Episode 3

PHP Types, Variables, and Constants

Introduction

Welcome back. In this course we are going to learn about 5 of the different PHP data types: integer, double or float, boolean, string, and array. And in this episode, we are going to explore the first three of those types. We will also learn how to store values in a variable, which in turn determine what data type those variables become, and finally, we'll learn how to define constants.

Goals

In this section of the course your goals are:

- [ ] To learn what a data type is.
- [ ] To learn how to declare an integer.
- [ ] To learn how to declare a float.
- [ ] To learn how to declare a boolean.
- [ ] To earn how to declare a constant.
- [ ] To know the difference between a variable and a constant.

Integers

In PHP an integer is a whole number like 5 or 5000. It can be positive or negative like 3 or -54, but it cannot have a decimal point. Thus 3.0 is not an integer.

Let's take a look at the following code:

```php
<?php
    $hoursInYear = 8760;
    $minutesInYear = 525600;
    $secondsInYear = 31536000;
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>PHP Data Types -- Integers</title>
</head>
<body>
    <?php
    echo "<p>There are $hoursInYear hours in a year.</p>";
    echo "<p>There are $minutesInYear minutes in a year.</p>";
    echo "<p>There are $secondsInYear seconds in a year.</p>";
    ?>
</body>
</html>
```

Let's examine this code for a moment. First, we started our file with PHP rather than HTML, and in the upper portion of our file, we have created three variables. Notice how PHP is different from most other programming languages in the fact that variables must start with a **$** dollar sign. Also, notice that we don't have to give the variable a data type. By assigning an integer to the variable, it becomes type integer. Now let's examine how we included these variables inside the HTML portion of our file. We have our variables inside the echo statements, in this case, surrounded by **double quotes**. This is important because as we will learn in the next episode variables will not "expand" when they are surrounded by single quotes.

Integers can hold pretty large values, but they do have a limit:

- On 32-bit systems, the range is between -2,147,483,648 and 2,147,483,647 or about +/- 2 billion
- On 64-bit systems the range is between -9223372036854775808 and 9223372036854775807 or about +/- 9 quintillion.

Let's quickly add the following lines to the previous code:

```php
    echo PHP_INT_SIZE;
    echo "<br>";
    echo PHP_INT_MAX;
    echo "<br>";
    echo PHP_INT_MIN;
```

The first line will let you know whether you are working on a 32-bit system (output will be 4) or a 64-bit system (output will be 8). The third and fifth lines tell you the maximum and minimum values for an integer type on your system.

Now, what happens when you enter an integer that is bigger than the maximum. Suppose that sometime in the future when you become a wildly successful PHP developer thanks to this course, your fortune is 9,223,372,036,854,775,808 dollars, (sorry 32-bit system developers, your fortune can only be 2,147,483,648). Let's see what happens when you try to assign that number to an integer type. Let's add the following code to our file:

```
$myFortune = PHP_INT_MAX + 1;
echo $myFortune;
echo "<br>";
var_dump($myFortune);
```

When we printed $myFortune to the screen this is what we got:
9.2233720368548E+18
That doesn't look like an integer, it has a decimal point, and it ends with E+18. What' going on? On the last line we used the var_dump function and what we find out is the value of the variable and its type, which is no longer an integer. It is a float. Let's talk about floats next.

Float

Floating point numbers or doubles are numbers that are written with a decimal part like 3.14 or -42.2.
Let's take a look at the following code:

```php
<?php
    // Common way to write floating point numbers.
    $monthlyRent = 1234.56;
    $bookPrice = 19.99;
    $concertTicketPrice = 119.99;
    $pi = 3.14159;

    // Floating point for Scientic Notation.
    $avogadrosNumber = 6.022e23;
    $lightKmSecond = 3E5;
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>PHP Data Types -- Floats</title>
</head>
<body>
    <?php
    echo "<p>Last month my rent was $monthlyRent. I had some money left " .
         "to buy a book for $bookPrice.  But I didn't have enough to " .
         "buy the $concertTicketPrice ticket to go to a concert.  This " .
         "will change when I become a PHP developer.</p>";
    echo "<p>Pi is equal to $pi. It is used in math.</p>";
    echo "<p>Avogadro's number is $avogadrosNumber. It is used in chemistry.</p>";
    echo "<p>Light travels $lightKmSecond kilometers per second. This is used in astronomy.</p
    ?>
</body>
</html>
```

Let's examine the code: In the upper portion of our file, we see the different ways in which a floating point number can be written. The first way, which covers the first four lines, is by far the most common way. The last two are usually used when the numbers have a large number of digits. As we did with integers, to output float variables in an echo statement, we put our variables inside double quotes.

Boolean

Boolean type is the simplest type in PHP. It accepts only two values: the PHP constants **TRUE** or **FALSE**. Constants are case insensitive, so **true** and **false** will also work, but generally, names of constants are written in uppercase to distinguish them from variables easily. Boolean variables are mainly used in control structures of which you will learn in future episodes. Let's take a look a the following code:

```php
<?php
    $isValid = TRUE;
    $isLocal = FALSE;
    $isOver = TRUE;
    $isRed = FALSE;
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>PHP Data Types -- Boolean</title>
</head>
<body>
    <?php
    var_dump($isValid);
    echo "<br>";
    var_dump($isLocal);
    echo "<br>";
    var_dump($isOver);
    echo "<br>";
    var_dump($isRed);
    echo "<br>";
    ?>
</body>
</html>
```

Do you remember earlier how your fortune changed from integer to float? Well, we can do that on purpose to convert different type values into one of the two valid boolean values. Let's see how this is done and what results we should expect:

```php
    // Converting Integer values to boolean:
    $boolValue1 = (bool) 0;
    var_dump ($boolValue1);
    $boolValue2 = (bool) -1;
    var_dump ($boolValue2);
    $boolValue3 = (bool) 4;
    var_dump ($boolValue3);

    // Converting Float values to boolean:
    $boolValue1 = (bool) 0.0;
    var_dump ($boolValue1);
    $boolValue2 = (bool) -1.2;
    var_dump ($boolValue2);
    $boolValue3 = (bool) 4.5;
    var_dump ($boolValue3);
```

We used what is called type casting to change an integer or a float into a boolean value. To do that we used **(bool) value**. And let's pay attention to the results. What happens is that only zero values, both of integer

and float type convert to the **false** boolean value, and every other value positive or negative converts to the **true** boolean value.

Constants

As their name suggests, a constant can't have its value changed. Once you define a constant its value will remain the same for the duration of the script. Here are some of the differences between constants and variables.

- Constant names do not start with a $.
- There are two ways to define a constant. Using the **define( )** function and using the **const** keyword.
- Once a constant's value is defined its value cannot be redefined or undefined.
- Constants defined using the **const** keyword are always case sensitive.
- Constants defined using the **define( )** function may be case insensitive.

Before PHP 5.3.0 a constant's value could only be set using the define( ) function. So as a developer if you work on a project that used a version of PHP older than 5.3.0 the **const** keyword won't work.

Let' take a look at the following code:

```
<?php
    define ("GREETING", "Hello, World.");
    define ("NAME", "Matthew");
    define ("TALENT", "Lego Master");
    define ("RECORD_TIME", 55.89);
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>PHP Constants</title>
</head>
<body>
    <?php
    echo GREETING, " My name is ", NAME, "<br>";
    echo "I am a ", TALENT, "<br>";
    echo "My record solving a Rubik's cube is ", RECORD_TIME, " seconds.";
    ?>
</body>
</html>
```

In this episode, we have learned about three different data types, as well as variables and constants. It is a good idea to have a consistent way of writing variables whether **lowerCaseCamel** or **name_with_underscore**. In addition, try always to use variable and constant names that are descriptive. For example, rather than using:

```
    $x = "Matthew";
    $y = "Lego Master";
    $z = 55.89;
```

Use instead:

```
    $firstName = "Matthew";
    $mastery = "Lego Master";
    $recordRubiksTime = 55.89;
```

Down the road you'll be thanking yourself for having used descriptive names rather than $x, $y, and $z.

Debugging Code

Examine the following code. Find and fix the errors.

```
<?php
    $firstName = "John";
    $lastName = "Smith";
    $age = 18;
    SchoolName = "University of New Haven";
    $major = "Computer Science";
    $gradYear == 2022;
    define ("DATE_OF_BIRTH", "2/29/1999");
    define ("SSN" 123456789);
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Debugging 2</title>
    <style>
    body {
        margin: 30px;
    }
    table, th, td {
        border: 1px solid navy;
        border-collapse: collapse;
    }
    th, td {
        padding: .5em;
    }
    caption {
        font-weight: bold;
        font-size: 1.2rem;
    }
    </style>
```

```
</head>
<body>
    <table>
    <caption>Student Information</caption>
        <tr>
            <th>First Name</th>
            <td><?php echo $firstname; ?></td>
        </tr>
        <tr>
            <th>Last Name</th>
            <td><?php echo $lastName; ?></td>
        </tr>
        <tr>
            <th>Age</th>
            <td><?php echo $Age; ?></td>
        </tr>
        <tr>
            <th>College</th>
            <td><?php echo $schoolName; ?></td>
        </tr>
        <tr>
            <th>Major</th>
            <td><?php echo $major; ?></td>
        </tr>
        <tr>
            <th>Graduation Year</th>
            <td><?php echo '$gradYear'; ?></td>
        </tr>
        <tr>
            <th>Date of Birth</th>
            <td><?php echo DATE_OF_BIRTH ?></td>
        </tr>
        <tr>
            <th>SSN</th>
            <td><?php echo SSN ?></td>
        </tr>
    </table>
</body>
</html>
```

Lab Exercises

The following exercises are meant to help you practice your coding skills. They need to be submitted for you to get a certification. As stated before coding is not a spectator sport, the more you practice (and make mistakes) the more you will learn.

1. Mark Twain once said, “When I was 18 my father knew nothing about the world. Now that I’m 25, I’m surprised to see how much the old man has learned in the past 7 years.”

- Create three variables that will hold the three integers in the quote.
- Give the variables descriptive names.
- Write a php script that will output the quote to the screen using the variables you have created.

2. Unit conversion:

- Create three variables named $miles, $pounds, and $seconds.
- Give each a float value.
- Add the following code:
  `$km = $miles * 1.609344;`
  `$kg = $pounds * 0.453592;`
  `$hours = $seconds / 3600;`
- Use echo statements to output something similar to:
  `"There are _____ kilometers in _____ miles."`
  `"There are _____ kilograms in _____ pounds."`
  `"There are _____ hours in _____ seconds."`
- Fill in the blanks with the appropiate variables.
- Change the value of $miles, $pounds and $seconds a few times and run the code.

3. Modify your script from exercise 2 by creating three constants to hold the values 1.609344, 0.453592, and 3600. Use descriptive names for the constants. (Hint: KM_PER_MILE).

In the next episode, we will learn about strings. See you then.