

Week 9: Introduction to PHP

Agenda

- Overview
 - What's Due?
 - Midterm
- Introduction to PHP
 - Server-Side and Client-Side Programming Languages
 - Communication Between Front-End and Back-End
- Basic PHP Syntax
 - Using PHP on a Page
 - Comments
 - Variables
 - Functions
 - Conditional Statements
 - Loops
 - Common Operators and Expressions
 - Creating Output
 - Working with Forms
 - Useful Functions
 - Running PHP Code
 - Examples
- PHP Exercise
- PHP Functions (Two Posts)
- Next Week

Overview

This week we are learning about PHP and how it can be used in a website. We are going to be learning PHP because it is one of the most widely used server-side programming languages. It is free, open-source, and can be used to add complex interactive components to our pages.

What's Due

- PHP Exercise
- PHP Functions (Two Posts)

Midterm

The Midterm due date has been moved to April 4th to provide all of you with a little more time to work on it. I know this semester has been stressful for some of you and I think extra time will result in higher quality work. If you need help with it, please come to office hours and I can walk you through creating layouts, features, and other things.

In total, you need to create 3 mostly done pages. The layouts for these pages should be complete. However, the content doesn't need to be finalized. You can use lorem ipsum for the text and image placeholders. For more details about requirements, please see the Final Project Description file on Blackboard→Course Materials→Final Project.

Introduction to PHP

PHP is a popular server-side programming language which is free, open-source, and relatively easy to learn. We've already worked with another programming language which was JavaScript.

The major difference between the two is that PHP is a server-side language while JavaScript is a client-side language. It's important to understand the difference between those types so you can use their strengths appropriately for your websites.

Server-Side and Client-Side Programming Languages

To understand the difference between these two things, we must think about what happens when we look at a website.

When we open a browser and type in a URL, or click a link on a page, our browser sends a request out to a web server. The request may be for a webpage or file (HTML, PDF, JPEG, etc). Depending on that status of the server (working or not), the web server will then gather up all resources necessary for the page requested and place them in your browser.

If a web page is requested, the HTML will be parsed by the browser. If the HTML references any CSS, JS, or any other resource, those files/resources will be used to help the page display. In this instance, CSS and JS are client-side codes (also known as browser-side or front-end). This is how the websites that we've been building so far work.

Client-side code is great for interactivity on a website, but it has a few downsides:

- The end user can see, alter, and disable the files in their browser, which makes it insecure
- Browser side code only has access to resources that are loaded on the browser and can't directly access any resources that are on the server, such as additional files or databases

This is where server-side programming languages like PHP come in. When the request from the browser reaches the server, the server-side programming language will run on the server. It will gather and process information from resources on the server and assemble it into finished HTML files that get sent back to the browser.

Some of the things that server-side code, specifically PHP gets used for:

- Dynamically creating HTML files
 - PHP can be used to assemble HTML files piece by piece. You can have your navigation in one file and your main body in another and 'glue' them together, or you can insert specific text on a website under certain conditions. PHP is especially good at this.
- Communicating with a Database
 - You can use PHP to manipulate data in a database (insert, update, delete, etc.)
- Processing User Input
 - PHP can be used to process user input after it has been submitted with a form. If you want to use the input, it is a good idea to validate it with a server-side language, because users might try to send malicious code to your server. If you only verified it on the client-side, the malicious user could easily disable the verification.
- Communicating with outside resources
 - You can use server-side code to communicate with outside resources. An example would be sending an email after a form is submitted.
- Work with files on the server
 - You can use PHP to access files that are located on the server that you may not want to send to the client. You can do things like get information out of a file, edit a file, or create a new file.

Communication between Front-End and Back-End:

Since PHP and JavaScript work in different environments, they can't communicate directly with each other. However, there are ways for PHP to pass information to JavaScript and vice versa. Simple variables like strings and numbers can be translated easily, but more complex ones like arrays might need to be encoded.

PHP to JavaScript:

- PHP can generate JavaScript code inside HTML files or insert values into code
- You can use PHP to create Cookies that JavaScript can read

Front-end to PHP:

- Requests to the server such as POST and GET can pass information to PHP, for example with the use of forms
- AJAX is a way of communicating with the server without having to reload a whole page

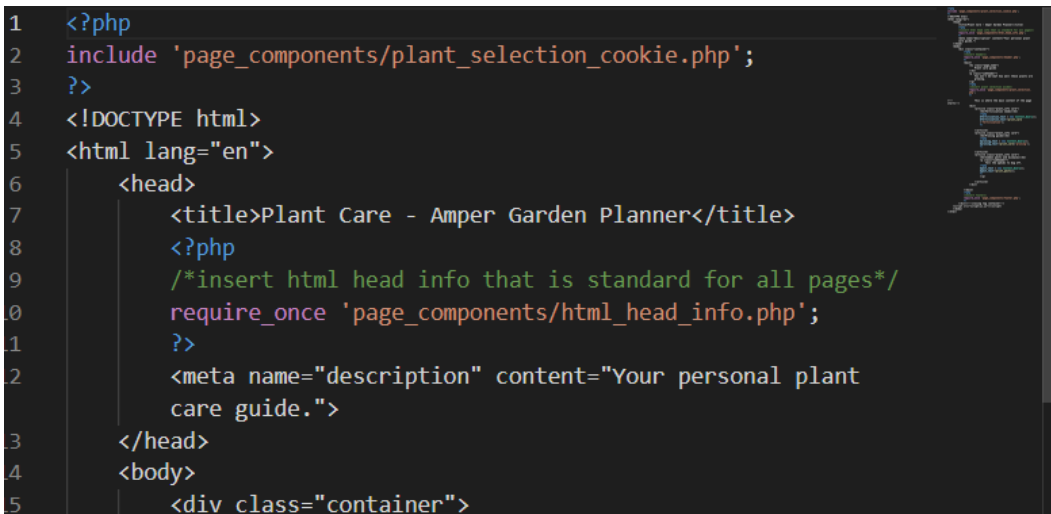
Basic PHP Syntax

Even though PHP and JavaScript run in different places and are used for different purposes, you're going to find that many of the basic concepts are similar. PHP uses things like variables, conditional statements, loops, and functions. You're already familiar with those concepts because of JavaScript taught earlier in the semester. The main goal now is to learn the specific syntax of this new language and how it handles different things.

Using PHP on a Page

PHP code can only execute in files with a .php extension. You can create a regular HTML file but save it with a .php extension instead. Then you can include PHP code in these files by enclosing it in php tags:

```
<?php
//Write your code
?>
```



```
1 <?php
2 include 'page_components/plant_selection_cookie.php';
3 ?>
4 <!DOCTYPE html>
5 <html lang="en">
6   <head>
7     <title>Plant Care - Amper Garden Planner</title>
8     <?php
9       /*insert html head info that is standard for all pages*/
10      require_once 'page_components/html_head_info.php';
11      ?>
12     <meta name="description" content="Your personal plant
13       care guide.">
14   </head>
15   <body>
16     <div class="container">
```

In the screenshot above, I use PHP in two different areas of the code (lines 1-3 and lines 8-11). Each section contains the opening/closing PHP tags with our PHP code between them. You can also have pure PHP files, but you still need to write your PHP code inside of these tags.

The code between those opening and closing PHP brackets will be executed by the server and then rendered as HTML. Any HTML placed inside of a PHP file must be separated from the PHP. That means you can't place HTML code directly inside of the PHP tags unless it is part of some sort of statement or function. If you have errors in your code, the PHP will fail and sometimes things won't render, or your back-end info could be exposed in error messages. For the purposes of this class, please make sure you aren't storing sensitive information on these websites. Also, don't use passwords like ones previously used in other classes/applications.

Comments

Comments in PHP are designated with # or // for a single line comment and /* */ for multi-line comments.

```
// comment on one line
/* comment on
multiple lines */
# comment on one line again
```

Variables

Variables in PHP are designated by the \$ sign. You don't have to declare them using var like in JavaScript, but every time you use them, you must have a \$ sign at the beginning. Variable names can contain letters, numbers, and underscores, but they can't start with a number. They are case-sensitive.

Like JavaScript, variables in PHP can be a variety of types. In the screenshot below, I've created a few variables which are named based on the variable type they represent. Included are string, integer, float, boolean, array, associative array, object, and resource. I am not covering each one in detail, but you should know they exist.

The variables below are created in what is known as statements. Like in JavaScript, statements should be terminated with a semicolon (“;”).

```
<?php
$string = "Hello, I am a string!";
$integer = 500;
$float = 4.52;
$boolean = true;
$array = [98, 34, "String in an array", "Hi"];
$associative_array = ["first"=>"item one", "second"=>43, "name"=>"Lu"];
$object = new stdClass(); //You can define and create your own classes or use the ones that come with PHP
$resource = fopen("file.txt", "r"); //resources are external resources like a file or a database

?>
```

Conditional Statements

In PHP you have conditional statements much like you do in JavaScript. You have if, else, elseif, and switch. In the left screen shot, we have a basic if/elseif/else structure.

In the right screen shot, we have a switch statement which operates similarly. The value to check goes inside of the parenthesis (\$x). We then check the value using “case :” with the value we’re checking displayed after the word case.

In this instance, we’re checking numerical values. When the code reaches a case that matches the switch portion, the code inside will execute until it reaches break. In the case of 0, it will find a

break immediately. In the case of 1, it will go and execute the code for 2 as well. Default is similar to “else” ...it only executes if none of the other cases are triggered.

```
<?
$x = 1;
//conditionals
if ($x > 0){
    //do stuff
} elseif ($x = 0){
    //do something else
} else {
    //do another thing
}
```

```
switch ($x){
    case 0:
        //do stuff
        break;
    case 1:
    case 2:
        //do something else
        break;
    default:
        //do another thing
}
```

Loops

Looping in PHP works along similar concepts to looping in JavaScript.

- for loops are almost exactly like in JavaScript. You specify a looping variable, a cutoff limit, and increment through the variable. This is useful for looping for a set number of times.
- while loops are also very similar to JavaScript. You specify a cutoff condition and then make sure that it will get met at some point inside the loop. It is possible for the while loop to not run at all if the condition evaluates to false on the first test.
- The do ... while loop is very similar to the while loop, except that it will run a minimum of one time.
- foreach lets you loop over an array. You have two options for this one. You can either only use the values of the array or you can use both the keys and the values.
- The break and continue statements work just like they do in JavaScript.

```
<?
for ($i=0; $i<50; $i++){
    //do stuff
}

$keep_going = "yes";
while ($keep_going == "yes"){
    //do stuff
    //make sure you change $keep_going so that the loop can end
}

$example_array = ["list"=>"item one", "id"=>43, "name"=>"Lu"];
foreach ($example_array as $value){
    //do stuff with $value
}
//or
foreach ($example_array as $key=>$value){
    //do stuff with both $key and $value
}

?>
```

Functions

Like in JavaScript, you can create functions out of things that you want to do in multiple different places. Functions are declared with the function keyword, then you have the name of the function. If you want to add any arguments, you add them in round parenthesis and the body of the function follows inside curly brackets.

You can return values inside functions and save them into variables. Variables that are used as arguments in a function are typically passed by value, which means that they don't change their value outside of that function. You can tell PHP to pass them by reference instead which lets you modify the variables inside the function. You can do this by putting an ampersand (&) symbol in front of the argument when you declare the function.

```
<?php
function my_function(&$argument1, $argument2){
    $result = $argument1 + $argument2;
    return $result;
}
$result = my_function(5, 7);

?>
```

Common Operators and Expressions

PHP uses many of the same operators and expressions that you will be used to from other programming languages.

- Math operators: The math operators +, -, *, /, %, and ** do what you expect them to do
- You can use the logical operators “and” or “&&”, “or” or “||”. “!” is used for not, “Xor” is used to check if one of two expressions are true, but not both at the same time.
- Unlike JavaScript, string concatenation is not done using “+”, but is done using “.”

For more operators you can look under <https://www.php.net/manual/en/language.operators.php>

Creating Output

A very common thing to do in PHP is writing to an HTML file. You'll want to write the PHP code in the spot where you want the text to appear in the HTML file. You can use the echo and the print functions to do this. They can be used to output, strings and numbers. You can even use them to output HTML code into the file.

Echo also has a shorthand version that can be useful if all you're trying to do is echo a short piece into the HTML file. The shorthand version goes “<?=”The bit you're echoing.”?>”

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>PHP Introduction</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <?php
      $language = "PHP";
      echo("<h1>Inserting HTML Code with " . $language . "</h1>");
    ?>
    <h2>
      <?php
        print("Printing to a File is useful");
      ?>
    </h2>
    <p class="<?=$language?>">
      You can print PHP code into any place of the HTML file.
    </p>
  </body>
</html>
```

Working with Forms

One of the main strengths of PHP is its ability to work with forms. Without adding any PHP to a page, you can have the page's data passed to a PHP page. The class example has a section titled "Working with Forms" which demonstrates how that can work. To use a form, you'll use the action and method attributes.

The action attribute lets the form know where to go while the method attribute is used to determine how the information is passed. The action attribute will typically have a page as the value and the method attribute will have "get" or "post." If no value is provided in the action attribute, the page will submit the data to itself.

```
<form action="somepage.php" method="get">
  // Cool form stuff
</form>
```

Get and Post differ in how they handle data. Post is much better if the information needs to be secure. When using get, the variables being passed are visible in the URL. It also has a 2000-character limit while post has no limit and keeps the variables hidden. With get, you can bookmark the content since the variables are available. For the final project, get should be used when retrieving data using non-sensitive information. Post will be used for inputting data and when pulling sensitive information (so it doesn't appear in the browser URL).

When a user submits a form with an action and method, information for that form is automatically stored on the PHP server in a variable. This variable is an array which contains all items passed through the method used. The array is known as a "super global" variable since it can be access anywhere throughout the script. The array for information sent through an HTTP post method is `$_POST[]` and the one for get is `$_GET[]`. After the user submits data, we can access these arrays to see what was submitted and manipulate it.

The values that are submitted depend on the name attributes from the form fields. For example, let's say we have the following snippet of code which is from the class example this week.

```
<form name="form3" method="get">
  <label for="food">Favorite Food:</label>
  <input type="text" name="food"><br>

  <label for="number">Favorite Number:</label>
  <input type="text" name="number"><br>
  <button type="submit" name="submitFavs">Submit Favorites</button>
</form>
```

When the user clicks submit, the value for the food, number, and button get passed to the page. Since there is no action attribute, the form submits to the page itself. We use the "get" method and therefore have to refer to the `$_GET[]` array to access those values.

`$_GET['food']` refers to the input value for the favorite food field.

`$_GET['number']` refers to the input value for the favorite number field.

Knowing how to access those values, we could create a variable and assign the value to that variable.

```
$favFood = $_GET['food'];
```

Useful Functions

I'm going to briefly go over some useful functions to check user input from forms. These functions are necessary for completing the assignment this week, so make sure you review this content and the Course Examples. The PHP below is based on the HTML from the screenshot above.

```
if(isset($_GET['submitFavs'])) {
    // Are they empty?
    if(empty($_GET['food']) || empty($_GET['number'])) {
        echo "<p class='red'>Please submit your favorite food and number.</p>";
    } else {
        // Is the number provided actually a number?
        if(is_numeric($_GET['number'])) {
            $favNum = $_GET['number'];
            $favFood = $_GET['food'];
            echo "<p>Your favorite number is $favNum and you like $favFood.</p>";
        } else {
            echo "<p class='red'>Please submit a valid number.</p>";
        }
    }
}
```

Isset() – This function takes in an argument and returns true or false depending on if it's set. If a value is set, it means that it has been submitted. In the course examples provided, I use `isset()` to check if the button was submitted. If the button was pressed and our submit button is “set”, I can then check our inputs using the `empty()` function.

Empty() – This function takes in an argument and returns true if it's empty. Empty strings or even the number 0 count as “empty”. In the screenshot above, I use `empty()` on `$_GET['food']` and `$_GET['number']` to check if something was submitted for them. If nothing is submitted, we output an error. If something was submitted, we can go to the else statement to do more input checking.

Is_numeric() – This function takes in an argument and returns true if it's a number and false if it isn't. I use this on `$_GET['number']` to check if the value provided is a number. After I have checked that something was submitted, I can then check to see if what was submitted is a number. If it's a number, we can generate output with an echo statement. Otherwise, we can tell the user to submit a valid number.

Please review the Course Examples section for a more in-depth explanation of PHP concepts. The PHP examples for this week demonstrate using the various data types in addition to working with forms. It will be very helpful for when you need to do the PHP Exercise assignment for this week. After reading through the class example, you can look at the additional resources as needed below. You don't need to read everything, but these are guides which will help explain additional concepts in PHP.

If you have trouble and need to check your PHP, use this link:

<https://phpcodechecker.com/>

It will generally tell you where your error is and even if you don't understand what it means, look at the line numbers around the error and check for syntax problems. You must copy and paste all your code inside of the checker.

Running PHP Code

Because PHP code runs on the server and not the browser, you can't just directly open them with your browser like you do with HTML files. To run them you must open them on a server that runs PHP. The UAlbany servers that we have access to don't run PHP, so you'll have to put your PHP files on a host that does run it, such as 000webhost. Check the second part of the file about viewing your web pages to upload your PHP files.

If you do want to run PHP files on your local computer, you can do so by installing a local server on your computer. This is not required for this class, but if you want to work more with PHP files or with SQL, I would recommend it. You can find guides under the following links:

https://www.youtube.com/watch?v=XBj_le81sAc

<https://studyopedia.com/php/run-first-php-program-xampp-server/>

The part that teaches you how to install a local server starts at 6:21

Examples

Online Resources

- <https://www.youtube.com/playlist?list=PL0eyrZgxdwhwBToawjm9faFlixePexft-> (video playlist about PHP, feel free to pick and choose videos on any topic you want to know more about)
- <https://www.php.net/manual/en/langref.php> (official documentation, useful as a reference guide)
- <https://www.codecademy.com/learn/learn-php> (interactive tutorial, not required, just helpful if you prefer interactive learning)
- <https://www.tutorialrepublic.com/php-tutorial/php-data-types.php> (articles about data types)

A new video will be released on Monday afternoon. This video will cover PHP Basics in the context of this week's assignment. When it is available, the link will be added to the assignment description.

My Examples

- <https://cinf362.000webhostapp.com/examples/php-syntax.php>
- <https://cinf362.000webhostapp.com/examples/input-example.php>
- <https://cinf362.000webhostapp.com/examples/output-example.php> (visit the input page first!)

PowerPoint documents have also been released under Blackboard → Course Materials → Resources → PHP. One of them offers a brief introduction to PHP while the other will serve as a debugging guide.

You won't be able to right-click the page and view the source this week to view my PHP code. Since PHP is processed by the server and rendered as HTML, the user can't see the PHP code you're using. The code for the example below is in the "examples" folder that was included with this week's lecture notes zip folder. You can take all the contents of the examples folder and

place them somewhere on your 000webhost account and the pages should work. There are comments in the code to explain what is happening. Feel free to look at the live link to see the page in action.

Here's a link to **all** course examples:

<https://cinf362.000webhostapp.com/examples/> (PHP/SQL Examples)

<https://www.albany.edu/~cv762525/cinf362/examples/> (HTML, CSS, and JS examples)

<https://www.albany.edu/~cv762525/cinf362/videos/> (videos)

Before completing the assignment(s) for this week, please read the “Viewing Your Web Pages.docx” file on Blackboard. You will not be able to submit anything for the assignment without completing that portion first. It is located directly in the Lecture Notes folder.

PHP Exercise (Extra Credit Available)

Due Sunday, April 3rd at midnight

Download the “PHP-Exercise.zip” folder from Blackboard under this week’s Lecture Notes or the Assignment folder. Inside of the zip folder will be two PHP files for you to use: input.php and output.php. Your task is to create a form in the input.php file which will be used to pass data to the output.php file. The output.php file will check the data and either provide an error or display a div based on the dimensions/color provided from the input page.

input.php

In this file, you will create a form which will take in 3 inputs: width, height, and color. When the user clicks the submit button, your form should pass data to the output.php page creating a div with the dimensions/color provided. For example, if a user types in 200, 200, and red in your 3 input boxes and then hits submit, the output page should appear with a div that is 200px wide, 200px tall, and has a background color of red.

output.php

All PHP will be placed inside of this file. For the input page, you simply need to create the HTML form so that it can pass data to the output.php file. On the output.php page, you’ll use PHP to do the following:

1. Check if the form was submitted
2. Check if the values are valid
3. If the values are correct, a paragraph tags should appear saying: “Here is the box you requested!” A div with the dimensions submitted should also appear after that.
4. If the values are incorrect (else), a message should appear explaining what went wrong (“Please provide any missing values.” if anything is empty or “Please submit numerical values for the width/height.” if the height/width values aren’t numbers or are 0)

The code for the newly created div will ultimately look like this if you entered 300, 300, and aquamarine for the inputs:

```
<div style= “width: 300px; height: 300px; background: aquamarine;”></div>
```

For your PHP, the variables you create from the input will replace 300, 300, and aquamarine. Make sure to use the string concatenation operator (a period in PHP) to create this output like the

line below. **Make sure you type this out...copying from Word to a text editor may cause the characters to register incorrectly.**

echo "<div style='width: ' . \$width . 'px; height: '....and so on

The class examples have code built into them that is directly related to this assignment. I would highly recommend going over that code. I've listed some tips below with specific info for functions that will be useful.

General Tips

- Use isset() to check if the submit button has been set
- Use empty() to check if input value is present
 - If any of the inputs are empty output an error
- Use is_numeric() to check if the input value is a number
- Use string concatenation to link your input values to the div's style attribute
- Use echo along with HTML to create output to the page
- Choose either GET or POST, not both
- You will need two pages in total for this (input.php and output.php which are provided)
- Add labels to your form
- Use the id and name attributes for the inputs/buttons, they can have the same value

Make sure your PHP files are on a server that supports them. The UAlbany servers don't, so you should be using 000webhost or another platform such as byethost. If your PHP doesn't work, put your code in the PHP Checker link provided in the lecture notes above. It should give you a line number and a reason why it's failing. Check the line number (if provided) and adjust the code as necessary. Also, make sure your file has a .php extension. Without this extension, the browser won't know to treat it like a PHP file.

Your webpage is **due on Sunday, April 3rd at midnight**. To submit the work for this exercise, visit Blackboard → Course Materials → Lectures Notes for this week's class. You can also go into the "Assignments" folder and the submission area will be titled "PHP Exercise." You should be submitting a live link to me; it **must be through 000webhost**. UAlbany servers don't allow for PHP to be used and so we must utilize a hosting platform such as 000webshot. The work submitted will be evaluated based on the rubric explained below.

PHP Exercise Rubric – 10pts

- Live link submitted – 1pt
- Zip folder submitted – 1pt
- HTML validates for input.php – 1pt
- Challenge completed – 7pts total
 - Errors output correctly – 3.5pts
 - Empty inputs
 - Non-number is entered for width/height
 - Colored box output correctly 3.5pts

Extra Credit (+1pt on your final grade)

Add two more inputs for the form. These inputs will be border and message. The border input can use an input tag, but the message input should use a textarea tag. When the user clicks submit, the border provided should be applied and the message should appear between paragraph

tags inside of the div. If your extra credit is done correctly, the source code might look something like this:

```
<div style="width: 300px; height: 300px; background: blue; border: 2px solid black;"><p>Hello world!</p></div>
```

You should validate the inputs to make sure they aren't empty. To do so, you would check if they are empty with the empty() function in PHP. This code would be inside the same if statement where you checked the inputs initially.

To receive points for the extra credit, submit a live link to your page in the same area where you submit the assignment. A zip folder with all your work should still be included as well.

PHP Functions (Two Posts)

Initial Post due Thursday, March 31st at midnight

There are some basic functions that are used so often that you will find them in most programming languages. The syntax for these functions will change from language to language and sometimes the functionality will be slightly different, so it's good to know how to look these things up.

In the list below, you'll find of common PHP functions. I would like you to choose a function, do some research on it, and then describe it in your discussion post with a real-world scenario where the function might be useful. Can you think of a way you might want to use it in your final project? Some key points that you can touch on about the functions are:

- What does the function do?
- How many parameters does the function take and what are they? Are any of them optional?
- Does the function return anything, or does it modify any existing variables?
- Are there any special cases where the function does something different?
- Can you find a function that does something similar in JavaScript? Does it work any differently?
- Come up with a scenario of how this function might be used in a real-world website. Do you think it is used often?

A good place to search for the functions is the official documentation:

<https://www.php.net/manual/en/langref.php>

You can find the list of functions and an example of what the post may look like below:

- isset()
- strlen()
- trim()
- str_replace()
- include()
- require()
- preg_match()
- explode()
- is_null()
- password_verify()

- htmlentities()
- htmlspecialchars()
- unset()
- count()
- is_numeric()
- stripslashes()
- strtotime()
- usort()
- filter_var()

Initial Post Example:

The function empty() determines if a function is empty. It is considered empty if the variable isn't set or if the variable is something that evaluates as false. It takes in a single parameter, which is the variable that it evaluates, and it returns either true for an empty value or false for a non-empty value.

It returns true for obvious values like undefined variables, empty strings, and empty arrays, but also for seemingly normal values like 0, false, and "0". I wouldn't use it to check for integers that can be zero or Booleans.

There isn't an exact equivalent to this function in JavaScript you would have to check if a variable is undefined or null and then check for empty strings and arrays.

A situation where I might use this in a website would be if I had a table in a database that stored customer orders and I ran a query to get all the orders for a specific customer. I would use the empty() function to figure out if my database query returned any results. If the result is empty, I'd output a message saying that the customer has no orders, otherwise I'd output the order information. I think this function is used frequently.

Submission

The initial post is **due Thursday, March 31st at midnight**. To submit, go to Blackboard → Course Materials → Lecture Notes for this week's class. There will be a discussion area called "PHP Functions" where you can post your initial post. You can also visit the Discussion Board area directly from the Course Materials folder.

Response Post due April 3rd at midnight

Read another student's post and come up with a different scenario about how this function might be used. Do you think you might want to use this function on your final project? You can also mention if you discovered something else about the function in question. Try to respond to at least one person who chose a different function from you.

The response post is **due Sunday, April 3rd at midnight**. To submit, go to Blackboard → Course Materials → Lecture Notes for this week's class. There will be a discussion area called "PHP Functions" where you can post your response post. You can also visit the Discussion Board area directly from the Course Materials folder.

PHP Functions Rubric – 2pts

The initial post is worth 1.5pts and the response post is worth .5pt for a total of 2 points. I will be evaluating your posts based on the following criteria:

- Did you mention the specific PHP function you evaluated?

- Were specific arguments/details about the function mentioned?
- Did your response post contribute to the original post?
 - Avoid summarizing the other person's post or simply saying that it was a good post.

Next Week

Next week, we will learn more PHP and go more in depth with forms and what they can do.

There will be no reading for now so focus on completing the exercise for this week in addition to the Midterm if you haven't finished it already.