Top-Level Menus

- Add a Top-Level Menu
  - Example
  - Using a PHP File for HTML
- Remove a Top-Level Menu
  - Example
- Submitting forms
  - Form action attribute
  - Processing the form

Add a Top-Level Menu

To add a new Top-level menu to WordPress Administration, use the `add_menu_page()` function.

```
add_menu_page(
  string $page_title,
  string $menu_title,
  string $capability,
  string $menu_slug,
  callable $function = '',
  string $icon_url = '',
  int $position = null
);
```

Example

Lets say we want to add a new Top-level menu called "WPOrg".

The first step will be creating a function which will output the HTML. In this function we will perform the necessary security checks and render the options we've registered using the Settings API.

[info]We recommend wrapping your HTML using a `<div>` with a class of `wrap`.[/info]

```
function wporg_options_page_html() {
?>
  <div class="wrap">
    <h1><?php echo esc_html( get_admin_page_title() ); ?></h1>
    <form action="options.php" method="post">
      <?php
      // output security fields for the registered setting "wporg_options"
      settings_fields( 'wporg_options' );
      // output setting sections and their fields
      // (sections are registered for "wporg", each field is registered to a specific section)
      do_settings_sections( 'wporg' );
      // output save settings button
      submit_button( __( 'Save Settings', 'textdomain' ) );
      ?>
    </form>
  </div>
<?php
}
```

The second step will be registering our WPOrg menu. The registration needs to occur during the `admin_menu` action hook.

```
add_action( 'admin_menu', 'wporg_options_page' );
function wporg_options_page() {
  add_menu_page(
    'WPOrg',
    'WPOrg Options',
    'manage_options',
    'wporg',
    'wporg_options_page_html',
    plugin_dir_url(__FILE__) . 'images/icon_wporg.png',
    20
  );
}
```

For a list of parameters and what each do please see the `add_menu_page()` in the reference.

Using a PHP File for HTML

The best practice for portable code would be to create a Callback that requires/includes your PHP file.

For the sake of completeness and helping you understand legacy code, we will show another way: passing a `PHP file path` as the `$menu_slug` parameter with an `null` `$function` parameter.

```
add_action( 'admin_menu', 'wporg_options_page' );
function wporg_options_page() {
  add_menu_page(
    'WPOrg',
    'WPOrg Options',
    'manage_options',
    plugin_dir_path(__FILE__) . 'admin/view.php',
    null,
    plugin_dir_url(__FILE__) . 'images/icon_wporg.png',
    20
  );
}
```

Remove a Top-Level Menu

To remove a registered menu from WordPress Administration, use the `remove_menu_page()` function.

```
remove_menu_page(
  string $menu_slug
);
```

[warning]Removing menus won't prevent users accessing them directly. This should never be used as a way to restrict user capabilities.[/warning]

Example

Lets say we want to remove the "Tools" menu from.

```
add_action( 'admin_menu', 'wporg_remove_options_page', 99 );
function wporg_remove_options_page() {
  remove_menu_page( 'tools.php' );
}
```

Make sure that the menu have been registered with the `admin_menu` hook before attempting to remove, specify a higher priority number for `add_action()`.

Submitting forms

To process the submissions of forms on options pages, you will need two things:

1. Use the URL of the page as the `action` attribute of the form.
2. Add a hook with the slug, returned by `add_menu_page`.

[info]You only need to follow those steps if you are manually creating forms in the back-end. The Settings API is the recommended way to do this.[/info]

Form action attribute

Use the `$menu_slug` parameter of the options page as the first parameter of `menu_page_url()`. By the function will automatically escape URL and echo it by default, so you can directly use it within the `<form>` tag:

```
<form action="<?php menu_page_url( 'wporg' ) ?>" method="post">
```

Processing the form

The `$function` you specify while adding the page will only be called once it is time to display the page, which makes it inappropriate if you need to send headers (ex. redirects) back to the browser.

`add_menu_page` returns a `$hookname`, and WordPress triggers the `"load-$hookname"` action before any HTML output. You can use this to assign a function, which could process the form.

[info]`"load-$hookname"` will be executed every time before an options page will be displayed, even when the form is not being submitted.[/info]

With the return parameter and action in mind, the example from above would like this:

```
add_action( 'admin_menu', 'wporg_options_page' );
function wporg_options_page() {
  $hookname = add_menu_page(
    'WPOrg',
    'WPOrg Options',
    'manage_options',
    'wporg',
    'wporg_options_page_html',
    plugin_dir_url(__FILE__) . 'images/icon_wporg.png',
    20
  );

  add_action( 'load-' . $hookname, 'wporg_options_page_submit' );
}
```

You can program `wporg_options_page_submit` according to your needs, but keep in mind that you must manually perform all necessary checks, including:

1. Whether the form is being submitted (`'POST' === $_SERVER['REQUEST_METHOD']`).
2. CSRF verification
3. Validation
4. Sanitization

□□□ #2
□ □□□ □□□ 2026-10-26 15:37:07 UTC
□ □□□ □□□ 2026-10-26 15:37:07 UTC