Tema 2. Características del lenguaje PHP.

# 1.- Elementos del lenguaje PHP.

Tienes que saber ya: las variables y tipos de datos, comentarios, operadores y expresiones.

Las instrucciones PHP se integran en el código HTML con los delimitadores: <?php ?>

Los programas escritos en PHP suelen estar escritos en varios **bloques de código**. Cada bloque debe ir entre dichos delimitadores, y en caso de que genere alguna salida, ésta se introduce en el código HTML en el mismo punto en el que figuran las instrucciones en PHP.

Por ejemplo:

<body>

<?php $a=1; ?>

<p>Página de prueba</p>

<?php $b=$a; ?>

Aunque no se utilice el valor de las variables, en el segundo bloque de código la variable $a mantiene el valor 1 que se le ha asignado anteriormente.

Para crear un nuevo proyecto en NetBeans, vete a Archivo > Proyecto nuevo > PHP Application.

En la siguiente pantalla de configuración del proyecto, debes indicar la ruta que se usará para acceder al mismo desde un navegador. Es decir, por ejemplo, si quieres guardar tu proyecto en /home/usuario/NetBeansProjects/DWES, **tienes que hacer que el servidor web Apache pueda acceder a la ruta anterior en la que vas a almacenar tus proyectos**. Esto puedes hacerlo, por ejemplo, **creando un enlace simbólico en la raíz del servidor web** (/var/www):

sudo ln –s /home/usuario/NetBeansProjects/DWES/ DWES

De esta forma, si creas un proyecto nuevo en la ruta **/home/usuario/NetBeansProjectos/DWES/Proyecto1**, la URL que tendrás que poner en la siguiente pantalla de configuración será **http://localhost/DWES/Proyect**1.1.- Generación de código HTML.

* void **echo** (string $arg1, ...); : puede recibir más de un parámetro.
* int **print** (string $arg);: a diferencia de echo, solo puede recibir un parámetro.
  + Ejemplo de uso de la función print. [Ver Anexo I](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%201.html).

Realmente, ninguna son funciones, por lo que **no es obligatorio poner paréntesis cuando las utilicemos**.

* **printf**: puede recibir varios parámetros:
  + Cadena de texto que indica el formato que se ha de aplicar. Debe contener un especificador de conversión por cada uno de los demás parámetros que se le pasen a la función, y en el mismo orden.
  + Resto de parámetros a formatear.

Texto

Descripción generada automáticamente con confianza media

* **sprintf**: similar a printf, pero permite guardar la salida en una variable.
  + Por ejemplo:

$txt\_pi = sprintf("El número PI vale %+.2f", 3.1416);

[Especificadores de conversión](https://es.acervolima.com/php-especificadores-de-formato/)

**Cada especificador de conversión** **va precedido del carácter %** y se compone de las siguientes partes:

* **signo** (opcional). Indica si se pone signo a los números negativos (por defecto) o también a los positivos (se indica con un signo +).

**Ejemplo**: % + d // Especifica el número entero junto con su signo (+ o -).

* **relleno** (opcional). Indica que carácter se usará para ajustar el tamaño de una cadena. Las opciones son el carácter 0 o el carácter espacio (por defecto se usa el espacio).

% '0d // Rellena con 0s para lograr la longitud correcta.

* **alineación** (opcional). Indica que tipo de alineación se usará para generar la salida: justificación derecha (por defecto) o izquierda (se indica con el carácter -).

% -s // Especifica la alineación justificada a la izquierda.

* **ancho** (opcional). Indica el mínimo número de caracteres de salida para un parámetro dado.

// Especifica que debe haber al menos 5 dígitos,

% '05d // si es menor, se rellenan 0 para obtener el resultado deseado.

* **precisión** (opcional). Indica el número de dígitos decimales que se mostrarán para un número real. Se escribe como un digito precedido por un punto.

% .5f // Define la precisión del número real.

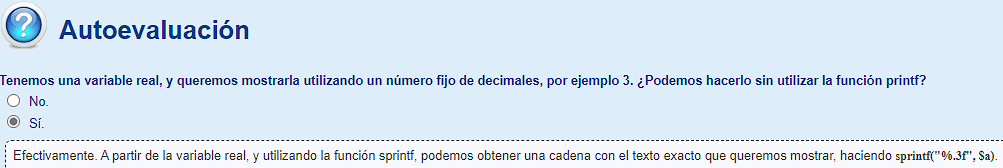
% .2s // Carácter máximo permitido en una string.

* **tipo** (obligatorio). Indica cómo se debe tratar el valor del parámetro correspondiente. En la siguiente tabla puedes ver una lista con todos los especificadores de tipo.

|  |  |
| --- | --- |
| **Especificadores de tipo para las funciones printf y sprintf** | |
| **Especificador** | **Significado** |
| b | Convierte un número entero a binario. |
| c | Convierte el valor entero ASCII a su carácter correspondiente. |
| d | Le pasas un número entero por argumento y te saca su valor en el sistema decimal. |
| u | Le pasas un número entero por argumento y te saca su valor en el sistema decimal, pero sin signo. |
| o | Le pasas un número entero por argumento y te saca su valor en el sistema octal. |
| x | Le pasas un número entero por argumento y te saca su valor en el sistema hexadecimal (con minúsculas). |
| X | Le pasas un número entero por argumento y te saca su valor en el sistema hexadecimal (con mayúsculas). |
| f | Le pasas un número double (con parte decimal) por argumento y te saca su valor en número de coma flotante (teniendo en cuenta la configuración regional o localidad). |
| F | Le pasas un número double (con parte decimal) por argumento y te saca su valor en número de coma flotante (sin tener en cuenta la configuración regional o localidad). |
| e | Sirve para indicar por argumento un nº en notación científica, utilizando la e minúscula. Por ejemplo: 1.2e+3 |
| E | Sirve para indicar por argumento un nº en notación científica, utilizando la E mayúscula. Por ejemplo: 1.2E+3 |
| g | Pasas un número double o en notación científica, y se muestra la forma más corta entre %f y %e. |
| G | Pasas un número double o en notación científica, y se muestra la forma más corta entre %F y %E. |
| s | Sirve para mostrar una cadena. |
| % | Se muestra el carácter %. No necesita argumento. |

**Ejemplo**: en la salida, el número PI se obtiene con signo y solo con dos decimales.

printf("El número PI vale %+.2f", 3.1416); // +3.14



### Anexo I. Ejemplo de utilización de la función print.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "

http://www.w3.org/TR/html4/loose.dtd">

<!-- Desarrollo Web en Entorno Servidor -->

<!-- Tema 2 : Características del Lenguaje PHP -->

<!-- Ejemplo: Utilización de print -->

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Desarrollo Web</title>

</head>

<body>

<?php

$modulo="DWES";

print "<p>Módulo: ";

print $modulo;

print"</p>"

?>

</body>

</html>

## 1.2.- Cadenas de texto. Secuencias de escape. Operadores.

En PHP, las cadenas de texto pueden usar tanto comillas simples como comillas dobles. Sin embargo:

* **Con comillas dobles**: podemos usar las variables directamente, sin hacer concatenaciones, y sin usar print varias veces para imprimir la variable, como en el Anexo II.

<?php

$modulo="DWES";

print "<p>Módulo: $modulo</p>"

?>

Para que PHP distinga correctamente el texto que forma la cadena del nombre de la variable, **a veces es necesario rodearla entre llaves**:

print "<p>Módulo: ${modulo}</p>"

* **Con comillas simples**: debemos concatenar para usar las variables.

Secuencias de escape

Tenemos una serie de secuencias de escape. Hay dos especiales, sombreados de color amarillo.

|  |  |  |
| --- | --- | --- |
| **Secuencia** | **Resultado** | **Usos** |
| **\\** | Se imprime una barra invertida. Se puede usar **también** **con** cadenas especificadas **con** **comillas simples ‘**. | “ y ‘ |
| **\’** | Se muestra una comilla simple. **Solo se utiliza** cuando especificamos una cadena **con comillas simples ’**. | Solo ‘ |
| **\”** | Se muestra una comilla doble. | Solo “ |
| **\n** | Se muestra un avance de línea (LF o 0x0A (10) en ASCII). | Solo “ |
| **\r** | Se muestra un retorno de carro (CR o 0x0D (13) en ASCII). | Solo “ |
| **\t** | Se muestra un tabulador horizontal (HT o 0x09 (9) en ASCII). | Solo “ |
| **\v** | Se muestra un tabulador vertical (VT o 0x0B (11) en ASCII). | Solo “ |
| **\f** | Se muestra un avance de página (FF o 0x0C (12) en ASCII). | Solo “ |
| **\$** | Se muestra un signo del dólar. | Solo “ |

Operadores exclusivos de cadenas

* **Operador de concatenación (.)**: te permite unir las cadenas de texto que le apses como operandos.
* **Operador de asignación y concatenación (.=)**: concatena al argumento del lado izquierdo la cadena del lado derecho.

<?php

$a = "Módulo ";

$b = $a . "DWES"; // ahora $b contiene "Módulo DWES"

$a .= "DWES"; // ahora $a también contiene "Módulo DWES"

?>

* **Heredoc (<<<)**: consiste en poner el operador <<< seguido de un identificador, y a continuación, en la siguiente línea, se empieza la cadena de texto, sin utilizar comillas.
  + **La cadena finaliza cuando escribes ese mismo identificador en una nueva línea**. Esta línea de cierre no debe llevar más caracteres, espacios ni sangría, solo **punto y coma después del identificador**.

<?php

$a = <<<MICADENA

Desarrollo de Aplicaciones Web<br />

Desarrollo Web en Entorno Servidor

MICADENA;

print $a;

?>

* + El texto se procesa de igual forma que si fuera una cadena entre comillas dobles, sustituyendo variables y secuencias de escape.
  + Si no quisieras que se realizara ninguna sustitución de secuencias de escape, debes poner el identificador de apertura entre comillas simples:

$a = <<<'MICADENA'

...

MICADENA;

## 1.3.- Funciones relacionadas con los tipos de datos (I). Gettype y más.

En primer lugar: los tipos de datos simples existentes en PHP:

* **Booleano** (boolean).
* **Entero** (integer).
* **Real** (float).
* **Cadena** (string).
* **null**: es un tipo de datos especial, que se usa para indicar que la variable no tiene valor.

Funciones para conocer el tipo de dato de una variable

* **gettype**: devuelve una cadena de texto, que puede ser:
  + **array**, **boolean**, **double**, **integer**, **object**, **string**, **null**, **resource** o **unknown type**.
* is\_array()
* is\_bool()
* is\_float()
* is\_integer()
* is\_null()
* is\_numeric()
* is\_object()
* is\_resource()
* is\_scalar()
* is\_string()

Funciones para hacer casting (conversión de datos explícita)

* **settype(variableACambiar, parametro)**; donde **parámetro** puede ser una de las siguientes cadenas:
  + boolean, integer, float, string, array, object o null.

<?php

$a = $b = "3.1416"; // asignamos a las dos variables la misma cadena de texto

settype($b, "float"); // y cambiamos $b a tipo float

print "\$a vale $a y es de tipo ".gettype($a);

print "<br />";

print "\$b vale $b y es de tipo ".gettype($b);

?>

Funciones relacionadas con la definición de variables

* isset: permite saber si una variable está definida y no es null.
* unset: destruye la variable o variables que se le pasa por parámetro.
* empty: permite saber si una variable se considera vacía debido al valor que contiene.

Constantes

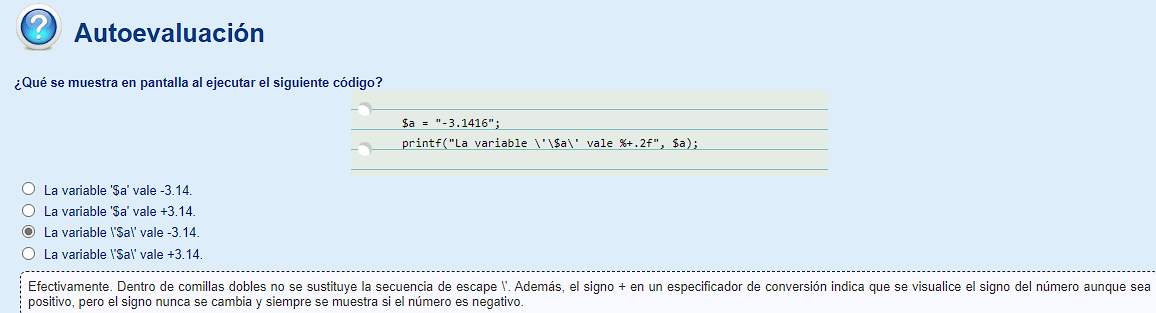
* Para definirlas, se usa la siguiente función:

bool define (string $identificador, mixed $valor [, bool $case\_insensitive = false ] );

También se pueden definir así:

const MIN\_VALUE = 0.0;

* Los identificadores de las constantes no van precedidos por el signo “$” y suelen escribirse en mayúsculas.
* El tercer parámetro indica si quieres que sea sensible a mayúsculas y minúsculas (false) o no (true).
* **Solo permiten los siguientes tipos de valores**: integer, float, string, boolean y null.



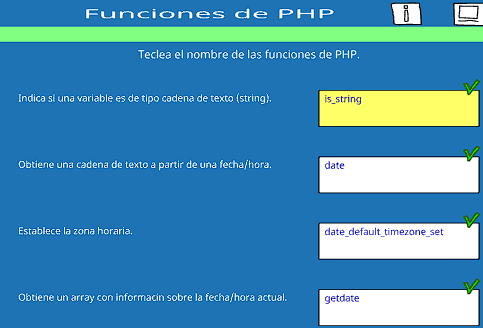
### 1.3.1.- Funciones relacionadas con los tipos de datos (II). Fechas

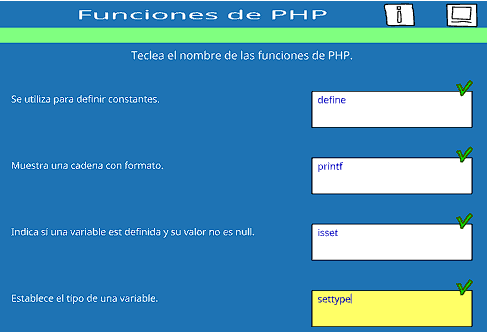
En PHP no existe un tipo de datos específico para trabajar con fechas y horas. La información de fecha y hora se almacena internamente como un número entero. Sin embargo, PHP te ofrece **funciones** para trabajar de manera cómoda con fechas.

* **string date** (string $formato [, int $timestamp]);:
  + Esta función permite obtener una cadena de texto a partir de una fecha y hora, con el formato que tu elijas.
  + **Primer parámetro**: descripción del formato.
  + **Segundo parámetro (opcional)**: número entero que identifica la fecha. Si no se indica, se utilizará la hora local para crear la cadena de texto.

|  |  |  |
| --- | --- | --- |
| **Función date**: caracteres de formato para fechas y horas. | | |
| **Tiempo** | **Carácter** | **Resultado** |
| **Día de la semana** | **N** | Día de la semana (1 = lunes,…, 7 = domingo) |
| **w** | Día de la semana (0 = domingo,…, 6 = sábado) |
| **l** | Texto del día de la semana, en inglés (Monday,…, Sunday). |
| **D** | texto del día de la semana, solo tres letras, en inglés ( Mon, ..., Sun |
| **Día del mes** | **d** | Día del mes con dos dígitos. |
| **j** | Día del mes con uno o dos dígitos (sin ceros iniciales). |
| **t** | Número de días que tiene el mes. |
| **Mes** | **m** | Número del mes, con dos dígitos. |
| **n** | número del mes con uno o dos dígitos ( sin ceros iniciales ). |
| **F** | texto del mes, en inglés (January , ..., December). |
| **M** | texto del mes, solo tres letras, en inglés ( Jan, ..., Dec ). |
| **Año** | **Y** | Número del año |
| **y** | Número del año, solo dos últimos dígitos. |
| **L** | 1 si es año **bisiesto**, 0 si no lo es. |
| **Día del año** | **z** | Día del año, comenzando por el cero (0 = 1 de enero). |
| **Semana del año** | **W** | Número de la semana del año |
| HORAS | | |
| **Horas** | **h** | Horas, dadas en formato de 12 horas, siempre con dos dígitos. |
| **H** | Horas, dadas en formato de 24 horas, siempre con dos dígitos |
| **g** | Horas, dadas en formato de 12 horas, con uno o dos dígitos (sin ceros iniciales ). |
| **G** | Horas, dadas en formato de 24 horas, con uno o dos dígitos ( sin ceros iniciales ). |
| **Minutos** | **i** | Minutos, siempre con dos dígitos. |
| **Segundos o inferior** | **s** | Segundos, siempre con dos dígitos |
| **u** | Microsegundos |
| **Formato** | **a** | Escribe **am** (de 0:00 a 12:00) o **pm** (de 12:00 a 0:00). |
| **A** | Escribe **AM** (de 0:00 a 12:00) o **PM** (de 12:00 a 0:00). |
| FECHA ENTERA | | |
| **Fecha + hora incluida** | **r** | Fecha entera con formato RFC 2822. |

* **date\_default\_timezone\_set('Europe/Madrid');**: establece la **zona horaria**. En este caso, establece la zona horaria en España peninsular.
  + [Consultar distintas zonas horarias soportadas en PHP](https://www.php.net/manual/es/function.date-default-timezone-set.php).
* [**getdate**](https://www.php.net/manual/es/function.getdate.php)**(int $timestamp = time()): array**: devuelve un array con información sobre la fecha y hora actual.
* **mktime**(hora = date(“H”), minuto = date(“i”), segundo = date(“s”), mes = date(“n”), día = date(“j”), año = date(“Y”), is\_dst = -1): genera un timestamp (representación interna de la fecha y hora en Unix, que no es más que los segundos transcurridos desde el 1 de enero del 1970 00:00:00 GMT)
* **strtotime**(string $time, int $now = time()): es como mktime, pero permite dar la fecha en formato string, para pasarla directamente a formato timestamp.
  + El argumento **time** se puede dar siguiendo los siguientes [formatos](https://www.php.net/manual/es/datetime.formats.php):
    - **notación española, día mes y año**: “1-1-2022”
    - **notación inglesa, mes, día y año**: "12/22/78", "1/17/2006", "1/17/6".
    - **año, mes y día**: "2008/6/30", "1978/12/22", "2008-6-30", "78-12-22", "8-6-21".
    - Y muchas más que puedes consultar en la página de PHP.
* Más [Información sobre más funciones relacionadas con las fechas](https://www.php.net/manual/es/ref.datetime.php).





## 1.4.- Variables especiales de PHP.

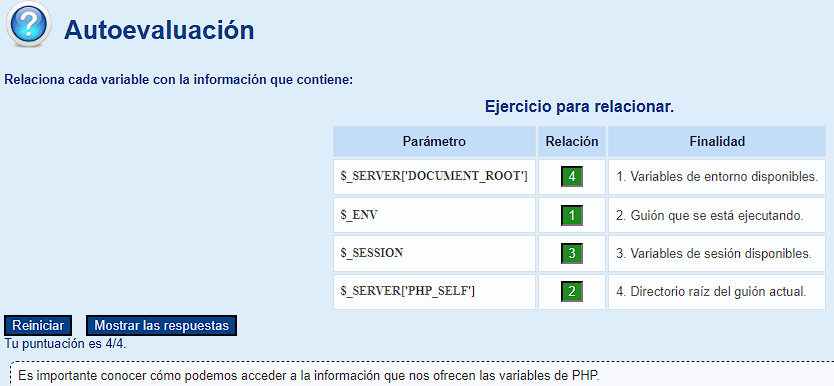
* **Variables globales**.
* **Variables superglobales**: son variables globales que incluye PHP, para las cuales no es necesario ni si quiera usar **global** (como en el caso de las globales que creemos nosotros) para acceder a ellas.
  + **$\_SERVER**: contiene información sobre el entorno del servidor web y de ejecución. Entre la información que nos ofrece esta variable, tenemos:

|  |  |
| --- | --- |
| **Principales valores de la variable $\_SERVER** | |
| **Valor** | **Contenido** |
| **$\_SERVER['PHP\_SELF']** | Guión que se está ejecutando actualmente. |
| **$\_SERVER['SERVER\_ADDR']** | Dirección IP del servidor web. |
| **$\_SERVER['SERVER\_NAME']** | Nombre del servidor web. |
| **$\_SERVER['DOCUMENT\_ROOT']** | Directorio raíz bajo el que se ejecuta el guión actual. |
| **$\_SERVER['REMOTE\_ADDR']** | Dirección IP desde la que el usuario está viendo la página. |
| **$\_SERVER['REQUEST\_METHOD']** | Método utilizado para acceder a la página ('GET', 'HEAD', 'POST' o 'PUT'). |

**Debes conocer**: [documentación PHP sobre la variable $\_SERVER](https://www.php.net/manual/es/reserved.variables.server.php).

* + **$\_GET, $\_POST, $\_COOKIE**: contienen las **variables que se han pasado al guion actual** utilizando respectivamente los métodos GET (parámetros en la URL), HTTP POST y Cookies HTTP.
  + **$\_REQUEST**: junta en uno solo el contenido de los tres arrays anteriores **$\_GET, $\_POST, $\_COOKIE**.
  + **$\_ENV**: contiene las variables que se puedan haber pasado a PHP desde el entorno en que se ejecuta.
  + **$\_FILES**: contiene los ficheros que se puedan haber subido al servidor utilizando el método POST.
  + **$\_SESSION**: contiene las variables de sesión disponibles para el guion actual.

**Para saber más**: conviene tener a mano la [información sobre estas variables superglobales](https://www.php.net/manual/es/language.variables.superglobals.php).



# 2.- Estructuras de control.

Para definir un flujo de un programa en PHP, hay sentencias de dos tipos:

* **Sentencias condicionales**: permite definir las condiciones bajo las que debe ejecutarse una sentencia o bloque de sentencias.
  + **Hay también una manera para decidir el flujo del programa**, mediante instrucciones goto, que te permite saltar directamente a otro punto del programa que indiques mediante una etiqueta:

<?php

$a = 1;

goto **salto**;

$a++; //esta sentencia no se ejecuta

**salto**:

echo $a; // el valor obtenido es 1

?>

* **Sentencias de bucle**: permite definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.

## 2.1.- Condicionales.

If, else, elseif

<?php

if ($a < $b)

print "a es menor que b";

elseif ($a > $b)

print "a es mayor que b";

else

print "a es igual a b";

?>

**Cuando**:

* La sentencia **if elseif** o **else** actúe sobre una única sentencia, no será necesario usar llaves. (como sucede en el ejemplo).
* Tendrás que **usar llaves para cuando quieres formar un conjunto de sentencias**, **siempre que quieras que el condicional actúe sobré más de una sentencia**.

Switch

Igual que en Java, C y otros lenguajes:

<?php

switch ($a) {

case 0:

print "a vale 0";

break;

case 1:

print "a vale 1";

break;

default:

print "a no vale 0 ni 1";

}

?>

**Ejercicio resuelto**: Haz una página web que muestre la fecha actual en castellano, incluyendo el día de la semana, con un formato similar al siguiente: "Miércoles, 13 de Abril de 2011". [Ver Anexo II de los apuntes](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%202.html). Se hace con switch, ya que, por defecto, PHP no trae ninguna función que te diga los días de la semana y los meses en español, solo en inglés.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media

## 2.2.- Bucles.

* While:

<?php

$a = 1;

while ($a < 8)

$a += 3;

print $a; // el valor obtenido es 10

?>

* Do/while:

<?php

$a = 5;

do

$a -= 3;

while ($a > 10);

print $a; // el bucle se ejecuta una sola vez, con lo que el valor obtenido es

?>

* For:

<?php

for ($a = 5; $a<10; $a+=3) {

print $a; // Se muestran los valores 5 y 8

print "<br />";

}

?>

Puedes usar las sentencias:

* **break**, para salir del bucle
* **continue**, para omitir la ejecución de las sentencias restantes y volver a la comprobación de la expresión respectivamente.

**Para saber más**: [videotutorial sobre la utilización de bucles en PHP](https://www.youtube.com/watch?v=VjqJwSy_BPQ).

Imagen que contiene Aplicación

Descripción generada automáticamente

# 3.- Funciones.

**Nos permiten asociar una etiqueta** (el nombre de la función) **con un bloque de código** a ejecutar. Además, al usar funciones estamos ayudando a estructurar mejor el código.

Como ya sabes, las funciones permiten crear variables locales que no serán visibles fuera del cuerpo de las mismas.

**El núcleo de PHP ya preincluye muchas funciones** que están a nuestra disposición y podemos utilizar. **Otras muchas se encuentran disponibles en** forma de **extensiones**, y se pueden incorporar al lenguaje cuando se necesitan. Con la distribución de PHP se incluyen varias extensiones.

En cualquier caso, **para poder usar las funciones** de una extensión, **tienes que asegurarte de activarla** mediante el uso de una directiva **extensión** en el fichero **php.ini**.

**Para obtener extensiones** para el lenguaje PHP puede **utilizar PECL**. PECL es un **repositorio** **de extensiones** para PHP. Junto con PHP se incluye un **comando pecl** que puedes utilizar para instalar extensiones de forma sencilla:

pecl install nombre\_extensión

**Para saber más**: [sobre PECL](https://www.php.net/manual/es/install.pecl.downloads.php).

## 3.1.- Inclusión de ficheros externos.

En ocasiones resulta útil **agrupar ciertos grupos de funciones** o bloques de código, y ponerlos **en un fichero aparte**. **Puedes hacer referencia a esos ficheros** para que PHP incluya su contenido como parte del programa actual.

**Para incorporar a tu programa contenido de un archivo externo**, tienes las siguientes opciones:

* **include**: evalúa el contenido del fichero que se indica y lo incluye como parte del fichero actual, en el mismo punto en que se realiza la llamada.
  + La **ruta** puede ser absoluta o **relativa** (lo más común). En este último caso, se toma como base la ruta especificada en la directiva include\_path del fichero php.ini. Si no se encuentra en esa ubicación, se buscará también en el directorio del guión actual, y en el directorio de ejecución.
  + Si no se encuentra, da un aviso y continua con la ejecución del guión.
* **include\_once**: funciona exactamente igual que include, pero solo incluye aquellos ficheros que aún no se hayan incluido.
* **requiere**: como include, pero si no encuentra el fichero, se detiene la ejecución del guión.
* **require\_once**: es la combinación de las dos anteriores. Asegura la inclusión del fichero indicado solo una vez, y genera un error si no se puede llevar a cabo.

**Buenas prácticas**: Muchos programadores utilizan la doble extensión inc.php para aquellos ficheros en lenguaje PHP cuyo destino es ser incluidos dentro de otros, y nunca han de ejecutarse por sí mismos.

**Ejercicio resuelto**: ejemplo de utilización de include.

**definiciones.php**

<?php

$modulo = 'DWES';

$ciclo = 'DAW';

?>

**programa.php**

<?php

print "Módulo $modulo del ciclo $ciclo<br />"; //Solo muestra "Modulo del ciclo"

**include** 'definiciones.php';

print " Módulo $modulo del ciclo $ciclo<br />"; // muestra "Modulo DWES del ciclo DAW"

?>

Texto

Descripción generada automáticamente con confianza baja

## 3.2.- Ejecución y creación de funciones.

* **Para ejecutar funciones, basta con poner su nombre y unos paréntesis**.

<?php

phpinfo();

?>

* **Para crear funciones**: usar la palabra function.

<?php

function precio\_con\_iva() {

global $precio;

$precio\_iva = $precio \* 1.18;

print "El precio con IVA es ".$precio\_iva;

}

$precio = 10;

precio\_con\_iva();

?>

* + En PHP **no es necesario definir una función antes de utilizarla, excepto cuando está condicionalmente definida**.

<?php

$iva = true;

$precio = 10;

precio\_con\_iva(); // Da error, pues aquí aún no está definida la función

if ($iva) {

function precio\_con\_iva() {

global $precio;

$precio\_iva = $precio \* 1.18;

print "El precio con IVA es ".$precio\_iva;

}

}

precio\_con\_iva(); // Aquí ya no da error

?>

## 3.3.- Argumentos.

* Usar variables globales no es una buena práctica. Por ello, se utilizan los **argumentos en las funciones**.
* Cuando una sentencia termina en **return**, termina su procesamiento y devuelve el valor que se indica. Si no tiene esta sentencia return, sencillamente, no devolverá nada (null).
* Por defecto, los argumentos se pasan por valor.

<?php

function precio\_con\_iva($precio) {

return $precio \* 1.18;

}

$precio = 10;

$precio\_iva = precio\_con\_iva($precio);

print "El precio con IVA es ".$precio\_iva

?>

* La lista de variables se separa por comas.
* No se indica el tipo de cada argumento. Por ello es necesario adoptar buenos nombres de argumentos para saber en todo momento que dato necesitamos meter en los argumentos de la función.
* **Puede haber valores por defecto definidos para varios argumentos**. Pero estos siempre que estar después de aquellos que no tienen ninguno definido.

<?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

?>

* **Se pueden pasar argumentos por referencia** (&). Esto es, cualquier cambio que se haga dentro de la función a los valores de los argumentos se reflejará fuera de la función.

<?php

function precio\_con\_iva(&$precio, $iva=0.18) {

$precio \*= (1 + $iva);

}

$precio = 10;

print "El precio con IVA es ".$precio

?>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Ejercicio resuelto**: Anteriormente hiciste un ejercicio que mostraba la fecha actual en castellano. Con el mismo objetivo (puedes utilizar el código ya hecho), crea una función que devuelva una cadena de texto con la fecha en castellano, e introdúcela en un fichero externo. Después crea una página en PHP que incluya ese fichero y utilice la función para mostrar en pantalla la fecha obtenida. [Ver Anexo III](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%203.pdf).

# 4.- Tipos de datos compuestos.

Un tipo de dato compuesto te permite almacenar más de un valor. Hay dos tipos:

* **Objeto**.
* **Array**.

Arrays

Permite almacenar varios valores.

* Cada miembro del **array** se almacena en una posición a la que se hace referencia utilizando un valor clave.
* Las claves pueden ser numéricas o asociativas.
* Si no se especifica una clave, se asignará por orden, con claves numéricas, así: 0, 1,….
* **Los valores pueden ser de distinto tipo**:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Correo electrónico

Descripción generada automáticamente

* [**print\_r**](https://www.php.net/manual/es/function.print-r.php): Función que nos muestra por pantalla todo el contenido del array que le pasamos. Muy útil para tareas de depuración.
  + También está var\_dump.
* Además de arrays unidimensionales, PHP **también soporta arrays de varias dimensiones** almacenando otro array en cada uno de los elementos de un array.

// array bidimensional

$ciclos = array(

"DAW" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarr

"DAM" => array ("PR" => "Programación", "BD" => "Bases de datos", ..., "PMDM" => "Progra

);

* Para hacer referencia a los elementos almacenados en un **array multidimensional**, hay que especificar las claves para cada una de las dimensiones:

$ciclos ["DAW"] ["DWES"]

* No es necesario especificar el tamaño del array antes de crearlo, ni especificar su tipo.
* **Formas de crear un array**:
  + **Con la función range**: podemos crear un array de x posiciones. Esta opción no nos permite combinarse entre sí para crear arrays de más de una dimensión con un nº de posiciones determinado.

$ejeX = range( 1, 10 );

$ejeY = range( 1, 10 );

* + **Con la función array**(valores), tal como ya hemos visto en ejemplos anteriores.
    - También se pueden crear arrays con esta función sin indicar los valores de las claves (de esta forma, por defecto, crea un array numérico con base 0)
    - También se pueden crear arrays vacíos.

$a = array(); // array vacío

$modulos = array("Programación", "Bases de datos", ..., "Desarrollo web en entorno servidor")

* + **Especificando claves y valores directamente**:

// array numérico

$modulos1 [0] = "Programación";

$modulos1 [1] = "Bases de datos";

...

$modulos1 [9] = "Desarrollo web en entorno servidor";

// array asociativo

$modulos2 ["PR"] = "Programación";

$modulos2 ["BD"] = "Bases de datos";

...

$modulos2 ["DWES"] = "Desarrollo web en entorno servidor";

* + **Especificando valores solamente**. Si omites la clave, se irá llenando a partir de la última clave numérica existente, o de la posición 0 si no existe ninguna.

$modulos1 [ ] = "Programación";

$modulos1 [ ] = "Bases de datos";

...

$modulos1 [ ] = "Desarrollo web en entorno servidor";

* **Las cadenas de texto o string se pueden tratar como arrays** en los que se almacena un carácter en cada posición, siendo el 0 el índice correspondiente al primero.

// cadena de texto

$modulo = "Desarrollo web en entorno servidor";

// $modulo[3] == "a";

## 4.1.- Recorrer arrays (I): con bucles.

Se puede utillizar el bucle especial **foreach**, de los siguientes modos:

* **Recorriendo solo los elementos**:

$modulos = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor");

foreach ($modulos as $modulo) {

print "Módulo: ".$modulo. "<br />"

}

* **Recorriendo elementos y sus claves**:

$modulos = arrays("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarrollo web en entorno servidor");

foreach ($modulos as $codigo => $modulo) {

print "El código del módulo ".$modulo." es ".$codigo."<br />"

}

**Ejercicio resuelto**: Haz una página PHP que utilice foreach para mostrar todos los valores del array $\_SERVER en una tabla con dos columnas. La primera columna debe contener el nombre de la variable, y la segunda su valor. [Ver Anexo IV](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%204.pdf).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### 4.1.1.- Recorrer arrays (II): con punteros.

**Otra forma de recorrer los arrays es con el puntero** **interno** que mantienen. Utilizando funciones específicas, podemos avanzar, retroceder o inicializar el puntero, así como recuperar los valores del elemento (o de la pareja clave / elemento) al que apunta el puntero en cada momento.

|  |  |
| --- | --- |
| **Funciones para recorrer arrays** | |
| **Función** | **Resultado** |
| **reset** | Sitúa el puntero interno al comienzo del array. Y devuelve el valor del elemento en dicha posición. |
| **next** | Avanza el puntero interno una posición. Y devuelve el valor del elemento en dicha posición. |
| **prev** | Mueve el puntero interno una posición hacia atrás. Y devuelve el valor del elemento en dicha posición. |
| **end** | Sitúa el puntero interno al final del array. Y devuelve el valor del elemento en dicha posición. |
| **current** | Devuelve el elemento de la posición actual. |
| **key** | Devuelve la clave de la posición actual. |
| **each** | Devuelve un array con 4 elementos. Además, **avanza el puntero interno una posición**. Los elementos del array son:   * **0 y ‘key’**: almacenan el valor de la clave en la posición actual del puntero interno. * **1 y ‘value’**: devuelven el valor almacenado. |

* **Si al mover el puntero** (con reset, next, prev o end) **te sales de los límites del array** (por ejemplo, estás ya en el último elemento y haces un **next**), **devuelve false**.
* Sin embargo, al comprobar este valor devuelto, no serás capaz de distinguir si te has salido de los límites del array, o si estás en una posición valida del array que contiene el valor “false”.
  + Para ello, la **función key** **devuelve null si el puntero interno está fuera de los límites del array**.
  + La **función each** también sirve para este cometido, **devolverá false si el puntero interno del array se ha pasado de los límites del array**.

while ($modulo = each($modulos)) {

print "El código del módulo ".$modulo[1]." es ".$modulo[0]."<br />"

}

**Ejercicio resuelto**: Haz una página PHP que utilice estas funciones para crear una tabla como la del ejercicio anterior. [Ver Anexo V](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%205.pdf).

## 4.2.- Funciones relacionadas con los tipos de datos compuestos (arrays).

* **Añadir elementos**: no definiendo el índice, o utilizando un índice nuevo.

$a = array(); // array vacío

$modulos = array("Programación", "Bases de datos", ..., "Desarrollo web en entorno servidor")

* **Eliminar elementos**: con la función **unset**.
  + **En el caso de los arrays numéricos**, eliminar un elemento significa que **las claves del mismo ya no estarán consecutivas**.

unset ($modulos [0]);

// El primer elemento pasa a ser $modulos [1] == "Bases de datos";

* **Función array\_values**: recibe un array por parámetro, y devuelve el mismo array, pero con los valores ordenados consecutivamente, con índices de base 0.
* **Función is\_array**: permite comprobar si la variable es un array.
* **Función count**: devuelve la cantidad de elementos que contiene un array.
* **Función in\_array(elementoABuscar, arrayEnElQueBuscar)**: permite descubrir si un elemento está en el array o no. Devuelve true si encontró el elemento, false si no.

$modulos = array("Programación", "Bases de datos", "Desarrollo web en entorno servidor");

$modulo = "Bases de datos";

if (in\_array($modulo, $modulos)) printf "Existe el módulo de nombre ".$modulo;

* **Función array\_search**: recibe los mismos parámetros que in\_array. **Devuelve la clave correspondiente al elemento**, false si no lo encuentra.
* **Función array\_key\_exists**: busca una clave en un array.

**Para saber más**: hay muchas funciones para gestionar arrays. [Consultar lista completa en el manual de PHP](https://www.php.net/manual/es/ref.array.php).

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

# 5.- Formularios web.

La forma natural para hacer llegar a la aplicación web los datos del usuario desde un navegador, es utilizar **formularios HTML**.

* **Etiquetas de cierre y apertura**: <FORM> y </FORM>
  + **Atributo action de <form>**: indica la página a la que se le enviarán los datos del formulario. En nuestro caso, se tratará de un guión PHP.
  + **Atributo method**: especifica el método usado para enviar esa información. Dos tipos de métodos:
    - **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.
* **Elementos de un formulario**:
  + <INPUT>
    - Un input de tipo “submit” es un botón que se encarga de enviar los datos del formulario.
  + <SELECT>
  + <TEXTAREA>
  + <BUTTON>

**Para saber más**: debes conocer el lenguaje HTML para trabajar con formularios en PHP. [Curso sobre HTML de aulaClic](https://www.aulaclic.es/html/t_8_1.htm).

Si en un formulario web tienes que enviar alguna variable en la que sea posible almacenar más de un valor (por ejemplo, cuando usamos casillas de verificación), tendrás que ponerle corchetes al nombre de la variable para indicar que se trata de un array.

<input type="checkbox" name="ciclos[]" value="DWES" /> Desarrollo web en entorno

servidor<br />

<input type="checkbox" name="ciclos[]" value="DWEC" /> Desarrollo web en entorno

servidor<br />

**Ejercicio resuelto**: Crea un formulario HTML para introducir el nombre del alumno y el ciclo que cursa, a escoger entre “Desarrollo Web en Entorno Servidor” y “Desarrollo Web en Entorno Cliente”. Envía el resultado a la página “procesa.php”, que será la encargada de procesar los datos. [Ver Anexo VI](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%206.pdf).

## 5.1.- Procesamiento de la información devuelta por un formulario web.

En el ejemplo del Anexo VI, se creó un formulario en una página HTML que recogía datos del usuario y los enviaba a una página PHP para que los procesara.

* **Si simplemente, los quisieras mostrar por pantalla**, este podría ser el código de “procesa.php”. [Ver Anexo VII](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%207.pdf).
* **Ejemplo usando el método GET**: solo haría falta cambiar POST por GET. [Ver Anexo VIII](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%208.pdf).
* **Ejemplo usando $\_REQUEST**: [ver Anexo IX](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%209.pdf).

Siempre que sea posible, **es mejor validar los datos primero en entorno cliente (JavaScript)**. Si hubiera que validar datos en el servidor (por ejemplo, consultar que los datos de un usuario no existan ya en la BB.DD antes de introducirlos), sería necesario hacerlo con PHP.

* **Se puede usar la misma página que muestra el formulario como destino de los datos**. Si tras comprobar los datos, estos son correctos, se reenvía a otra página. Si son incorrectos, se rellenan los datos correctos en el formulario y se indican cuáles son incorrectos y por qué. Procedimiento:
  + **En la parte HTML**: en vez de escribir el nombre del fichero, como este puede cambiar con el tiempo, simplemente, podemos usar la variable **$\_SERVER['PHP\_SELF']** para obtener el nombre del fichero al que nos referiremos en el atributo action.
  + **Comprobar si la página recibe datos** (mediante la función isset).
    - Mostrarlos en caso de recibirlos y no generar el formulario.
    - Mostrar el formulario si no recibe datos.

([Ver Anexo X](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%2010.pdf))

Fíjate en la forma de englobar el formulario dentro de una sentencia else para que sólo se genere si no se reciben datos en la página.

## 5.2.- Generación de formularios web en PHP.

Podemos revisar los datos que se obtienen antes de mostrarlos. En nuestro ejemplo, podríamos comprobar que el nombre no esté vacío, y que se haya seleccionado como mínimo uno de los módulos.

Además, **en el caso de que falte algún dato**, **deberás generar el formulario rellenando aquellos datos que el usuario haya introducido correctamente**.

**Para validar los datos**, podemos hacer:

if (!empty($\_POST['modulos']) && !empty($\_POST['nombre'])) {

// Aquí se incluye el código a ejecutar cuando los datos son correctos

}

else {

// Aquí generamos el formulario, indicando los datos incorrectos

// y rellenando los valores correctamente introducidos

}

**Para que el usuario no pierda**, después de enviar el formulario, **los datos correctamente introducidos**, utiliza el atributo **value** en las entradas de texto:

Nombre del alumno:

<input type="text" name="nombre" value="<?php echo $\_POST['nombre'];?>" />

Y el atributo **checked** en las casillas de verificación: (ejemplo para la casilla de DWES):

<input type="checkbox" name="modulos[]" value="DWES"

<?php

if(in\_array("DWES",$\_POST['modulos']))

echo 'checked="checked"';

?>

/>

**Para indicar al usuario los datos que no ha rellenado** (o que ha rellenado de forma incorrecta): deberás comprobar si es la primera vez que se visualiza el formulario, o si ya se ha enviado. Se puede hacer por ejemplo de la siguiente forma:

Nombre del alumno:

<input type="text" name="nombre" value="<?php echo $\_POST['nombre'];?>" />

<?php

if (isset($\_POST['enviar']) && empty($\_POST['nombre']))

echo "<span style='{color:red}'> <-- Debe introducir un nombre!!</span>"

?><br />

**Para ver el ejemplo completo, con todo**: [Ver Anexo XI](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%2011.pdf).

Una **forma de enviar información de una página PHP a otra**, es incluyéndola en **campos ocultos** **dentro de un formulario**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Ejercicio resuelto**: **Modifica el ejercicio que mostraba la fecha en castellano, para que obtenga lo mismo a partir de un día, mes y año introducido por el usuario**. Antes de mostrar la fecha, se debe comprobar que es correcta. Utilizar la misma página PHP para el formulario de introducción de datos y para mostrar la fecha obtenida en castellano.

Consultar las funciones [checkdate](https://www.php.net/manual/en/function.checkdate.php) y [mktime](https://www.php.net/manual/es/function.mktime.php) en el manual de PHP. [Ver Anexo XII](../Apuntes/Anexos%20Ejercicios%20resueltos/Anexo%2012.pdf).