**Lenguaje PHP**

**El código no se visualiza del lado del servidor porque php lo renderiza**

**Se puede meter todo el código html que se requiera en la pagina la etiqueta para declarar que vamos a usar php es**

**<?php y su cierre ?>**

**Declaramos variables**

**$variable(cualquier nombre no se admiten números de inicio)**

**Php es un lenguaje que se lee de arriba hacia abajo asi que una variable puede r cabiando su valor o funcionalidad uso**

**Tipode de datos admitidos por php**

**$variables = “contendio”;**

**$numero = 30; (números)**

**$nobrevariable = true; (trues o false admite boleanos)**

**Se concatena con . y un espacio seguido de las variables que queremos**

**Función eco para imprimir en pantalla**

**Tenemos las funciones matemáticas suma resta multiplicación y división**

**Comentarios en php // por línea y /\* \*/ para un comentario mas largos igual que en javascritp5**

**Existen también el sumar unoy restar uno con el ++ o –**

**Introducción libro resumen**

**Creado en 1995 por Rasmur Lerdorf ( personal home page tools)**

**Software libre**

**Dos tipos de paginas**

**1 páginas estáticas: una vez que sale a imprenta no es posible realizar cambios en su contenido**

**2 páginas dinámicas: puede mostrar información diferente que cada vez que**

**Es invocada.**

**Lenguaje interpretado por ello no necesita ser compilado**

1. **Aspectos básicos php** 
   1. **El escript se ejecuta en el servidor siempre debe empezar etiqueta de inicion <?php> etiqueta de cierre <?>**
   2. **Siempre que se acaba una sentencia o línea de código se pondrá “;”**
   3. **‘echo’ para mostrar texto por pantalla**
   4. **Las variables de declara con $ al inicio**
   5. **Texto plano: se puede escribir texto que no vaya a cambiar (plano) mezclando con etiqueta HTML**
   6. **Texto con etquetas HTML enriquece el código y será mas legible**
   7. **Echo para imprimir texto se usara de forma general ya que es un constructor del lenguaje**
   8. **Print para imprimir texto es una función asi que se ejecuta y devuelve resultado**

**2 Funciones**

**Sirven para hacer subrutinas predefinidas. Pueden recibir parámetros que se pueden trabajar como variables asi cada llamado puede dar un valor diferente.**

**Las funciones pueden devolver valor o no y se puede usar dicho valor en otras líneas de código**

**¿Llamado de una función <?php> llamandoFuncion($parametros); <?>**

**Para crea una función se debe utilizar una palabra llamada “funtion” seguido del nombre de la función y paréntesis con o sin argumentos parámetros**

**A continuación dentro de {} se pondrá lo que va a realizar la función**

**PARAMETROS POR DEFECTO**

**Se puede declarar parámetros que tengan una valor por defecto esto es optimo con los valores bolean true o false**

**Ejemplo: <?php funtion nombredelafuncion( $aargumento1= 1, $argumentos2= “hola”){**

**Echo ‘$argumento1. $argumento2;**

**}Nombredelafuncion(“otro valor”);**

**?>**

**PARAMETROS POR REFERECIA**

**Si el objetivos es que las variables se vean afectadas por fuera de l función tras modificarlas habrá que enviar los parámetros por referencia**

**Ejemplo: <?php**

**Funtion sumantorio(&$variable){**

**$variable ++;**

**$num = 1;**

**Sumatorio($num);**

**Echo $num; ?>**

**FUNCIONES VARIABLES**

**PHP permite llamar funciones tomando como nombre de la función valor de la variable**

**TIPO EN LOS PARAMETROS**

**Los parametros podemos obligarlos ejemplo con un bolenao dándole valor al paramtro y puediendolo cambiar al llamar la función**

**RETORNO DE VALORES**

**Las funciones como funcionalidad nos devuelven un resultado que lo captaremos con la palabra return sino tiene resultado nos devovlera null**

**VARIABLES**

**Contenedores para guardar números strings boleanos etc tienen caso sensitivo se tiene siempre que escribir igual**

**Las variables se asignan por valor esto quire decir que si se usan dos vaiables y se asigan a en b si se modifica a b nosufre cambio alguno**

**Se puede apuntar por referencia y para ello se debe usar el &**

**Se pueden unir una o varia variables con el “.”**

**Las variables se asignan por el propio lenguaje en el momento de ejecución se uede obligar a una variable que tenga un tipo concreto usando settype()**

**TIPOS DE DATOS**

**1 String**

**2 integer**

**3 float**

**4 bool**

**5 array**

**6 object**

**7 Null**

**8 resource**

* **String: cadena de caracteres cada cracter es guardado en bytes tiene como longitud máxima 256 caracteres y un peso de 2gigas** 
  + **Entre comillas simples ‘’ forma eficiente de escribir texto**
  + **Entre comillas dobles “” el texto se revisara ya que tenemos mas funciones como saltos de línea, retonrno de carro tabulación además de también la posibilidad de incrustar varibles que podrán ser leidas**
  + **Heredoc para escribir texto no aceptara varibles**
  + **Nowdoc para escribir texto e identificarlo aceptara vaiables con sus valores**
  + **Integer: números enteros o naturales**
  + **Float: numero decimal**
  + **Bool : expresiones boleanas verdarero o falso**
  + **Array: los arreglos son mapas de valores ordenados que pueden ser listas , pilas, colas etc**
  + **Object: para usar un objeto los primero es crearlo con el neww object**
  + **Null las variables pueden se de tipo null si al construirse no se les asigna un valor**
  + **Resourse: permite saber si existen recursos y si se pueen usar**

**ALCANSE DE LAS VARIABLES**

**Si se declara una variable global no se podrá usar dentro de la función nos dira que dich variable no esta definida en la función**

**Las variables se pueden pasar por referencia se pueden usar variables globales dentro de una función con [“$nombredelavariable”]**

**VARIABLES ESTATICAS**

**Se pueden declarar variables estaticas para un uso propio con la palabra static antes del nombre de la variable**

**MANEJO DE VARIABLES**

**Para comprobar que tipo de datos es el que estamos trabajado con la variable tenemos**

**varias funciones predefinidas como**

* **Empty devuelve cero si el valor introducido es o o o.o $bol =empty($variable)**
* **Boolval: devuelve un boleano $bool = bollval($variable)**
* **Intval: deuvle un int**
* **Floatval : devuelve los decimales**
* **In\_null: comprueba si es nulo o no**
* **Is\_bool: comprueba si es buelano**
* **Is\_array: comprueba si es de tipo array**

**ARRAYS**

**Para trabajar con variables mas complejas usamos los arrays seria contenedores con asociados calve valor**

**Las claves pueden ser números enteros o cadenas de caracteres si se pone un decimal lo convierte a entero los valores boleanos se covierten en 1 para verdadero y 0 para false**

**Los valores pueden ser números o cadenas de caracteres**

**ARRAY INDEXADOS**

**Sirven para guargar varios valores de forma simple podemos acceder a ello mediante la clave también debemos tener en cuenta que los array empiezan en la posición 0. Se pueden crear arrays vacios para posteriormente manipularlos pudiendo agregar nuevos valores si se agrega un valor sin el índice este se posicionara después del ultimo dato en cambio si se agrega con un índice que aya existe este se sobreescribira**

**Se puede acceder a alas variables como arrays usando los corchetes podríamos modificar una letra pasando dicha letra**

**$nombre = “Carlos”;**

**$nombre[4] = “a”; 🡪queraria como carlas**

**ARRAY ASOCIATIVOS**

**Sirve para referenciar datos concretos como el saliro de los empleados y mediante su clave podemos hacer cambios**

**OPERACIONES CON ARRAYS**

**Para comparar arrays debemos comprobar su clave, valor or den y tipos**

* **$a +$b = unir a con b**
* **$a === b = comprueba si a y b contienen las mismas claves y valores**
* **A != b = comprueba que a es distito que b**
* **A <>b = comprueba que a y b no tienen las mismas claves ni valores**
* **A ===b = comprueba que a y b tienen las mismas claves los mismos tipos y orden y valores**
* **A ¡= b comprueba que a y b no tienen las mismas claves y valores distintos orden o tipo**

**MANEJO DE ARRAYS**

**Funciones para el uso de arrays**

* **Count sirve para saber cuantos elementos contiene el array formula $c = count($array)**
* **In\_array permite saber si existe el valor introducido formula $encontrado = in\_array (“valor”, $array) nos devolvería true o false**
* **Key\_exist devuelve verdadero si ecuentra la clave o índice enviado formula $encontrado = key\_exist(“clave”, $array)**
* **Array\_ values devuelve una array con todos los valores del array introducidos formula $valores = array\_values ($array)**
* **Array\_keys devuelve un array con todas las claves del Aray formula $claves = array\_keys($array)**
* **Array\_combine devuelv un array usando un array para las clves y otro para los valores**
* **Aa\_merge combina dos o mas arrays y unifica los array introducidos y los devuelve como un array los valores se anidad uno detrás de otr**
* **Array \_unique devuelve un array son valores duplicados el primer parametr de la fucion será el array a eliminar el segundo parámetro es opcional no indic que compracion es tenemos varias que son** 
  + **SORT\_REGULAR compara ítems sin modificar los tipos**
  + **SORT\_ NUMERIC compara numéricamente**
  + **SORT\_STRING compra con cadenas de caracteres**
  + **SORT\_LOCALE\_STRING compra con cadena de caracteres usando ña codiguracion regional actual**
* **Sort devuelve un array ordenado se le pueden pasar varios flagts seria** 
  + **SORT\_NATURAL para ordenar de forma natural**
  + **SORT\_FLG\_CASE para ordear caracteres de forma sensible a las mayúsculas y minúsculas**
* **Suffle baraja un array devuelve verddero si lo pudo hacer**
* **Rango crea un array desde el valor inicial hasta el final se le puede dar un incremento secuancial**
* **Arry\_shift elimina el primer elemento del array y lo devuelve como valor si el array esta vacio devuelve null**
* **Array pop elimina el ultimo elemnto del array y lo deuvle como valor**
* **Array\_replace dado un primer array ira cambiando todas las claves valor**
* **Array\_slice puede recortar un array sin modificarlo y devuelve el array modificado**

**CONSTANTES**

**Valores únicos que no se van a poder modificar formula**

**Define(nombre de la constante, valor, sensitiva) en la nuevas versione se usa el CONST para crear las constantes en vez del define**

**Recomendable escribir las constantes en mayúsculas y no dejar acivo el sensitivo**

**Se puede acceder a ella de manera global**

**Define() saber si una constante existe**

**Se pueden crear constante de tipos como: int, float, bool o string**

**CONSTANTES PREDEFINIDAS**

* **PHP\_VERSION nos da la versión php**
* **E\_ERROR Constante de error**
* **NULL VARIABLE NULA**

**ETC PGINA 42 PHP BASICO**

**CONSTANTES MAGICAS**

**Existen constantes cuyo valo se modifica en función de donde se est ejecutando ejemplo**

* **\_LINE\_ El numero de la línea actual del fichero**
* **\_FILE\_ ruta completa y nombre de archivo**
* **\_DIR\_directorio del fichero**
* **\_FUNTION\_ nombre de la función**
* **\_CLASS\_C NOMBRE DE LA CLASE**
* **PG 43 PHP BASICO**

**OPERADORES**

**Estos pueden recibir uno o mas valores para trabajar con ellos y nos devuelven un resultado estos pueden ser almacenados en variables o escitos en el código**

**Operadores de asigancion los mas usados asignan un valor a una variable**

**Operadores aritméticos suma resta multiplicación división exponencial etc**

**Pagina 45 php básico todos los operadores**

**Asignacion con operadores trabajar con un variable a que se pueda calcular aritméticamente y el valor se guardar en a a diferencia de otros lenguajes que usaría ua tercera variable para el resultado php pemite guardarlo en la primera variable**

**COMPRACION**

**Para comparar nos devuelve true en caso de que se cumpla la compracion pg 45**

**COMPRACION DE CARACTERES**

**Debemos tener en cuenta que comparar caracteres es mas complicado que números o buleanos para ello se tendrá en cuenta que cuando el resultado de comparar sea que a es menor que b nos dará como resultado un valo negativo y positivo si a es mayor que b un cero en caso de igualdad**

**Con las siguientes funciones predterminadas podemos comprar caracteres mas fácilmente ejemplo**

* **Strcmp compara dos cadenas $res= strcmp(“hola”,”hola”); resultado 0**
* **Strcasecmp compara dos cadnas sin discriminar entre mayúsculas y minúsculas $res = strcasecmp(“variable”,”variable”);0**
* **Strncmp aparte de las dos cadenas se dene introducir un entero para comparar los primero x caracteres $res = strncmp(“variable”,”variable”,4);resultado “vari”**
* **Strcasecmp nos discrimina entre mayúsculas y minuslculas caracteres $res = strncmp(“variable”,”variable”,4);resultado “vari”**

**LOGICOS**

* **$a en $b ay b son verdaderos**
* **$a && $b ay b son verdaderos**
* **$a or $b a o b son verdaderos**
* **$a || $b a y b son verdaderos**
* **¡$a a no es a**
* **$a xor $b a o b son verdaderos pero no amabas**

**INCREMENTOS**

* **++$A pre-incremento**
* **%a++ post-incremento**
* **-$a pre-decremento**
* **$a- pos-decremento**

**CONCATENCAION DE CADENAS**

**Para unir cadenas se usa el “.” Se puede unir tanto variables como texto plano tanta veces como queramos**

**Sei sse quiee concatenar y dar vaor se hará <.=>**

**Si se pone @ delante de una línea de código esta no dará error en caso de lo hubiera**

**ESTRUCTURAS DE CONTROL**

**Para controlar que hacer con el retun de la las funciones para ellos usaremos**

**1 if else -> controlara en su caso dos opciones verdadero o falso. Compararemos la expresión entre paréntesis y las intruciones entre corchetes**

**Else se pone después de comprobar la expresión en caso de que no se cumpla usaremos esta sentencia ua forma de poner expresiones en una misma línea de codig es**

**Se pone un ? después seguido el caso verdadero dos puntos y el caso negativo**

**<?php**

**$edad = 66;**

**Echo $edad >= 65 ? “viajeparamayores”: “viajeparajovenes”;**

**?>**

**SWITCH**

**Podemos pedir como parámetro una variable y todos los casos que se quieran controlar cada caso debe terminar con la sentencia break**

**Se pude dar muchos caso seguidos sin el break este seria útil si queremos que todas las sentencias tengan el mismo valor**

**Existe un calusula default en caso de que no se cumpla ninguna condición**

**WHILE**

**Bucle para controlar el código funciona como un if pero con la diferencia que se repite por lo tanto abra que modificar el valor dentro del bucle y tener cuidado de no crear bucles infinitos**

**DO WHILE**

**Se usa siempre y cuando se vaya hacer una iteracion por el bucle obligatoriamente sintaxis**

**Primero la palaba do seguido de un bloque con llaves con el código se cierran las llaves pones while y ldentro de arentesis la condición para seguir**

**FOR**

**Bucle pensado para que se repita un numero fijo de ocaciones a diferencia del while se puede usar hasta que la exprexion lógica sea falsa**

**Admite tres parámetros separados por punto y coma**

**El primer para metro es para inicializar el bucle acontiuacion viene la expresión lógica a comprobar y por ultimo tenemos el parámetro de salida del bucle**

**FOR EACH**

**Se utiliza para iterar un array de un forma rápida y sencilla los parámetros a usar son los siguiente primero nombre del array luego necesita un variable que va a coger las posiciones de cada elemento del arrive si se quiere trabajrcon un array de tipo clave y valor necesitamos igualar valor con clave y para ello lo hacemos dándole ese valor a la variable ejemple**

**alumnos = [ana = 20, juan = 30 , ]**

**Foreach (alumnos as alum=nota){**

**Echo alum. ‘ : ’.nota ‘<br/>**

**Código para ejecutar try cat con clases lazar varios cats personalizados**

**¡!!!class CeroExcepcion extends Exception{}**

**class NumerLargoExcepcion extends Exception{}**

**try{**

**throw new CeroExcepcion("Error personalizado");**

**}**

**catch (CeroExcepcion $e){**

**//Tratar la excepcion**

**}**

**catch (NumerLargoExcepcion $e){**

**//Tratar la excepcion**

**}**

**catch (Exception $e){**

**//Trata la excepcion**

**}**

**finally{**

**//Tratar finally**

**}!!!!!**

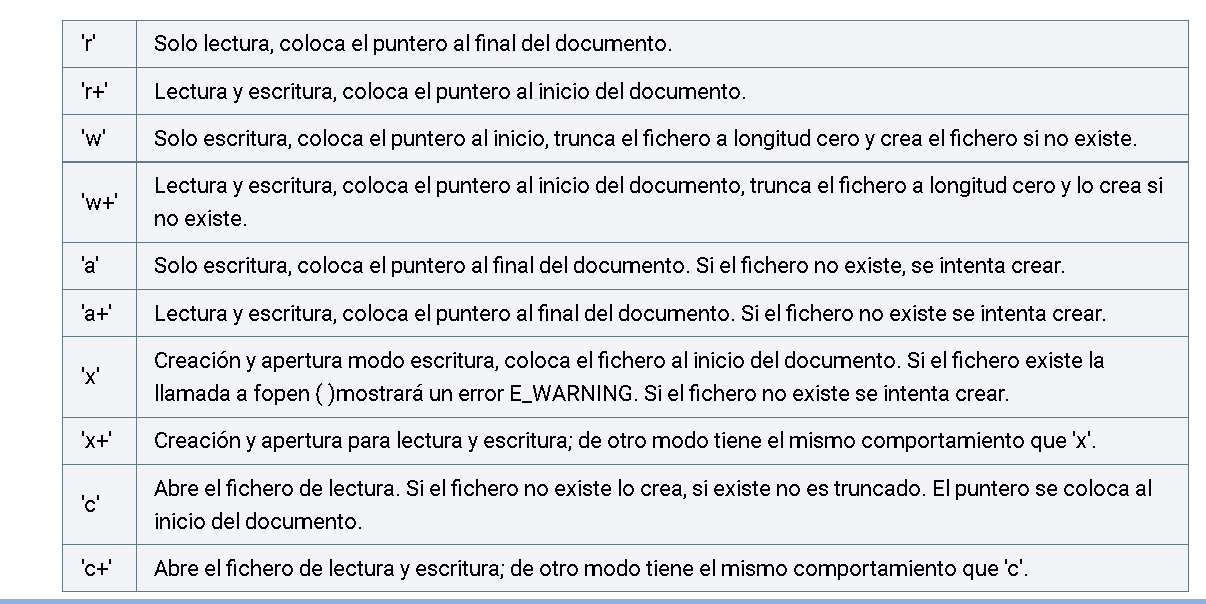
**LIBRERIAS EN PHP**

Se dispone de dos formas para incluir la librería en el *script* : en la primera, como ya se ha visto, se usará la sentencia *include* ( ); en la segunda, la sentencia *require* ( ).  
La única diferencia es que si la sentencia *include* ( ) no encuentra el fichero en la ruta especificada mostrará un aviso y continuará la ejecución. En cambio, *require* ( ) avisará de que ha habido un error crítico y se parará la ejecución.

**FICHEROS EN PHP**

Lo primero que hay que tener en cuenta es que siempre se parte de la carpeta raíz donde está instalado el servidor, es decir, que si se quiere tener varias carpetas almacenadas, hay que empezar escribiendo el “./” seguido del nombre de la carpeta o directorio de carpetas y finalizar por el nombre del fichero con la extensión.

La función que se va a utilizar para abrir un fichero es ***fopen*** ( ), a la que se enviará, como primer parámetro, la ruta donde se aloja el fichero y, como segundo parámetro, el modo en que se quiere abrir, En la siguiente tabla se muestran los modos en que se puede abrir el fichero

****

**Ejemplo escritura**

 $texto = "Ejemplo";

fwrite($archivo, $texto);

La función *fwrite* ( ) acepta un tercer parámetro opcional que limita la cadena de caracteres a un índice máximo introducido.

 fwrite($archivo, "Ejemplo", 3)

**LEER UN FICHERO**

*file\_exists* ( ) se puede comprobar que exista el fichero, al que se le puede enviar como parámetro una cadena de texto con la ruta completa y el nombre del fichero con la extensión

A la hora de leer un fichero se usará un **puntero** interno que recorrerá el fichero hasta que se acabe. Para ello, se creará un bucle y se irá leyendo el fichero hasta encontrar el carácter ‘EOF’ (End Of File), en español «fin de fichero»

*ftell* ( ), a la que se enviará el fichero y devolverá un entero con el índice en el que se encuentra. Cuando se usen funciones de lectura, el puntero se irá incrementando automáticamente.

e usará la función *feof* ( ), que sirve para comprobar que el puntero que recorre el fichero no ha terminado. En el caso de que *feof* ( ) se encuentre con EOF, devolverá un *boolean* verdadero.

***fgetc*** ( ) se puede leer un carácter. Se le envía el archivo y la función devuelve el carácter y aumenta en uno el puntero para que avance la lectura del fichero.

Como *fgets* ( ) captura la línea entera hasta encontrar un fin de línea

Se dispone de una función para capturar todo el fichero, *fread* ( ), que, al igual que *fgets* ( ), permite añadir un segundo parámetro para añadir una longitud máxima.

*rewind* ( ), que permitirá rebobinar y empezar de nuevo.

En el caso de que se quiera mover a un índice en concreto, ya sea avanzar o retroceder, se utilizará la función *fseek* ( ), a la que se le indicará el archivo y el índice al que se quiere posicionar.

**Funciones parar tratar con ficheros**

|  |  |  |
| --- | --- | --- |
| **Valor** | **Rol** | **Modo** |
| 400 | Autor | Lectura |
| 200 | Autor | Escritura |
| 100 | Autor | Ejecución |
| 40 | Grupo | Lectura |
| 20 | Grupo | Escritura |
| 10 | Grupo | Ejecución |
| 4 | Global | Lectura |
| 2 | Global | Escritura |
| 1 | Global | Ejecución |

chmod,()

Se pueden ir sumando los permisos, por ejemplo, si solo se quiere dar permiso al autor, habría que usar el valor 700; si es solo lectura para todos los perfiles, el 444. Este valor debemos introducirlo en octal, es decir, añadiendo un cero delante del número. Con la función *fileperms* ( ) enviándole el fichero mostrará los permisos del fichero.  
Ya se ha explicado cómo crear un fichero en el caso de que la ruta no exista. Con la función *copy* ( ) se puede copiar el fichero de un origen a un destino. Si se quiere renombrar el fichero o el directorio, se puede usar *rename* ( ); si se quiere borrar el fichero, se usará *unlink* ( ) y para borrar un directorio *rmdir* ( ). En caso de que no se pida un directorio de destino, el código dará un error.