**The Form**

<form enctype="multipart/form-data" action="script.php" method="post">

<input type="hidden" name="MAX\_FILE\_SIZE" value="30000" />

File <input type="file" name="upload" />

The enctype part of the initial form tag indicates that the form should be able to handle multiple types of data, including files. If you want to accept file uploads, you must include this enctype!

The form must use the POST method.

The MAX\_FILE\_SIZE hidden input is a form restriction on how large the chosen file can be, in bytes, and must come before the file input.

While it’s easy for a user to circumvent this restriction, it should still be used.

Finally, the file input type will create the proper button in the form.

(Try a few browsers.)

Upon form submission, the uploaded file can be accessed using the $\_FILES superglobal.

Once the file has been received by the PHP script, the move\_uploaded\_file( ) function can transfer it from the temporary directory to its permanent location.

**Show Image:**

The javascript first

var url = "show\_image.php?image=" + image;

Note that popup is global.

**Create JavaScript with PHP**

**Images.php**

5. Start the PHP code and create an array of images by referring to the uploads directory:

<?php # Script 11.4 - images.php

$dir = '../uploads';

$files = scandir($dir);

This script will automatically list and link all of the images stored in the uploads folder.

6. Begin looping through the $files array:

foreach ($files as $image) {

 if (substr($image, 0, 1) != '.') {

This loop will go through every image in the array and create a list item for it.

Within the loop, there is one conditional that checks if the first character in the file’s name is a period. On non-Windows systems, hidden files start with a period, the current directory is referred to using just a single period, and two periods refers to the parent directory.

7. Get the image information and encode its name:

$image\_size = getimagesize ("$dir/$image");

$image\_name = urlencode($image);

8. Print the list item:

echo "<li><a href=\"javascript: create\_window('$image\_name', $image\_size[0],$image\_size[1])\"> $image</a>\n";
.

See how different browsers handle the sizing.

Some versions of Windows create a Thumbs.db file in a folder of images.

if ( (substr($image, 0, 1) != '.') && ($image != 'Thumbs.db') )

Most everything Web developers do with JavaScript (for example, resize or move the browser window) cannot be done using the server-side PHP.

There is a little overlap between the PHP and JavaScript. Both can set and read cookies, create HTML, and do some browser detection.

A proxy script is able to provide access to content on the server that would otherwise be unavailable.

To use the header() function

1. Begin a new PHP document in your text editor or IDE, to be named show\_image.php

<?php # Script 11.5 - show\_image.php

$name = FALSE;

Because this script will use the header( ) function, nothing—absolutely nothing—can be sent to the Web browser. No HTML, not even a blank line, tab, or space before the opening PHP tag. The $name variable will be used as a flag, indicating if all of the validation routines have been passed.

2. Check for an image name:

if (isset($\_GET['image'])) {

The script needs to receive a valid image name in the URL. This should be appended to the URL in
the JavaScript function that calls this page.

3. Validate the image’s extension:

$ext = strtolower ( substr ($\_GET['image'], -4));

if (($ext = = '.jpg') OR ($ext = = 'jpeg') OR ($ext = = '.png')) {

The next check is that the file to be sent to the Web browser has a .jpeg, .jpg, or .png extension.

This way the script won’t try to send something bad to the user. For example, if a malicious user changed the address in the pop-up window from http://www.example.com/show\_image.php?image=image.png to http://www.example.com/show\_image.php?image=../../../path/to/ something/important, this conditional would catch, and prevent, that hack.

To validate the extension, the substr( ) function returns the last four characters from the image’s name (the -4 accomplishes this). The extension is also run through the strtolower( ) function so that .PNG and .png are treated the same.

Then a conditional checks to see if $ext is equal to any of the three allowed values.

4. Check that the image is a file on the server:

$image = "../uploads/{$\_GET['image']}";

if (file\_exists ($image) && (is\_file($image))) {

Before attempting to send the image to the Web browser, make sure that it exists and that it is a file (as opposed to a directory). As a security measure, the image’s full path is defined as a combination of ../uploads and the received image name.

5. Once the image has passed all of these tests, the $name variable is assigned the value of the image.

$name = $\_GET['image'];

6. Complete the conditionals begun in Steps 2, 3, and 4:

} // End of file\_exists( ) IF.

 } // End of $ext IF.

} // End of isset($\_GET['image']) IF.

There are no else clauses for any of these three conditions. If all three conditions aren’t TRUE, then the flag variable $name will still have a FALSE value.

7. If no valid image was received by this page, use a default image: if (!$name) {   $image = 'images/unavailable.png';   $name = 'unavailable.png'; } If the image doesn’t exist, if it isn’t a file, or if it doesn’t have the proper extension, then the $name variable will still have a value of FALSE. In such cases, a default image will be used instead.

I can’t get it to this place.

8. Retrieve the image’s information:

$info = getimagesize($image);

$fs = filesize($image);

To send a file to the Web browser, the script needs to know the file’s MIME type and size.

An image file’s type can be found using getimagesize( ).

The file’s size, in bytes, is found using filesize( ).

Because the $image variable represents either ../uploads/{$\_GET['image']} or images/unavailable.png, these lines will work on both the correct and the unavailable image.

9. Send the file:

header ("Content-Type: {$info['mime']}\n");

header ("Content-Disposition: inline; filename=\"$name\"\n");

header ("Content-Length: $fs\n");
.

These header( ) calls will send the file data to the Web browser.

The first line uses the image’s MIME type for the value of the Content-Type header. The second line tells the browser the name of the file and that it should be displayed in the browser (inline).

The last header( ) function indicates how much data is to be expected.

The file data itself is sent using the readfile( ) function, which reads in a file and immediately sends the content to the Web browser.

10. Notice that this page contains no HTML. It only sends an image file to the Web browser.

Also note that I omitted the terminating PHP tag. This is acceptable, and in certain situations like this, preferred. If you included the closing PHP tag, and you inadvertently had an extra space or blank line after that tag, the browser could have problems displaying the image (because the browser will have
received the image data of X length, matching the Content-Length header, plus a bit of extra data).

**Tip I cannot stress strongly enough that nothing can be sent to the Web browser before using the header( ) function. Even an included file that has a blank line after the closing PHP tag will make the header( ) function unusable.**

To avoid problems when using header( ), you can call the headers\_sent( ) function first. It returns a Boolean value indicating if something has already been sent to the Web browser:

if (!headers\_sent( )) {

// Use the header( ) function.

}

else {

// Do something else.

}