asgn11 - outdated irish length measurements and refactoring

objectives

- Refactor code for better code modularity
- Create a git branch
- Merge code in git
- Push code to GitHub

introduction

We will continue working with measurements and learn about the importance of refactoring code.

Watch **Chapter 2** from the LinkedIn Learning video by Kevin Skoglund titled Easy PHP Projects: Measurement Conversion. Pay close attention to the the two subheadings

1. Refactor Length Conversions
2. Refactor Area Conversion

Another important concept in the videos is creating an array as a PHP constant. Constants are similar to variables in that they hold values, however, the difference is that you cannot change the value stored in a constant.

As your programs start to get larger you will need a way to manage all of the files and organize them in a logical manner. There are many ways to do this. Better code organization will lead to applications that are more robust (users don't break them as easily) and they are easier to maintain. At first, the code reorganization seems like overkill, but as your programs continue to increase in size this becomes an important concept.

outdated irish length measurements

Let's add a new group of measurements to our program. We will focus on mostly outdated Irish units of length.

Use the **centimeter** as the standard unit of measurement for this exercise. I have used the English translation for the **Length Measurements** and included the **Relative Value**. The **Relative Value** is helpful but not necessary for the assignment. The **Relative Measurement** uses the **foot** as the standard unit. Therefor 1/12 of a thumb-length = 1 foot. It helped me wrap my head around the relationship between the **Length Measurements**.

outdated irish lengths table

Length Measurement	Centimeters	Relative (foot)
1 grain	.7	1/36
1 thumb-length	2.1	1/12
1 palm	3.3	1/3
1 fist	10.4	5/12
1 foot	25.0	1
1 step	62.5	2 1/2
1 double-step	1500	6
1 rod	3000	12

Here is the link where I gathered the data from Wikipedia. If you want to read up on this stuff.

getting started

- Add a new folder named `web182/asgn11` .
- Copy you're code from `asgn10` and paste it into your `asgn11` folder.
- Create a new file named `outdated-irish-length-measurements.php` in `asgn11` .
- Update the `asgn11/index.php` file so it contains a link to the new folder `outdated-irish-length-measurements.php` .
- NOTE: We are not going to change anything in `outdated-irish-length-measurements.php` to begin with, instead we will first refactor our code, *then* we will create the measurements. This is designed to show how a more modular code structure makes it easier to develop.

git

Depending how you copied your code from `asgn10` to `asgn11` there may be a hidden folder named `.git` that came along with it. If it exists, then delete it. You can do this in the file manager by turning on **hidden files** or use the terminal. Here are the instructions for the terminal

Start by opening your BASH (or PowerShell) or Terminal (Mac) and navigate to your folder. Here are some examples, your files may be stored somewhere similar.

windows

```
cd c:/xampp/htdocs/web182/asgn11
```

mac

```
cd /Applications/MAMP/htdocs/web182/asgn11
```

removing the .git folder

See if the folder exists by listing all of the files and their attributes

```
ls -la
```

To delete it, you need to recursively (delete all of the files and folders inside of .git) with the following command. The `-rf` if for recursive and force.

```
rm -rf .git
```

Initialize git

The easiest way to manage your programs is to create a repository for each project. First you will need to initialize git for `asgn11`.

```
git init
```

stage and commit the files

```
git add .
git commit -m "Init commit"
```

add a branch

Until now, we have been working on the **Master** branch. That is fine while getting started with git, however, git branches offer an extremely flexible way to code. It will give you more comfort in making mistakes and it creates organization.

In short, you create a branch off of the **Master** branch, work on it until you are finished, then merge it with **Master**. If something goes horribly wrong while you are on the branch and you can't fix your code -- no problem -- just delete the branch and create a new one. Your original code on the master branch remains in its original state. ``Here is a link to git branching from Atlassian. Read thought the content from Atlassian. It is important to understand this concept.

Create the branch and checkout

First you will create the branch.

```
git branch outdated-irish-length-measurements
```

Run `git status` and notice that the branch exists but you are still on the **Master**. We need to use the `checkout` command to move to the branch.

Hint: you can use the **tab** key to 'tab out' what you are typing. To do so start typing `git out` and press the **tab** key. It should fill in the remainder of the word for you.

```
git checkout outdated-irish-length-measurements
```

Use `git status` again and notice that you have changed to the new branch.

Combine the two commands (a quicker way)

It can get a little tiresome to type both commands (but it's not a bad practice). Another way you can do it is

```
git checkout -b someBranchName
```

This will create a new branch and checkout to it.

refactor your liquid measurements

Follow along with the refactoring section in **Chapter 2** from the *Easy PHP Projects* video and refactor your code. Here are the points you need to follow.

- Move your functions to `includes/functions.php` .
- Update `liquids.php` and use the `include_once` function to reference the moved files.
- Test your code.
- Move your conversion factors into an array. The array should be declared as a constant. Name the array, `LIQUID_MEASUREMENT_TO_IMPERIAL_GALLONS` . Here is the documentation on using constants in PHP.
- I was unable to use the `const` keyword, so I ended up using the following syntax from tutorials point.

In addition, I was unable (without some unnecessary extra code) to get the `if(array_key_isset)` function to work without it throwing an error. You may eliminate that unless you find an easy fix. I think it may be that I am using PHP version 7.2 and not 5.3.

More refactoring (this stuff never ends until the deadline)

This will look familiar to you since we have already done something similar earlier in the term.

Add a copyright to create a footer

Before we go much further, let's add a copyright to our code. Open your index.php page and add the following

```
<footer>
    <small>&copy; Copyright 2019, Conversions!</small>
</footer>
```

DRY (Don't Repeat Yourself) Code

Read through the code and look for areas that repeat. We have two choices

1. Create functions
2. Make included files

The decision is often up to the developer. A general rule of thumb is that you should use included files when it is just a chunk of code that you are going to repeat and use functions when you need to perform an operation.

We can find repetitive HTML code in the `index.php`, `archaic-liquid-measurements.php`, `outdated-irish-measurements.php` files. All of them have header and footer code.

Create a new file in includes named `header.php`. Even though the code we are going to add to the file is only HTML, I like to add the .php to the file incase I need to add some php code later.

Remove the following code from index.php and put it includes/header.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Archaic Conversions</title>
    <link href="styles.css" rel="stylesheet" type="text/css">
  </head>
  <body>
```

We will fix the problem with `<title>` later (not all files will have the same title).

Create a new file named `includes/footer.php`.

Cut the footer code you just created and place it in `includes/footer`.

Now we need to reference the code using `include_once`. Place the following line to reference `includes/header.php`.

```
<?php include_once 'includes/header.php'; ?>
```

Do the same for includes/footer.php by placing the following line where you just cut your code.

```
<?php include_once 'includes/footer.php'; ?>
```

Now do the same to the `archaic-liquid-measurements.php` and `outdated-irish-length-measurements.php`.

Fixing the title

The `<title>` tag needs to be fixed. Right now all of the pages are named Conversions. We can use a variable to pass a value to the `includes/header.php` file. In your `includes/header.php` file, replace the current `<title>` tag with

```
<title><?php echo $pageTitle; ?></title>
```

Short PHP tag to echo a variable

In some cases such as this we can shorten `<?php echo $pageTitle; ?>` by using a php short tag. The following line means echo out a variable. It is a standard practice when using PHP. It also looks cleaner. You can find a reference to it in the PHP Best Practices.

```
<?= $pageTitle; ?>
```

Update `archaic-liquid-measurements.php` and `outdated-irish-length-measurements.php` so they can take advantage of the `<title>` tag variable.

Test your code.

add outdated irish length measurements

Now that you have refactored some code (there is still more we can do!), it is time to add our next measurement unit. You already have a working template.

Pay close attention to your naming conventions. Remember that we are using camelCase.

git status, stage and commit

Now that you have completed a section of the assignment, it is a good time to commit.

Wash, rinse, and repeat. Run the commands from above but change your commit message to reflect that you have successfully completed the **outdated Irish length measurements**.

Merge your branch into master

Now that your program is ready for deployment, it is time to merge with the **Master** branch. Here is a link from Atlassian on merging for further reading. Otherwise, here is the short version

It is a good practice to check your status before merging. Make sure you have committed on your current branch

before taking the next steps. Actually, git will not let you checkout until you commit. It's a nice safety feature.

There are a couple of ways to merge. For this assignment, first **checkout** to the **Master** branch.

```
git checkout master
```

Check the status so you can see what branch you are on

```
git status
```

This should show that you are on the master branch.

Now you can merge from the branch you have been working on

```
git merge outdated-irish-length-measurements
```

Your output in the terminal will look something like

```
Updating d71403b..1d62975
Fast-forward
 holding.txt                           | 66 ++++++++++++++++++++++++++++++++++++++++++++
 includes/functions.php                | 56 ++++++++++++++++++++++++++++++++++++++++++++
 index.php                             | 30 ++++++++++++++++++++-----------
 liquids.php                           | 77 ++++++++++++++++++++++++++++++++++++++++++++
 outdated-irish-length-measurements.php | 80 ++++++++++++++++++++++++++++++++++++++++++++
 styles.css                            | 85 ++++++++++++++++++++++++++++++++++++++++++++
 6 files changed, 383 insertions(+), 11 deletions(-)
 create mode 100644 holding.txt
 create mode 100644 includes/functions.php
 mode change 100644 => 100755 index.php
 create mode 100755 liquids.php
 create mode 100644 outdated-irish-length-measurements.php
 create mode 100755 styles.css
```

Create an online repository

You have been working locally up till this point. Now we need to create a place to push our code. While at github.com

This is the same process that we have completed for the last couple of assignments.

Head out to GitHub and log in.

- Click on the green **New** button to create a new repository (often called 'repo').
- Type `asgn11` as the repository name.

- Type `School assignment on conversions based on the LinkedIn Learning tutorial by Kevin Skolund.` in the Description.
- Click on `Create Repository`.

This will bring up the following under the section titled **…or push an existing repository from the command line**. We will use this because we have been working locally and have already done the other steps in the section titled **…or create a new repository on the command line**. Your commands will have your name instead of mine.

```
git remote add origin https://github.com/charliekwallin/asgn11.git
```

Copy and paste the first line in BASH (Windows) or Terminal (Mac). Now your repo is set up and you can push your code to GitHub by copying and pasting.

```
git push -u origin master

Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 3.11 KiB | 3.11 MiB/s, done.
Total 9 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/charliekwallin/asgn09.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Check it out by going to your GitHub account and viewing your repository.

update your website

Update your website and remember to add a link to `asgn11` on your `web182/index.php` page.

Test your code to make sure it runs as expected.

submit your work

- Make sure your current code is in GitHub
- Put your GitHub address for this assignment in the Comments section of Moodle.
- Make sure you have updated your `web182/index.php` file and your `web182/asgn11/index.php` file