

Bloque B

Páginas Web Dinámicas

Unidad 5. Funciones integratedas



Contenidos

1. Introducción
2. Funciones de cadenas
3. Funciones numéricas
4. Funciones de arrays
5. Constantes
6. Función header()
7. Funciones de archivos
8. Resumen
9. Actividad propuesta
10. Referencias



1. Introducción

Introducción

Esta unidad introduce un conjunto de funciones que están incorporadas en el intérprete PHP. Cada función realiza una tarea específica.

Introducción

Las definiciones de las funciones incorporadas están integradas en el intérprete PHP, lo que significa que no necesitan ser incluidas en una página PHP antes de ser llamadas.

Fueron diseñadas para realizar tareas que los desarrolladores web a menudo necesitan realizar al crear páginas web dinámicas y evitarles tener que escribir sus propias funciones para realizar esas tareas.

Introducción

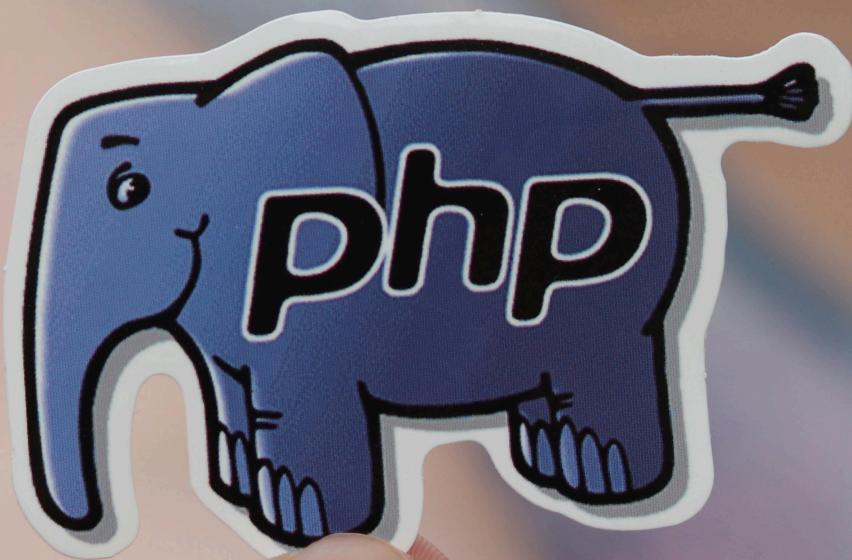
Para llamar a una función incorporada, se necesita saber su nombre, qué parámetros tiene y qué datos devolverá. Por lo tanto, esta unidad contiene varias tablas que muestran los nombres de las funciones y los parámetros, seguidos de descripciones de las funciones, y qué valores devuelven.

Introducción

El primer conjunto de funciones se agrupa en función de los tipos de datos con los que se utiliza para trabajar: cadenas, números y arrays. Más adelante en la unidad, también verás:

- Cómo crear constantes (que son como variables cuyos valores no pueden cambiar una vez establecidos).
- Una función que se utiliza para controlar las cabeceras HTTP que se envían de vuelta al navegador cuando el intérprete PHP devuelve una página que ha sido solicitada.
- Un conjunto de funciones que te permiten obtener información sobre archivos en el servidor.

Las funciones que conozcas en esta unidad serán utilizadas a lo largo del resto de este módulo.



2. Funciones de cadenas

Mayúsculas/minúsculas y comprobación de la longitud

Estas funciones transforman el texto en mayúsculas o minúsculas y cuentan el número de caracteres o palabras de la cadena.

Las siguientes funciones están diseñadas para trabajar con texto (el tipo de datos cadena). Toman una cadena como argumento, la actualizan y devuelven la cadena modificada.

Por ejemplo, la función `strtolower()` toma una cadena y convierte todo el texto a minúsculas. A continuación, devuelve el valor actualizado.

Mayúsculas/minúsculas y comprobación de la longitud

Cambio de mayúsculas y minúsculas

FUNCTION	DESCRIPTION
<code>strtolower(\$string)</code>	Returns a string with all characters in lowercase.
<code>strtoupper(\$string)</code>	Returns a string with all characters in uppercase.
<code>ucwords(\$string)</code>	Returns a string with the first letter of every word in uppercase.

La función `ucwords()` se llama así porque es un acrónimo de "Uppercase Words"

Mayúsculas/minúsculas y comprobación de la longitud

Contar caracteres y palabras

FUNCTION	DESCRIPTION
<code>strlen(\$string)</code>	Returns the number of characters in the string. Spaces and punctuation count as characters. (See also <code>mb_strlen()</code> on p210-211 for multibyte characters.)
<code>str_word_count(\$string)</code>	Returns the number of words in the string.

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

El siguiente ejemplo realiza varias operaciones sobre la cadena de texto '*Home sweet home*' y luego incluye dos archivos, `header.php` y `footer.php`, para estructurar la salida HTML.

1. Una cadena que dice "Home sweet home" se almacena en una variable llamada `$texto`. Esto se utilizará como argumento cuando se llame a cada una de las funciones.
2. Se llama a la función `strtolower()`. La función convierte el texto a minúsculas y devuelve ese valor. El valor que devuelve la función se escribe en la página utilizando la abreviatura del comando echo.

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

3. La función `strtoupper()` devuelve la cadena en mayúsculas.
4. La función `ucwords()` devuelve la cadena con la primera letra de cada palabra en mayúsculas.
5. La función `strlen()` cuenta cuántos caracteres hay en la cadena y devuelve ese número.
6. La función `str_word_count()` cuenta el número de palabras de la cadena y devuelve ese número.

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

```
<?php  
① $text = 'Home sweet home';  
?>  
<?php include 'includes/header.php'; ?>  
<p>  
    <b>Lowercase:</b>  
② <?= strtolower($text) ?><br>  
    <b>Uppercase:</b>  
③ <?= strtoupper($text) ?><br>  
    <b>Uppercase first letter:</b>  
④ <?= ucwords($text) ?><br>  
    <b>Character count:</b>  
⑤ <?= strlen($text) ?><br>  
    <b>Word count:</b>  
⑥ <?= str_word_count($text) ?>  
    </p>  
<?php include 'includes/footer.php'; ?>
```

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

Lowercase: home sweet home

Uppercase: HOME SWEET HOME

Uppercase first letter: Home Sweet Home

Character count: 15

Word count: 3

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

Crea un pequeño script que podría ser parte de una aplicación de blog o gestión de contenidos, donde el texto de las entradas de blog necesita ser procesado y analizado antes de mostrarse.

El script transformará y analizará el texto utilizando las funciones `strtoupper()`, `ucwords()`, `strlen()` y `str_word_count()`, y presentará los resultados de una manera que podría ser útil para los administradores del blog.

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

Pasos a realizar:

1. Define una cadena de texto que simule el contenido de una entrada de blog.
2. Aplica las funciones `strtoupper()`, `ucwords()`, `strlen()` y `str_word_count()` a la cadena.
3. Calcula la longitud del texto sin contar los espacios.

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

4. Presenta un resumen del análisis que incluya:

- El texto original.
- El texto en mayúsculas.
- El texto con cada palabra capitalizada.
- La longitud del texto en caracteres.
- La longitud del texto en caracteres sin espacios.
- La cantidad de palabras en el texto.

Ejemplo: conversión mayúsculas/minúsculas y contando caracteres

Extensión de la Actividad:

Añade funcionalidades adicionales, como:

- Contar la frecuencia de cada palabra en el texto.
- Detectar y marcar palabras clave predefinidas (por ejemplo, "importante", "procesar").
- Generar un resumen del texto mostrando solo las primeras 50 palabras.

Búsqueda de caracteres en una cadena

Estas funciones buscan uno o varios caracteres en una cadena. Si encuentran una coincidencia, devuelven la posición de ese carácter. Si no encuentran ninguna coincidencia, devuelven *false*.

Búsqueda de caracteres en una cadena

Cada carácter de una cadena tiene una **posición**; un número que empieza por 0.

Así, el primer carácter está en la posición 0, el segundo en la 1, y así sucesivamente.



Búsqueda de caracteres en una cadena

Cuando se busca un conjunto de caracteres dentro de una cadena, esos caracteres se denominan **subcadena**.

Algunas de las funciones **distinguen entre mayúsculas y minúsculas (case-sensitive)**, por lo que sólo se encuentra una coincidencia si la cadena y la subcadena tienen la misma combinación de caracteres en mayúsculas y minúsculas.

Las tres últimas funciones, marcadas con un asterisco fueron añadidas en PHP 8; todas ellas distinguen entre mayúsculas y minúsculas.

NOTA: Los parámetros opcionales se muestran entre corchetes.

Búsqueda de caracteres en una cadena

FUNCTION	DESCRIPTION
<code>strpos(\$string, \$substring[, \$offset])</code>	Returns position of first match for substring (case-sensitive). If offset is used, it only looks <i>after</i> this character position.
<code>stripos(\$string, \$substring[, \$offset])</code>	Case-insensitive version of strpos().
<code>strrpos(\$string, \$substring[, \$offset])</code>	Returns position of last match for substring (case-sensitive).
<code>strripos(\$string, \$substring[, \$offset])</code>	Case-insensitive version of strrpos().
<code>strrstr(\$string, \$substring)</code>	Returns text from first occurrence of a substring (including the substring) to the end of the string (case-sensitive).
<code>stristr(\$string, \$substring)</code>	Case-insensitive version of strrstr().
<code>substr(\$string, \$offset[, \$characters])</code>	Returns characters from the position specified in \$offset to the end of the string. If the \$characters parameter is used, it specifies the number of characters to return after \$offset.
<code>* str_contains(\$string, \$substring)</code>	Checks if a substring is found in a string, returns true/false.
<code>* str_starts_with(\$string, \$substring)</code>	Checks if string starts with substring, returns true/false.
<code>* str_ends_with(\$string, \$substring)</code>	Checks if string ends with substring, returns true/false.

Ejemplo: comprobación de caracteres en una cadena

El siguiente ejemplo define una cadena y usa varias funciones de PHP para buscar subcadenas y extraer partes de la cadena original.

Las funciones incluyen búsquedas sensibles e insensibles a mayúsculas y minúsculas, búsqueda de la primera y última aparición de subcadenas, y extracción de subcadenas.

Los resultados se muestran en una página HTML entre un encabezado y un pie de página incluidos externamente.

Ejemplo: comprobación de caracteres en una cadena

1. La cadena *Home sweet home* se almacena en `$text`.
2. Se llama a la función `strpos()` para buscar la primera vez que aparece la subcadena *ho* en la cadena. Devuelve 11.
3. Se llama a la función `stripos()` para buscar la primera vez que aparece la subcadena *me* después de la posición 5. Devuelve 13.
4. Se llama a la función `strrpos()` para encontrar la última vez que se encuentra la subcadena *Ho*. Devuelve 0 porque distingue entre mayúsculas y minúsculas.

Ejemplo: comprobación de caracteres en una cadena

5. Se llama a la función `stripos()` para encontrar la última vez que se encontró la subcadena *Ho*. Devuelve 11 porque no distingue entre mayúsculas y minúsculas.
6. Se llama a la función `strstr()` para obtener el texto de la primera aparición de la subcadena *ho*. Devuelve *home*.
7. Se llama a la función `stristr()` para obtener el texto de la primera aparición de *ho*. Devuelve *Home sweet home* porque no distingue entre mayúsculas y minúsculas.
8. Se llama a la función `substr()` y devuelve cinco caracteres, empezando por el carácter en la quinta posición.

Ejemplo: comprobación de caracteres en una cadena

```
<?php  
① $text = 'Home sweet home';  
    ?> ...  
    <b>First match (case-sensitive):</b>  
② <?= strpos($text, 'ho') ?><br>  
    <b>First match (not case-sensitive):</b>  
③ <?= stripos($text, 'me', 5) ?><br>  
    <b>Last match (case-sensitive):</b>  
④ <?= strrpos($text, 'Ho') ?><br>  
    <b>Last match (not case-sensitive):</b>  
⑤ <?= strripos($text, 'Ho') ?><br>  
    <b>Text after first match (case-sensitive):</b>  
⑥ <?= strstr($text, 'ho') ?><br>  
    <b>Text after first match (not case-sensitive):</b>  
⑦ <?= striistr($text, 'ho') ?><br>  
    <b>Text between two positions:</b>  
⑧ <?= substr($text, 5, 5) ?><br>
```

Ejemplo: comprobación de caracteres en una cadena

First match (case-sensitive): 11

First match (not case-sensitive): 13

Last match (case-sensitive): 0

Last match (not case-sensitive): 11

Text after first match (case-sensitive): home

Text after first match (not case-sensitive): Home sweet home

Text between two positions: sweet

Ejemplo: comprobación de caracteres en una cadena

Desarrolla un script en PHP que analice y manipule el contenido de un artículo web. El script deberá identificar ciertas subcadenas dentro del texto y realizar diferentes operaciones basadas en la detección de estas subcadenas.

Ejemplo: comprobación de caracteres en una cadena

Instrucciones:

1. Definir el Contenido:

- Define una cadena de texto que simule el contenido de un artículo web.

2. Detección y Análisis de Subcadenas:

- Detectar la primera y última aparición de una palabra específica.
- Comprobar si el artículo contiene ciertas palabras clave.
- Extraer partes del texto basadas en subcadenas específicas.
- Comprobar si el texto comienza o termina con ciertas palabras.

3. Presentación de Resultados:

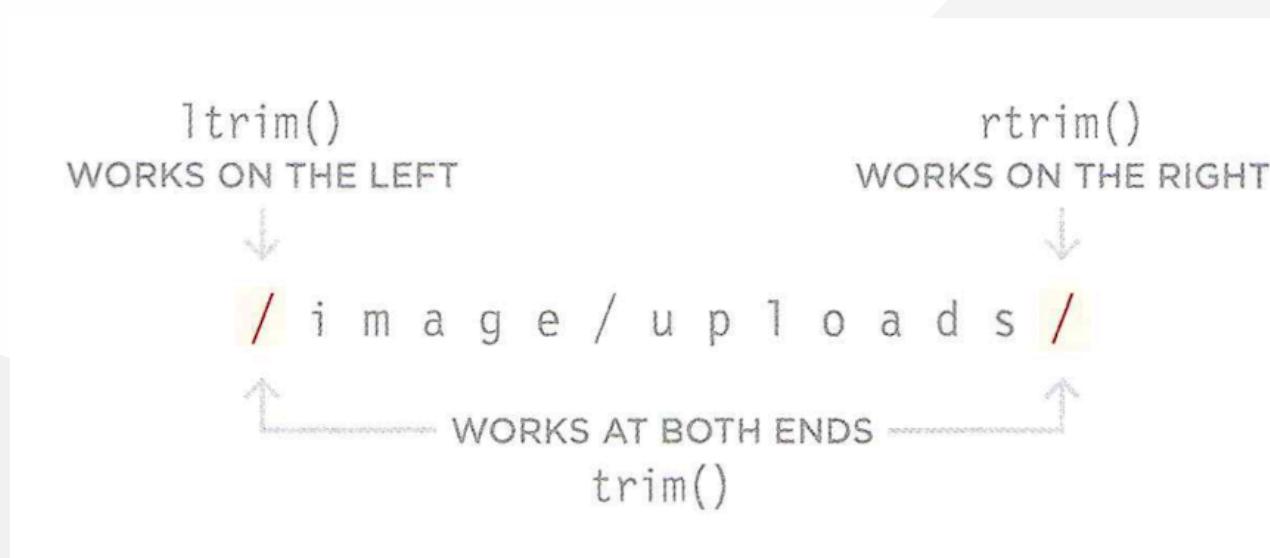
- Mostrar los resultados del análisis de manera clara y ordenada.

Eliminar y sustituir caracteres

Estas funciones pueden eliminar caracteres específicos (incluidos los espacios en blanco), sustituir caracteres (como una herramienta de búsqueda y sustitución) y repetir una cadena un número determinado de veces.

Eliminar y sustituir caracteres

Las funciones de **recorte** (`trim`) eliminan caracteres de una cadena. Pueden buscar caracteres específicos al principio y/o al final de la cadena y eliminarlos si están presentes.



Si no especifica los caracteres que desea eliminar, las funciones de recorte eliminan cualquier espacio en blanco que se encuentre al principio y/o al final de la cadena, incluidos espacios, tabulaciones (`\t`), retornos de carro (`\r`) y saltos de línea (retornos suaves `\n`).

Eliminar y sustituir caracteres

Las funciones de **sustitución** buscan caracteres en una cadena. Si encuentra una coincidencia, sustituye esos caracteres por otros nuevos.

La función **repetir** repite una cadena un número fijo de veces.

Eliminar y sustituir caracteres

FUNCTION	DESCRIPTION
<code>ltrim(\$string[, \$delete])</code>	Remove whitespace from the left-hand side of the string. If specified, \$delete provides a set of characters that should be removed if they are found at the start of the string. It is case-sensitive.
<code>rtrim(\$string[, \$delete])</code>	Removes whitespace from the right-hand side of the string.
<code>trim(\$string[, \$delete])</code>	Removes whitespace from the left and right-hand sides of the string.
<code>str_replace(\$old, \$new, \$string)</code>	Replaces the substring \$old with the one in \$new (case-sensitive).
<code>str_ireplace(\$old, \$new, \$string)</code>	Replaces the substring \$old with the one in \$new (case-insensitive).
<code>str_repeat(\$string, \$repeats)</code>	Repeats the string a specified number of times.

Ejemplo: reemplazando caracteres en una cadena

El siguiente ejemplo manipula una cadena de texto ('/images/uploads/') realizando varias operaciones: elimina caracteres específicos con `trim`, `ltrim`, y `rtrim`, reemplaza subcadenas con `str_replace` y `str_ireplace`, y repite la cadena con `str_repeat`.

Ejemplo: reemplazando caracteres en una cadena

1. La ruta '*/images/uploads/*' es una cadena que se almacena en una variable llamada `$text`.
2. La función `trim()` devuelve la cadena con `/` eliminado de la izquierda y derecha del texto.
3. La función `ltrim()` devuelve la cadena con `/` eliminado de la parte izquierda.
4. La función `rtrim()` devuelve la cadena con `s/` eliminado del lado derecho.

Ejemplo: reemplazando caracteres en una cadena

5. La función `str_replace()` devuelve la cadena con las letras `images` sustituidas por las letras `img`. Distingue entre mayúsculas y minúsculas.
6. La función `str_ireplace()` devuelve la cadena con las letras `IMAGES` sustituidas por las letras `img`. La búsqueda por subcadena distingue entre mayúsculas y minúsculas, por lo que encontraría tanto `IMAGES` como `images` y las sustituiría por `img`.
7. La función `str_repeat()` devuelve la cadena con todos los caracteres repetidos dos veces.

Ejemplo: reemplazando caracteres en una cadena

```
<?php  
① $text = '/images/uploads/';  
    ?> ...  
    <b>Remove '/' from both ends:</b><br>  
② <?= trim($text, '/') ?><br>  
    <b>Remove '/' from the left of the string:</b><br>  
③ <?= ltrim($text, '/') ?><br>  
    <b>Remove 's/' from the right of the string:</b><br>  
④ <?= rtrim($text, 's/') ?><br>  
    <b>Replace 'images' with 'img':</b><br>  
⑤ <?= str_replace('images', 'img', $text) ?><br>  
    <b>As above but case-insensitive:</b><br>  
⑥ <?= str_ireplace('IMAGES', 'img', $text) ?><br>  
    <b>Repeat the string:</b><br>  
⑦ <?= str_repeat($text, 2) ?></p>
```

Ejemplo: reemplazando caracteres en una cadena

Remove '/' from both ends:

images/uploads

Remove '/' from the left of the string:

images/uploads/

Remove 's/' from the right of the string:

/images/upload

Replace 'images' with 'img':

/img/uploads/

As above but case-insensitive:

/img/uploads/

Repeat the string:

/images/uploads//images/uploads/

Ejemplo: reemplazando caracteres en una cadena

Escribe un script PHP que simule la entrada de datos de un usuario y luego utilice las funciones de manipulación de cadenas para limpiar y formatear estas entradas.

Ejemplo: reemplazando caracteres en una cadena

Instrucciones

1. Configura el Script PHP:

- Crea un archivo PHP llamado `procesar_datos.php`.
- Define variables para simular datos de entrada de un usuario: `$nombre`,
`$correo`, y `$mensaje`.

2. Muestra los Datos Originales:

- Utiliza `echo` para mostrar en pantalla los datos originales.

Ejemplo: reemplazando caracteres en una cadena

3. Realiza las Siguientes Operaciones de Manipulación de Cadenas:

- Eliminar Espacios en Blanco Adicionales: Utiliza `trim()` para eliminar los espacios en blanco al inicio y al final de las cadenas.
- Asegurar que el Correo Está en Minúsculas: Aplica `strtolower()` para convertir toda la cadena a minúsculas.
- Reemplazar Ciertas Palabras en el Mensaje: Utiliza `str_replace()` para reemplazar palabras inapropiadas con "*****".
- Reemplazo Insensible a Mayúsculas/Minúsculas: Emplea `str_ireplace()` para reemplazar "urgente" con "Prioridad Alta".
- Repetir una Cadena para Enfatizar: Añade "!!!" al final del mensaje usando `str_repeat()`.

Ejemplo: reemplazando caracteres en una cadena

4. Muestra los Datos Procesados:

- Utiliza echo para mostrar en pantalla los datos después de haber sido procesados.

Funciones de cadena multibyte

Algunas de las funciones de cadena que hemos visto hasta ahora devuelven un resultado erróneo si se utilizan con caracteres multibyte.

Las funciones de cadena multibyte que se muestran a continuación admiten todos los caracteres de UTF-8.

Funciones de cadena multibyte

Cuando el texto se codifica utilizando UTF-8, algunos caracteres utilizan más de un byte de datos. Por ejemplo, el símbolo £ utiliza dos bytes y € utiliza tres.

Si utiliza caracteres multibyte como argumentos para algunas de las funciones de cadena, pueden producir un resultado incorrecto (en el ejemplo a continuación se muestran casos de resultados inexactos).

Funciones de cadena multibyte

Las funciones de cadena multibyte que se muestran a continuación tienen los mismos nombres que las funciones que has conocido hasta ahora en este capítulo, pero van precedidas de los caracteres *mb*.

Algunas funciones de cadena no tienen equivalente multibyte, por ejemplo, `trim()` y `str_replace()`. Estas funcionan con UTF-8 siempre que se haya establecido como codificación de caracteres por defecto en los archivos `php.ini` o `.htaccess`.

Funciones de cadena multibyte

FUNCTION	DESCRIPTION
<code>mb_strtoupper(\$string)</code>	Returns string with all characters in uppercase.
<code>mb_strtolower(\$string)</code>	Returns string with all characters in lowercase.
<code>mb_strlen(\$string)</code>	Returns the number of characters in the string.
<code>mb_strpos(\$string, \$substring[, \$offset])</code>	Returns the position of the first place the substring is found (case-sensitive). If an <code>\$offset</code> is specified, it only looks after this character position.
<code>mb_stripos(\$string, \$substring[, \$offset])</code>	Case-insensitive version of <code>mb_strpos()</code> .
<code>mb_strrpos(\$string, \$substring[, \$offset])</code>	Returns position of last match for substring (case-sensitive).
<code>mb_stripos(\$string, \$substring[, \$offset])</code>	Case-insensitive version of <code>mb_strrpos()</code> .
<code>mb_substr(\$string, \$start[, \$characters])</code>	Returns text from the first occurrence of a substring (including the substring) to the end of the string (case-sensitive).
<code>mb_stristr(\$string, \$substring)</code>	Case-insensitive version of <code>mb_substr()</code> .
<code>mb_substr(\$string, \$start[, \$characters])</code>	Returns characters from position specified in <code>\$start</code> to the end of the string. If <code>\$characters</code> is specified, it returns this number of characters after <code>\$start</code> .

Ejemplo: Uso de funciones de cadenas multibyte

Este ejemplo utiliza funciones de cadena con el símbolo £ , que UTF-8 necesita dos bytes de datos para codificar.

1. Se crea una cadena utilizando el símbolo £ y se almacena en \$text . Tiene una longitud de 11 caracteres.
2. La función strlen() funciona contando el número de bytes necesarios para representar una cadena, no el número de caracteres de la cadena. Por eso dice que hay 12 caracteres en la cadena (no 11).

Ejemplo: Uso de funciones de cadenas multibyte

3. La función `mb_strlen()` tiene en cuenta la codificación que utiliza el intérprete de PHP y muestra el número correcto de caracteres de la cadena como 11.
4. La función `strpos()` encuentra la primera posición de 444. La posición se calcula utilizando el número de bytes antes de encontrar la subcadena (no el número de caracteres). Devuelve 9, en lugar de 8.
5. `mb strpos()` encuentra la primera posición de 444, devolviendo correctamente el número 8.

Ejemplo: Uso de funciones de cadenas multibyte

```
<?php  
① $text = 'Total: £444';  
    ?> ...  
    <b>Character count using <code>strlen()</code>:</b>  
② <?= strlen($text) ?><br>  
    <b>Character count using <code>mb_strlen()</code>:</b>  
③ <?= mb_strlen($text) ?><br>  
    <b>First match of 444 <code>strpos()</code>:</b>  
④ <?= strpos($text, '444') ?><br>  
    <b>First match of 444 <code>mb_strpos()</code>:</b>  
⑤ <?= mb_strpos($text, '444') ?><br>
```

Ejemplo: Uso de funciones de cadenas multibyte

Character count using `strlen()`: 12

Character count using `mb_strlen()`: 11

First match of 444 `strpos()`: 9

First match of 444 `mb_strpos()`: 8

Ejemplo: Uso de funciones de cadenas multibyte

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- Modifica el texto en la variable `$text` del siguiente código para incluir caracteres de distintos idiomas (por ejemplo, chino, japonés) y explica cómo se comportan las funciones.

Expresiones regulares

Los números de tarjetas de crédito, códigos postales y números de teléfono utilizan patrones específicos de caracteres.

Las expresiones regulares describen un patrón de caracteres y PHP tiene funciones integradas para comprobar si esos patrones se encuentran en una cadena.

Expresiones regulares

Las expresiones regulares se sitúan entre dos barras diagonales. El patrón de la expresión siguiente describe:

- [A–z] Las letras A–z (mayúsculas/minúsculas)
- {3,9} que aparecen entre 3 y 9 veces

/[A–z]{3,9}/

Expresiones regulares

Si se utilizara esta expresión para comprobar la cadena "*Thomas was 1st!*", el intérprete de PHP encontraría la primera vez que hay una secuencia de 3-9 caracteres que utilizan las letras mayúsculas o minúsculas A-z.

Estos caracteres se destacan a continuación:



```
T h o m a s   w a s   1 s t !
```

Expresiones regulares

La sintaxis de las expresiones regulares puede ser bastante compleja, y hay libros enteros sobre cómo escribirlas, pero a continuación veremos lo básico para trabajar con ellas.

La siguiente tabla muestra cómo buscar caracteres específicos, un rango de caracteres y caracteres al principio o al final de una cadena.

Expresiones regulares

EXPRESSION	DESCRIPTION	EXAMPLE
/1st/	Matches the characters 1st	Thomas was 1st!
/[abcde]/	If characters are in square brackets, it matches any one of the letters; this matches any letter a, b, c, d or e	Thom a s was 1st!
/[K-Z]/	A hyphen in square brackets creates a range of characters to match This matches any uppercase character between K and Z	Thom a s was 1st!
/[a-e]/	This matches any lowercase character between a and e	Thom a s was 1st!
/[0-9]/	This matches any number between 0 and 9	Thomas was 1st!
/[A-z0-9]/	This matches any upper or lowercase letter A-z or number 0-9	Thom a s was 1st!
/^ [A-Z]/	A caret ^ at the start of a pattern indicates that the string must start with these characters; this matches if the first character is A-Z	Thom a s was 1st!
/1st\!\$/	A dollar \$ at the end of a pattern indicates the string must end with the specified characters; this matches if the last characters are 1st!	Thomas was 1st!
/\s/	Matches a space	Thomas was 1st!

Expresiones regulares

Los siguientes caracteres tienen un significado especial en las expresiones regulares:

```
\ / . | $ ( ) ^ ? { } + *
```

Para crear un patrón que coincida con cualquiera de estos caracteres, debemos colocar una **barra invertida** (`\`) antes del carácter.

EXPRESSION	DESCRIPTION	EXAMPLE
<code>/[\!?\(\)]/</code>	Matches an exclamation mark, question mark or parentheses	Thomas was 1st!

Expresiones regulares

Puede añadir un **cuantificador** para especificar el número de veces que debe aparecer un patrón en la cadena.

Los ejemplos siguientes buscan caracteres que aparecen un número determinado de veces.

EXPRESSION	DESCRIPTION	EXAMPLE
/[a-z]+/	The plus sign + indicates one or more of the specified characters	T h o m a s was 1st !
/[a-z]{3}/	A number in curly braces {} indicates exact number of times the pattern must be found	T h o m a s was 1st !
/[A-z]{3,5}/	Two numbers separated by a comma in curly braces {} indicate the minimum and maximum number of times the pattern must be found	T h o m a s was 1st !
/[a-z]{3,}/	A number, then a comma (without a second number) in curly braces {} is the minimum number of times the pattern can be found	T h o m a s was 1st !

Expresiones regulares

Para buscar una secuencia de patrones, utiliza un patrón seguido de otro.

En el siguiente ejemplo, una coincidencia con el primer patrón debe ir seguida de una coincidencia con el segundo patrón.

EXPRESSION	DESCRIPTION	EXAMPLE
/[0-9][a-z]/	Matches a number 0-9 followed by a lowercase letter a-z	Thomas was 1st!

Expresiones regulares

Colocando paréntesis alrededor de parte de una expresión se crea un **grupo**. Se puede añadir un cuantificador después del grupo para indicar cuántas veces debe aparecer.

Si se desea encontrar una de un conjunto de opciones, se pueden especificar las opciones dentro de un grupo, y separar cada opción con un carácter `|`.

EXPRESSION	DESCRIPTION	EXAMPLE
<code>/[0-9]([a-z]{2})/</code>	[0-9] matches any number 0-9; it is followed by ([a-z]{2}) which matches two lowercase letters	Thomas was 1st!
<code>/[1-31](st nd rd th)/</code>	[1-31] matches any number 1-31 followed by (st nd rd th) which matches st, nd, rd or th	Thomas was 1st!

Funciones de expresiones regulares

Estas funciones comprueban si una cadena contiene un patrón de caracteres descrito por una expresión regular.

Si encuentran una coincidencia, cada una realizará una tarea diferente.

Funciones de expresiones regulares

Todas las funciones siguientes utilizan expresiones regulares para buscar un patrón específico de caracteres en una cadena.

Realizan tareas como:

- Comprobar si se encuentra el patrón
- Contar cuántas veces se encuentra un patrón
- Buscar el patrón de caracteres y sustituirlo por un nuevo conjunto de caracteres (como una función de búsqueda y sustitución en un procesador de textos).

Funciones de expresiones regulares

En cada caso, la función tiene parámetros para la:

- La expresión regular que describe el patrón de caracteres que está buscando
- La cadena en la que busca ese patrón

La función que reemplazará un conjunto de caracteres coincidentes por un nuevo conjunto de caracteres también necesita saber con qué reemplazar esos caracteres.

NOTA: Los nombres de estas funciones comienzan con el prefijo `preg` (que significa *Perl regular expressions*) porque las expresiones regulares que utiliza PHP siguen las de otro lenguaje de programación llamado Perl.

Funciones de expresiones regulares

FUNCTION	DESCRIPTION
<code>preg_match(\$regex, \$string)</code>	Looks for a matching pattern in a string. It returns 1 if a match was found, 0 if no match was found, false if an error occurred.
<code>preg_match_all(\$regex, \$string)</code>	Looks for a matching pattern in a string. It returns the number of matches found (0 if none found), or false if an error occurred.
<code>preg_split(\$regex, \$string)</code>	Looks for a matching pattern in a string. It splits the string each time a match is found, then returns each of those parts in an indexed array.
<code>preg_replace(\$regex, \$replace, \$string)</code>	Replaces specified characters with an alternative string. It is similar to a find and replace tool in a word processor. It returns the string with characters replaced or null if an error occurred. To delete characters, replace them with a blank string.

Ejemplo: usando expresiones regulares

En el siguiente ejemplo se pone en práctica algunas de estas funciones de expresiones regulares:

1. El texto se almacena en una variable llamada `$text`.
2. A la ruta del archivo se almacena en una variable llamada `$path`.
3. La función `preg_match()` comprueba si la cadena PHP se encuentra en el texto que se almacenó en la variable llamada `$text` en el paso 1. Si se encuentra, `$match` almacena 1.
4. La función `preg_split()` divide la ruta que se almacenó en la variable `$path` en el Paso 2 cada vez que hay una barra diagonal, y coloca cada parte en un nuevo elemento de un array.

Ejemplo: usando expresiones regulares

5. La función `preg_replace()` busca las letras PHP en el texto que se almacenó en `$text` en el Paso 1. Si se encuentra, entonces se busca en la variable `$text`. Si se encuentra, entonces se reemplaza con las mismas letras dentro de un elemento HTML ``.
6. A continuación, se utiliza un operador ternario para comprobar si la variable `$match` tiene el valor 1. En caso afirmativo, se escribe la palabra Yes en la página. Si no es así, en su lugar se muestra la palabra No.
7. Un bucle muestra cada elemento del array `$path` en una nueva línea.
8. Se muestra el texto actualizado almacenado en `$text`, con los caracteres PHP en etiquetas ``.

Ejemplo: usando expresiones regulares

```
<?php  
① $text = 'Using PHP\'s regular expression functions';  
② $path = 'code/section_b/c05/';  
  
③ $match = preg_match('/PHP/', $text);  
④ $path  = preg_split('/\//', $path);  
⑤ $text  = preg_replace('/PHP/', '<em>PHP</em>', $text);  
    ?> ...  
    <b>Was a match found?</b><br>  
⑥ <?= ($match === 1) ? 'Yes' : 'No' ?><br><br>  
  
    <b>Parts of a path:</b><br>  
⑦ <?php foreach($path as $part) { ?>  
    <?= $part ?><br>  
    <?php } ?>  
  
    <b>Updated text:</b><br>  
⑧ <?= $text ?>
```

Ejemplo: usando expresiones regulares

Was a match found?

Yes

Parts of a path:

code

section_b

c05

Updated text:

Using *PHP's* regular expression functions

Ejemplo: usando expresiones regulares

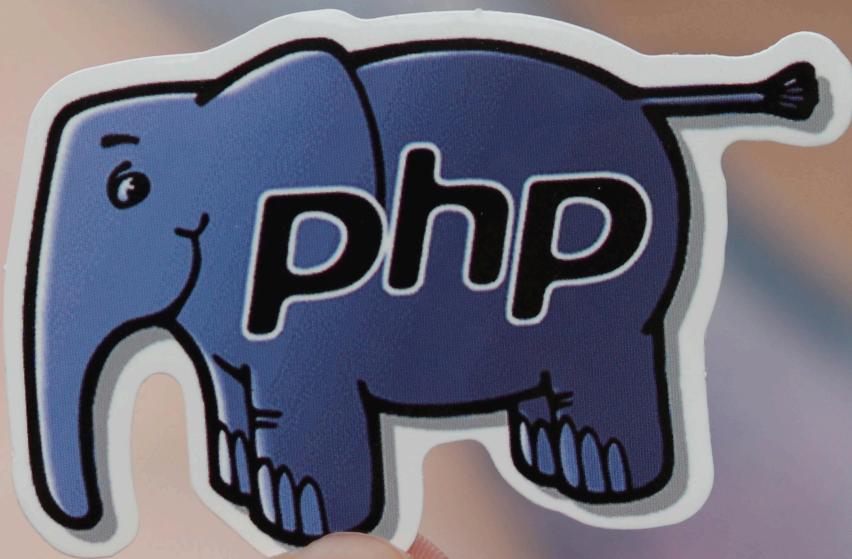
Desarrollar un script en PHP que realice diversas operaciones de validación y manipulación de texto usando funciones de expresiones regulares. El script procesará un array de cadenas que contiene varias líneas de datos, como direcciones de correo electrónico, números de teléfono y URLs.

```
$data = [
    "john.doe@example.com",
    "jane-doe@website.org",
    "invalid-email@com",
    "123-456-7890",
    "987.654.3210",
    "http://www.example.com",
    "https://site.org/path?query=string",
    "not-a-url"
];
```

Ejemplo: usando expresiones regulares

Pasos a seguir

1. Crear un archivo PHP (`regex_practice.php`) y definir el array `$data` con el contenido proporcionado.
2. Utilizar `preg_match` para validar correos electrónicos: Crear una expresión regular que valide las direcciones de correo electrónico y comprobar cada elemento del array.
3. Utilizar `preg_match_all` para extraer números de teléfono: Crear una expresión regular que coincida con los números de teléfono y extraer todos los números válidos.
4. Utilizar `preg_split` para dividir una URL en sus componentes: Crear una expresión regular que divida una URL en protocolo, dominio y ruta.
5. Utilizar `preg_replace` para limpiar las direcciones de correo inválidas: Crear una expresión regular que reemplace las direcciones de correo electrónico inválidas con una cadena vacía.



3. Funciones numéricas

Trabajando con números

Además de los operadores matemáticos que conocimos en la unidad 1, existen funciones para tareas comunes que los programadores realizan con números.

Trabajando con números

FUNCTION	DESCRIPTION
<code>round(\$number, \$places, \$round)</code>	Rounds floating point numbers up or down: \$number is the number to round up or down. \$places is the number of decimal places to round the number to. \$round specifies how to round numbers up or down, the options are:
OPTION	PURPOSE
<code>PHP_ROUND_HALF_UP</code>	Rounds halves up (e.g., 3.5 becomes 4).
<code>PHP_ROUND_HALF_DOWN</code>	Rounds halves down (e.g., 3.5 becomes 3).
<code>PHP_ROUND_HALF_EVEN</code>	Rounds halves to the nearest even number.
<code>PHP_ROUND_HALF_ODD</code>	Rounds halves to the nearest odd number.
<code>ceil(\$number)</code>	Rounds a number up to the nearest integer (whole number).
<code>floor(\$number)</code>	Rounds a number down to the nearest integer (whole number).
<code>mt_rand(\$min, \$max)</code>	Creates a random number between \$min and \$max.
<code>rand(\$min, \$max)</code>	Since PHP 7.1 rand() has been the same as mt_rand(). Before this, rand() used an algorithm that was less random and slower.
<code>pow(\$base, \$exponent)</code>	Returns the base to the exponent power (e.g., 3 ⁴ would return 81).
<code>sqrt(\$number)</code>	Returns the square root of a number.
<code>is_numeric(\$number)</code>	Checks if a value is a number (either an integer or a float). Returns true if it is a number, false if not.
<code>number_format(\$number [, \$decimals] [, \$decimal_point] [, \$thousand_separator])</code>	Specifies how a number should be formatted. If just \$number is given, it formats it without decimals and a comma is used to separate thousands. \$decimals shows the given number of decimal places with a dot to separate decimals and a comma to separate thousands. \$decimal_point and \$thousand_separator let you specify the characters used to separate decimals and thousands. To use decimal_point or thousand_separator you must use both.

Ejemplo: funciones numéricas

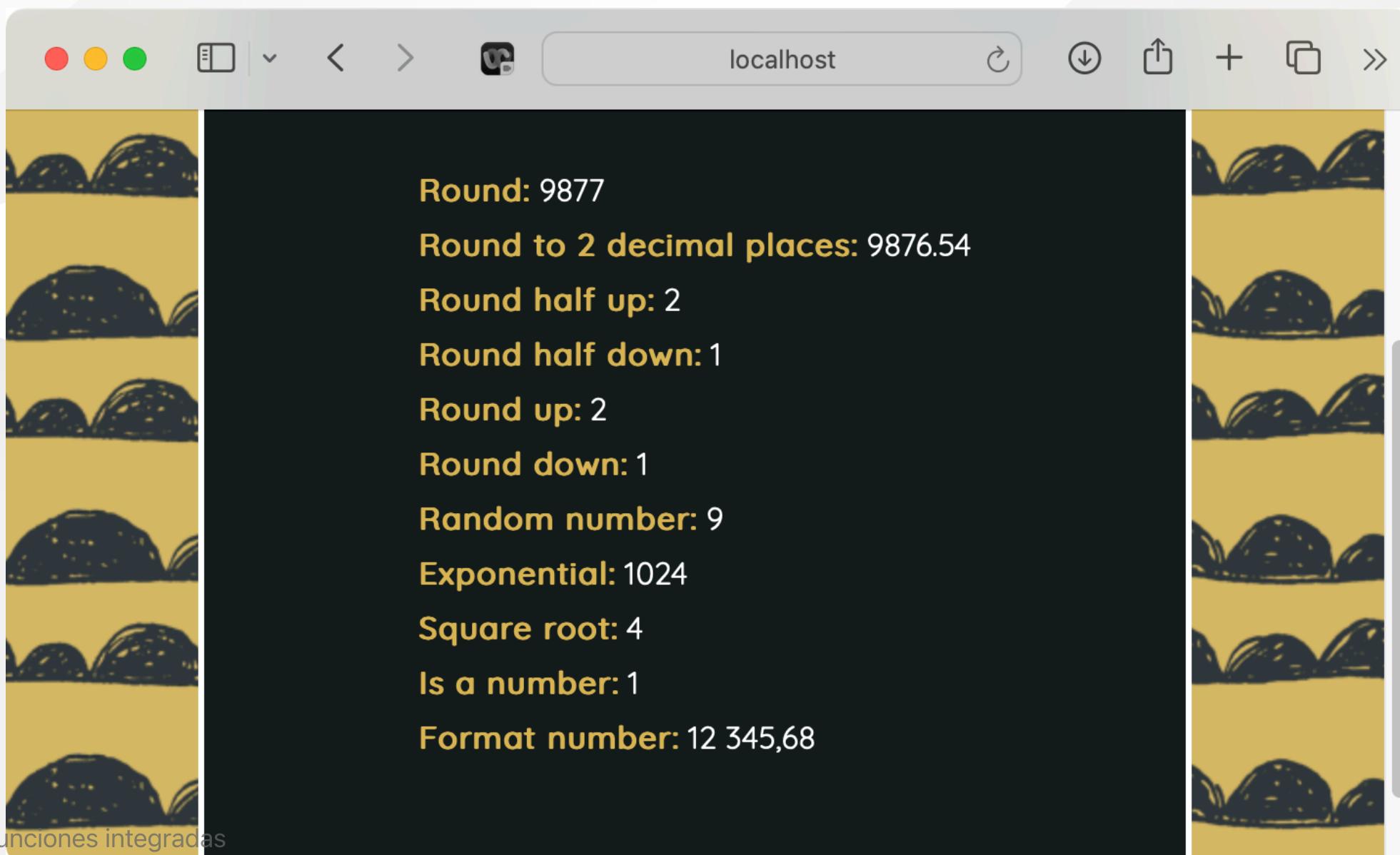
En este ejemplo se ponen en práctica algunas de las funciones de la tabla anterior:

1. Los números se redondean de varias formas.
2. Se genera un número aleatorio entre 0 y 10.
3. Se calcula 4 a la potencia de 5.
4. Se muestra la raíz cuadrada de 16.
5. Comprueba si un valor es un número (un int o un float). Devuelve verdadero si lo es (muestra 1 en la página); falso si no lo es (no muestra nada).
6. El número se formatea con 2 decimales. Un espacio separa los miles y una coma los decimales.

Ejemplo: funciones numéricas

①	Round:	<?= round(9876.54321) ?>
	Round to 2 decimal places:	<?= round(9876.54321, 2) ?>
	Round half up:	<?= round(1.5, 0, PHP_ROUND_HALF_UP) ?>
	Round half down:	<?= round(1.5, 0, PHP_ROUND_HALF_DOWN) ?>
	Round up:	<?= ceil(1.23) ?>
	Round down:	<?= floor(1.23) ?>
②	Random number:	<?= mt_rand(0, 10) ?>
③	Exponential:	<?= pow(4, 5) ?>
④	Square root:	<?= sqrt(16) ?>
⑤	Is a number:	<?= is_numeric(123) ?>
⑥	Format number:	<?= number_format(12345.6789, 2, ',', ',') ?>

Ejemplo: funciones numéricas



Ejemplo: funciones numéricas

Vamos a simular las ventas de un día en una hamburguesería. La hamburguesería tiene varios tipos de hamburguesas con precios diferentes, y cada venta incluirá una cantidad aleatoria de hamburguesas. Usaremos las funciones `round`, `ceil`, `floor`, `mt_rand`, `rand`, `pow`, `sqrt`, `is_numeric` y `number_format` para procesar las ventas y calcular los totales.

Ejemplo: funciones numéricas

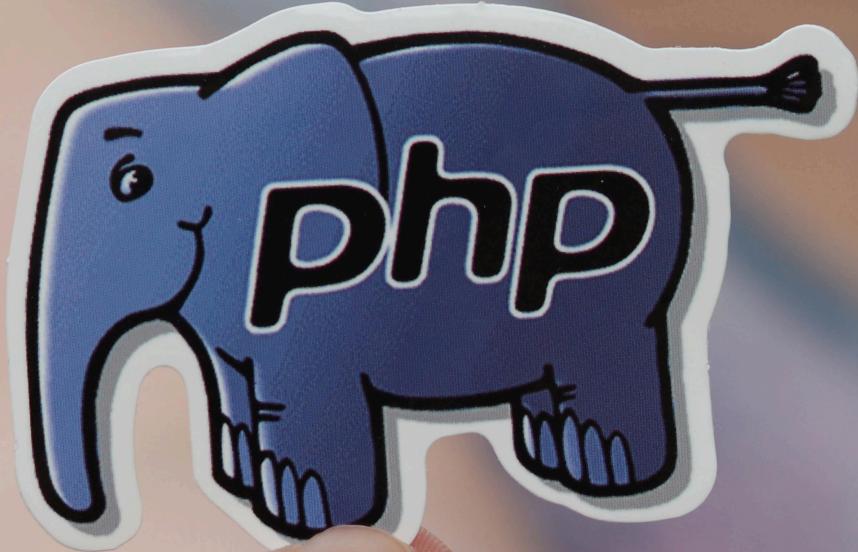
Tipos de Hamburguesas y Precios:

- Hamburguesa Clásica: \$5.50
- Hamburguesa con Queso: \$6.75
- Hamburguesa BBQ: \$7.25
- Hamburguesa Vegetariana: \$6.00

Ejemplo: funciones numéricas

Tareas:

- 1. Generar Ventas Aleatorias:** Utilizar `mt_rand` para generar una cantidad aleatoria de ventas entre 50 y 100 para el día.
- 2. Asignar Hamburguesas y Cantidades:** Para cada venta, asignar aleatoriamente un tipo de hamburguesa y una cantidad entre 1 y 5.
- 3. Calcular el Total de Cada Venta:** Calcular el total de cada venta y redondearlo a dos decimales usando `round`.
- 4. Calcular el Total del Día:** Sumar todas las ventas para obtener el total del día y formatearlo con `number_format`.
- 5. Estadísticas:** Calcular la raíz cuadrada del total de ventas y elevar el total a la potencia de 2 usando `sqrt` y `pow` respectivamente. También, usar `ceil` y `floor` para redondear hacia arriba y hacia abajo los totales de ventas.



4. Funciones de arrays

Trabajando con arrays

Estas funciones pueden buscar en el contenido de un array, contar el número de elementos que contiene y seleccionar claves aleatorias.

También pueden convertir arrays en una cadena o una cadena en un array.

Trabajando con arrays

Como ya hemos visto, los arrays contienen un conjunto de pares clave/valor en una única variable.

En un array indexado, la clave es un número índice. Indica la posición del elemento en el array.

Un array asociativo actúa más como una colección de variables relacionadas. Cada clave es una cadena.

Trabajando con arrays

Obteniendo información sobre un array

FUNCTION	DESCRIPTION
<code>array_key_exists(\$key, \$array)</code>	Checks for a key in the array. Returns true if it exists, otherwise returns false.
<code>array_search(\$value, \$array[, \$strict])</code>	Searches the values stored in the array, and returns the key for the first match. If \$strict has a value of true, it indicates that the match must be the same data type.
<code>in_array(\$value, \$array)</code>	Checks if a value is in an array. Returns true if it is, false if not.
<code>count(\$array)</code>	Returns the number of items in the array.
<code>array_rand(\$array[, \$number])</code>	Selects a random item from the array and returns its key. If a number is specified as the second parameter, it returns an array containing that number of random keys.

Trabajando con arrays

Convertir arrays en cadenas y viceversa

FUNCTION	DESCRIPTION
<code>implode([\$separator,]\$array)</code>	Turns the values of an array into a string (keys are not included). If you specify a separator, it is inserted between each value.
<code>explode(\$separator, \$string[, \$limit])</code>	Turns a string into an indexed array. The separator is the character that separates each of the items in the string. An optional limit can be used to set the maximum number of items to add to the array.

Ejemplo: Funciones de array

1. Se crea un array llamado `$greetings` para contener varios saludos.
2. Se selecciona una clave aleatoria del array y se almacena en una variable llamada `$greeting_key`.
3. La clave aleatoria se utiliza para seleccionar el saludo del array y almacenarlo en `$greeting`.
4. Un array de los artículos más vendidos se almacena en `$bestsellers`.
5. La función `count()` se utiliza para contar el número de elementos en el array; esto se almacena en `$bestseller_count`.

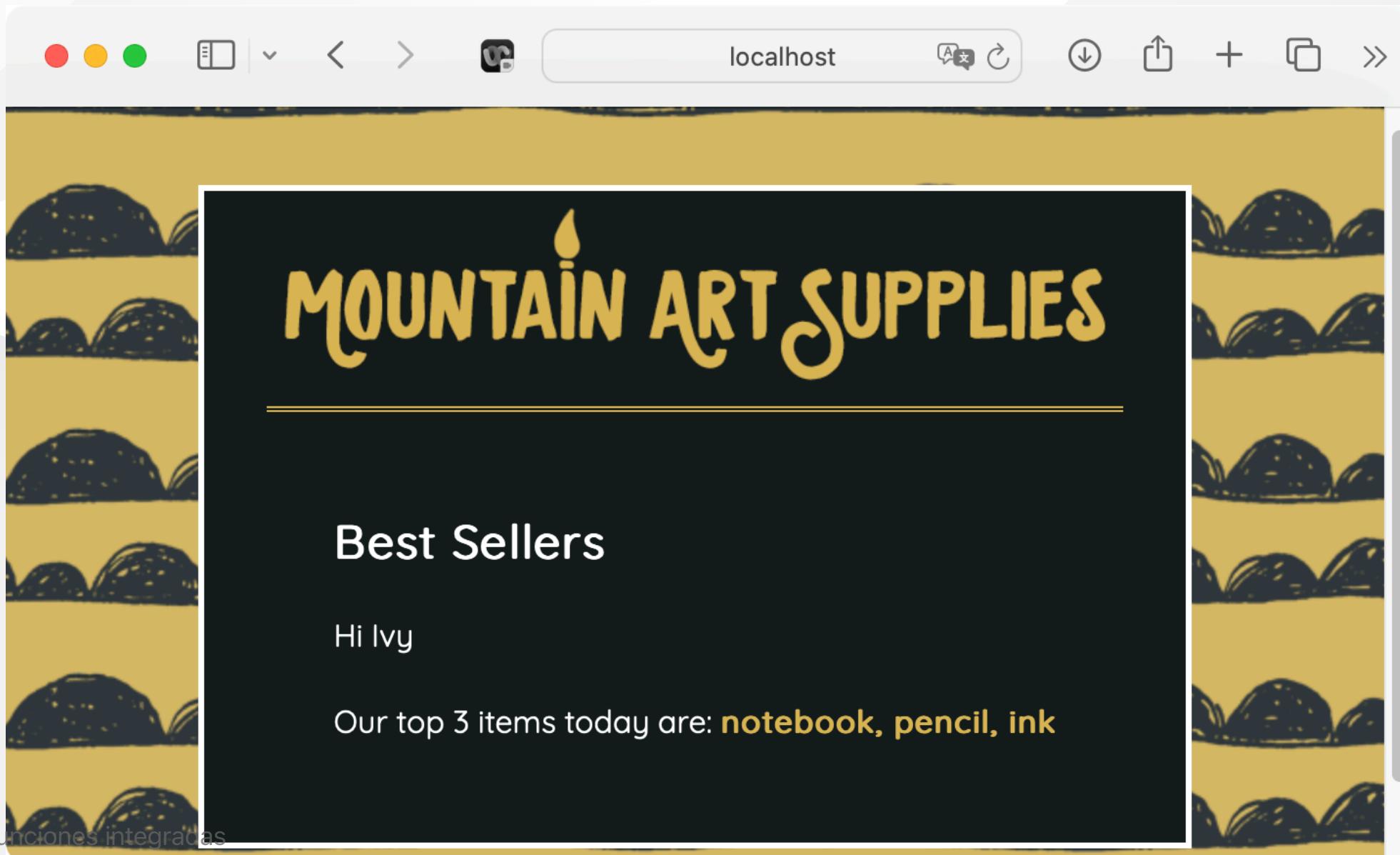
Ejemplo: Funciones de array

6. El array se convierte en una cadena utilizando `implode()`, con una coma para separar cada elemento. Se almacena en `$bestseller` text.
7. Se crea un array asociativo con los datos del cliente.
8. `array_key_exists()` comprueba si existe un valor para el nombre del cliente. Si lo hay, se añade a `$greeting`.
9. Se escribe el saludo.
10. Se muestra el número de los más vendidos y sus nombres.

Ejemplo: Funciones de array

```
<?php
    // Create array of greetings then get random value
    ① [ $greetings    = ['Hi ', 'Howdy ', 'Hello ', 'Hola ',
                        'Welcome ', 'Ciao ',];
    ② $greeting_key = array_rand($greetings);
    ③ $greeting    = $greetings[$greeting_key];
    // Array of best sellers, count items, list top items
    ④ $bestsellers    = ['notebook', 'pencil', 'ink',];
    ⑤ $bestseller_count = count($bestsellers);
    ⑥ $bestseller_text = implode(' ', $bestsellers);
    // Array holding customer details
    ⑦ [ $customer    = ['forename' => 'Ivy',
                        'surname'   => 'Stone',
                        'email'     => 'ivy@eg.link',];
    // If you have a customer forename, add it to greeting
    ⑧ [ if (array_key_exists('forename', $customer)) {
        $greeting .= $customer['forename'];
    }
    ?> ...
    <h1>Best Sellers</h1>
    ⑨ <p><?= $greeting ?></p>
    ⑩ [ <p>Our top <?= $bestseller_count ?> items today are:
        <b><?= $bestseller_text ?></b></p>
```

Ejemplo: Funciones de array



Añadir y eliminar elementos de un array

Estas funciones añaden y eliminan elementos de un array.

Puedes especificar si los nuevos elementos deben añadirse al principio o al final del array.

Añadir y eliminar elementos de un array

Para añadir un elemento a un array, se especifica el valor a añadir.

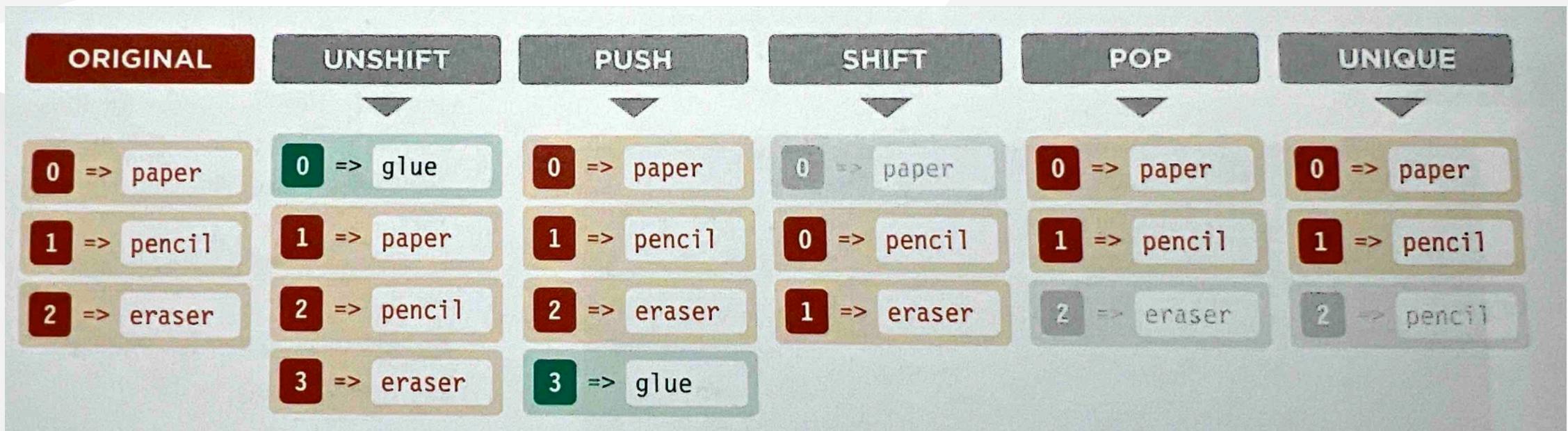
Para eliminar un elemento de un array, basta con especificar su clave.

El diagrama muestra las posiciones de los elementos que se añaden o eliminan.

Añadir y eliminar elementos de un array

FUNCTION	DESCRIPTION
<code>array_unshift(\$array, \$items)</code>	Adds one or more items to the start of an indexed array. Returns number of items in the array. (For associative arrays, see p42.)
<code>array_push(\$array, \$items)</code>	Adds one or more items to the end of an indexed array. Returns number of items in array. (For associative arrays, see p42.)
<code>array_shift(\$array)</code>	Removes the first item from the array. Returns value of the removed item.
<code>array_pop(\$array)</code>	Removes the last item from the array. Returns value of the removed item.
<code>array_unique(\$array)</code>	Removes duplicate entries from an array. Returns the new array.
<code>array_merge(\$array1, \$array2)</code>	Joins two or more arrays and returns the new array. If both are indexed arrays, the index numbers of the new array start at 0. You can also join two arrays using the + operator: \$array1 + \$array2

Añadir y eliminar elementos de un array



Ejemplo: funciones que actualizan arrays

1. Se crea un array indexado y se almacena en `$order`.
2. La función `array_unshift()` añade un elemento al principio del array. El primer parámetro es el array; el segundo es el elemento a añadir (esto sólo funciona con arrays indexados).
3. La función `array_pop()` elimina el último elemento.
4. El array se convierte en una cadena utilizando `implode()` y se almacena en `$items`. Cada elemento se separa con una coma y un espacio.
5. Se crea un array asociativo y se almacena en `$classes`.

Ejemplo: funciones que actualizan arrays

6. `array_shift()` elimina el primer elemento del array.
7. Se crea otro array asociativo para contener nuevos elementos.
8. Se utiliza `array_merge()` para tomar el array `$classes` y añadir los nuevos elementos creados en el paso 7.
9. Se escribe `$items`.
10. Un bucle `foreach` escribe las claves y valores del array asociativo.

Ejemplo: funciones que actualizan arrays

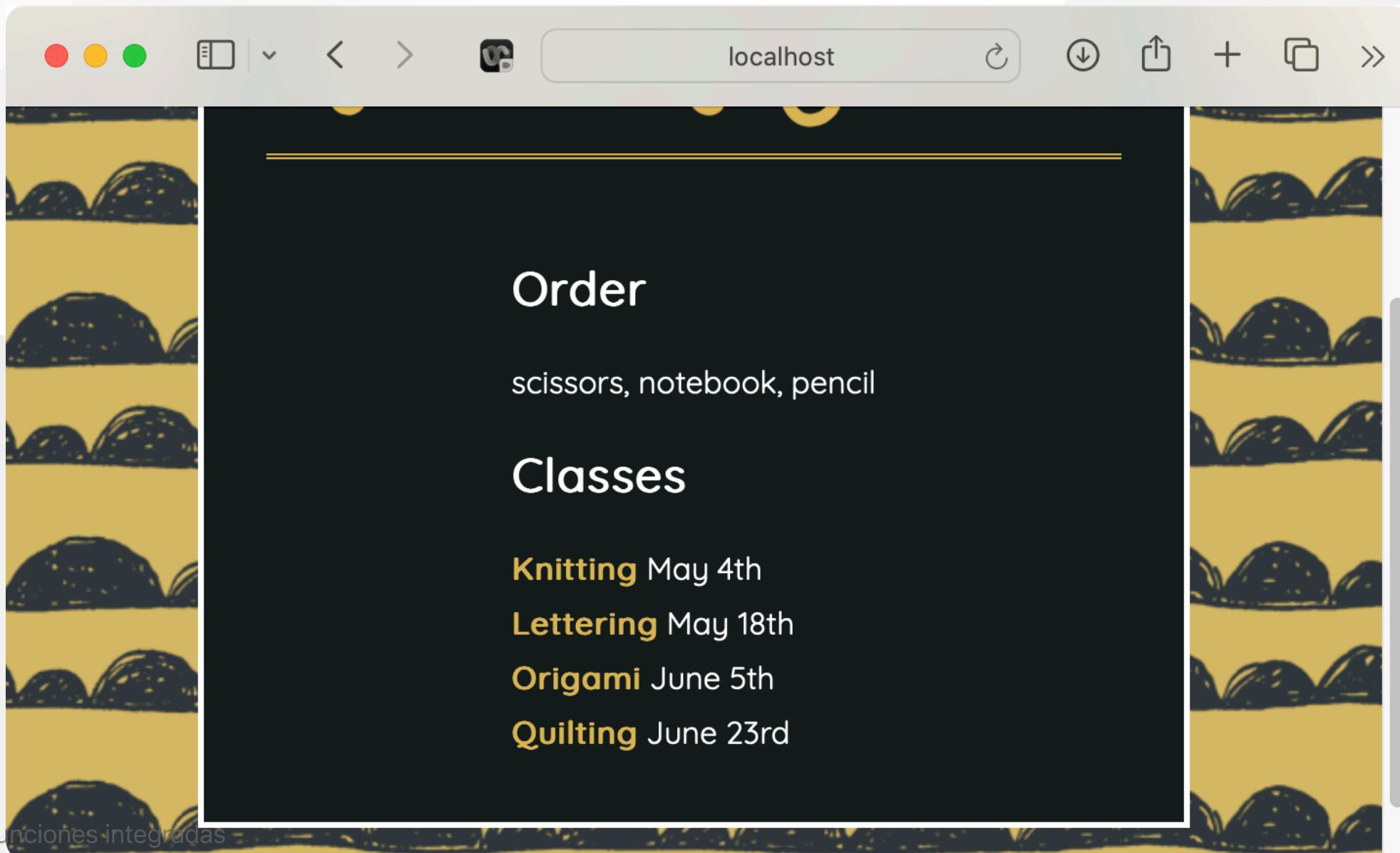
```
<?php
// Array of items being ordered
$order = ['notebook', 'pencil', 'eraser',];
array_unshift($order, 'scissors'); // Add to start
array_pop($order); // Remove last
$items = implode(', ', $order); // Convert to string

// Array of classes
$classes = ['Patchwork' => 'April 12th',
            'Knitting' => 'May 4th',
            'Lettering' => 'May 18th',];
array_shift($classes); // Remove 1st
$new      = ['Origami' => 'June 5th',
            'Quilting' => 'June 23rd',]; // New items
$classes = array_merge($classes, $new); // Add to end
?>
<?php include 'includes/header.php'; ?>

<h1>Order</h1>
<?= $items ?>
<h1>Classes</h1>
<?php foreach($classes as $description => $date) { ?>
    <b><?= $description ?></b> <?= $date ?><br>
<?php } ?>

<?php include 'includes/footer.php'; ?>
```

Ejemplo: funciones que actualizan arrays



Ejemplo: Funciones de array

Vamos a crear y gestionar una lista de reproducción de música utilizando diversas funciones de PHP. La lista contendrá varias canciones y se realizarán diversas operaciones como agregar, buscar, eliminar y mostrar canciones aleatorias.

Ejemplo: Funciones de array

Lista de Canciones Inicial:

- "Blinding Lights" - The Weeknd
- "Estoy enfermo" - Pignoise
- "Levitating" - Dua Lipa
- "One more night" - Maroon 5
- "Feel Good Inc." - Gorillaz

Ejemplo: Funciones de array

Tareas:

1. **Agregar Canciones a la Lista:** Utilizar `array_key_exists`, `array_unshift` y `array_push` para agregar canciones al inicio y al final de la lista.
2. **Eliminar Canciones de la Lista:** Utilizar `array_shift` y `array_pop` para eliminar canciones del inicio y del final de la lista.
3. **Buscar una Canción en la Lista:** Utilizar `array_search` para encontrar la posición de una canción en la lista.
4. **Verificar si una Canción Está en la Lista:** Utilizar `in_array` para comprobar si una canción está en la lista.
5. **Contar el Número de Canciones:** Utilizar `count` para contar el número total de canciones en la lista.

Ejemplo: Funciones de array

Tareas:

6. **Seleccionar Canciones Aleatorias:** Utilizar `array_rand` para seleccionar canciones aleatorias de la lista.
7. **Mostrar la Lista de Reproducción:** Utilizar `implode` para mostrar todas las canciones de la lista como una cadena.
8. **Dividir la Cadena en Canciones:** Utilizar `explode` para convertir una cadena con canciones en un array.
9. **Eliminar Duplicados:** Utilizar `array_unique` para eliminar canciones duplicadas en la lista.
10. **Fusionar Listas:** Utilizar `array_merge` para combinar dos listas de reproducción.

Ordenando arrays

Las funciones de ordenación cambian el orden de los elementos listados en un array.

- Las listas ascendentes ordenan los elementos de menor a mayor valor (por ejemplo, A-Z o 0-9).
- Las listas descendentes ordenan los elementos de mayor a menor valor (por ejemplo, Z-A o 9-0).

Ordenando arrays

Ordenar por valor cambiando claves

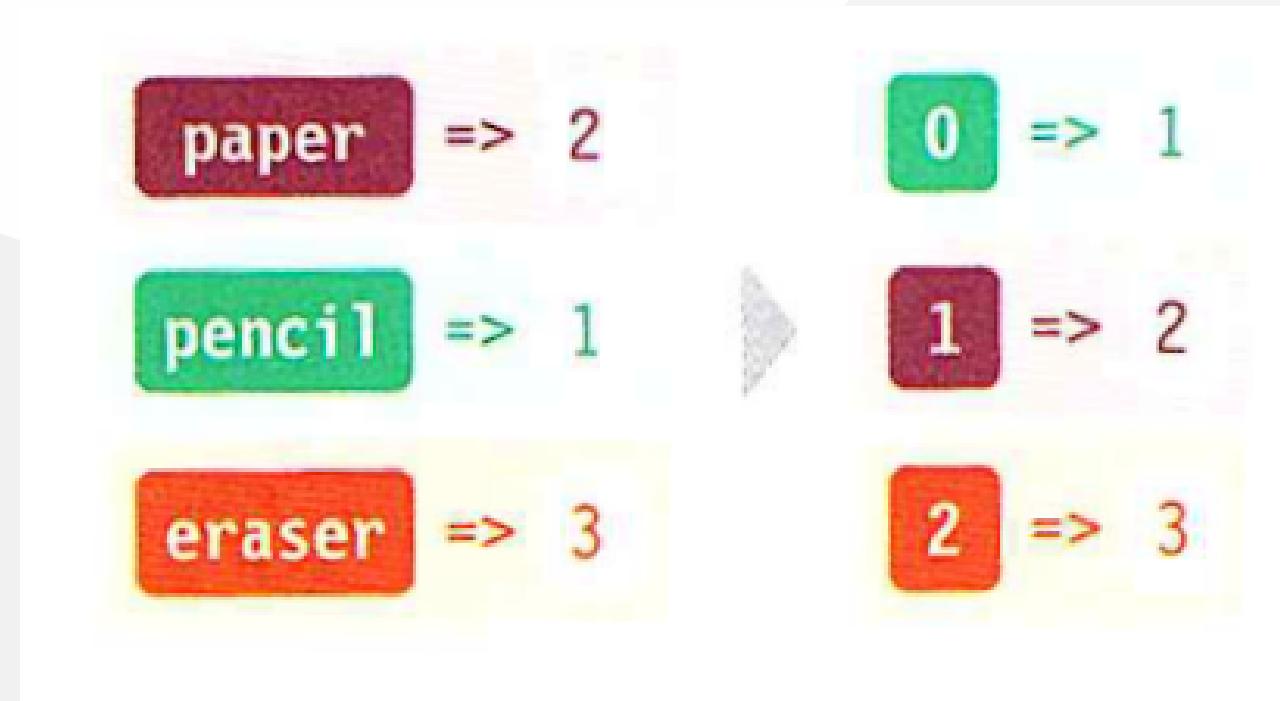
Cuando se ordena el array utilizando las funciones que se indican a continuación, las claves se convierten en números índice empezando por 0 (tanto si se trata de un array indexado como asociativo).

La `r` en `rsort()` significa reverso.

FUNCTION	DESCRIPTION
<code>sort(\$array)</code>	Ascending according to value
<code>rsort(\$array)</code>	Descending according to value

Ordenando arrays

Ordenar por valor cambiando claves



Ordenando arrays

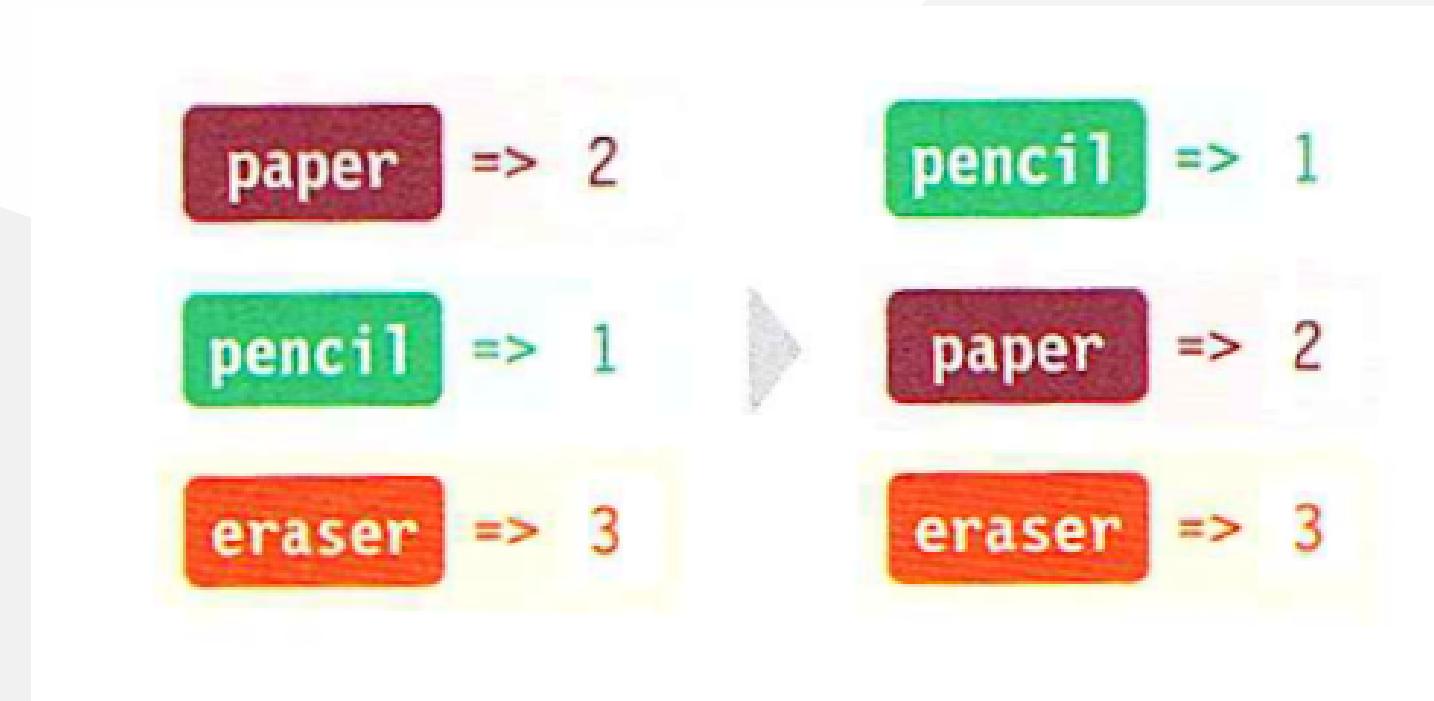
Ordenar por valor manteniendo claves

Cuando ordenas el array utilizando las funciones de abajo, las claves se mueven con sus valores.

FUNCTION	DESCRIPTION
<code>asort(\$array)</code>	Ascending according to value
<code>arsort(\$array)</code>	Descending according to value

Ordenando arrays

Ordenar por valor manteniendo claves



Ordenando arrays

Ordenar por clave manteniendo valores

Al ordenar el array utilizando las funciones que se indican a continuación, los valores se desplazan con sus claves.

FUNCTION	DESCRIPTION
<code>ksort(\$array)</code>	Ascending according to key
<code>krsort(\$array)</code>	Descending according to key

Ordenando arrays

Ordenar por clave manteniendo valores

The diagram illustrates the process of sorting an array by key while keeping the original values. It shows two states of an array:

Key	Value
paper	2
pencil	1
eraser	3

An arrow points from the initial state to the sorted state:

Key	Value
paper	2
pencil	1
eraser	3

Ejemplo: funciones de ordenación de arrays

En el siguiente ejemplo se ponen en práctica las funciones de la tabla anterior:

1. Se crea un array indexado y se almacena en una variable llamada `$order`.
2. Los valores del array se ordenan en orden alfabético ascendente utilizando `sort()`.
Esto da a cada elemento del array un nuevo número de índice que comienza en 0.
3. El array se convierte en una cadena utilizando `implode()`. Cada elemento se separa con una coma seguida de un espacio. La cadena resultante se almacena en una variable llamada `$items`.

Ejemplo: funciones de ordenación de arrays

4. Se crea un array asociativo y se almacena en `$classes`.
5. Las claves del array se ordenan por orden alfabético utilizando la función `ksort()` (sus valores se mueven con ellas).
6. Se escribe la cadena almacenada en `$items`.
7. Se utiliza un bucle `foreach` para escribir las claves y valores de la matriz `$classes` en la página.

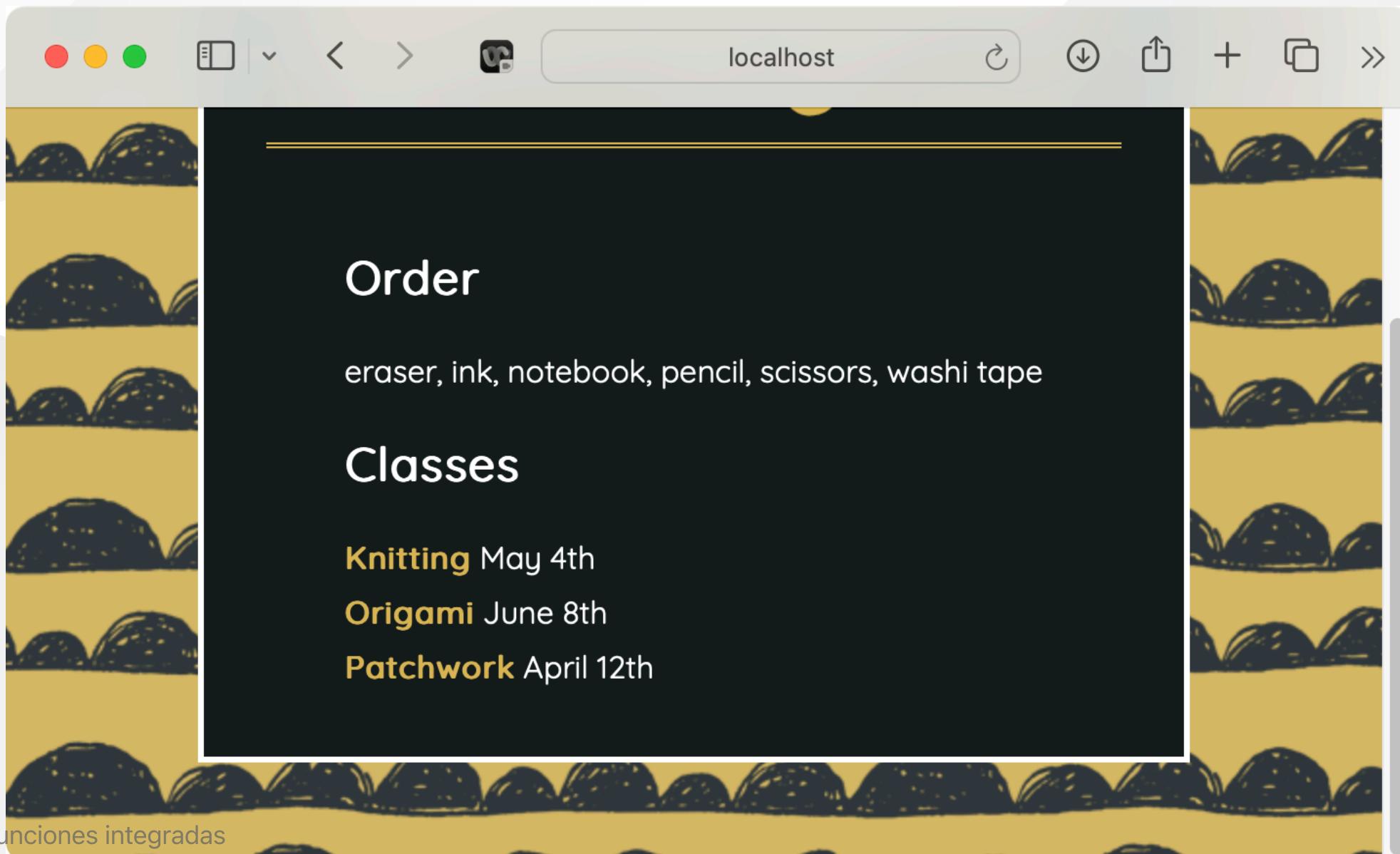
Ejemplo: funciones de ordenación de arrays

```
<?php
    // Array holding order
    ① [ $order = ['notebook', 'pencil', 'scissors',
                  'eraser', 'ink', 'washi tape',];
    ② sort($order);                      // Sort ascending
    ③ $items = implode(', ', $order);    // Convert to text

        // Create array holding classes
        $classes = ['Patchwork' => 'April 12th',
                    'Knitting'   => 'May 4th',
                    'Origami'     => 'June 8th',];
    ④ ksort($classes);                  // Sort by key
    ?>

    <h1>Order</h1>
    ⑥ <?= $items ?>
    <h1>Classes</h1>
    ⑦ [<?php foreach($classes as $description => $date) { ?>
            <b><?= $description ?></b> <?= $date ?><br>
        <?php } ?>
```

Ejemplo: funciones de ordenación de arrays



Ejemplo: funciones de ordenación de arrays

Vamos a crear y gestionar una lista de reproducción de música, y cada vez que se ejecute el script, se generarán reproducciones aleatorias para las canciones. Luego, ordenaremos las canciones utilizando diversas funciones de ordenación de arrays en PHP.

Puedes tomar como punto de partida la actividad anterior "*Ejemplo: Funciones de arrays*"

Ejemplo: funciones de ordenación de arrays

Lista de Canciones Inicial con Reproducciones:

```
$canciones_con_reproducciones = [  
    "Blinding Lights - The Weeknd" => 1500,  
    "Estoy enfermo - Pignoise" => 1200,  
    "Levitating - Dua Lipa" => 1800,  
    "One more night - Maroon 5" => 1600,  
    "Feel Good Inc. - Gorillaz" => 1700  
];
```

Ejemplo: funciones de ordenación de arrays

Tareas:

1. Generar Reproducciones Aleatorias: Utilizar `mt_rand` para generar un número aleatorio de reproducciones para cada canción.
2. Ordenar la Lista por Nombre de Canción en Orden Ascendente: Utilizar `sort`.
3. Ordenar la Lista por Nombre de Canción en Orden Descendente: Utilizar `rsort`.
4. Ordenar la Lista por Número de Reproducciones en Orden Ascendente: Utilizar `asort`.

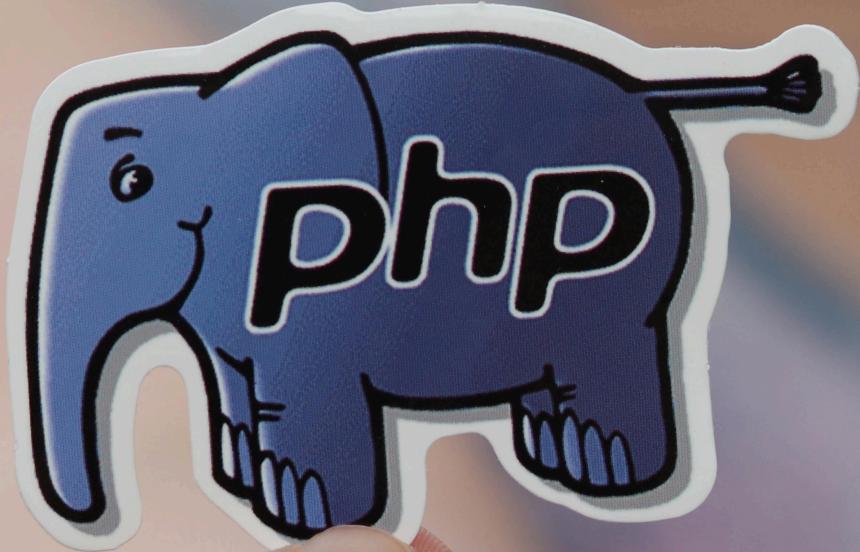
Ejemplo: funciones de ordenación de arrays

Tareas:

5. Ordenar la Lista por Número de Reproducciones en Orden Descendente: Utilizar
`arsort`.

6. Ordenar la Lista por Clave (Nombre de Canción) en Orden Ascendente: Utilizar
`ksort`.

7. Ordenar la Lista por Clave (Nombre de Canción) en Orden Descendente: Utilizar
`krsort`.



5. Constantes

Constantes

Una constante es un par nombre/valor, como una variable, pero:

- Se crea utilizando la función `define()`
- Su valor no se puede actualizar una vez que se ha establecido
- Se puede acceder a ella en cualquier parte de la página PHP (incluso dentro de funciones)

Su nombre debe describir el tipo de datos que contiene y debe comenzar con una letra o guión bajo (no con el símbolo del dólar). Su valor puede ser un tipo de dato escalar o un array.

Constantes

Los parámetros de la función `define()` son:

- El nombre de la constante, que suele ir en mayúsculas.
- Su valor; las cadenas deben ir entre comillas, los números y los booleanos no.
- Un booleano opcional para indicar si el nombre distingue entre mayúsculas y minúsculas (true en caso afirmativo, false en caso negativo). Si no se introduce el tercer parámetro, el nombre distinguirá entre mayúsculas y minúsculas.

```
define('SITE_NAME', 'Mountain Art Supplies');
```



Constantes

Las constantes suelen utilizarse para almacenar información que un sitio necesita para funcionar, pero cuyos valores sólo cambian cuando se instala un sitio (ya sea la primera vez, cuando se traslada a un nuevo servidor o cuando se utiliza el mismo código para alimentar un sitio web distinto).

También se puede crear una constante utilizando la palabra clave `const`, seguida del nombre de la constante, el operador de asignación y el valor que debe contener. Este método puede utilizarse para definir una constante dentro de una clase (mientras que la función `define()` no puede).

Ejemplo: usando constantes

En este ejemplo, un archivo *include* llamado `settings.php` creará dos constantes que contienen información sobre el sitio.

1. La función `define()` se utiliza para crear una constante llamada `SITE_NAME`. Su valor es el nombre del sitio.
2. La palabra clave `const` se utiliza para crear una constante llamada `ADMIN_EMAIL`. Su valor es la dirección de correo electrónico del propietario del sitio.

Ejemplo: usando constantes

El segundo archivo de este ejemplo es una página llamada `constants.php`, que utiliza los valores de estas dos constantes.

3. El archivo `settings.php` se incluye para que la página pueda acceder a las constantes.
4. La abreviatura del comando `echo` se utiliza para escribir el contenido de la constante que contiene el nombre del sitio.
5. Se muestra la dirección de correo electrónico del propietario del sitio.

Ejemplo: usando constantes

```
// settings.php
<?php
define('SITE_NAME', 'Mountain Art Supplies');
const ADMIN_EMAIL = 'admin@eg.link';
```

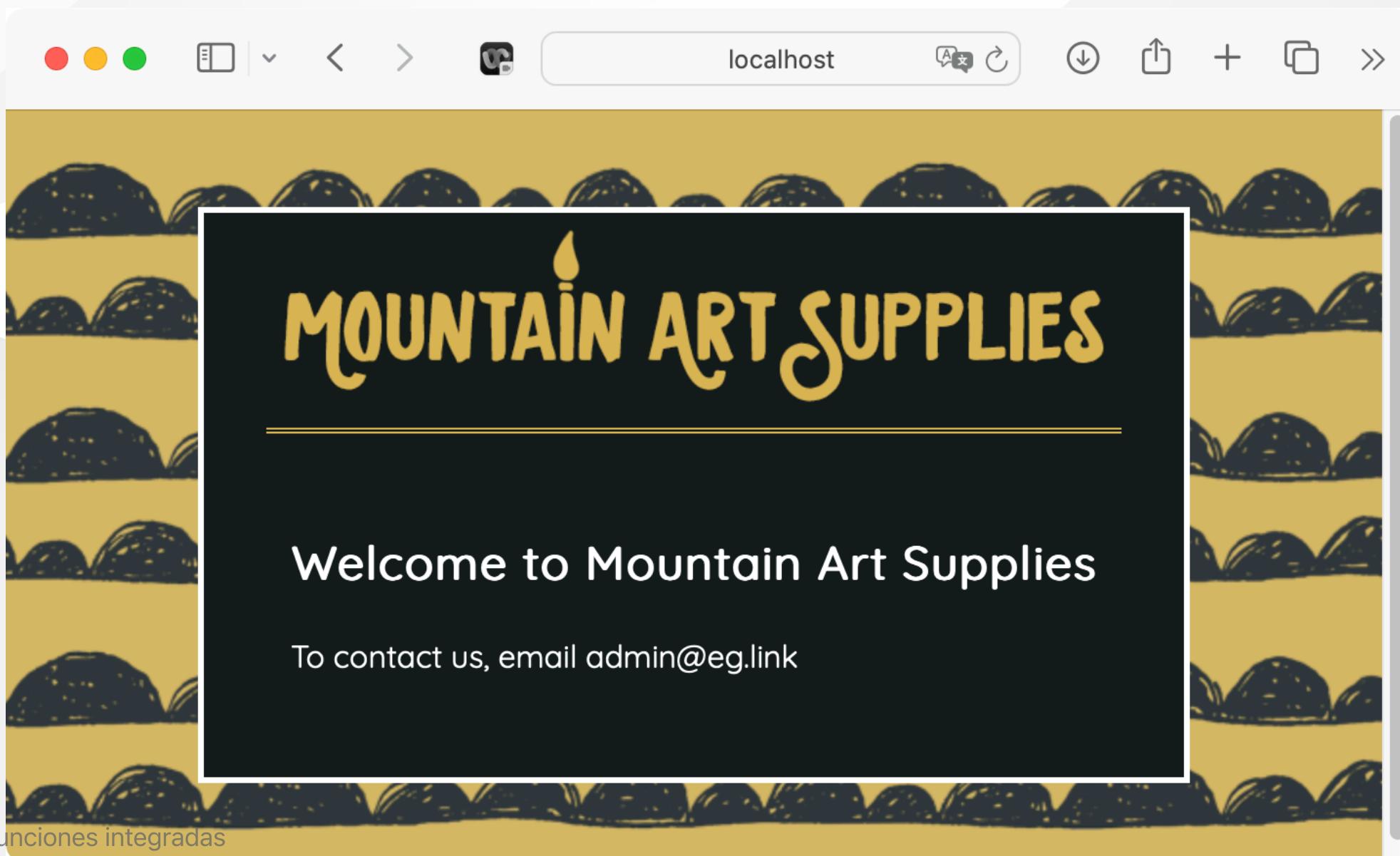
Ejemplo: usando constantes

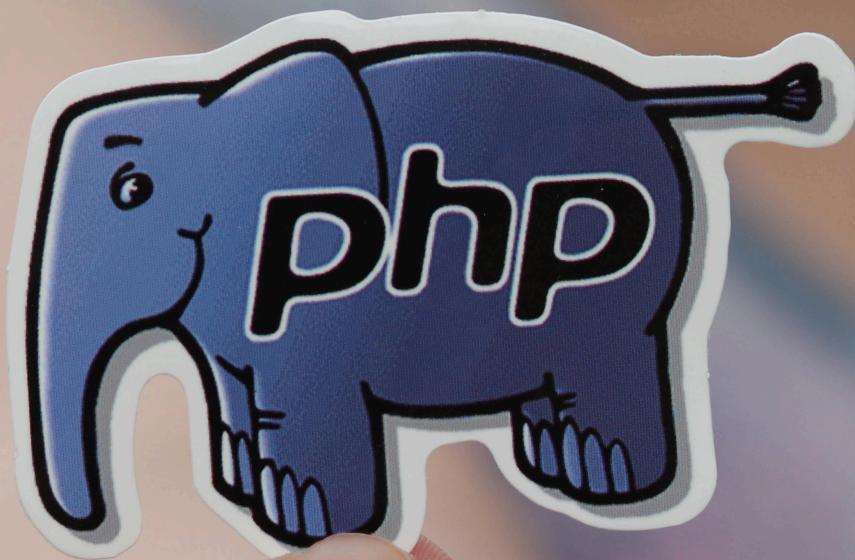
```
// constants.php
<?php
include 'includes/settings.php';
include 'includes/header.php';
?>

<h1>Welcome to <?= SITE_NAME ?></h1>
<p>To contact us, email <?= ADMIN_EMAIL ?></p>

<?php include 'includes/footer.php'; ?>
```

Ejemplo: usando constantes





6. Función header()

Añadir/Actualizar cabeceras HTTP

La función `header()` actualiza las cabeceras HTTP que el intérprete PHP envía al navegador. También podemos añadir nuevas cabeceras.

Su único argumento es el nombre de la cabecera a establecer, seguido de dos puntos, y su valor.

Añadir/Actualizar cabeceras HTTP

A veces, los usuarios solicitan una página, pero hay que enviarlos a otra. Por ejemplo, si

- Ya no está disponible
- ha cambiado de URL
- Le faltan datos que necesita

En este caso, la función `header()` tiene un argumento que se compone de tres partes:

- El nombre de la cabecera `Location`
- Dos puntos (`:`)
- La nueva URL

Cuando el navegador recibe la cabecera `Location`, solicita la nueva URL. Esto debe ir seguido del comando `exit` para evitar que el intérprete ejecute más código PHP (lo veremos en el ejemplo a continuación).

Añadir/Actualizar cabeceras HTTP

```
header('Location: http://www.example.com/');
```



The code snippet shows the `header` function being called with a single argument: a string containing the header name ('Location') and its value ('http://www.example.com/'). A bracket below the argument is divided into two segments: 'HEADER' covers the first part ('Location'), and 'NEW URL' covers the second part ('http://www.example.com/').

Añadir/Actualizar cabeceras HTTP

La mayoría de los archivos PHP crean HTML para enviar al navegador, pero PHP puede ser utilizado para crear otros tipos de archivos como JSON, XML o CSS.

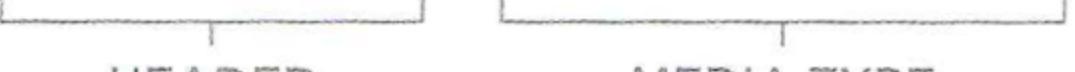
Para hacer esto, `header()` necesita:

- El encabezado `Content-type`
- Dos puntos (`:`)
- El tipo de medio del contenido

Esto crea una cabecera HTTP que indica al navegador el tipo de medio del archivo. Para más información sobre los tipos de medios, revisa el anexo I: Tipos de medios.

Añadir/Actualizar cabeceras HTTP

```
header('Content-type: application/json');
```



The code snippet shows the `header()` function being used to set the `Content-type` header to `application/json`. The string is annotated with brackets below it. The first part, `'Content-type:`, is labeled `HEADER`. The second part, `application/json`, is labeled `MEDIA TYPE`.

Añadir/Actualizar cabeceras HTTP

Los navegadores pueden almacenar en caché las páginas que han visto los usuarios. Si el usuario vuelve a solicitar la página, puede mostrar la página que ha almacenado, en lugar de volver a solicitar el archivo (esto hace que la página parezca cargarse más rápido).

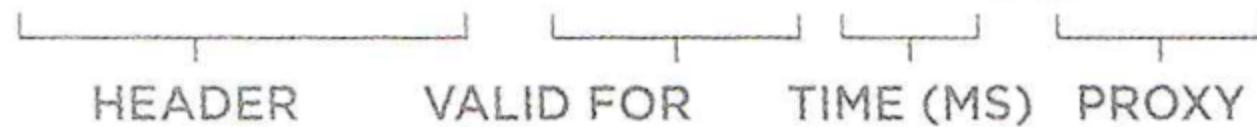
Para indicar al navegador cuánto tiempo puede almacenar en caché una página, se utiliza:

- La cabecera Cache-Control
- Dos puntos (:)
- max-age= seguido del número de milisegundos durante los que se puede almacenar en caché la página

Los ISP y las redes utilizan **proxies** para almacenar en caché las páginas web. Si las páginas contienen datos personales, añade a continuación de los milisegundos una coma, un espacio y la palabra *private*, para evitar que un proxy los almacene en caché. Si no hay datos personales, establece que sea *public*.

Añadir/Actualizar cabeceras HTTP

```
header('Cache-Control: max-age=3600, public');
```



Ejemplo: redirigir usuarios usando cabeceras HTTP

Este ejemplo muestra cómo redirigir a los usuarios a otra página utilizando la función `header()`.

No puedes haber enviado ninguna marca o texto al navegador antes de utilizar la función `header()`, ni siquiera un espacio o un retorno de carro (`\r`).

1. Una variable llamada `$logged_in` almacena un valor booleano que indica si el usuario ha iniciado sesión.
2. En una sentencia `if`, una condición comprueba si el valor de `$logged_in` es falso.
3. Si es falso, el usuario es redirigido a la página `login.php` utilizando la función `header()` (En la unidad 16 aprenderás a crear un área de miembros con una página de inicio de sesión funcional).

Ejemplo: redirigir usuarios usando cabeceras HTTP

4. Después de redirigir a un visitante utilizando la función `header()`, se utiliza el comando `exit` para evitar que se ejecute más código PHP en el archivo.

Si el valor en `$logged_in` es true, el bloque de código anterior habrá sido saltado y se mostrará el resto de la página.

Ejemplo: redirigir usuarios usando cabeceras HTTP

```
// redirect.php
<?php
$logged_in = true;

if ($logged_in == false) {
    header('Location: login.php');
    exit;
}
?>
<?php include 'includes/header.php'; ?>
<h1>Members Area</h1>
<p>Welcome to the members area</p>
<?php include 'includes/footer.php'; ?>
```

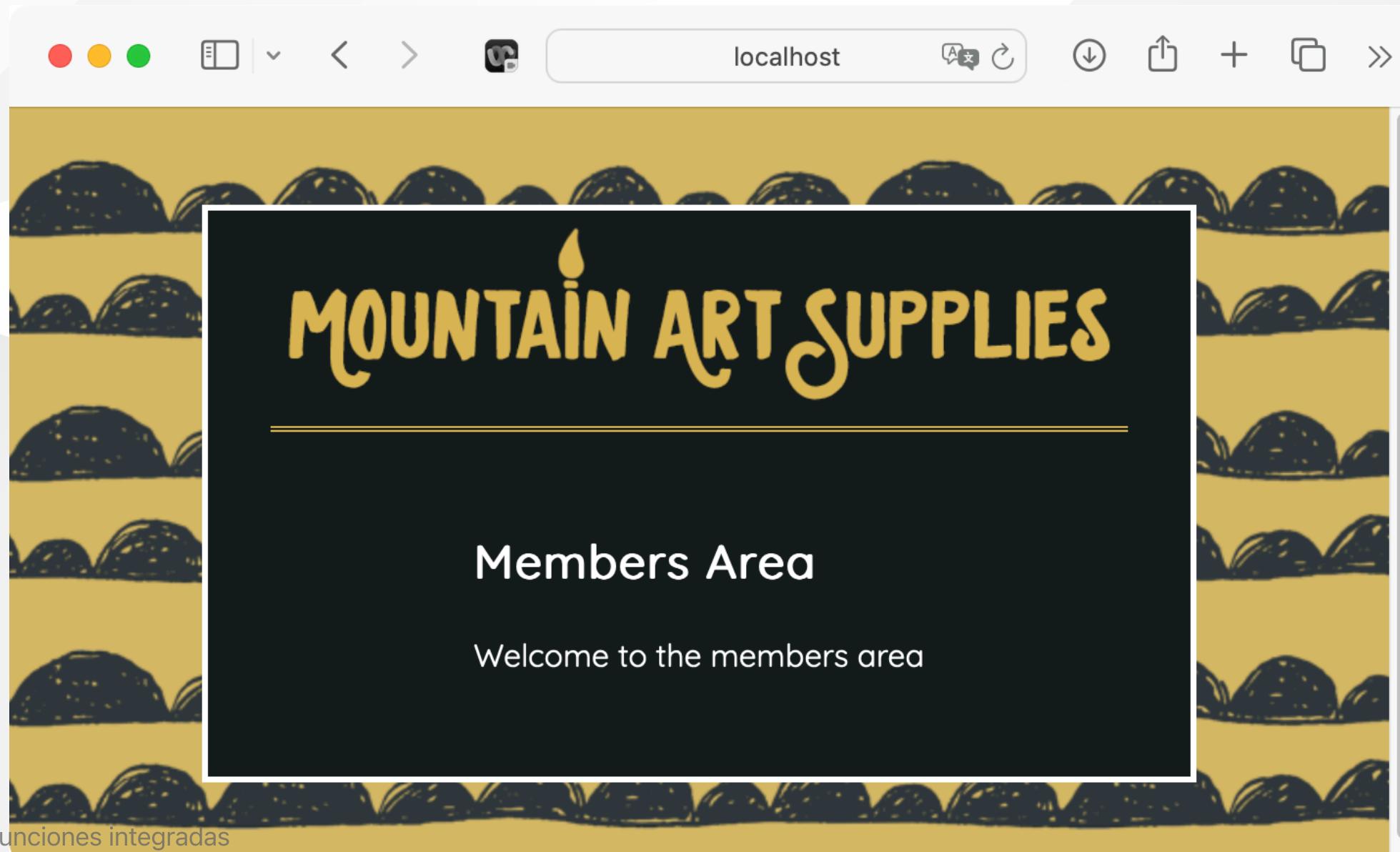
Ejemplo: redirigir usuarios usando cabeceras HTTP

```
// login.php
<?php include 'includes/header.php'; ?>

<h1>Login</h1>
<b>You need to log in to view this page.</b>
<p>(You learn how to create a login system in Unit 16.)</p>

<?php include 'includes/footer.php'; ?>
```

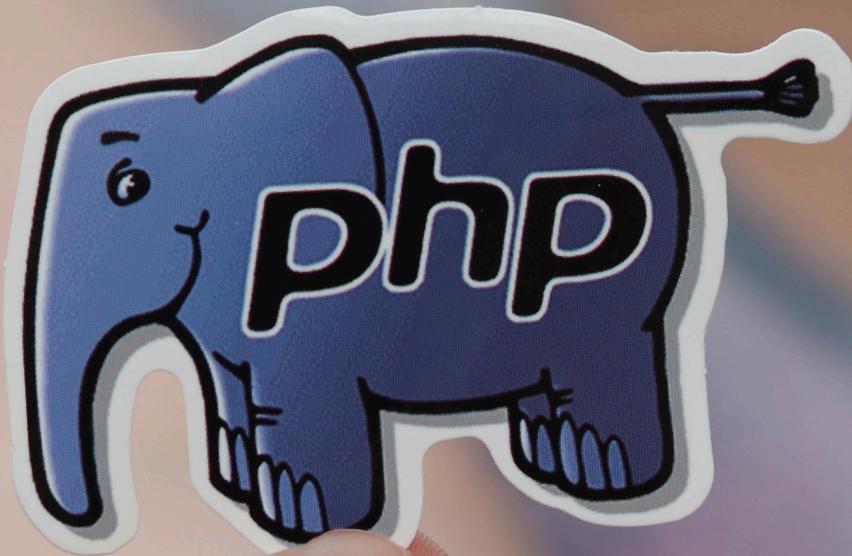
Ejemplo: redirigir usuarios usando cabeceras HTTP



Ejemplo: redirigir usuarios usando cabeceras HTTP

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el Paso 1, cambia el valor de la variable `$logged_in` a *false*. Comprueba que se redirige entonces a la página de inicio de sesión.



7. Funciones de archivos

Datos sobre archivos y eliminación de ficheros

Las funciones de archivo toman una ruta de archivo como parámetro y, a continuación, realizan una tarea, como devolver información sobre el archivo y su ruta, o eliminar el archivo.

Datos sobre archivos y eliminación de ficheros

La tabla de la siguiente diapositiva muestra las funciones de archivo. Algunas de estas funciones devuelven diferentes partes de una ruta de archivo; esas partes se describen en este diagrama:



PHP también tiene constantes incorporadas que contienen rutas:

- `__FILE__` contiene la ruta del archivo actual.
- `__DIR__` contiene el directorio del archivo actual.

Datos sobre archivos y eliminación de ficheros

FUNCTION	DESCRIPTION										
<code>file_exists(\$path)</code>	Checks if a file exists. Returns true if it exists, false if not.										
<code>filesize(\$path)</code>	Returns the size of the file in bytes.										
<code>mime_content_type(\$path)</code>	Returns the media type of the file. (See http://notes.re/media-types)										
<code>unlink(\$path)</code>	Tries to delete a file. Returns true if it worked, false if it did not.										
<code>pathinfo(\$path[, \$part])</code>	Returns parts of the filepath. You can specify the part of the path to retrieve. If you do not, it returns an array with the following four keys. <table><thead><tr><th>PART</th><th>DESCRIPTION</th></tr></thead><tbody><tr><td><code>PATHINFO_DIRNAME</code></td><td>Path to directory the file is in.</td></tr><tr><td><code>PATHINFO_BASENAME</code></td><td>Basename of the file.</td></tr><tr><td><code>PATHINFO_FILENAME</code></td><td>Filename (no extension).</td></tr><tr><td><code>PATHINFO_EXTENSION</code></td><td>File extension.</td></tr></tbody></table>	PART	DESCRIPTION	<code>PATHINFO_DIRNAME</code>	Path to directory the file is in.	<code>PATHINFO_BASENAME</code>	Basename of the file.	<code>PATHINFO_FILENAME</code>	Filename (no extension).	<code>PATHINFO_EXTENSION</code>	File extension.
PART	DESCRIPTION										
<code>PATHINFO_DIRNAME</code>	Path to directory the file is in.										
<code>PATHINFO_BASENAME</code>	Basename of the file.										
<code>PATHINFO_FILENAME</code>	Filename (no extension).										
<code>PATHINFO_EXTENSION</code>	File extension.										
<code>basename(\$path)</code>	Returns the basename of a file from a path.										
<code>dirname(\$path[, \$levels])</code>	Returns the path to the directory the specified file is in. If <code>\$levels</code> is specified, this is the number of parent directories to go up.										
<code>realpath(\$path)</code>	Returns the absolute path to the file.										

Ejemplo: Obtener información sobre archivos

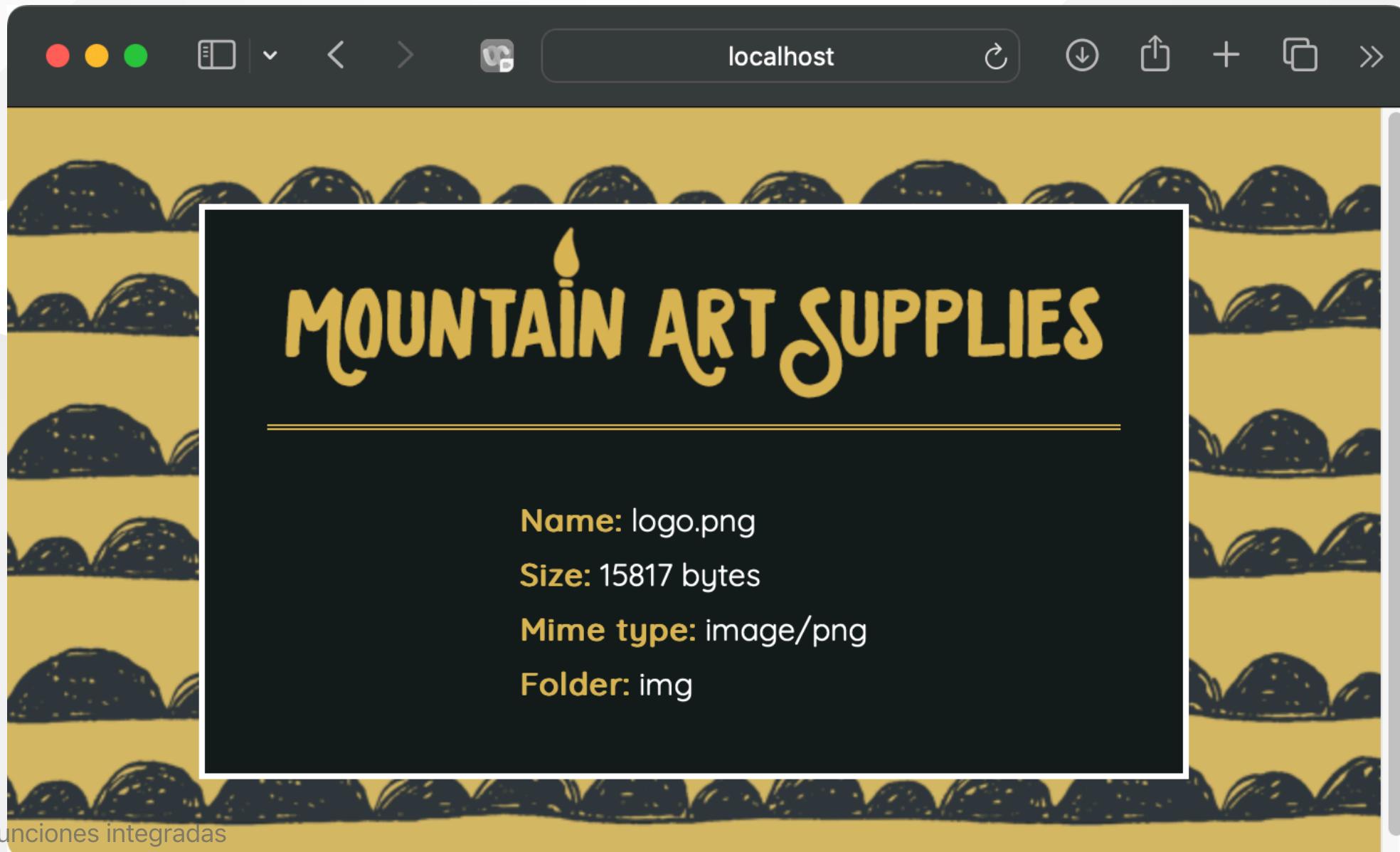
En el siguiente ejemplo se ponen en práctica algunas de las funciones vistas en la tabla anterior:

1. `$path` almacena una ruta a un archivo.
2. Una sentencia `if` utiliza `file_exists()` para comprobar si el fichero existe. Si existe, escribe información sobre el archivo.
3. `pathinfo()` muestra el nombre del archivo incluyendo su extensión (conocido como nombre base).
4. `filesize()` muestra el tamaño del archivo en bytes.
5. `mime_content_type()` muestra el tipo de medio del archivo.
6. `pathinfo()` muestra la carpeta en la que se encuentra el archivo.
7. Si el archivo no existe, se indica al usuario que no existe tal archivo.

Ejemplo: Obtener información sobre archivos

```
<?php  
① $path = 'img/logo.png';  
?>  
<?php include 'includes/header.php'; ?>  
② <?php if (file_exists($path)) { ?>  
③   <b>Name:</b>      <?= pathinfo($path, PATHINFO_BASENAME) ?><br>  
④   <b>Size:</b>       <?= filesize($path) ?> bytes<br>  
⑤   <b>Mime type:</b> <?= mime_content_type($path) ?><br>  
⑥   <b>Folder:</b>     <?= pathinfo($path, PATHINFO_DIRNAME) ?><br>  
?php } else { ?>  
⑦   <p>There is no such file.</p>  
?php } ?>  
<?php include 'includes/footer.php'; ?>
```

Ejemplo: Obtener información sobre archivos



Ejemplo: Obtener información sobre archivos

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el Paso 1, cambia `$path` a `img/pattern.png`. Verás el nuevo nombre y tamaño (el tipo mime y la carpeta siguen siendo los mismos).
- En el Paso 1, cambia `$path` a `img/nologo.png`. Como este archivo no existe, deberías ver el mensaje de error mencionado en el Paso 7.



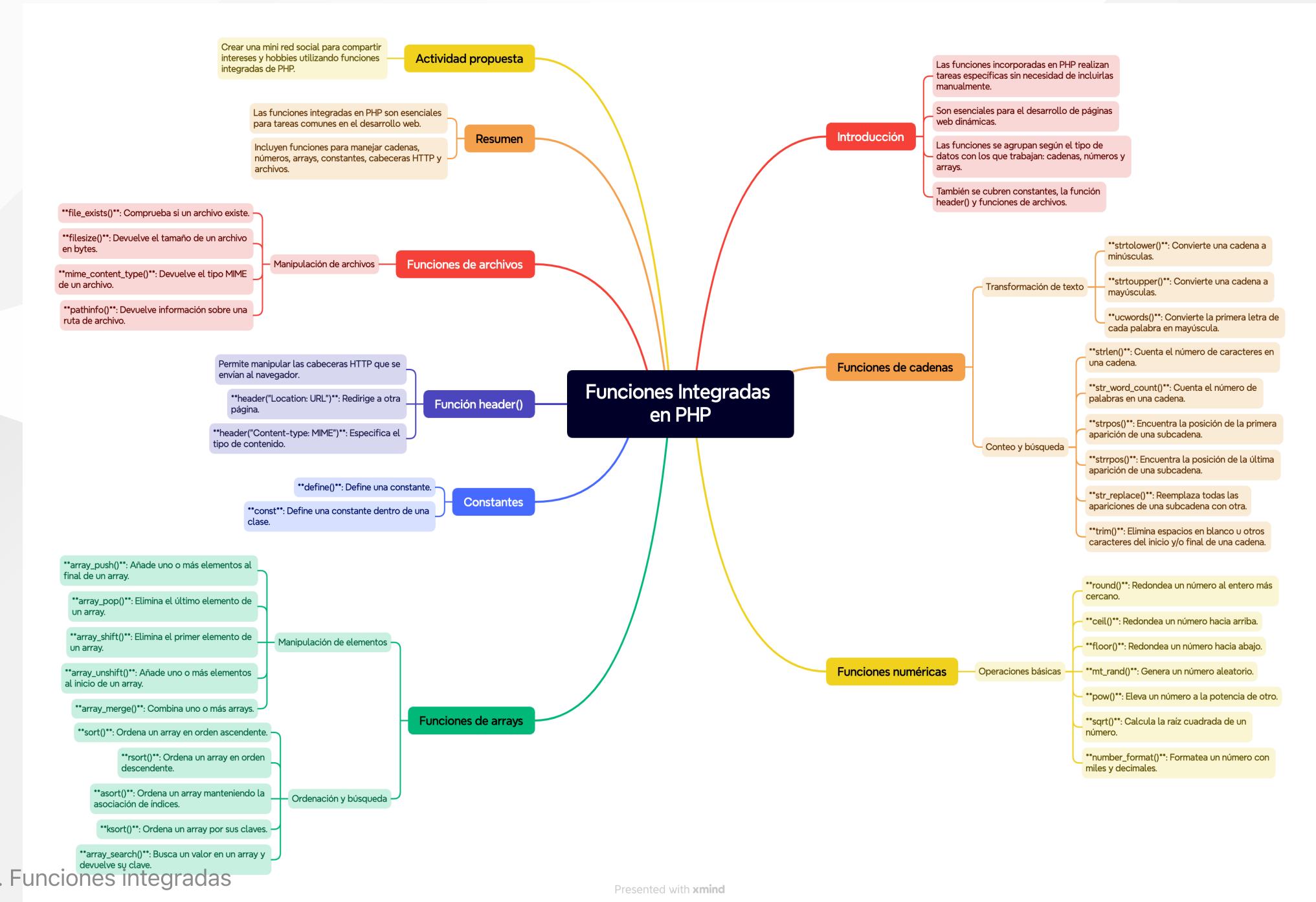
8. Resumen

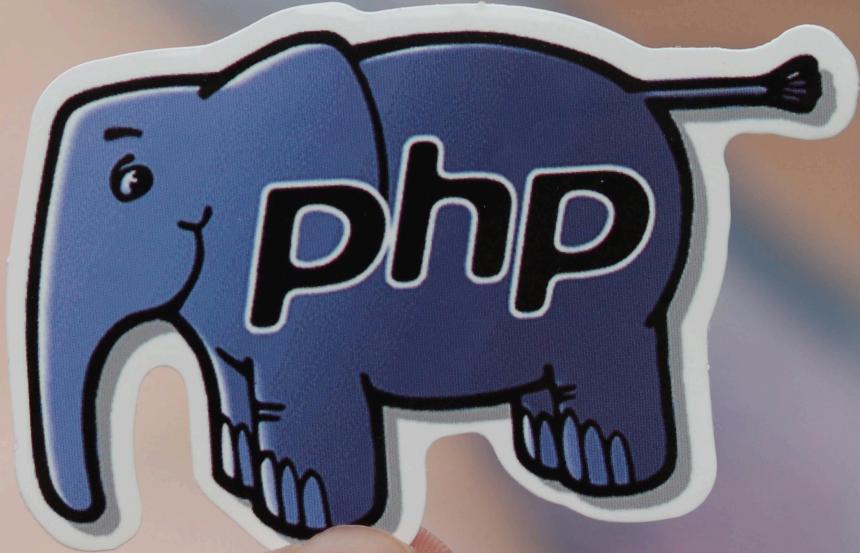
Resumen

- Las **funciones incorporadas** de PHP logran tareas que muchos programadores necesitan realizar al crear sitios.
- Las funciones incorporadas se llaman como cualquier función, pero no se añade una definición de función en la página.
- Las **funciones de cadena** encuentran, cuentan y reemplazan caracteres o cambian sus mayúsculas y minúsculas.
- Las **funciones numéricas** redondean números, seleccionan números aleatorios y realizan funciones matemáticas.

Resumen

- Las **funciones de array** añaden y eliminan elementos, ordenan el contenido de un array, buscan claves o valores y convierten arrays en cadenas y viceversa.
- Las **constantes** son como las variables, pero su valor no puede actualizarse una vez establecido.
- La función `header()` actualiza las **cabeceras HTTP** que se envían al navegador (y puede redirigir a los usuarios a otra página).





9. Actividad propuesta

Actividad propuesta

En esta actividad, vamos a crear una mini red social donde los usuarios pueden compartir sus intereses y hobbies. Vamos a utilizar las funciones integradas de PHP para realizar diferentes tareas que permitirán gestionar y mostrar de forma sencilla estos intereses de manera dinámica.

Actividad propuesta

Requisitos de la Actividad

1. Lista de Intereses:

- Define un array de intereses predefinidos para simular los intereses de los usuarios.
- Usa funciones de array para manipular esta lista de intereses.

2. Mostrar y Modificar Intereses:

- Utiliza funciones de cadena para mostrar y modificar los intereses.
- Los intereses deben mostrarse en una cadena separada por comas.
- Permite agregar nuevos intereses al array utilizando una función que simule la entrada del usuario.

Actividad propuesta

Requisitos de la Actividad

3. Numeración de Intereses:

- Usa funciones numéricas para numerar aleatoriamente los intereses y mostrar la lista con números aleatorios al lado de cada interés.
- Redondea algún valor si es necesario.
- Ordena la lista para mostrarla por orden ascendente en función de su identificador asignado aleatoriamente (ojo: trata de no repetir un mismo identificador)

Actividad propuesta

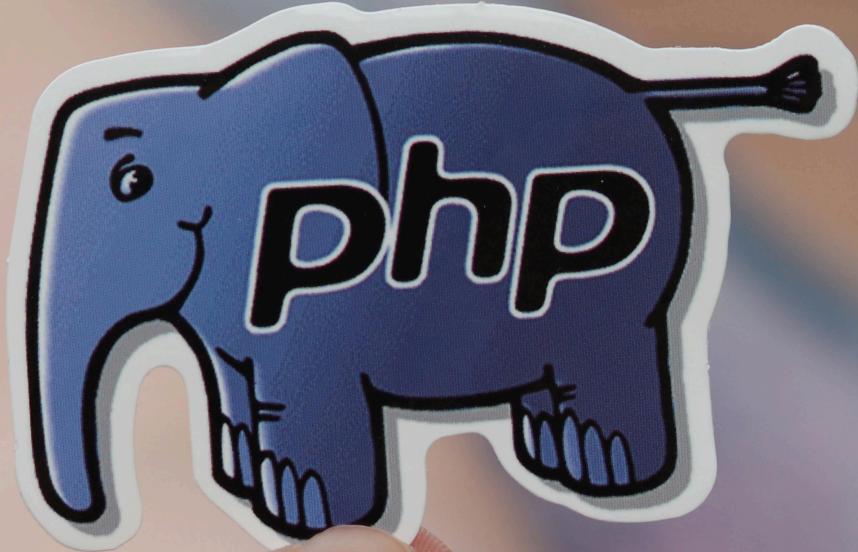
Requisitos de la Actividad

4. Constantes:

- Define una constante que represente el nombre de la red social y muéstralala en la página.

5. Redirección:

- Usa la función `header()` para redirigir a los usuarios a una página de agradecimiento (simulada) después de "agregar" un nuevo interés.



10. Referencias

Referencias

- [Funciones de cadena](#) - Documentación oficial de PHP sobre funciones de cadena.
- [Funciones numéricas](#) - Documentación oficial de PHP sobre funciones numéricas.
- [Funciones de arrays](#) - Documentación oficial de PHP sobre funciones de arrays.
- [Tipos de medios](#) - Información sobre tipos de medios (MIME types) de la IANA.
- [Función header\(\)](#) - Documentación oficial de PHP sobre la función `header()`.
- [Funciones de archivos](#) - Documentación oficial de PHP sobre funciones de archivos.

Anexo I: Tipos de medios

Muchos programas envían distintos tipos de archivos a través de Internet:

- Los navegadores muestran páginas HTML que también incluyen hojas de estilo, scripts, imágenes, audio y vídeo.
- Las aplicaciones de mensajería suelen utilizarse para enviar imágenes, audio y vídeo.
- Los programas de correo electrónico pueden recibir muchos tipos de archivos adjuntos

Los **tipos de medios (media types)** indican a estos programas el formato de los archivos y su contenido para que puedan ser procesados y/o visualizados correctamente.

Anexo I: Tipos de medios

Los tipos de medios se definieron originalmente en una especificación llamada Multipurpose Internet Mail Extensions y se denominaron **tipos MIME**.

El nombre se cambió a tipos de medios porque no sólo se utilizan en los correos electrónicos, sino también en muchos otros protocolos de Internet, como HTTP, y otros formatos de documento, como HTML. A veces se denominan **tipos de contenido (Content-Types)**.

Están controlados por la Autoridad de Asignación de Números de Internet (Assigned Numbers Authority, IANA), que también controla las direcciones IP, los nombres de dominio y otros protocolos de Internet.

Anexo I: Tipos de medios

Un tipo de medio se compone de dos partes separadas por una barra oblicua:

- El tipo es una categoría de formato de archivo, como audio, imagen, texto o vídeo.
- El subtipo identifica el formato del archivo; por ejemplo, una imagen puede ser GIF, JPEG, PNG o WebP.

Existen dos clases de tipos: **discretos** y **multiparte**. Los tipos discretos representan un único archivo o medio (como un único fragmento de texto, música o vídeo), mientras que los tipos multiparte representan un documento compuesto de varias partes.

Anexo I: Tipos de medios

TYPE	/	SUBTYPE
image	/	jpg
text	/	html
audio	/	mp3
video	/	quicktime

Lista completa de tipos de medios: <https://www.iana.org/assignments/media-types/>

Anexo I: Tipos de medios

Tipos discretos

Los tipos de medios discretos representan un archivo o medio individual. A continuación puedes ver los tipos discretos y algunos ejemplos de sus subtipos.

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio *imagen*

El tipo de medio *imagen* se utiliza para datos de imagen o gráficos, incluidas imágenes de mapa de bits y vectoriales. También se utiliza para formatos animados de imágenes fijas como GIF.

TYPE	DESCRIPTION
image/gif	GIF image
image/jpeg	JPEG image
image/png	PNG image
image/svg+xml	SVG image
image/webp	WebP image

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio audio

El tipo *audio* es para archivos de audio y datos musicales.

TYPE	DESCRIPTION
audio/mpeg	includes MPEG-based formats such as MP3
audio/mp4	includes formats such as M4A (non-protected audio), M4B (audiobooks / podcasts) and M4P (DRM protected audio)
audio/aac	Advanced Audio Coding format
audio/rtp- midi	for transporting midi in Real-Time Protocol (RTP) packets

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio vídeo

El tipo de vídeo es para archivos de vídeo y datos.

TYPE	DESCRIPTION
video/h264	h264 video
video/mp4	mp4 video
video/quicktime	Quicktime video
video/raw	Raw video

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio texto

El tipo de texto es para datos de sólo texto que están diseñados para que los lean las personas o para que los procesen los ordenadores.

TYPE	DESCRIPTION
text/plain	Plain text
text/html	HTML documents
text/css	CSS style sheets
text/php	PHP code
text/csv	Comma-separated values

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio fuente

El tipo de *fuente* se utiliza para especificar datos de fuentes o tipos de letra.

TYPE	DESCRIPTION
font/ttf	True Type font files
font/oft	Open-Type font files
font/woff	Web Open Font Format

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio aplicación

El tipo de *aplicación* es para archivos que son utilizados por muchos tipos de aplicaciones y que no entran en ninguno de los otros tipos.

TYPE	DESCRIPTION
application/javascript	JavaScript files
application/pdf	PDF files
application/sql	SQL files
application/zip	ZIP files
application/octet-stream	Generic binary data, or unknown files

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio modelo

El tipo de *modelo* es para datos de modelos u objetos y escenas 3D.

TYPE	DESCRIPTION
model/vrml	Virtual Reality Modelling Language
model/vnd.collada+xml	Collada file format for 3D applications, written in XML

Anexo I: Tipos de medios

Tipos discretos

Tipo de medio de ejemplo

El tipo de *ejemplo* se utiliza como marcador de posición para ejemplos que muestran cómo se utilizan los tipos de medios. Sólo deben utilizarse en código de ejemplo y documentación.

TYPE	DESCRIPTION
example/format	Here, example is being used to describe a type
audio/example	Here, example is being used to describe a subtype

Anexo I: Tipos de medios

Tipos multiparte

Los tipos multiparte se utilizan para documentos que están divididos en partes. Esto puede deberse a una de estas dos razones

- Las partes individuales del documento tienen distintos tipos de soporte.
- Un campo grande necesita ser dividido en partes para su transmisión, y luego reensamblado cuando ha sido recibido

A continuación se muestran los dos tipos de soporte multiparte.

Anexo I: Tipos de medios

Tipos multiparte

Tipo mensaje (*message*)

El tipo de *mensaje* es para mensajes que incluyen otros mensajes (por ejemplo, un correo electrónico que contiene un correo electrónico reenviado), o para mensajes muy grandes que se envían en trozos.

TYPE	DESCRIPTION
message/rfc822	For forwarded messages and replied-to messages
message/partial	For parts of messages that are split up into chunks

Anexo I: Tipos de medios

Tipos multiparte

Tipo multiparte (*multipart*)

El tipo *multipart* se utiliza para datos que contienen múltiples componentes que pueden tener su propio tipo de medio.

TYPE	DESCRIPTION
multipart/form-data	Used to send form data that may also contain files

Anexo I: Tipos de medios

Información adicional en los subtipos

Los subtipos pueden utilizar prefijos, sufijos y parámetros opcionales para proporcionar información adicional.

Anexo I: Tipos de medios

Información adicional en los subtipos

Prefijos

Se pueden utilizar los tres prefijos siguientes al principio de un subtipo. Estos prefijos a veces se denominan árboles de registro, ya que contienen más información sobre cómo se registran los tipos de medios en la IANA.

- vnd. indica que son para utilizar con software creado por un proveedor específico
- prs. indica que se trata de un formato personal o experimental
- x- indica que no están registrados en la IANA

Anexo I: Tipos de medios

Información adicional en los subtipos

Prefijos

TYPE	DESCRIPTION
application/vnd.apple.keynote	Apple Keynote
application/vnd.ms-powerpoint	Microsoft Powerpoint
image/vnd.adobe.photoshop	Adobe Photoshop
application/x-x509-ca-cert	X.509 security certificates

Anexo I: Tipos de medios

Información adicional en los subtipos

Sufijos

Cuando un formato de archivo se deriva de una norma o de otro formato de archivo, puede indicarse mediante un sufijo.

Para ello, el subtipo va seguido del símbolo + y, a continuación, del nombre de la norma o del formato en el que se basa.

Los ejemplos muestran que los formatos se escriben utilizando el Lenguaje Extensible de Marcado (XML).

Anexo I: Tipos de medios

Información adicional en los subtipos

Sufijos

TYPE	DESCRIPTION
image/svg+xml	Indicates that SVG is written in XML
application/mathml+xml	Indicates that MathML is written in XML
application/atom+xml	Indicates that the Atom feed is written in XML

Anexo I: Tipos de medios

Información adicional en los subtipos

Parámetros

Los parámetros se especifican añadiendo un punto y coma después del tipo de medio, seguido del nombre del parámetro, un símbolo igual y el valor del parámetro.

TYPE	DESCRIPTION
text/html; charset=UTF-8	Indicates that the HTML file uses UTF-8 character encoding
video/mp4; codecs="ac-3"	Indicates that the MP4 video file uses AC-3 audio encoding

Anexo I: Tipos de medios

Información adicional en los subtipos

Para obtener una lista completa de tipos de medios, consulta:

<https://www.iana.org/assignments/media-types/>

NOTA: Los navegadores utilizan el tipo de medio de los archivos, no su extensión, para determinar cómo procesarlos. Por lo tanto, los servidores web deben estar configurados para enviar el tipo de medio correcto en la cabecera `Content-Type` de la respuesta; de lo contrario, el archivo podría no renderizarse o manejarse correctamente.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

Una **ruta absoluta** es una ruta desde la raíz del sistema (la carpeta superior de un sistema operativo) a otro archivo o directorio (carpeta). La raíz del sistema se indica en:

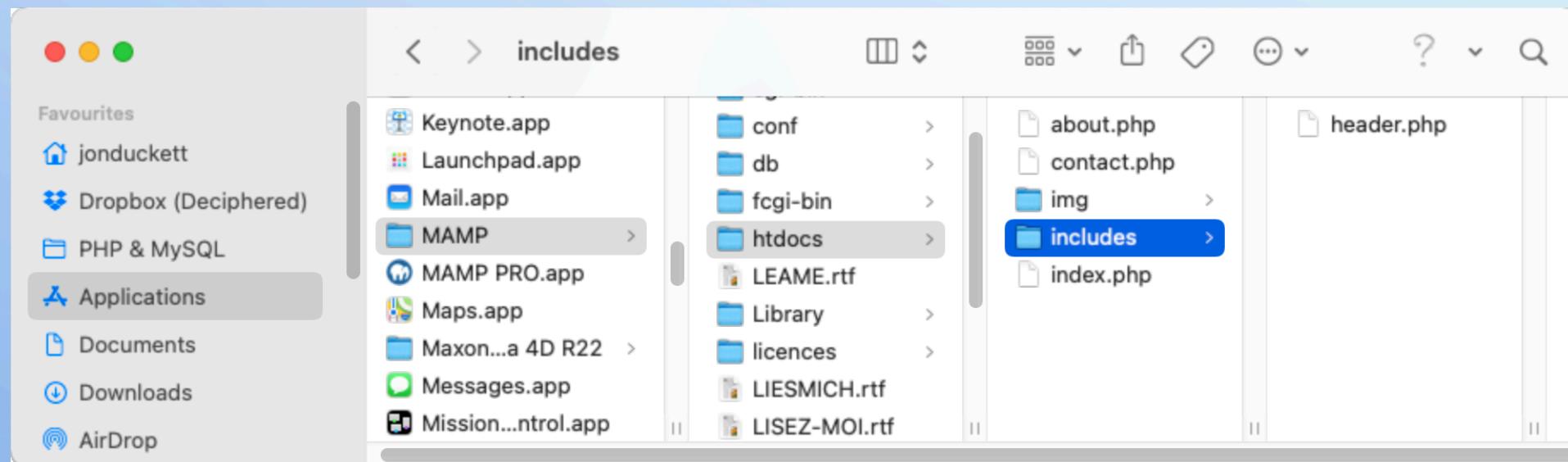
- Mac/Unix mediante una barra oblicua / .
- Windows mediante una letra de unidad, dos puntos y barra diagonal inversa, que suele ser C:\

Una **ruta relativa** apunta a la ubicación de un archivo relativa al directorio de trabajo actual (`cwd` : *Current Work Directory*). Cuando se escribe PHP, el directorio de trabajo actual es la carpeta que contiene el script que se está procesando en ese momento.

Si las páginas utilizan un *include* en una carpeta diferente, cuando el *include* se está ejecutando, el `cwd` es la carpeta que contiene el *include*.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

Esta captura de pantalla muestra la ubicación de los archivos que se ejecutan en MAMP en un Mac.



A continuación, puedes ver algunos ejemplos de rutas absolutas que se relacionan con las carpetas mostradas en esta captura de pantalla.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

La ruta absoluta a `index.php` es

```
/Applications/MAMP/htdocs/index.php
```

El archivo `index.php` puede incluir `header.php` utilizando la ruta relativa:

```
includes/header.php
```

Anexo II: Rutas absolutas VS relativas (y el *document root*)

Cuando el código en el archivo `index.php` se está ejecutando, el `cwd` es:

```
/Applications/MAMP/htdocs/
```

En `index.php`, la ruta relativa `img/logo.png` apuntaría al archivo `logo.png` de la carpeta `img`.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

La ruta absoluta a header.php es

```
/Applications/MAMP/htdocs/includes/header.php
```

Si una ruta empieza por `../`, indica que el archivo está en un directorio superior al directorio de trabajo actual.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

Cuando el código en el archivo `header.php` se está ejecutando, el `cwd` es:

```
/Applications/MAMP/htdocs/includes/
```

En `header.php`, la ruta relativa `../img/logo.png` apuntaría al archivo `logo.png` de la carpeta `img`.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

Carpeta Document Root

Los sitios web tienen otro tipo de carpeta raíz llamada **carpeta raíz de documentos (document root)**. Se trata de una carpeta del servidor web que corresponde al nombre de dominio/host.

Por ejemplo, cuando un servidor web aloja el nombre de dominio

```
http://example.com
```

Se podría asignar a la siguiente carpeta en el servidor:

```
/var/www/examplecom/
```

Anexo II: Rutas absolutas VS relativas (y el *document root*)

Carpeta Document Root

Todos los archivos que pueda solicitar un navegador deben estar dentro de la carpeta raíz de documentos, incluidas las páginas PHP, las imágenes, los archivos CSS y JavaScript.

Si se solicita la URL

```
http://example.com/index.php
```

el servidor web ejecutaría el archivo

```
/var/www/examplecom/index.php
```

Anexo II: Rutas absolutas VS relativas (y el *document root*)

Carpeta Document Root

Si se solicita la URL

```
http://example.com/about.php,
```

el servidor web ejecutaría el archivo

```
/var/www/examplecom/about.php
```

La carpeta raíz de documentos por defecto para:

- MAMP en Mac es /Applications/MAMP/htdocs/
- XAMPP en PC es C:\XAMPP\htdocs\

Anexo II: Rutas absolutas VS relativas (y el *document root*)

URLs relativas a la raíz

Cuando una ruta utilizada en una página web comienza con una barra oblicua, ésta indica la carpeta raíz del documento.

Una **ruta relativa a la raíz** comienza con una barra oblicua y va seguida de una ruta relativa a la carpeta raíz del documento.

En la estructura de carpetas que se muestra en la captura de pantalla anterior, cualquier página podría utilizar la ruta /img/logo.png para hacer referencia al archivo logo.png de la carpeta img.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

URLs relativas a la raíz

Sólo debes utilizar rutas relativas a la raíz en URLs para archivos que el navegador pueda solicitar. En lo que concierne al intérprete PHP, la barra oblicua se referiría a una carpeta diferente.

Cuando tu código PHP intenta hacer referencia a otro archivo, debes utilizar rutas absolutas.

Anexo II: Rutas absolutas VS relativas (y el *document root*)

URLs relativas a la raíz

Los sitios web a menudo almacenan la ruta a la carpeta raíz del documento en una constante, y luego la utilizan al comienzo de cualquier ruta a otros archivos. Por ejemplo:

```
define('ROOT_PATH', '/path/to/document/root/');
include(ROOT_PATH . 'includes/header.php');
```

Esto tiene dos ventajas:

- Se ahorra escribir la ruta a la raíz del documento.
- Si la ruta a la carpeta raíz del documento cambia (quizás porque el sitio se traslada a otro servidor), la ruta sólo necesita actualizarse en un lugar: la definición de la constante *ROOT_PATH*.