APUNTES PHP 2024/25

ÍNDICE

1. ¿QUÉ ES PHP?

Lenguaje de programación del lado del servidor. Ésto quiere decir que se ejecuta en el servidor antes de que la página sea enviada al navegador del usuario.

Algunas tareas que usan PHP:

- Procesar formularios
- Acceder y manipular bases de datos
- Autenticar usuarios
- Generar contenido dinámico

2. SINTAXIS BÁSICA DE PHP

2.1 Etiquetas de apertura y cierre de PHP:

El código PHP siempre debe ir dentro de las etiquetas de apertura y cierre:

```
<?php
// Código PHP aquí
?>
```

2.2 Comentarios:

Comentarios de una línea:

```
// Esto es un comentario de una línea
# Esto también es un comentario de una línea
```

Comentarios de múltiples líneas:

```
/*
Esto es un comentario de
múltiples líneas.
*/
```

3. VARIABLES

3.1 Declaración de variables:

Las variables se declaran usando el símbolo “$” seguido del nombre de la variable

```
$nombre = "Ana";  // Variable de tipo string
$edad = 25;  // Variable de tipo entero
```

3.2 Mostrar información de variables:

print_r($variable): Imprime el contenido completo de una variable, incluyendo la estructura interna.

- Si la variable es un array, muestra las claves y los valores y si es un objeto muestra las propiedades y sus valores.

```
$array = [
        "nombre" => "Carlos",
        "edad" => 35,
];
```

Se mostraría así:

```
Array
(
    [nombre] => Carlos
    [edad] => 35
)
```

var_dump($variable): Similar a print_r($variable) aún que más detallada. Muestra el tipo de dato, longitud y valor de una variable.

Utilizando el mismo ejemplo de antes, se mostraría asi:

```
array(2) {
    ["nombre"]=> string(6) "Carlos"
    ["edad"]=> int(35)
}
```

Implode($variable): Une los elementos de un array en una cadena, utilizando un separador específico que se define en los argumentos.

Sintaxis: *implode(string $separator, array $array) // Salida string*

```
$array = ["Carlos", "Ana", "Juan"];
$resultado = implode(", ", $array); // Imprime: Carlos, Ana, Juan
```

3.3 Tipado:

PHP es un lenguaje de tipado débil, no es necesario declarar el tipo de una variable antes de usarla ya que detecta automáticamente el tipo según el valor que se le asigna:

```
$nombre = "Ana";  // String
$edad = 30;  // Integer
$precio = 99.99;  // Float
$activo = true;  // Boolean
```

3.4 Constantes:

Las constantes se definen con la función define(), el nombre de la constante debe ir en mayúsculas y no pueden cambiar su valor:

```
define("NOMBRE", "Ana");
echo NOMBRE;  // Imprime: Ana
```

3.5 Variables de estado:
Almacenan y gestionan la información sobre el estado actual de un proceso, aplicación o sistema. Mantienen su valor a lo largo del tiempo mientras el usuario interactúa con una aplicación o durante la ejecución de un script.

Aunque algunas variables de estado, como $_SESSION o $_COOKIE, se consideran **superglobales** en PHP, no todas lo son.

$_SESSION permite almacenar información en el servidor mientras el usuario navega por el sitio, persistiendo sus datos entre diferentes páginas.

$_COOKIE almacena información en el navegador del cliente, permitiendo que algunos datos persistan entre sesiones, incluso después de que el usuario haya cerrado el navegador.

4. CONCATENACIÓN DE CADENAS
En PHP, las cadenas de texto se concatenan con el operador ".":

```
$saludo = "Hola, " . $nombre;
echo $saludo;  // Imprime: Hola, Ana
```

5. ESTRUCTURAS DE CONTROL

4.1 Condicionales:
Las condicionales permiten ejecutar código dependiendo de una condición:

```
if ($edad >= 18) {
        echo "Eres mayor de edad";
} else {
        echo "Eres menor de edad";
}
```

4.2 Funciones importantes relacionadas con condiciones:

- isset(): Comprueba si una variable está definida y no es "null"

```
if (isset($nombre)) {
        echo "La variable \$nombre está definida";
}
```

- empty(): Comprueba si la variable está vacia o no definida

```
if (empty($nombre)) {
        echo "La variable \$nombre está vacía o no existe";
}
```

6. ARRAYS Y COMO RECORRERLOS

Los arrays permiten almacenar múltiples valores en una sola variable.

6.1 Arrays numéricos

Utiliza índices numéricos para acceder a sus elementos:

```
$num = [10, 20, 30];
```

¿Cómo acceder a sus elementos?

```
- echo $num[0];  // Imprime: 10

- echo implode("<br>", $num); // Imprime cada valor seguido del valor
indicado entre " ", en este caso <br>.

- foreach ($num as $elementos){
        echo $elementos . " , " // Imprime 10 , 20 , 30
}
```

6.2 Arrays asociativos:

Utiliza claves (de texto o numéricas) en lugar de índices numéricos:

```
$persona = [
["nombre" => "Carlos", "edad" => 35],
["nombre" => "Ana", "edad" => 20 ],
];

$persona[0]["nombre"] // Imprime: Carlos
```

6.3 **Añadir elementos a un array**

Si es un array numérico, asigna el valor a la siguiente posición disponible al final.

```
$array[ ] = "Nuevo valor";  // Añade "Nuevo valor" al final
```

Por otro lado, si el array es asociativo y añadimos un nuevo valor, se debe indicar la clave a la que va asociado, si no, se añade sin estar asociado a nada.

```
$array = [
   "clave1" => "valor1",
   "clave2" => "valor2"
];

$array["clave3"] = "Nuevo valor"; // De esta manera, el valor “Nuevo valor” queda asociado a “clave3”.
```

6.4 Array multidimensionales

Contiene arrays dentro de otros arrays:

```
$ls_people [0]["name"]= "ana";
$ls_people [0]["ap"]= "garcia";
$ls_people [1]["name"]= "maria";
$ls_people [1]["ap"]= "lopez";
```

Este array contiene 2 elementos (índices 0 y 1), y cada uno de ellos es un array asociativo que incluye dos claves: "name" y "ap".

¿Cómo acceder a sus elementos?

```
- foreach ($ls_people as  $persona){
   echo $persona["name"] . " " . $persona["ap"] . "<br>";
}
```

En esta estructura, $persona representa cada elemento del array $ls_people **durante la iteración**.

En la primera iteración, **$persona equivale a $ls_people[0].** Por lo tanto, cuando se indica **$persona["name"], es equivalente a acceder a $ls_people[0]["name"]** en la primera iteración.

```
- foreach ($ls_people as $pos => $persona){
   echo $pos . " " . $persona["name"] . " " . $persona["ap"] . "<br>";
}
```

En este caso, se está accediendo tanto al índice ($pos), que **representa la posición** del elemento en el array $ls_people (0, 1, etc.), como al valor ($persona) asociado a ese índice, que es un array asociativo con las claves "name" y "ap".

7. PHP Y SUPERGLOBALES

Las superglobales son un conjunto de arrays predefinidos que están disponibles en todos los scripts de PHP. Se utilizan para almacenar y gestionar datos que provienen de diversas fuentes, como formularios, URLs, cookies, sesiones...etc.

Dos de las más importantes son:

- $_GET: Contiene los parámetros enviados en una URL

```
// URL: http://mi-sitio.com/pagina.php?nombre=Ana&edad=25

// Accedemos a los parámetros 'nombre' y 'edad' en la URL
$nombre = $_GET['nombre'];  // "Ana"
$edad = $_GET['edad'];  // 25

echo "Hola, $nombre. Tienes $edad años.";
```

- **$_POST:** Contiene los datos enviados a través de un formulario

```
//Ejemplo de formulario:
<form method="POST" action="procesar.php">
        <input type="text" name="usuario">
        <input type="password" name="contrasena">
        <button type="submit">Enviar</button>
</form>

// procesar.php
$usuario = $_POST['usuario'];
$contrasena = $_POST['contrasena'];

echo "Usuario: $usuario, Contraseña: $contrasena";
```

Consideraciones:

Es importante validar y sanitizar los datos que provienen de superglobales, ya que pueden ser manipulados por los usuarios. Nunca confiar completamente en los datos recibidos del cliente sin una correcta validación, para prevenir problemas de seguridad como la inyección de código.