Características del lenguaje PHP.

Caso práctico

El nuevo proyecto va a ponerse en marcha. En **BK Programación** las tres personas asignadas comienzan los preparativos. **Juan**, **el jefe de proyecto**, está elaborando un calendario para intentar definir las distintas fases del desarrollo. **María**, en el tiempo que le puede dedicar, se ha puesto a refrescar sus conocimientos de **programación en PHP**. **Carlos** es el que más trabajo tiene por delante, sabe que si quiere aportar algo, debe aprender a programar aplicaciones web, y pronto.

Carlos le pide ayuda a Juan, que le orienta sobre los conceptos fundamentales del lenguaje y le ofrece su ayuda en todo lo que le sea posible. Sabe que es importante adquirir unos conocimientos sólidos antes de comenzar el desarrollo, para no cometer errores al principio que después sea complicado solucionar.

Con la ayuda de María, ponen en funcionamiento un **servidor de aplicaciones** dentro de la empresa. De momento lo utilizarán para ir haciendo pruebas, pero dentro de poco será la plataforma sobre la que programarán la nueva aplicación.

Una vez vistas distintas tecnologías de programación web, esta unidad te introduce los conceptos fundamentales de la programación en lenguaje PHP. Aprenderás a crear programas simples, utilizando tipos de datos sencillos y arrays, estructurando el código en distintos ficheros y utilizando funciones. Se crearán programas interactivos utilizando formularios web para la comunicación con el usuario.

1.- Elementos del lenguaje PHP.

Caso práctico

Carlos comienza su aprendizaje del nuevo lenguaje. Conforme va avanzando, comprueba que muchos de los conceptos que aprende son similares a lo que ya conocía. Otros, sin embargo, son muy distintos y los tiene que practicar para entender bien su manejo.

Aunque es consciente que al principio va a cometer fallos, se pone como meta utilizar lo que va aprendiendo para hacer pequeños programas que pueda volver a usar en el futuro. Y si consigue hacer algo que pueda tener alguna utilidad para la nueva aplicación, ¡mejor!

En la unidad anterior, aprendiste a preparar un entorno para programar en PHP. Además también viste algunos de los elementos que se usan en el lenguaje, como las variables y tipos de datos, comentarios, operadores y expresiones.

También sabes ya cómo se integran las etiquetas HTML con el código del lenguaje, utilizando los delimitadores "**<?php** y **?>**". Ya vimos en la unidad anterior que podíamos usar "**<? ?>**" si teníamos la directiva **short_open_tags = On** en el archivo ***php.ini***, pero no se recomendaba.

En esta unidad aprenderás a utilizar otros elementos del lenguaje que te permitan crear programas completos en PHP. Los programas escritos en PHP, además de encontrarse estructurados normalmente en varias páginas (ya veremos más adelante cómo se pueden comunicar datos de unas páginas a otras), suelen incluir en una misma página varios bloques de código. Cada bloque de código debe ir entre 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, en las siguientes líneas tenemos dos bloques de código en PHP.

```
<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.

En esta unidad empezarás a crear tus propios programas en PHP. Para ello vas a usar el IDE VSC, que instalaste anteriormente, aunque puedes utilizar cualquiera de los mencionados en la unidad anterior. Deberías organizar tus programas en proyectos, para ello lo ideal es:

- Si utilizas **Linux,** y seguisteis los pasos de la unidad anterior, crea una carpeta en **public_html,** por ejemplo **tema2,** y ábrela con el VSC. Para acceder a tú proyecto desde el navegador tendrás que indicar la **url:** ***http://localhost/~tuUsuario/tema2/***
- Si utilizas **Windows** y **Xampp**, crea una carpeta en ***c:\xampp\htdocs,*** por ejemplo **tema2,** y ábrela con el VSC. Para acceder a tú proyecto tendrás que indicar la **:URL** ***http://localhost/tema2/***

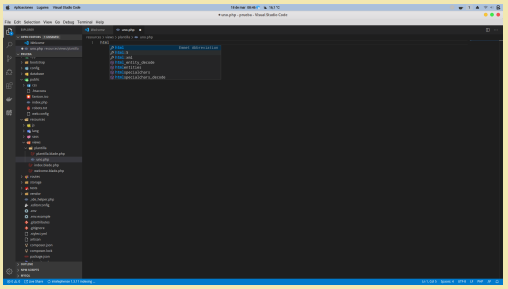
Recomendación

Para los archivos que van a ser solamente de código PHP, es decir no vamos a incluir marcas HTML es recomendable no cerrar el guion php, por ejemplo veamos el contenido de archivo ***uno.php*** :

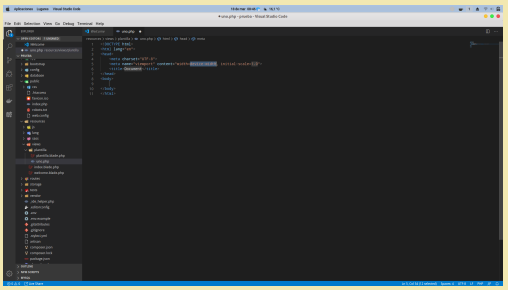
```php
<?php    //archivo uno.php
    echo "este archivo solo contiene código php";
    for($i=1; $i<10: $i++){
        echo $i. '<br>';
    }
    //no cerraremos el script con ?>
```

Recomendación

VSC nos facilita mucho el trabajo automatizando tareas, fíjate que solo escribiendo **html** al inicio de una página nueva (con extensión **htm** o **php**) nos crea una plantilla completa de una página HTML.



Visual Studio Code (Elaboración propia.)



Visual Studio Code (Elaboración propia.)

1.1.- Generación de código HTML.

Existen varias formas incluir contenido en la página web a partir del resultado de la ejecución de código PHP. La forma más sencilla es usando **echo**, que no devuelve nada (**void**), y genera como salida el texto de los parámetros que recibe.

Anónimo (Dominio público)

```
void echo (string $arg1, ...);
```

Otra posibilidad es **print**, que funciona de forma similar. La diferencia más importante entre **print** y **echo**, es que **print** sólo puede recibir un parámetro y devuelve siempre 1.

```
int print (string $arg);
```

Tanto **print** como **echo** no son realmente funciones, por lo que no es obligatorio que pongas paréntesis cuando las utilices. Por ejemplo, el código del siguiente documento puede hacerse igualmente utilizando **echo**.

Utilización de la función **print** en PHP

printf (**print** con formato) es otra opción para generar una salida desde PHP. Puede recibir varios parámetros, el primero de los cuales es siempre una cadena de texto que indica el formato que se ha de aplicar. Esa cadena 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 en que figuran en la lista de parámetros. Por ejemplo:

```
<?php
$ciclo="DAW";
$modulo="DWES";
print "<p>";
printf("%s es un módulo de %d curso de %s", $modulo, 2, $ciclo);
print "</p>";
?>
```

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

- **signo** (opcional). Indica si se pone signo a los número negativos (por defecto) o también a los positivos (se indica con un signo +).
- **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).
- **alineación** (opcional). Indica que tipo de alineación se usará para generar la salida: justificación derecha (por defecto) o izquierda (se indica con el carácter -).

- **ancho** (opcional). Indica el mínimo número de caracteres de salida para un parámetro dado.
- **precisión** (opcional). Indica el número de dígitos decimales que se mostrarán para un número real. Se escribe como un dígito precedido por un punto.
- **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	el argumento es tratado como un entero y presentado como un número binario.
c	el argumento es tratado como un entero, y presentado como el carácter con dicho valor ASCII.
d	el argumento es tratado como un entero y presentado como un número decimal.
u	el argumento es tratado como un entero y presentado como un número decimal sin signo.
o	el argumento es tratado como un entero y presentado como un número octal.
x	el argumento es tratado como un entero y presentado como un número hexadecimal (con minúsculas).
X	el argumento es tratado como un entero y presentado como un número hexadecimal (con mayúsculas).
f	el argumento es tratado como un doble y presentado como un número de coma flotante (teniendo en cuenta la localidad).
F	el argumento es tratado como un doble y presentado como un número de coma flotante (sin tener en cuenta la localidad).
e	el argumento es presentado en notación científica, utilizando la e minúscula (por ejemplo, 1.2e+3).
E	el argumento es presentado en notación científica, utilizando la e mayúscula (por ejemplo, 1.2E+3).
g	se usa la forma más corta entre %f y %e.
G	se usa la forma más corta entre %f y %E.
s	el argumento es tratado como una cadena y es presentado como tal.

Especificador	Significado
%	se muestra el carácter %. No necesita argumento..

Por ejemplo, al ejecutar la línea siguiente, en la salida el número π se obtiene con signo y sólo con dos decimales.

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

Existe una función similar a **printf** pero en vez de generar una salida con la cadena obtenida, permite guardarla en una variable: **sprintf**.

```
$txt_pi = sprintf("El número PI vale %+.2f", 3.1416);
```

Autoevaluación

Tenemos una variable real, y queremos mostrarla utilizando un número fijo de decimales, por ejemplo 3. ¿Podemos hacerlo sin utilizar la función printf?

- No.
- Sí.

Incorrecto. ¿Podemos obtener una cadena con el texto exacto que queremos mostrar?

Efectivamente. A partir de la variable real, y utilizando la función sprintf, podemos obtener una cadena con el texto exacto que queremos mostrar, haciendo **sprintf("%.3f", $a)**.

Solución

1. Incorrecto

2. Opción correcta

1.2.- Cadenas de texto.

En PHP las cadenas de texto pueden usar tanto comillas simples como comillas dobles. Sin embargo hay una diferencia importante entre usar unas u otras. Cuando se pone una variable dentro de unas comillas dobles, se procesa y se sustituye por su valor. Así, el ejemplo anterior sobre el uso de **print** también podía haberse puesto de la siguiente forma:

xresch (Pixabay License)

```php
<?php
    $modulo="DWES";
    print "<p>Módulo: $modulo</p>"
?>
```

La variable **$modulo** se reconoce dentro de las comillas dobles, y se sustituye por el valor "DWES" antes de generar la salida. Si esto mismo lo hubieras hecho utilizando comillas simples, no se realizaría sustitución alguna.

Para que PHP distinga correctamente el texto que forma la cadena del nombre de la variable, a veces es necesario rodearla entre llaves. Sobre todo cuando trabajemos con POO.

```php
print "<p>Módulo: ${modulo}</p>"
```

Cuando se usan comillas simples, sólo se realizan dos sustituciones dentro de la cadena:

1.- Cuando se encuentra la secuencia de caracteres **\'**, se muestra en la salida una comilla simple.
2.- Cuando se encuentra la secuencia ****, se muestra en la salida una barra invertida.

Estas secuencias se conocen como **secuencias de escape**. En las cadenas que usan comillas dobles, además de la secuencia \\, se pueden usar algunas más, pero no la secuencia **\'**. En esta tabla puedes ver las secuencias de escape que se pueden utilizar, y cuál es su resultado.

Secuencias de escape.

Secuencia	Resultado
\\	se muestra una barra invertida.

Secuencia	Resultado
\'	se muestra una comilla simple.
\"	se muestra una comilla doble.
\n	se muestra un avance de línea (LF o 0x0A (10) en ASCII).
\r	se muestra un retorno de carro (CR o 0x0D (13) en ASCII).
\t	se muestra un tabulador horizontal (HT o 0x09 (9) en ASCII).
\v	se muestra un tabulador vertical (VT o 0x0B (11) en ASCII).
\f	se muestra un avance de página (FF o 0x0C (12) en ASCII).
\$	se muestra un signo del dólar.

En PHP tienes dos operadores exclusivos para trabajar con cadenas de texto. Con el operador de concatenación punto (.) puedes unir las dos cadenas de texto que le pases como operandos. El operador de asignación y concatenación (.=) concatena al argumento del lado izquierdo la cadena del lado derecho.

```php
<?php
    $a = "Módulo ";
    $b = $a . "DWES"; // ahora $b contiene "Módulo DWES"
    $a .= "DWES"; // ahora $a también contiene "Módulo DWES"
?>
```

En PHP tienes otra alternativa para crear cadenas: la sintaxis **heredoc**. Consiste en poner el operador "<<<" seguido de un identificador de tu elección, y a continuación y empezando en la línea siguiente 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, ni siquiera espacios o sangría, salvo quizás un punto y coma después del identificador.

```php
<?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, debes poner el identificador de apertura entre comillas simples.

```
$a = <<<'MICADENA'

...

MICADENA;
```

1.3.- Funciones relacionadas con los tipos de datos.

WikiMedia Commons (Dominio público)

En PHP existen funciones específicas para comprobar y establecer el tipo de datos de una variable, **gettype** obtiene el tipo de la variable que se le pasa como parámetro y devuelve una cadena de texto, que puede ser **array**, **boolean**, **double**, **integer**, **object**, **string**, **null**, **resource** o **unknown type**.

También podemos comprobar si la variable es de un tipo concreto utilizando una de las siguientes funciones: **is_array()**, **is_bool()**, **is_float()**, **is_integer()**, **is_null()**, **is_numeric()**, **is_object()**, **is_resource()**, **is_scalar()** e **is_string()**. Devuelven true si la variable es del tipo indicado.

Análogamente, para establecer el tipo de una variable utilizamos la función **settype** pasándole como parámetros la variable a convertir, y una de las siguientes cadenas: **boolean**, **integer**, **float**, **string**, **array**, **object** o **null**. La función **settype** devuelve true si la conversión se realizó correctamente, o false en caso contrario.

```php
<?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);
?>
```

El resultado del código anterior es:

```
$a vale 3.1416 y es de tipo string
$b vale 3.1416 y es de tipo double
```

Si lo único que te interesa es saber si una variable está definida y no es **null**, puedes usar la función **isset**. La función **unset** destruye la variable o variables que se le pasa como parámetro.

```php
<?php
    $a = "3.1416";
    if (isset($a)) // la variable $a está definida
    unset($a); //ahora ya no está definida
?>
```

Es importante no confundir el que una variable esté definida o valga **null**, con que se considere como vacía debido al valor que contenga. Esto último es lo que nos indica la función **empty**. Puedes consultar información al respecto haciendo click en el enlace siguiente: Documentación empty.

Existe también en PHP una función **define()**, con la que puedes definir constantes, esto es, identificadores a los que se les asigna un valor que no cambia durante la ejecución del programa.

```
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.

```
<?php
    define ("PI", 3.1416, true);
    print "El valor de PI es ".pi; //El identificador se reconoce tanto por PI como por pi
?>
```

Sólo se permiten los siguientes tipos de valores para las constantes: **integer**, **float**, **string**, **boolean** y **null**.

Autoevaluación

¿Qué se muestra en pantalla al ejecutar el siguiente código?

```
$a = "-3.1416";

printf("La variable \'\$a\' vale %+.2f", $a);
```

- ○ La variable '$a' vale -3.14.
- ○ La variable '$a' vale +3.14.
- ○ La variable \'$a\' vale -3.14.
- ○ La variable \'$a\' vale +3.14.

No es correcto. Fíjate bien, ¿cuáles son las secuencias de escape que se pueden usar entre comillas dobles?

No es así. Fíjate bien, ¿cuáles son las secuencias de escape que se pueden usar entre comillas dobles? ¿Qué significa el signo + en un especificador de conversión?

Efectivamente. Dentro de comillas dobles no se sustituye la secuencia de escape \'. Además, el signo + en un especificador de conversión indica que se visualice el signo del número aunque sea positivo, pero el signo nunca se cambia y siempre se muestra si el número es negativo.

Incorrecto. Fíjate bien, ¿qué significa el signo + en un especificador de conversión?

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

1.3.1.- Funciones de Fecha I.

dannya (Dominio público)

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, dentro de las funciones de PHP tienes a tu disposición unas cuantas para trabajar con ese tipo de datos.

Una de las más útiles es quizás la función **date**, que te permite obtener una cadena de texto a partir de una fecha y hora, con el formato que tú elijas. La función recibe dos parámetros, la descripción del formato y el número entero que identifica la fecha, y devuelve una cadena de texto formateada.

```
string date (string $formato [, int $fechahora]);
```

El formato lo debes componer utilizando como base una serie de caracteres de los que figuran en la siguiente tabla.

Función date(): caracteres de formato para fechas y horas.

Carácter	Resultado
d	día del mes con dos dígitos.
j	día del mes con uno o dos dígitos (sin ceros iniciales).
z	día del año, comenzando por el cero (0 = 1 de enero).
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).
W	número de la semana del año.
m	número del mes con dos dígitos.
n	número del mes con uno o dos dígitos (sin ceros iniciales).
t	número de días que tiene el mes.

Carácter	Resultado
F	texto del día del mes, en inglés (January, ..., December).
M	texto del día del mes, solo tres letras, en inglés (Jan, ..., Dec).
Y	número del año.
y	dos últimos dígitos del número del año.
L	1 si el año es bisiesto, 0 si no lo es.
h	formato de 12 horas, siempre con dos dígitos.
H	formato de 24 horas, siempre con dos dígitos.
g	formato de 12 horas, con uno o dos dígitos (sin ceros iniciales).
G	formato de 24 horas, con uno o dos dígitos (sin ceros iniciales).
i	minutos, siempre con dos dígitos.
s	segundos, siempre con dos dígitos.
u	microsegundos.
a	am o pm, en minúsculas.
A	AM o PM, en mayúsculas.
r	fecha entera con formato RFC 2822.

Además, el segundo parámetro es opcional. Si no se indica, se utilizará la hora actual para crear la cadena de texto.

Para que el sistema pueda darte información sobre tu fecha y hora, debes indicarle tu zona horaria. Puedes hacerlo con la función **date_default_timezone_set**. Para establecer la zona horaria en España peninsular debes indicar:

```
date_default_timezone_set('Europe/Madrid');
```

De igual manera para que los días de la semana o el nombre de los meses aparezca en español deberás indicar los "locales" de la siguiente forma:

```
setlocale(LC_ALL, 'es_ES.UTF-8');
```

Debes tener en cuenta que la función **date()** no lee los "locales", para hacer uso de los

nombres en español (lunes, Enero...) deberás usar la función `strftime()`

Para saber más

En la documentación siguiente puedes consultar las distintas zonas horarias que se pueden usar.

Distintas zonas horarias soportadas en PHP.

De la misma forma puedes consultar las distintas configuraciones de localismo.

Información Localismos en PHP.

Documentación de la función `strftime`. I

nformación strftime() .

Si utilizas alguna función de fecha y hora sin haber establecido previamente tu zona horaria, lo más probable es que recibas un error o mensaje de advertencia de PHP indicándolo.

Otras funciones como `getdate()` devuelven un array con información sobre la fecha y hora actual.

Debes conocer

En la documentación de PHP puedes consultar todas las funciones para gestionar fechas y horas:

Fechas y horas.

1.3.2.- Funciones de Fecha II.

Desde la versión 5.2.2, PHP ofrece mecanismos modernos y potentes para crear fechas y convertir entre formatos, usando la clase **DateTime** de PHP. Veremos cómo usarla para crear y convertir fechas, crear cadenas con valores de fecha en cualquier formato o convertirlas, cosa que nos vendrá muy bien cuando trabajemos con bases de datos como MySql, pues el formato de fecha de la misma suele ser: '***Y-m-d***' y no '***d-m-Y***' que solemos usar, veamos algunos ejemplos de su uso.

dannya (Dominio público)

- **Ejemplo 1.-** Crear una fecha a partir de cualquier cadena.

```php
$fechaMySql="2020-12-31";
$objetoDateTime=date_create_from_format('Y-m-d', $fechaMySql);
var_dump($objetoDateTime);
//o más rápido
$fecha1=new DateTime("2020-12-31");
echo "<br>";
var_dump($fecha1);
```

- **Ejemplo 2.-** Pasar la fecha al formato que queramos y sacar el "*timestamp*" (marca de tiempo a una fecha).

```php
//Ejemplo 2.- Pasar la fecha al formato que queramos$miFecha=new DateTime();
$fecha=date_format($miFecha, 'd/m/Y');
echo "<br>";
var_dump($fecha);
//Sacar la marca de tiempo a un objeto de tipo dateTime
$ahora=new DateTime();echo "<br>Timestamp: " . date_timestamp_get($ahora);
```

- **Ejemplo 3.-** Las posibilidades son múltiples.

```php
//fecha actual
        $ahora=new DateTime();
        echo "<br>";
        var_dump($ahora);
//Fecha de ayer
        $ayer=new dateTime("yesterday");
        echo "<br>";
        var_dump($ayer);
//Fecha del último lunes
        $ultimoLunes=new DateTime("Last Monday");
        echo "<br>";
```

```
var_dump($ultimoLunes);
```

Para saber más

Todos los formatos relativos de fecha (los que usan nombres o frases en inglés como las vistas en el ejemplo 3) las puedes consultar en este enlace.

Formatos relativos a fechas en PHP

Recomendación

Si vas a trabajar con fechas y deseas que el formato y los nombres de dias y meses aparezcan en castellano no es mala idea poner al principio de tu archivo lo siguiente:

```
<?php
    setlocale(LC_ALL, 'es_ES.UTF-8');
    date_default_timezone_set('Europe/Madrid');
?>
```

Ejercicio Resuelto

Carlos se ha propuesto que en todas las páginas de un sitio Web, a efectos de impresión y demás, aparezca la fecha y la hora actual en castellano. Piensa que con ello mejorará mucho la presentación de las mismas.

Crea el código necesario para que cada vez que abramos o actualicemos una página Web nos aparezca la fecha y la hora actual en castellano.

Mostrar retroalimentación

```php
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejercicio 1</title>
</head>
<body style="background-color: antiquewhite">
<?php
    setlocale(LC_ALL, 'es_ES.UTF-8');
    date_default_timezone_set('Europe/Madrid');
    $ahora=new DateTime();
    $fecha = strftime("Hoy es %A, %d de %B de %Y y son las %H:%M:%S",
    echo $fecha;
?>
</body>
</html>
```

1.4.- Variables especiales de PHP.

En la unidad anterior ya aprendiste qué eran y cómo se utilizaban las variables globales. PHP incluye unas cuantas variables internas predefinidas que pueden usarse desde cualquier ámbito, por lo que reciben el nombre de **variables superglobales**. Ni siquiera es necesario que uses `global` para acceder a ellas.

WikiMedia Commons (Dominio público)

Cada una de estas variables es un array que contiene un conjunto de valores (en esta unidad veremos más adelante cómo se pueden utilizar los arrays). Las variables superglobales disponibles en PHP son las siguientes:

1.- `$_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
`$_SERVER['PHP_SELF']`	guion que se está ejecutando actualmente.
`$_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'`)

2.- `$_GET`, `$_POST` y `$_COOKIE` contienen las variables que se han pasado al guión actual utilizando respectivamente los métodos `GET` (parámetros en la url), HTML `POST` y Cookies HTML.
3.- `$_REQUEST` junta en uno solo el contenido de los tres arrays anteriores, `$_GET`, `$_POST` y `$_COOKIE`.
4.- `$_ENV` contiene las variables que se puedan haber pasado a PHP desde el entorno en que se ejecuta.
5.- `$_FILES` contiene los ficheros que se puedan haber subido al servidor utilizando el método `POST`.
6.- `$_SESSION` contiene las variables de sesión disponibles para el guion actual.
7.- `$GLOBALS` Es un array asociativo que contiene las referencias a todas la variables que están definidas en el ámbito global del script. Los nombres de las variables son las claves del array.

En posteriores unidades iremos trabajando con estas variables.

Debes conocer

En la documentación de PHP puedes consultar toda la información que ofrece $_SERVER:

Documentación de $_SERVER

Para saber más

Conviene tener a mano la información sobre estas variables superglobales disponible en el manual oficial de PHP:

Variables superglobales.

Autoevaluación

Relaciona cada variable con la información que contiene:

Ejercicio para relacionar.

Parámetro	Relación	Finalidad
$_SERVER['DOCUMENT_ROOT']		1. Variables de entorno disponibles.
$_ENV		2. Guión que se está ejecutando.
$_SESSION		3. Variables de sesión disponibles.
$_SERVER['PHP_SELF']		4. Directorio raíz del guión actual.

Enviar

Es importante conocer cómo podemos acceder a la información que nos ofrecen las variables de PHP.

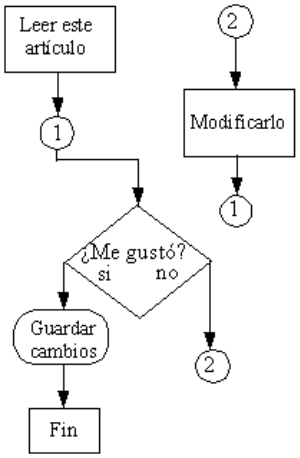
2.- Estructuras de control.

Caso práctico

¡Bien! Ya están claros los fundamentos del lenguaje.

Pero con lo visto hasta el momento, solo es posible hacer programas muy sencillos. Para poder empezar a programar, **Carlos** sabe qué debe estudiar a continuación. Una de las partes más importantes de cualquier lenguaje es la que permite tomar decisiones, es decir, las sentencias que se pueden usar para indicar bajo qué condiciones se debe ejecutar una instrucción o un bloque de instrucciones. Y cómo no, también las sentencias para repetir la ejecución de ciertas líneas de código.

Cuando domine esas estructuras, podrá empezar a probar todo lo que lleva aprendido.



Gengiskanhg (CC BY-SA)

Los guiones PHP se construyen en base a sentencias. Utilizando llaves, puedes agrupar las sentencias en conjuntos, que se comportan como si fueran una única sentencia.

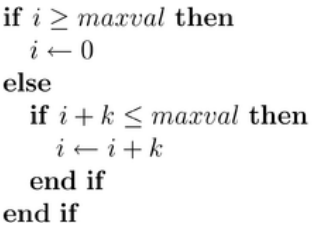
Para definir el flujo de un programa en PHP, al igual que en la mayoría de lenguajes de programación, hay sentencias para dos tipos de **estructuras de control**: **sentencias condicionales**, que permiten definir las condiciones bajo las que debe ejecutarse una sentencia o un bloque de sentencias; y **sentencias de bucle**, con las que puedes definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.

Además, en PHP puedes usar también (aunque no es recomendable) la sentencia `goto`, que te permite saltar directamente a otro punto del programa que indiques mediante una etiqueta.

```php
<?php
    $a = 1;
    goto salto;
    $a++;  //esta sentencia no se ejecuta
    salto:
    echo $a;  // el valor obtenido es 1
?>
```

2.1.- Condicionales.

Las sentencias condicionales son sentencias que alteran el flujo de ejecución haciendo que se ejecute un bloque u otro según la evaluación de una determinada codición.



Nemti (Dominio público)

- **if / elseif / else**. La sentencia **if** permite definir una expresión para ejecutar o no la sentencia o conjunto de sentencias siguiente. Si la expresión se evalúa a **true** (verdadero), la sentencia se ejecuta. Si se evalúa a **false** (falso), no se ejecutará.
  Cuando el resultado de la expresión sea **false**, puedes utilizar **else** para indicar una sentencia o grupo de sentencias a ejecutar en ese caso. Otra alternativa a **else** es utilizar **elseif** y escribir una nueva expresión que comenzará un nuevo condicional.

```php
<?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, como sucede en el ejemplo, la sentencia **if elseif** o **else** actúe sobre una única sentencia, no será necesario usar llaves. Tendrás que usar llaves para formar un conjunto de sentencias siempre que quieras que el condicional actúe sobre más de una sentencia.

- **switch**. La sentencia **switch** es similar a enlazar varias sentencias **if** comparando una misma variable con diferentes valores. Cada valor va en una sentencia **case**. Cuando se encuentra una coincidencia, comienzan a ejecutarse las sentencias siguientes hasta que acaba el bloque **switch**, o hasta que se encuentra una sentencia **break**. Si no existe coincidencia con el valor de ningún **case**, se ejecutan las sentencias del bloque **default**, en caso de que exista.

```php
<?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

Carlos ha decidido hacer su primer programa, un taller mecánico les ha propuesto que le hagan una web que, en función del tipo de motor, recomiende un aceite u otra.

Haz una página que en función de la variable `$motor` que puede tomar los valores 1 (Gasolina), 2 (Diésel), 3 (Motocicleta), 4 (Eléctrico) nos muestre el tipo de motor, es decir si `$motor=1` nos mostrará "***El motor es de Gasolina***". Hazlo de dos formas, con bucles `if` y con `switch`.

Mostrar retroalimentación

Compara la solución propuesta con la que has obtenido.

Solución

Autoevaluación

¿Siempre se puede sustituir una sentencia `switch` por otra sentencia o sentencias `if`?

- No.
- Sí.

Incorrecto. Piensalo bien y busca algún ejemplo en el que creas que no se pueda sustituir.

Efectivamente. Una sentencia `switch` se puede sustituir siempre por una o varias sentencias `if` en las que las condiciones de cada `if` serán comparar el valor de la variable con cada una de las constantes que figuran en las sentencias `case`.

Solución

1. Incorrecto
2. Opción correcta

2.2.- Bucles.

Los bucles o sentencias repetitivas son estructuras que permiten repetir una secuencia de sentencias mientras se de cierta condición.

dannya (Dominio público)

- **while:** Usando while puedes definir un bucle que se ejecuta mientras se cumpla una expresión. La expresión se evalúa antes de comenzar cada ejecución del bucle.

```php
<?php
    $a = 1;
    while ($a < 8){
        $a += 3;
    }
    echo $a; // el valor obtenido es 10
?>
```

- **do / while:** Es un bucle similar al anterior, pero la expresión se evalúa al final, con lo cual se asegura que la sentencia o conjunto de sentencias del bucle se ejecutan al menos una vez.

```php
<?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 2
?>
```

- **for:** Son los bucles más complejos de PHP. Al igual que los del **lenguaje C**, se componen de tres expresiones:

```
for (expr1; expr2; expr3){
     sentencia o conjunto de sentencias;
}
```

La primera expresión, **expr1**, se ejecuta solo una vez al comienzo del bucle.

La segunda expresión, **expr2**, se evalúa para saber si se debe ejecutar o no la sentencia o conjunto de sentencias. Si el resultado el **false**, el bucle termina.

Si el resultado es **true**, se ejecutan las sentencias y al finalizar se ejecuta la tercera expresión, **expr3**, y se vuelve a evaluar **expr2** para decidir si se vuelve a ejecutar o no el bucle.

```php
<?php
    for ($a = 5; $a<10; $a+=3) {
        print $a; // Se muestran los valores 5 y 8
        print "<br />";
    }
?>
```

Puedes anidar cualquiera de los bucles anteriores en varios niveles. También puedes usar las sentencias **break**, para salir del bucle, y **continue**, para omitir la ejecución de las sentencias restantes y volver a la comprobación de la expresión respectivamente. ***break*** acepta un argumento numérico opcional que indica de cuántas estructuras anidadas circundantes se debe salir y de igual manera ***continue*** también acepta un argumento numérico opcional, que indica a cuántos niveles de bucles encerrados se ha de saltar al final.

Para saber más

En el siguiente enlace puedes encontrar un videotutorial en el que se muestra con ejemplos la utilización de bucles en PHP.

Videotutorial sobre la utilización de bucles en PHP

Autoevaluación

Si quieres mostrar una cadena de texto letra a letra, y no sabes si está vacía, ¿qué tipo de bucle emplearías, while o do-while?

- ○ **while.**
- ○ **do-while.**

Efectivamente. Como no sabemos si está o no vacía, antes de imprimir algún carácter hay que comprobar si existe, incluido el primero. Con un bucle do-while se intentaría mostrar una letra antes de comprobar si existe.

Incorrecto. Piensa en cómo harías el algoritmo y lo que se mostraría en pantalla si la cadena estuviese vacía.

Solución

1. Opción correcta
2. Incorrecto

3.- Funciones.

Caso práctico

Juan observa con agrado los **progresos que va haciendo Carlos** en su aprendizaje del lenguaje PHP. Con la ilusión que está poniendo, se integrará sin problemas en el nuevo proyecto. Cuantos más puedan colaborar, mejor.

Tras lo que ya ha visto, le recomienda que aprenda a **crear y utilizar funciones**. Sabe que no sólo es muy importante saber usarlas, sino también conocer todas las que hay disponibles en el lenguaje, o al menos, saber cómo buscarlas. En un lenguaje abierto como PHP, si sabes utilizar el código que ya hay programado puedes ahorrarte una gran parte del trabajo.

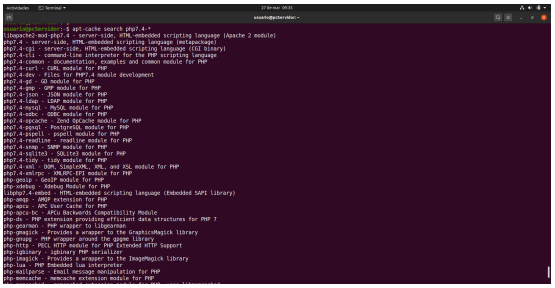
Cuando quieres repetir la ejecución de un bloque de código, puedes utilizar un bucle. Las **funciones** tienen una utilidad similar: **nos permiten asociar una etiqueta** (el nombre de la función) **con un bloque de código** a ejecutar. Además, al usar funciones estamos ayudando a estructurar mejor el código. Como ya sabes, las funciones permiten crear variables locales que no serán visibles fuera del cuerpo de las mismas.

Como programador puedes aprovecharte de la gran cantidad de funciones disponibles en PHP. De éstas, muchas están incluidas en el núcleo y se pueden usar directamente. Otras muchas se encuentran disponibles en forma de extensiones, y se pueden incorporar al lenguaje cuando se necesitan.

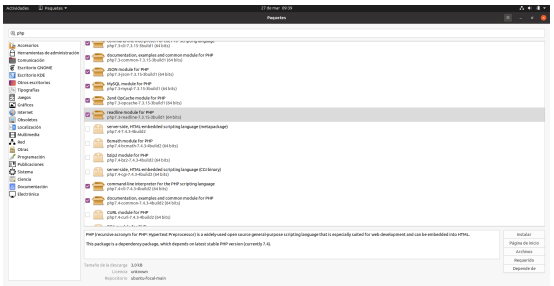
Con la distribución de PHP se incluyen varias extensiones. Para poder usar las funciones de una extensión, tienes que asegurarte de activarla mediante el uso de una directiva `extension` en el fichero `php.ini`. Muchas otras extensiones no se incluyen y antes de poder utilizarlas tienes que descargarlas.

Si estamos trabajando con Xampp ya viene PHP con todas las extensiones habilitadas, si trabajas con **linux** es posible que tengas que instalarlas a medida que vayas necesitándolas. Hay varias varias formas:

- Si sabes la extensión a instalar simplemente podemos teclear desde la consola: `sudo apt install extension`, por ejemplo `sudo apt install php-mbstring.`, normalmente el paquete es `php-nombre_extensión.`
- Si no sabemos exactamente el nombre de la extensión podemos buscar todas la extensiones disponibles escribiendo en la consola `sudo apt-cache search php7.4-*` (sustituye **7.4** por la versión que tengas). Esto te mostrará las extensiones disponibles, luego con el comando de arriba instalas la extensión que necesites, o bien usando el gestor de paquetes de **Ubuntu** o los de la distrubución Linux con la que trabajes.



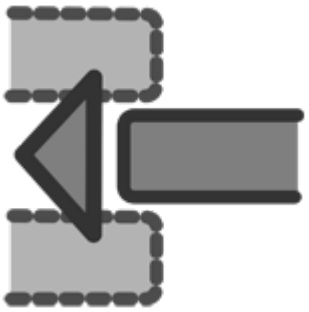
Terminal de Ubuntu (Elaboración propia)



Gestor de paquetes de Ubuntu (Elaboración propia)

3.1.- Inclusión de ficheros externos.

Conforme vayan creciendo los programas que hagas, verás que resulta trabajoso encontrar la información que buscas dentro del código. En ocasiones resulta útil agrupar ciertos grupos de funciones o bloques de código, y ponerlos en un fichero aparte. Posteriormente, puedes hacer referencia a esos ficheros para que PHP incluya su contenido como parte del programa actual.



dannya (Dominio público)

Para incorporar a tu programa contenido de un archivo externo, tienes varias posibilidades:

- **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 ubicación del fichero puede especificarse utilizando una ruta absoluta, pero lo más usual es con una ruta relativa. En este caso, se toma como base la ruta que se especifica en la directiva **include_path** del fichero **php.ini**. Si no se encuentra en esa ubicación, se buscará también en el directorio del guion actual, y en el directorio de ejecución.
- **include_once**: Si por equivocación incluyes más de una vez un mismo fichero, lo normal es que obtengas algún tipo de error (por ejemplo, al repetir una definición de una función). **include_once** funciona exactamente igual que **include**, pero solo incluye aquellos ficheros que aún no se hayan incluido, hay que tener cuidado de no abusar de esta opción pues es mucho más pesada al tener que comprobar una y otra vez los ficheros incluidos.
- **require**: Si el fichero que queremos incluir no se encuentra, **include** da un aviso y continua la ejecución del guion. La diferencia más importante al usar **require** es que en ese caso, cuando no se puede incluir el fichero, se detiene la ejecución del guion.
- **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, al igual que **include_once** y por los mismos motivos no conviene abusar de esta opción.

Cuando se comienza a evaluar el contenido del fichero externo, se abandona de forma automática el modo PHP y su contenido se trata en principio como etiquetas HTML. Por este motivo, es necesario delimitar el código PHP que contenga nuestro archivo externo utilizando dentro del mismo los delimitadores **<?php** y **?>**.

Ejercicio resuelto de ejemplo

Te presentamos un ejemplo de utilización de include. Tendremos dos ficheros en el mismo directorio definiciones.php y programa.php.

Mostrar retroalimentación

definiciones.php

```php
<?php
    $modulo = 'DWES';
    $ciclo = 'DAW';
?>
```

programa.php

```php
<?php
    echo "Módulo $modulo del ciclo $ciclo<br />"; //Solo muestra "Modu
    include 'definiciones.php'; //si estuviera en otro directorio tend
    print " Módulo $modulo del ciclo $ciclo<br />"; // muestra "Modulo
?>
```

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.

Autoevaluación

¿Puedes utilizar include o require para incluir el mismo encabezado HTML en varias páginas?

- Sí.
- No.

Efectivamente. Como ya viste, el contenido de los ficheros externos se traba como HTML a no ser que figuren los delimitadores <?php y ?>. No es necesario incluir código PHP en un fichero externo para poder utilizar include o require con él.

Incorrecto, ¿hay alguna limitación que lo impida?

Solución

1. Opción correcta
2. Incorrecto

3.2.- Ejecución y creación de funciones.

Ya sabes que para hacer una llamada a una función, basta con poner su nombre y unos paréntesis.

```php
<?php phpinfo();
    // al ser un archivo que solo va a contener php no se
```

mothinator (Dominio público)

Para crear tus propias funciones, deberás usar la palabra **function**.

En PHP no es necesario que definas una función antes de utilizarla, excepto cuando está condicionalmente definida, veamos dos ejemplos que hacen lo mismo:

```php
<?php
    $iva = true;
    $precio = 10;
    precioConIva();     // esta línea dará error, coméntala
    if ($iva) {
        function precioConIva() {
            global $precio; //podemos usar también $precio = $GLOBALS["precio"];
            $precioIva = $precio * 1.18;
            echo "El precio con IVA es ".$precioIva;
        }
    }
    precioConIva();     // Aquí ya no da error
?>
```

```php
<?php
    $iva = true;
    $precio = 10;
    if ($iva) {
        //podemos hacer uso de la función
        //Antes de implementarla.
        precioConIva();
    }
    function precioConIva() {
        $precio=$GLOBALS["precio"];
        $precioIva = $precio * 1.18;
        echo "El precio con IVA es ".$precioIva;
    }
?>
```

Cuando una función está definida de una forma condicional sus definiciones deben ser procesadas antes de ser llamadas. Por tanto, la definición de la función debe estar antes de cualquier llamada.

3.3.- Argumentos.

En el ejemplo anterior en la función usabas una variable global, lo cual no es una buena práctica. Siempre es mejor utilizar argumentos o parámetros al hacer la llamada. Además, en lugar de mostrar el resultado en pantalla o guardar el resultado en una variable global, las funciones pueden devolver un valor usando la sentencia **return**. Cuando en una función se encuentra una sentencia **return**, termina su procesamiento y devuelve el valor que se indica.

ben (Dominio público)

Puedes reescribir la función anterior de la siguiente forma:

```php
<?php
    function precioConIva($precio){
        return $precio * 1.18;
    }
    $precio = 10;
    $precioIva = precioConIva($precio);
    echo  "El precio con IVA es $precioIva";
?>
```

Los argumentos se indican en la definición de la función como una lista de variables separada por comas. No se indica el tipo de cada argumento, al igual que no se indica si la función va a devolver o no un valor (si una función no tiene una sentencia **return**, devuelve **null** al finalizar su procesamiento).

Al definir la función, puedes 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.

```php
<?php
    function precioConIva($precio, $iva=0.18) {
        return $precio * (1 + $iva);
    }
    $precio = 10;
    $precioIva = precioConIva($precio); //al no especificar tomará el valor 0.18
    echo "El precio con IVA es $precioIva";
?>
```

```php
<?php
    function precioConIva($precio, $iva=0.18) {
        return $precio * (1 + $iva);
    }
    $precio = 10;
    $precioIva = precioConIva($precio, 0.23); //ahora $iva=0.23
    echo "El precio con IVA es $precioIva";
```

```
?>
```

Puede haber valores por defecto definidos para varios argumentos, pero en la lista de argumentos de la función todos ellos deben estar a la derecha de cualquier otro argumento sin valor por defecto.

En los ejemplos anteriores los argumentos se pasaban **por valor**. Esto es, cualquier cambio que se haga dentro de la función a los valores de los argumento no se reflejará fuera de la función. Si quieres que esto ocurra debes definir el parámetro para que su valor se pase **por referencia**, añadiendo el símbolo **&** antes de su nombre.

```php
<?php
    function precioConIva(&$precio, $iva=0.18) {
        $precio *= (1 + $iva);
    }
    $precio = 10;
    echo "El precio con IVA es  $precio";
?>
```

Para saber más

A partir de la versión 7.0 de PHP, se puede especificar el tipo de dato que le pasamos a la función y lo que va a devolver la misma. Fíjate en el código siguiente:

```php
<?php
    function precioConIva(float $precio) :float{ //con :float especificamos
        return $precio * 1.18;
    }
    $precio = 10;
    $precioIva = precioConIva($precio);
    echo  "El precio con IVA es $precioIva";
?>
```

Autoevaluación

¿Está bien definida una función con el siguiente encabezado?

```
function precio_final (&$precio, $iva=0.18, $aplicar_iva)
```

- Sí.
- No.

Incorrecto, fíjate en los argumentos.

Correcta. La función no puede tener esos argumentos, pues los opcionales ($iva) deben estar a la derecha de cualquier otro que no tenga valor por defecto ($aplicar_iva).

Solución

1. Incorrecto
2. Opción correcta

Ejercicio resuelto

Con la ayuda de las funciones que necesites, haz un programa que, dados dos número enteros positivos, inicio y cantidad, nos muestre cantidad de números primos a partir de inicio, si no pasamos ningún valor cantidad=10.

Mostrar retroalimentación

Compara la solución propuesta con la que has obtenido.

Solución

4.- Tipos de datos compuestos.

Caso práctico

Es muy raro el programa que utilice solo tipos simples, y más en PHP. **Carlos ya ha asumido que tendrá que utilizar arrays** para casi cualquier código que haga. La información del servidor, los datos que introduce el usuario, o las cadenas de texto. Gran parte de las variables que se usan en un programa están en forma de array.

Por tanto, el siguiente paso está claro: **hay que dominar los arrays**. Crearlos, utilizarlos, recorrerlos... Y como acaba de aprender, antes de ponerse a programar le echará un vistazo a las funciones que ya existen para manejarlos. Si las puede aprovechar en sus programas, ¡mejor!

Un tipo de datos compuesto es aquel que te permite almacenar más de un valor. En PHP puedes utilizar dos tipos de datos compuestos: el **array** y el **objeto**. Los objetos los veremos más adelante; vamos a empezar con los arrays.

Un **array** es un tipo de datos que nos 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.

```php
// array numérico

$modulos1 = array(0 => "Programación", 1 => "Bases de datos", ..., 9 => "Desarrollo web en

// array asociativo

$modulos2 = array("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "Desarr
```

A partir de PHP 5.4 también se puede usar la sintaxis de array corta, la cual reemplaza *array()* con *[]*. El código anterior quedaría:

```php
//array número

$modulos1=[0=>"Programacion", 1=>"Bases de Datos", ... ,2=>"Desarrollo web en

//array asociativo
```

```
$modulos2=["PR"=>"Programacion", "BD"=>"Bases de datos", ... , "DWES"=>"Desarr
```

👀 Debes conocer

En PHP existen varias funciones para ver el contenido de un array sin tener que recorrerlo. Las más usadas son `print_r($array)` y `var_dump($array)`. En realidad `var_dump()` tiene más usos que iremos viendo a lo largo del curso.

Para hacer referencia a los elementos almacenados en un array, tienes que utilizar el valor clave entre corchetes:

```
$modulos1 [9]

$modulos2 ["DWES"]
```

Los arrays anteriores son vectores, esto es, arrays unidimensionales. En PHP puedes crear también arrays de varias dimensiones almacenando otro array en cada uno de los elementos de un array.

```
//array bidemensional

$ciclos = array(
    "DAW" => array ("PR" => "Programación", "BD" => "Bases de datos", "PMDMO"=>"Programaci
    "DAM" => array ("PR" => "Programación", "BD" => "Bases de datos", "DWES"=>"Desarrollo
);

//En formato [ ]

$ciclos=[
    "DAW"=>["PR" => "Programación", "BD" => "Bases de datos", "PMDMO"=>"Programacion Multi
    "DAM"=>["PR"=>"Programacion", "BD"=>"Bases de datos", "DWES"=>"Desarrollo web"]
];
```

Para hacer referencia a los elementos almacenados en un **array multidimensional**, debes indicar las claves para cada una de las dimensiones: `$ciclos ["DAW"] ["DWES"]`

No es necesario que indiques el tamaño del array antes de crearlo. Ni siquiera es necesario indicar que una variable concreta es de tipo array. Simplemente puedes comenzar a asignarle valores:

```php
// 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";
```

Ni siquiera es necesario que especifiques el valor de la clave. Si la omites, el array se irá llenando a partir de la última clave numérica existente, o de la posición **0** si no existe ninguna:

```php
$modulos1[ ] = "Programación"; //asignará el índice 0
$modulos1[ ] = "Bases de datos"; //asignará el índice 1
$modulos1[7] = "Desarrollo web en entorno servidor";  //asignamos el índice 7
$modulos1[] = "Desarrollo Web"; //asignará el índice 8
```

4.1.- Recorrer arrays (I).

Las **cadenas de texto** o **strings** se pueden tratar como arrays en los que se almacena una letra en cada posición, siendo **0** el índice correspondiente a la primera letra, **1** el de la segunda, etc.

dannya (Dominio público)

Para recorrer los elementos de un array, puedes usar un bucle específico: **foreach()**. Utiliza una variable temporal para asignarle en cada iteración el valor de cada uno de los elementos del array.

Puedes usarlo de dos formas:

- Recorriendo sólo los elementos.

```php
$modulos = arrays("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "
foreach ($modulos as $modulo) {
    echo "Módulo: ".$modulo. "<br />";
}
```

- Recorriendo los elementos y sus valores clave de forma simultánea.

```php
$modulos = arrays("PR" => "Programación", "BD" => "Bases de datos", ..., "DWES" => "
foreach ($modulos as $key => $value) {
    echo  "El código del módulo ".$value." es ".$key."<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.

Mostrar retroalimentación

Compara la solución propuesta con la que has obtenido.

Solución

Autoevaluación

La inicialización del array en el código siguiente, ¿generará algún error?

```php
$a[0] = 0;
$a[1] = "uno";
$a["tres"] = 3;
$a[] = 8;
```

- Sí.
- No.

Incorrecto. ¿Cuál crees que generará y por qué?

Correcto. No es necesario que la clave de un array sea siempre numérica o texto. Pueden mezclarse ambos tipos de valores o incluso omitirse.

Solución

1. Incorrecto
2. Opción correcta

4.2.- Recorrer arrays (II).

Pero en PHP también **hay otra forma de recorrer los valores de un array**. Cada array mantiene un **puntero** interno, que se puede utilizar con este fin. 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. Algunas de estas funciones son:

dannya (Dominio público)

Funciones para recorrer arrays.

Función	Resultado
reset()	Sitúa el puntero interno al comienzo del array.
next()	Avanza el puntero interno una posición.
prev()	Mueve el puntero interno una posición hacia atrás.
end()	Sitúa el puntero interno al final del array.
current()	Devuelve el elemento de la posición actual.
key()	Devuelve la clave de la posición actual.
each()	Devuelve un array con la clave y el elemento de la posición actual. Además, avanza el puntero interno una posición.

Las funciones reset(), next(), prev() y end(), además de mover el puntero interno devuelven, al igual que current(), el valor del nuevo elemento en que se posiciona. Si al mover el puntero te sales de los límites del array (por ejemplo, si ya estás en el último elemento y haces un next()), cualquiera de ellas 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 válida del array que contiene el valor "false".

La función key() devuelve null si el puntero interno está fuera de los límites del array.

La función each() devuelve un array con cuatro elementos: Los elementos **0** y **'key'** almacenan el valor de la clave en la posición actual del puntero interno y los elementos **1** y **'value'** devuelven el valor almacenado.

Si el puntero interno del array se ha pasado de los límites del mismo, la función each() devuelve false, por lo que la puedes usar para crear un bucle que recorra el array de la

siguiente forma:

```
while ($modulo = each($modulos)) {
    echo "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.

Mostrar retroalimentación

Compara la solución propuesta con la que has obtenido.

Solución

4.3.- Funciones relacionadas con los tipos de datos compuestos.

Además de asignando valores directamente, **la función** `array()` **permite crear un array con una sola línea de código**, tal y como vimos anteriormente. Esta función recibe un conjunto de parámetros, y crea un array a partir de los valores que se le pasan. Si en los parámetros no se indica el valor de la clave, crea un array numérico (con base 0). Si no se le pasa ningún parámetro, crea un array vacío.

dannya (Dominio público)

```php
$a = array();  // array vacío

$modulos = array("Programación", "Bases de datos", ..., "Desarr
```

Una vez definido un array puedes añadir nuevos elementos (no definiendo el índice, o utilizando un índice nuevo) y modificar los ya existentes (utilizando el índice del elemento a modificar). También se pueden eliminar elementos de un array utilizando la función `unset()`.

En el caso de los arrays numéricos, eliminar un elemento significa que las claves del mismo ya no estarán consecutivas.

```php
unset ($modulos [0]);

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

La función `array_values()` recibe un array como parámetro, y devuelve uno nuevo con los mismos elementos y con índices numéricos consecutivos con base 0.

Para comprobar si una variable es de tipo array, utiliza la función `is_array()`. Para obtener el número de elementos que contiene un array, tienes la función `count()`.

Si quieres buscar un elemento concreto dentro de un array, puedes utilizar la función `in_array()`. Recibe como parámetros el elemento a buscar y la variable de tipo array en la que buscar, y devuelve `true` si encontró el elemento o `false` en caso contrario.

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

$modulo = "Bases de datos";

if (in_array($modulo, $modulos)) echo "Existe el módulo de nombre $modulo";
```

Otra posibilidad es la función `array_search()`, que recibe los mismos parámetros pero devuelve la clave correspondiente al elemento, o `false` si no lo encuentra.

Y si lo que quieres buscar es un clave en un array, tienes la función `array_key_exists()`, que devuelve `true` o `false`.

Para saber más

Según la documentación de PHP, existen un total de 79 funciones de arrays. Puedes consultarlas en el siguiente enlace.

Lista completa.

Autoevaluación

¿Se puede usar el siguiente código para recorrer un array `$a` cualquiera?

```
while ($variable = $current($a))
{

    …

    next($a);
}
```

- No.
- Sí.

Correcto. El bucle anterior funcionará bien sólo en aquellos arrays en los que estemos seguros de que no existe ningún valor "`false`".

Incorrecto. Piensa si existe alguna condición que haga que se pueda detener el procesamiento del array antes de que se hayan recorrido todos sus elementos.

Solución

1. Opción correcta
2. Incorrecto

5.- Formularios web.

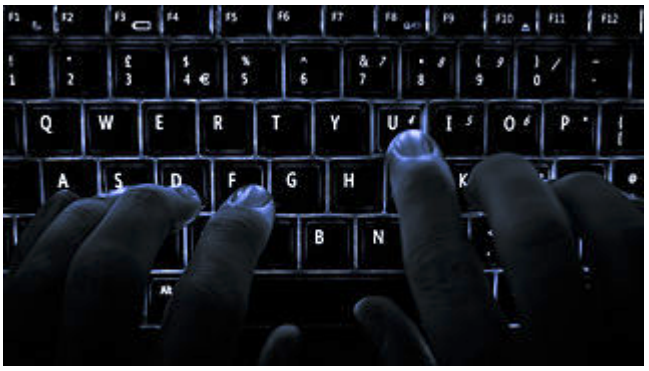
Caso práctico

Carlos está viendo que el esfuerzo que le dedica al **aprendizaje del nuevo lenguaje** empieza a dar sus frutos. Hace unos días casi no sabía ni que existía PHP, y ahora ya es capaz de realizar programas sencillos por sí mismo.

Para poder avanzar aún más, sabe cuál ha de ser su siguiente paso: **obtener y utilizar información de un usuario**. De esta forma, los programas que haga no serán lineales, sino que tendrán un comportamiento u otro en función de los datos que aporte el usuario.

Como le ha comentado **Juan**, para obtener información de un usuario, en PHP se utilizan los **formularios HTML**. ¡A por ellos!

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



Colin (CC BY-SA)

Los formularios HTML van encerrados siempre entre las etiquetas `<FORM>` `</FORM>`. Dentro de un formulario se incluyen los elementos sobre los que puede actuar el usuario, principalmente usando las etiquetas `<INPUT>`, `<SELECT>`, `<TEXTAREA>` y `<BUTTON>`.

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 script PHP.

Por su parte, 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, si hay varios se separan por "&".
- `post`: con este método los datos se incluyen en el cuerpo del formulario y se envían utilizando el protocolo HTML.

Como vamos a ver, los datos se recogerán de distinta forma dependiendo de cómo se envíen.

Para saber más

Para no tener problemas al programar en PHP, debes conocer el lenguaje HTML, concretamente los detalles relativos a la creación de formularios web. Puedes consultar esta información por ejemplo en el curso sobre HTML de aulaClic:

Curso sobre formularios HTML de aulaClic.

Ejercicio resuelto

Crea un formulario HTML para introducir el nombre del alumno y el módulo o los módulos que cursa, a escoger "Desarrollo Web en Entorno Servidor", "Desarrollo Web en Entorno Cliente" o ambas. Envía el resultado a la página "procesa.php", que será la encargada de procesar los datos. No es necesario, en este ejercicio, que hagas la página de procesar datos.

Mostrar retroalimentación

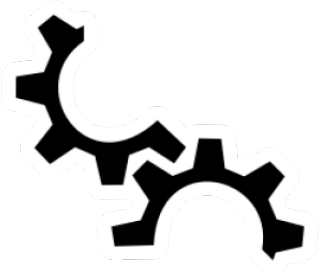
Compara la solución propuesta con la que has obtenido.

Solución

Fíjate que si en un formulario web tienes que enviar alguna variable en la que sea posible almacenar más de un valor, como es el caso de las casillas de verificación en el ejemplo anterior (se pueden marcar varias a la vez), tendrás que ponerle corchetes al nombre de la variable para indicar que se trata de un array.

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

En el ejemplo anterior creaste un **formulario en una página HTML** que recogía datos del usuario y los enviaba a una página PHP para que los procesara. Como usaste el método POST, los datos se pueden recoger utilizando la variable $_POST. Si simplemente los quisieras mostrar por pantalla, éste podría ser el código de "***procesa.php***" dependiendo que hayamos elegido POST o GET en <form>



dannya (CC0)

Si "method" es GET

```php
<?php
    echo "Tu nombre es: {$_GET['nombre']}";
    $totalModulos = 0;
    //comprobamos si nos ha llegado algún módulo
    if (isset($_GET['modulo'])) {
        $totalModulos = count($_GET['modulo']); //los contamos
        echo "<br>Los módulos elegidos han sido: ";
        echo "<ol>";
        foreach ($_GET['modulo'] as $k => $v) { //los recorremos y mostramos
            echo "<li>$v</li>";
        }
       echo "</ol>";
    }
    echo "<br>Has elegido un total de: $totalModulos módulos";
?>
```

Si "method" es POST

```php
<?php
     echo "Tu nombre es: {$_POST['nombre']}";
     $totalModulos = 0;
     //comprobamos si nos ha llegado algún módulo
     if (isset($_POST['modulo'])) {
        $totalModulos = count($_POST['modulo']); //los contamos
         echo "<br>Los módulos elegidos han sido: ";
```

```php
        echo "<ol>";
        foreach ($_POST['modulo'] as $k => $v) { //los recorremos y mostramos
            echo "<li>$v</li>";
        }
        echo "</ol>";
    }
    echo "<br>Has elegido un total de: $totalModulos módulos";
?>
```

En cualquiera de los dos casos podrías haber usado `$_REQUEST` sustituyendo respectivamente a `$_POST` y a `$_GET`.

Siempre que sea posible, es preferible **validar los datos que se introducen en el navegador antes de enviarlos**. Para ello deberás usar código en **lenguaje JavaScript**.

Si por algún motivo hay datos que se tengan que validar en el servidor, por ejemplo, porque necesites comprobar que los datos de un usuario no existan ya en la base de datos antes de introducirlos, será necesario hacerlo con código PHP en la página que figura en el atributo `action` del formulario.

Para saber más

Es bastante usual que el "**action**" del formulario sea él mismo, para ello podemos usar conceptos ya estudiados como `$_SERVER['PHP_SELF']` y bloques condicionales. Echa un vistazo al ejemplo que se propone.

Ejemplo

5.2.- Validación de formularios web en PHP.

Si pensamos en validaciones de formulario es inevitable pensar en Javascript, ya que uno de los motivos por los cuales se implementó este lenguaje es para evitar carga de trabajo inútil en el servidor, no obstante hay que tener presente que las validaciones en el navegador, aunque necesarias para extender la funcionalidad del Frontend, pueden ser vulneradas fácilmente, ya que el usuario puede tener Javascript desactivado o bien, puede tener conocimientos medios de web y saltearse estas validaciones.

dannya (CC0)

Por eso las validaciones en el lado del servidor (backend) son una herramienta mucho más segura, y es importante que las validaciones con PHP, o cualquier otro lenguaje de servidor nunca falten, ya que serán necesarias para evitar la entrada de datos incorrectos por ejemplo en una base de datos.

Precisamente para evitar trabajo inútil al servidor podemos usar las características del lenguaje HTML sobre todo en su versión 5 con algunas prácticas recomendadas como son:

- **required** en campos que no vayamos a permitir vacíos, no comprueba que por ejemplo enviemos solo espacios en blanco.
- Especificar longitud de cadenas con atributos como el **maxlength, max, min…**
- Poner en el **type** de los **input,** el tipo de datos que esperamos: **mail, text, number, date …**
- Especificar en los **<input type='file'>** el atributo **accept** para que el navegador de archivos nos muestre solo el tipo permitido (ejemplo **accept=".pdf"**)
- Y muchos más.

Veamos, con el ejercicio resuelto siguiente, cómo podemos hacer algunas validaciones de formularios en el lado del servidor usando PHP Por claridad haremos que el **action** del formulario vaya a otra página, pero ya sabemos que lo podemos incluir en la misma podemos ver que el HTML nos ha ayudado con algunas validaciones.

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

Ejercicio Resuelto

Haremos un formulario con los campos nombre, apellidos, mail, edad (numérico tipo entero y mayor que 0), y checkboxs de módulos de los que nos matricularemos (de los que al menos uno será necesario). Validaremos con ayuda de HTML y con PHP, que todos los campos contengan algún dato y que hayamos elegido algún módulo.

Si todo está bien mostraremos los datos y si no mostraremos los errores.

Mostrar retroalimentación

Intenta hacer uso de funciones. Compará tu solución con la propuesta.

Solución

Ejercicio resuelto

Haz un formulario que, a partir de un día, mes y año introducido por el usuario (antes de mostrar la fecha, se debe comprobar que es correcta) se muestre en la misma página del formulario la fecha en castellano, por ejemplo si el usuario introduce:

Día=3, Mes=6, Año=2020

Nos mostrará: Miércoles 3 de Junio de 2020

(Por comodidad usaremos años a partir de 1900)

Consulta la función `checkdate` en el manual de PHP.

Mostrar retroalimentación

Compara la solución propuesta con la que has obtenido.

Solución

5.3.- Subiendo archivos al servidor.

Es muy frecuente tener que subir un archivo al servidor como una imagen o cualquier otro tipo de fichero, para ello tendremos que hacer varias cosas tanto en el formulario como en el tratamiento que hagamos del mismo. Veremos en este apartado como podemos implementarlo.

OpenClipart-Vectors en Pixabay (Pixabay License)

Debes conocer

Recurda que había unas directivas (en el archivo **php.ini**) que permitían que nuestro servidor permitiese la subida de archivos y especificaban el tamaño máximo de las mismas, estas eran básicamente: **post_max_size, upload_max_filesize y file_uploads.**

Para poder subir archivos a nuestro servidor, lo primero será especificarlo en el formulario HTML con el atributo **enctype="multipart/form-data"**, sin esto no será posible. El campo de formulario que nos permitirá subir un archivo será un **input** de tipo **file.**

Si hemos hecho lo anterior, procesaremos los datos de tipo **file** enviados por el formulario a través del array global **$_FILES**, éste contendrá toda la información de los ficheros subidos. Los valores de este array serán:

- **$_FILES['fichero']['name']**: El nombre original del fichero en la máquina del cliente.
- **$_FILES['fichero']['type']**: El tipo MIME del fichero, si el navegador proporcionó esta información.
- **$_FILES['fichero']['size']**: El tamaño, en bytes, del fichero subido.
- **$_FILES['fichero']['tmp_name']**: El nombre temporal del fichero en el cual se almacena el fichero subido en el servidor.
- **$_FILES['fichero']['error']**: El código de error asociado a esta subida.

Si queremos especificar el tamaño máximo del archivo a subir debemos insertar el siguiente campo oculto antes de campo de tipo **file**. El tamaño es en bytes (lógicamente no debe ser superior a lo establecido en las directivas **php.ini**).

```
<input type="hidden" name="MAX_FILE_SIZE" value="30000" />
```

Si el archivo supera este tamaño `$_FILES['fichero']['error']` nos dará error el error 2.

Para saber más

Puedes ver los distintos códigos de error de `"$_FILES['fichero']['error']"` en el enlace siguiente:

Códigos de error

Básicamente los pasos que haremos en el servidor para guardar el archivo serán:

- Comprobamos si hemos subido un archivo usando: `is_uploaded_file($_FILES["fichero"]["tmp_name"]`.
- Comprobamos el tipo de archivo si fuese necesario usando: `$_FILES['fichero']['type']`.
- Movemos el archivo de la carpeta temporal (`tmp`) a donde vayamos a guardarlo usando: `move_uploaded_file($_FILES["fichero"]["tmp_name"], carpeta)`, lógicamente el Servidor Web tiene que tener permisos de escritura en "`carpeta`".

Ejercicio Resuelto

Veamos un ejercicio resuelto donde detallamos el proceso de subir un archivo, controlaremos el tamaño máximo del archivo (ve cambiando los valores de `MAX_FILE_SIZE` para probar los errores) y el tipo del mismo.

Crearemos un formulario con un `input` de tipo `file` para subir un documento PDF a una carpeta "documentos" del servidor. Comprueba errores, el tamaño máximo del archivo será de 50000 bytes.

Mostrar retroalimentación

Mira una propuesta del ejercicio resuelto:

Solución

Autoevaluación

¿Serviría el siguiente código para comprobar si se han recibido los datos de un formulario?

```
if (count($_REQUEST)>0)
{

    …

}
```

- Sí.
- No.

Correcto. De esta forma se comprueba si se ha enviado algún formulario. Habría que comprobar también si los datos que se han enviado son correctos.

Eso no es correcto. La función count comprueba el número de elementos que contiene el array, y $_REQUEST solo debe contener elementos si se han recibido datos de un formulario.

Solución

1. Opción correcta
2. Incorrecto

Anexo I.- Utilización de la función print en PHP.

```
<!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
</head>
<body>
<?php
   $modulo="DWES";
   print "<p>Módulo: ";
   print $modulo;
   print "</p>"
?>
</body>
</html>
```

Anexo II.- Solución propuesta I.

```php
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Taller</title>
</head>
<body style="background:azure">
    <h3 style="text-align:center; font-size:bold">Taller</h3>
    <?php
    $motor = 1; //podemos dar valores 2,3,4, o probar uno no válido
    //1.- Con if elseif else ------------------------------------------
    if ($motor == 1) {
        echo "El motor es de Gasolina<br>";
    } elseif ($motor == 2) {
        echo "El motor es Diesel<br>";
    } elseif ($motor == 3) {
        echo "El motor es de una Motocicleta<br>";
    } elseif ($motor == 4) {
        echo "El motor es Eléctrico<br>";
    } else {
        echo "Error, el tipo de motor NO es válido<br>";
    }
    //2.- con switch ---------------------------------------------------
    switch ($motor) {
        case 1:
            echo "El motor es de Gasolina<br>";
            break;
        case 2:
            echo "El motor es Diesel<br>";
            break;
        case 3:
            echo "El motor es de una Motocicleta<br>";
            break;
        case 4:
            echo "El motor es Eléctrico<br>";
            break;
        default:
            echo "Error, el tipo de motor NO es válido<br>";
    }
    ?>
</body>
</html>
```

Anexo III.- Solución propuesta II.

Archivo "funciones.inc.php"

```php
<?php
    //
    function isPrimo($num){
        if($num==1) return false;
        for($i=2; $i<$num; $i++){
            if($num%$i==0) return false; //si encuentro un divisor distinto de 1 o $num el
            if($i>$num/2) break; //si no he encontrado divisores a la mitad, no los encont
        }
        return true;
    }
//No es necesario cerrar el script
```

Archivo "ejercicio.php"

```php
<!DOCTYPE html>
<?php
    //incluimos el archivo "funciones.inc.php"
    include "funciones.inc.php";
    function mostrarPrimos($inicio, $cantidad=10){
        //si no especifico nada, cantidad=10
        $cont=0;
        do{
            if(isPrimo($inicio)){
                echo '<strong>'. ++$cont . '=></strong> '. $inicio. '<br>';
            }
            $inicio++;
        }while($cont<$cantidad);

    }
?>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejercicio Primos</title>
</head>
<body style="background:bisque">
<h3 style="text-align:center; font-weight:bold">Solución Propuesta Ejercicio Primos</h3>
<?php
    $cantidad=10;
    $inicio=1;
    mostrarPrimos($inicio, $cantidad);
    //Nos deberá mostrar los primeros 10 primos a partir del número 1
?>
</body>
</html>
```

Anexo IV.- Solución propuesta III.

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Ejercicio</title>
</head>
<body>
    <table align="center" border="1" cellpadding="2" cellspacing="2">
        <tbody style="background-color: grey; text-align: center; font-weight: bold">
            <td>Clave</td>
            <td>Valor</td>
        </tbody>
    <?php
        foreach($_SERVER as $key=>$value){
            echo "<tr>";
            echo "<td>$key</td>";
            echo "<td>$value</td>";
            echo "<tr>";
        }
    ?>
    </table>
</body>
</html>
```

Anexo V.- Solución propuesta IV.

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Ejercicio</title>
</head>
<body>
    <table align="center" border="1" cellpadding="2" cellspacing="2">
        <tbody style="background-color: grey; text-align: center; font-weight: bold">
            <td>Clave</td>
            <td>Valor</td>
        </tbody>
    <?php
       while($dato=each($_SERVER)){
           echo "<tr>";
           echo "<td>$dato[0]</td>";
           echo "<td>$dato[1]</td>";
           echo "</tr>";
       }
    ?>
    </table>
</body>
</html>
```

Anexo VI.- Solución propuesta V.

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Formularios</title>
</head>
<body style="background: gainsboro">
<fieldset style="width:40%; margin:auto">
    <legend style="font-weight: bold">Datos</legend>
    <div name="form1" action="procesa.php" method="POST">
        <p>Nombre: <input type="text" name="nombre" placeholder="Nombre" required /></
        <fieldset style="width:50%">
            <legend style="font-weight: bold">Elige Modulos</legend>
            <p><input type="checkbox" name="modulo[]" value="DWESE" /> Desarrollo Web
            <p><input type="checkbox" name="modulo[]" value="DWEC" /> Desarrollo Web
        </fieldset>
        <div style="text-align: center; margin-top: 5px">
            <input type="submit" value="Enviar" name="enviar" />  
            <input type="reset" value="Limpiar" />
        </div>
    </form>
</fieldset>
</body>
</html>
```

Anexo VII.- Ejemplo action Formulario.

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Formularios</title>
</head>
<body style="background: gainsboro">
<h3 style="text-align: center; font-weight: bold">Login</h3>
<?php
    /*aqui procesaremos el formulario comprobando si hemos pulsado el submit
      si lo hemos hecho procesamos los datos, si no mostramos el formulario
      fijate donde cerramos el "else" */
    if(isset($_POST['enviar'])){
        // --> procesamos los datos
        echo "Tu nombre es: <b>{$_POST['nombre']}</b> y tu mail <b>{$_POST['mail']}</
    }
    else{
        // -->Si no hemos dado al boton enviar, pintamos el formulario
?>
<fieldset style="width:50%; margin:auto">
    <legend>Datos</legend>
    <form name="form1" action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
        <input type="text" name="nombre" placeholder="Nombre" required><br><br>
        <input tpe="mail" name="mail" placeholder="e-mail" name="mail" required><br><
        <input type="submit" value="Enviar" name="enviar">  
        <input type="reset" value="Limpiar">
    </form>
</fieldset>
<?php }  ?>
</body>
</html>
```

Fíjate en las siguientes líneas:

- En la línea **13** comprobamos si hemos dado al botón enviar "$_POST" por qué el "method" del formulario es POST, si hubiese sido GET hubiésemos puesto "$_GET". En ambos casos podíamos haber utilizado "$_REQUEST". Si es así, procesamos el formulario (en este caso solo mostramos los valores enviado con un "echo").
- En la línea **17,** si no hemos hecho el **submit** del formulario, lo que hacemos es mostrar el mismo.
- En la línea **22** podemos ver en el **action,** el valor de "$_SERVER['PHP_SELF']" que ya vimos que hacía referencia a la página actual.
- En la línea **29** cerramos el else abierto más arriba.

Anexo VIII.- Ejemplo de validación.

Código Formulario.

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width,
          user-scalable=no, initial-scale=1.0,
          maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Formularios</title>
</head>
<body style="background: gainsboro">
<fieldset style="width:40%; margin:auto">
    <legend style="font-weight: bold">Datos</legend>
    <form name="form1" action="procesa.php" method="POST">
        <p>
            <label for="n">Nombre:</label>
            <input type="text" name="nombre" placeholder="Nombre" id="n" required>
        </p>
        <p>
            <label for="a">Apellidos: </label>
            <input type="text" maxlength="60" name="apellidos" id="a" placeholder="Apellid
        </p>
        <p>
            <label for="e">Correo-e: </label>
            <input type="mail" placeholder="e-mail" required name="mail" required id="e">
        </p>
        <p>
            <label for="ed">Edad: </label>
            <input type="number" placeholder="edad" min="0"  id="ed" step="1" name="edad"
        </p>
        <fieldset style="width:50%">
            <legend style="font-weight: bold">Elige Modulos (Al menos 1)</legend>
            <p>
                <input type="checkbox" name="modulo[]" value="DWES">
                Desarrollo Web en Entorno Sevidor.
            </p>
            <p>
                <input type="checkbox" name="modulo[]" value="DWEC">
                Desarrollo Web en Entorno Cliente.
            </p>
            <p>
                <input type="checkbox" name="modulo[]" value="HLC">
                Horas de Libre Configuración.
            </p>
        </fieldset>
        <div style="text-align: center; margin-top: 5px">
            <input type="submit" value="Enviar" name="enviar">  
            <input type="reset" value="Limpiar" />
```

```
        </div>
    </form>
</fieldset>
</body>
</html>
```

Código action del formulario (procesa.php).

```
function existenModulos()
{
    global $errores;
    if (!isset($_POST['modulo'])) {
        $errores[] = "No has elegido ningun módulo revíselo";
        return false;
    }
    return true;
}
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no,
          initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Formularios</title>
</head>
<body style="background: gainsboro">
<?php
$nombre = trim($_POST['nombre']);
comprobarCadenas($nombre, "Nombre");
$apellidos = trim($_POST['apellidos']);
comprobarCadenas($apellidos, "Apellidos");
$mail = trim($_POST['mail']);
comprobarCadenas($mail, "Mail");
$edad = $_POST['edad'];
if (existenModulos()) {
    foreach ($_POST['modulo'] as $k => $v) {
        $modulos[] = $v;
    }
}
if (count($errores) > 0) {
    echo "Ha habido " . count($errores) . "errores, estos han sido:<br>";
    echo "<ol>";
    foreach ($errores as $k => $v) {
        echo "<li> $v";
    }
    echo "<ol>";
} else {
    echo "Sin errores. los datos son: ";
    echo "Apellidos, Nombre: " . $apellidos . ", " . $nombre;
    echo "<br>e-mail: " . $mail;
    echo "<br>Edad: " . $edad . " años";
```

```
    echo "<br>Módulos matriculados: <br>";
    echo "<ol>";
    foreach ($modulos as $k => $V) {
        echo "<li> $v";
    }
    echo "</ol>";
}
?>
</body>
</html>
```

Anexo IX.- Solución propuesta VI.

```
<!doctype html>
<?php
setlocale(LC_ALL, 'es_ES.UTF-8');
date_default_timezone_set('Europe/Madrid');
?>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no,
          initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Formulario Fecha</title>
    <style type="text/css">
        label{
            display: inline-block;
            width: 50px;
            height: 20px;
            font-weight: bold;
        }
        form input {
            height: 20px;
        }
    </style>
</head>
<body style="background: cadetblue">
<h3 style="text-align: center; font-weight: bold">Formulario Fecha</h3>
<?php
    if(isset($_POST['enviar'])){
        $dia=$_POST['dia'];
        $mes=$_POST['mes'];
        $anio=$_POST['anio'];
        if(checkdate($mes, $dia, $anio)){
            $fecha="$dia-$mes-$anio";
            $objFecha=date_create_from_format('d-m-Y', $fecha);
            $fecha1 = strftime("La fecha es %A, %d de %B de %Y", date_timestamp_get($objFe
            echo "<p style='font-weight: bold; color: blue'>$fecha1</p>";
        }
        else{
            echo "<p style='font-weight: bold; color: #FF2525'>La fecha Introducida es ERR
        }
        echo "<a href='".$_SERVER['PHP_SELF']."' style='text-decoration:none'>";
        echo "<button>Probar otra Fecha</button>";
        echo "</a>";
    }
    else{
?>
<fieldset style="width:40%; margin:auto">
<form name="fecha" method="POST" action="<?php echo $_SERVER['PHP_SELF']; ?>">
    <p>
```

```
        <label for="dia">Día:</label>
        <input type="number" step="1" max="31" min="1" id="dia" name="dia" required>
    </p>
    <p>
    <label for="mes">Mes:</label>
    <input type="number" step="1" max="12" min="1" name="mes" id="mes" required>
    </p>
    <p>
    <label for="anio">Año:</label>
    <input type="number" step="1" max="9999" min="1900" name="anio" id="anio" required>
    </p>
    <div style="text-align: center">
        <input type="submit" value="Enviar" name="enviar" style="margin-right: 5px; width:
        <input type="reset" value="Limpiar" style="width: 60px; height:30px; font-weight:
    </div>
</form>
</fieldset>
<?php } ?>
</body>
</html>
```

Anexo X.- Ejemplo Subir PDF.

```
!DOCTYPE html>
<html lang="en">
<?php
function isTipoOk($tipo){
    if($tipo=="application/pdf") return true;
    return false;
}
function comprobarError($i){
        switch ($i){
            case 1:
                echo "<p>El fichero subido excede la directiva upload_max_filesize de
                break;
            case 2:
                echo "<p>El fichero subido excede la directiva MAX_FILE_SIZE especifi
                break;
            case 3:
                echo "<p>El fichero fue sólo parcialmente subido.</p>";
                break;
            case 4:
                echo "<p>No se subió ningún fichero.</p>";
                break;
            case 6:
                echo "<p>No se pudo escribir en la carpeta temporal</p>";
                break;
            case 8:
                echo "<p>No se pudo escribir el fichero en el disco.</p>";

        }
}
?>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>R1FOR</title>
</head>
<body style="background-color:#1565c0;color:#ffffff;">
<?php
if (isset($_POST["enviar"])) {
    if (is_uploaded_file($_FILES["documento"]["tmp_name"])) {
        if (isTipoOk($_FILES["documento"]["type"])) {
            $nombre = $_FILES["documento"]["name"];
            move_uploaded_file($_FILES["documento"]["tmp_name"], "./documentos/" . $n
            echo "<p>Archivo subido correctamente</p>";
        } else {
            echo "Error, El tipo del archivo no es <b>pdf</b>";
        }
    } else {
        echo "<p>Han habido errores, estos han sido:</p>";
        comprobarError($_FILES["documento"]["error"]);

```

```
        }
    }else{
    ?>
    <h3 class="text-center">Subida de un fichero</h3>
    <fieldset style="width:40%; margin:auto;">
        <form name="fichero" action="<?php echo $_SERVER['PHP_SELF']; ?>" ENCTYPE="multip
            <!-- Establecemos el tamaño máximo del archivo a 50000 bytes -->
            <input type="hidden" name="MAX_FILE_SIZE" value="50000" />
            <label for="file" style="font-weight: bold">Elige el Fichero: </label> <
            <br><br>
            <button type="submit" value="Subir" name="enviar">Enviar</button>
            <button type="reset" value="reset">Limiar</button>
        </form>
    </fieldset>
    <?php } ?>
    </body>
    </html>
```