CURSO .PHP

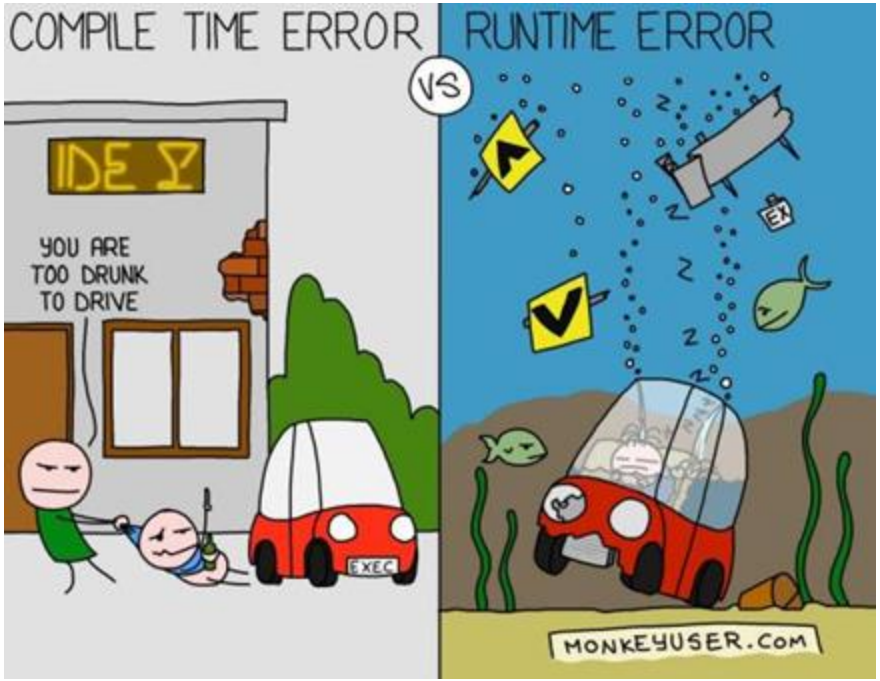
FUNDAMENTOS BÁSICOS

Autor: Jon Vadillo

Modificado por: Inés Larrañaga

Contenidos

- Fundamentos básicos
- Sintaxis
- Variables y tipos
- Operadores
- Funciones
- Arrays
- Estructuras de control
- Include y Require

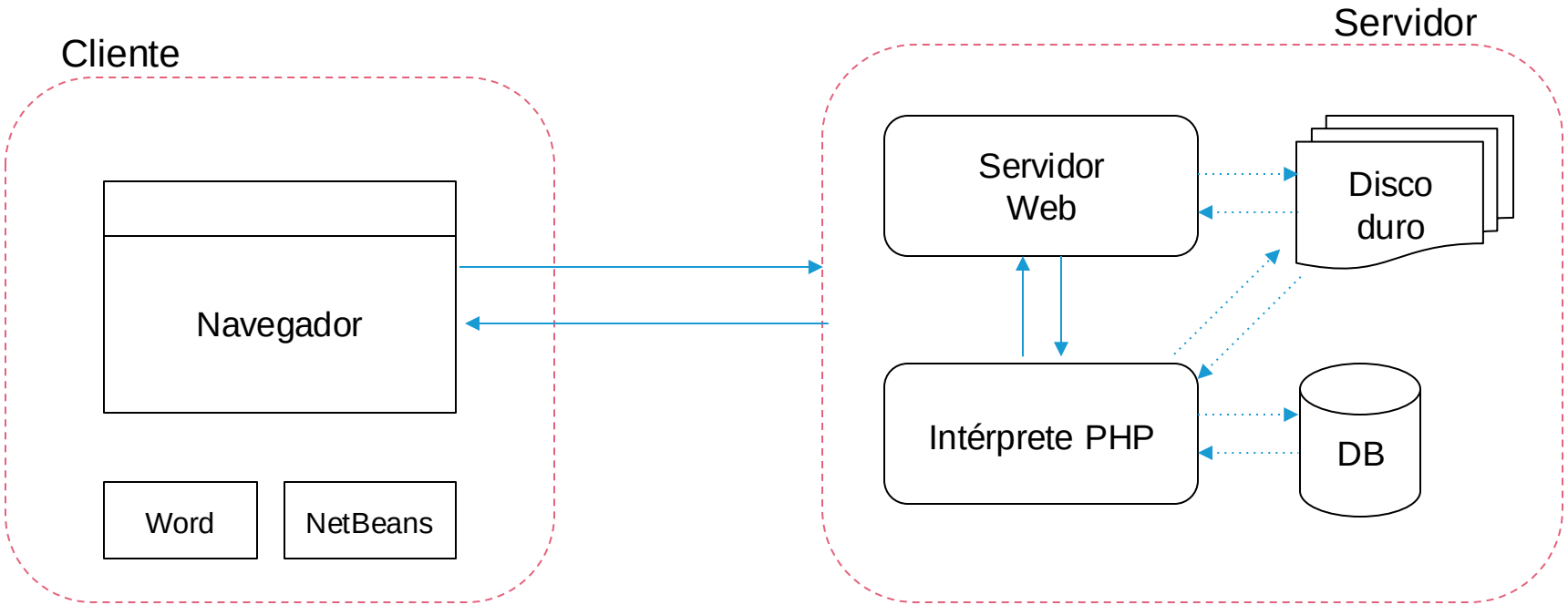

COMPILE TIME ERROR
VS
RUNTIME ERROR
IDE
YOU ARE TOO DRUNK TO DRIVE
EXEC
EX
MONKEYUSER.COM

Lenguaje de programación interpretado y ~~de tipado estático~~ dinámicamente tipado del lado del servidor.

Características

- Permite crear aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El servidor el que se encarga de **ejecutar el código y enviar su resultado HTML** al navegador.
- El tipo de las variables no se conoce hasta la ejecución.
- Permite la programación orientada a objetos.
- La extensión utilizada es .php

Funcionamiento



Entorno de desarrollo

- Instalar servidor “todo en uno”: Apache + PHP + MySQL)
  - Más conocidos: XAMPP, MAMP, …
  - Laragon: añade posibilidad de hosts virtuales, Ngnix, etc.
- Entorno de desarrollo virtualizado
  - Vagrant → Laravel Homestead, Scotch Box

Entorno de desarrollo

Entorno de desarrollo

- Pasos
  - Descargar e Instalar Laragon: https://laragon.org
  - Instalar y configurar PHP Storm o VS Code
  - (Opcional) Instalar y configurar Xdebug
- Repositorio
  - https://github.com/jvadillo/howto-configure-xdebug-xcode

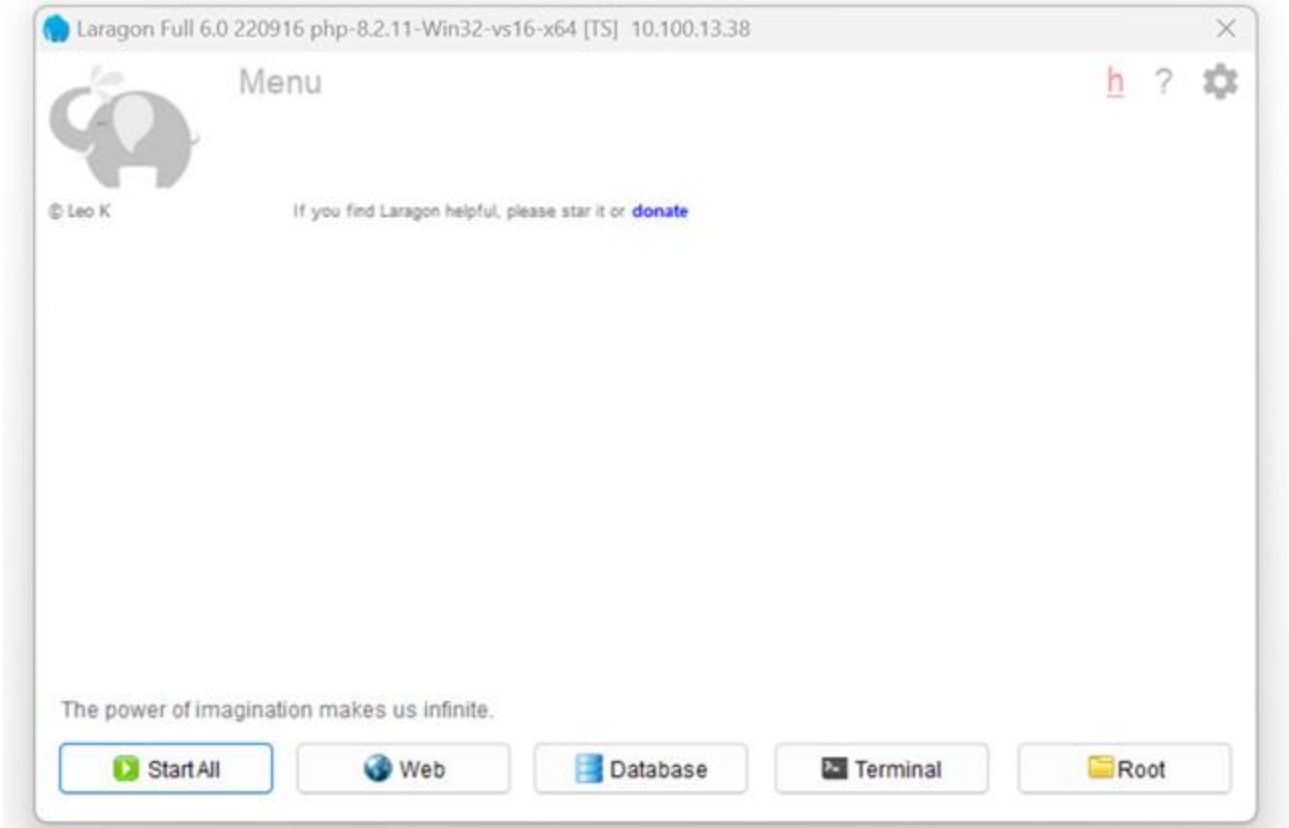
Docs Why Laragon? Testimonials Download About Community English

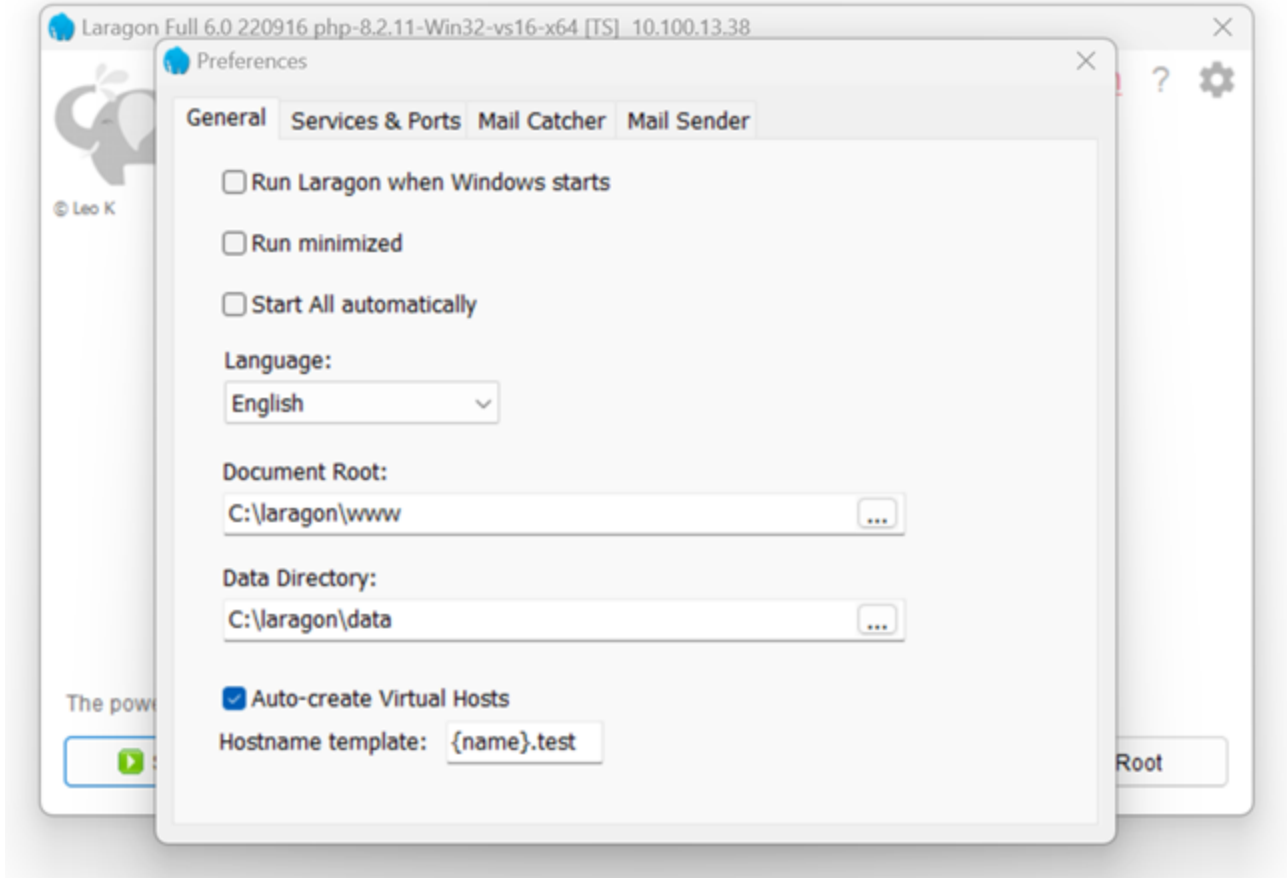
Laragon

Modern & Powerful - Easy Operation
Productive. Portable. Fast. Effective. Awesome!

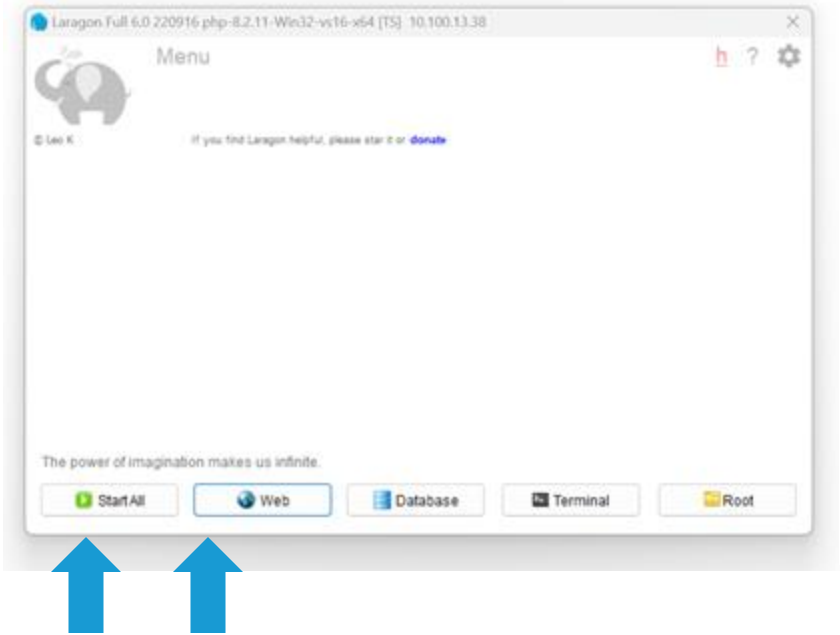
I was using Xampp for local development until about a year ago when I found Laragon & I haven't looked back since. This is seriously the most powerful local development tool I have in my entire "developer toolkit". I started working for a new company about 2 months ago & their entire team was using various setups for local dev and were always encountering problems. Setting up a new project locally was always a nightmare for them due to mysql version conflicts, having to change php versions, etc. When I introduced Laragon to them, they were amazed at how easy it was to install, maintain and how flexible it was to update/switch their dependencies. My manager told me that since the team has started using Laragon for local dev, **project setup time has gone down from an average of 6 HOURS, to less than 30 mins**. AMAZING.

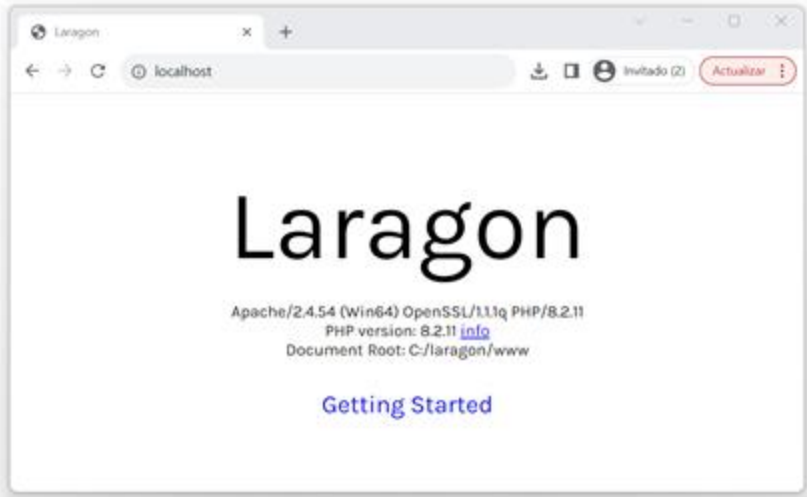
Just saying thank you!


Laragon Full 6.0 220916 php-8.2.11-Win32-vs16-x64 [TS] 10.100.13.38
Menu
© Leo K
If you find Laragon helpful, please star it or donate
The power of imagination makes us infinite.
Start All
Web
Database
Terminal
Root


Laragon Full 6.0 220916 php-8.2.11-Win32-vs16-x64 [TS] 10.100.13.38
© Leo K
The powe
Root
Preferences
General
Services & Ports
Mail Catcher
Mail Sender
Run Laragon when Windows starts
Run minimized
Start All automatically
Language:
English
Document Root:
C:\laragon\www
Data Directory:
C:\laragon\data
Auto-create Virtual Hosts
Hostname template:
{name}.test

Comprobar que todo funciona (1/2)





Comprobar que todo funciona (2/2)

- Abrir VS Code y crear una nueva carpeta llamada test en: C:\laragon\www
- Dentro de la carpeta test, crear un nuevo fichero llamado index.php
- Insertar el contenido de la derecha y guardar.
- Reiniciar el servidor y abrir en un navegador la URL: http://test.test/

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP Test</title>
    </head>
    <body>
        <?php echo '<p>Hello World</p>'; ?>
    </body>
</html>
```

Extensiones recomendadas de VS Code

- PHP Intelephense: https://intelephense.com/
- PHP Getter & Setter
- PHP Debug: https://marketplace.visualstudio.com/items?itemName=xdebug.php-debug

Sintaxis

```
<?php
    ...
    ...
?>
```

- Cualquier cosa fuera del par de etiquetas **es ignorado por el intérprete**.
- Las instrucciones deben terminar en **punto y coma** (excepto la última antes del cierre del bloque).
- Si un fichero contiene solamente código de PHP, es preferible omitir la etiqueta de cierre de PHP al final del mismo

hola-mundo.php

```
<html>
<head>
    <title>Hola Mundo PHP</title>
</head>
<body>
    <?php
        echo "<p>Hola Mundo</p>";
    ?>
</body>
</html>
```

Guía para realizar el curso

1. **Vamos a usar la tarea de Github Classroom como repositorio** en GitHub para todos los ejercicios llamado workspace-dwes
2. **Acepta la tarea** para que se te cree el repositorio.
3. **Clona** el repositorio en local mediante el comando **git clone**
4. Añade el fichero .**gitignore** antes del primer commit
5. Crea una carpeta llamada pruebas (**mkdir pruebas**) para almacenar en ella los códigos de ejemplo o pruebas que irás realizando.
6. En el caso de los **ejercicios**, crea **una carpeta para los de cada tema**: 01-Introduccion, 02-Formularios, 03-Sesiones-Cookies, 04-Acceso-a-datos, ...
7. Cada vez que vayas a realizar un ejercicio, crea una **nueva rama** con numero del tema seguida del nombre del ejercicio (01-ejercicio01, 01-ejercicio02,...). Una vez lo termines, haz un merge a la rama principal.

GIT BASICS

`git init <directory>`	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
`git clone <repo>`	Clone repo located at `<repo>` onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.
`git config user.name <name>`	Define author name to be used for all commits in current repo. Devs commonly use `--global` flag to set config options for current user.
`git add <directory>`	Stage all changes in `<directory>` for the next commit. Replace `<directory>` with a `<file>` to change a specific file.
`git commit -m "<message>"`	Commit the staged snapshot, but instead of launching a text editor, use `<message>` as the commit message.
`git status`	List which files are staged, unstaged, and untracked.
`git log`	Display the entire commit history using the default format. For customization see additional options.
`git diff`	Show unstaged changes between your index and working directory.

UNDOING CHANGES

`git revert <commit>`	Create new commit that undoes all of the changes made in `<commit>`, then apply it to the current branch.
`git reset <file>`	Remove `<file>` from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes.
`git clean -n`	Shows which files would be removed from working directory. Use the `-f` flag in place of the `-n` flag to execute the clean.

REWRITING GIT HISTORY

`git commit --amend`	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
`git rebase <base>`	Rebase the current branch onto `<base>`. `<base>` can be a commit ID, branch name, a tag, or a relative reference to HEAD.
`git reflog`	Show a log of changes to the local repository's HEAD. Add `--relative-date` flag to show date info or `--all` to show all refs.

GIT BRANCHES

`git branch`	List all of the branches in your repo. Add a `<branch>` argument to create a new branch with the name `<branch>`.
`git checkout -b <branch>`	Create and check out a new branch named `<branch>`. Drop the `-b` flag to checkout an existing branch.
`git merge <branch>`	Merge `<branch>` into the current branch.

REMOTE REPOSITORIES

`git remote add <name> <url>`	Create a new connection to a remote repo. After adding a remote, you can use `<name>` as a shortcut for `<url>` in other commands.
`git fetch <remote> <branch>`	Fetches a specific `<branch>`, from the repo. Leave off `<branch>` to fetch all remote refs.
`git pull <remote>`	Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
`git push <remote> <branch>`	Push the branch to `<remote>`, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

Fuente: Atlassian

echo & print

- En PHP hay dos funciones principales imprimir texto: **`echo`** y **`print`**.

```php
<?php
echo "Hola, mundo!<br>";
echo "Hola,", " mundo!";
echo("<h2>Hola, mundo!</h2>");

print("Hola, mundo!");
?>
```

Comentarios

```php
<?php
echo 'Comentario de una línea'; // Mi comentario

/* Esto es un comentario multilínea
 y otra lína de comentarios */

echo 'Comentario de una línea'; # Otro comentario
?>
```

Variables y tipos

```
$a = 12; //integer
$b = "foo"; //string
$c = True; //boolean
//Punto flotante (floats):
$d = 1.234;
$e = 1.2e3;
$f = 7E-10;
```

Tipos de variables

- boolean
- integer
- float
- string

- array
- object
- NULL (representa una variable sin valor)
- ...

Reglas

- Las variables tienen que empezar con un signo de dólar seguido por una letra o guión bajo.
- Las variables no pueden empezar con un número o símbolos especiales
- PHP es “case sensitive” en nombres de variables, funciones o clases.

```
$a = "¡Hola!"; //letra (OK)
$_b = "¡Hola!"; //guión bajo (OK)
$4c = "¡Hola!"; //numero (Mal)
$*d = "¡Hola!";
//símbolo especial (Mal)
```

<?= ... =>

- Sirve como abreviación de **`echo`**:

```php
<?php
  $nombre = "Nora";
?>

<?php
   echo $nombre;
?>
```

```php
<?php
  $nombre = "Nora";
?>

  <?= $nombre ?>
```

Hands on!

01. Crea dos variables llamadas nombre y edad. La variable nombre tendrá el valor “Mikel” y la variable edad tendrá asignado el valor 22. A continuación muéstralas por pantalla.

Strings

```php
echo 'Esto es un string sencillo';

echo 'Es posible definir un string
  en varias líneas y seguirá
 funcionando perfectamente';


echo 'Escape de caracteres: "I\'m Jon"';

// Resultado: Escape de caracteres: I'm Jon"
```

Strings

```php
// Resultado: Ha borrado C:\*.*?

echo 'Has entrado en C:\\*.*?';

// Resultado: Ha borrado C:\*.*?

echo 'Has entrado en C:\*.*?';

// Resultado: Esto no se expandirá: \n una nueva línea

echo 'Esto no se expandirá: \n una nueva línea';
```

Concatenación

```php
$cadena1 = "Mi nombre es ";

$cadena2= "Mikel";

$resultado = $cadena1 . ' ' . $cadena2;

//Otra forma:

$cadena1 = "Mi nombre es: ";

$cadena1 .= "Mikel"
```

Concatenación

En caso de que el string no necesite interpolar variables, es decir sea un literal, es mejor usar comilla simple ‘’.

En caso de que sea necesario la interpolación, entonces usar comilla doble “”.

```php
// Incluir variables directamente
$cadena1 = "Mi nombre es ";
$cadena2= "Mikel";
$resultado = "{$cadena1} {$cadena2}";
// Otra forma
$resultado = "$data1 $data2";
```

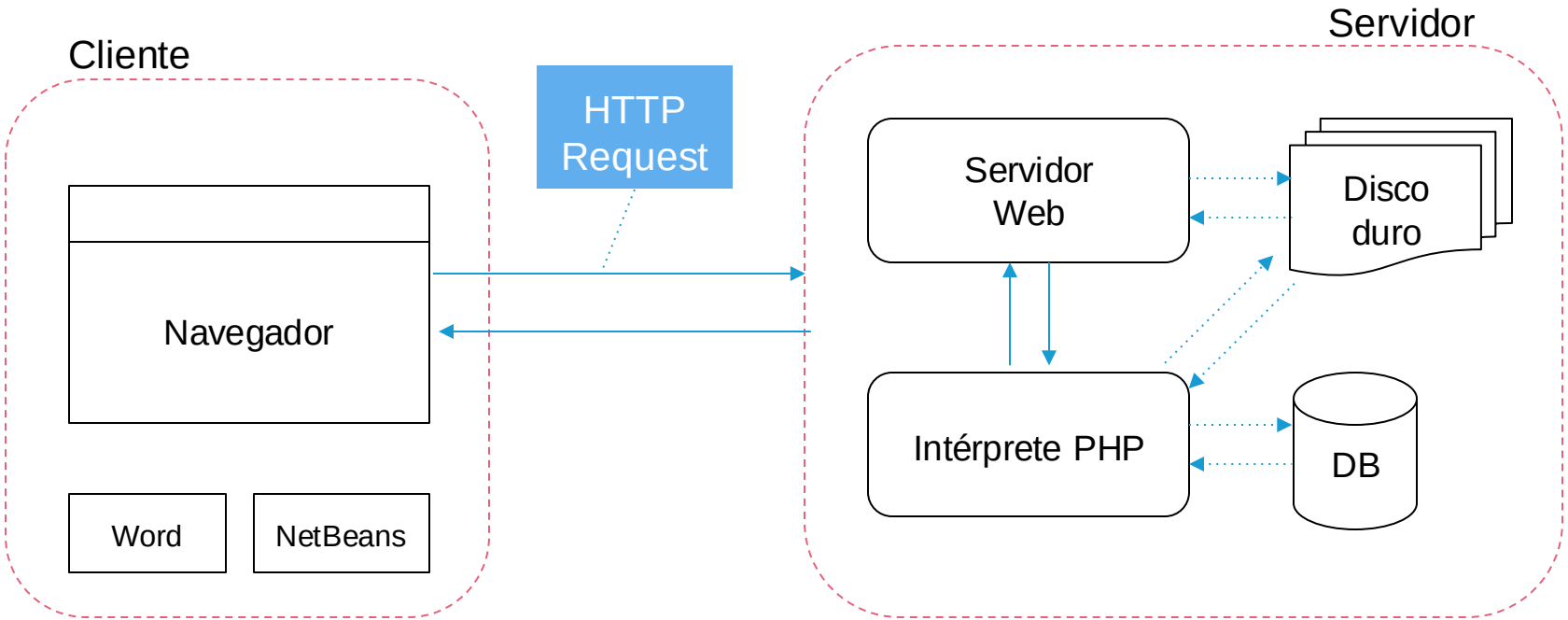
Hands on!

02. Crea dos variables llamadas nombre y edad. La variable *nombre* tendrá el valor “Mikel” y la variable *edad* tendrá asignado el valor 22. A continuación crea una página que muestre la siguiente frase:

Mi amigo Mikel tiene 22 años.

HTML y PHP

Fundamentos básicos



Fundamentos básicos

- Existen distintos métodos HTTP para enviar información al servidor: GET, POST, PUT, DELETE,...
- PHP proporciona tres **variables superglobales** para acceder a la información enviada en la petición del cliente:
  - $_REQUEST
  - $_GET
  - $_POST

HTTP GET

- Envía la información **en la URL**.
- La información se codifica en un esquema de **clave - valor** separados por un ampersand **&** que comienza a partir del interrogante:

http://miapp.com/index.php?key1=value1&key2=value2

http://miapp.com/index.php?nombre=Juan&edad=22

HTTP GET

- La información enviada **es visible.** No se debe enviar información sensible.
- No se pueden enviar datos binarios (imágenes u otros archivos).
- Los datos se manejan mediante el array asociativo **`$_GET`**

http://miapp.com?nombre=Julen → $_GET[“nombre”]

Hands on!

03. Crea un programa que muestre el nombre de usuario enviado en la URL:

ejercicio03.php?**usuario=admin**

Mostrará el siguiente texto por pantalla:

Bienvenido, admin

Hands on!

04. Modifica el ejercicio 02 para que recoja el nombre y la edad desde la URL. Es decir, para probar la página la URL será como la siguiente:

ejercicio03.php?**nombre=Markel&edad=22**

Mostrará el siguiente texto por pantalla:

Mi amigo Mikel tiene 22 años.

Constantes

```
define("FOO", "valor de FOO");
```

Variables predefinidas

- Variables predefinidas y **disponibles en cualquier script.**
- Algunas de las más utilizadas:
  - $_SERVER — Información del entorno del servidor y de ejecución
  - $_GET — Variables HTTP GET
  - $_POST — Variables POST de HTTP
  - $_FILES — Variables de subida de ficheros HTTP
  - $_REQUEST — Variables HTTP Request
  - $_SESSION — Variables de sesión
  - $_ENV — Variables de entorno
  - $_COOKIE — Cookies HTTP

Operadores

```php
$numero1 = 5;
$numero2 = 6;
$suma = $numero1 + $numero2;
$multiplicacion = $numero1 * $numero2;
echo $suma;// Resultado: 11
echo $multiplicacion;// Resultado: 30
```

Operadores de 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 lógicos

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**.

Hands on!

05. Crea dos variables, a y b que recojan los valores enviados mediante el método GET. Almacena el resultado de las siguientes operaciones en nuevas variables y muéstralas por pantalla.

- Resta de *a* y *b*
- División de *a* entre *b*
- Resultado de *a* mayor que *b*
- Resultado de *a* menor o igual que *b*

Nota: para imprimir una variable con valor booleano, utiliza la función `var_export($variable)`

Funciones

```
function saludo( ) {
    echo “Hola Ane”;
}
```

Funciones

```php
function suma($num1, $num2) {
    $resultado = $num1 + $num2;
    echo "El resultado es: $resultado";
}

suma(5, 10);
```

Hands on!

06. Crea una función llamada *multiplicar* que reciba dos variables, a y b, y muestre el resultado de la multiplicación por pantalla. Para probar el ejercicio se enviarán los valores mediante una petición GET.

Devolviendo valores

```php
function cuadrado($numero){
    return $numero*$numero;
}
echo "4 al cudrado es: " . cuadrado(4);
```

Hands on!

07. ¿Cómo mejorarías el ejercicio anterior para que la función sea más reutilizable? Piénsalo bien y modifica la función.

08. Crea una función llamada `esMayor()` que reciba como parámetros dos números y devuelva TRUE si el primer número es mayor que el segundo.

Parámetros con valores por defecto

```php
function saludo($nombre = 'Anónimo') {
    echo "Hola: " . $nombre;
}

saludo("Izaskun"); // Hola: Izaskun
saludo(); // Hola: Anónimo
```

Últimas novedades en PHP

- Cada versión de PHP introduce novedades, especialmente a partir de la versión 7. Algunos ejemplos:
  - Type hints
  - Union types
  - Parámetros con nombre
  - Parámetros nullables
  - Operador “match”
  - Operador ??
  - Funciones con flechas (Arrow functions)
  - ...y más!

```php
// Type hints
function sumar(int $x, int $y): int
{
    return $x + $y;
}
```

```php
// Parámetros con nombre
function mostrarMensaje($nombre, $edad, $color){
    …
}
mostrarMensaje(
    edad: 25,
    nombre: 'Ainhoa',
    color: 'verde'
);
```

```php
//Union type (version 8)
function multiplicar($x, $y): int | float
{
    return $x * $y;
}
echo multiplicar(10, 20); // 200 (int)
echo multiplicar(1.5, 2.5); // 3.75 (float)
```

```php
// Nullable parameter (version 7.1)
function mayus(?string $str): string
{
    return strtoupper($str);
}
```

Ámbito de las variables

- Las variables están **accesibles en función de su ámbito** (*scope*):
  - Ámbito global: variables definidas fuera de funciones o clases.
    - Las variables globales **no están accesibles dentro de las funciones**.
  - Ámbito local: variables definidas dentro de funciones o clases.
    - Las variables locales **no afectan al ámbito global**.
- Es posible utilizar una variable global dentro de una función utilizando la palabra reservada **`global`** o el array **`$GLOBALS.`**
  - No se recomienda su uso. Lo recomendable es que la función reciba la variable como argumento en la llamada.

Ámbito de las variables

```
function imprimir()
{
    $mensaje = "Mensaje local";
    print $mensaje;
}

$mensaje = "Mensaje global";
imprimir(); // Output: Mensaje local
print $mensaje; // Output: Mensaje global
```

Hands on!

09. Escribe una función que reciba como parámetro dos cadenas de texto y devuelva la concatenación de dichas cadenas. Muestra el resultado obtenido por pantalla.

10. Modifica la función anterior para que establezca un valor por defecto a una de las cadenas.

Include y Require

- Permiten incluir y evaluar código ubicado en otros ficheros.
- Require es idéntico pero en caso de fallo produce un error y detiene la ejecución del script (include lanzará un warning).
- include_once y require_once: son idénticos a los anteriores pero PHP primero comprobará si el archivo ya ha sido incluido, y de ser así no lo vuelve a incluir.

Include y Require

vars.php

```php
<?php

$color = 'verde';
$fruta = 'pera';

?>
```

test.php

```php
<?php

echo "Una $fruta $color";//Una
include 'vars.php';
echo "Una $fruta $color";//Una pera verde

?>
```

Ordena tu código

- El código PHP se puede complicar, volviéndose difícil de leer si está mezclado con código HTML.
- Es una buena práctica **separar la presentación (HTML+CSS) de la lógica de la aplicación (PHP, acceso a datos, etc.)**.
- Tendremos un fichero encargado de preparar los datos que mostrará la vista.
- Una vez ha preparado los datos, cargará la vista mediante la sentencia require

index.php

```php
/* Preparar los datos */
$saludo = "Hola, mundo!");

/* Cargar la vista */
require "index.view.php";
```

index.view.php

```php
<html>
<head>
    <title>Hola Mundo PHP</title>
</head>
<body>
    <?php
        echo $saludo;
    ?>
</body>
</html>
```

Hands on!

11. Separa el código del ejercicio anterior en dos archivos distintos. Uno de ellos tendrá todo el contenido relacionado con la presentación de la página y el otro deberá contener todo aquello relacionado con los datos y la lógica de la aplicación.

Arrays simples

```
$arr1 = array( );

$arr2 = [ ];
```

Arrays simples

```php
/* Crear array mediante la función array() */
$fruta = array('Manzana', 'Pera', 'Platano');

echo "Me gusta: {$fruta[0]}, {$fruta[1]} y {$fruta[2]}.";
```

Arrays simples

```php
/* Otras formas de inicializar un array. */
$impares = [1,3,5,7,9];

$fruta[0] = 'Manzana';
$fruta[1] = 'Pera';
$fruta[2] = 'Platano';

echo "Me gusta {$fruta[0]}, {$fruta[1]} y {$fruta[2]}.";
```

Arrays simples

```
$array = [1,3,5,7,9];
function my_func($a){
    $a[] =30;
}
my_func($array);
var_dump($array); //$array no sufre cambios porque el
contenido de $array se copia en memoria en otra variable
local referenciada por $a
```

Arrays simples

```php
/* Solución 1: Paso por valor */
$array = [1,3,5,7,9];
function my_func($a){
        $a[] =30;
        return $a;
}
$salida= my_func($array);
var_dump($salida);
```

Arrays simples

```php
/* Solución 2: Paso por referencia */
$array = [1,3,5,7,9];
function my_func(& $a){
        $a[] =30;
}
my_func($array);
var_dump($array);
```

Hands on!

12. Crea un array con 4 nombres de ciudades (Paris, Berlin, Amsterdam, Praga). A continuación crea las siguientes funciones:

- `getValor($array, $posicion)`: recibe un número como parámetro y devuelve el valor almacenado en el array en dicho índice.
- `setValor($array, $posicion, $valor)`: establece el valor del elemento indicado en la posición y devuelve el array.

Crea algunos ejemplos con llamadas a las dos funciones.

Funciones útiles

unset()

```php
$impares = [1,3,5,7,9];
// eliminar el tercer elemento (5) de la lista:
unset($impares[2]);
print_r($impares);
```

count()

```php
$impares = [1,3,5,7,9];
// contar el número de elementos de un array
echo count($impares);
```

array_push()

```php
$numeros = [1,2,3];
array_push($numeros, 4);
// ahora el array es [1,2,3,4];

// imprimir el nuevo array
print_r($numeros);
```

array_pop()

```php
$numeros = [1,2,3,4];
array_pop($numeros);
// ahora el array es [1,2,3];

// imprimir el nuevo array
print_r($numeros);
```

array_unshift()

```php
$numeros = [1,2,3];
array_unshift($numeros, 0);
// ahora el array es [0,1,2,3];

// imprimir el nuevo array
print_r($numeros);
```

array_shift()

```php
$numeros = [0,1,2,3];
array_shift($numeros);
// Ahora el array es [1,2,3];

// imprimir el nuevo array
print_r($numeros);
```

array_merge()

```php
$impares = [1,3,5,7,9];

$pares = [2,4,6,8,10];

$todos = array_merge($impares, $pares);

print_r($todos);
```

sort()

```php
$numeros = [4,2,3,1,5];

sort($numeros);

// imprimir el array ordenado

print_r($numeros);
```

array_slice()

```php
$entrada = array("a", "b", "c", "d", "e");

$salida = array_slice($entrada, 2); // devuelve "c", "d", y "e"
$salida = array_slice($entrada, 1, 3); // devuelve "b", "c" y "d"
$salida = array_slice($entrada, -3, 2); // devuelve "c", y "d"
```

Hands on!

13. Crea dos arrays, uno con 4 tipos de animales ($animales) y otro con 4 nombres de colores ($colores).

- Calcula el número de elementos de cada array.
- Añade un elemento al final del array $animales utilizando una función.
- Añade un elemento al principio del array $colores utilizando una función.
- Crea un tercer array que incluya los elementos de los dos arrays.

Arrays asociativos (clave-valor)

```php
$telefonos = [
    "Aitor" => "615-235-8573",
    "Lorea" => "655-492-4856",
];

echo "Lorea: " . $telefonos["Lorea"];
```

Arrays asociativos (clave-valor)

```
$usuario = [
        "nombre" => "Ane",
        "edad" => 25,
                “estudiante” => True
];

echo "La edad de: " . $usuario["nombre"] .” es  . $usuario["edad"];
```

Añadir un elemento

```php
$telefonos = [
    "Aitor" => "615-235-873",
    "Lorea" => "655-492-456",
];

$telefonos["Mikel"] = "656-729-856";
```

array_key_exists()

```php
if (array_key_exists("Aitor", $telefonos)) {
    echo "El telefono es " . $telefonos["Aitor"];
} else {
    echo "Teléfono no encontrado";
}
```

Hands on!

14. Crea un array asociativo que incluya 5 palabras castellano y sus respectivas traducciones al inglés. Muestra por pantalla las palabras y sus traducciones en frases como esta:

La traducción de CASA en inglés es HOUSE.

Arrays multidimensionales

```php
$salarios = array(
    "Developers" => array (
        "Itziar" => 35000,
        "Unai" => 32500,
        "Itxaso" => 39000
    ),

    "Testers" => array (
        "Ander" => 30000,
        "Idoia" => 32000,
        "Mikel" => 29000
    )
);
```

Arrays multidimensionales

```php
/* Acceder a los valores */
echo "Los salarios de los desarrolladores son : " ;
echo $salarios[Developers]['Itziar'] . "<br />";
echo $salarios[Developers]['Unai'] . "<br />";
```

Hands on!

15. Crea un array multidimensional llamado “diccionario” que almacene para cada usuario su nombre, apellidos e email. A continuación crea una función llamada “`getDatos()`” que reciba como primer parámetro el nombre de usuario y como segundo parámetro el dato a obtener (“nombre”, “apellidos” o “email”) y lo muestre por pantalla.

`getDatos($diccionario,“jvadillo”,”email”)` ➔ jvadillo@egibide.org

Estructuras de control

- Sentencias condicionales
  - If … else
  - switch
- Bucles
  - for
  - while
  - do … while
  - foreach

if ... else

```
if (condicion) {
    /* código */
} else {
    /* código */
}
```

if ... else

```php
$edad = 14;

if ($edad >= 18) {
    echo "Eres mayor de edad";
} else {
    echo "Eres menor de edad";
}
```

elseif

```
if (condicion) {
    /* código */
} elseif (condicion2) {
    /* código */
} else {
    /* código */
}
```

elseif

```php
$edad = 14;

if ($edad <= 18) {
    echo "Eres mayor de edad";
} else {
    echo "Eres menor de edad";
}
```

Hands on!

16. Crea una función que reciba 2 números obtenidos mediante GET. La aplicación deberá realizar la siguiente operación:

- Si los números son distintos, mostrará el resultado de su suma.
- Si los números son iguales, mostrará el resultado de su multiplicación.

Hands on!

17. Crea una función que reciba un usuario y contraseña mediante GET. La aplicación deberá mostrar si el usuario existe, y en caso de existir si la contraseña recibida es correcta. La validación se realizará contra un array multidimensional como el siguiente:

```
$usuarios = [
    "user1" => [
        "nombre" => "Nora",
        "password" => "123123",
        "email" => "nora@php.net"
    ],
    ...
];
```

switch

```
switch (variable) {
    case valor1:

        sentencias
            break;
    case valor2:

        sentencias
            break;
    default:

        sentencias
```

switch

```php
$pais = "Francia";

switch ($pais) {
    case "España":
        echo "Bienvenido";
        break;
    case "Inglaterra":
        echo "Welcome";
        break;
    case "Francia":
        echo "Bienvenue";
        break;
    default:
        echo "Welcome";
}
```

Hands on!

18. Crea una función que reciba un número indicando el día de la semana y que muestre por pantalla el día de la semana. En caso de recibir otro número que no esté entre el 1 y el 7, mostrará el mensaje “No es ningún día de la semana". Utiliza un switch para realizar el ejercicio.

diaSemana(2) → “El día número 2 es martes.”

diaSemana(9) → “No es ningún día de la semana”

for

```
for (inicialización; condición; incremento) {
        // código a ejecutar
}
```

for

```php
for ($x = 0; $x < 10; $x++) {
    echo "El número actual es: {$x} <br>";
}
```

break

```php
for ($x = 0; $x < 10; $x++) {
    if ($x == 5) {
        break; //Finaliza el for
    }
    echo "El número es: $x <br>"; //Se ejecuta del 0 al 4
}
```

continue

```php
for ($x = 0; $x < 10; $x++) {
    if ($x == 5) {
        continue; //Salta a la siguiente iteración
    }
    echo " $x "; //0 1 2 3 4 6 7 8 9
}
```

Hands on!

19. Crea un programa que reciba un número y realice la suma de todos los números entre el 0 y el número dado (este incluido). Deberá mostrar el resultado.

20. Modifica el programa anterior para que sume únicamente los números pares.

21. Modifica el programa anterior de forma que en el momento en el que la suma sea mayor que 100, devuelva el último valor antes de superar 100.

Hands on!

22. Crea un array con nombres de países y a continuación una función que reciba el nombre de un país y recorra el array, comprobando valor por valor si alguno de ellos es igual al del parámetro recibido. Devolverá la posición en la que se encuentre (en caso de no encontrarlo devolverá -1)..

["Brasil", "Portugal", "Islandia", "Mexico", "Filipinas", "Marruecos"]

encontrarPosicion("Islandia") → 2

encontrarPosicion("Dinamarca") → -1

Nota: Puedes utilizar la función count() para conocer el tamaño del array que debes recorrer.

Hands on!

23. Crea un array con un listado de estudiantes (Ane, Markel, Nora, Danel, Amaia, Izaro). A continuación recorre el array mediante un *FOR*, generando una lista HTML como la siguiente:

- Ane
- Markel
- Nora
- Danel
- Amaia
- Izaro

Ahora añade un atributo “id” a cada elemento de la lista. Por ejemplo:

```
<li id="1">Ane</li>
```

Hands on!

24. Crea un array multidimensional que simule una agenda de contactos (contendrá de cada persona su nombre, apellidos, teléfono y dirección de correo electrónico). Implementa una función que imprima en una tabla los datos de las personas utilizando la sentencia *for*.

Nota: Puedes utilizar la función count() para conocer el tamaño del array que debes recorrer.

Nombre	Apellidos	Teléfono	Email
Amaia	Gorbea Jainaga	945010101	agorbea@php.net
Ane	Larrain Ogeta	945010101	alarrain@php.net
Maite	Murgiondo Lekue	945010102	mmurgiondo@php.net
Lorea	Aranceta Otxoa	945010102	laranceta@php.net
Markel	Gurrutxaga Abarra	945010102	mgurrutxaga@php.net
Urtzi	Iriondo Baiona	945010102	uriondo@php.net

```php
$agenda = [
    [
        "nombre" => "Amaia",
        "apellidos"=>"Gorbea Jainaga",
        "telefono"=>"945111111",
        "email"=>"agorbea@php.net"
    ],
    ...
];
```

while

```
while (condicion) {
        //código a ejecutar
}
```

while

```php
$x = 1;

while($x <= 10) {
    echo "El número actual es: {$x} <br>";
    $x++;
}
```

Hands on!

25. Crea un array con un listado de estudiantes (Ane, Markel, Nora, Danel, Amaia, Izaro). A continuación recorre el array utilizando una estructura de control WHILE generando una lista HTML como la siguiente:

- Ane
- Markel
- Nora
- Danel
- Amaia
- Izaro

do ... while

```
do {
        //código a ejecutar
} while (condicion);
```

do ... while

```php
do {
    echo "El número actual es: {$x} <br>";
    $x++;
} while($x <= 10);
```

Hands on!

26. Crea un array con marcas de coches y una función que imprima por pantalla una lista con todos los nombres utilizando la sentencia do-while.

- Audi
- Seat
- Mercedes
- Volkswagen
- BMW
- Fiat

foreach

```php
foreach ($array as $value) {
    //código a ejecutar
}

foreach ($array as $key => $value) {
    //código a ejecutar
}
```

foreach

```php
$paises = array("España", "Francia",
"Inglaterra", "Portugal");

foreach ($paises as $value) {
   echo "El país es: {$value} <br>";
}
```

Hands on!

27. Crea un array con el nombre de 8 grupos de música y una función que imprima por pantalla todos los elementos utilizando la estructura foreach.

28. Modifica el ejercicio número 26 para obtener el mismo resultado empleando una estructura *foreach*.

29. Modifica el ejercicio número 24 para obtener el mismo resultado empleando una estructura *foreach*.

foreach

```php
$capitales = array(
    'España' => 'Madrid',
    'Francia' => 'Paris',
    'Inglaterra' => 'Londres'
);

foreach ($capitales as $key => $value) {
    echo "La capital de {$key} es {$value} </br>";
}
```

Hands on!

30. Crea un array asociativo con el nombre de 6 estudiantes y la nota media de cada uno. Recorre el array utilizando la estructura foreach y mostrando una lista con frases con la siguiente estructura: “La nota media de Julen es 6.8”

- La nota media de Amaia es 6.5
- La nota media de Ane es 7.5
- La nota media de Maite es 5.2
- La nota media de Lorea es 9.1
- La nota media de Markel es 6.8
- La nota media de Urtzi es 8.4

Repasa lo aprendido

31. Crea una aplicación que genere un array con 20 números aleatorios con valores entre 1 y 999, utilizando la función `random_int(1, 999)` para generar los valores. A continuación recorre el array utilizando la sentencia `foreach` y muestra por pantalla el valor más alto y el más bajo.

- El array generado es: 891,590,309,903,664,159,955,678,59,824,256,787,757,955,823,802,663,757,807,379
- El valor más bajo del array es 59
- El valor más alto del array es 955

Repasa lo aprendido

32. Crea un array asociativo que almacene los nombres de 4 estudiantes y las notas de dos exámenes. A continuación, crea de forma dinámica una tabla como la de la imagen utilizando los valores del array y la estructura de repetición que creas conveniente. Cuando la nota sea inferior a 5, esta deberá mostrarse en rojo.

Nombre	Nota 1	Nota 2	Nota media
Luis Scola	9	8	8.5
Pablo Prigioni	8	4	6
Sergi Vidal	7	6	6.5
Ramón Rivas	3.5	6	4.75

Sources

- The PHP Group: https://www.php.net/
- learn-php: http://www.learn-php.org
- W3 Schools: http://w3schools.com/