Bloque B

Páginas Web Dinámicas

Unidad 7. Imágenes & Archivos

Contenidos

1. Introducción

Introducción

Esta unidad muestra cómo permitir a los visitantes **subir imágenes al servidor y cómo mostrarlas de forma segura en tus páginas PHP**. Estas técnicas **también funcionan para otros tipos de archivos**.

Primero, aprenderás **cómo los usuarios suben imágenes y cómo el servidor las recibe**. Veremos cómo:

- El control de subida de archivos HTML se utiliza en un formulario HTML para que los usuarios puedan subir archivos.
- El intérprete de PHP agrega datos sobre el archivo a un array superglobal llamado `$_FILES`.
- El archivo se coloca en una carpeta temporal en el servidor.
- El archivo debe ser movido a una carpeta donde se almacenarán los archivos subidos.

Introducción

A continuación, aprenderemos a validar los archivos que se han subido y comprobar que:

- El nombre del fichero sólo contiene caracteres permitidos.
- No existe ya un archivo con ese nombre.
- Es un tipo de medio y extensión de archivo permitidos.
- El tamaño del archivo no es demasiado grande.

Introducción

Por último, descubrirás cómo manipular las imágenes para crear:

- Miniaturas de la imagen.
- Versiones recortadas de la imagen.

A lo largo de la unidad, conocerás más funciones integradas que te ayudarán con estas tareas. Aunque la unidad demuestra estas técnicas utilizando imágenes, también pueden utilizarse para permitir a los visitantes cargar archivos de audio, vídeo, PDF y de otros tipos.

2. Subiendo ficheros desde un navegador

Subiendo ficheros desde un navegador

Los formularios HTML pueden contener un **control de carga de archivos**, que los visitantes pueden utilizar para cargar archivos en el servidor.

Al crear un formulario HTML que permita a los visitantes cargar archivos, la etiqueta <form> de apertura debe tener los tres atributos siguientes:

- `method` con un valor de *POST* para especificar que **el formulario debe enviarse mediante HTTP POST** (porque los archivos no deben enviarse mediante HTTP GET)
- `enctype` con valor *multipart/form-data* para especificar el **tipo de codificación** que debe utilizar el navegador para enviar los datos
- `action` cuyo valor es **el archivo PHP al que deben enviarse los datos** del formulario

Subiendo ficheros desde un navegador

El control de carga de archivos se crea utilizando el elemento HTML `<input>`. Su atributo `type` debe tener el valor *file*. En el navegador, esto crea un botón que abre una nueva ventana que permite al usuario seleccionar el archivo que desea subir:

```
<input type="file" name="image">
```

Al igual que los otros controles de formulario, el control de entrada del archivo envía un par nombre/valor al servidor:

- El **nombre** es el valor del atributo `name` para ese control de archivo (en el ejemplo anterior se llama *image*).
- El **valor** es el archivo que el usuario está enviando.

Subiendo ficheros desde un navegador

Para ayudar a restringir el tipo de archivo que un usuario puede cargar, el control de entrada de archivos tiene un atributo `accept`.

Su valor debe ser una lista separada por comas de los tipos de medios que el sitio acepta. (Los tipos de medios suelen denominarse tipos MIME, para más información consultar el *Anexo I de la Unidad 5*).

```
<input type="file" name="image" accept="image/jpeg, image/png">
```

Subiendo ficheros desde un navegador

Si se utiliza el atributo `accept`, cuando el visitante haga clic en el botón para subir un archivo, los navegadores modernos desactivarán los archivos que no estén en la lista de tipos aceptados para que no puedan ser seleccionados.

Esto es útil para la usabilidad, pero **no se puede confiar en el atributo `accept` para restringir el tipo de archivos que suben los visitantes** porque pueden anular la configuración y los navegadores más antiguos no lo soportan. (Chrome 10, Internet Explorer 10, Firefox 10 y Safari 6 fueron las primeras versiones de los principales navegadores en admitir esta función). Por lo tanto, también deberías intentar validar el tipo de medio en el servidor utilizando PHP (lo veremos en esta unidad).

Subiendo ficheros desde un navegador

Para permitir todos los subtipos de un tipo de medio, se puede añadir un asterisco en lugar de un subtipo.
Lo siguiente permite todos los formatos de imagen (incluyendo BMP, GIF, JPEG, PNG, TIFF y WebP):

```
<input type="file" accept="image/*">
```

Ejemplo: Subiendo ficheros desde un navegador

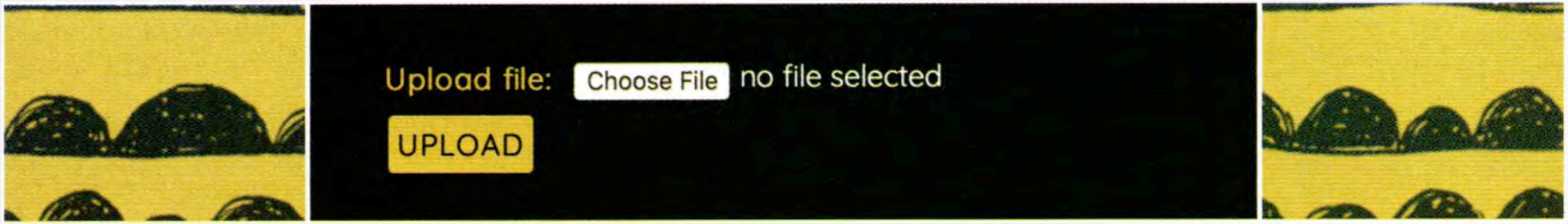
1. El siguiente formulario permite a los visitantes cargar imágenes. Se utiliza en todos los ejemplos de este capítulo. La etiqueta `<form>` de apertura necesita:
    - atributo `method` con el valor *POST*.
    - atributo `enctype` con el valor *multipart/form-data*.
    - atributo `action` especificando el archivo al que enviar los datos del formulario (este valor cambia en cada ejemplo).
2. Para crear un control de carga de archivos, el elemento `<input>` lleva un atributo `type` cuyo valor es *file*. Como los ejemplos de este capítulo muestran cómo permitir a los visitantes cargar una imagen, el valor del atributo `name` es *image*.
3. El botón *submit* se utiliza para enviar el formulario.

Ejemplo: Subiendo ficheros desde un navegador

```
① <form method="post" action="filename.php" enctype="multipart/form-data">
    <label for="image"><b>Upload file:</b></label>
②   <input type="file" name="image" accept="image/*" id="image"><br>
③   <input type="submit" value="Upload">
  </form>
```

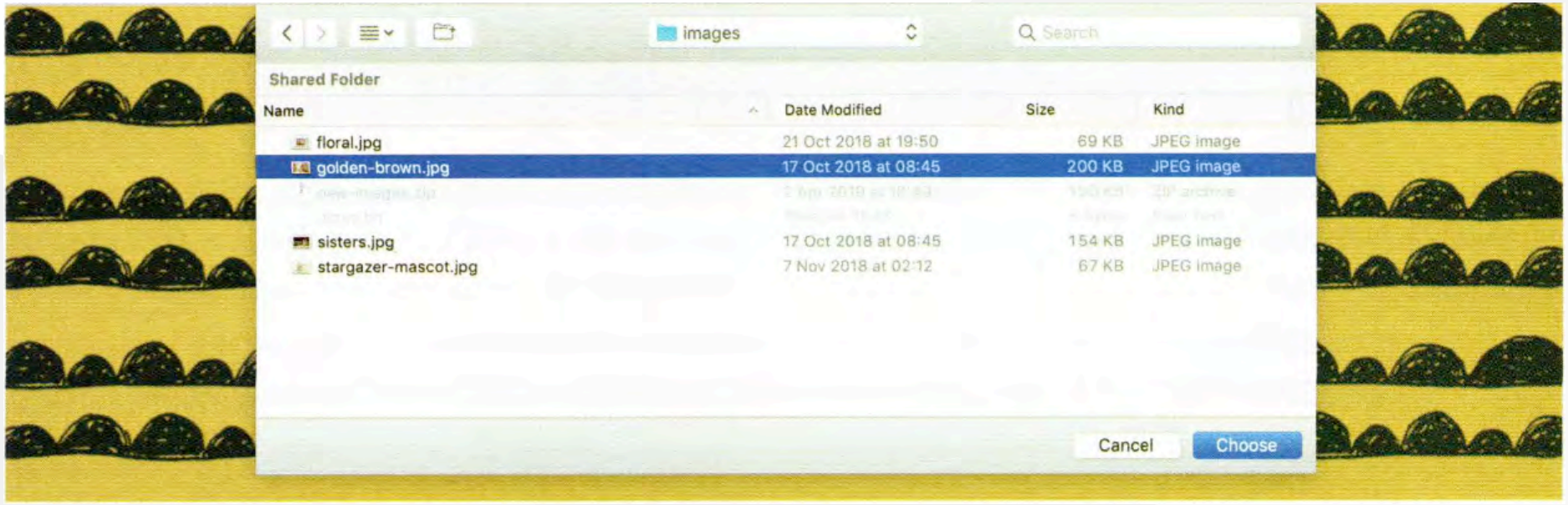
Ejemplo: Subiendo ficheros desde un navegador

La imagen siguiente muestra el formulario que crea el control de carga de archivos. Cuando se selecciona una imagen, el texto junto al botón se sustituye por el nombre del archivo.



Ejemplo: Subiendo ficheros desde un navegador

Aquí se puede ver la ventana que se abre cuando el usuario hace clic en *Choose File*. Un archivo de texto y un archivo zip están **desactivados porque no son imágenes**.



> El aspecto visual de la ventana que aparece para seleccionar archivos varía según el navegador y el sistema operativo (no es posible controlar su aspecto mediante CSS).

3. Recibiendo archivos en el servidor

Recibiendo archivos en el servidor

Cuando se carga un archivo a través de una página web, **el servidor web lo guarda en una carpeta temporal**, y el intérprete de PHP **almacena los detalles sobre el archivo en un array superglobal** llamado `$_FILES` .

Recibiendo archivos en el servidor

Un formulario puede tener múltiples controles de subida de archivos, por lo que el intérprete de PHP creará u**n elemento en el array superglobal** `$_FILES` **por cada control de subida de archivos** que envíe el formulario.

El **nombre del elemento** coincide con el nombre del control de subida de archivos y su **valor** es un array de datos sobre el archivo que se subió a través de ese control de formulario.

Recibiendo archivos en el servidor

La siguiente tabla muestra la información que el array superglobal `$_FILES` almacena para cada archivo que se ha subido.

Las imágenes de esta unidad se suben utilizando un control de subida de archivos cuyo nombre es *image*, por lo que el array `$_FILES` tendrá un elemento llamado *image*, y su valor será un array que contendrá información sobre esa imagen.

KEY	VALUE	HOW TO ACCESS VALUE
name	File name	$_FILES['image']['name']
tmp_name	Temporary location of the file (set by the PHP interpreter)	$_FILES['image']['tmp_name']
size	Size in bytes	$_FILES['image']['size']
type	Media type (according to the browser)	$_FILES['image']['type']
error	0 if file uploaded successfully, an error code if there was a problem	$_FILES['image']['error']

Recibiendo archivos en el servidor

Una vez que se ha subido un archivo, el código PHP debe comprobar que el intérprete PHP no ha encontrado ningún error con la subida.

Si la clave de error en el array que fue creado para este archivo tiene un valor de 0 entonces significa que el intérprete PHP no encontró ningún error.

```php
if ($_FILES['image']['errors'] === 0) {
    // Process image
} else {
    // Show error message
}
```

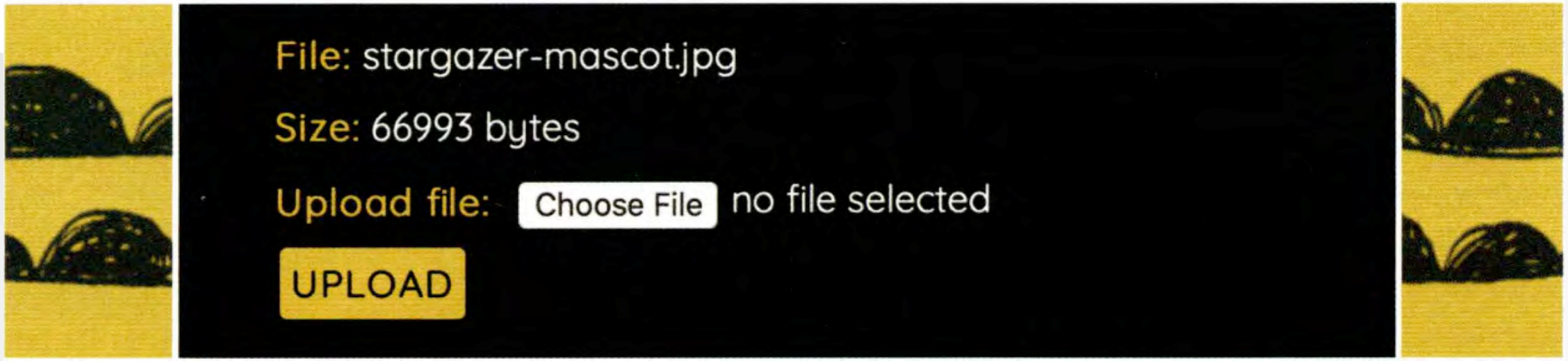
Ejemplo: Comprobar que se ha cargado un archivo

1. La variable `$message` se inicializa con una cadena en blanco. Almacena un mensaje si se envía el formulario.
2. Si el formulario se ha enviado utilizando *HTTP POST*...
3. Una sentencia if comprueba que no hay errores.
4. Si no hay errores, el nombre y el tamaño del archivo se almacenan en `$message` .
5. En caso contrario, `$message` almacena un mensaje de error.
6. Se muestra el valor de la variable `$message` .

Ejemplo: Comprobar que se ha cargado un archivo

```php
<?php
① $message = '';                                                    // Initialize
② if ($_SERVER['REQUEST_METHOD'] == 'POST') {                       // If form sent
③     if ($_FILES['image']['error'] === 0) {                        // If no errors
④         $message  = '<b>File:</b> ' . $_FILES['image']['name'] . '<br>';   // File name
          $message .= '<b>Size:</b> ' . $_FILES['image']['size'] . ' bytes'; // File size
      } else {                                                      // Otherwise
⑤         $message  = 'The file could not be uploaded.';            // Error message
      }
  }
  ?> ...
⑥ <?= $message ?>
  <form method="POST" action="upload-file.php" enctype="multipart/form-data">
    <label for="image"><b>Upload file:</b></label>
    <input type="file" name="image" accept="image/*" id="image"><br>
    <input type="submit" value="Upload">
  </form>
```

Ejemplo: Comprobar que se ha cargado un archivo



Actividad: Comprobar que se ha cargado un archivo

Un cliente te ha solicitado que crees una funcionalidad para un portal donde los usuarios puedan subir archivos. Como desarrollador, deberás asegurarte en tu implementación de que el fichero se ha subido correctamente al servidor. Si la subida ha sido exitosa, deberás mostrar un mensaje de confirmación.

4. Moviendo un archivo a su destino

Moviendo un archivo a su destino

La función `move_uploaded_file()` de PHP **mueve un archivo** de su ubicación temporal a donde debe ser almacenado en el servidor.

Moviendo un archivo a su destino

Cuando se carga un archivo en el servidor, se le asigna un nombre de archivo temporal y se coloca en una carpeta temporal. (El nombre de archivo temporal es creado por el intérprete PHP).

El intérprete de PHP borrará el archivo temporal de esta carpeta cuando el script termine de ejecutarse. Por lo tanto, para almacenar un archivo subido en el servidor, se debe llamar a la función `move_uploaded_file()` para moverlo a otra carpeta. Tiene dos parámetros:

- La **ubicación temporal** del archivo
- El **destino** donde debe guardarse el archivo

Devuelve *true* si pudo mover el archivo a la nueva ubicación y *false* en caso contrario.

Moviendo un archivo a su destino

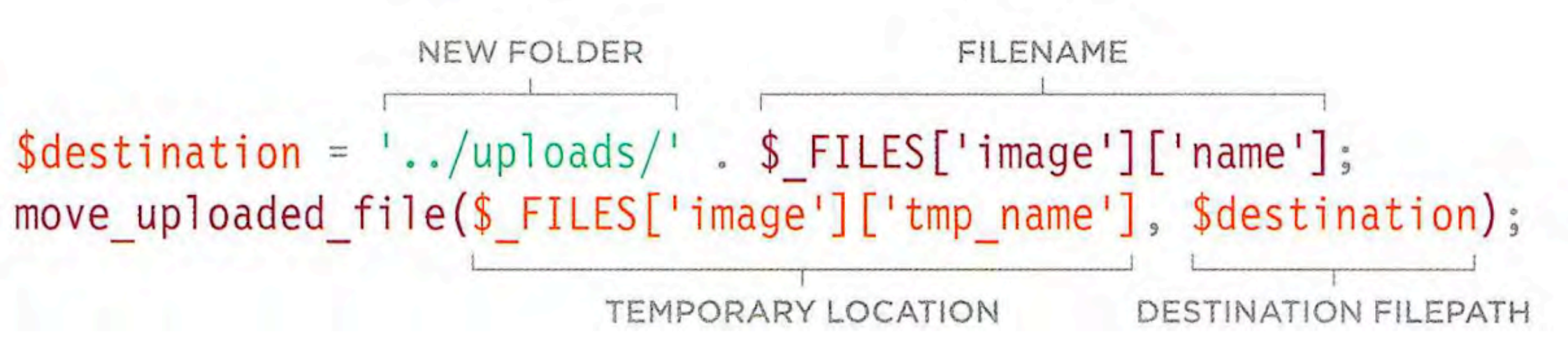
El destino (la ubicación donde debe guardarse el archivo) está formado por:

- Ruta a la carpeta que almacenará el archivo subido (esta carpeta debe haber sido creada antes de intentar mover un archivo a ella).
- Nombre del fichero (su nombre original o un nombre nuevo).

Si se desea utilizar el nombre original del fichero que se ha subido, se puede acceder a él a través del array que el intérprete PHP ha creado para ese fichero. Su clave es name.

Moviendo un archivo a su destino

En el siguiente ejemplo, la ruta del archivo de destino se almacena en una variable llamada `$destination`. Se crea especificando la carpeta *uploads*, seguida del nombre original del archivo utilizado al subir la imagen.



Moviendo un archivo a su destino

Permisos

Los permisos del directorio de destino deben:

- Permitir al servidor web leer/escribir archivos - esto le permite guardar y mostrar imágenes y otros tipos de ficheros subidos al servidor.
- Deshabilitar los permisos de ejecución: esto impide que se ejecuten secuencias de comandos maliciosas.

Moviendo un archivo a su destino

Verificar que un archivo fue subido

La función `move_uploaded_file()` de PHP verifica que un archivo fue subido vía HTTP POST antes de moverlo. Si necesitas utilizar un archivo antes de moverlo, hay que utilizar `is_uploaded_file()` de PHP para realizar esta comprobación.

Ejemplo: moviendo un archivo subido

1. Una variable llamada `$moved` se inicializa con un valor de false. Esto cambiará a true si la imagen se mueve con éxito.
2. Si el formulario fue enviado y no hubo errores...
3. `$temp` contiene la ubicación donde el intérprete PHP almacenó temporalmente el archivo.
4. `$path` almacena la ruta donde se guardará el archivo. (El archivo mantendrá el mismo nombre de archivo que tenía cuando fue subido).

Ejemplo: moviendo un archivo subido

5. `move_uploaded_file()` intenta mover el archivo desde su ubicación temporal (en `$temp`) a la nueva ubicación (en `$path`) . La función devuelve true si funcionó o false si falló . Este valor reemplaza el valor almacenado en la variable `$moved` en el Paso 1.
6. Una sentencia condicional comprueba si `$moved` tiene el valor true, indicando que el movimiento funcionó.
7.
8. Si es así, `$message` almacena una etiqueta HTML <img> que muestra la imagen subida.
9. En caso contrario, `$message` almacena un mensaje de error.
10. El valor almacenado en `$message` se muestra al usuario.

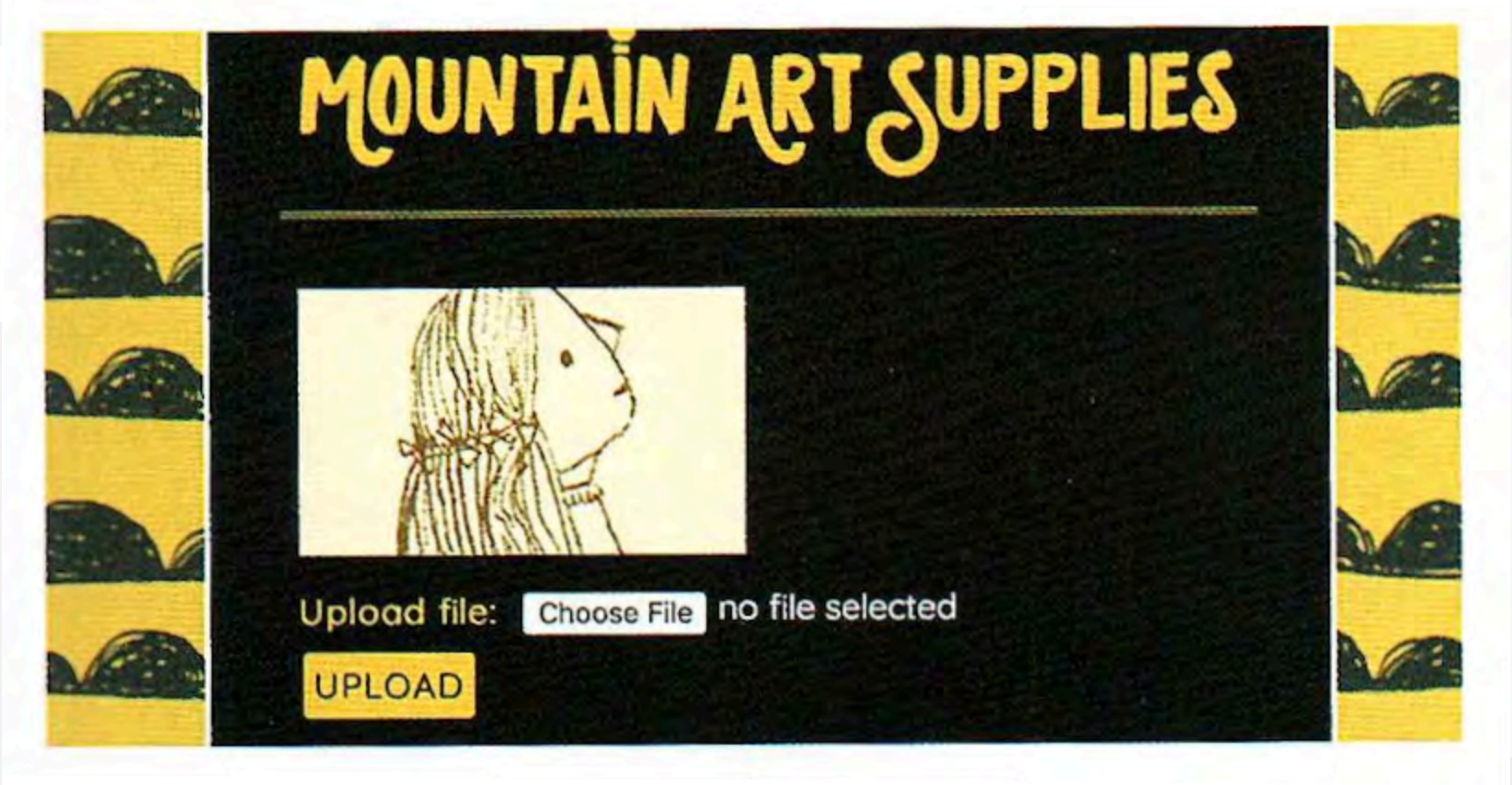
Ejemplo: moviendo un archivo subido

```php
<?php
$message = '';                                  // Initialize
$moved   = false;                               // Initialize

if ($_SERVER['REQUEST_METHOD'] == 'POST') { // If sent +
    if ($_FILES['image']['error'] === 0) {  // No errors
        // Store temporary path and new destination
        $temp = $_FILES['image']['tmp_name'];
        $path = 'uploads/' . $_FILES['image']['name'];
        // Move the file and store result in $moved
        $moved = move_uploaded_file($temp, $path);
    }

    if ($moved === true) { // If move worked, show image
        $message = '<img src="' . $path . '">';
    } else {                    // Else store error message
        $message = 'The file could not be saved.';
    }
}
?> ...
<?= $message ?>
```

Ejemplo: moviendo un archivo subido



Actividad: Moviendo un archivo subido

Modifica la actividad anterior (*Actividad: Comprobar que se ha cargado un archivo*) para almacenar la imagen de perfil que el usuario suba en un directorio denominado *images* ubicado en */var/www/*. Comprueba que se ha realizado correctamente la operación y en caso afirmativo muestra la imagen almacenada en la página PHP (en caso contrario, muestra un mensaje de error).

5. Limpiando un nombre de archivo y archivos duplicados

Limpiando un nombre de archivo y archivos duplicados

Antes de mover un archivo de su ubicación temporal, debes:

1. **Eliminar los caracteres** del nombre del fichero **que puedan causar problemas**.
2. **Asegurarte de que no se sobrescribirá otro archivo** con el mismo nombre.

Limpiando un nombre de archivo y archivos duplicados

Los caracteres como los ampersands, los dos puntos, los puntos y los espacios deben eliminarse de los nombres de archivo, ya que pueden causar problemas. Para ello, se pueden sustituir los caracteres que no sean A-Z, a-z y 0-9, por un guión.

Limpiando un nombre de archivo y archivos duplicados

1. Utiliza la función `pathinfo()` de PHP, para obtener el nombre base y la extensión del archivo.
2. Utiliza la función `preg_replace()` de PHP para reemplazar cualquier carácter en el nombre base que no sea A-L, a-z, y 0-9, por un guión.
3. Crea la ruta del archivo de destino uniendo el directorio de subida, el nombre base, un punto y la extensión del archivo. Este valor debe guardarse en una variable.

```
① $basename  = pathinfo($filename, PATHINFO_FILENAME);
  $extension = pathinfo($filename, PATHINFO_EXTENSION);
② $basename  = preg_replace('/[^A-z0-9]/', '-', $basename);
③ $filepath  = 'uploads/' . $basename . '.' . $extension;
```

Limpiando un nombre de archivo y archivos duplicados

Si se llama a `move_uploaded_file()` y existe un archivo con el mismo nombre, el archivo antiguo se sustituye por el nuevo. Para evitar esto, cada archivo necesita un nombre único:

4. Pon un contador a **0** y guárdalo en una variable llamada `i`.
5. En la condición de un bucle while, utiliza la función `file_exists()` de PHP para comprobar si ya existe un archivo con el mismo nombre.
6. Si lo hace, suma 1 al valor almacenado en el contador.

Limpiando un nombre de archivo y archivos duplicados

7. Actualiza el nombre del fichero, añadiendo el valor en el contador después del nombre base, antes de la extensión. Por ejemplo, si *upload.jpg* existe, llama al archivo *upload1.jpg*.

La condición del bucle se ejecuta de nuevo para comprobar si existe el nuevo nombre de fichero. El bucle repite los pasos 5-7 hasta que tenga un único nombre de fichero.

```
④ $i = 1;
⑤ while (file_exists('uploads/' . $filename)) {
⑥     $i = $i + 1;
⑦     $filename = $basename . $i . '.' .  $extension;
  }
```

6. Validando el tamaño y tipo de un fichero

Validando el tamaño y tipo de un fichero

Para asegurarse de que un sitio puede trabajar con un archivo cargado, antes de moverlo, hay que comprobar:

1. El **archivo no es demasiado grande** (los archivos grandes tardan más en descargarse o procesarse).
2. El **sitio puede trabajar con el tipo de medio y la extensión** del archivo.

Se puede establecer un tamaño máximo de subida de archivos en *php.ini* o *.htaccess*, o crear código de validación para restringir los tamaños en las páginas que aceptan subidas.

Validando el tamaño y tipo de un fichero

Para ver si un archivo es mayor que el **tamaño máximo de subida** establecido en php.ini o .htaccess mira en el array `$_FILES`. Si lo es, la clave de error para ese archivo tendrá un valor de `1`.

También se puede **comprobar el tamaño de un archivo** en el array `$_FILES`; la clave de tamaño del archivo contiene el tamaño en bytes. Los dos operadores ternarios siguientes se utilizan para realizar ambas comprobaciones. La condición del:

- Primer operador ternario comprueba si el código de error es 1.
- Segundo, comprueba si el tamaño es superior a 5mb.

```
$error = ($_FILES['image']['error'] === 1)     ? 'Too large' : '';
$error = ($_FILES['image']['size'] <= 5242880) ? '' : 'Too large';
```

Validando el tamaño y tipo de un fichero

Validar el **tipo de medio** y la **extensión** de un archivo ayuda a garantizar que un sitio pueda manejarlo con seguridad. En el siguiente ejemplo:

1. `$allowed_types` es un array de tipos de medios permitidos.
2. La función `mime_content_type()` de PHP intenta detectar el tipo de medio del archivo y lo almacena en `$type` .
3. iii. La función `in_array()` de PHP comprueba si el tipo de medio de este archivo está en el array de tipos de medios permitidos.

Validando el tamaño y tipo de un fichero

4. `$allowed_exts` almacena un array de extensiones permitidas.
5. El nombre del archivo se convierte a minúsculas y se almacena en `$filename`.
6. La extensión del archivo se recoge y se almacena en `$ext`.
7. La función `in_array()` de PHP comprueba si esta extensión de archivo está en el array de extensiones permitidas.

```
① $allowed_types = ['image/jpeg', 'image/png', 'image/gif',];
② $type  = mime_content_type($_FILES['image']['tmp_name']);
③ $error = in_array($type, $allowed_types) ? '' : 'Wrong file type ';
④ $allowed_exts = ['jpeg', 'jpg', 'png', 'gif',];
⑤ $filename = strtolower($_FILES['image']['name']);
⑥ $ext      = pathinfo($filename, PATHINFO_EXTENSION));
⑦ $error   .= in_array($ext, $allowed_exts) ? '' : 'Wrong extension ';
```

Ejemplo: Validando subidas de ficheros

Este ejemplo reúne el código para cargar, validar y guardar un archivo.

1. Se crean seis variables para contener el:

- Resultado de si se ha subido o no el archivo
- Mensaje de éxito / fracaso que ve el usuario
- Errores si hay problemas con la imagen
- Ruta a la carpeta que almacena los archivos subidos
- Tamaño máximo del archivo en bytes
- Tipos de medios permitidos
- Extensiones de archivo permitidas

Ejemplo: Validando subidas de ficheros

2. Se define una función llamada `create_filename()`.
   Utiliza el código que hemos estudiado en esta unidad previamente para limpiar el nombre del fichero y asegurarse de que es único, y devuelve el nuevo nombre. Sus dos parámetros son:
   - Nombre del fichero
   - Ruta relativa a la carpeta donde se almacenará
3. Una sentencia if comprueba si el formulario ha sido enviado.
4. Se utiliza un operador ternario para comprobar si hubo un error al subir esta imagen porque era mayor que el límite de tamaño establecido en php.ini o .htaccess. Si es así, se almacena un mensaje de error en `$error`.

Ejemplo: Validando subidas de ficheros

5. Otra sentencia if comprueba si el archivo se ha subido sin errores.
6. Se valida el tamaño del archivo. Si es menor o igual que el tamaño máximo almacenado en `$max_size` en el Paso 1, `$error` almacena una cadena en blanco; si es mayor que el tamaño máximo permitido, `$error` contiene el mensaje ' too big ' (demasiado grande).
7. La función incorporada de PHP `mime_content_type()` obtiene el tipo de medio del archivo y lo almacena en `$type` .
8. La función `in_array()` de PHP comprueba si el tipo de medio almacenado en `$type` está en el array `$allowed_types` . Si lo está, se añade una cadena en blanco a la variable `$error` . Si no lo está, se añade un mensaje de error a `$error` .

Ejemplo: Validando subidas de ficheros

9. La función `pathinfo()` de PHP obtiene la extensión del archivo de la imagen cargada. Esta función es llamada dentro de la función `strtolower()` de PHP para asegurar que la extensión está en minúsculas. Luego se almacena en `$ext`.
10. La función `in_array()` de PHP se utiliza para comprobar si la extensión del archivo está permitida. Si lo está, se añade una cadena en blanco a la variable `$error`. Si no lo es, se añade un mensaje para indicar que se trata de una extensión incorrecta.
11. La condición de una sentencia if comprueba si `$error` contiene un valor que no se trate como verdadero. Una cadena en blanco se trata como falso (no hay errores).

Ejemplo: Validando subidas de ficheros

12. Si no hay errores, se llama a la función `create_filename()` (del Paso 2) para asegurar que el nombre del archivo es limpio y único.
13. `$destination` contiene la ruta para guardar el nuevo archivo.
14. Se llama a la función de PHP `move_uploaded_file()` para mover el archivo desde su ubicación temporal a la carpeta uploads. Devuelve true si funciona; false si no. El resultado se almacena en una variable llamada `$moved` .

Ejemplo: Validando subidas de ficheros

15. Si la variable `$moved` tiene un valor de true, la imagen se ha subido, ha pasado las comprobaciones y se ha guardado, por lo que la variable `$message` almacena una etiqueta HTML `<img>` que mostrará la imagen.
16. En caso contrario, se almacena un mensaje de error en `$message`.
17. El valor almacenado en la variable `$message` se muestra antes del formulario de subida.

Ejemplo: Validando subidas de ficheros

```php
<?php
$moved         = false;                                    // Initialize
$message       = '';                                       // Initialize
$error         = '';                                       // Initialize
$upload_path   = 'uploads/';                               // Upload path
$max_size      = 5242880;                                  // Max file size (in bytes)
$allowed_types = ['image/jpeg', 'image/png', 'image/gif',]; // Allowed file types
$allowed_exts  = ['jpeg', 'jpg', 'png', 'gif',];           // Allowed file extensions

function create_filename($filename, $upload_path)          // Function to make filename
{
    $basename   = pathinfo($filename, PATHINFO_FILENAME);  // Get basename
    $extension  = pathinfo($filename, PATHINFO_EXTENSION); // Get extension
    $basename   = preg_replace('/[^A-z0-9]/', '-', $basename); // Clean basename
    $i          = 0;                                       // Counter
    while (file_exists($upload_path . $filename)) {        // If file exists
        $i        = $i + 1;                                // Update counter
        $filename = $basename . $i . '.' . $extension;     // New filepath
    }
    return $filename;                                      // Return filename
}
```

Ejemplo: Validando subidas de ficheros

```
③ if ($_SERVER['REQUEST_METHOD'] == 'POST') {                    // If form submitted
④     $error = ($_FILES['image']['error'] === 1) ? 'too big ' : '';  // Check size error

⑤     if ($_FILES['image']['error'] == 0) {                      // If no upload errors
⑥         $error .= ($_FILES['image']['size'] <= $max_size) ? '' : 'too big '; // Check size
          // Check the media type is in the $allowed_types array
⑦         $type   = mime_content_type($_FILES['image']['tmp_name']);
⑧         $error .= in_array($type, $allowed_types) ? '' : 'wrong type ';
          // Check the file extension is in the $allowed_exts array
⑨         $ext    = strtolower(pathinfo($_FILES['image']['name'], PATHINFO_EXTENSION));
⑩         $error .= in_array($ext, $allowed_exts) ? '' : 'wrong file extension ';
          // If there are no errors, create the new filepath and try to move the file
⑪         if (!$error) {
⑫           $filename    = create_filename($_FILES['image']['name'], $upload_path);
⑬           $destination = $upload_path . $filename;
⑭           $moved       = move_uploaded_file($_FILES['image']['tmp_name'], $destination);
          }
      }
```

Ejemplo: Validando subidas de ficheros

```
⑮      if ($moved === true) {                                    // If it moved
           $message = 'Uploaded:<br><img src="' . $destination . '">';   // Show image
⑯      } else {                                                  // Otherwise
           $message = '<b>Could not upload file:</b> ' . $error;         // Show errors
       }
   }
⑰  ?> ... <?= $message ?> <!-- Show form -->
```

Actividad: Validando subidas de ficheros

Modifica la actividad anterior (*Actividad: Moviendo un archivo subido*) para implementar las técnicas de:

- Limpieza de nombres de ficheros
- Evitar sobreescribir ficheros ya existentes
- Validación de tipos de medios
- Validación del tamaño de ficheros

Implementa las comprobaciones necesarias para que el fichero que el usuario sube como imagen de perfil no contenga caracteres raros en su nombre, no sobreescriba otra imagen de perfil ya subida previamente, sea un fichero en formato .png o .jpeg y contenga la extensión apropiada.