Linting WordPress PHP files

While I've been doing WordPress theme development I've also been working on getting PHPCS (PHP Code Sniffer) to work properly on my projects.

I think I've finally got it. As usual, I was making things too hard on myself and it ended up being far easier to set up and use.

The first tool we need is Composer to install and manage dependencies. Think of Composer as the PHP version of NPM.

Since I'm working on a Mac, I chose to install it with Homebrew using the command below:

```
brew install composer
```

The next step is to install PHPCS and the WordPress Coding Standards globally for the current user using Composer.

```
composer global require squizlabs/php_codesniffer wp-coding-standards/wpcs
```

Installing the Coding Standards doesn't make them available to PHPCS. To make them available we need to add them to the version of PHPCS we just installed.

```
phpcs --config-set installed_paths \
~/.composer/vendor/wp-coding-standards/wpcs
```

If you get an error saying that phpcs was not found, it is likely that the directory where you installed PHPCS is not on your path. Try the following command:

```
~/.composer/vendor/bin/phpcs \
--config-set installed_paths \
~/.composer/vendor/wp-coding-standards/wpcs
```

The last step is to validate that we successfully installed the WordPress Coding Standards. Running the command below will produce a list of all the coding

standards available for the current PHPCS installation.

```
phpcs -i
```

If the WordPress Coding Standards were successfully installed, the command should return:

```
The installed coding standards are PEAR, Zend, PSR2,
MySource, Squiz, PSR1, PSR12, WordPress,
WordPress-Extra, WordPress-Docs and WordPress-Core
```

If it doesn't you need to review the prior steps and, possibly, reinstall the WordPress Coding Standards.

Configuration

PHPCS uses XML configuration files to tell the tools how to lint the files. This may not be what most people prefer but it's a one time only configuration that can be reused for different projects

```
<?xml version="1.0"?>
<ruleset name="Rivend<ruleset name="Rivendellweb">endellweb Wordpress Cod

    <file>.</file> </description><!-- 1 -->e="extensions" value="php"></a

  </arg><!-- 2 -->olors"></arg> <!-- 3 -->

    <arg</arg><!-- 3 -->g> <!-- 4 -->

    <rule ref=<!-- 4 --><!-- 4 -->!-- 5 -->
      <exclude name="WordPress.PHP.DisallowShortTernary"></exclude> <!-- 6
    </rule>

    <rule re<exclude name="WordPress.PHP.DisallowShortTernary"><!-- 6 -->

    <config name="minim</rule><!-- 8 -->ersion" value="4.9"></config> <!--
```

```
</rulese</config><!-- 8 -->

    <config name="minimum_supported_wp_version" value="4.9"></config> <!--
</ruleset>
```

The example configuration file performs the following tasks that match the comments in the file above:

1. What files we want to scan
2. Scan only files with a PHP extension
3. Pass an argument to include colors in the console report
4. Show sniff codes in all reports. The default is to hide them
5. Tell PHPCS what rules you want to use
6. Include specific rules you want to ignore as children of the rule element
7. You can include additional rules packages
8. You can also include specific rules from a package without using the full package
9. Tell PHPCS the minimal version that we want to work with

You can see a more detailed configuration file in the Code Sniffer repo's Sample PHPCS configuration file