Contents

Preface

Building code is your proficiency exercise routine. Code challenges exercise your problem solving, programmatic thought processes, and quality chops.

My intent in writing both the book and workbook is to help you fine-tune your refactoring proficiencies. We all refactor. It is a natural process in programming. No one and no codebase is immune to improvement.

Your goal is to constantly improve your code building processes and codebase. As your craftmanship skills advance, you will steadily build-in quality. The effect directly and positively impacts your bottom line, profitability, and competitive advantage. You are reducing your costs and time as well as making your code more reusable, readable, and maintainable.

Practice hones your skills. Period. It doesn't matter if you are an athlete, musician, engineer, programmer, plumber, or carpenter. Skills are not learned through absorption or osmosis.

Programming requires practice. Osmosis and memorization will *not* make you a better developer.

This workbook fine-tunes your refactoring skills. You will gain real, practice programming skills that you can use in your projects...right now.

And the solutions are a cable from my head to yours.

Let's get to work.

Listen to me. You need to exercise knowledge. In doing so, you actively build the mental muscles necessary to advance yourself in this profession.

This Code Challenge Workbook intends to stretch your programming chops. Each challenge presents you with the opportunity to *practice* refactoring. As you actively work on these challenges, you improve and build your programming proficiencies.

My goals are to help you gain and fine-tune *real, practical skills* that **you can use right now in your projects**.

Who Should Read This Workbook?

This book is for anyone who wants to build better code.

Do you build software solutions? Then code refactoring is a necessary skill that propels you forward. There is a significant difference between code that works right now and quality code that you can maintain and reuse for years.

Regardless of your title or years of experience, refactoring is a continuous learning process.

If you build solutions in code, then this Code Challenges Workbook and the Refactoring Tweaks book are for you.

How to Read This Book

The Code Challenge Workbook includes two distinct sections: Code Challenges and

Solutions.

Code Challenges Section

The Code Challenges section is where you get to practice. Each challenge gives you a snippet of code. Each snippet has multiple quality improvement opportunities. Your task is to do the refactoring process:

1. Review the code
2. Identify each of the problematic areas
3. Determine why it could be better
4. Then rewrite the code.

Try to do these on your own without flipping to the Solutions section.

Remember, osmosis and memorization will not build your programming skills. Programming is a thinking profession first and then an active building profession second. You can't cheat mastery or craftsmanship. There are no shortcuts. It takes work and practice.

Solutions Section

The Solutions section is where you and I walk through the refactoring process together. Step-by-step, bit-by-bit, we work together. This section provides you with the why and then the how. It includes extra tips and insights within the context of what we are refactoring.

Call-out Key

Throughout the workbook, I provide asides and call-outs to share additional information or to emphasize an important point. This key will help you to quickly identify the intent.

Master Tip

The bullseye is a master tip or insight that I'm sharing with you. Each of these will give more of *the why* to further solidify your understanding and implementation.

Additional Information

The coffee cup provides you with additional information, such as a definition or clarification. For example, if I'm teaching you about the context of post type, I may explain why it exists and how it's used to classify content.

Question or Thought Experiment

The question circle challenges you to think and consider *why*. These are meant to inspire you to rethink how you do stuff.

Doing It Wrong

The rocket ship plummeting towards the ground means "Whoopsie, you're doing it wrong". At times, you may think of going in a particular direction, but that direction has problems. This call-out helps you see why that direction will cause you woes.

Credit Time

The ringing bell is to give credit to someone who has contributed to this book, the way I think, or this profession.

Code Examples

Throughout the book, code examples are provided. Refactoring is language and platform independent. In other words, these strategies you will learn work for PHP, JavaScript, Perl, Python, C#, Visual Basic, etc. Refactoring is a process. It doesn't care what high level language you use to express your code.

The code examples are real code snippets taken from various codebases. To keep it concise, a `..` is used as a placeholder for code that has been removed for brevity.

Terminology

WordPress has two different meanings for the term `post`:

- All content that is stored in the database table `wp_posts` is considered a `post`.
- Each `post` gets a more descriptive classification called a post type. Post types can be post, page, revision, attachment, navigation menu, or custom post type.

Ah confusing, I know. To avoid confusion, let's change the first definition and term it `content`. Therefore, the `content` is a record out of the `wp_posts` database table and can be any post type.

Code Challenges

Doing. Building. These activities fine-tune your programming chops.

It Works

Do you want to step out and maintain it?

Quality

Reusable and maintainable

Let's put your new found skills and knowledge to work. These refactoring code challenges are real code snippets that are from various codebases. Using the Refactoring Tweaks book, do the following tasks:

1. Identify each of refactoring tweak opportunity.
2. Note why you think each could be improved.
3. Then refactor each one.
4. Compare to the solutions provided.

Have fun!

Code Challenge 1

Roll up your sleeves and see how many refactoring tweak opportunties you can find.

```
<?php

/*
 * Check if custom post type exists, to provide data
 */
function prefix_is_post_type($type) {
    global $wp_query;

    if($type == get_post_type($wp_query->post->ID)) return true;
    return false;
}
```

Code Challenge 2

Look hard. There are several of them in this one function.

```
<?php

/* This function grabs the custom header from the current theme so that it can be
displayed. */
function prefix_get_header_image() {
  $theme_slug = prefix_actual_current_theme();
  $mods       = get_option( “theme_mods_{$theme_slug}” );

  if ( isset( $mods[‘header_image’] ) &&
       ‘remove-header’ != $mods[‘header_image’] &&
       ‘random-default-image’ != $mods[‘header_image’] &&
       ‘random-uploaded-image’ != $mods[‘header_image’] ) {
       return $mods[‘header_image’];
  }

  return false;
}
```

Code Challenge 3

Don't be intimidated by the lines of code in this one function. That's a clue.

You can access the code challenge raw code on GitHub.

```
<?php

class Form {
  ..
  /**
   * Render a repeatable group row
   * @since  1.0.2
   * @param  Field $field_group  Field group field object
   * @param  string  $remove_disabled Attribute string to disable the remove button
   */
  public function render_group_row( $field_group, $remove_disabled ) {

       $field_group->peform_param_callback( 'before_group_row' );
       $closed_class = $field_group->options( 'closed' ) ? ' closed' : '';

       echo '
       <div class="postbox plugin-row plugin-repeatable-grouping', $closed_class,
'" data-iterator="', $field_group->index, '">';

       if ( $field_group->args( 'repeatable' ) ) {
            echo '<button type="button" ', $remove_disabled, 'data-selector="',
$field_group->id(), '_repeat" class="dashicons-before dashicons-no-alt
plugin-remove-group-row"></button>';
       }

       echo '
            <div class="pluginhandle" title="' , esc_attr__( 'Click to toggle',
'plugin' ), '"><br></div>
            <h3 class="plugin-group-title pluginhandle-title"><span>', $field_group-
>replace_hash( $field_group->options( 'group_title' ) ), '</span></h3>

            <div class="inside plugin-td plugin-nested plugin-field-list">';
       // Loop and render repeatable group fields
       foreach ( array_values( $field_group->args( 'fields' ) ) as $field_args ) {
            if ( 'hidden' == $field_args['type'] ) {

                 // Save rendering for after the metabox
                 $this->add_hidden_field( $field_args, $field_group );

            } else {

                 $field_args['show_names'] = $field_group->args( 'show_names' );
                 $field_args['context']    = $field_group->args( 'context' );

                 $field = $this->get_field( $field_args, $field_group )->render_field();
            }
       }
```

```
        if ( $field_group->args( ‘repeatable’ ) ) {
            echo ‘
                    <div class=”plugin-row plugin-remove-field-row”>
                        <div class=”plugin-remove-row”>
                            <button type=”button” ‘, $remove_disabled, ‘data-
selector=”’, $field_group->id(), ‘_repeat” class=”button plugin-remove-group-row
alignright”>’, $field_group->options( ‘remove_button’ ), ‘</button>
                        </div>
                    </div>
                    ‘;
        }
        echo ‘
            </div>
        </div>
        ‘;

        $field_group->peform_param_callback( ‘after_group_row’ );
  }
    ..
}
```

Code Challenge 7

Don't worry. More fun is coming.

You can find the following code on GitHub in this gist.

```
window.PluginName = (function(window, document, $, undefined){
  'use strict';

  var plugin = {
       ..
  }

  ..

  plugin.toggleCheckBoxes = function( event ) {
       event.preventDefault();
       var $this = $( this );
       var $multicheck = $this.closest( '.plugin-td' ).find(
'input[type=checkbox]:not([disabled])' );

       // If the button has already been clicked once...
       if ( $this.data( 'checked' ) ) {
            // clear the checkboxes and remove the flag
            $multicheck.prop( 'checked', false );
            $this.data( 'checked', false );
       }
       // Otherwise mark the checkboxes and add a flag
       else {
            $multicheck.prop( 'checked', true );
            $this.data( 'checked', true );
       }
  };
  ..
  return plugin;

})(window, document, jQuery);
```