Variable Types

For this lab, you will write a PHP script that will initialize variables of different types and values, then display the results of casting each variable to another data type. You will also output the data type and value using the var_dump() method. You output will look like the results below.

Initial Type	Cast to bool	Cast to int	Cast to float	Cast to string	Variable info	Int as hex
string	1	-2	-2.6	-2.6 degrees	string(12) "-2.6 degrees"	
string	1	0	0	a dinosaur	string(10) "a dinosaur"	
boolean		0	0		bool(false)	
NULL		0	0		NULL	
double	1	-8675309	-8675309.99	-8675309.99	float(-8675309.99)	
double		0	0	0	float(0)	
string		0	0		string(0) ""	
integer	1	42	42	42	int(42)	2a
integer	1	-21	-21	-21	int(-21)	ffffffffffffffeb
array	1	1	1	Array	array(3) { [0]=> string(11) "The Shining" [1]=> string(6) "Carrie" [2]=> string(2) "IT" }	

PHP is a loosely typed language, which means when you create a variable you do not have to explicitly specify the intended data type. This does not mean the variable does not have a data type. PHP has a few methods to evaluate a variable's type and cast the variable to a different type as needed.

1. Using the HTML5 template you started with in Lab 1, create a file called **lab02.php**. For now remove the php tag you created in Lab 1 from the body**.** Remember to test your PHP scripts you will need to upload them to **csphplinux.northeaststate.edu.**
1. After your file header and before the <html> tag, use PHP to create the set of variables as shown below. Notice that none of them are cast, so the value they are set to determines their type.

```php
<?php
$a = "-2.6 degrees";
$b = "a dinosaur";
$c = false;
$d = NULL;
$e = -8675309.99;
$f = 0.0;
$g = "";
$h = 42;
$i = -21;
$j = array ("The Shining", "Carrie", "IT");
?>
```

2. After initializing the variables, create a table with the headings from the table shown earlier. Your code might look something like below.

```
<html>
  <head>
    <title>Lab 02</title>
  </head>
  <body>
    <table border='1'>
      <thead>
        <tr>
          <th style='text-align:center'>Initial Type</th>
          <th style='text-align:center'>Cast to bool</th>
          <th style='text-align:center'>Cast to int</th>
          <th style='text-align:center'>Cast to float</th>
          <th style='text-align:center'>Cast to string</th>
          <th style='text-align:center'>Variable info</th>
          <th style='text-align:center'>Int as hex</th>
        </tr>
      </thead>
```

3. Next, you will need to create a table row for each of the variables **$a** through **$j**. This means that for each variable, your script needs to output the open table row tag using echo, **<tr>**, then output a table data item for each element using the **<td>**…**</td>** tag pair. There will need to be a table data tag pair for each of the following items.
   a. For the **Initial Type** column, use **gettype()** to determine the type of each variable.
   b. For **Variable info** column, use **var_dump().** Keep in mind, most functions return a value and then you echo it. Here the var_dump function calls echo itself.
   c. For each of the columns where you cast the variable to a different type, you may either use the *val() methods such as **intval(), boolval(), strval(),** and **floatval()**, or you may simply cast the variable using the new type in parenthesis followed by the variable to cast.
   d. You are going to need two parts for the last column. First, you will need to determine if the variable is an integer. You can use **is_int()** to do that. Second, if it is an integer, you'll need to use a method like **dechex()** to do the conversion.

Remember to close all tags and to properly embed your tags.

HINT: If you make a function that receives a variable and outputs the contents of a row for that variable, it will make the code much neater!

Note that each of these methods shown above only do a conversion or assign a new value and do not actually output the resulting string so that the browser can read it. You will still need to use **echo** for the output.

The following are some resources that might be helpful in your programming.

- Evaluation with **gettype(), is_int()**, etc.: http://php.net/manual/en/language.types.intro.php
- Getting info about a variable using **var_dump()**: http://php.net/manual/en/function.var-dump.php
- Display in hexadecimal using **dechex()**: http://php.net/manual/en/function.dechex.php
- Ternary **if** for 1 line if statements: https://davidwalsh.name/php-shorthand-if-else-ternary-operators

print_r Method

Although we have not yet discussed arrays, you may have noticed that an array was included as one of the data types in this exercise. You may also have noticed the rather odd way that it generates output when using the **echo** command. Basically, when an array is cast to a string, it generates the output, "Array." (Note: If you see output on your screen that says, "Array," it's likely that you've tried to print a variable of type array.)

Although this next exercise does not help you generate useable output for your client, it may come in handy when you are trying to debug your code. PHP provides a method called **print_r()**. **print_r()** is used to output elements such as arrays in human-readable form. Let's try it.

1. In lab02.php, expand on the declaration of the array $j so that it also includes the names "Dolores Claiborne" and "The Stand"
2. After the code that generates the table, add a single line of PHP code that calls the method print_r() with $j as its argument.
3. Run the code. What did print_r() generate for output? Was it more readable than the output generated in the table?

Deliverables

1. Demonstrate the code and execution to the instructor during this lab, during office hours, or during the next lab period.
2. Upload the instructor approved .php file (source code only) to the Lab 2 D2L drop box.