Tabla de contenido

18/11/2019 (Presencial 2): Unidad 2: Características del lenguaje PHP.

No había dudas de la unidad, por lo dimos un repaso a la teoría viendo algunos ejemplos.

Os envío más ejemplos de los que pudimos ver, ya que no me funcionaba bien la ejecución de los proyectos, pero ya está solucionado. Echadles un vistazo, intentad entender lo que hacen y probar a hacer algún cambio

1.2 CADENAS DE TEXTO

Ejercicio 12_strings.php

Formas de especificar las cadenas de texto:

- Usando como delimitador las **comillas simples**: el texto que va encerrado entre las dos comillas simples puede contener cualquier carácter, pero si queremos que el carácter comilla simple aparezca en el texto, este deberá ir precedido del carácter de escape (\). En este caso si escribimos el nombre de una variable dentro del texto, no se cambiará por su contenido, sino que contendrá su nombre. Si en la cadena **incluimos la secuencia \n, no la interpreta, sino que escribirá esos caracteres.**
- Usando como delimitador las **comillas dobles**: más avanzada que la anterior, en este caso si en la cadena aparece **la secuencia \n, el intérprete de PHP entiende que ha de insertar un salto de línea en esa posición de la cadena dentro del código (no en la pantalla, se ve en el código fuente de la página)**. Si se hace referencia a una variable dentro de la cadena, se sustituye por su valor.
- Usando la sintaxis **heredoc**: tiene las mismas características que el anterior método (comillas dobles). La diferencia está en que esta forma permite que la cadena de caracteres ocupe tantas líneas como sea necesario **(una string por defecto admite 256 caracteres, puede llegar a alcanzar 2GB).** La cadena guardará la apariencia con la que fue escrita. Para delimitar un texto con este método se hace de la siguiente forma:

```
<<<Identificador
Cadena de caracteres
Sigue la cadena de caracteres…
Identificador;
```

  - El identificador será un nombre con el que vayamos a identificar esa cadena, precedida de los caracteres <<<, y un salto de línea. El identificador debe aparecer tanto al principio como al final de la cadena. Cuando aparece al final de la cadena, debe aparecer al principio de esa línea y terminado por un ";" y justo después un salto de línea, si no da error. También da error si la última línea tiene espacios a la izquierda. **Debe estar pegada al margen izquierdo. Además los espacios que haya por la izquierda en la cadena, los coge como espacios en blanco y ocupa sitio.**

Las cadenas pueden tratarse como un array, accediendo a cada carácter mediante la posición que ocupa en la cadena.

Existe también la sintaxis **nowdoc** que tiene las mismas características que el método de comillas simples.

1.1 FUNCIONES RELACIONADAS CON LOS TIPOS DE DATOS

- Las funciones isset() y empty() se utilizan mucho en PHP para validación de datos que nos llegan de un formulario.

- Para definir constantes se usa la función define.

```
bool define ( string $identificador , mixed $valor [, bool
$case_insensitive = false ] );
```

- Los identificadores no van precedidos por el signo "$" y suelen escribirse en mayúsculas, aunque existe un tercer parámetro opcional, que si vale true hace que se reconozca el identificador independientemente de si está escrito en mayúsculas o en minúsculas.

1.3.1 Trabajando con fechas en php

Desde el punto de vista de muchos sistema informáticos, el origen del tiempo es una fecha concreta, el 1 de enero de 1970 a las 00:00:00 GMT. Esta fecha es la considerada como era UNIX.

Cuando se pide la hora a un ordenador, lo que se obtiene es un número **(denominado marca de tiempo)** que representa el n.º de segundos que han transcurrido desde el inicio de la era UNIX.

Las funciones de fecha y hora permiten obtener la fecha y la hora del servidor en el que se esté ejecutando el script, por tanto dependerá de la configuración horaria de este.

PHP dispone de la función **time()** que devuelve un n.º entero que representa la marca de tiempo correspondiente al instante en que se ejecutó dicha función. Ya que la marca de tiempo es un número entero se puede operar con él igual que con el resto de los números enteros.

Dado un valor en segundos, con unas sencillas operaciones podríamos llegar a obtener la fecha que representa. Pero PHP cuenta con funciones que hacen este trabajo por nosotros y que nos permite trabajar con fechas en formato años, meses, días, horas, minutos y segundos:

- la función **getDate()** obtiene un array asociativo con la información de la fecha y hora del servidor.

- Podemos pasar como parámetro a la función getDate un valor entero que representa una determinada fecha. Por defecto utiliza la hora actual del servidor.

Ejemplo 16_fechas_01.php

En este ejemplo:

- *Se usa la función time para obtener la marca de tiempo de la fecha actual.*
- *Se muestra el contenido del array asociativo que devuelve getDate con la fecha actual*
- *Se muestra el contenido del array asociativo que devuelve getDate con una marca de tiempo como parámetro*

- Para evitar usar un array asociativo tenemos la función **date()** que devuelve una cadena de carácter con una fecha a la que se le puede aplicar un determinado formato. Para la definición del formato de fecha tenemos las siguientes opciones:

  https://www.php.net/manual/es/function.date.php

  - a: hace que en la hora aparezca “a.m.” o “p.m.”.
  - A: hace que en la hora aparezca la cadena “A.M.” o “P.M.”.
  - d: día del mes con dos dígitos. Los días del 1 al 9 los muestra con un 0 a la izquierda.
  - D: día de la semana como una cadena de tres letras: por ejemplo “Mon”. La cadena corresponde con el nombre en inglés.
  - F: nombre del mes completo en inglés como una cadena de caracteres.
  - h: hace que la hora aparezca en formato de 01 a 12
  - H: hace que la hora aparezca en formato de 00 a 23
  - g: hace que la hora aparezca en formato 1 a 12
  - G: hace que la hora aparezca en formato 0 a 23
  - i: hace que los minutos aparezcan en formato 00 a 59
  - j: hace que el día aparezca en formato 1 a 31
  - l: (L minúscula) día de la semana completo en inglés
  - L: escribe un 0 o un 1 en función de si el año no es bisiesto (0) o sí (1).
  - m: hace que el mes aparezca en formato 01 a 12
  - n: hace que el mes aparezca en formato 1 a 12
  - M: mes como una cadena de 3 letras en inglés.
  - s: Hace que los segundos aparezcan en formato 00 a 59
  - S: cadena de dos caracteres con el sufijo ordinal en inglés, por ejemplo “th”, “nd”, “st”.

- t: número de días del mes especificado, de 28 a 31.
- U: número de segundos desde el comienzo de la era UNIX
- w: número del día de la semana teniendo en cuanta que el 0 es el domingo y el 6 el sábado.
- Y: año con cuatro cifras: 2019
- y: año con dos cifras: 19
- z: día del año, de 0 a 365. El primer día del año es el 0.
- T: abreviatura de la zona horaria, por ejemplo EST, MDT

Ejemplo 16_fechas_02.php

En este ejemplo:

- *Se muestra el contenido del array asociativo obtenido con la función getDate().*
- *Varios ejemplos de salida formateada de la fecha que se pasa por parámetro a la función date()*
- *Muestra la salida de la función time*
- *Muestra la salida de la función time sumándole 7 días en segundos (7*24*60*60)*
- *Uso de la función mktime, devuelve fecha y hora en formato UNIX*
- *Función mktime pasando valores de fecha y hora mostrando formateado con date.*
- *Mostrar misma fecha que el anterior pero pasando marca de tiempo a date.*
- *Pasando 0 como valor del parámetro día coge el último día del mes anterior.*
- *Pasando 0 como valor del parámetro mes, coge el último mes del año anterior*
- *Establece la zona horaria en Nueva York con **date_default_timezone_set** y muestra la fecha y hora . Lo hace con Auckland (Nueva Zelanda) y después vuelve a establecer la zona horaria en Madrid,*

Muestra el año para añadírselo al copyright

- Las función **strftime()** representa otra posibilidad para dar formato a una fecha. Esta función usa las convenciones locales de la máquina desde la que se ejecuta el script para devolver una cadena con el formato definido. Los nombres del mes y del día de la semana, así como el resto de las cadenas dependientes del idioma siguen los valores establecidos por la función setlocale(). El formato queda definido por los ??? PAG212
- La función **mktime()** devuelve una determinada fecha también en formato UNIX. Como parámetros se le pasan: hora, minuto, segundo, mes (entre 1 y 12), día y año. Si introducimos un 0 como valor del parámetro día tomará el último día del mes anterior, y si introducimos 0 como valor del parámetro mes, tomará el mes anterior.

- La función **checkDate()** permite comprobar si una determinada combinación de día, mes y año representa una fecha válida. Recibe 3 parámetros numéricos, el mes va acotado entre 1 y 12, el día depende del mes indicado teniendo en cuenta los años bisiestos, y el año está acotado entre 0 y 32767. El orden de los parámetros es mes, día y año. Devuelve true si la fecha es válida.
- Devuelve true si la fecha es válida.

16_fechas_03_checkdate.php

En este ejemplo:

Se comprueba mediante la función checkdate si Varias fechas son o no correctas

---

- Para sumar o restar fechas debemos obtener el número de segundos transcurridos desde las 12 de la madrugada del día 1 de enero de 1970 (formato horario UTC) y a continuación realizar la operación deseada.
- A la hora de realizar cálculos hemos de tener en cuenta que:
  - 1 minuto = 60 segundos.
  - 1 hora = 60*60 = 3.600 segundos
  - 1 día = 24*3600 = 86.400 segundos

16_fechas_04_operacionesFechas.php

En este ejemplo usamos la función mktime para obtener marca de tiempo y trabajar con ella sumando y restando fechas)

- La función **strtotime():** convierte una cadena de texto en formato fecha a formato UNIX.
- Las funciones mktime y strtotime son similares, la segunda permite más flexibilidad en la entrada de parámetros, puedes indicarle, por ejemplo, “last Saturday” y permite aritmética de fechas, por ejemplo $fecha = strtotime(+1day); devuelve la marca de tiempo de un día más de la fecha.

16_fechas_05_strtotime.php

En este ejemplo:

- *se muestra la marca de tiempo obtenida con la función strtotime().*
- *Se muestra en formato entendible con la función date*
- *Se suma un día usando la expresión + 1 day y después 1 semana (+ 1 week)*
- *Se suma una semana, dos días 4 horas y 2 segundos usando la expresión + 1 week2 days 4 hours 2 seconds*
- *Se muestra el próximo jueves usando la expresión next thursday*
- *Se muestra el pasado lunes usando la expresión last monday*
- *Se muestra la marca de tiempo de la fecha 1996-09-21*
- *Se muestra la marca de tiempo de la fecha 1996-09-21 a las 13:25:05*
- *Se muestra mediante un bucle las fechas de los próximos 6 sábados strtotime(saturday) y se va sumando + 1 week*

- La función **setLocale()** permite establecer el idioma en el que aparecerán los valores de las fechas, el carácter separador decimal, y en general, toda aquella información que dependa de la configuración regional de la máquina desde la que se ejecuta el script. Libro fecha5.php

---

16_fechas_05.php
https://www.php.net/manual/es/function.strftime.php

En este ejemplo:

- *Obtenemos la marca de tiempo de una fecha determinada.*
- *Se llama a una función "muestra($marca) que recibe una marca de tiempo y muestra los datos de la fecha en español, habiendo cambiado el idioma con setlocale.*
- *Las fechas se muestran usando la función strftime que muestra la fecha formateada según la configuración local.*

---

https://www.php.net/manual/es/timezones.php

1.2 VARIABLES ESPECIALES PHP

Variables supergobales: son variables internas que pueden usarse desde cualquier ámbito del programa.

Cada una de estas variables es un array que contienen un conjunto de valores. No es necesario usar la palabra GLOBAL.

https://www.php.net/manual/es/language.variables.superglobals.php

Algunas de ellas son (hay más, estas son con las que vamos a trabajar este curso):

- **$_SERVER:** este array contiene información como cabeceras, rutas y ubicaciones de script. Las entradas de este array son creadas por el servidor web. 14_superglobal_1_Global
- **$_POST:** array asociativo que contiene las variables que se han pasado al script a través del método POST de HTTP (en los formularios se ha indicado POST como método de envío de los datos).
- **$_GET:** array asociativo que contiene las variables que se han pasado al script vía URL (en los formularios se ha indicado POST como método de envío de los datos)
- **$_COOKIE:** array asociativo que contiene las variables que se han pasado al script a través de COOKIES HTTP **(de las cookies ya hablaremos en otro tema)**
- **$_REQUEST:** Un array asociativo que por defecto contiene el contenido de $_GET, $_POST y $_COOKIE.
- **$_SESSION:** array asociativo con las variables de sesión disponibles para el script actual. **(se verán más adelante)**

- **$_FILES:** contiene los ficheros que se pueden haber subido al servidor.
- **$_ENV:** variables que se pueden haber pasado a PHP desde el entorno en que se ejecuta.

http://php.net/manual/es/reserved.variables.globals.php

Ejemplo: superglobalPost-1.php

Este ejemplo muestra un formulario con un campo de tipo input en el que debemos introducir nuestro nombre. Al pulsar el botón del formulario la página se vuelve a cargar y muestra el nombre que habíamos introducido obteniéndola mediante la variable $_POST. Es decir, se pasan los argumentos a la página a la que se envía el formulario, en este caso la misma, mediante el método POST, y al volver a cargarse se comprueba si la variable $_POST tiene algún valor, en cuyo caso lo muestra.

Ejemplo variable global $_GET: carpeta 14_superglobal_3_Get

Este ejemplo consta dos ficheros. El archivo "superglobalGet-1.php" simplemente contiene un enlace que al pulsarlo lleva a la página "superglobalGet-1-test.php" pasando dos argumentos en la URL, uno se llama "asig" y otro "donde".

La página "superglobalGet-1-test.php" lee los argumentos que le han llegado por el método GET y los muestra en pantalla en un texto que dice "Estudia Desarrollo de Aplicaciones Web en fpdistancia.educa.madrid.org".

2.- ESTRUCTURAS DE CONTROL

2.2 Switch

Ejemplo 09_switch.php

En el ejemplo se obtiene un nº aleatorio entre 1 y 7.

En función del nº obtenido se mostrarán diferentes mensajes. **Observa que en algunos casos no hay instrucción break.**

En el caso de default no es necesario el break puesto que es el último caso escrito.

2.3 Bucles

Ejemplo 10_break_continue.php

En el ejemplo tenemos un bucle while mientras una variable i sea menor de 10

Si i es impar vuelve al inicio del bucle, si no entra en un bucle for con valor inicial de i. Si j es par sale del bucle y continua dentro del while, si i vale 6, 7 u 8 hace un break del bucle while.

3.- FUNCIONES

3.1 Incluir ficheros externos: ejemplo carpeta include1

- Según vayan creciendo nuestros programas, el código cada vez será más complejo. Resulta conveniente agrupar ciertos bloques de funciones o bloques de código en archivos separados y hacer referencia a ellos cuando lo necesitemos para que PHP incorpore su contenido dentro del programa en el que se incluya. Por verificación de un DNI, etc. De esta forma podremos además reutilizarlos en otras aplicaciones.
- Para ello tenemos las siguientes posibilidades:
    - **Require(fichero):** sustituye la llamada por el contenido del fichero especificado en el parámetro de la función. Si no se puede incluir el fichero se detiene la ejecución del script.
    - **Include(fichero):** hace lo mismo que si no se encuentra el fichero da un aviso y continua la ejecución del script.
    - **Require_once() /include_once()** para asegurar que el fichero se incluye solo una vez, de esta forma evitamos errores que se puedan producir si por error incluimos más de una vez un mismo fichero (por ejemplo la definiciones de funciones repetidas).
- **Se suele utilizar require para invocar código que, si no es incluido, el programa puede llegar a errores muy graves y por lo tanto, es mejor parar la ejecución del programa (por ejemplo código que se conecta la una base de datos y que si esa conexión no es posible el programa no debe seguir funcionando**, es mejor lanzar un mensaje y terminar el programa). En cambio include() se suele usar para la llamada a archivos cuyo código no afecta a otras partes de la aplicación y que, por tanto, si no están, no afectará al resto del programa o afectarán de forma leve.
- Se pueden incluir ficheros en cualquier punto del programa. Estos a su vez pueden incluir otros ficheros. Estos ficheros pueden contener variables, constantes, funciones, … o pueden incluir la parte lógica de un programa generar salida, modificar variables… pero no se deben hacer ambas cosas en un mismo fichero. **Se puede usar include o require como resultado de una condición ejemplo if($i<0) include “pag1.php”;**
- **Al incluir un fichero en algún lugar de un documento PHP ¿qué pasa con las variables?**
    - Las variables definidas en el fichero que se ha incluido heredan las reglas de ámbito de las variables que tenía la línea de código donde se insertó el fichero.
    - Si la llamada al fichero que se incluye se hace desde el cuerpo principal del script, las variables definidas en el fichero insertado pasan a ser globales, si la llamada se hace desde una función, las variables definidas en el fichero incluido pasan a ser locales a no ser que se hayan declarado como globales.

Ejemplo: Incluir ficheros externos: ejemplo carpeta 13_funciones_include1

- En este ejemplo tenemos 3 ficheros. El primero contiene una función que escribe código HTML que se muestra por pantalla código correspondiente a una cabecera de una página. Le llega un parámetro que es el título de la página.
- El segundo y tercer archivos son ficheros php que incluyen este archivo y le pasan una cadena de texto que será el título de la página.
- **Fichero ejemInclude-1.php:** es el fichero a incluir, contiene una función llamada cabecera que escribe el código HTML de cabecera de un fichero, poniendo en la etiqueta título un texto que le llega por parámetro. La última etiqueta que escribe la función es la etiqueta de apertura del body
- **Fichero ejemInclude-1-pag1.php:** este script incluye el fichero anterior y llama a la función cabecera incluida en él pasándole un texto como parámetro que será el título de la página. Después muestra un texto incluido en un párrafo y cierra las etiquetas finales HTML.
- **Fichero ejemInclude-1-pag2.php:** igual que el anterior, pero con otro título
- **Fichero ejemInclude-1-pag3.php:** este fichero es un fichero html que contiene etiquetas HTML. Si se llama a la función dentro del body no da error pero si se observa el código con las herramientas de desarrollador, no se obtiene el resultado esperado.

Ejemplo Incluir ficheros externos: ejemplo carpeta 13_funciones_include3

Este ejemplo consta muestra como las variables que se definen en ficheros que se incluyen pueden usarse en el fichero que los incluye

El fichero **pag1.php** define una variable "$i" y le asigna el valor 1. El fichero **pag2.php** suma 10 al valor de la variable "$i". Por último, el fichero **pag0.php** utiliza la instrucción "include" para incluir el código de los otros dos ficheros (el fichero pag2.php lo incluye 2 veces) y mostrar el contenido de la variable.

Se pueden incluir ficheros en cualquier punto del programa. Estos a su vez pueden incluir otros ficheros. Estos ficheros pueden contener variables, constantes, funciones, … o pueden incluir la parte lógica de un programa generar salida, modificar variables… pero no se deben hacer ambas cosas en un mismo fichero.

3.2 Ejecución y creación de funciones

- Las funciones de usuario son bloques de código independientes que sirven para desarrollar tareas concretas.
- Los nombres de las funciones no pueden empezar por un número.
- En la definición de las funciones no se indica el tipo de dato de los argumentos, ni el tipo de dato de retorno de la función (en caso de que devolviera algo).
- Para devolver valores se usa la sentencia **return**. Por defecto, si no devuelve nada, devuelve **null.**
- En caso de que al llamar una función no se incluyan todos los parámetros que necesita, se producirá un error.

Ejemplo: 13_funciones_1_ambitoVariables.php

- En PHP puede utilizarse las variables en cualquier punto del programa
- Las variables que se definen en el interior de una función solo son válidas dentro de esa función. Si hay variables con el mismo nombre fuera y dentro de una función php las toma como diferentes.
- En el ejemplo podemos ver que tenemos dos variables con el mismo nombre, una definida dentro de la función duplicar, y otra dentro del cuerpo principal del script. Se comprueba que las dos variables son diferentes al mostrar el valor de la variable definida fuera de la función antes y después de llamarla.
- Después vemos un ejemplo de cómo se devuelve un resultado desde una función.
- El siguiente ejemplo muestra como si se intenta acceder a una variable definida fuera desde una función, esa variable no existe.
- El siguiente ejemplo muestra cómo definir una variable como global para poder usarla en una función (de usa la palabra global dentro de la función).

3.3 Argumentos

Argumentos, lista de variables que se le pasan a la función separadas por comas. No se indica el tipo de argumento ni el tipo de dato que devuelve (si devolviera alguno)

Por defecto si no devuelve nada retorna null.

Se puede indicar **valores por defecto para los argumentos**, de forma que cuando hagas una llamada a la función puedes no indicar el valor de un argumento; en este caso se toma el valor por defecto indicado. Los valores por defecto tienen que estar a la derecha de cualquier otro valor sin valor por defecto en la lista de argumentos.

```
<?php
function precio_con_iva($precio, $iva=0.18) {
return $precio * (1 + $iva);
}
$precio = 10;
$precio_iva = precio_con_iva($precio);
print "El precio con IVA es ".$precio_iva
?>
```

Para pasar valores por referencia en lugar de por valor se antepone el símbolo & al nombre de la variable. &$var.

4.- TIPOS DE DATOS COMPUESTOS

4.1 Recorrer arrays

Ejemplo: 11_ejerciciosArrays_02_recorrer.php

- En este ejemplo se muestran diferentes formas de crear arrays en PHP y algún ejemplo de arrays asociativos.
- Arrays asociativos: los índices son textos en lugar de números. Si se intenta acceder usando un nº da error.
- Si una clave aparece repetida al referenciar a ese elemento aparecerá el último.

- Los arrays asociativos no se pueden manejar mediante un bucle tradicional ya que es puede ocurrir que no se conozcan las claves ni la posición. Para recorrer este tipo de bucles utilizaremos la instrucción **foreach** (que veremos más adelante)

4.2 Funciones relacionadas con los arrays

Ejemplo: 11_ejerciciosArrays_03_multidimensional.php

- **count():** función para saber el nº de elementos de un array o un objeto.
- **sizeof()** es un alias de count
- Si el array no está inicializado o es null devolverá 0.
- Por defecto no cuenta el nº de elementos en un array multidimensional. Para que lo haga, hay que indicarle que lo haga de forma recursiva indicándole el parámetro COUNT_RECURSIVE.
- Al mostrar el array tridimensional "población" vemos que hay un primer array de dos elementos (España y Alemania), el array España a su vez contiene un array de dos elementos (Castilla y León y Asturias), Castilla y León es un array de 2 elementos (Palencia y Valladolid), Asturias es un array de un elemento (Oviedo), y Alemania contiene un array de un elemento (Brandeburgo), que a su vez contiene un array de un elemento (Berlín)
- https://www.php.net/manual/es/function.count.php

Ejemplo: 11_arrays_06_foreach

- Bucle que se utiliza para recorrer arrays de forma sencilla sin tener que preocuparnos de si hay elementos indefinidos ya que el intérprete es capaz de ignorarlos. Ideal para recorrer arrays asociativos. Funciona con arrays y objetos.
- Permite recorrer cada elemento del array asignándole un nombre más sencillo de procesar, ya no nos preocuparemos de utilizar un índice.
- Hay dos sintaxis:
    - foreach(array as $valor): recorre el array. En cada vuelta el valor del elemento actual se asigna a la variable $valor y el puntero interno del array avanza una posición.
    - foreach(array as $clave=>$valor): además asigna a la variable $clave la clave del elemento actual.
- En el ejemplo partimos de un array unidimensional y lo recorremos de la forma tradicional, utilizando índices. Después utilizamos el bucle foreach con las dos posibles sintaxis para ver la diferencia.
- Hacemos lo mismo con un array asociativo. Podemos ver que en el primer caso del bucle foreach, accedemos directamente al valor del elemento del array, en el segundo caso, la clave corresponde a la clave de cada elemento del array (nombre) y el valor a su contenido
- Vamos a ver cómo se recorren arrays multidimensionales usando foreach. Partimos del array bidimensional agenda. El índice de la primera dimensión del array es numérico y el de la segunda texto (array asociativo).
- Si intentamos hacer un foreach simple, nos dará error ya que el valor de cada elemento del primer array es otro array, no se puede mostrar con la función echo,

da error al intentar convertir un array a string. Si usamos la segunda sintaxis la clave (contacto) contiene el valor del índice del primer array, y valor el array que está contenido dentro de ese índice (un contacto de la agenda).

- Usando la segunda sintaxis foreach(array as clave => valor) en el primer bucle, clave tiene el índice de cada elemento del primer array y valor, su contenido, que es un array asociativo";
- usando dos bucles foreach, el primero con sintaxis simple, foreach(array as valor), donde para cada elemento de array valor contiene un array asociativo, y el segundo bucle usando también la sintaxis simple donde para cada elemento del segundo array se muestra el valor que contiene, en este caso el dato final.
- usando la segunda sintaxis en el segundo bucle, la variable contacto del primer bucle contiene un array.
- si usamos la sintaxis foreach(array as clave =>valor en los dos bucles, en el primero la clave equivale al índice numérico del primer array, y el valor al array asociativo de cada contacto
- el último ejemplo prueba un array asociativo

11_arrays_08_otrasFunciones

Existen otras funciones que podemos utilizar con los arrays, algunas son:

- **Array_push():** inserta un elemento al final del array. Para añadir un elemento al final del array también se puede usar $array[]=valor; **este último método es más recomendable puesto que evitamos llamar a una función.**
- **Array_unshift()** inserta un elemento al principio del array
- **Array_splice():** permite tanto insertar como borrar.
    - Para insertar: en el primer parámetro indicamos el array, en el segundo parámetro decimos en qué posición queremos insertar, en el tercero un 0 (es el que se usa para borrar), y en el cuarto el valor que queremos poner en dicha posición. En el ejemplo estamos insertando el valor 88 en la posición 2, desplazando lo que hubiera en esa posición anteriormente.
    - Para borrar, si en el tercer parámetro indicamos un valor distinto de 0 estamos señalando cuántos elementos queremos borrar. En la primera parte del ejemplo borra el elemento 88 sustituyéndolo por “DWES”. En la segunda parte borra dos elementos, DWES y miércoles
- **Con array_shift()** borraremos el primer elemento del array
- **Con array_pop()** borramos el último. Podemos guardar en una variable el elemento que se ha borrado en ambos casos.
- La función **array_unique ()** devuelve un array sin duplicados.
- Con **unset()** podemos eliminar el array o un elemento concreto del array. Si lo volvemos a crear vacío elimina lo que contuviera.
- **sort()** ordena de forma ascendente un array
- **rsort()** lo ordena de forma descendente.
- **array_reverse()** lo ordena en orden inverso, es decir el elemento con el último índice pasa a ser el primero.

- Para hacer estas operaciones con arrays asociativos se usa **asort(), arsort()**
- **array_keys()** obtiene un array con las claves existentes en un array asociativo, si se le pasa un valor a buscar devuelve un array con las claves en las que se encuentra dicho valor o un array vacío si no lo encuentra. Al final del ejemplo, hay un array asociativo de superhéroes. Usamos la instrucción array_keys para obtener las claves del primer array y luego usamos un bucle for para ir accediendo a cada elemento mediante si clave **(útil si no conocemos las claves)**. Por último otro ejemplo, va guardando las claves del segundo array.
- Función **array_key_exists()** permite averiguar si un array asociativo contiene una clave.

11_arrays_09_otrasFunciones1.php

- Para extraer parte de un array podemos usar la función **array_slice()** que devuelve los elementos del array a partir de la posición indicada. En el segundo parámetro se indica la posición a partir de la cual se extraerán los elementos, y en el tercer parámetro cuantos elementos se van a extraer. Si se omite se extraen hasta el final del array.
- La función **array_diff()** devuelve un nuevo array **con los valores en el primer array que no estén en el resto de arrays pasados como parámetros**. En el primer ejemplo el valor 22 es el único del primer array que no está en el segundo. En el segundo ejemplo, el valor naranja es el único que no está en los otros.
- La función **array_dif_key()** devuelve un array con las claves del primer array pasado en el primer parámetro que no están en ninguno de los otros. Si cambiamos el orden de los arrays en la llamada a la función, cambia el resultado.
- Hay muchas funciones relacionadas con los arrays en la documentación de PHP: https://www.php.net/manual/es/function.array-diff.php

11_arrays_10_otrasFunciones2.PHP

- Con la función **implode()** extraemos los valores contenidos en un array en formato de cadenas de texto especificando en el primer parámetro el texto delimitador entre ellos (si no se indica nada será una cadena vacía).
- Con la función **explode()** obtenemos un array a partir de una cadena de texto, indicando en el primer parámetro el carácter separador.
- Con la función **str_split()** creamos un array a partir de una cadena pudiendo especificar el nº de caracteres que va a haber en cada elemento del array. Si no se especifica nada cada elemento contendrá un carácter incluyendo los espacios.

5.- FORMULARIOS WEB

- Los formularios permiten hacer llegar datos de los usuarios desde un navegador a una aplicación web.
- Los formularios HTML van encerrados entre las etiquetas <form> y </form>, y los elementos sobre los que puede actuar un usuario son principalmente input, select, textarea, button.

- **Es importante que los elementos del formulario susceptibles de ser recogidos y tratados en el lado servidor tengan valor en el atributo name del elemento, ya que este es el nombre de la clave por la que accederán a su contenido.**
- El atributo **action** del elemento FORM indica la página a la que se le enviarán los datos del formulario. En nuestro caso se tratará de un guion PHP.
- El atributo **method** especifica el método usado para enviar la información. Este atributo puede tener dos valores:
    - **get**: con este método los datos del formulario se agregan al URI utilizando un signo de interrogación "?" como separador.
    - **post:** con este método los datos se incluyen en el cuerpo del formulario y se envían utilizando el protocolo HTTP.
- Los datos se recogerán de distinta manera dependiendo de cómo se envíen.

¿Cómo se procesa la información enviada por un formulario? Usando las variables superglobales $_GET , $_POST o $_REQUEST

Argumentos POST

- Como ya hemos dicho usar GET es inseguro ya que los datos que enviamos están visibles en la URL. No debemos enviar datos como contraseñas o datos bancarios.
- Los datos se envían en el cuerpo del mensaje HTTP en lugar de en la cabecera, por lo que no se ven a simple vista, pero hay herramientas como firebug que permiten verlos por lo que el método POST es algo más seguro, aunque en caso de envío de datos sensibles conviene cifrarlos de alguna manera antes.
- La forma de recuperar los datos que se envían usando el método POST es utilizando la variable superglobal $_POST Ejemplo 14_superglobal_2_Post.php
- Con $_REQUEST recuperas tanto las enviadas con GET, POST y las COOKIES.

Elementos de tipo radio

- Los elementos de tipo radio de los formularios deben tener el mismo valor en el atributo "name", ya que es la forma en que los elementos se vinculan entre sí y hace que las elecciones sean excluyentes una de la otra. En el atributo value se pondrá el valor que tendrán esos campos y que podremos comprobar en el servidor.
- En el servidor se comprueba si el contenido del array asociativo $_POST con clave el valor del atributo name que pusimos en los radio button es 1 , que es el valor que llega del atributo value) se hará lo que sea. **Ten en cuenta que el formulario solo envía el radiobutton que se ha seleccionado, no los dos.**

Elementos de tipo checkbox

- Los elementos de tipos checkbox permiten seleccionar una a varias opciones. En este ejemplo, podemos ver que el atributo name tiene un valor diferente para cada elemento checkbox y un atributo value, igual para todos.
- En el servidor se comprueba para cada checkbox si el valor es el que indicamos en el atributo value, lo que indicará que fue seleccionado en el formulario.

- otra forma de indicar que se ejecute el script actual es no poner nada en el action. Recuerda que cuando pulsamos el botón de envío de un formulario, el navegador hace una petición http del servidor enviándole el contenido del formulario cuyos atributos name tengan valor. Cuando la petición llega al servidor, como en este caso es el mismo script el que contiene el código PHP, se ejecutaría la parte del servidor, y se genera la respuesta que se devuelve al navegador, que de nuevo es el formulario web en su estado inicial. Esta no es la forma habitual de trabajo, se ha hecho en el ejemplo por tener todo el código a la vista.

Ejemplos

Otra forma de trabajar con checkbox

Ejemplo: carpeta 15_formularios_1

- En el fichero index.php hay un formulario con un campo de tipo input en el que se deberá introducir el nombre de una persona, y dos campos de tipo checkbox correspondientes a módulos que se cursan. Observa que el atributa name de los checkbox contiene un nombre con corchetes. Esa es la forma de pasar en un array los elementos checkbox que han sido seleccionados.
- En el ejemplo al no comprobar si un elemento del array existe antes de asignarle su valor a una variable, si en el formulario pulsamos el botón enviar sin haber rellenado ningún campo del formulario daría error. Por eso es importante la validación y comprobación de los datos tanto en el cliente como en el servidor **siempre que sea posible es preferible validar los datos en el navegador antes de enviarlos al servidor, usando HTML5 y/o JS.**
- Ahora recibimos todos los checkbox que han sido seleccionados en el formulario, pero no necesitamos saber el name de cada uno de ellos, podemos recorrer el array e seleccionados con un bucle foreach. El código así es más limpio.
- También se podría usar esta técnica para radiobuttons, pero no tiene mucho sentido ya que este tipo de botones solo permiten que se seleccione un elemento.

Ejercicios con formularios

A continuación se proponen una serie de ejercicios para practicar con formularios. Se adjunta la solución en la carpeta indicada.

Ejercicio carpeta 15_formularios_ejercicios_1

Este ejemplo consta de dos ficheros. Un fichero HTML en el que hay un formulario con un par de campos de tipo input y el botón enviar. En el “action” se pone el nombre del fichero al que se va a ir al enviar el formulario.

Fichero formularios_1.php

En el segundo fichero se comprueba si el array asociativo $_POST está vacío. En caso contrario se muestra, en primer lugar, el contenido del array asociativo, que contiene todos los campos del formulario, incluido el botón “Enviar formulario”, y en segundo lugar los campos del formulario de tipo texto accediendo por el valor definido en el campo “name” del formulario.

Ejercicio carpeta 15_formularios_ejercicios_2: comprobar los campos requeridos

Fichero formularios_2.html

Este fichero contiene un formulario con tres campos de tipo texto para introducir el nombre, apellidos y edad, un campo de tipo checkbox y el botón de envío de formulario.

Fichero formularios_2.php

En este fichero además de comprobar si el array asociativo $_POST no está vacío, comprueba si están definidos los campos nombre y apellidos. Si el checkbox se ha seleccionado, muestra el valor de su estado (1). Si no se ha seleccionado no se envía

Ejercicio carpeta 15_formularios_ejercicios_3: EJERCICIO CONTROLES_SIMPLES_Y_ARRAYS

La solución de este ejercicio está en la carpeta controles_simples_y_arrays

En este ejercicio se muestra la diferencia entre recibir de un formulario campos como controles simples o como array, es decir, que en el atributo name figure un nombre simple o un array. Consta de dos ficheros:

Fichero index.html

Contiene un formulario con campos de tipo texto y checkbox. Los campos de texto simple, aunque tienen el mismo nombre en el atributo name, el archivo php que procesa los datos solo toma el último escrito en el código HTML. En cambio, los campos de tipo texto con la etiqueta “texto array” aunque tienen el mismo nombre en la etiqueta name, al ser pasados como un array, si llegan todos al script php.

En cuanto los checkbox en el primer caso, en el que no se pasa como array, recibe el último checkbox marcado (en su orden), es decir si están marcados los tres solo recibe alemán, si están marcados los dos primeros recibe francés. En cambio, en el segundo caso en que se pasa como array, recibe todos los que estén marcados (los que no se hayan seleccionado no se envían.

Controles simples independientes frente a array de controles

CAMPOS DE TEXTO:

Independientes, pero con nombre duplicado

Texto simple 1: alfa `<input type="text" name="textoSimple" value="alfa" />`

Texto simple 2: beta `<input type="text" name="textoSimple" value="beta" />`

En array

Texto array 1: gamma `<input type="text" name="textoArray[]" value="gamma" />`

Texto array 2: delta `<input type="text" name="textoArray[]" value="delta" />`

CHECKBOX:

Independientes, pero con nombre duplicado

Inglés ☐ Francés ☑ Alemán ☐

```
<input type="checkbox" name="cbSimple" value="English" />
<input type="checkbox" name="cbSimple" checked="true" value="Francais" />
<input type="checkbox" name="cbSimple" value="Deutch" />
```

En array

Inglés ☐ Francés ☑ Alemán ☐

```
<input type="checkbox" name="cbArray[]" value="English" />
<input type="checkbox" name="cbArray[]" checked="true" value="Francais" />
<input type="checkbox" name="cbArray[]" value="Deutch" />
```

Resertear el formulario | Enviar formulario al servidor

Ejercicio carpeta 15_formularios_ejercicios_5: Formularios: aplicaciónTablas

Realizar una aplicación que imprima en pantalla una tabla con N filas y M columnas. Estos valores N y M se reciben a través de un formulario WEB que pregunta por estos dos valores, número de filas y número de columnas. Como máximo las tablas podrán tener 10 filas y 10 columnas.

Además, el color de fondo de las filas pares será #00FFFF y el de las pares #E0FFFF.

Cada celda de la tabla resultante deberá contener un texto que indique el número de fila y el número de columna de la celda, de la siguiente manera por ejemplo para una tabla 2x3:

Fila 1 - Columna 1	Fila 1 - Columna 2	Fila 1 - Columna 3
Fila 2 - Columna 1	Fila 2 - Columna 2	Fila 2 - Columna 3

Ejercicio carpeta 15_formularios_ejercicios_6: Formularios: calcular la edad

Realizar un script PHP que calcule la edad a partir de una fecha de nacimiento que se introduce en un formulario. El script comprobará que la fecha introducida es anterior a la fecha actual.

Ejercicio carpeta 15_formularios_ejercicios_7: Formularios: calculadora sencilla

Realiza una calculadora sencilla utilizando un formulario. La práctica constará de dos archivos, el primero llamado “calculadora.html” contendrá un formulario con dos campos en el que se introducirán los números con los que se va a operar, cuatro campos de tipo **checkbox** con las cuatro operaciones aritméticas posibles a realizar y un botón de envío de formulario que enviará el formulario a la página “calcula.php”.

El segundo archivo llamado “calcula.php” comprobará si han llegado datos. Si no han llegado se redirige la página a la página calculadora.html. Si los ha recibido comprueba que los datos con lo que se va a operar son numéricos. Si es así realizan las operaciones que se hayan seleccionado y muestra el resultado. En caso contrario muestra un mensaje de error.

A la hora de hacer la división comprobará si el divisor es 0 mostrando un mensaje de error en caso de ser así.

Ejercicio carpeta 15_formularios_ejercicios_8: Formularios: conversor de euros

Hacer un conversor de euros a libras o a dólares (que el usuario elija una moneda y sólo una) entre dos posibles utilizando radiobuttons.

Valor de cambio:

1€ = 1,13742 $

1€ = 0,889868 libras

Ejercicio carpeta 15_formularios_ejercicios_9: Formularios: coger datos de formulario

Escribe un programa PHP que permita al usuario rellenar un formulario de registro con los datos de nombre, contraseña, email, fecha de nacimiento, tienda (campo de tipo select con tres opciones: Madrid, Valencia y Barcelona), edad y suscripción (campo de tipo checkbox, por defecto marcado). El programa recibe los datos del formulario y los muestra en pantalla tal y como los escribió el usuario.

Ejercicio carpeta 15_formularios_ejercicios_11: Formularios: trabajar con texto y array

Escribe un programa PHP que permita al usuario introducir un texto mediante una caja de texto. El programa informará al usuario si en dicho texto se incluyen palabras que comience por mayúscula, tenga entre 8 y 10 letras, contenga 4 vocales y termine en “ero”.

El programa devolverá un listado con las palabras que cumplan dichas condiciones. Dichas palabras aparecerán en mayúsculas y separadas por un ‘-‘ y sin repetirse.

Se listarán ordenadas de mayor a menor longitud y si hay palabras de igual longitud se ordenarán en orden alfabético. Podemos asumir que las palabras pueden estar separadas por blancos, carácter de nueva línea, tabuladores, puntos, comas, dos puntos y punto y coma.

Fin del resumen