CREATING A PLUGIN

WHY PLUGINS?

- Separate your own code from WordPress core code
- If something goes wrong with your plugin, the rest of the site will generally continue to function.

WHAT DO YOU NEED

- Text Editor (VS Code)
- Access to your hosting server files (localhost will work for now)
- A working WordPress installation (let's go back to sait-wordpress)

WHAT ARE WORDPRESS PLUGINS?

- Standalone set of code that enhances and extends the functionality of WordPress
- Add new features to any part of your website, including the Admin Control Panel
- Modify the default behavior of WordPress or remove unwanted behavior completely
- Customize and personalize WordPress to fit your needs

HOOKS

- WordPress plugins interact with core code using **hooks**. There are 2 different types:
  - Action hooks (to add/remove functions)
  - Filter hooks (to modify data that is produced by functions)

ACTIONS AND ACTION HOOKS

- When you visit any page of a WordPress website, a series of PHP functions (**actions**) are called at various points, and they are attached to **action hooks**.
- You can use **action hooks** to add your own functions to the list of actions that run when that hook is called.
- You can also remove pre-existing functions from any action hook.

Reference

ADDING FUNCTIONS TO ACTION HOOKS

```
add_action( actionHook, customFunction );
```

1. Name of the action hook that you want to attach to
2. Name of the function that you want to run

EXAMPLE CODE

- Let's create a plugin to display text after the footer of every page.
- This plugin will use the `wp_footer()` action hook. This hook is called right before the closing `</body>` tag of ever page.
- We'll create a custom function name sait_Add_Text. You may use the initials of your plugin name or something else unique.

PLUGIN CODE

```php
<?php
/*
Plugin Name: Add Text To Footer
*/

// Hook the 'wp_footer' action hook, add the function named 'sait_Add_Text' to it
add_action("wp_footer", "sait_Add_Text");

// Define 'sait_Add_Text'
function sait_Add_Text()
{
    echo "<p style='color: black;'>After the footer is loaded, my text is added!
</p>";
}
```

ACTIVITY: CREATE A PLUGIN

1. Create a sait-add-text.php file in VS Code.
2. Save this file in `wp-content/plugins/` folder.
3. Add the previous code to this file.
4. Go to your WordPress dashboard and activate this plugin.
5. View your homepage. You should see some extra text at the bottom of your page now.

REMOVING FUNCTIONS

- To remove an action from an action hook, you must write a new function that calls `remove_action()`, then call the function you have written using `add_action()`.
- We must remove the function before it is called or else it won't be removed.

```
// Hook the 'init' action, which is called after WordPress is finished loading the
core code, add the function 'remove_My_Meta_Tags'
add_action( 'init', 'remove_My_Meta_Tags' );

// Remove the 'add_My_Meta_Tags' function from the wp_head action hook
function remove_My_Meta_Tags()
{
    remove_action( 'wp_head', 'add_My_Meta_Tags');
}
```

REMOVE TEXT ON TUESDAYS

```
// Hook the 'wp_head' action, run the function named 'sait_Remove_Text()'
add_action("wp_head", "sait_Remove_Text");

// Define the function named 'sait_Remove_Text()' to remove our previous function
from the 'wp_footer' action
function sait_Remove_Text() {
    if (date("l") === "Tuesday") {
        // Target the 'wp_footer' action, remove the 'sait_Add_Text' function from it
        remove_action("wp_footer", "sait_Add_Text");
    }
}
```

ACTIVITY: USE REMOVE_ACTION() IN YOUR PLUGIN

1. Go back to your sait-add-text plugin.
2. Add the code from the previous page.
3. Refresh!

FILTERS AND FILTER HOOKS

- A filter function allows you to modify the resulting data that is returned by existing functions and must be hooked to one of the filter hooks.
- Filter hooks are different from action hooks but they are also called at various points in the script.

ADDING FILTERS USING ADD_FILTER()

- To add a filter function to any filter hook, your plugin must call the WordPress function named `add_filter()` with at least 2 parameters

```
add_filter(filterHook, customFunction);
```

ALTER THE EXERPT OF A POST

```php
<?php
/*
Plugin Name: Add Excerpt
*/

// Hook the get_the_excerpt filter hook, run the function named
sait_Add_Text_To_Excerpt
add_filter("get_the_excerpt", "sait_Add_Text_To_Excerpt");

// Take the excerpt, add some text before it, and return the new excerpt
function sait_Add_Text_To_Excert ($old_Excerpt) {
    $new_Excerpt = "<b>Excerpt: </b>" . $old_Excerpt;
    return $new_Excerpt;
}
```

ACTIVITY: CUSTOMIZE YOUR EXCERPT

1. Create a new file called sait-add-text-to-excerpt.php and save it in `wp-content/plugins/` folder.
2. Add the code from the previous slide to this file.
3. Go to your WordPress dashboard and activate your new plugin.
4. Refresh your homepage to see the new excerpts.

REMOVING FILTERS

- Removing filters is much simpler than removing an action because you can call the `remove_filter()` function without defining a new function.
- Remove the filter only after adding it.

```
remove_filter(filterHook, customFunction);
```

ACTIVITY: REMOVING EXCERPT TEXT ON TUESDAYS

1. Go back to your sait-add-text-to-excerpt plugin.
2. Add an if statement to remove `sait_Add_Text_To_Excerpt` if the day is Tuesday.

PLUGIN BEST PRACTICES

- Create a folder in `/wp-content/plugins/` folder with a unique and descriptive name for your plugin to ensure it doesn't clash with another plugin.
- Add header information to your plugin that can be used by WordPress to give users more information.

```
/*
Plugin Name: SAIT First Plugin
Description: Example Plugin created in SAIT's WordPress class. It makes a new admin
menu link!
Author: Your Name
*/
```

ACTIVITY: RESTRUCTURE YOUR PLUGINS

1. Create new folders for your plugins and move them into those folders.
2. Add header information to your plugin files.