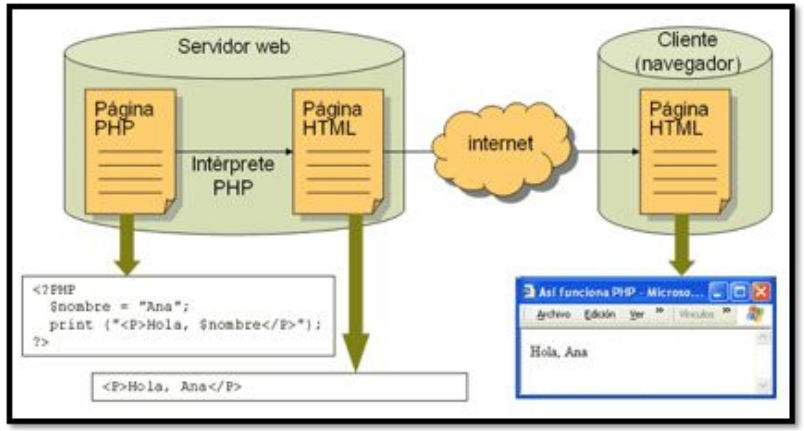
PROGRAMACIÓN BÁSICA CON PHP

UD3 - IMPLANTACIÓN DE APLICACIONES WEB

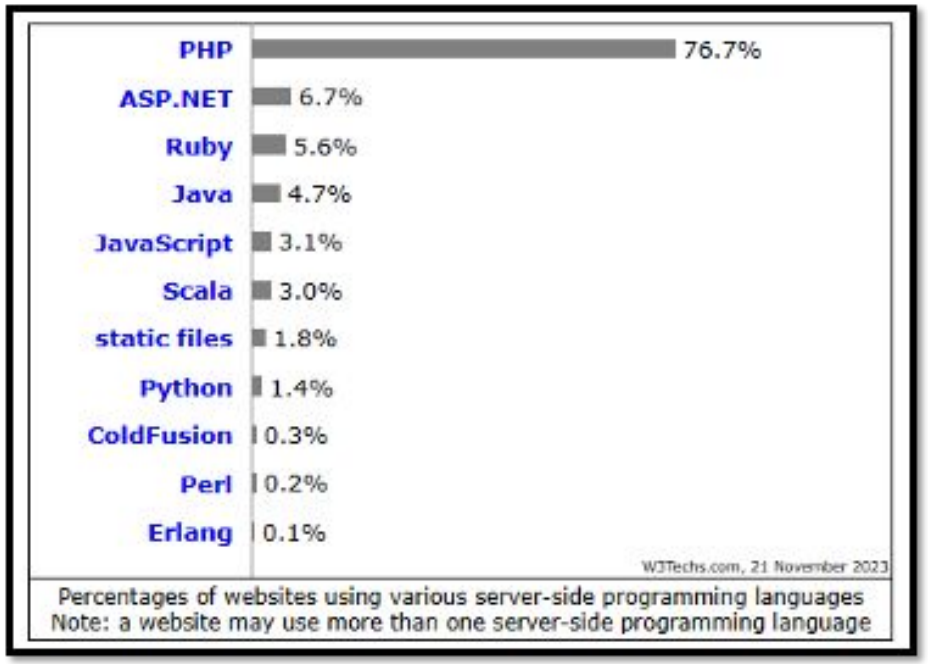
¿POR QUÉ UTILIZAR LENGUAJES DE SERVIDOR?

- **HTML** está muy **limitado** para los requerimientos de las aplicaciones web actuales.
- **HTML** precisas de **CSS** para dar **formato** y **Javascript** para ofrecer **interactividad**.
- **JavaScript** se ejecuta en el lado del **cliente** y por lo tanto su código es **visible**. Por lo tanto no son válidos para determinadas tareas como acceso a datos, acceso, sesiones..etc.
- Por estos motivos aparecieron los lenguajes **script** de servidor. Añaden **código incrustado** en el HTML que se interpreta en el lado de servidor.
- El **cliente** tan solo recibe la página en **HTML** y por lo tanto ofrece mayor **compatibilidad** con los diferentes navegadores.



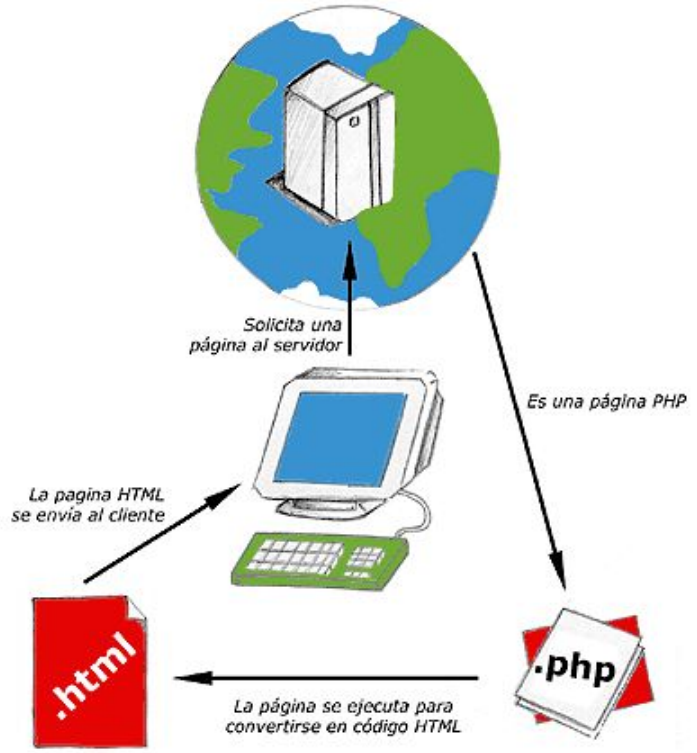
TIPOS DE LENGUAJES SCRIPT DE SERVIDOR

- **PHP.** Es muy utilizado, se basa en el lenguaje C y en Perl. Fácil de aprender.
- **ASP.NET.** Tecnología de Microsoft similar al php para servidores de aplicaciones IIS.
- **JSP.** Lenguaje basado en java.
- **Ruby.** Este lenguaje de programación open-source se inspiró en lenguajes como Perl, Eiffel y LISP.
- **Scala.** Nacido en 2001, es un lenguaje de programación de propósito general y de los más usados actualmente en el sector Big Data.
- **Phyton.** Es un lenguaje de programación orientado a objetos de alto nivel y fácil de interpretar con sintaxis fácil de leer. Python tiene un amplio uso en computación científica, desarrollo web y automatización.



¿QUÉ ES PHP?

- Hypertext Preprocessor. Preprocesador de hipertexto.
- PHP es el **lenguaje de lado servidor más extendido** en la web.
- Es un lenguaje que ha tenido una gran aceptación en la comunidad de desarrolladores, debido a la **potencia y simplicidad**.
- La facilidad de PHP se basa en que permite embeber pequeños **fragmentos de código dentro de lo que sería una página común creada con HTML**.
- Otra de las claves del éxito de PHP es que la mayoría de **los CMS más populares** (WordPress, Joomla!, Drupal), así como otros cientos de herramientas, **están desarrollados en PHP**.



CARACTERÍSTICAS PRINCIPALES

- PHP, aunque **multiplataforma**, fue concebido **inicialmente** para entornos **Linux** y es en este sistema operativo donde se pueden aprovechar mejor sus prestaciones. Además la mayoría de **servidores** web están instalados en **Linux**.
- El **estilo** de programación con PHP es totalmente **libre**. Puedes usar tanto programación estructurada (funciones) como Programación Orientada a Objetos (clases y objetos).
- Existen multitud de **herramientas**, librerías, funciones, frameworks gratuitos que llevan PHP a un nuevo nivel.
- **Compatible** con los principales **SGBD** (MYSQL, ORACLE…)
- Dispone de un gran número de **librerías** o **extensiones**.
- **Abierto** y **gratuito**.

ETIQUETA <?PHP...?>

- Los scripts de php deben estar contenidos en ficheros .php.
- El código PHP debe estar contenido en <?PHP..?>.
- Cada instrucción acaba en punto y coma;
- Para incluir comentarios de una línea utilizamos # y //. Para incluir párrafos /* y */.
- PHP entiende de manera distinta las mayúsculas y minúsculas, sobretodo con las variables. Sin embargo no es tan estricto con las palabras reservadas del propio lenguaje.
- Se puede agrupar código en bloque mediante llaves {}.

```php
<?php
$mensaje="Tengo hambre!!"; //Comentario de una linea
echo $mensaje; #Este comentario también es de una linea
/*En este caso
mi comentario ocupa
varias lineas, lo ves? */
?>
```

VARIABLES

- Técnicamente una variable apunta a una posición de la memoria en donde se almacena un dato.
- Las variables siempre deberían tener un nombre descriptivo sobre lo que ellas van a almacenar.
- Tienen que comenzar por $. El segundo carácter puede ser un _ o una letra.
- A partir del tercer carácter se pueden incluir números.
- No hay límite de tamaño para el nombre.
- No puede contener espacios en blanco.

```
$value1 = 1;
$value2 = 2.00;
```

```
$value2 = 'I can change the data type on the fly';
```

```
<?php
$var = 'Roberto';
$Var = 'Juan';
echo "$var, $Var";      // imprime "Roberto, Juan"

$4site = 'aun no';      // inválido; comienza con un número
$_4site = 'aun no';     // válido; comienza con un carácter de subrayado
$täyte = 'mansikka';    // válido; 'ä' es ASCII (Extendido) 228
?>
```

TIPOS DE DATOS BÁSICOS

Variables numéricas

- Este tipo de variables almacena cifras, números, que pueden tener dos clasificaciones distintas:
  - **Enteros** $entero=2002; Números sin decimales
  - **Reales** $real=3.14159; Números con o sin decimal

Variables alfanuméricas

- Este tipo de datos almacena textos compuestos, cadenas de caracteres, que pueden contener letras, símbolos y números o cifras.
  - **Cadenas** Almacenan variables alfanuméricas $cadena="Hola amigo";

Booleanas

- Este tipo de variables almacena un valor lógico, que puede valer verdadero o falso. Es muy común en la programación este tipo de variables booleanas.
  - **Booleano verdadero** $verdadero = true;
  - **Booleano falso** $falso = false;

TIPADO DINÁMICO

A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando definimos una variable asignándole un valor, el ordenador le atribuye un tipo.

Sin embargo, si pedimos en nuestro script realizar una operación matemática con esta variable, no obtendremos un mensaje de error sino que la variable cadena será asimilada a numérica (PHP hará todo lo posible por interpretar nuestra operación, aunque técnicamente no tenga mucho sentido hacer determinadas operaciones):

```php
<?
$cadena="5"; //esto es una cadena
$entero=3; //esto es un entero
echo $cadena+$entero
?>
```

```php
$variable="5"; //esto es una cadena
```

Este script dará como resultado "8". La variable cadena ha sido asimilada en entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, podemos operar entre variables tipo entero y real.

VARIABLES ASIGNADOS POR REFERENCIA

- En PHP también podemos asignar variables por referencia, aunque a decir verdad no es una característica que se use mucho. En ese caso no se les asigna un valor, sino otra variable, de tal modo que las dos variables comparten espacio en memoria para el mismo dato. La notación para asignar por referencia es colocar un "&" antes del nombre de la variable.

```php
<?php
$foo = 'Bob'; // Asigna el valor 'Bob' a $foo
$bar = &$foo; // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $foo; // $foo también se modifica.
echo $bar;
?>
```

- Esto dará como resultado la visualización dos veces del string "Mi nombre es Bob". Algo como: **Mi nombre es BobMi nombre es Bob**

FORZADO DE TIPOS

Tenemos varios métodos para forzar el tipo de datos:

A. Para cambiar el tipo de una variable simplemente le asignamos un valor con un nuevo tipo.

```
$cadena = 'esto es una cadena';
$cadena = 34 //La variable $cadena cambió de tipo
```

B. Podemos forzar una variable para que cambie de tipo con la función settype(). Entre "nuevo_tipo" tenemos: "integer“, “double”, “string“, “array“ y "object“.

```
settype($variable,"nuevo_tipo");
```

C. Por último podemos utilizar otros mecanismos de forzado que es similar al de otros lenguajes como C o Java.(Casting variables). Los forzados permitidos son: (int), (integer) - fuerza a entero (integer) (real), (double), (float) (string) (array) (object) (unser) - fuerza a null (binary) - fuerza a "binary string"

```
$variable = "23";
$variable = (int) $variable;
```

CONSTANTES Y VARIABLES DE VARIABLES

- Las constantes son valores que no cambian durante la ejecución del programa y para definirlas se utiliza la función **define y desde la versión PHP 5.3 también con const**.

  Por ejemplo: **define("PI", 3.141592);** y se utiliza sin el símbolo &. Echo PI;

  const PI = 3.141592;

- PHP permite utilizar lo que se conoce como **macrosustitución de variables**. Se trata de utilizar dos veces el símbolo $, de modo que se toma el contenido de la variable definida mediante el segundo $ y con este contenido se da nombre a la variable.

  $X="variable1";

  $$X=5;

  echo $variable1; //Escribe 5 por pantalla.

OPERADORES

- Un operador es algo que toma uno o más valores y produce otro valor. Sumar, restar, comparación de string, etc..

- Vamos a ver los principales:
  - Aritméticos
  - Asignación
  - Comparación
  - Incremento/decremento
  - Lógicos
  - String

OPERADORES ARITMÉTICOS

Ejemplo	Nombre	Resultado
+$a	Identidad	Conversión de $a a int o float según el caso.
-$a	Negación	Opuesto de $a.
$a + $b	Adición	Suma de $a y $b.
$a - $b	Sustracción	Diferencia de $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo	Resto de $a dividido por $b.
$a ** $b	Exponenciación	Resultado de elevar $a a la potencia $bésima. Introducido en PHP 5.6.

```php
<?php

echo (5 % 3)."\n";          // muestra 2
echo (5 % -3)."\n";         // muestra 2
echo (-5 % 3)."\n";         // muestra -2
echo (-5 % -3)."\n";        // muestra -2

?>
```

OPERADORES ASIGNACIÓN

- El operador de asignación es el = pero que no se entienda “igual a” sino que significa que el operando de la izquierda se asigna el valor del de la derecha.
  $a = “Uno”; // a vale Uno
  $b = $a; // b vale Uno .

```php
<?php

$a = ($b = 4) + 5; // ahora $a es igual a 9 y $b se ha establecido en 4.

?>
```

- Existen dos operadores de string ambos para concatenar cadenas
  - . con el punto se concatenan cadenas
  - .= es el operador de asignación sobre concatenación.

```php
<?php

$a = 3;
$a += 5; // establece $a en 8, como si se hubiera dicho: $a = $a + 5;
$b = "Hola ";
$b .= "ahí!"; // establece $b en "Hola ahí!", al igual que $b = $b . "ahí!";

?>
```

OPERADORES COMPARACIÓN

Ejemplo	Nombre	Resultado
$a == $b	Igual	**TRUE** si *$a* es igual a *$b* después de la manipulación de tipos.
$a === $b	Idéntico	**TRUE** si *$a* es igual a *$b*, y son del mismo tipo.
$a != $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a <> $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a !== $b	No idéntico	**TRUE** si *$a* no es igual a *$b*, o si no son del mismo tipo.
$a < $b	Menor que	**TRUE** si *$a* es estrictamente menor que *$b*.
$a > $b	Mayor que	**TRUE** si *$a* es estrictamente mayor que *$b*.
$a <= $b	Menor o igual que	**TRUE** si *$a* es menor o igual que *$b*.
$a >= $b	Mayor o igual que	**TRUE** si *$a* es mayor o igual que *$b*.

OPERADORES INCREMENTO/DECREMENTO/LÓGICOS

Ejemplo	Nombre	Efecto
++$a	Pre-incremento	Incrementa *$a* en uno, y luego retorna *$a*.
$a++	Post-incremento	Retorna *$a*, y luego incrementa *$a* en uno.
--$a	Pre-decremento	Decrementa *$a* en uno, luego retorna *$a*.
$a--	Post-decremento	Retorna *$a*, luego decrementa *$a* en uno.

Ejemplo	Nombre	Resultado
$a and $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**.
$a or $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**.
$a xor $b	Xor (o exclusivo)	**TRUE** si *$a* o *$b* es **TRUE**, pero no ambos.
! $a	Not (no)	**TRUE** si *$a* no es **TRUE**.
$a && $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**.
$a || $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**.

ESTRUCTURAS DE CONTROL.

- **IF … ELSE**
  - Es una estructura de control condicional, hace una cosa u otra
  - if (expr) sentencia else sentencia

```php
<?php
if ($a > $b) {
  echo "a es mayor que b";
} else {
  echo "a NO es mayor que b";
}
?>
```

```php
<?php
if ($a > $b) {
  echo "a es mayor que b";
}
?>
```

ESTRUCTURAS DE CONTROL.

- **WHILE / DO … WHILE**
  - While
    while (expr)
    sentencia

  - Evalúa la expresión y mientras es true se ejecuta la sentencia

  - do … while es al revés
    do {
    sentencias
    }while(expr)

  - Primero ejecuta la sentencia mientras la expresión es true que se evalúa la expresión

```php
<?php
/* ejemplo 1 */

$i = 1;
while ($i <= 10) {
    echo $i++;  /* el valor presentado sería
                   $i antes del incremento
                   (post-incremento) */
}

/* ejemplo 2 */

$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
?>
```

```php
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

ESTRUCTURAS DE CONTROL.

- **FOR.** Formato.
  for (expr1; expr2; expr3)
  sentencia
- Se evalúa la expr1 mientras la expr2 es TRUE
  La expr3 es un cambio de variable que se incrementa o decrementa.

```php
<?php
/* ejemplo 1 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

/* ejemplo 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}
```

```php
/* ejemplo 3 */

$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}

/* ejemplo 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
?>
```

ESTRUCTURAS DE CONTROL.

- **FOREACH**
  - Se utiliza exclusivamente para recorrer arrays u objetos
  - Tiene dos formas de utilizarse

  foreach (expresión_array as $valor)
  sentencias
  foreach (expresión_array as $clave => $valor)
  sentencias

!OJO!

Si usamos referencia &$valor permanece aún después del bucle foreach. Se recomienda destruirla con unset(). Más info: https://www.php.net/manual/es/control-structures.foreach.php

```php
<?php
$array = array(1, 2, 3, 4);
foreach ($array as &$valor) {
    $valor = $valor * 2;
}
// $array ahora es array(2, 4, 6, 8)
unset($valor); // rompe la referencia con el último elemento
?>
```

```php
foreach ($array as $clave => $valor) {
    // $array[3] se actualizará con cada valor de $array...
    echo "{$clave} => {$valor} ";
    print_r($array);
}
```

ESTRUCTURAS DE CONTROL.

- **SWITCH**
  - Es un condicional
  - Es similar a la sentencia IF

```
switch (variable)
{
case 'value':
# code...
break;
default:
# code...
break;
}
```

```php
switch ($i) {
    case 0:
        echo "i es igual a 0";
        break;
    case 1:
        echo "i es igual a 1";
        break;
    case 2:
        echo "i es igual a 2";
        break;
}
```

```php
<?php
for ($i = 0; $i < 5; ++$i) {
    if ($i == 2)
        continue
    print "$i\n";
}
?>
```

- **Break**
  Finaliza la ejecución de la estructura for, foreach, while, do-while o switch en curso.
- **Continue**
  Se utiliza dentro de las estructuras for, foreach, while, do-while o switch para pasar a la siguiente iteración.

ARRAYS.

Un array o tabla nos permite almacenar varios elementos y es necesario el uso de un índice o clave para poder referirnos a cada uno de ellos.

Arrays comunes, índices numéricos.

En este caso este array cataloga sus elementos, comúnmente llamados valores, por números. Los números del 1 al 5 son por lo tanto las claves y los sentidos ("tocar", "oir"...) son los valores asociados.

```
$sentido[1]="ver";
$sentido[2]="tocar";
$sentido[3]="oir";
$sentido[4]="gustar";
$sentido[5]="oler";
```

Arrays asociativos.

Si lo deseamos, es posible emplear nombres (cadenas) para clasificar los elementos del array.

```
$moneda["espana"]="Peseta";
$moneda["francia"]="Franco";
$moneda["usa"]="Dolar";
```

Otra forma de definir idénticamente este mismo array y que nos puede ayudar para la creación de arrays más complejos es la siguiente sintaxis:

```
<?
$moneda=array("espana"=> "Peseta","francia" => "Franco","usa" => "Dolar");
?>
```

ARRAYS MULTIDIMENSIONALES.

Una forma muy practica de almacenar datos es mediante la creación de arrays multidimensionales (tablas o matrices con más de una dimensión).

Pongamos el ejemplo siguiente: Queremos almacenar dentro de una misma tabla el nombre, moneda y lengua hablada en cada país. Para hacerlo podemos emplear un array llamado país que vendrá definido por estas tres características (claves)

- A partir de PHP 5.4 también se puede usar la sintaxis de array corta, la cual reemplaza *array()* con *[]*.
- La clave puede ser un integer o un string.

```php
<?
$pais=array
(
"espana" =>array
   (
   "nombre"=>"España",
   "lengua"=>"Castellano",
   "moneda"=>"Peseta"
   ),
"francia" =>array
   (
   "nombre"=>"Francia",
   "lengua"=>"Francés",
   "moneda"=>"Franco"
   )
);
echo $pais["espana"]["moneda"] //Saca en pantalla: "Peseta"
?>
```

```php
$array = array(
    1    => "a",
    "1"  => "b",
    1.5  => "c",
    true => "d",
);
```

FUNCIONES DE ARRAYS.

PHP incluye un nutrido conjunto de funciones para trabajar con Arrays. En ellas nos podemos apoyar para realizar toda una serie de operaciones típicas como ordenar elementos por orden alfabético directo o inverso, por claves, contar el numero de elementos que componen el array además de poder movernos por dentro de él hacia delante o atrás.

Función	Descripción
array_values (mi_array)	Lista los valores contenidos en mi_array
asort(mi_array) y arsort(mi_array)	Ordena por orden alfabético directo o inverso en función de los valores
count(mi_array)	Nos da el numero de elementos de nuestro array
ksort(mi_array) y krsort(mi_array)	Ordena por orden alfabético directo o inverso en función de las claves
list ($variable1, $variable2...)=mi_array	Asigna cada una variable a cada uno de los valores del array
next(mi_array), prev(mi_array), reset(mi_array) y end(mi_array)	Nos permiten movernos por dentro del array con un puntero hacia delante, atras y al principio y al final.
each(mi_array)	Nos da el valor y la clave del elemento en el que nos encontramos y mueve al puntero al siguiente elemento.

Trabajando con arrays