

## U.D.2.- Características da linguaxe PHP

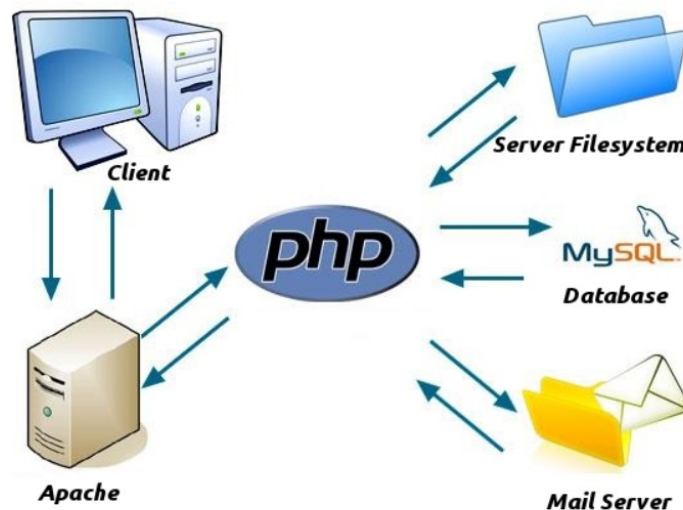


### Sumario

U.D.2.- Características da linguaxe PHP.....	1
Introdución.....	2
Inserción de código.....	3
Sentencias.....	3
Variables.....	3
Variables predefinidas.....	4
Tipos de datos en PHP.....	5
Operadores.....	7
Estruturas de control.....	7
Alternativa simple: if, else.....	7
Alternativa múltiple: if, elseif.....	8
Alternativa con switch.....	8
Bucle 1. while.....	9
Bucle 2. do_while.....	9
Bucle 3. for.....	10
Bucle4. foreach.....	10
Arrays.....	11
Como crear un array.....	11
Mostrar contido dun array.....	13
Eliminar elemento dun array: unset( ) e array_values( ).....	14

## Introdución

**PHP**, orixinalmente o acrónimo de “*Personal Home Pages*” e posteriormente o acrónimo recursivo de “*PHP Hypertext Preprocessor*” é unha linguaxe de programación de servidor: é executado no servidor e o resultado (normalmente en formato HTML) é recibido polo navegador do cliente:



Foi creado no 1994 como un subconxunto de scripts en Perl creados por [Rasmus Lerdof](#). Posteriormente foi modificado por outros programadores ata a versión actual, que é a 8.0 (neste 2021).

Para poder traballar con PHP temos que ter instalado un servidor de http/https co Apache. Pode ser Apache, Nginx, etc.

Na aula traballaremos en linux con Docker e contenedores, mentres que en Windows unha opción rápida pode ser empregar XAMPP.

Hai moitos sitios web nos que podes ter acceso a información de php, pero o máis usado polos programadores, e onde podemos obter toda a información que empregaremos no presente curso, pode ser o sitio oficial:

<https://www.php.net/manual/es/>

ou algo máis resumido:

<https://manuais.iessanclemente.net/index.php/PHP>

## Inserción de código

Fíxate que o código PHP insírese dentro do código HTML, empregando as etiquetas de apertura e peche **<?php** e **?>**, respectivamente.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Primeiro exemplo</title>
</head>
<body>
    <h2> Primeiro exemplo en PHP </h2>
    <?php
        echo "Ola mundo<br>";
    ?>
</body>
</html>
```

## Sentencias

- As sentencias teñen que rematar con ';'.
- A última sentencia do bloque non precisa do ';' pero é recomendable.
- Agruparemos varias sentencias coas chaves { .... }, por exemplo dentro dun bucle **for**

## Variables

As variables comezan co símbolo de \$, e non é preciso definilas antes de usalas, como nalgúñas linguaxes de programación.

**Olo** coas minúsculas e maiúsculas, é distinto \$apelido que \$Apelido.

O nome da variable só pode comezar por unha letra ou por barra baixa: '\_'

O resto dos caracteres pode ser calquera letra, números ou '\_'

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Variables</title>
</head>
<body>
    <h2> Variables </h2>
    <?php
        $a=5;
        $_b=7.35;
        $animal33="Pantera Rosa";

        echo $animal33, "<br>", $a, "<br>", $_b;
    ?>
</body>
</html>

```

Cada variable é visible no ámbito no que está definida, teremos variables locais dentro de cada función, e variables globais co seu ámbito fóra das funcións. Para que as variables globais se poidan empregar dentro da función empregaremos a palabra reservada **global**. Veremos exemplos disto polo miúdo cando vexamos as funcións.

## Variables predefinidas

Existen algunhas variables predefinidas , gardando información do entorno de execución do intérprete e do propio PHP.

As máis empregadas, todos array asociativos, son:

**\$\_GET['variable']**: variables recibidas por GET

**\$\_POST['variable']**: variables recibidas por POST

**\$\_SERVER['variable']**: todas as variables de alcance global

**\$\_COOKIE['variable']**: variables pasadas a través de cookies

**\$\_FILES['variable']**: ficheiros pasados a través de POST

As principais variables do servidor están nun array asociativo **\$\_SERVER**: **\$\_SERVER['SERVER\_NAME']**, **\$\_SERVER['SERVER\_PORT']**, **\$\_SERVER['SERVER\_SOFTWARE']**, **\$\_SERVER['SERVER\_REMOTE\_PORT']**, **\$\_SERVER['REMOTE\_ADDR']**, etc.

Por exemplo, se temos un formulario que envía por GET, con 2 input de tipo texto, con name igual a "nome" e "apelido", un programa que captura os valores introducidos e os amosa por pantalla podería ser:

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Paso por GET</title>
</head>
<body>
    <h2> Variables </h2>
    <?php
        $onome=$_GET['nome']; //GARDAMOS O nome NOUTRA VARIABLE

        echo $onome <br>, $_GET['apelido']; /* E O apelido DIRECTAMENTE */
    ?>
</body>
</html>

```

## Tipos de datos en PHP

Xa vimos que as variables en PHP comezan sempre polo símbolo \$. O seu tipo é asignado automaticamente, pero pode cambiar se cambia o seu contido.

Os tipos de datos simples en PHP son:

- ✓ **booleano** (*boolean*). Os seus posibles valores son **true** y **false** . Ademais, calquera número enteiro considérase como *true* , salvo o 0 que é *false* .
- ✓ **entero** (*integer*). Calquera número sen decimais. Pódense representar en formato decimal, octal (comenzando por un 0 ), ou hexadecimal (comenzando por 0x ).
- ✓ **real** (*float*). Calquera número con decimais. Pódense representar tamén en notación científica.
- ✓ **cadena** (*string*). Conxuntos de caracteres delimitados por comiñas simples o dobres.
- ✓ **null**. Es un tipo de datos especial, que se usa para indicar que la variable non ten valor.

Por exemplo:

```

$oBooleano = false;
$oEnteiro= 0x2A;
$oReal = 2.35;
$aCadea="ola que tal?";
$a = null;

```

Cando se fan operacións con variables de distintos tipos, ambas convértense primeiro a un tipo común. Por exemplo, se sumamos un enteiro a un real, o enteiro convértese primeiro a real antes de facer a suma:

```
$enteiro=4;
$real= 3.7;
$resultado = $enteiro + $real; // O resultado será real, e valerá 7.7
```

Tamén podemos realizar a conversión de xeito forzado:

```
$enteiro=4;
$real= 3.7;
$resultado = $enteiro + (int) $real; // Agora a variable $real convértese a enteiro
//antes de sumarse. Así, $ resultado será enteiro e valerá 7
```

Existen funcións predefinidas para comprobar o tipo de dato. Todas teñen unha sintaxe parecida:

- **`is_bool()`** : Comproba se unha variable é de tipo booleano .
- **`is_float()`** : Comproba se unha variable é de tipo float .
- **`is_numeric()`** : Comproba se unha variable é de tipo número ou un string que se pode converter a número.
- **`is_string()`** : Comproba se una variable é de tipo string .
- **`is_array()`** : Comproba se una variable é un array .
- **`is_object()`** : Comproba se una variable é un obxecto.

Todas devolven *true* ou *false*. Así:

```
$a = 5;
if (is_int($a))
    echo "A variable é enteira"; // Mostrarase a mensaxe
```

## Operadores

Os operadores aritméticos serán os mesmos que en Javascript: +, -, \*, /, %

Ademais do operador de asignación = están permitidos os operadores de asignación +=, -=, \*=, /=.

Os operadores de comparación son ==, !=, <, >, <=, >=. Ademais, existe o operador === que compara o valor e tipo. E !=. Así:

```
12 === 12.0 // é falso porque non coincide o tipo de dato
10 === 10 //será verdadeiro
```

O operador de concatenación para as cadeas é o operador punto (.)

Por exemplo:

```
$cad1 = "01a, ";
$cad2 = "que tal";
$cadea = $cad1 . $cad2; // $cadea valerá '01a, que tal'
```

## Estruturas de control

### Alternativa simple: if, else

De forma similar a Javascript:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Alternativa if-else</title>
</head>
<body>
    <h2> Variables </h2>
    <?php
        $num = $_GET['numero1'];
        if ($num < 5)
            echo "O número é menor que 5";
        else
            echo "Número maior ou igual que 5";
    ?>
</body>
</html>
```

## Alternativa múltiple: if, elseif

A diferenza con outras linguaxes é que o elseif non ten espazo polo medio (~~else if~~):

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Alternativa if-else</title>
</head>
<body>
    <h2> Variables </h2>
    <?php
        $num=$_GET['numero1'];
        if ($num < 5)
            echo "0 número é menor que 5";
        elseif ($num == 5)
            echo "0 número é igual a 5";
        else
            echo "Número maior que 5";
    ?>
</body>
</html>
```

## Alternativa con switch

Cando a comparación é sempre coa mesma variable e con ==, podemos empregar a sentencia switch:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Alternativa switch</title>
</head>
<body>
    <h2> Variables </h2>
    <?php
        $num=$_GET['numero1'];
        switch ($num)
        {
            case 1: echo "Número igual a 1";
                    break;
            case 3: echo "Número igual a 3";
                    break;
        }
```



```

        case 5: echo "Número igual a 5";
                break;
        default: echo "Número distinto a 1,3 e 5";
        }
    ?>
</body>
</html>

```

## Bucle 1. while

O bucle while repítese mentres a condición sexa certa, Sintaxe similar a Javascript ou C:

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>While. Conta atrás </title>
</head>
<body>
    <h2> Variables </h2>
    <?php
        $num=$_GET['numero1'];
        while ($num > 0)
        {
            echo $num, "<br>";
            $num--; //dentro do bucle DEBE cambiar a variable da condición
        }
        echo "Chegamos ao 0!";
    ?>
</body>
</html>

```

## Bucle 2. do\_while

Agora a condición avaláase ao final, polo que o bucle sempre se executa unha vez:

```

<?php
    $num=$_GET['numero1'];
    do
    {
        echo $num, "<br>";
        $num--; //dentro do bucle cambia a variable da condición
    } while ($num > 0); //Obrigatorio o ';'
    echo "Chegamos ao 0!"; ?>

```

## Bucle 3. for

```
<?php
    for ($i=0; $i < 20; $i++)
        echo $i,"<br>";
?>
```

## Bucle4. foreach

Veremos o **foreach** na sección dos **arrays**.

## Arrays

Os arrays serán un tipo de dato que asocian claves e valores. Serán principalmente de 2 tipos:

**INDEXADOS:** a clave é numérica, un ÍNDICE numérico. Empezamos a contar no 0 :

```
$notas = [ 8.5, 6, 9 ];
echo $notas[0]; // Mostrará o 8.5
echo $notas[1] ; // Mostrará o 6
echo $notas[2] ; // Mostrará o 9
```

8.5	6	9
[ 0 ]	[ 1 ]	[2]

**ASOCIATIVOS:** a clave é unha cadea de texto.

```
$arrayAsociativo = [
    'nome' => 'Federico',
    'apelidos' => 'Caeiro',
    'idade' => 17 ];
echo $arrayAsociativo['nome'] // Mostrará Federico
echo $arrayAsociativo['apelidos'] // Mostrará Caeiro
echo $arrayAsociativo['idade'] // Mostrará 17
```

'Federico'	'Caeiro'	17
[ 'nome' ]	[ 'apelidos' ]	[ 'idade' ]

## Como crear un array

Existen varios xeitos de crear un array en PHP

1.- Con **corchetes**:

Podemos crear un array como fixemos arriba:

```
$notas = [ 8.5, 6, 9 ]; //array Indexado

$arrayAsociativo = [ //array asociativo
    'nome' => 'Federico',
    'apelidos' => 'Caeiro',
    'idade' => 17 ];
```

2.- Co construtor **array**:

```
$notas = array( 8.5, 6, 9 );    //array Indexado

$arrayAsociativo = array(      //array asociativo
    'nome' => 'Federico',
    'apelidos' => 'Caeiro',
    'idade' => 17 );
```

## 3.- Elemento a elemento:

```
$notas[0]=8.5;
$notas[1]=6;
$notas[2]=9;
```

Ou ben:

```
$notas[0]=8.5;
$notas[]=6;
$notas[]=9;
```

8.5	6	9
[ 0 ]	[ 1 ]	[2]

Por defecto, nos arrays o primeiro elemento é o 0. Cando utilizamos a notación `$notas[]` o índice será o seguinte ao maior índice numérico xa asignado. Así, se creamos un array así:

```
$cidade[3]="París";
$cidade[]="Londres";
echo $cidade[4]; // Amosará Londres
```

E mesmo se están combinados:

```
$datos = array(3 => 17, 27, 55, 'c' => 121, 8);
echo $datos[3]."<br>"; // mostra 17
echo $datos[4]."<br>"; // Amosará 27
echo $datos[5]."<br>"; // Amosará 55
echo $datos[6]."<br>"; // . 0LL0: Amosará 8
echo $datos['c']."<br>"; // Amosará 121
```

## Mostrar contido dun array

Normalmente percorrерemos o **array** cos bucles **for** e **foreach**.

Nos array indexados é máis claro empregar o **for**:

```
//ARRAY INDEXADO
$ciade=array("París", "Londres", "Lisboa");
for ($i=0; $i<3; $i++)
    echo $ciades[$i]."<br>";
```

'París'	'Londres'	'Lisboa'
[ 0 ]	[ 1 ]	[2]

Aínda que menos empregado, tamén se pode empregar o **foreach**:

```
//ARRAY INDEXADO
$ciade=array("París", "Londres", "Lisboa");
foreach($ciade as $valor)
    echo $valor;
```

Nos arrays asociativos é máis claro empregar o **foreach**:

```
//ARRAY ASOCIATIVO
$capital=array("Francia"=>"París",
               "Inglaterra"=>"Londres",
               "Portugal"=>"Lisboa");

foreach($capital as $pais=>$ciade)
    echo "A capital de $pais é $ciade";
```

Se queremos imprimir unha posición dada, podemos empregar as chaves: {}

```
//ARRAY ASOCIATIVO
$capital=array("Francia"=>"París",
               "Inglaterra"=>"Londres",
               "Portugal"=>"Lisboa");
echo "A capital de Francia é {$capital["Francia"]}"; //ou empregar o '.' e concatenar:
echo " A capital de Francia é ". $capital["Francia"];
```

Para amosar o contido do array completo amosando o contido podemos empregar as funcións

```
print_r($capital); //os contidos
var_dump($capital); //tamén o tipo dos datos que contén cada posición
```

## Eliminar elemento dun array: `unset( )` e `array_values( )`

Para eliminar un elemento dun array, empregaremos `unset()`, e logo empregaremos `array_values()` para reordenar o array (eliminamos o elemento borrado e reordenamos os índices):

```
//ARRAY INDEXADO
$cidade=array("París", "Londres", "Lisboa");

//ELIMINAMOS O SEGUNDO ELEMENTO
unset($cidade[1]);
$cidade = array_values($cidade);
```