Tema1: Cliente-Servidor

Cliente-Servidor: Solicitamos a un servidor web una página, servidor lo busca la recupera y la envía a nootros.

Páginas estáticas: no varía el contenido después de crearla, programador debe cambiarla todas las veces de variaciones necesarias. No es necesario saber programar, Solo un servidor web. Actualizaciones de forma manual

Páginas dinámicas: está en diversas fuentes. 2 tipos: código que ejecuta el navegador (JS dentro de HTML); las no HTML (PHP…). Conexión: solicitamos al servidor, servidor busca, si es dinámica la página ejecuta el código y lo procesa, puede que tenga que consultar BBDD, posteriormente lo envía en formato HTML.

Aplicaciones web emplean páginas web dinámicas para crear aplicaciones que se ejecuten en un servidor web. No es necesario instalarlas en aquellos equipos en que se vayan a utilizar, sencillo de gestionar, puede usarlo en aquellos sistemas que dispongan de un navegador web, desde cualquier lugar con conexión.

AJAX: posibilita realizar programas en los que el JS se ejecuta en cliente también se puede comunicar con el servidor.

Java EE: orientado a desarrollo de aplicaciones <%java en html%>(si pregunta eso se ha fumado un porro del tamaño de Torre eiffel)

AMP: apache, MySQL y PHP/Perl/Python ASP.Net: desarrollo de aplicaciones

Lenguajes de guiones: (PHP, Python) Lenguajes compilados a código nativo: C . a código intermedio: Java EE

Tema 2: PHP

PHP: Hipertext PreProcessor, alto nivel, embebido en HTML y ejecutado en el lado servidor

Se escribe con <?php…?> o <?...?>

Es case-sensitive, nombre de la variable debe empezar por _ o una letra seguido de movidas.

Es débilmente tipado, no es necesario definir el tipo de variables porque se declaran cuando se asigna un valor. También pueden declararse por referencia &, variable puede contener datos de diferentes tipos a lo largo de script.

Ámbito de variables:

- Local: dentro de una función, significa solo dentro de esa función puede usarse.
- Global: que está declarada fuera de toda la función. Podemos usar global $a; dentro de una función para hacer la llamada a la global.
- Superglobales: son predefinidas accesibles desde cualquier ámbito. $_GLOBALS, $_SERVER, $_POST, $_GET, $_REQUEST, $_COOKIE, $_FILES, $_ENV, $_SESSION.
    - Todas las variables globales se almacenan en $_GLOBALS[‘nombre’]

Constantes: no empiezan por $, pueden definirse mediante define(¨Begoña¨, 20) o const, son accesibles desde cualquier sitio, no pueden ser eliminadas ni redifinidas.

Podemos visualizar con echo “”; print “”; print_r($aray); primero claves luego valor, var_dump(‘$array”); tipo luego valor. $a++ devuelve $a luego lo incrementa, ++$a incrementa $a y devuelve el nuevo valor.

Ejecución pasos: los archivos .php, servidor web apache iniciado, los archivos en el directorio raíz del “c\xampp\htdocs”, en la barra de direcciones del navegador mediante protocolo http.

Require y requiere_once: insertar en documento código del archivo externo, sino encuentra fatal error.

Include e include_once: lo mismo de arriba pero antes de ser ejecutado por el servidor. Sino encuentra warning pero sique ejecutándose.

Funciones pueden pasarse por valor referencia y defecto.

Tema 3: Fundamentos PHP

Conversión de tipos de datos en operaciones:

- Lógicas: Null, 0, '0' es false. Cualquier otro es true.
- Aritméticas: cadenas se intentan leer como números, sino se convierten en 0, true en 1, false en 0.
  - Enteros: si esta el resultado en el rango de los enteros devuelve un entero. Si sale devuelve float.
  - Enteros con float: el resultado es float
- Comparación: si operando es numero el otro se convierte también, si ambos son cadenas se comparan como cadenas.
  - Numeros: el 0 toma false, otros como true
  - Cadena: vacia toma false, el resto true
- Cadenas de caracteres: Null y false se convierten en ¨ y true en 1
  - Con dígitos: los extrae y opera con ello como entero. Sino toma el valor 0
- Booleanos: true lo toma como 1 y false como 0

Casting: establece el tipo para un dato sin tener en cuenta su contenido $var = (bool)/(int)/(float) $num;

HereDoc: $cad = <<<FOO que tal begoña, todo bien? FOO;

gettype: devuelve un string con el tipo del dato recibido como argumento.

settype($que, "integer"): cambia el tipo de dato.

printf("%[relleno][alineacion][longitud][precision][tipo]"):

printf("%010d" , 32) ; // Ancho de 10 y rellena a izq con 0 -------- 0000000032

printf("%'*10d", 32) ; // rellena a izq con * ---------- ********32

printf("%'*-10d", 32) ; // rellena a derecha con * ---------- 32********

printf("%'*10.5d", 32.25) ; // con enteros no se aplica precisión ********32

printf("%'*10.5f", 32.25) ; // con tipos float si se aplica ---------- **32.25000

printf("%'*10b", 17) ; // dato en binario ---------- *****10001

printf("%0-10s", 32) ; // dato tipo cadena y relleno a dcha con 0 3200000000

d: entero e: notacion científica

Tema 4

if($movidas):

echo ""

elseif ($otras movidas): //nunca aquí separar el elseif

endif:

Elseif y else if son iguales solamente cuando utilizamos llaves

While(condición){} Ó While(condicion): y luego endwhile;

For(){} o for: y luego endfor;

Break finaliza la ejecución de for, foreach, while, do-while, stirch en curso

Acepta un numerillo que indica cuantos debemos de salir veces de un bloque anterior explicado.

Continue: se comporta igual que break en un switch cuando no se proporcionan argumentos

El nombre de una función puede ser una cadena de caracteres almacenada en una variable.

$nombreFuncion = "miFuncion";

function miFuncion($mensaje) { echo "La función miFuncion se ha llamado con el mensaje: $mensaje"; }

$mensaje = "Hola desde la función"; $nombreFuncion($mensaje);

Func_num_args(): devuelve el numero total de argumentos de la función.

Func_get_args(posicion) da el dato recibido por la función en la posición indicada.

Func_get_args() da un array de todos los argumentos recibidos por la función.

Formularios y recogida de datos

Get> path/movida.php?param=begoña&param2=auretkoexeva

$_GET[‘nombre’] para recoger datos

$_POST[‘contraseña’] para recoger datos, no se mostraran en el url, utilizamos cuando queremos ocultar la caída de Begoña.

Necesitamos validar ha que la info que llegue bien ya que podemos recibir la información incorrecta y/o estar siendo atacados con código malicioso.

Isset() comprobar si existe aunque este vacía

Empty() si no existe y/o esta vacía.

Stripslashes($str) filtrar las / que nos envían a la pagina de destino.

Htmlspecialchars() convertir caracteres especiales a entidades html

Una vez validado la existencia y limpiado de ataques, debemos validar las características que tiene que tener la variable.