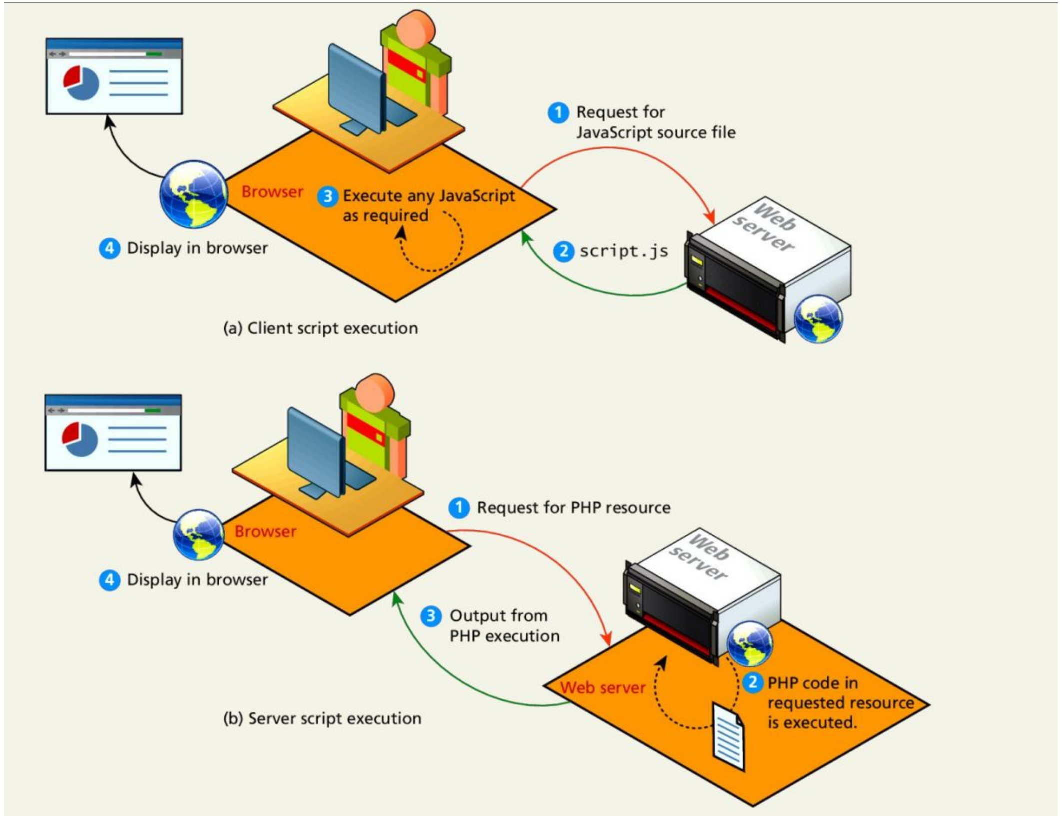
Internet Computing

Introduction to PHP

Origins of PHP

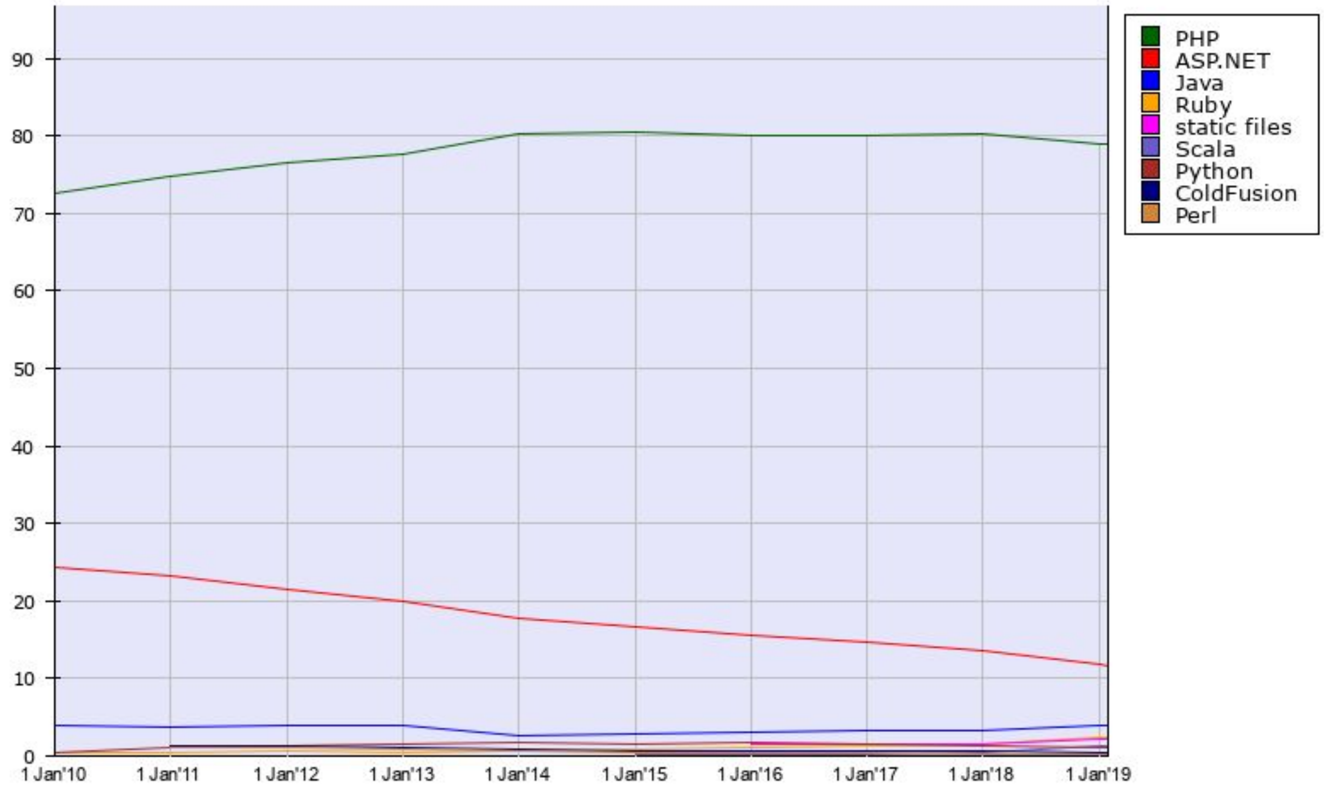
- Developed by Rasmus Lerdorf - 1994
    - To allow him to track visitors to his Web site
- PHP was originally an acronym for Personal Home Page, but later it became
- PHP: Hypertext Preprocessor
    - that means we're processing the hypertext or the HTLM before providing that to the user's browser.
- PHP is used for form handling, file processing, and database access

Comparing client and server scripts


1 Request for JavaScript source file
Browser
3 Execute any JavaScript as required
4 Display in browser
2 script.js
Web server
(a) Client script execution
1 Request for PHP resource
Browser
4 Display in browser
Web server
3 Output from PHP execution
Web server
2 PHP code in requested resource is executed.
(b) Server script execution

Comparing Server-side technologies

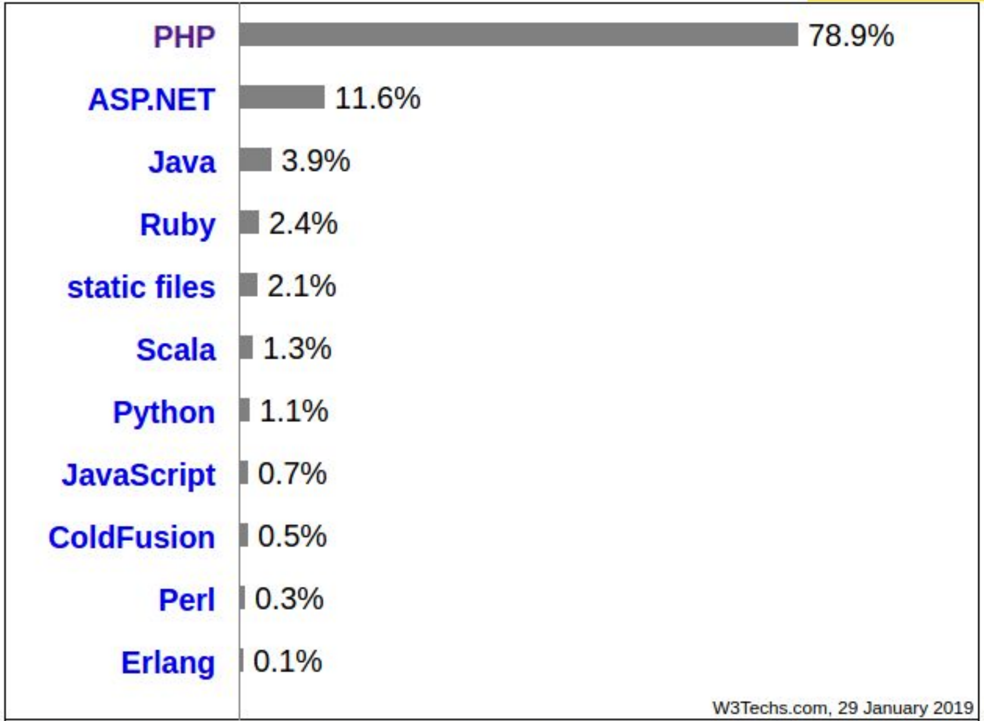
https://w3techs.com/



Usage of server-side programming languages for websites, 29 Jan 2019, W3Techs.com

Comparing Server-side technologies

https://w3techs.com/

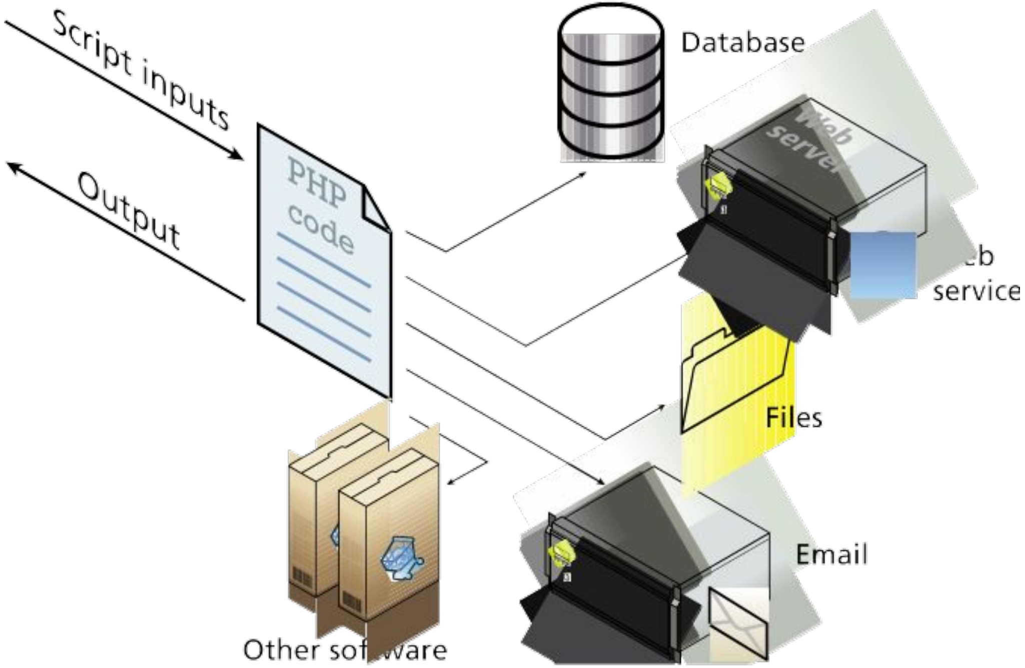


W3Techs.com, 29 January 2019

Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

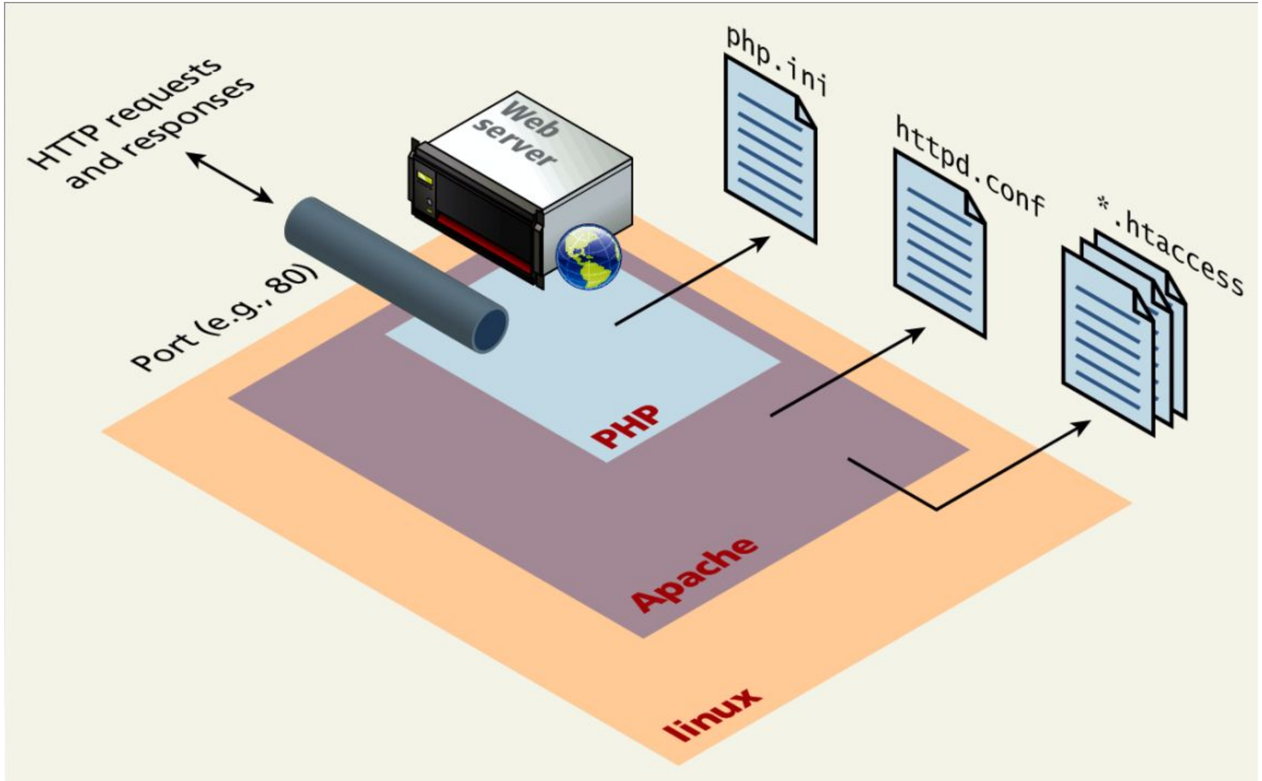
Server-side development

The web server act as the intermediary that interprets HTTP requests that arrive through a network port and decides how to handle the request, which often requires working in conjunction with PHP.



LAMP Stack

- Linux Operating System
- Apache web server
- MySQL DBMS
- PHP scripting language



XAMPP

- Stands for "XAMPP Apache + MariaDB + PHP + Perl”
- PHP file should be located on a web server.
- Two run a php file:
  - Put it on hopper server
  - To work on your computer locally
    - Download and install XAMPP from the following link:
    - https://www.apachefriends.org/index.html
    - https://www.apachefriends.org/faq_linux.html

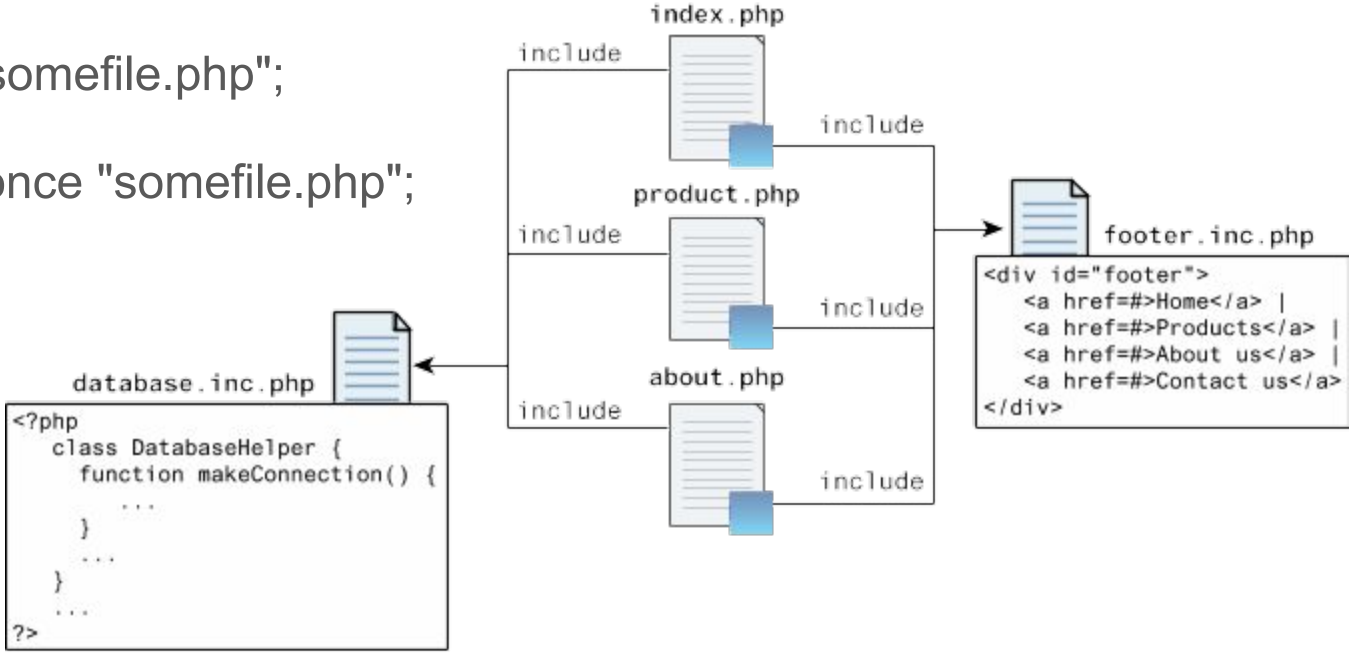
General Syntactic Characteristics

- PHP code can be linked to an HTML document in two ways:
  - internally
    - <?php tag and a matching closing ?>
    - Inside is code to execute, outside is HTML to echo directly
  - Externally: include ("somefile.php")
    - the file can have both PHP and HTML
    - If the file has PHP, the PHP must be in <?php .. ?>, even if the include is already in <?php .. ?>
    - include runs code from another php file. It works the same as if you just copied the code from the other file and pasted it where the include was.

Include files

include "somefile.php";

include_once "somefile.php";



```
<div id="footer">
    <a href=#>Home</a> |
    <a href=#>Products</a> |
    <a href=#>About us</a> |
    <a href=#>Contact us</a>
</div>
```

```
<?php
   class DatabaseHelper {
     function makeConnection() {
         ...
     }
     ...
   }
   ...
?>
```

General Syntactic Characteristics

- Comments - three different kinds (Java and C)
  - // ...
  - # ...
  - /* ... */
- Compound statements are formed with braces
- Compound statements cannot be blocks

Variables

- Every variable name begin with a $ on both declaration and usage
- Names are case-sensitive; separate multiple words with _ (as in $user_name)
- Implicitly declared by assignment
    - There are no type declarations: a "loosely-typed" language
- An unassigned (unbound) variable has the value, NULL
- The `unset` function sets a variable to NULL
- The `isset` function is used to determine whether a variable is NULL
    - error_reporting(15); - prevents PHP from using unbound variables
- PHP has many predefined variables, including the environment variables of the host operating system
    - You can get a list of the predefined variables by calling `phpinfo()` in a script
-

Arithmetic Operations

+ - * / %

. ++ --

= += -= *= /= %= .=

Many operators auto-convert types: 5 + "7" is 12

Data Types

- There are eight primitive types:
    - Four scalar types: `Boolean, integer, double, and string`
        - `Integer & double` are like those of other languages
    - Two compound types: `array` and `object`
    - Two special types: `resource` and `NULL`

Scalar Type conversions

- Determining type of the value of a avariable
  - Test what type a variable is with `is_type` functions
  - `is_string,is_int, is_integer, is_long, is_double, is_float, is_real, is_bool`
  - **`gettype`** function returns a variable's type as a string
    - not often needed
- Implicit type conversion
  - PHP converts between types automatically in many cases:
- Explicit type conversion
  - Type-cast with (type):variable_name: $sum = 4.77; (int) $sum;
  - Use of the intval, doubleval, strval
  - Using settype function: settype($sum, “integer”);

Example:Implicit type conversion

string to integer auto-conversion
("1" + 1 == 2) is true
5 + "2 beautiful birds" produces 7

integer to string auto conversion
5 . "2 beautiful birds" produces "52 beautiful birds"

int to float auto-conversion on /
(3 / 2 == 1.5) is true

Data Types: strings

- Can be specified with "" or ''
  - 0-based indexing using [] bracket notation
  - String Concatenation
    - Important note! String concatenation is . (period) not +
      Characters are single bytes

0-based indexing using [] bracket notation

```
$favorite_food = "Pizza";
print $favorite_food[2]; #prints z
```

Data Types: strings

- String literals use single or double quotes
    - Single-quoted string literals
        - Embedded variables are NOT interpolated
        - Embedded escape sequences are NOT recognized
    - Double-quoted string literals
        - Embedded variables ARE interpolated
            - Variables that appear inside them will have their values inserted into the string
        - If there is a variable name in a double-quoted string but you don’t want it interpolated, it must be backslashed
        - Embedded escape sequences ARE recognized

```
Escaping Strings
\n Line feed
\t Horizontal tab
\\ Backslash
\$ Dollar sign
\” Double quote
```

Data Types: strings: Examples

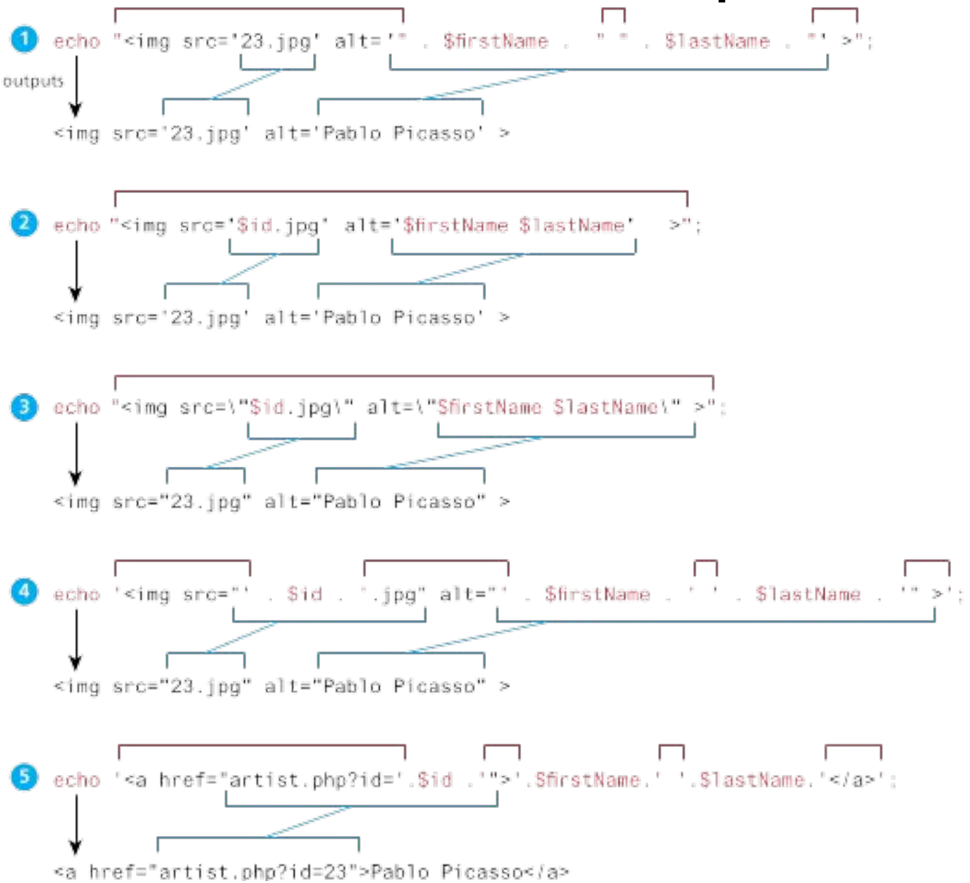
```
$sum = 16;
$age=20;
print "The value of sum is: \n" . $sum ;  #The value of sum is: 16
NOT RECOMMENDED
print "The value of sum is: $sum \n"; # The value of sum is: 16
print 'The value of sum is: $sum \n’; # The value of sum is: $sum
print "Today is your $ageth birthday.\n"; # ageth not found..error
print "Today is your {$age}th birthday.\n"; Today is your 20th
birthday
```

Data Types: Strings : functions

Name	Java Equivalent
strlen	length
strpos	indexOf
substr	substring
strtolower, strtoupper	toLowerCase, toUpperCase
trim	trim
explode, implode	split, join

Data Types: Strings: Concatenation Examples

DEMO: concatenation.php



```
1 echo "<img src='23.jpg' alt='" . $firstName . " " . $lastName . "' >";
  <img src='23.jpg' alt='Pablo Picasso' >

2 echo "<img src='$id.jpg' alt='$firstName $lastName'  >";
  <img src='23.jpg' alt='Pablo Picasso' >

3 echo "<img src=\"$id.jpg\" alt=\"$firstName $lastName\" >";
  <img src="23.jpg" alt="Pablo Picasso" >

4 echo '<img src="' . $id . '.jpg" alt="' . $firstName . ' ' . $lastName . '" >';
  <img src="23.jpg" alt="Pablo Picasso" >

5 echo '<a href="artist.php?id=' .$id .'">'.$firstName.' '.$lastName.'</a>';
  <a href="artist.php?id=23">Pablo Picasso</a>
```

Data Types: Boolean

- Values are true and false
- Values are case insensitive: `false, False, TRUE, true`
- The following values are `FALSE`; others are true
  - `0 , 0.0, "" , "0" , and NULL (` include unset variable)
  - Arrays with `0` elements
- Can cast to boolean using `(bool)`
- `FALSE` prints as an empty string (no output); `TRUE` prints as a 1
- Boolean operators: `and, or, xor, !, &&, ||`
  - `&&` and `||` work the same way as in Java or JavaScript
  - The precedence of `and` and `or` is lower than that of `&&` and `||`.

Data Types: `NULL`

- A variable is `NULL` if
  - It has been assigned the constant `NULL`
  - It has not been set to any value yet (undefined)
  - It has been deleted using the unset function
- `NULL` prints as an empty string (no output)
- Use the `isset()` function to test if a variable is set and not `NULL`
- If a variable is `unset()`, `isset()` will return `FALSE`

Data Types: Constant

- Use `define()`
  - uppercase for constants is a programming convention
  - Then use the word without quotes (or $)

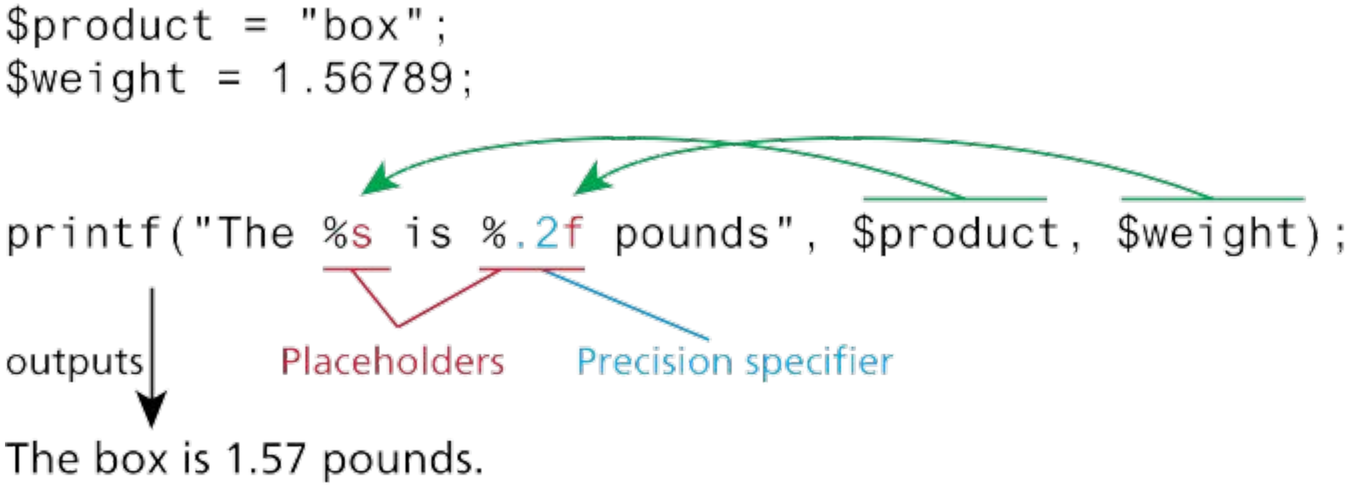
```
define("DATABASE_LOCAL", "localhost");
echo DATABASE_LOCAL;
```

Output

- Output from a PHP script is HTML that is sent to the browser
- There are multiple ways to produce output:
    - `print` and `printf` and `echo`
    - `print/echo` takes a string, but will coerce other values to strings
    - `printf` is exactly as in C
- DEMO: today.php

output

`printf`



```
$product = "box";
$weight = 1.56789;

printf("The %s is %.2f pounds", $product, $weight);
```

```
The box is 1.57 pounds.
```

Control Statements

- Control Expressions
  - Relational operators - same as JavaScript, (including `===` and `!==`)
  - Boolean operators - same as C (two sets, && and and, etc.)
  - Selection statements
    - `if, if-else`
      - Can `use elseif` instead of `else if`
    - `switch` - as in C
      - The `switch` expression type must be `integer`, `double`, or `string`
    - `while` - just like C
    - `do-while` - just like C
      - `break` and `continue` keywords also behave as in Java
    - `for` - just like C
    - `foreach`

Arrays

- All arrays in PHP are generally referred to as **Associative Arrays**
  - Associative arrays are arrays that use named keys that you assign to them
- The array in PHP replaces many other data structures in Java
  e.g. list, stack, queue, set, map, ...

Arrays

- Creating Arrays
- Accessing Elements
- Output arrays: `print_r`

```
Creating Arrays format
$name = array(); # create
$name = array(value0, value1, ..., valueN);
$name = [value0, value1, value2, …,
valueN];

Access Array's elements format
$name[index] # get element value
$name[index] = value; # set element value
```

Arrays

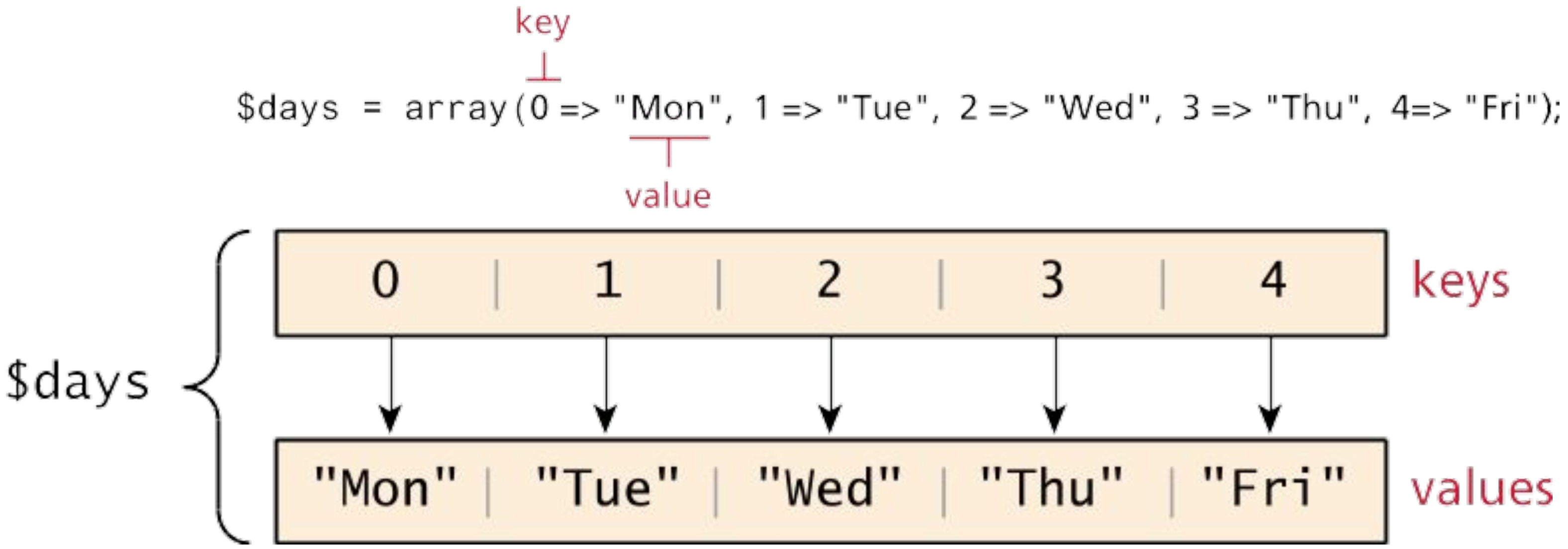
- Creating an Array

```
Example: four different ways to create an array

$days = array(); #creates an empty array
$days = array("Mon","Tue","Wed","Thu","Fri");
$days = ["Mon","Tue","Wed","Thu","Fri"]; //
$days=array(0 => "Mon", 1 => "Tue", 2 => "Wed", 3 =>
"Thu", 4 => "Fri");
```

    - $days[5]= "Sat";
    - $days[]= "Sun";

Arrays

- Associative arrays



```
$days = array(0 => "Mon", 1 => "Tue", 2 => "Wed", 3 => "Thu", 4=> "Fri");
```

keys	0	1	2	3	4
values	"Mon"	"Tue"	"Wed"	"Thu"	"Fri"

Arrays

- Adding and Deleting Elements
  - An element can be added to an array simply by using a key/index that hasn’t been used
    - $days[5]= "Sat";
  - As an alternative to specifying the index, a new element can be added to the end of any array using empty square brackets after the array name
    - $days[]= "Sun";
- Delete with unset()

```
$a = array(); # empty array (length 0)
$a[0] = 23; # stores 23 at index 0 (length
1)
$a2 = array("some", "strings", "in", "an",
12, True); #don’t need to have the same
values
$a2[] = "Ooh!"; # add string to end (at
index 5)

$age = array("Spot"=>16, "Whitney"=>16,
"Jack"=>12); # create
$age["Mowgli"] = 1;
$age["Whitney"] = 17; # stores 17 at the
location where "Whitney" is stored
```

Array functions

- Count
  - number of elements in the array
- print_r
  - print array's contents
- array_pop, array_push, array_shift, array_unshift
  - using an array as a stack/queue
- in_array, array_search, array_reverse sort, rsort, shuffle
  - searching and reordering
- array_fill, array_merge, array_intersect, array_diff, array_slice, range
  - creating, filling, filtering
- array_sum, array_product, array_unique, array_filter, array_reduce
  - processing elements

Arrays: Iterating through an Array

- for loop, while loop, do-while loop
- foreach
  - A convenient way to loop over each element of an array without indices

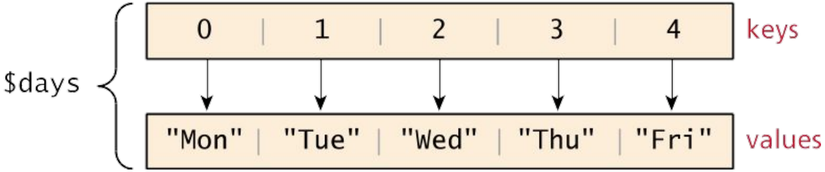
```
foreach ($array as variableName)
{
  ...
}
```

```
<?php
$forecast= array("Mon" => 40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" =>
37);

// foreach: iterating through the values
foreach ($forecast as $value) {
        echo $value . "<br>";
}
// foreach: iterating through the values AND the keys
foreach ($forecast as $key => $value) {
        echo "day" . $key . "=" . $value;
}
?>
```

Array functions: Examples

```
$provinces = array("ON", "QC", "MB", "BC", "NL", "PE", "AB");
for ($i = 0; i < count($provinces); $i++) {
  $provinces[$i}] = strtolower($provinces[$i]);
  #$provinces: Array ( [0] => on [1] => qc [2] => mb [3] => bc [4] => nl [5] => pe [6] => ab )
}
$conner = array_shift($provinces);      #$provinces: Array ( [0] => qc [1] => mb [2] => bc [3] => nl [4] => pe [5] => ab )
array_pop($provinces);                  #$provinces: Array ( [0] => qc [1] => mb [2] => bc [3] => nl [4] => pe )
array_push($provinces);                 #$provinces: Array ( [0] => qc [1] => mb [2] => bc [3] => nl [4] => pe [5] => nt )
$rarray = array_reverse($provinces);    #$rarray: Array ( [0] => nt [1] => pe [2] => nl [3] => bc [4] => mb [5] => qc )
sort($provinces);                       #$provinces: Array ( [0] => bc [1] => mb [2] => nl [3] => nt [4] => pe [5] => qc )
$ks = array_slice($tas, 2, 3);          #$ks = Array ( [0] => nl [1] => nt [2] => pe )
```

Arrays



```
$days = array("Mon","Tue","Wed","Thu","Fri");
```

```
sort($days);
```

As the values are all strings, the resulting array would be:

```
Array ([0] => Fri [1] => Mon [2] => Sat [3] => Sun [4] => Thu
[5] => Tue [6] => Wed)
```

```
asort($days);
```

The resulting array in this case keeps associations so is:

```
Array ([4] => Fri [0] => Mon [5] => Sat [6] => Sun [3] => Thu
[1] => Tue [2] => Wed)
```

More examples: `Look at arrays.php`

The functions

- parameter types and return types are not written
    - a function with no return statements is implicitly "void"
- Calling functions
    - If the wrong number of parameters are passed, it's an error

```
#function definition
function name(parameterName, ...,
parameterName) {
  statements;
}

function bmi($weight, $height) {
  $result = 703 * $weight /
$height / $height;
  return $result;
}

#calling a function
name(expression, ..., expression);
$w = 163; # pounds
$h = 70; # inches
$my_bmi = bmi($w, $h);
```

The functions : parameters

- Default values
    - In PHP you can set parameter default values for any parameter in a function. However, once you start having default values, all subsequent parameters must also have defaults.
    - If no value is passed, the default will be used (defaults must come last)

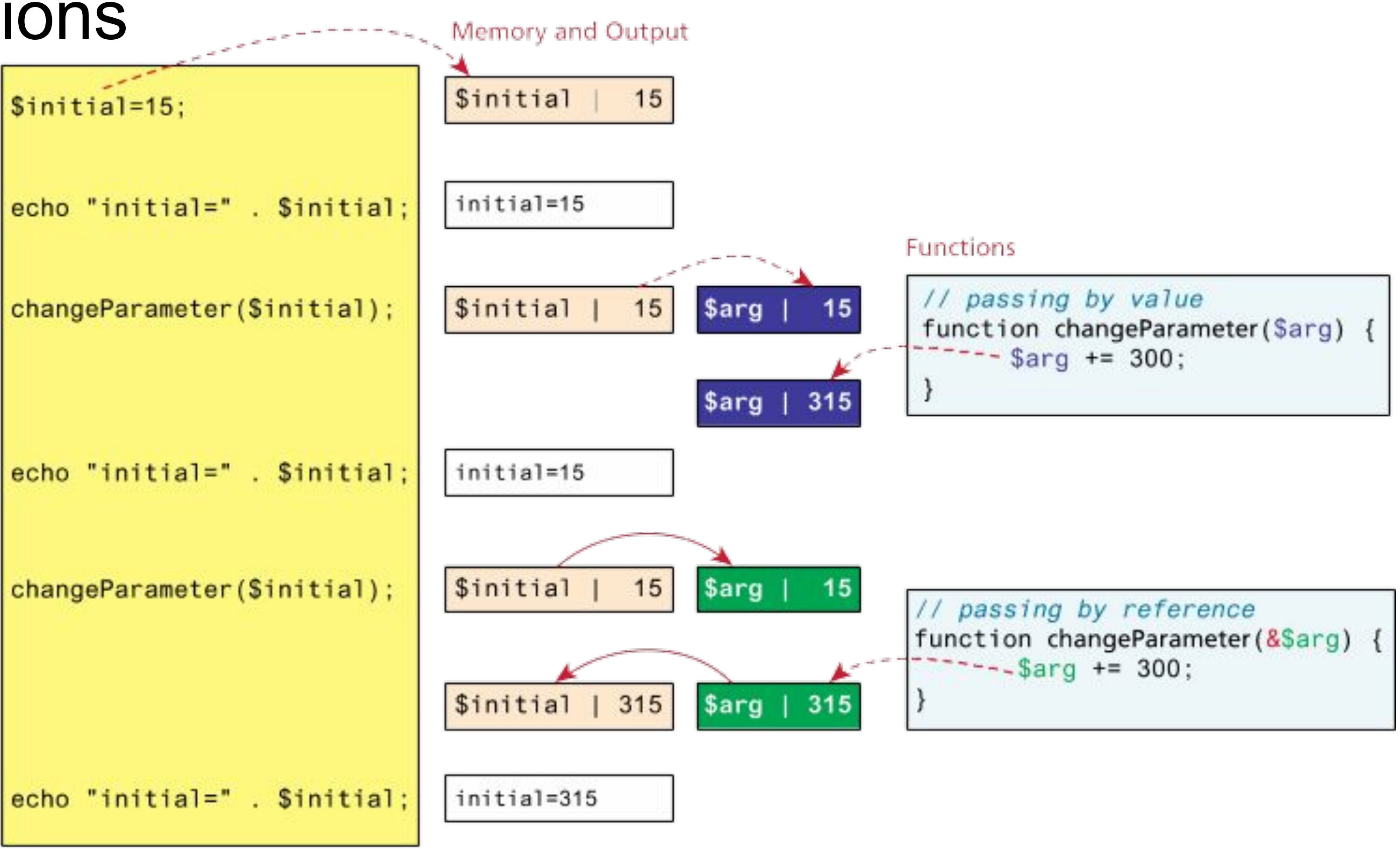
```
function getNiceTime($showSeconds=true) {
    if ($showSeconds==true)
        return date("H:i:s");
    else
        return date("H:i");
}
```

The functions : parameters passing

- Pass-by-value
  - Default behavior
  - By default, arguments passed to functions are passed by value in PHP.
- Pass-by-reference
  - Add an ampersand (&) to the beginning of the name of the formal parameter that you want to be passed by reference
  - Add an ampersand to the argument in the function call

The functions

Parameters Passing



```
$initial=15;

echo "initial=" . $initial;

changeParameter($initial);

echo "initial=" . $initial;

changeParameter($initial);

echo "initial=" . $initial;
```

Memory and Output

```
initial=15
```

```
initial=15
```

```
initial=315
```

Functions

```
// passing by value
function changeParameter($arg) {
    $arg += 300;
}
```

```
// passing by reference
function changeParameter(&$arg) {
    $arg += 300;
}
```

Challenge: Expression in PHP

Test the following expressions in PHP:

```
5 + 19 / 4 + 3 * -2
1 + 1 . "(1 + 1)" . 1 + 1
13 / 2 - 35 / 5 / 2.0 + (15 / 10.0)
11 < 2 + 4 || !(5 / 2 == 2)
20 % 6 + 6 % 20 + 6 % 6
```

Variable Scope: global

- Variables declared in a function are local to that function
- variables defined in the main script have global scope
  - these global variables are not by default, available within functions.
- To access variables with global scope within a function use the global keyword,
  - don't abuse this; mostly you should use parameters

```
$big_sum = 0;  # global

function summer ($list) {
  global $big_sum;
  foreach ($list as $value)
    $sum += $value;
  big_sum += $sum;
  return $sum;
 }

$ans1 = summer($list1);
$ans2 = summer($list2);
```

PHP Error Mode

```
error_reporting(E_ALL);
```

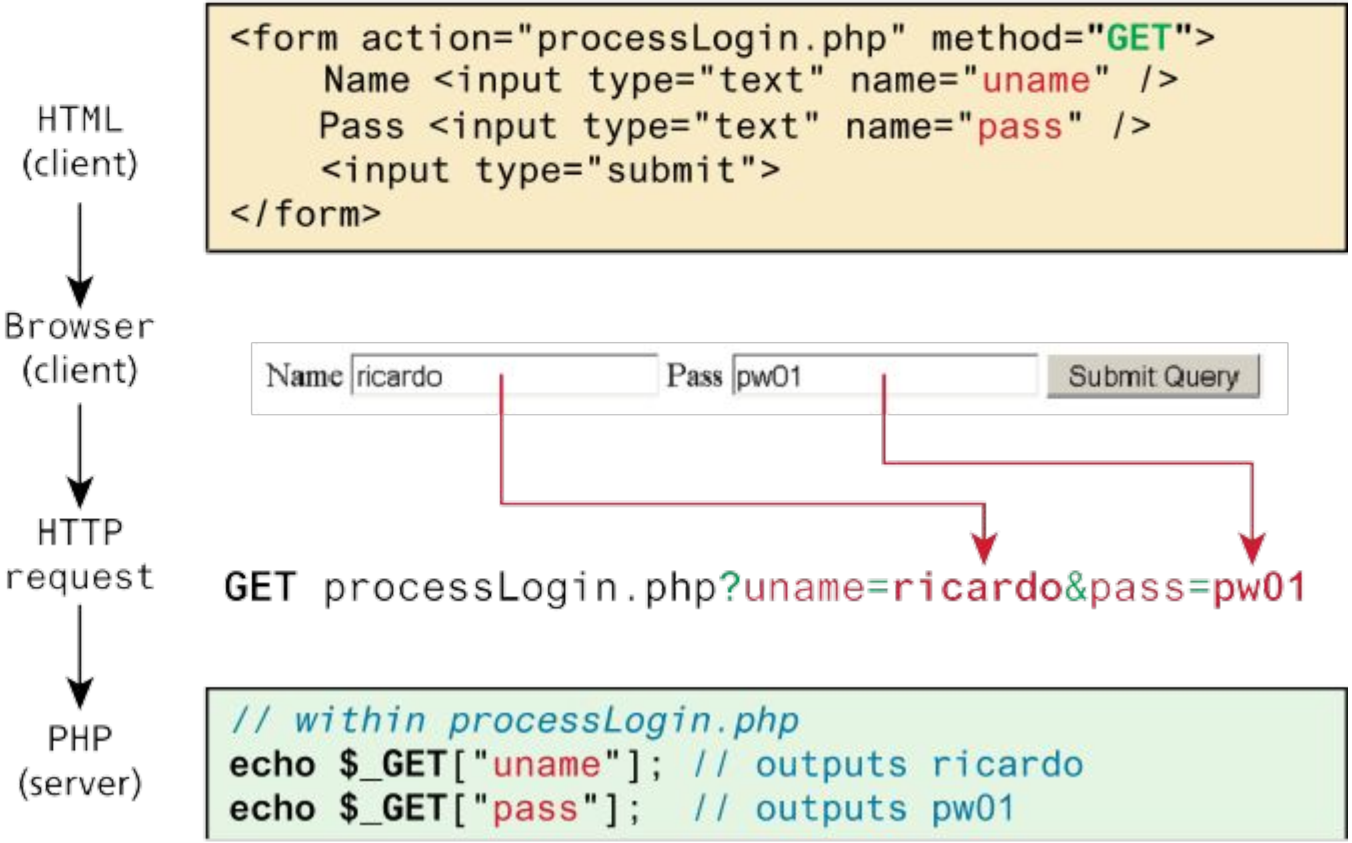
This command makes PHP report more errors

Superglobal Arrays

- PHP uses special predefined associative arrays called superglobal variables that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information
  - $GLOBALS Array for storing data that needs superglobal scope
  - $_COOKIES Array of cookie data passed to page via HTTP request
  - $_ENV Array of server environment data
  - $_FILES Array of file items uploaded to the server
  - $_GET Array of query string data passed to the server via the URL
  - $_POST Array of query string data passed to the server via the HTTP header
  - $_REQUEST Array containing the contents of $_GET, $_POST, and $_COOKIES
  - $_SESSION Array that contains session data
  - $_SERVER Array containing information about the request and the server
-

$_GET

Relating sent query string elements in PHP



```
<form action="processLogin.php" method="GET">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

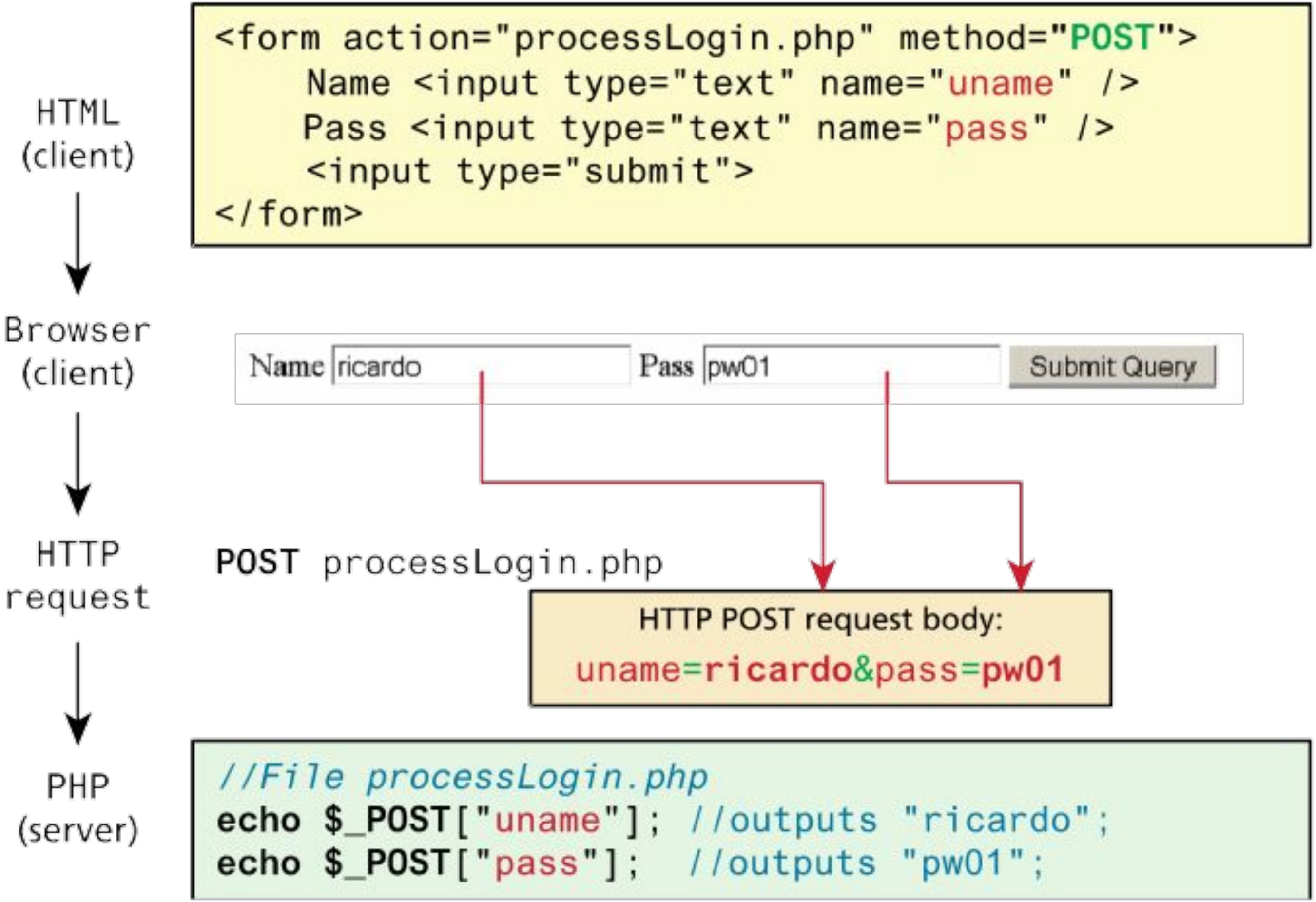
Name ricardo Pass pw01 Submit Query

```
GET processLogin.php?uname=ricardo&pass=pw01
```

```
// within processLogin.php
echo $_GET["uname"]; // outputs ricardo
echo $_GET["pass"];  // outputs pw01
```

$_POST

Relating sent query string elements in PHP (POST)



HTML (client)

```
<form action="processLogin.php" method="POST">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser (client)

Name ricardo Pass pw01

HTTP request

```
POST processLogin.php
```

HTTP POST request body:

```
uname=ricardo&pass=pw01
```

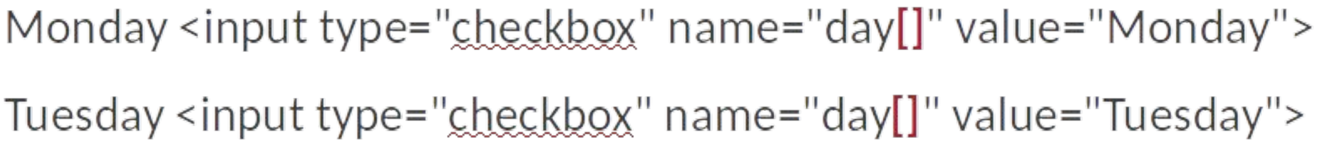
PHP (server)

```
//File processLogin.php
echo $_POST["uname"]; //outputs "ricardo";
echo $_POST["pass"];  //outputs "pw01";
```

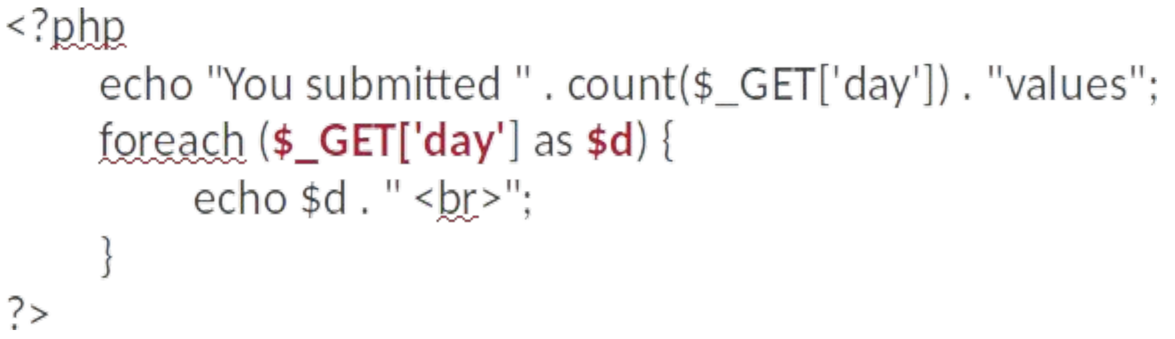
- Form display and processing on same page: login.php
- Form display and processing on separate pages:
    - login.html
    - processLogin.php
- Determining If Any data Sent
    - use the isset() function in PHP to see if there is any value set for a particular expected key
    - if ($_SERVER["REQUEST_METHOD"] == "POST") {
        - if (isset($_POST["uname"]) && isset($_POST["pass"])) {
            - // handle the posted data.
    - DEMO: superglobals_SERVER_2.php

Accessing Form array data

DEMO: superglobals_GET_formarray.php and superglobals_GET_form_chkbox.html



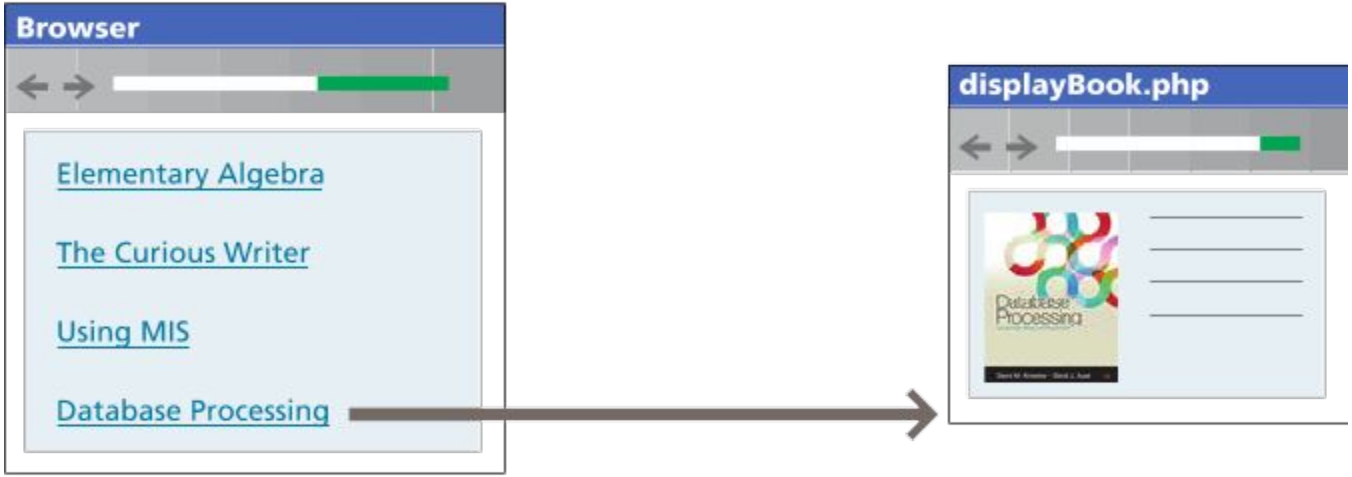
```
Monday <input type="checkbox" name="day[]" value="Monday">

Tuesday <input type="checkbox" name="day[]" value="Tuesday">
```



```
<?php
      echo "You submitted " . count($_GET['day']) . "values";
      foreach ($_GET['day'] as $d) {
            echo $d . " <br>";
      }
?>
```

Using query strings in hyperlinks

DEMO:
extended-example.php

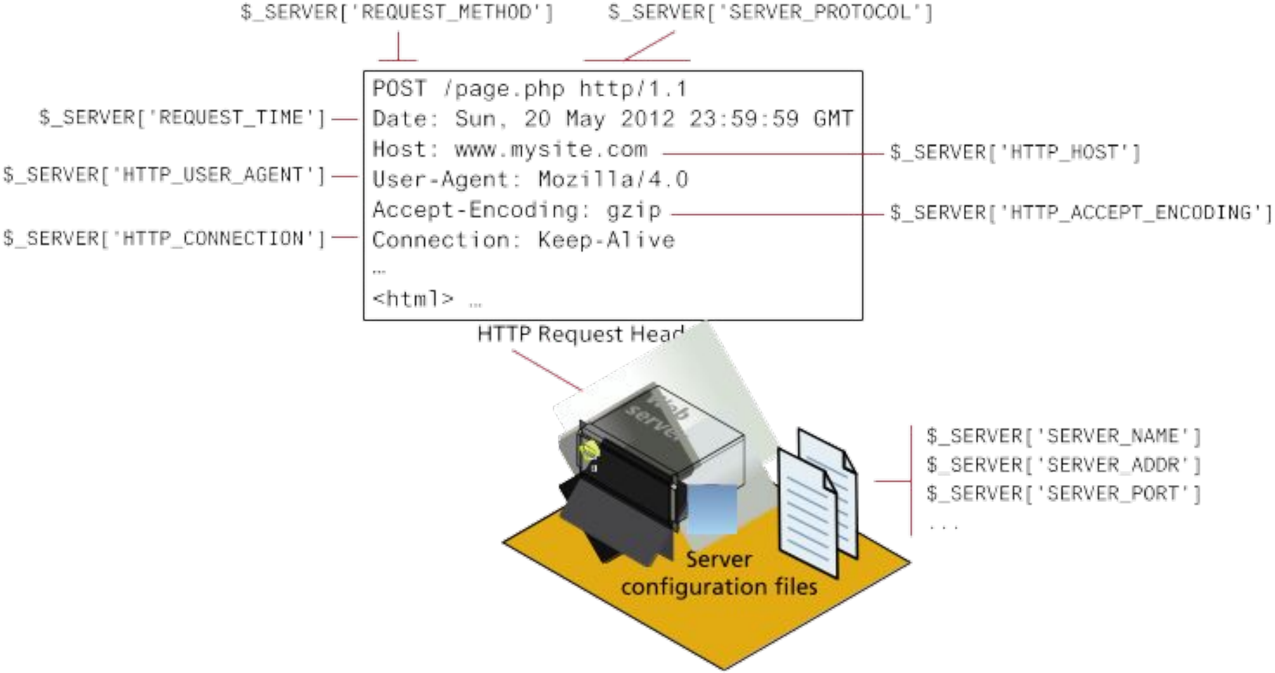


```
<a href="displayBook.php?isbn=0132145375">Database Processing</a>
```

Query string

$_SERVER Array

Server Information Keys



$_SERVER['REQUEST_METHOD']

$_SERVER['SERVER_PROTOCOL']

$_SERVER['REQUEST_TIME']

$_SERVER['HTTP_USER_AGENT']

$_SERVER['HTTP_CONNECTION']

```
POST /page.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Accept-Encoding: gzip
Connection: Keep-Alive
…
<html> …
```

$_SERVER['HTTP_HOST']

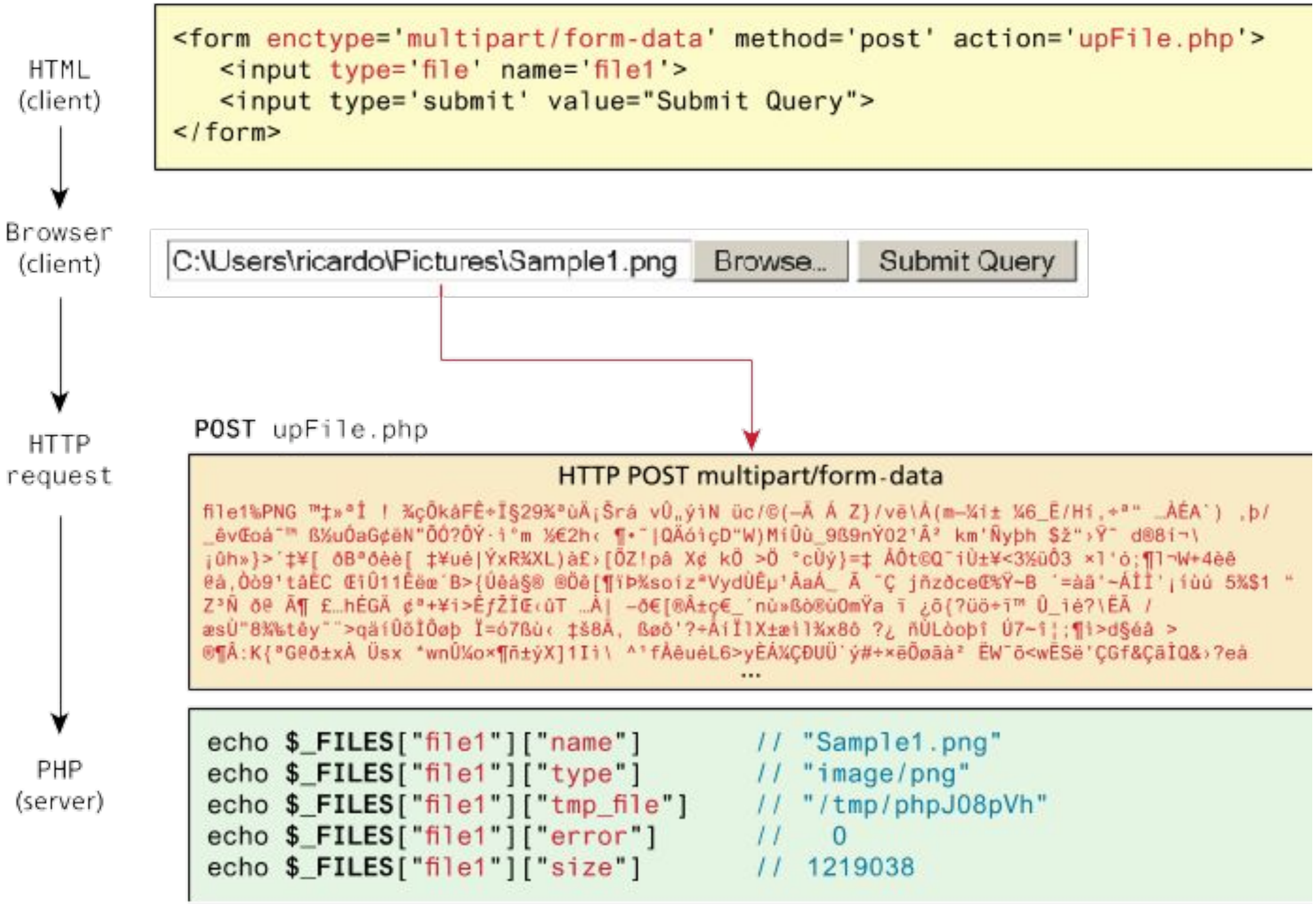
$_SERVER['HTTP_ACCEPT_ENCODING']

HTTP Request Head

Server configuration files

$_SERVER['SERVER_NAME']
$_SERVER['SERVER_ADDR']
$_SERVER['SERVER_PORT']
...

$_FILES



HTML
(client)

```
<form enctype='multipart/form-data' method='post' action='upFile.php'>
   <input type='file' name='file1'>
   <input type='submit' value="Submit Query">
</form>
```

Browser
(client)

C:\Users\ricardo\Pictures\Sample1.png

HTTP
request

POST upFile.php

HTTP POST multipart/form-data

```
file1%PNG ™‡»ªÎ ! %çÕkâFÊ÷Ï§29%ªùÄ¡Šrá vÛ„ýìN üc/©(–Ã Á Z}/vē\Å(m–¼i± ¼6_Ē/Hí,+ª" …ÀÉA`) ,þ/
_êvŒoá¯™ ß½uÔaG¢ëN"ÕÓ?ÔÝ·ì°m ½€2h‹ ¶•¯|QÄóìçD"W)MíÛù_9ß9nÝ02'Âª km'Ñyþh $ž"›Ÿ¯ d®8í¬\
¡ûh»}>´‡¥[ ðBªôèè[ ‡¥uė|ÝxR%XL)à£›[ÕZ!pâ X¢ kÖ >Ö °cÙý}=‡ ÅÔt©Q¯iÙ±¥<3%ûÔ3 ×1'ò;¶l¬W+4èê
®à,Òò9'tåÊC ŒiÛ11Êëæ´B>{Úêâ§® ®Öê[¶ïÞ%soíz*VydÙÊµ'ÂaÁ_ Ã ¯Ç jñzðceŒ%Ÿ–B ´=àâ'–ÁÎÏ'¡íùú 5%$1 "
Z³Ñ ð® Ã¶ £…hĖGÄ ¢ª+¥i>ÊƒŽÏŒ‹ûT …À| –ð€[®Â±ç€_´nù»ßò®ùOmŸa ī ¿õ{?üö÷ī™ Û_ìė?\ËÃ /
æsÙ"8%‰têy¨¨>qäíÛõÎÔøþ Ÿ=ó7ßù‹ ‡š8Ä, ßøô'?÷ÄíÏlX±æìl%x8ô ?¿ ñÙLòoþî Ú7~î¦;¶ì>d§éâ >
®¶Å:K{ªG®ð±xÀ Üsx ʻwnÛ¼o×¶ñ±ýX]1Ii\ ^'fÅêuėL6>yÈÁ%ÇÐUÜ`ý#+×ēÕøãà² ËW¯õ<wĖSë'ÇGf&ÇãÎQ&›?eå
...
```

PHP
(server)

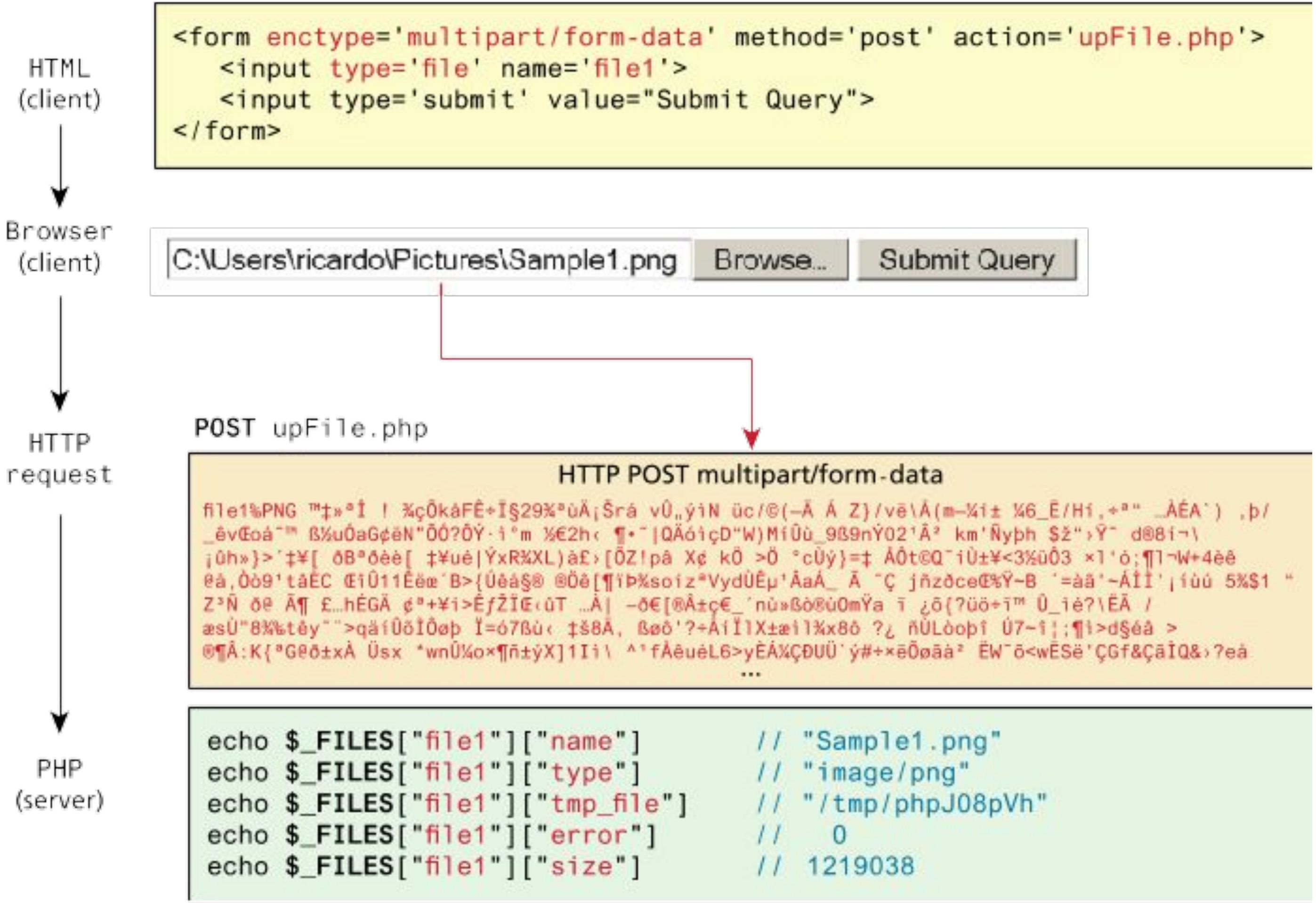
```
echo $_FILES["file1"]["name"]        //  "Sample1.png"
echo $_FILES["file1"]["type"]        //  "image/png"
echo $_FILES["file1"]["tmp_file"]    //  "/tmp/phpJ08pVh"
echo $_FILES["file1"]["error"]       //    0
echo $_FILES["file1"]["size"]        //  1219038
```

$_FILES examples

- Uploading a file: **superglobals_FILES_0.php**
- checking for errors: **superglobals_FILES_1.php**
- File Size restriction
  - HTML form attributes in inputs (browser): superglobals_FILES_2.php
  - JavaScript (browser): **superglobals_FILES_3.php**
  - Php validation (server): **superglobals_FILES_4.php**
- Limiting the Type of File Upload: **superglobals_FILES_5.php**
- Moving the file: **superglobals_FILES_6.php**

- Form display and processing on same page: login.php
- Form display and processing on separate pages:
    - login.html
    - processLogin.php
- Determining If Any data Sent
    - use the isset() function in PHP to see if there is any value set for a particular expected key
    - if ($_SERVER["REQUEST_METHOD"] == "POST") {
        - if (isset($_POST["uname"]) && isset($_POST["pass"])) {
            - // handle the posted data.
    - DEMO: superglobals_SERVER_2.php

$_FILES



```
<form enctype='multipart/form-data' method='post' action='upFile.php'>
   <input type='file' name='file1'>
   <input type='submit' value="Submit Query">
</form>
```

C:\Users\ricardo\Pictures\Sample1.png

POST upFile.php

HTTP POST multipart/form-data

```
file1%PNG ™‡»ª Î ! %çÕkâFÊ+Ï§29%ªùÄ¡Šrá vÛ„ýiN üc/©(–Ã Á Z}/vē\Å(m–¼í± ¼6_Ē/Hí,+ª" …ÀÉA`) ,þ/
_êvŒoá¯™ ß½uÔaG¢ëN"ÕÓ?ÔÝ·ì°m ½€2h‹ ¶•¯|QÄóìçD"W)MíÛù_9ß9nÝ02¹Â² km'Ñyþh $ž"›Ÿ¯ d®8í¬\
¡Ûh»}>´‡¥[ ðBªõèè[ ‡¥uė|ÝxR%XL)à£›[ÕZ!pâ X¢ kÖ >Ö °cÙý}=‡ ÅÔt©Q¯iÙ±¥<3½ûÔ3 ×1'ò;¶1¬W+4èê
®à,Òò9¹tâÈC ŒiÛ11Ëëæ´B>{Úêâ§® ®Öê[¶ïÞ%soíz*VydÙÊµ¹ÅaÁ_ Ã ¯Ç jñzðceŒ%Ÿ–B ´=àâ'–ÁÎÏ'¡íùú 5%$1 "
Z³Ñ ð® Ã¶ £…hÉGÄ ¢ª+¥i>ÊƒŽÏŒ‹ûT …À| –ð€[®Â±ç€_´nù»ßò®ùOmŸa ï ¿õ(?üö+ï™ Û_ìė?\ÊÃ /
æsÙ"8%‰têy˝˝>qäíÛõÎÔøþ Ï=ó7ßù‹ ‡š8Å, ßøô'?+ÅiÏ1X±æì1%x8ô ?¿ ñÙLòoþî Ú7–î¦;¶ì>d§éâ >
®¶Å:K{ªG®õ±xÀ Üsx ‘wnÛ%o×¶ñ±ÿX]1Ii\ ^¹fÅêuéL6>yÈÁ%ÇÐUÜ´ý#+×ēÕøãà² ËW¯õ<wÉSë'ÇGf&ÇãÎQ&›?eá
                                        ...
```

```
echo $_FILES["file1"]["name"]        //  "Sample1.png"
echo $_FILES["file1"]["type"]        //  "image/png"
echo $_FILES["file1"]["tmp_file"]    //  "/tmp/phpJ08pVh"
echo $_FILES["file1"]["error"]       //    0
echo $_FILES["file1"]["size"]        //  1219038
```

Mixing HTML and PHP in one file

The login.html and login.html example

$_FILES examples

- Uploading a file: **superglobals_FILES_0.php**
- checking for errors: **superglobals_FILES_1.php**
- File Size restriction
  - HTML form attributes in inputs (browser): superglobals_FILES_2.php
  - JavaScript (browser): **superglobals_FILES_3.php**
  - Php validation (server): **superglobals_FILES_4.php**
- Limiting the Type of File Upload: **superglobals_FILES_5.php**
- Moving the file: **superglobals_FILES_6.php**
  - **The owner of the folder to move the file to should be the same as the owner of httpd process**
  - **https://stackoverflow.com/questions/8103860/move-uploaded-file-gives-failed-to-open-stream-permission-denied-error-after**

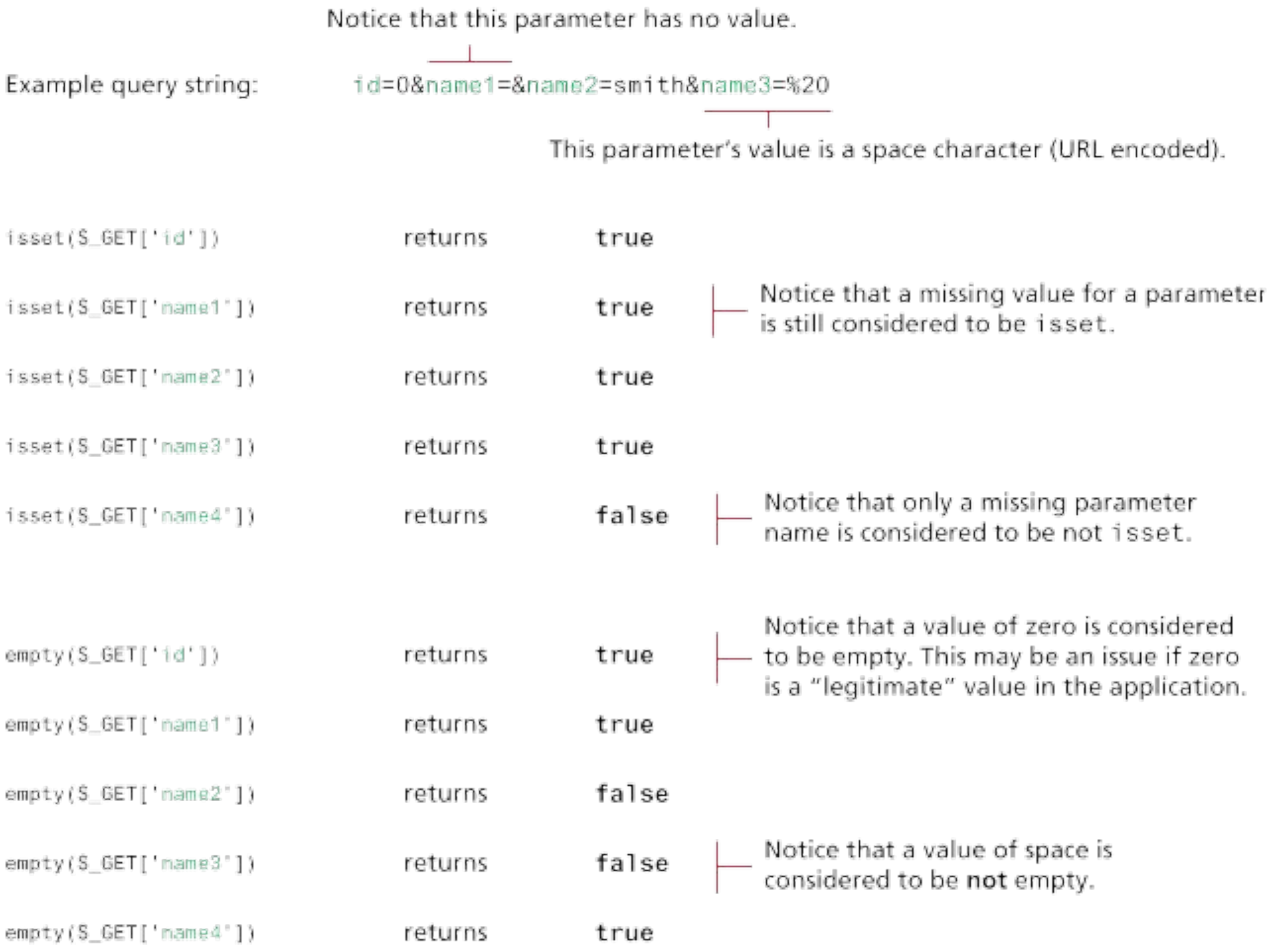
Reading/Writing Files

- There are two basic techniques for read/writing files in PHP:
    - **All-In-Memory access** . In this technique, we can read the entire file into memory (i.e., into a PHP variable). While not appropriate for large files, it does make processing of the file extremely easy.
        - **file()** Reads the entire file and returns an array, with each array element corresponding to one line in the file.
        - **file_get_contents()** Reads the entire file and returns a string variable.
        - **file_put_contents()** Writes the contents of a string variable out to a file.
- Example: streamAccess.php

Types of error

- Expected errors
- Warnings
  - Problems that generate a php warning message but will not halt the execution of the page.
  - e.g., Calling a function without a required parameter
- Fatal Error
  - The execution of page will terminate

How to deal with expected Errors?



Notice that this parameter has no value.

Example query string: `id=0&name1=&name2=smith&name3=%20`

This parameter's value is a space character (URL encoded).

`isset($_GET['id'])`	returns	`true`	
`isset($_GET['name1'])`	returns	`true`	Notice that a missing value for a parameter is still considered to be `isset`.
`isset($_GET['name2'])`	returns	`true`	
`isset($_GET['name3'])`	returns	`true`	
`isset($_GET['name4'])`	returns	`false`	Notice that only a missing parameter name is considered to be not `isset`.
`empty($_GET['id'])`	returns	`true`	Notice that a value of zero is considered to be empty. This may be an issue if zero is a "legitimate" value in the application.
`empty($_GET['name1'])`	returns	`true`	
`empty($_GET['name2'])`	returns	`false`	
`empty($_GET['name3'])`	returns	`false`	Notice that a value of space is considered to be **not** empty.
`empty($_GET['name4'])`	returns	`true`	

PHP error reporting

- **error_reporting** specifies which type of errors are to be reported
    - error_reporting (E_ALL)
- The **display_error** setting specifies whether error messages should or should not be displayed in the browser
    - ini_set('display_errors','0');
    - It can also be set within the php.ini file:
    - display_errors = Off
- The **log_errors** setting specifies whether error messages should or should not be sent to the server error log.
    - ini_set('log_errors','1');
    - It can also be set within the php.ini file:
    - log_errors = On

Regular Expression

- A regular expression is a set of special characters that define a pattern.
- Use regular expressions to ensure that input data follows a specific format.
  - PHP, JavaScript, Java, the .NET environment, and most other modern languages support regular expressions (each slightly different)

Pattern Matching in PHP

- PHP has two functions:
    - preg_match(regex, str)
        - Returns a Boolean value
    - preg_split(regex, str)
        - Returns an array of the substrings
- SHOW word_table.php

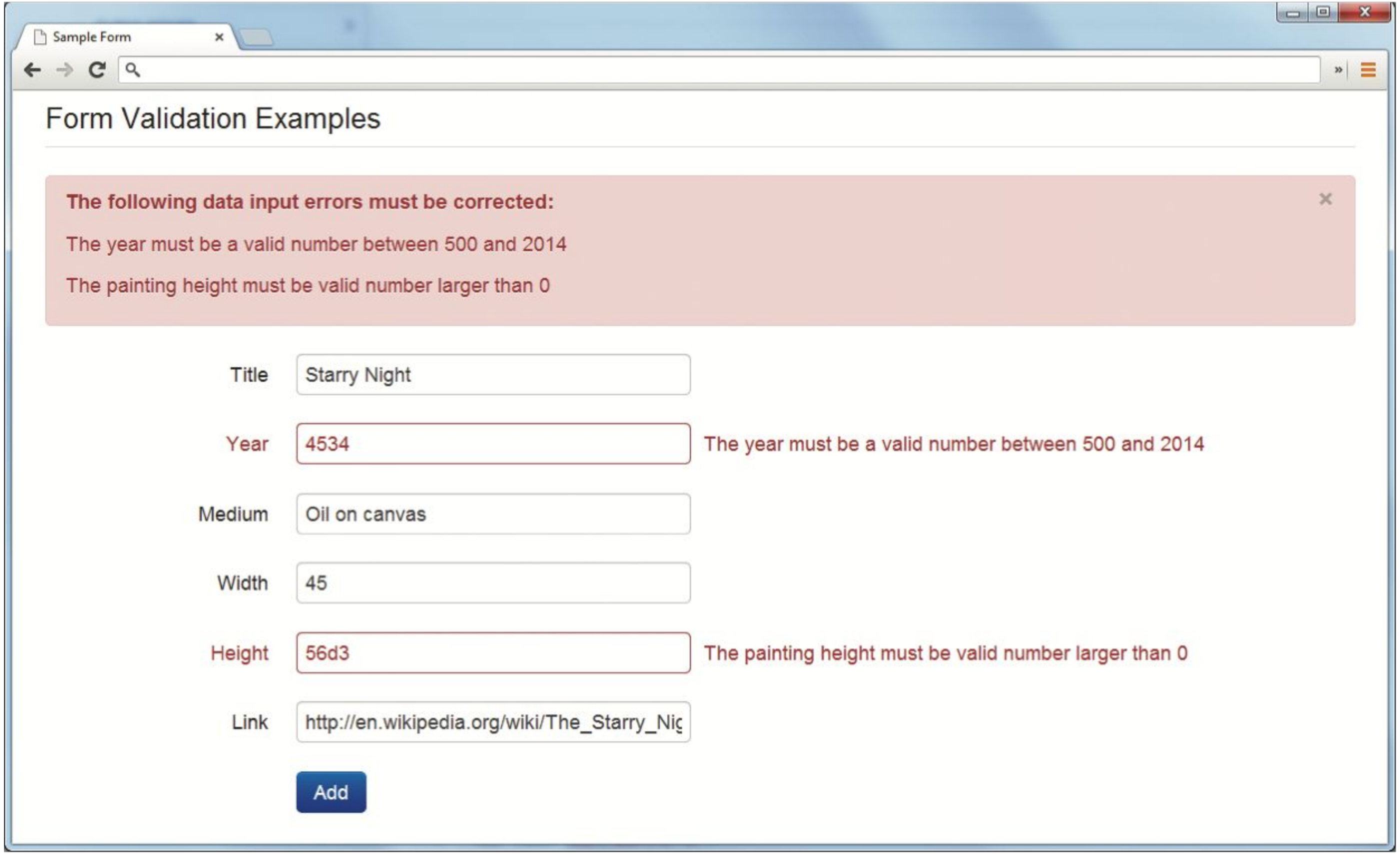
Validating User Input

- Types of Input Validation
  - Required information: email
  - Correct data type: date
  - Correct format: phone-number
  - Comparison
  - Range Check: numbers
  - Custom

Validating User Input

Validating User Input

Notifying the user



Form Validation Examples

The following data input errors must be corrected:

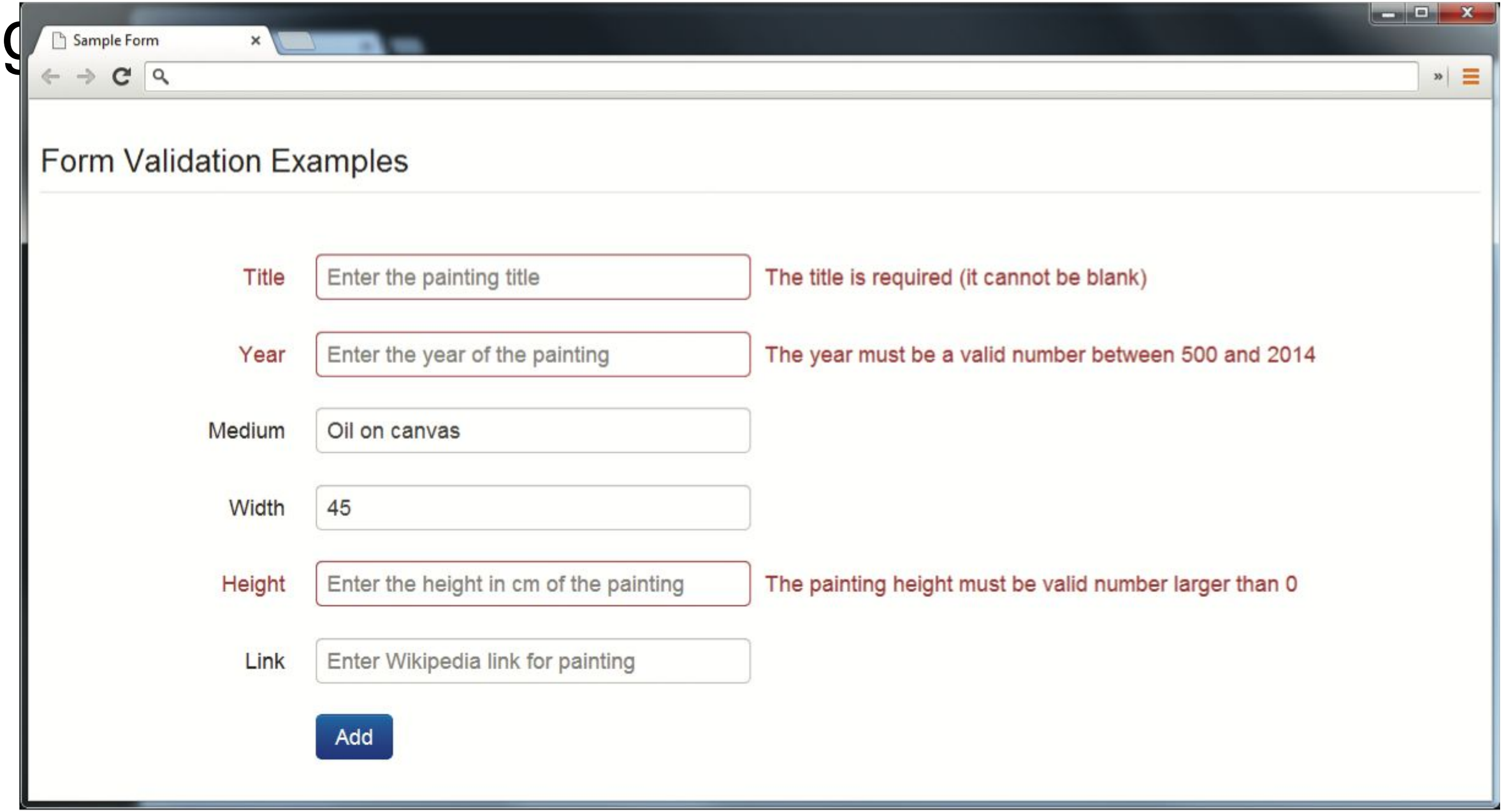
The year must be a valid number between 500 and 2014

The painting height must be valid number larger than 0

Title	Starry Night	
Year	4534	The year must be a valid number between 500 and 2014
Medium	Oil on canvas	
Width	45	
Height	56d3	The painting height must be valid number larger than 0
Link	http://en.wikipedia.org/wiki/The_Starry_Nig	

Validating

How to Reduce Validation Errors – show where error located



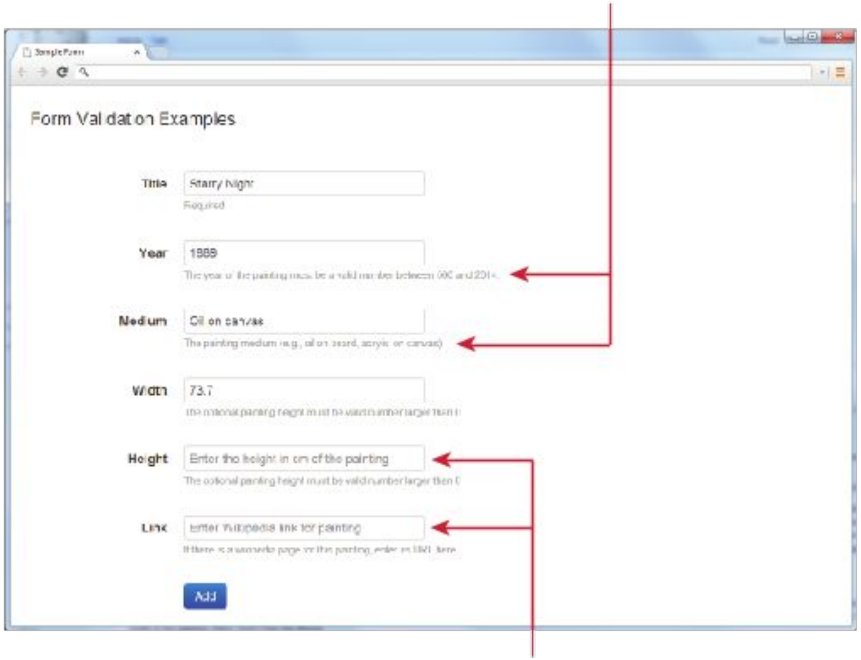
Form Validation Examples

Title	Enter the painting title	The title is required (it cannot be blank)
Year	Enter the year of the painting	The year must be a valid number between 500 and 2014
Medium	Oil on canvas	
Width	45	
Height	Enter the height in cm of the painting	The painting height must be valid number larger than 0
Link	Enter Wikipedia link for painting	

Validating User Input

How to Reduce Validation Errors – providing textual hints

Static textual hints



Sample Form

Form Validation Examples

Title: Starry Night

Required

Year: 1889

The year of the painting must be a valid number between 900 and 2014

Medium: Oil on canvas

The painting medium (e.g., oil on board, acrylic on canvas)

Width: 73.7

The optional painting height must be valid number larger than 0

Height: Enter the height in cm of the painting

The optional painting height must be valid number larger than 0

Link: Enter Wikipedia link for painting

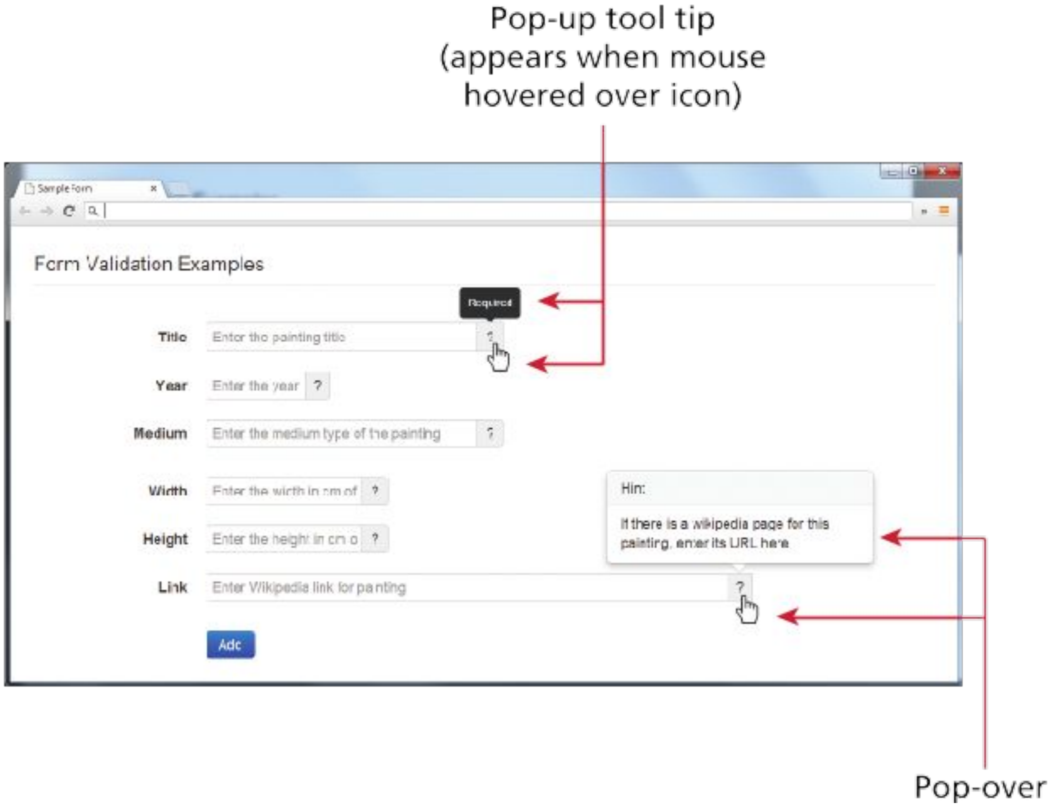
If there is a wikipedia page for this painting, enter its URL here

Placeholder text
(visible until user enters a value into field)

```
<input type="text" … placeholder="Enter the height ...">
```

Validating User Input

How to Reduce Validation Errors – use tool tips



Pop-up tool tip (appears when mouse hovered over icon)

Form Validation Examples

Required

Title Enter the painting title ?

Year Enter the year ?

Medium Enter the medium type of the painting ?

Width Enter the width in cm of ?

Height Enter the height in cm o ?

Link Enter Wikipedia link for painting ?

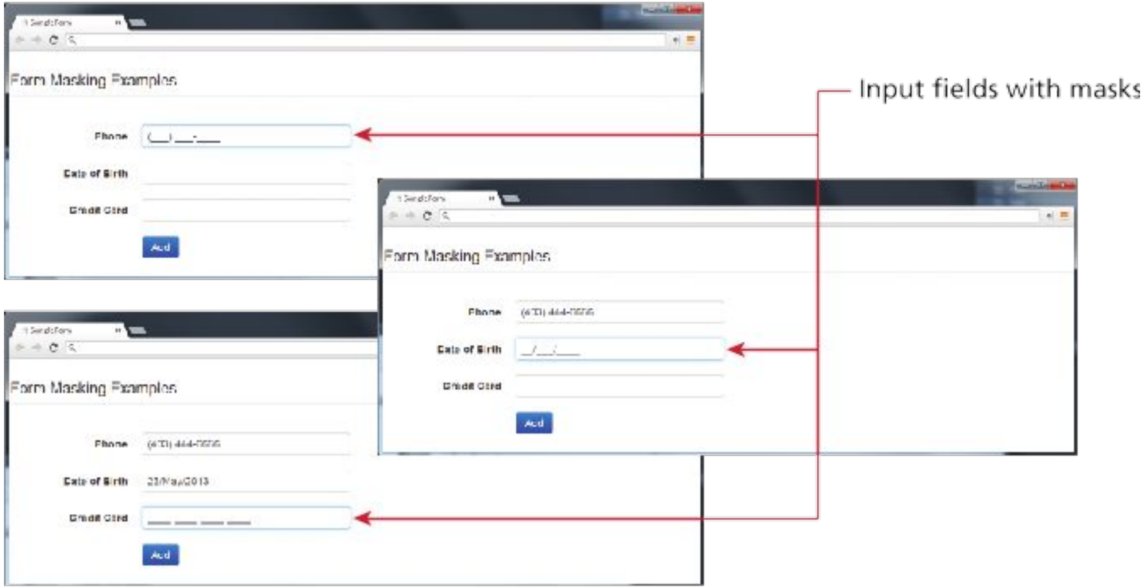
Hint

If there is a wikipedia page for this painting, enter its URL here

Pop-over

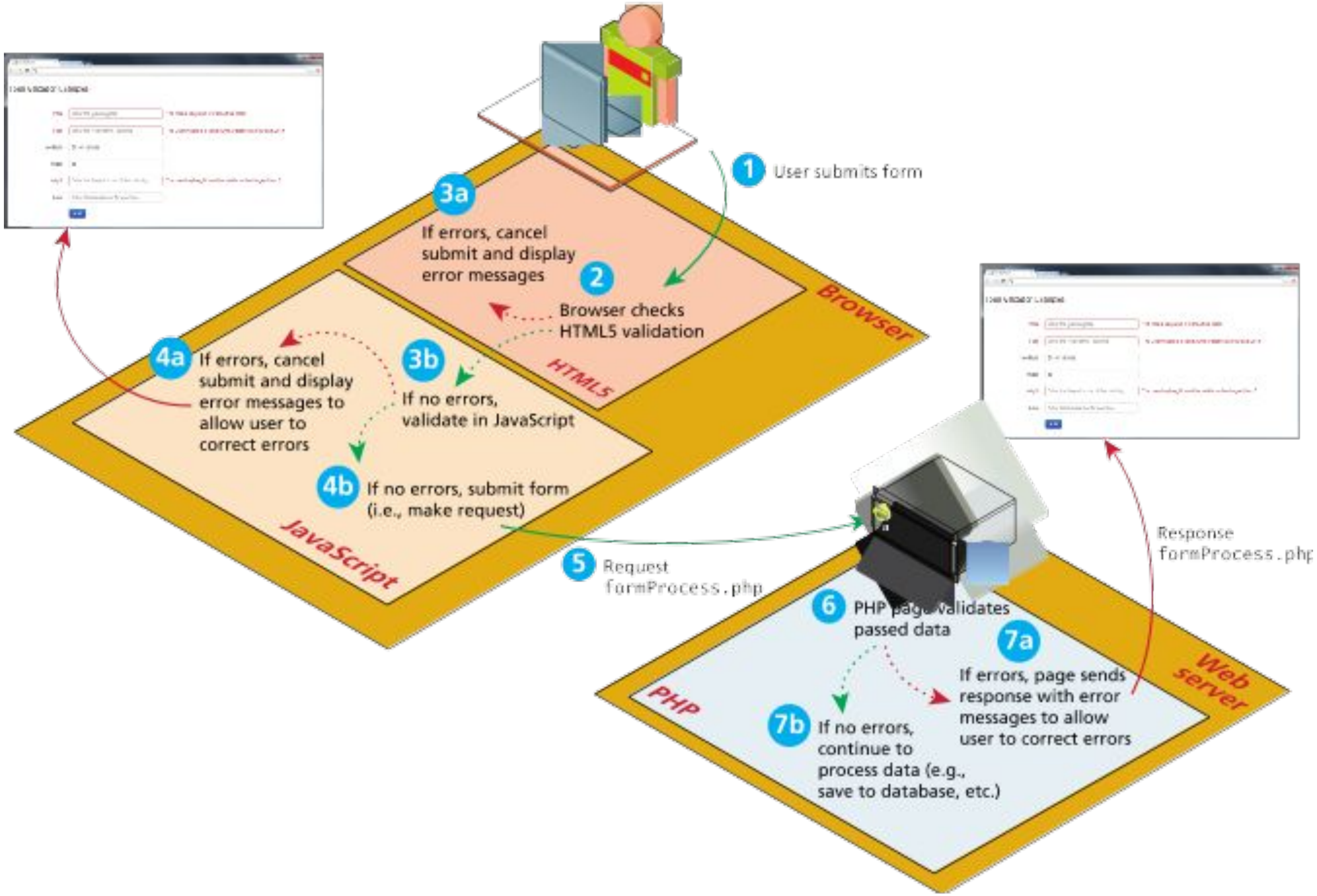
Validating User Input

How to Reduce Validation Errors – use input masks



Input fields with masks

Where to Perform Validation?



1 User submits form

2 Browser checks HTML5 validation

3a If errors, cancel submit and display error messages

3b If no errors, validate in JavaScript

4a If errors, cancel submit and display error messages to allow user to correct errors

4b If no errors, submit form (i.e., make request)

5 Request formProcess.php

6 PHP page validates passed data

7a If errors, page sends response with error messages to allow user to correct errors

7b If no errors, continue to process data (e.g., save to database, etc.)

Response formProcess.php

Where to Perform Validation

- Validation at the JavaScript Level: HTML5, JavaScript
    - Client process
    - Can reduce server load
    - Can be bypassed
- Validation at the PHP Level
    - Validation on the server side using PHP is the most important form of validation and the only one that is absolutely essential.
-

Form Handling

- Forms could be handled by the same document that creates the form, but that may be confusing
- PHP particulars:
    - It does not matter whether GET or POST method is used to transmit the form data
    - PHP builds an array of the form values ($_GET for the GET method and $_POST for the POST method – subscripts are the widget names)
- SHOW popcorn3.html & popcorn3.php