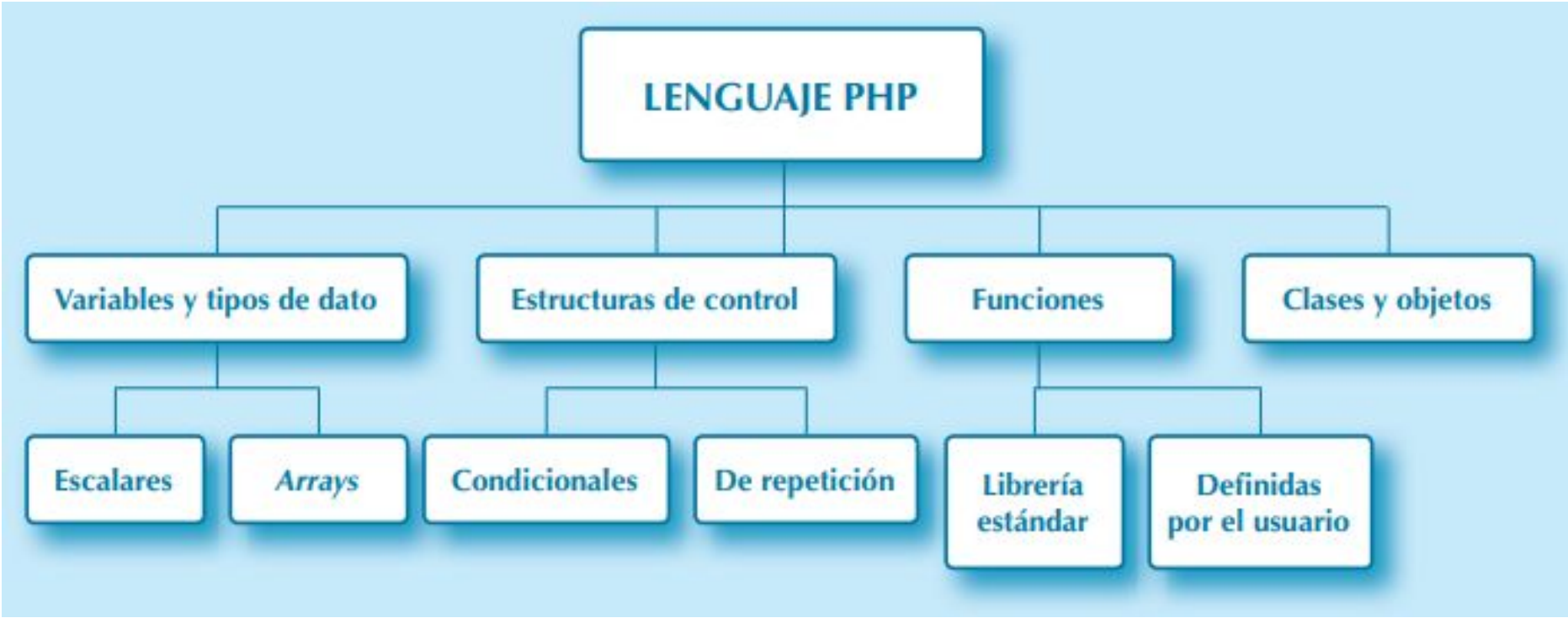
INTRODUCCIÓN A PHP

Luis Enrique González de la Fuente

CONCEPTOS BÁSICOS

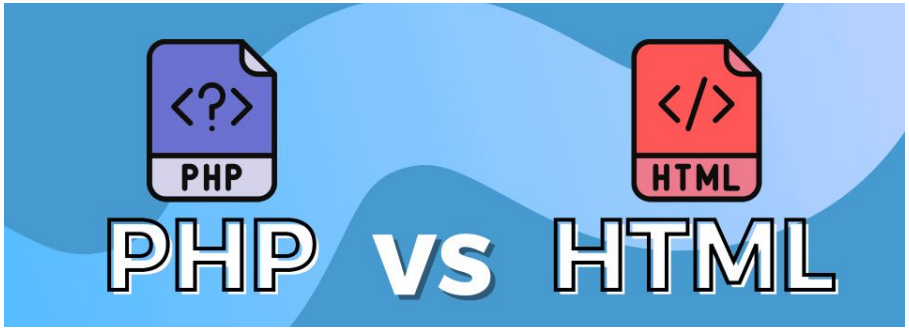
- Hola mundo.
- Comentarios.
- Variables.
- Tipos de datos.
- Constantes.
- Ámbito de variables.
- Variables predefinidas.


LENGUAJE PHP
Variables y tipos de dato
Estructuras de control
Funciones
Clases y objetos
Escalares
Arrays
Condicionales
De repetición
Librería estándar
Definidas por el usuario

PHP y HTML. Código incrustado

- PHP es el lenguaje de programación para desarrollo web en el lado del servidor. Desde su aparición en 1994 ha tenido gran aceptación y se puede decir que es el lenguaje más extendido para el desarrollo en el lado del servidor.
- Necesita de un **intérprete** de PHP instalado junto con el servidor web.
- Es flexible y permite programar pequeños **scripts** con rapidez.
- Comparado con Java requiere escribir menos código y, en general, resulta menos engorroso.
- La **sintaxis** de los elementos básicos es bastante parecida a Java.
- En el desarrollo web es muy habitual utilizar PHP incrustado dentro de ficheros HTML. El código PHP se introduce dentro del HTML utilizando la **etiqueta** <?php para abrir el bloque de PHP y la etiqueta ?> para cerrarlo.

"HOLA MUNDO"




```
<!DOCTYPE html>
<html>
    <head>
        <title>Hola mundo</title>
    </head>
    <body>
        <?php
            echo "Hola mundo";
        ?>
    </body>
</html>
```

"HOLA MUNDO"

1. Guarda como "holaMundo.html"
2. Guarda como "holaMundo.php"
3. Abrir en navegador.
4. Abrir en localhost.
5. Visualiza el html (inspector)
    a. F12
    b. Ctrl+u

SINTAXIS PHP

El elemento básico en PHP es el bloque. Un bloque PHP está delimitado por las etiquetas correspondientes y contiene una serie de sentencias separadas por punto y coma.

```
<?php
    sentencia1;
    ...
    sentenciaN;
? >
```

Cuando un fichero contiene solo PHP es posible no cerrar la etiqueta del último bloque. De hecho es recomendable no cerrarlo puesto que puede dar problemas si la respuesta del servidor involucra varios ficheros.

COMENTARIOS

Para mejorar la legibilidad del código y que se recomienda siempre como una buena práctica a la hora de programar, es la inclusión de comentarios explicativos intercalados en el código.

En PHP, podemos incluir los comentarios utilizando un estilo fácilmente reconocible en otros lenguajes de alto nivel para introducir comentarios en una línea o en varias.

```
// esto es un comentario de una línea

# esto es otra forma de comentario de una línea


/* esta es la forma de

escribir comentarios

de varias líneas */
```

VARIABLES

PHP es un lenguaje **NO FUERTEMENTE TIPADO** lo que significa que no hace falta indicar el tipo de dato al declarar una variable.

Cuando se les asigna un dato se crea la variable con el tipo de dato correspondiente a la inicialización.

Debemos tener mucho cuidado con ello puesto que a corto o medio plazo podemos dar lugar a problemas de difícil depuración.

```
// Los identificadores de las variables siempre van precedidos del símbolo $

$nombre = valor;

// Los identificadores como tal deben comenzar por letra o guión bajo (_) y sólo se permiten números, caracteres y guiones bajos.

$variableCorrecta = Verdadero;

$_otra_variable_correcta_1 = Verdadero;

$%variableNoCorrecta = Falso;
```

VARIABLES

En este ejemplo podemos ver cómo se declaran variables de distinto tipo sin indicar nada más allá del nombre y el valor inicial.

Tener en cuenta que se debe usar el "." en lugar de la ",".

Para las cadenas es necesario utilizar las comillas simples o dobles.

Para los valores boolenanos se pueden escribir en mayúsculas o minúsculas.

```
<?php

$entero = 8; // tipo integer

$flotante = 9.5; // tipo coma flotante

$cadena = "cadena"; // tipo cadena de caracteres

$bool = TRUE; //tipo booleano
```

TIPOS DE DATOS ESCALARES

En PHP podemos encontrar 4 tipos de datos básicos:

- Integer: números enteros.
- Float: números decimales.
- Boolean: valores verdadero (TRUE) y falso (FALSE).
- String: caracteres.

Además podemos encontrar otros tipos de datos más complejos que son admitidos nativamente por PHP.

OTROS TIPOS DE DATOS

Los otros tipos de datos complejos son:

- Array: colecciones de elementos.
- Object: para soportar la programación orientada a objetos.
- Callable: tipo especial para representar funciones de callback (funciones que se pasan a otras funciones)
- Null: para representar una variable no asignada. Tiene como valor NULL por lo que si asignamos dicho valor convertimos nuestra variable a tipo null.
- Resource: para representar recursos externos como una conexión a BD

Tipo	Descripción
integer	Números enteros
float	Números reales en coma flotante
string	Cadenas de caracteres
boolean	Booleanos, TRUE o FALSE
array	Colección de elementos identificados
object	Un objeto es una instancia de una clase
callable	Para las funciones de *callback*
null	Para representar variables no asignadas
resource	Representa recursos externos

TIPOS DE DATOS ESCALARES. ACLARACIONES

Cada plataforma PHP podría soportar distintos rangos de los datos básicos como por ejemplo el máximo o mínimo de los números enteros que pueden ser configurados (php.ini).

La conversión entre enteros y decimales es automática en caso de encontrarse PHP en la necesidad de hacerla y podemos obligarla anteponiendo (int) o (float) para realizar la conversión.

En las cadenas se pueden usar comillas simples o dobles, aunque sólo con las dobles las variables que vayan dentro de las comillas serán sustituidas por el valor correspondiente.

Se pueden usar los caracteres especiales "\n", "\r" y "\t" pero no tendrán validez en el HTML generado.

TIPOS DE DATOS ESCALARES. BOOLEANOS

En el caso de los valores booleanos debemos tener en cuenta que si se busca un booleano pero encontramos otro tipo de dato se transforma siguiendo éstas normas:

Tipo	Valor como *boolean*
integer	Si es 0 se toma FALSE, en otro caso como TRUE
float	Si es 0.0 se toma FALSE, en otro caso como TRUE
string	Si es una cadena vacía o "0", se toma como FALSE, en otro caso como TRUE
variables no inicializadas	FALSE
null	FALSE
array	Si no tiene elementos se toma FALSE, en otro caso como TRUE

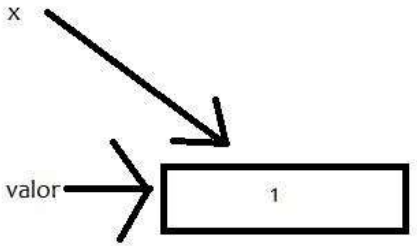
VARIABLES

En este ejemplo vamos a cambiar el tipo de una variable.

La salida del ejemplo confirma que la variable cambia de tipo de dato de integer a string.

```
<?php
$variable = 5; // entero
// imprime el tipo de dato de a
echo gettype($variable);
echo "<br>";
$variable = "Cambiando"; // cambia a cadena
// se comprueba que ha cambiado
echo gettype($variable);
```

VARIABLES. ASIGNACIÓN



Se pueden asignar las variables de dos maneras diferentes:

- Por **copia**: se crea una nueva variable con el mismo valor por lo que ocupan diferentes posiciones de memoria.
- Por **referencia**: no se crea una nueva variable sino un nombre que apunta a la misma posición de memoria.

```
<?php

$var1 = $var2; // Asignación por copia

$var1 = &$var2; // Asignación por referencia

// En el caso por referencia si se cambia var1 también lo hace var2 y al revés.
```

VARIABLES. NO INICIALIZACIÓN

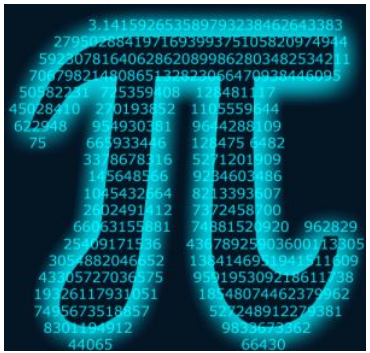
¿Qué ocurre con la variables que se utilizan antes de inicializarse?

Obtendremos un warning indicando que la variable no está definida pero nuestro programa seguirá ejecutándose.

Si se necesitase, PHP toma el valor por defecto para el tipo de dato necesitado y continua.

```
<?php

$n1 = 100;

$n3 = 100 + $n2; // como $n2 no existe, se toma 0

echo "$n3 <br>"; // muestra 100

$n3 = 100 * $n2; // $n2 no existe, se toma como 0

echo "$n3 <br>"; // muestra 0
```

CONSTANTES



En PHP para definir constantes tenemos que utilizar la función "define()" cuyos parámetros son:

- Nombre de la constante.
- Valor de la constante.

Por comodidad se suelen nombrar todo en mayúsculas.

Tener en cuenta que no hace falta el símbolo $.

```
<?php

define ("CONSTANTE", 100);

echo CONSTANTE; // muestra 100
```

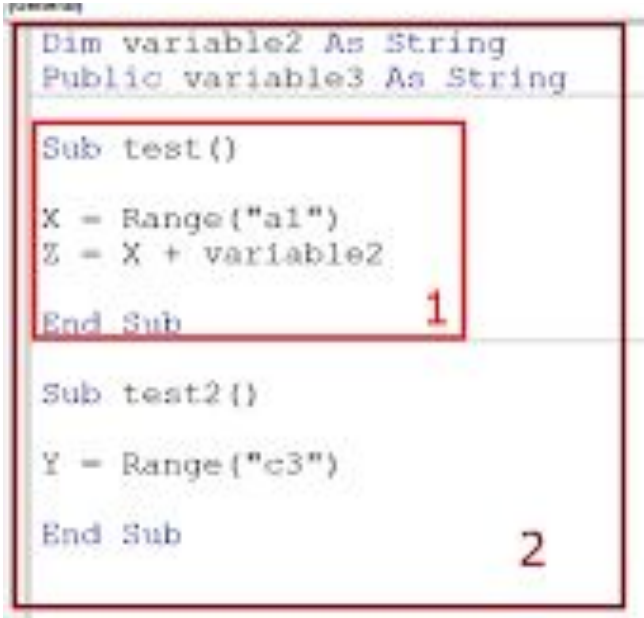
ÁMBITO DE LAS VARIABLES

El ámbito de una variable es la zona de código en la que se puede utilizar porque es visible.

Las variables declaradas en un fichero son visibles en dicho fichero y aquellos que incluyan dicho fichero.

En las funciones, las variables son locales y además dentro de la función sólo se podrán acceder a las variables locales y los argumentos de la función.

Se pueden definir variables globales ("global" o $_GLOBALS) para que sean accesibles desde cualquier función y fichero de la aplicación.



VARIABLES PREDEFINIDAS

En PHP hay una serie de variables disponibles que se pueden utilizar para obtener información sobre el servidor, los datos enviados, etc.

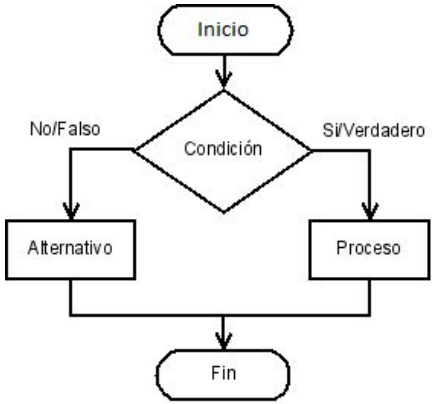
Un grupo de ellas son las llamadas superglobales que contienen información especialmente relevante.

Nombre	Descripción
$GLOBALS	Variables globales definidas en la aplicación
$_SERVER	Información sobre el servidor
$_GET	Parámetros enviados con el método GET (en la URL)
$_POST	Parámetros enviados con el método POST (formularios)
$_FILES	Ficheros subidos al servidor
$_COOKIE	*Cookies* enviadas por el cliente
$_SESSION	Información de sesión
$_REQUEST	Contiene la información de $_GET, $_POST y $_COOKIE
$_ENV	Variables de entorno

CONCEPTOS AVANZADOS

- Estructuras condicionales.
- Estructuras repetitivas.
- Uso de otros ficheros.
- Operadores
- Arrays
- Funciones
- Excepciones y errores
- Clases y objetos

ESTRUCTURAS CONDICIONALES



En PHP podemos encontrar las siguientes estructuras condicionales:

- if
- if-else
- if-elseif
- switch

```
<?php
    $var=5;
    if ($var<0)
        echo "Es menor";

    if($var>0)
        echo "Es mayor";
    else
        echo "Es menor";
```

ESTRUCTURAS CONDICIONALES

Hay que tener en cuenta que si queremos que se ejecute más de una opción debemos encerrar las instrucciones entre llaves.

```
if ($var<0){
        echo "Es menor";
        echo "Otra línea";
}
```

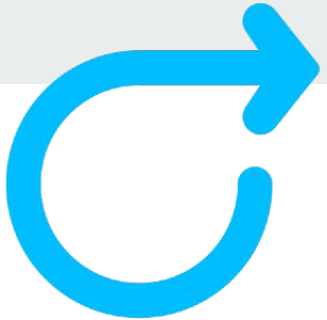
```
<?php
        $var=5;
        if ($var==0)
                echo "Es cero";
        elseif ($var<0)
                echo "Es menor que 0";
        elseif ($var>0)
                echo "Es mayor que 0";
        else
                echo "Qué será…";
```

ESTRUCTURAS CONDICIONALES

En la estructura switch tenemos una opción por defecto, default, que se ejecutará si no se ha cumplido ninguno de los casos. Además en PHP, como en Java, es necesario el uso del comando break para no continuar la ejecución de las siguientes opciones.

```
<?php
        $var=5;
        switch ($var){
                case 1:     echo "Es 1";
                            break;
                case 2:     echo "Es 2";
                            break;
                case 3:     echo "Es 3";
                            break;
                default:
                            echo "No es ni 1, ni 2 ni 3";
        }
```

ESTRUCTURAS REPETITIVAS



En PHP podemos encontrar las siguientes estructuras repetitivas:

- for
- while
- do-while
- foreach

```
<?php
        for ($i = 0; $i < 5; $i = $i + 1){
                echo "For $i <br>\n";
        }
        $i = 0;
        while (i < 5){
                echo "While $i <br>\n";
                $i = $i + 1;
        }
        $i = 0;
        do {
                echo "While $i <br>\n";
                $i = $i + 1;
        } while (i < 5);
```

ESTRUCTURAS REPETITIVAS

Como siempre en este tipo de estructuras cada una tiene sus particularidades siendo el bucle do-while utilizado cuando al menos se va a realizar una iteración.

Foreach se utiliza para una lista de valores por lo que se verá más adelante.

```php
<?php
    //No se ejecuta ninguna vez
    $i = 0;
    while ($i < 0){
        echo "While $i <br>\n";
        $i = $i + 1;
    }
    //Se ejecuta una vez
    $i = 0;
    do {
        echo "Do_While $i <br>\n";
        $i = $i + 1;
    } while ($i < 0);
```

ESTRUCTURAS REPETITIVAS

Se puede utilizar el comando "break;" para romper la ejecución del bucle en cualquier momento.

También se puede utilizar el comando "continue;" para que el bucle salte directamente a la siguiente iteración.

```
<?php
    //Sólo saldrán los pares hasta el 8
    $i = 0;
    while ($i < 10){
        if($i%2==1){
            //$i = $i + 1;
            continue;
        }
        if($i==8) break;
        echo "Iteración: $i <br>\n";
        $i = $i + 1;
    }
```

ESTRUCTURAS REPETITIVAS

El comando "break;" permite además salir de bucles anidados. Para ello se debe indicar como parámetro el número de niveles que deseamos romper.

En cualquier caso "break 1;" es lo mismo que "break;".

Si se utiliza en un "switch" obtenemos el mismo efecto.

```
<?php
    for ($i = 0; $i < 5; $i = $i + 1){
        for ($j = 0; $j < 5; $j = $j + 1){
            echo "Loop $i, $j<br>\n";
            if ($j==2)
                //break;
                //break 1;
                break 2;
        }
    }
```

USO DE OTROS FICHEROS

Hay 2 sentencias especiales para trabajar con otros ficheros php, "include" y "require".

Ambos ejecutan el fichero que indican, pero en el caso de "require" si no existe genera un error mientras que "include" sólo genera un warning.

```
<?php
        echo "Vamos a incluir un fichero.<br>\n";
        $a = "Variable del fichero original";
        include "fichero.php";
        //require "fichero.php";
        echo "Después de usar el fichero.<br>\n";
        echo "Variable b: $b<br>\n";
```

```
<?php
        echo "Dentro del fichero.<br>\n";
        echo "Variable a: $a<br>\n";
        $b = "Variable del otro fichero";
```

USO DE OTROS FICHEROS

Hay otras 2 sentencias especiales similares, "include_once" y "require_once" que sólo cargan el fichero si no se ha hecho con anterioridad.

```
<?php
        echo "Vamos a incluir un fichero.<br>\n";
        $a = "Variable del fichero original";
        include "fichero.php";
        include_once "fichero.php";
        echo "Después de usar el fichero.<br>\n";
        echo "Variable b: $b<br>\n";

<?php
        echo "Dentro del fichero.<br>\n";
        echo "Variable a: $a<br>\n";
        $b = "Variable del otro fichero";
```

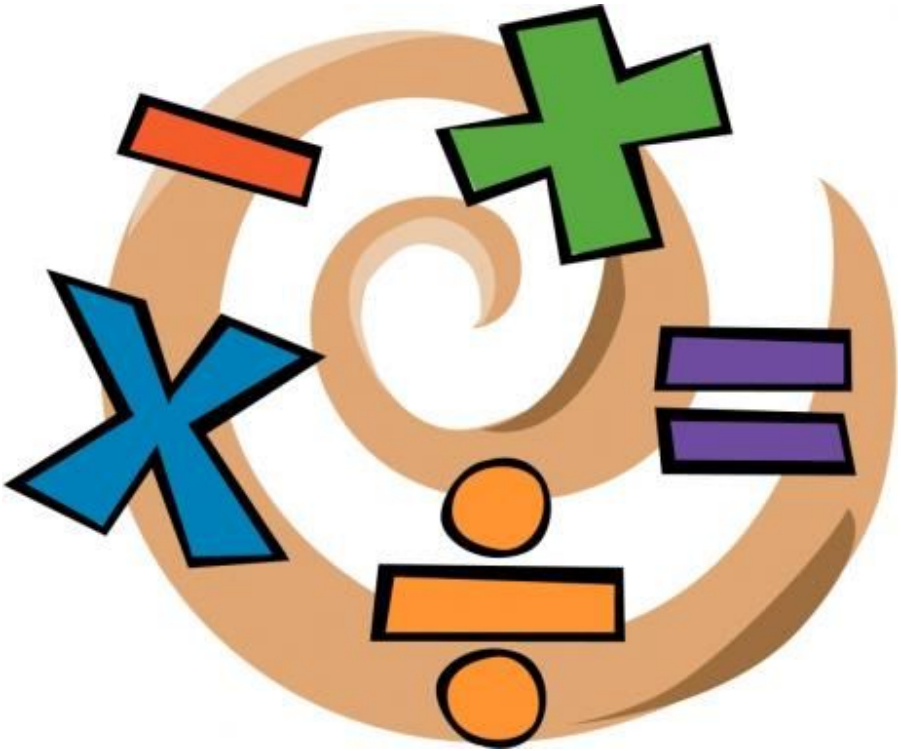
COMANDO RETURN

Por último podemos utilizar el comando return que hace las veces de retorno de las funciones (con un parámetro) pero que si es lanzado en medio de un fichero PHP lo que se consigue es salir de la ejecución de dicho fichero y si está incluido en otro entonces seguirá su ejecución.

```
<?php
        echo "Vamos a incluir un fichero.<br>\n";
        include "fichero.php";
        echo "Después de usar el fichero.<br>\n";

<?php
        echo "Dentro del fichero.<br>\n";
        return;
        echo "Dentro, después del return.<br>\n";
```

OPERADORES

En PHP podemos encontrar la inmensa mayoría de operadores habituales. En las páginas siguientes podremos ver los más habituales así como algunos nuevos.



OPERADORES

Operadores de comparación: == (igualdad), != o <> (diferente), >=, >, <=, <

En PHP podemos encontrar el operador idéntico, **===,** que devuelve True si ambos lados son del mismo tipo y además tienen el mismo valor mientras que == intenta convertir los datos al mismo tipo para poder comparar. En contrapartida podemos encontrar el operador !== (no idéntico).

También podemos encontrar el operador **??** que devolverá la primera expresión no nula empezando por la izquierda.

OPERADORES

En los siguientes ejemplos podemos ver cómo funcionan los nuevos comandos.

```
<?php
    if ("3"===3) echo "Son idénticos<br>\n";
    else echo "No son idénticos<br>\n";

    if ("3"==3) echo "Son iguales<br>\n";
    else echo "No son iguales<br>\n";

    $a=null;
    $b=2;
    $c=5;
    echo $a??$b??$c;
```

OPERADORES

Operadores de aritméticos: +, -, *, /, %.

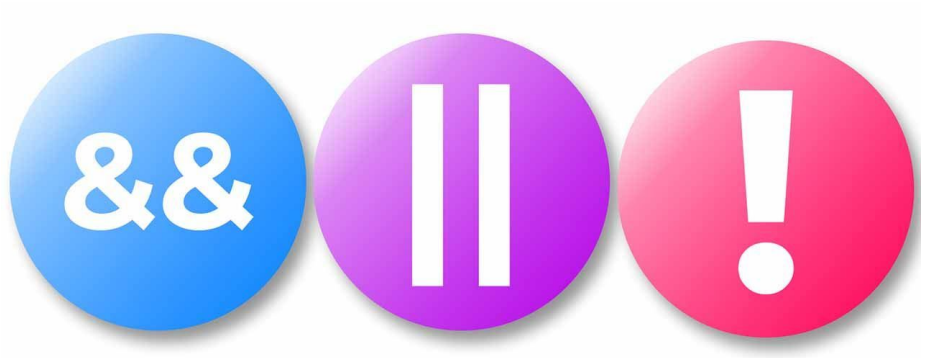
En este caso podemos encontrar los operadores + y - como operadores unarios (situados delante de una expresión) para conseguir en el primer caso hacer una conversión a número (integer o float según corresponda) y en el segundo caso cambiar de signo la expresión.

Además podemos realizar una potencia con la forma 2 ** 3.

OPERADORES

Operadores lógicos: and o &&, or o || y ! (not).

Existe la posibilidad de utilizar la operación **xor** o OR exclusivo en la que si una y sólo una de las expresiones es verdadera el resultado será verdadero.



OPERADORES

Operadores a nivel de bit: & (and), | (or), ^ (xor) y ~ (not).

Se pueden desplazar los bits de una variable tanto a la izquierda como a la derecha en un número determinado con los comandos:

- $var >> N
- $var << N

Operadores de asignación: = (valor), =& (referencia), +=, -=, *=, /=.

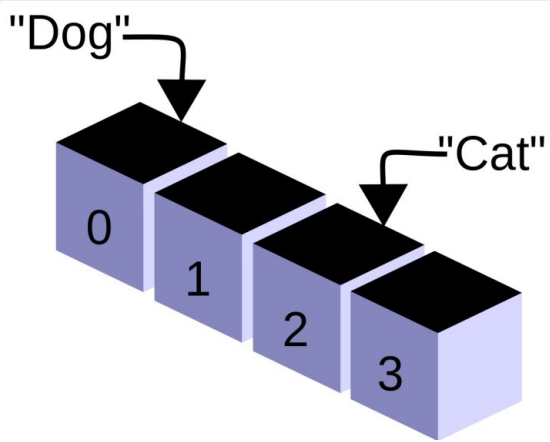
Otros operadores: ++, --, y el "." que sirve para concatenar cadenas.

ARRAYS

Un array en PHP comprende los conceptos de muchas estructuras vistas en otros lenguajes de programación como diccionarios, listas o vectores.


```php
<?php
    //Uso de la función array() para declararlos
    $varArray = array(
        clave1 => valor1,
        clave2 => valor2,
        ...
        claveN => valorN
    );
    //Uso de corchetes para declararlos
    $varArray2 =[
        clave1 => valor1,
        clave2 => valor2,
        ...
        claveN => valorN
    ];
```

ARRAYS



Para acceder a las posiciones de un array se debe utilizar su palabra clave entre corchetes:

```
$varArray["clave1"]
```

Podemos utilizar la función "print_r" para visualizar un array al completo:

```
print_r($varArray);
```

```
<?php
    $varArray = array(
      "clave1" => 1,
      "clave2" => 2,
          "claveN" => 3
    );
    print_r($varArray);
```

ARRAYS. RECORRIDOS

Para recorrer un array se suele utilizar el bucle **foreach**.

Con este bucle se realiza una iteración por cada elemento de los valores que encontremos.

```
foreach ($array as $valor){
    ...
}
```

```
<?php
    $varArray = array(
        "clave1" => 1,
        "clave2" => 2,
        "claveN" => 3
    );
    print_r($varArray);
    foreach($varArray as $valor){
        echo "$valor <br>\n";
    }
```

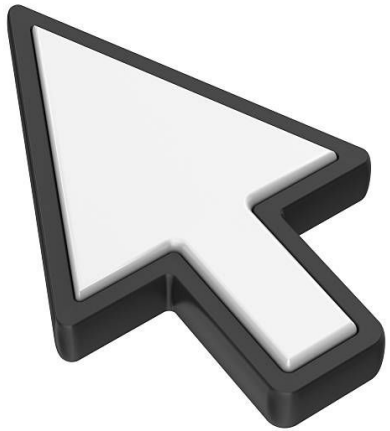
ARRAYS. RECORRIDOS

Si queremos disponer de los valores de las claves también podemos usar el bucle **foreach**.

```
foreach ($array
        as $clave => $valor){
            ...
}
```

```
<?php
    $varArray = array(
        "clave1" => 1,
        "clave2" => 2,
        "claveN" => 3
    );
    print_r($varArray);
    foreach($varArray as $clave => $valor){
        echo "$clave: $valor <br>\n";
    }
```

ARRAYS. RECORRIDOS

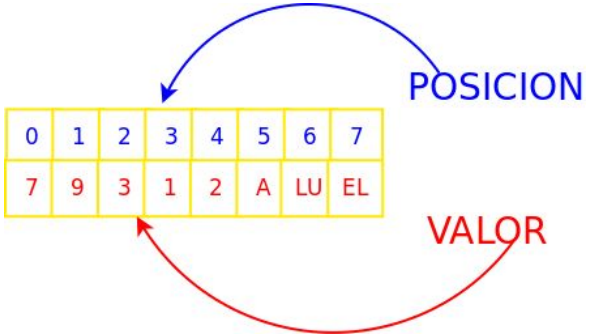


Si queremos modificar las posiciones reales del array durante el recorrido tendremos que asignar los valores por referencia al declarar el **foreach**.

```
foreach ($array as &$valor){

        ...

}
```

```
<?php
    $varArray = array(
        "clave1" => 1,
        "clave2" => 2,
        "claveN" => 3
    );
    print_r($varArray);
    foreach($varArray as &$valor){
        $valor = $valor+5;
    }
    print_r($varArray);
```

ARRAYS. DEFINICIÓN POR DEFECTO

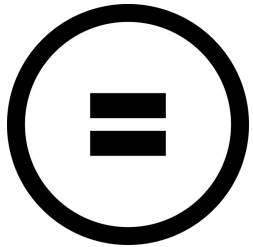


Si al generar un array no se indican claves, el array toma valores a partir del 0 consecutivamente.

También se podrán añadir nuevos valores sin indicar claves, y tomarán el siguiente valor del mayor utilizado.

```
<?php
    $varArray = array(5, 10, 15, 20);
    print_r($varArray);
    $varArray[]=25;
    print_r($varArray);
    $varArray[9]=30;
    print_r($varArray);
    $varArray[]=35;
    print_r($varArray);
```

ARRAYS. IGUALDADES



Se puede comparar 2 arrays con el operador identidad, ===, en cuyo caso devolverá True si todas las claves y valores son iguales y además están en el mismo orden.

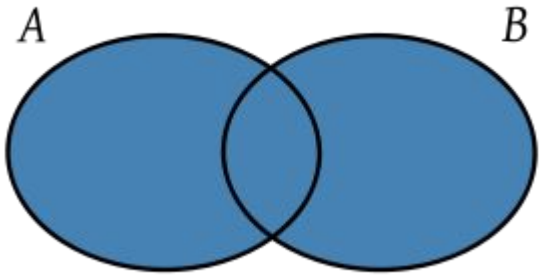
Con el operador igualdad, ==, no importa el orden.

```
<?php
    $varArray = array(1=>5, 2=>10, 3=>15, 4=>20);
    $varArray2 = array(1=>5, 2=>10, 4=>20, 3=>15);
    if($varArray===$varArray2)
        echo "1 y 2 Son idénticos <br>\n";
    if($varArray==$varArray2)
        echo "1 y 2 Son iguales <br>\n";
    $varArray3 = array(1=>5, 2=>10, 4=>20, 3=>15);
    if($varArray3===$varArray2)
        echo "2 y 3 Son idénticos <br>\n";
    if($varArray3==$varArray2)
        echo "2 y 3 Son iguales <br>\n";
```

ARRAYS. UNIONES

Se pueden juntar todos los elementos de 2 arrays en uno solo mediante el operador +.

En dicha unión no aparecerán las claves repetidas.

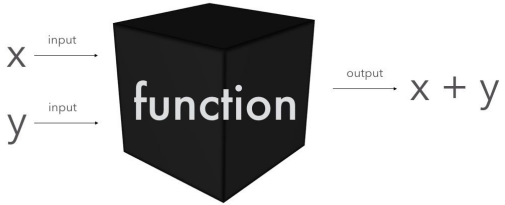



```
<?php
    $varArray = array(1=>5, 2=>10, 3=>15);
    $varArray2 = array(3=>20, 4=>25);
    $sumaArray = $varArray + $varArray2;
    print_r($sumaArray);
```

ARRAYS. RESUMEN DE OPERACIONES

Ejemplo	Nombre	Resultado
$a + $b	Unión	Unión de *$a* y *$b*.
$a == $b	Igualdad	TRUE si *$a* i *$b* tienen las mismas parejas clave/valor.
$a === $b	Identidad	TRUE si *$a* y *$b* tienen las mismas parejas clave/valor en el mismo orden y de los mismos tipos.
$a != $b	Desigualdad	TRUE si *$a* no es igual a *$b*.
$a <> $b	Desigualdad	TRUE si *$a* no es igual a *$b*.
$a !== $b	No-identidad	TRUE si *$a* no es idéntica a *$b*.

FUNCIONES



El concepto de función en PHP es igual que en el resto de lenguajes habituales, un grupo de instrucciones encapsuladas que son utilizadas para realizar una tarea concreta.

Pueden recibir argumentos de entrada y devolver un valor o no devolverlo (nulo).

En PHP no es necesario declarar el tipo de dato que devuelve una función y de hecho podría devolver distintos tipos según la ejecución realizada.

FUNCIONES PREDEFINIDAS

En PHP existen multitud de funciones predefinidas que siempre están accesibles para poder ser utilizadas con distinta finalidad:

- De variables: funciones que permiten conocer o modificar el estado o el tipo de una variable.
- De cadenas: funciones para conocer la longitud, comparar, dividir o juntar cadenas.
- De arrays: para ordenar, contar y buscar dentro de los elementos de un array.

En las páginas siguientes podemos ver algunas de ellas, aunque no son las únicas.

FUNCIONES PREDEFINIDAS DE VARIABLES

- **is_null($var)**: si la variable es NULL devuelve verdadero.
- **isset($var)**: si la variable ha sido inicializada y no es NULL devuelve verdadero.
- **unset($var)**: elimina la variable como si no se hubiera usado.
- **empty($var)**:si la variable no ha sido inicializada o es FALSE devuelve verdadero.
- **is_int($var), is_float($var), is_bool($var), is_array($var)**: devuelve verdadero en el caso correspondiente.
- **intval($var), floatval($var), boolval($var), strval($var)**: obtiene el valor de $var en el tipo de dato correspondiente.
- **print_r($var), var_dump($var)**: devuelven información de $var.

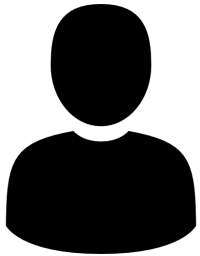
FUNCIONES PREDEFINIDAS DE CADENAS

- **strlen($cad)**: devuelve la longitud de $cad.
- **explode($separador, $cad)**: divide la cadena $cad utilizando $separador y devuelve un array de cadenas.
- **implode($separador, $array)**: devuelve una única cadena juntando todas las cadenas de $array introduciendo $separador entre cadena y cadena.
- **strcmp($cad1, $cad2)**: compara ambas cadenas devolviendo 0 si son iguales, -1 si $cad1 es menor y 1 si $cad2 es menor.
- **strtolower($cad), strtoupper($cad)**: devuelven la cadena $cad convertida toda en minúsculas o en mayúsculas.
- **strstr($cad1, $cad2)**: busca la primera ocurrencia de $cad2 dentro de la cadena $cad1 y devuelve $cad1 desde donde comienza la ocurrencia. En caso de no aparecer devuelve falso.

FUNCIONES PREDEFINIDAS DE ARRAYS

- **ksort($array), krsort($array)**: ordena el contenido del array según las **claves** en orden ascendente o descendente (reverse).
- **sort($array), rsort($array)**: ordena el contenido del array según los **valores** en orden ascendente o descendente (reverse). Mucho cuidado con las claves que se pierden.
- **asort($array), arsort($array)**: ordena el contenido del array según los **valores** en orden ascendente o descendente (reverse) manteniendo claves
- **array_values($array)**: devuelve un array con los valores.
- **array_keys($array)**: devuelve un array con las claves.
- **array_key_exists($clave, $array)**: devuelve verdadero si dentro del array encontramos la clave.
- **count($array)**: devuelve el número de elementos del array.

FUNCIONES DEFINIDAS POR EL USUARIO

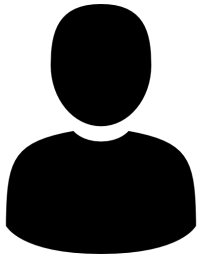


Si no fuese suficiente con las funciones que vienen predefinidas en PHP, es posible que el usuario defina las suyas propias.

```
function nombre(argumentos){

    instrucciones

}
```

```
<?php
    //Para declararla se escriben entre paréntesis
    // los diferentes parámetros
    function suma($a, $b){
        //Aquí dentro $a y $b son locales
        return $a+$b;
        //Con return se devuelve la salida
        //El tipo de la salida no es relevante
    }

    //Para utilizarla
    $c = suma($a, $b);
```

FUNCIONES DEFINIDAS POR EL USUARIO



Los argumentos pueden tener valores por defecto, en cuyo caso se tienen que indicar en la declaración de la función con un

($var = "valor")

```php
<?php
    function suma($a=1, $b=2){
        //Si no se indica $a, toma el valor 1
        //Si no se indica $a, toma el valor 2
        return $a+$b;
    }

    //Para utilizarla podría omitirse alguno
    $c = suma(2);
    $c = suma(6);
    $c = suma();
```

FUNCIONES. PASO POR VALOR Y POR REFERENCIA

En las funciones de PHP el paso de argumentos por defecto es por valor (una copia de la variable que se pasa como entrada) pero es posible pasar las variables por referencia si así se declara (con &) y así poder modificar la variable original.

```php
<?php
    function incrementar(&$a){
        //Al usar &, la variable es una referencia
        // a la variable original
        $a++;
        //En este caso no hace falta hacer return
    }

    //Para utilizarla no hay cambios
    $var=5;
    incrementar($var);
    echo $var;
```

FUNCIONES. FUNCIONES ARGUMENTOS

Una particularidad de PHP es que se pueden pasar funciones como argumentos de otras funciones.

Para ello se pasa el nombre de la función entre comillas y en la función que la recibe se podrá usar siempre con los argumentos adecuados.

```
<?php
    function fLlamante($argumentoFuncion){
        // Al hacer uso de la función realmente lo
        // que se hace es llamar a la otra función
        $argumentoFuncion();
    }
    // La función debe estar declarada
    function otraFuncion(){
        echo "HOLA<br>\n";
    }
    // Al llamarla debemos utilizar una cadena
    fLlamante("otraFuncion");
```

EXCEPCIONES Y ERRORES

Como es habitual en los lenguajes de programación, en PHP hay un sistema básico de control de errores en el que se generan errores representados por números.

Los errores que se generan en PHP pueden ser controlados a través de una serie de directivas en php.ini.

Además desde la versión 5 hay un sistema de excepciones parecido a Java (try, catch, finally).

EXCEPCIONES Y ERRORES

configurar php.ini

Las directivas que podemos modificar en php.ini son:

- error_reporting: indica qué errores deben reportarse. Lo normal es usar E_ALL, es decir, todos. (Página siguiente)
- display_error: indica que se muestren los mensajes de error en la salida del script.
- log_errors: indica si se deben almacenar los mensajes de error en un fichero de log.
- error_log: para indicar la ruta del fichero de log.

ERRORES (PARA INDICAR EN error_reporting)

Valor	Constante	Descripción
1	E_ERROR	Errores Fatales no recuperables en tiempo de ejecución.
2	E_WARNING	Advertencias en tiempo de ejecución (errores no fatales).
4	E_PARSE	Errores de análisis de sintaxis.
8	E_NOTICE	Avisos en tiempo de ejecución. Podría indicar un error.
16	E_CORE_ERROR	Errores fatales que ocurren durante el arranque incial de PHP.
32	E_CORE_WARNING	Advertencias (errores no fatales) en el arranque inicial de PHP.
64	E_COMPILE_ERROR	Errores fatales en tiempo de compilación.
128	E_COMPILE_WARNING	Advertencias en tiempo de compilación.
256	E_USER_ERROR	Mensaje de error generado por el usuario con trigger_error().
512	E_USER_WARNING	Mensaje de advertencia generado por el usuario con trigger_error().
1024	E_USER_NOTICE	Mensaje de aviso generado por el usuario con trigger_error().
2048	E_STRICT	Se habilita para que PHP sugiera cambios en el código.
4096	E_RECOVERABLE_ERROR	Error fatal capturable (set_error_handler()).
8192	E_DEPRECATED	Avisos en tiempo de ejecución de código obsoleto.
16384	E_USER_DEPRECATED	Mensajes de advertencia generados por el usuario con trigger_error().
32767	E_ALL	Todos los errores y advertencias soportados.

ERRORES

La función "error_reporting()" se puede utilizar para cambiar el valor de la directiva error_reporting en tiempo de ejecución.

Además se puede utilizar la función "set_error_handler()" para que se encargue de los errores.

```
<?php
        error_reporting(E_ALL); //Todos los errores
        ini_set("error_reporting", E_ALL);
                //Otra forma para error_reporting(E_ALL);
        error_reporting(0); //Se dejan de reportar errores
        error_reporting(E_ERROR | E_WARNING); // OR
        error_reporting(E_ALL & ~E_NOTICE);
                // Todos menos los E_NOTICE

        // Función para manejar los errores
        //Tiene unos parámetros obligatorios
        function manejaError($errno, $str, $file, $line){
                echo "Error $errno, $str<br>\n";
        }
        set_error_handler("manejaError");
        $a=$b; //generaría un error
```

EXCEPCIONES

Al igual que en Java, podemos lanzar una excepción con los bloques try/catch:

```
try{
        Instrucciones a controlar
}catch(Exception e){
        Instrucciones si hay excepción
}finally{
        Instrucciones haya o no excepción
}
```

```
<?php
        function generaExcepcion($a){
                if($a==0){
                        throw new Exception ("Excepción por 0");
                }
                return "No es 0";
        }
        try{
                $res = generaExcepcion(0); //Generará excepción
                //$res = generaExcepcion(5); //No la genera
                echo "Salida si no hay excepción: $res<br>\n";
        }catch(Exception $e){
                echo "Excepción generada:",
                        $e->getMessage(),"<br>\n";
        }finally{
                echo "Instrucciones que siempre se ejecutan<br>\n";
        }
```

EXCEPCIONES DE TIPO ERROR

En PHP podemos encontrar un tipo diferente de excepción, un Error que debe ser capturado de distinta manera:

```
try{
        ...
}catch(Error $e){
        ...
}
```

https://www.php.net/manual/es/class.error.php

Una manera de capturar tanto las excepciones como los errores es usar la clase Throwable de la que heredan ambas:

```
try{
        ...
}catch(Throwable $t){
        ...
}
```

https://www.php.net/manual/es/class.throwable.php

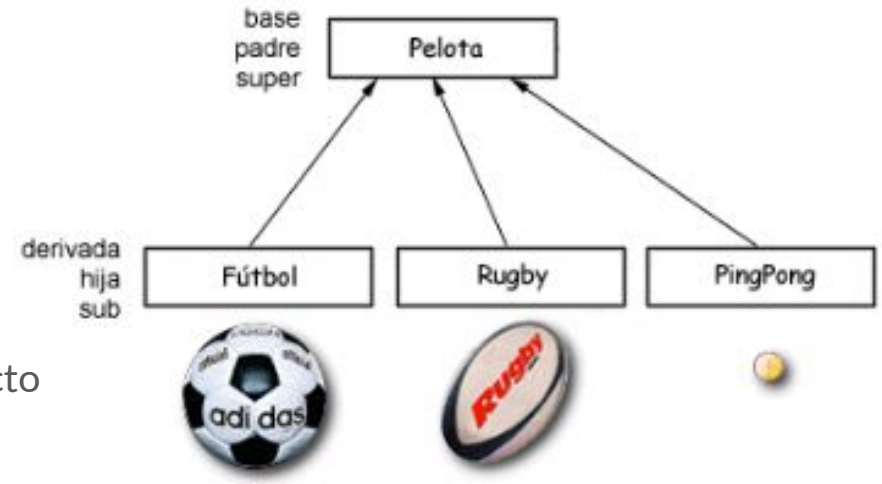
CLASES Y OBJETOS

PHP es un lenguaje orientado a objetos.

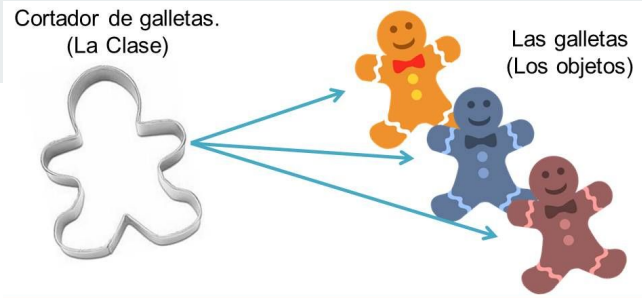
Permite definir clases, herencia, interfaces, etc.

Para ello hacemos:

```
class Pelota{
        private $atributo1 = 1; //con valor por defecto
        private $atributo2; //sin valor por defecto
        private static $atributo3=0; /estático
        …
}
```



CLASES Y OBJETOS

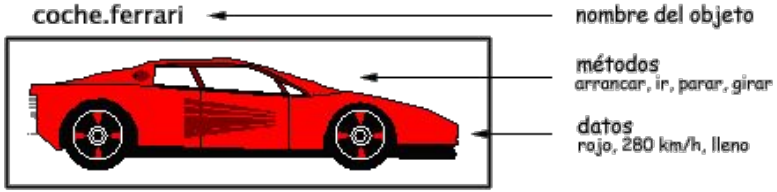


Los conceptos son los mismos que en Java e incluso la mayoría de palabras reservadas se repiten (public, private, protected, static...), aunque hay algunos diferentes:

- "__construct()": existen algunos métodos que comienzan con doble guión bajo que se refieren a "métodos mágicos" que tienen tareas concretas como los constructores.
- "__toString()": método para convertir la información de una objeto en un string.

Para crear un objeto nuevo de una clase se utiliza "$obj = new Clase();"

CLASES Y OBJETOS



A la hora de declarar los atributos suele usarse el modificador "private" mientras que para los métodos se suele usar "public". Para su acceso se utilizan:

```
$obj->atributo;

$obj->método(argumentos);
```

Desde un método (si no es estático) se puede utilizar la variable $this para referirse al propio objeto desde el que se invoca al método.

CLASES Y OBJETOS. STATIC

En este ejemplo podemos ver cómo definir una clase con un método estático y uno no estático.

Para llamar al método estático la sintaxis cambia a "::" en lugar de "->".

```php
<?php
	class ClaseMetodoEstatico{
		public static function metodoEstatico(){
			echo "Esto es un método estático\n";
		}
		public function metodoNoEstatico(){
			echo "Esto es un método NO estático\n";
		}
	}
	ClaseMetodoEstatico::metodoEstatico();
	//ClaseMetodoEstatico::metodoNoEstatico();
	$obj=new ClaseMetodoEstatico();
	//$obj::metodoNoEstatico();
	$obj->metodoNoEstatico();
	$obj::metodoEstatico();
```

CLASES Y OBJETOS. EJEMPLO

Este ejemplo completo genera clase de persona y trabajador, y luego se crea una persona, se modifica y luego se crea un trabajador. En todos los casos se muestra el contenido del objeto creado.

```
// crear una persona
$per = new Persona("1111111A", "Ana");
// mostrarla, usa el método __toString()
echo $per . "<br>\n";
// cambiar el nombre
$per->setNombre("María");
// volver a mostrar
echo $per . "<br>\n";
// crea un trabajador
$tra = new Trabajador("22222245A", "Pedro", 100);
// y lo muestra
echo $tra . "<br>\n";
```

CLASES Y OBJETOS. EJEMPLO

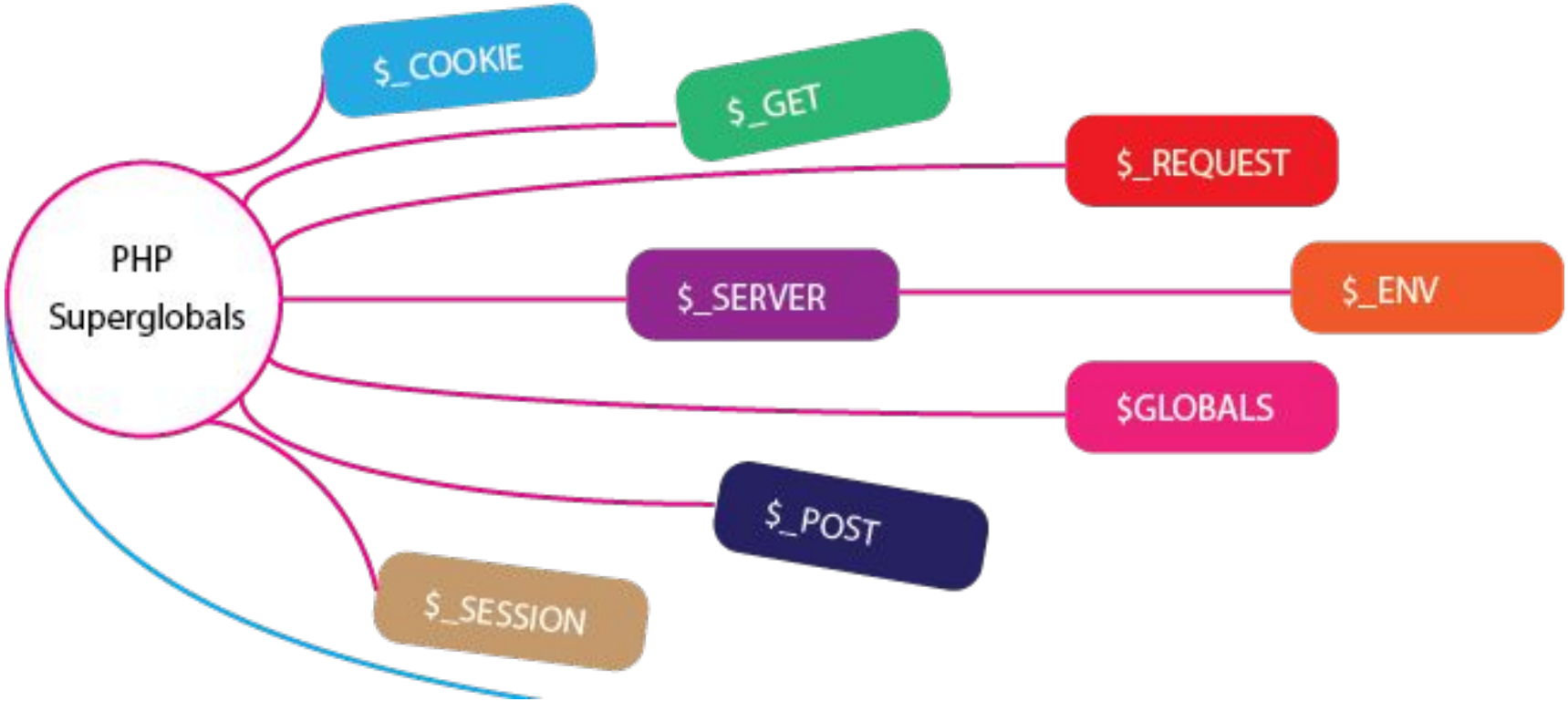

```php
<?php
    class Persona {
        private $DNI, $nombre;
        function __construct($DNI, $nombre) {
            $this->DNI = $DNI;
            $this->nombre = $nombre;
        }
        public function getNombre() {
            return $this->nombre;
        }
        public function setNombre($nombre) {
            $this->nombre = $nombre;
        }
        public function __toString() {
            return "Persona: " . $this->nombre . " ". $this->DNI;
        }
    }
```

```php
class Trabajador extends Persona{
    private $sueldo = 0;
    function __construct($DNI, $nombre, $sueldo){
        parent::__construct($DNI, $nombre);
        $this->$sueldo = $sueldo;
    }
    public function getSueldo(){
        return $this->sueldo;
    }
    public function setSueldo($sueldo){
        $this->sueldo = $sueldo;
    }
    public function __toString(){
        return  "Trabajador: ". $this->getNombre() ;
    }
}
```

CONCEPTOS “NUEVOS”

- Variables predefinidas.
- Métodos mágicos.
- Envío de parámetros a través de la URL.

VARIABLES PREDEFINIDAS: print_r($GLOBALS)



MÉTODOS MÁGICOS

Los métodos mágicos son aquellos métodos especiales que se pueden usar para modificar las acciones por defecto cuando ocurren ciertos comportamientos de un objeto.

https://www.php.net/manual/es/language.oop5.magic.php

Son éstos:

- **__construct()**
- __destruct()
- __call()
- __callStatic()
- __get()
- __set()
- __isset()
- __unset()
- __sleep()
- __wakeup()

- __serialize()
- __unserialize()
- **__toString()**
- __invoke()
- __set_state()
- __clone()
- __debugInfo().

ENVÍO DE PARÁMETROS POR LA URL

La forma más sencilla de pasar parámetros a nuestra página PHP es a través de la URL añadiendo distintas opciones a nuestra petición.

Se pueden pasar varios valores uniéndolos con "&".

abc.com/web.php**?v1=1&v2=2**

```
//En el HTML
<a href='pagina.php?opcion=valor'>
        Mensaje</a><br/>
//En el PHP
<?php
        $var=$_GET["opcion"];
        echo $var;
```

EJERCICIOS

1.1. Realiza el “Hola Mundo”.

1.2. Realiza un programa en PHP que defina las variables nombre y apellidos con tus datos y las concatene en una línea. Después se deben definir 2 variables con 2 números y hacer las 5 operaciones, +,-,*, /, % y mostrarlo.

1.3. Condicionales:

- Realiza un programa en el que se encuentren 6 variables diferentes (al menos la primera booleano) y las escribas todas juntas en una línea y en distintas líneas dependiendo del valor booleano de la primera.
- Escribe en PHP un programa que según una variable numérica del 1 al 7 devuelva el nombre del día de la semana correspondiente con las 4 estructuras por separado, if, if-else, if-elseif y switch. ¿Qué diferencia hay? ¿Cuál es mejor?
- Realiza un programa que resuelva una ecuación de segundo grado (2 soluciones, 1 solución o incluso no hay solución)

EJERCICIOS

1.4. Bucles:

- Escribe un programa que calcule el factorial de un número.
- Escribe un programa que indique todas las áreas de un rectángulo para una base de 1 a 8 y una altura de 2 a 5.

1.5. Ficheros (include, require)

- Escribe un fichero PHP para ser incluido en posteriores usos. Dicho fichero debe declarar una variable "$nombre" que almacene tu nombre y realizar un echo "Dentro del fichero saludamos a $nombre".
- Realiza un fichero PHP que incluya al otro mediante include 3 veces. Antes de las incluisones debe haber un echo "Antes de las inclusiones" y justo después debe sacar otro mensaje "Después de las inclusiones".
- Ahora el fichero debe incluir al primero 2 veces pero una vez con include y otra con include_once.
- Utiliza el comando require para realizar 3 inclusiones de nuevo.
- Realiza ahora 2 inclusiones, una con require y la otra con require_once.
- Ahora utiliza primero include y luego require pero con un nombre erróneo a drede.

EJERCICIOS

1.6. Operadores:

- Realiza un fichero PHP que utilice el operador potencia (**) para calcular y mostrar la tabla de las potencias de 2 hasta 2 elevado a 10.
- Realiza un fichero PHP que utilice las comparaciones lógicas para elegir datos que sean mayores que 10 pero distintos de 15 y de 25.
- Realiza un fichero PHP que utilice los operadores de desplazamiento de bits con una variable que tenga de valor 10, ¿Qué ocurre al desplazar 1 y 2 bits a cada lado?
- Utiliza la asignación por referencia para modificar el valor de una variable que originalmente valga 10 añadiéndole 5. Muestra el contenido de dicha variable antes y después de la nueva asignación.

1.7. Arrays:

- Realiza el ejercicio de la ecuación de segundo grado pero almacena los resultados en un array.
- Realiza un programa que tenga un array de enteros, reciba un número y se genere un nuevo array con los valores que están por encima.
- Sabiendo que las cadenas de caracteres pueden ser accedidas como si fueran un array, cuenta el número de "a" que hay en una cadena cualquiera de texto.

EJERCICIOS

1.8. Funciones:

- Realiza un programa que utilice todas las funciones predefinidas de variables, otro con las funciones predefinidas de cadenas y otro con las funciones predefinidas de arrays. Define para ello las variables necesarias.
- Crea una función que reciba 2 parámetros y ejecute la potencia de ellos.
- Crea una función que reciba la base y la altura de un rectángulo y devuelva el área.
- Crea una función que reciba la base y la altura de un triángulo y devuelva el área.
- Crea una función que reciba los valores de una ecuación de 2º grado y devuelva un array con las soluciones.
- Crea una función que reciba 2 parámetros (en caso de no recibirlos por defecto deben ser 2) y devuelva el producto de ellos.
- Crea las funciones matemáticas de suma, resta, división y multiplicación y guárdalas en un fichero mate.php. Luego incluye dicho fichero en otro programa que haga uso de ellas.
- Realiza una función que devuelva la letra del DNI pasado por argumento.
- Realiza una función que diga si una cadena pasada como argumento es palíndromo (capicúa).
- Realiza una función que reciba un argumento array y otro argumento valor y devuelva un array con los valores por encima de dicho valor que estuvieran en el array.

EJERCICIOS

1.9. Excepciones:

- Realiza una función "dividir" que emita una excepción en caso de que se reciba un parámetro para dividir entre 0. También realiza el programa que use esa función y reciba la excepción correspondiente.
- Realiza una función "factorial" que emita una excepción en caso de que se reciba un parámetro negativo. También realiza el programa que use esa función y reciba la excepción correspondiente.

1.10. Clases y objetos. Realiza la implementación de las siguientes clases: Coche genérico, deportivo, familiar y furgoneta. En dichas clases realiza las siguientes acciones:

- Implementa los atributos comunes en coche (por ejemplo matrícula, color, plazas, peso, longitud o ruedas) y en cada uno de los demás los atributos específicos (CV, numPuertgas, sillitaIsofix, capacidadCarga o velocidadMaxima)
- Implementa los métodos get y set correspondientes.
- Implementa además un método estático para incrementar una variable de nº de coches estática en todos los constructores.
- Realiza un programa que lo utilice y muestre convenientemente.

PROYECTO PHP BÁSICO

Con un menú HTML predefinido, debéis generar una serie de páginas PHP (menú que si recibe un parámetro lo indica en la propia página y programa que ejecute las siguientes opciones) que reciban los parámetros necesarios y ejecuten las siguientes utilidades:

1. Login de usuario correcto y si eso saludar HOLA MUNDO.
2. Login de usuario incorrecto.
3. Crear array con parámetros, recorrer, borrar y mostrar.
4. Ordenar un array con los datos de nombre de 5 personas.
5. Función que devuelva área de rectángulo (con datos introducidos en código).
6. Página que pruebe todas las funciones predefinidas de la presentación (isset, isnull, …, strlen…). Las de array ya se usan en el apartado 4.
7. Mostrar variables superglobales.
8. Hacer un control excepciones con una llamada a función que falle y otra que no.
9. Crear, utilizar y mostrar clases referentes a animales, mamíferos y aves.

En la página de menú se escribirá de donde se viene (se pasa como parámetro) o nada, y luego el menú. Desde todas las opciones podremos volver al menú inicial. Todo con comentarios explicativos.