice $indice tiene el valor: $valor<br>") ;
	}
?>
```

El resultado sería así:

El indice 0 tiene el valor: Badajoz
El indice 1 tiene el valor: Mérida
El indice 2 tiene el valor: Cáceres
El indice 3 tiene el valor: Plasencia

PASO DE VARIABLES

Hasta ahora hemos aprendido a crear algoritmos gracias a estructuras de control, la utilización de variables, funciones de PHP que nos ayudarán, y nuestras propias funciones para crear un código más corto. Aún así, no hemos aprendido algo tan fundamental como recibir valores. La forma más simple de recibir valores en PHP es gracias a los formularios, para ello disponemos de dos métodos POST y GET

El método de traspaso de datos se especifica al crear un formulario en HTML, un ejemplo sencillo de un formulario que envíe sus datos por post sería algo así:

```
<html>
	<head></head>
	<body>
		<form id="formulario" action="proceso.php" method="post">
			//aqui el cuerpo del formularios
		</form>
	</body>
</html>
```

Seguro se preguntarán a que se refiere action, esto indica a donde se enviarán los datos del formulario, en caso de querer enviarle los datos a la misma página sería algo como action=”#” .

MÉTODO POST

El método POST es muy sencillo y se entenderá con un simple ejemplo:

```
<html>
	<head></head>
	<body>
		<?php
			$nombre = $_POST["nombre"];
			echo $nombre;
		?>
		<form id="formulario" action="#" method="post">
			<input type="text" name="nombre">
		</form>
	</body>
</html>
```

Este es un ejemplo de autollamada ya que action es igual a “#”, por lo tanto los datos del formulario se enviarán a la misma página. Además, se ve como se especifica el method como post.

Lo más importante de aquí es notar la utilización de $_POST[“nombre”] para recibir el nombre, lo único que hay que saber es que lo que va dentro de los corchetes es el name que se le da al input del formulario de donde queremos extraer el valor.

El resto es simple, recibe el valor del nombre que se ingresa en el input mediante POST y lo imprime en pantalla

NOTA: Es importante saber que este archivo deberá tener la extensión .php cuando sea guardado, un ejemplo sería formulario.php

MÉTODO GET

El uso de GET es muy parecido al de POST, podríamos recrear lo mismo que el formulario que hicimos como muestra de POST pero en GET y quedaría algo tan parecido como esto:

```
<html>
	<head></head>
	<body>
		<?php
			$nombre = $_GET["nombre"];
			echo $nombre;
		?>
		<form id="formulario" action="#" method="get">
			<input type="text" name="nombre">
		</form>
	</body>
</html>
```

Notarán que la única diferencia es el valor de method y la utilización de $_GET para recibir los valores en lugar de $_POST

Aún así existe una gran diferencia entre ambas y es que los datos enviados por GET se envían a través de la URL, mientras que cuando se envía por POST el envío no es visible. Esto hace que cuando queramos enviar datos privados, como la contraseña de un usuario que se acaba de registrar, utilizaremos POST, mientras qué cuando queramos enviar datos como el nombre de user del cual queremos acceder al perfil utilizaremos GET.

La gran ventaja de GET es esa misma, ya qué nos permite trabajar manualmente, veremos el ejemplo del user a cual queremos acceder al perfil y mostraré tres formas de acceder mediante HTML:

Imaginemos que tenemos una página llamada perfil.php que recibe un usuario por GET con el name user y desde allí te muestra todos los datos de ese usuario en el perfil. Mediante un formulario el acceso sería algo como:

```
<form id="formulario" action="perfil.php" method="get">
        <input type="text" name="user">
</form>
```

Esa es la que ya conocemos, a su vez, el usuario podría ingresar desde su navegador a

www.nuestrapagina.com/perfil.php?user="usuarioquequierabuscar"

Esto es algo fundamental, y es la forma en la que se envían los datos GET a través de la URL, primero se ingresa el sitio web, luego el signo "?" y por último el name del campo al que correspondería, en este caso seria user y su valor correspondiente que se une con un signo de igual. Es importante saber que si quisiéramos buscar un user donde su nombre fuera un número no se utilizaría comillas.

En caso de que se quisiera enviar más de un dato a través de GET se utilizaría &&
Veamos un ejemplo para mostrar su utilización:

www.pagina.com/archivo.php **?** user="pepe" && edad=33

En esta ocasión mostré todo por separado para que sea más claro apreciar. Se puede ver como se utilizo un número sin comillas también. En caso de seguir queriendo agregar más datos se seguirían agregando gracias al uso de &&

Por último, la otra manera de acceder sería gracias a nuestro ingenio como programadores, haciendo uso de todo esto. Podríamos crear un enlace, que nos redirija a la página perfil.php donde el usuario sea un usuario en particular:

**Ingrese al perfil de Pepe **

NOTA: Se puede ver como utilicé comillas simples luego de href facilitando el uso de comillas dobles dentro de ellas

SESIONES

Durante todos los capítulos siempre hemos trabajado con variables que persisten en una sola página y en caso de querer trabajar con variables de otros archivos necesitaríamos el traspaso de ellas por GET o por POST. Estos métodos, aunque útiles, no son todo lo prácticos que podrían en determinados casos en los que la variable que queremos conservar ha de ser utilizada en varios scripts diferentes y distantes los unos de los otros.

Ante ello, se nos vuelve necesario el poder trabajar con variables que puedan ser reutilizadas un sin fin de veces en una misma sesión.

Pensemos en una red social, en donde constantemente se trabaja con el nombre del usuario que se ha logueado, sería un trabajo muy duro y prácticamente imposible ir pasando el nombre de usuario y sus datos por formularios constantemente.

Este tipo de situaciones son solventadas a partir de las variables de sesión. Una sesión es considerada como el intervalo de tiempo empleado por un usuario en recorrer nuestras páginas hasta que abandona nuestro sitio o deja de actuar sobre él durante un tiempo prolongado o bien, sencillamente, cierra el navegador.

PHP nos permite almacenar variables llamadas de sesión que, una vez definidas, podrán ser utilizadas durante este lapso de tiempo por cualquiera de los scripts de nuestro sitio. Estas variables son existentes para todos los visitantes pero son independientes entre si, de modo que mi variable de sesión idioma es distinta a la variable de sesión idioma de otro usuario que se está conectando al mismo tiempo que yo en otra parte del mundo.

TRABAJANDO CON VARIABLES DE SESIÓN

Cuando queremos utilizar variables de sesión en una página tenemos que iniciar la sesión con la siguiente función:

session_start()

IMPORTANTE: La sesión se tiene que inicializar antes de escribir cualquier texto en la página. Esto es importante y de no hacerlo así corremos el riesgo de recibir un error,

Una vez que hayamos inicializado la sesión podemos acceder a las variables gracias al array $_SESSION donde se accede al valor de las variables de la siguiente manera:

$_SESSION[“variable_de_sesion”] = “este es el valor de la variable de sesion”;

Pueden apreciar que se trabaja como con un array normal, teniendo como defecto que para obtener el valor de esa variable simplemente tendremos que ingresar lo siguiente:

$variable = $_SESSION[“variable_de_ sesion”];

En este ejemplo se le asigna a $variable el valor de la variable de sesión.

Ahora miremos un simple ejemplo:

```
<?
        session_start();
?>
<html>
        <head>
                <title>Variables de sesión</title>
        </head>
        <body>
                Declaro variable
                <?
                        $SESSION["nombre"] = "pepe"
                ?>
        </body>
</html>
```

Y en otra página:

```
<?
session_start();
?>
<html>
        <head>
                <title>Variables de sesión</title>
        </head>
        <body>
                <?
                Echo 'Hola'. $_SESSION["nombre"].'¿Como estas?'
                ?>
        </body>
</html>
```

La segunda página devolverá: Hola pepe ¿Como estas?

NOTA: Es importante primero acceder a la primer página así declaramos la variable de sesión

NOTA2: Podemos ver como se utiliza session_start() al principio de la página

Por último, es importante saber que para destruir una sesión sin necesidad de que el usuario cierre el navegador se puede utilizar la función session_destroy()

FINAL

Recordamos a todos que esto es una simple introducción a PHP y que siempre sigan investigando porque nunca terminarás de aprender.

Recomendamos a ustedes visitar la biblioteca digital de Underc0de desde aquí: http://biblioteca.underc0de.org/ y desde allí ingresar a la sección de PHP dentro de Programación para poder encontrar muchos libros que les ayudarán muchísimo