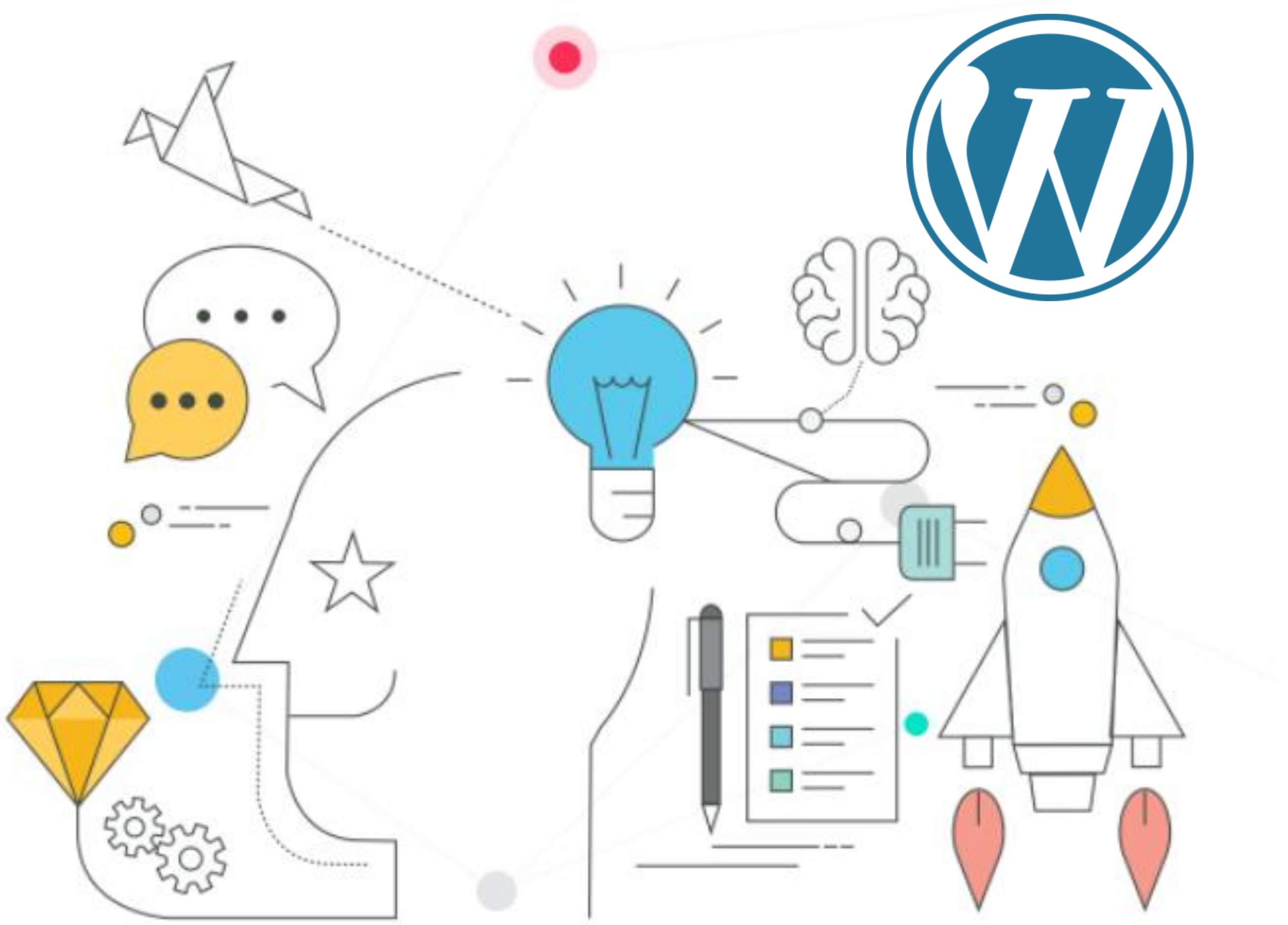
WordPress Plugin Boilerplate – wppb.me



WordPress Plugin Boilerplate – wppb.me

- A video series on scaffolding out a plugin using the WP Plugin Boilerplate.
- Base theme is underscores with some Tailwind CSS via CDN.
- Theme and Plugin code available in available repo.
- A basic Events Plugin using MySQL table, REST API, wp_nonce and wp_localize_script as well as scaffolding for React, Vue and JS apps.

With thanks for helping me learn WPPB to:

1. Online Web Tutor: Boilerplate WordPress Plugin Development on YouTube.

2. Rao Abid Ali: https://www.udemy.com/course/wordpress-plugin-development-using-boilerplate/

And of course Devin Vinson who maintains this awesome project!

Resources

- GitHub repo for this series:
- wppb.me for generator.
- wppb.io
- https://www.toptal.com/wordpress/ultimate-guide-building-wordpress-plugin
- https://www.udemy.com/course/wordpress-plugin-development-using-boilerplate/
- OWT YouTube series - https://www.youtube.com/watch?v=ZTrekTzoc_c&list=PLT9miexWCpPUC5jFln66z-8uemSkxMxCa
- WPPB on GitHub - https://github.com/DevinVinson/WordPress-Plugin-Boilerplate

Features covered (1)

- Install, activation, deactivation and uninstall.
- How to wire in hooks and define constants.
- Creating MySQL tables.
- Create custom CRUD REST APIs.
- Use of wp_localize_script to pass PHP variables to JS files.
- Create Admin menu and pages.
- Create front end template in plugin to use with page created on install, so that the plugin is self contained for all pages.
- Using wp_nonce to add security to forms.

Features covered (2)

- We will create a basic events calendar plugin that stores an event in a custom MySQL table.
- We will have an admin page that has templates for CRUD on MySQL
- In the front end, we will programmatically create an page so that in our plugin we can use a filter on page_template so that we can use template files from our plugin.
- This will enable the plugin to be self contained.
- As a bonus, the front end will have a page that is wired to use React.js by using @wordpress/scripts.

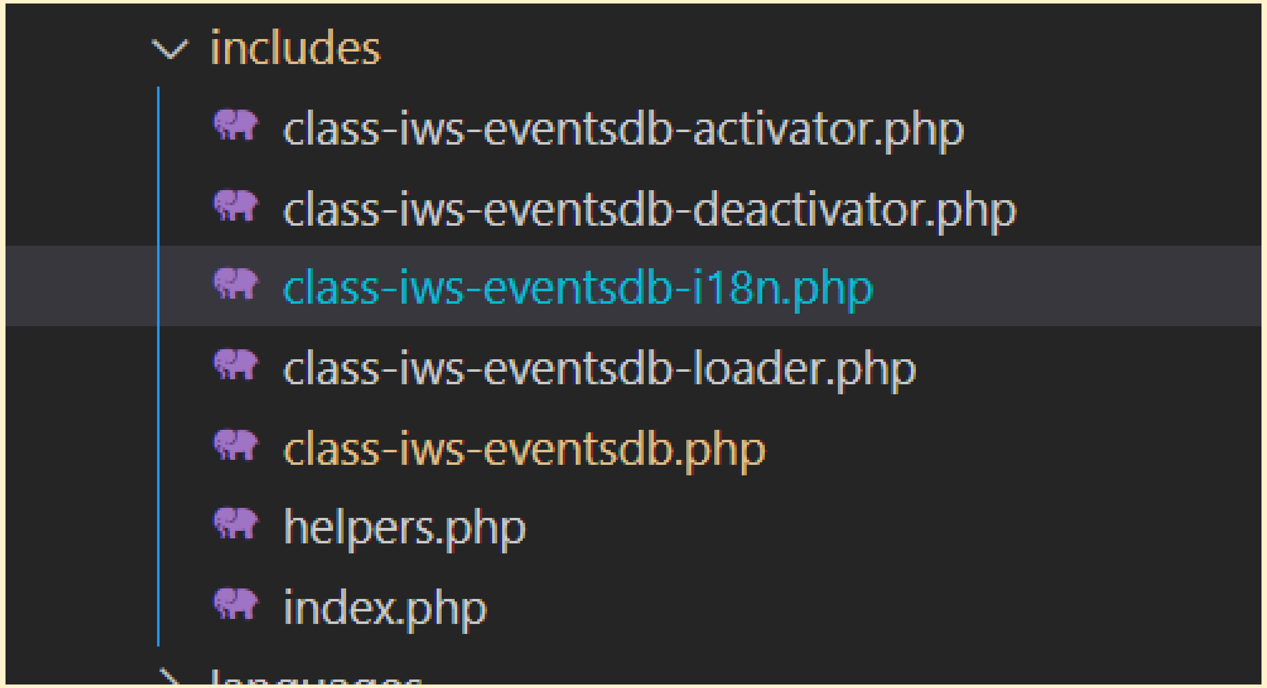
Features covered (3)

- By working through this process, you will be able to see how WPPB is wired as well as how to wire MySQL, Custom REST APIs, React.js and self contained plugin pages.
- It can be really useful to use some Javascript or JS Library to create dynamic client side pages. 1). Infinite Scroll of posts using vanilla JS, 2). A boiler plate React two field form.
- These can be Svelte, React, Vue and by using @wordpress/scripts, WP scaffolds out how to create a bundled JS file that can be enqueued like any other JS file.

Process

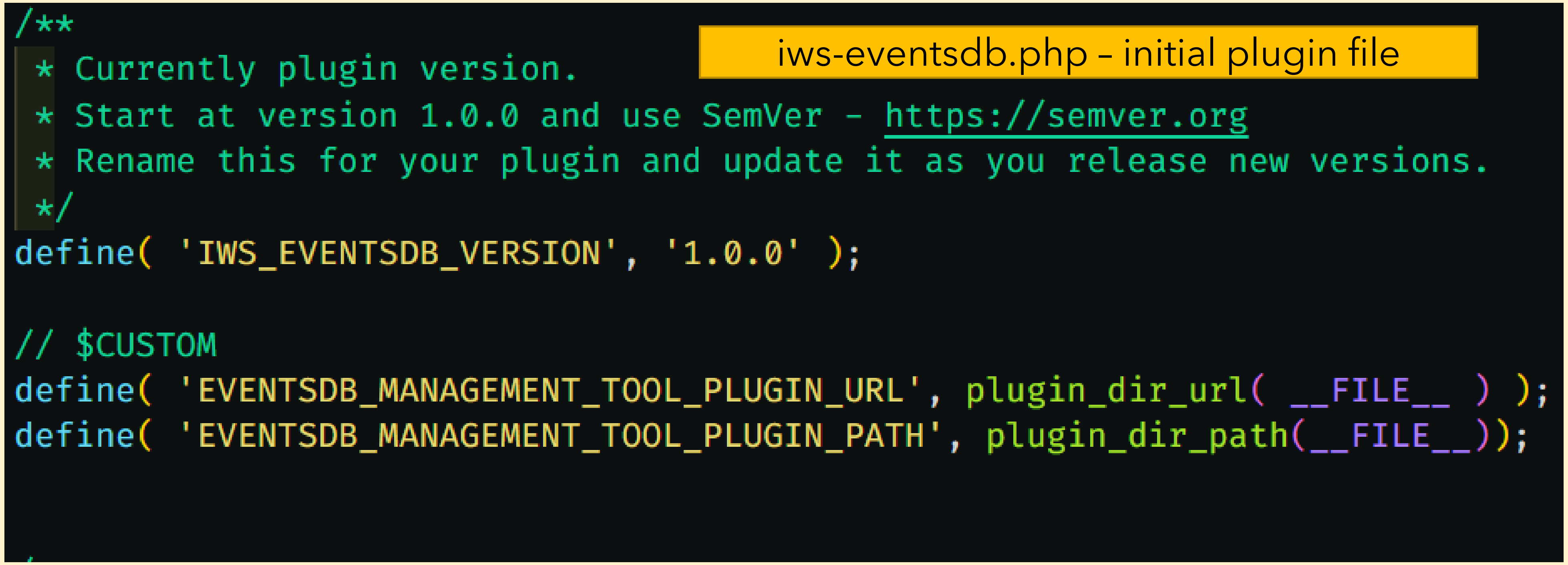
- We will use a generator at wppb.me to create our plugin with our custom name, (very handy).
- We will then compare it with our final version and as we recreate our plugin, we will place comments to show what are our additions and what is in the boilerplate initially.
- With all the code and this PowerPoint pdf, you will then be able to practice creating your own plugins.
- It can seem convoluted at first, but it creates great structure for standardisation and for team development.
- Once you see it in action you will see why it is structures as it is.
- Thanks Devin Vinson!

Admin Folder	This contains files for admin section: **class-<name>-admin.php** starter main file for functions use in hooks
Includes Folder	This contains files for admin section: **class-<name>.php** starter main file that fires of loaders etc. This is the start of the plugin functionality with functions refactored into admin and public as appropriate.
Public Folder	This contains files for admin section : **class-<name>-public.php** starter main file functions use in hooks
iws-events.php	Initial plugin file to correspond with folder name.
uninstall.php	File that is automatically run when plugin is UNINSTALLED (not deactivated).



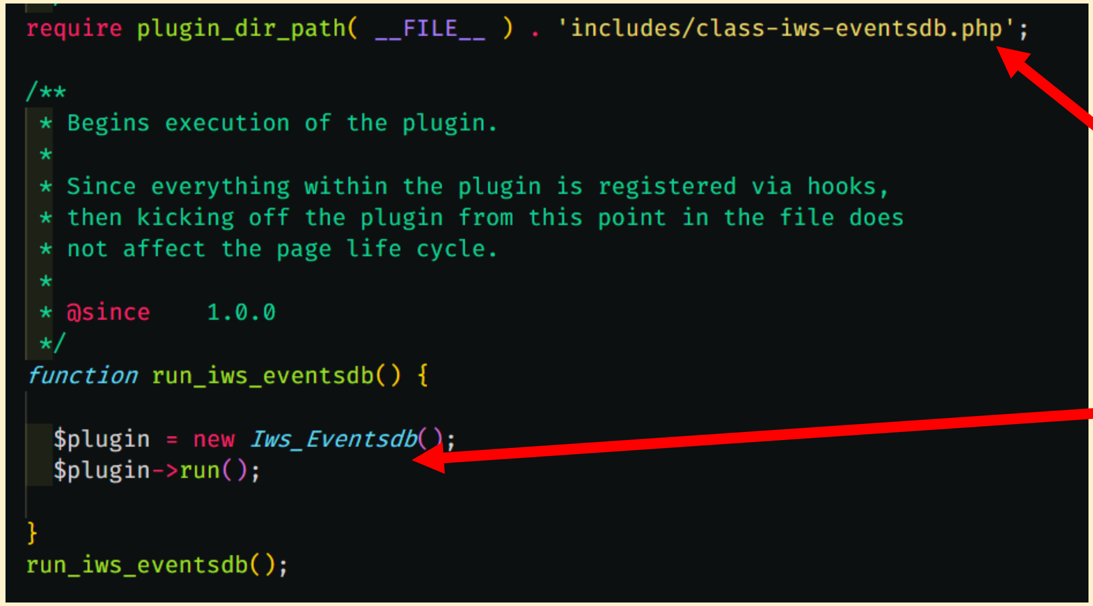
```
includes
  class-iws-eventsdb-activator.php
  class-iws-eventsdb-deactivator.php
  class-iws-eventsdb-i18n.php
  class-iws-eventsdb-loader.php
  class-iws-eventsdb.php
  helpers.php
  index.php
```

- Loader has class to load in dependencies – we don’t add code here.
- Activator has our code to run on Activation and Deactivator has our code to run on deactivation.
- Uninstall.php in root has our code to run when plugin is deleted.
- We can add common helper files etc here and the load_dependencies in includes/class-plugin-name.php will require them in see 5.



iws-eventsdb.php – initial plugin file

```php
/**
 * Currently plugin version.
 * Start at version 1.0.0 and use SemVer - https://semver.org
 * Rename this for your plugin and update it as you release new versions.
 */
define( 'IWS_EVENTSDB_VERSION', '1.0.0' );

// $CUSTOM
define( 'EVENTSDB_MANAGEMENT_TOOL_PLUGIN_URL', plugin_dir_url( __FILE__ ) );
define( 'EVENTSDB_MANAGEMENT_TOOL_PLUGIN_PATH', plugin_dir_path(__FILE__));
```



```php
require plugin_dir_path( __FILE__ ) . 'includes/class-iws-eventsdb.php';

/**
 * Begins execution of the plugin.
 *
 * Since everything within the plugin is registered via hooks,
 * then kicking off the plugin from this point in the file does
 * not affect the page life cycle.
 *
 * @since    1.0.0
 */
function run_iws_eventsdb() {

	$plugin = new Iws_Eventsdb();
	$plugin->run();

}
run_iws_eventsdb();
```

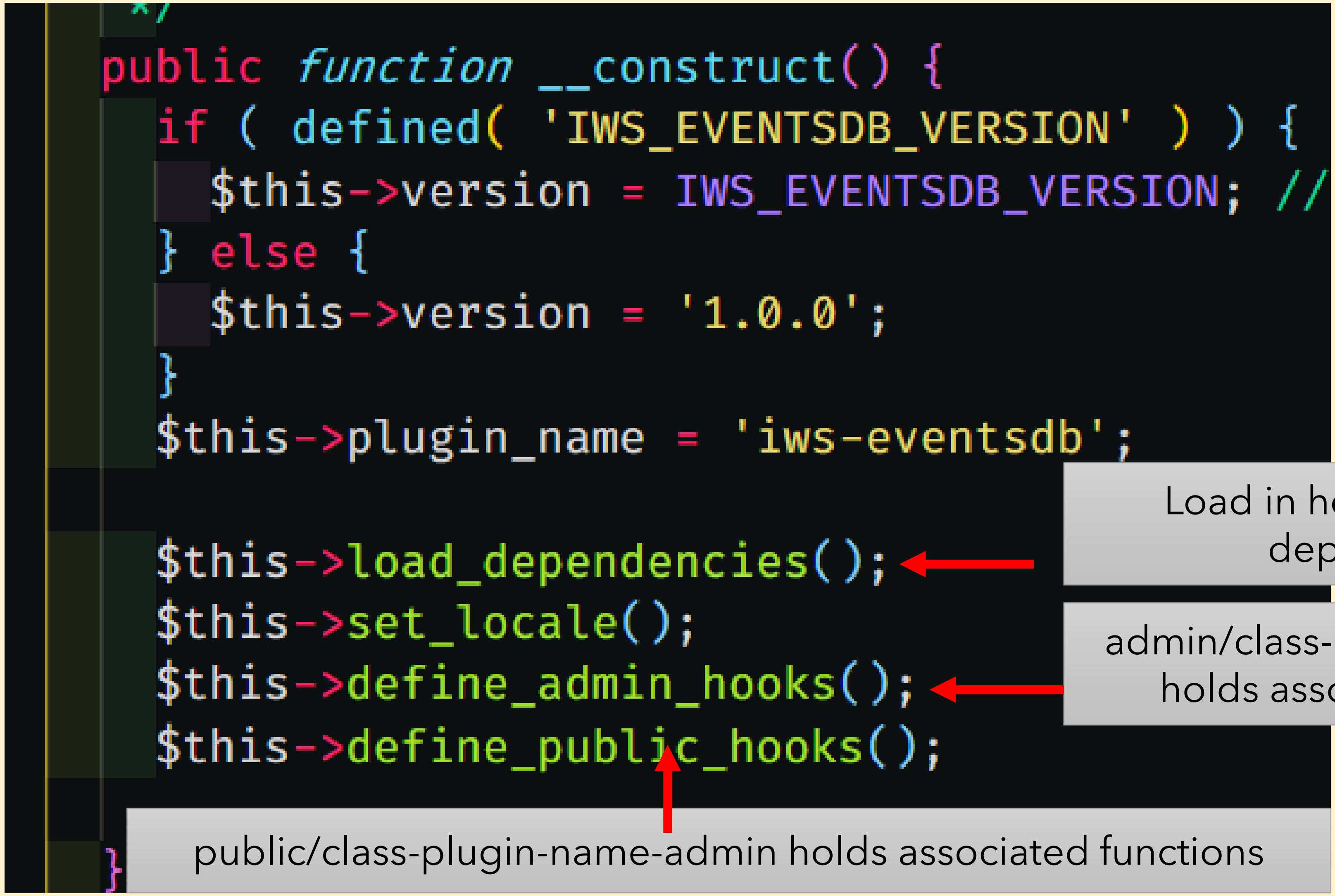
iws-eventsdb.php – initial plugin file

Loads in includes/class-plugin-name.php

Fires off its run method

includes/class-plugin-name.php

Hooks and Dependencies like helper files and global utilities:



```
*/
public function __construct() {
    if ( defined( 'IWS_EVENTSDB_VERSION' ) ) {
        $this->version = IWS_EVENTSDB_VERSION; //
    } else {
        $this->version = '1.0.0';
    }
    $this->plugin_name = 'iws-eventsdb';

    $this->load_dependencies();
    $this->set_locale();
    $this->define_admin_hooks();
    $this->define_public_hooks();

}
```

Load in helper files etc for dependencies

admin/class-plugin-name-admin holds associated functions

public/class-plugin-name-admin holds associated functions

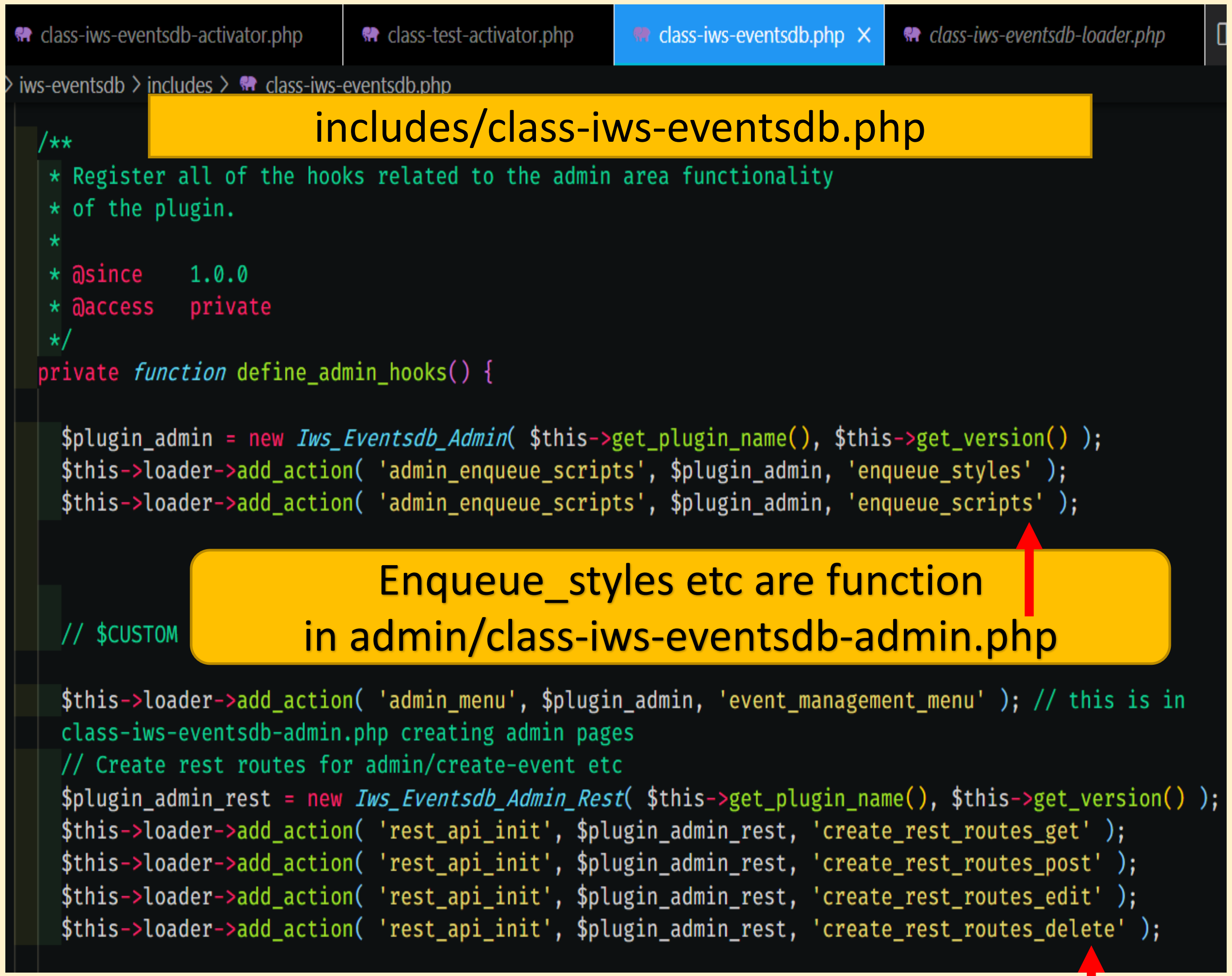
Functions for admin hooks are located in admin/class-plugin-name-admin.php and for the public hooks they are in public/class-plugin-name-public.php

In these functions we can also enqueue and other files that we may need.

In admin and public we have JS and CSS folders and the files in them are automatically enqueued.

We also have a partials folder to store templates.

We will look later at how we can use filter hooks to redirect theme pages to pages in our plugin.

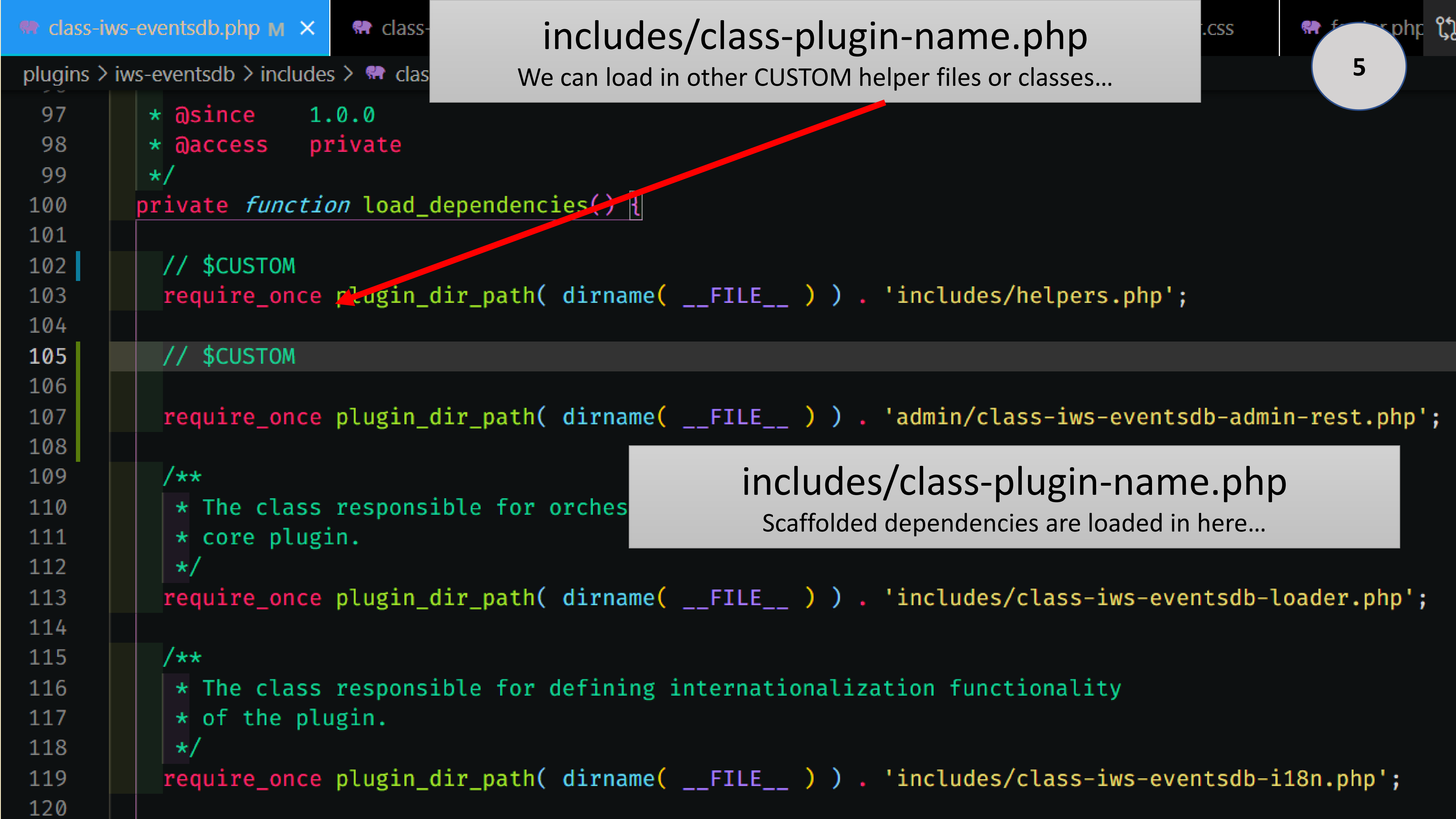

class-iws-eventsdb-activator.php
class-test-activator.php
class-iws-eventsdb.php
class-iws-eventsdb-loader.php
iws-eventsdb > includes > class-iws-eventsdb.php

includes/class-iws-eventsdb.php

```
/**
 * Register all of the hooks related to the admin area functionality
 * of the plugin.
 *
 * @since    1.0.0
 * @access   private
 */
private function define_admin_hooks() {

	$plugin_admin = new Iws_Eventsdb_Admin( $this->get_plugin_name(), $this->get_version() );
	$this->loader->add_action( 'admin_enqueue_scripts', $plugin_admin, 'enqueue_styles' );
	$this->loader->add_action( 'admin_enqueue_scripts', $plugin_admin, 'enqueue_scripts' );


	// $CUSTOM

	$this->loader->add_action( 'admin_menu', $plugin_admin, 'event_management_menu' ); // this is in
	class-iws-eventsdb-admin.php creating admin pages
	// Create rest routes for admin/create-event etc
	$plugin_admin_rest = new Iws_Eventsdb_Admin_Rest( $this->get_plugin_name(), $this->get_version() );
	$this->loader->add_action( 'rest_api_init', $plugin_admin_rest, 'create_rest_routes_get' );
	$this->loader->add_action( 'rest_api_init', $plugin_admin_rest, 'create_rest_routes_post' );
	$this->loader->add_action( 'rest_api_init', $plugin_admin_rest, 'create_rest_routes_edit' );
	$this->loader->add_action( 'rest_api_init', $plugin_admin_rest, 'create_rest_routes_delete' );
```

Enqueue_styles etc are function
in admin/class-iws-eventsdb-admin.php

Our own CUSTOM hooks - Admin/class-iws-eventsdb-rest.php holds custom REST API functions



includes/class-plugin-name.php

We can load in other CUSTOM helper files or classes…

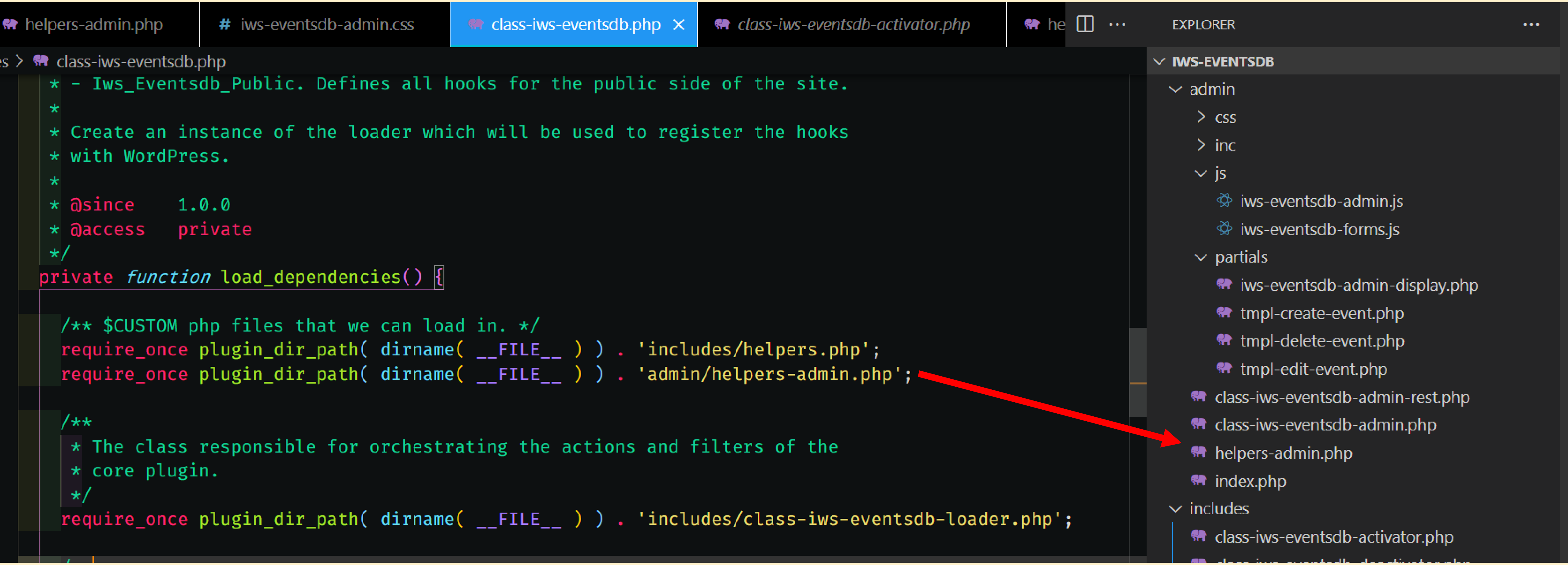
```php
 * @since     1.0.0
 * @access    private
 */
private function load_dependencies() {

    // $CUSTOM
    require_once plugin_dir_path( dirname( __FILE__ ) ) . 'includes/helpers.php';

    // $CUSTOM

    require_once plugin_dir_path( dirname( __FILE__ ) ) . 'admin/class-iws-eventsdb-admin-rest.php';

    /**
     * The class responsible for orches
     * core plugin.
     */
    require_once plugin_dir_path( dirname( __FILE__ ) ) . 'includes/class-iws-eventsdb-loader.php';

    /**
     * The class responsible for defining internationalization functionality
     * of the plugin.
     */
    require_once plugin_dir_path( dirname( __FILE__ ) ) . 'includes/class-iws-eventsdb-i18n.php';
```

includes/class-plugin-name.php

Scaffolded dependencies are loaded in here…

includes/class-plugin-name.php

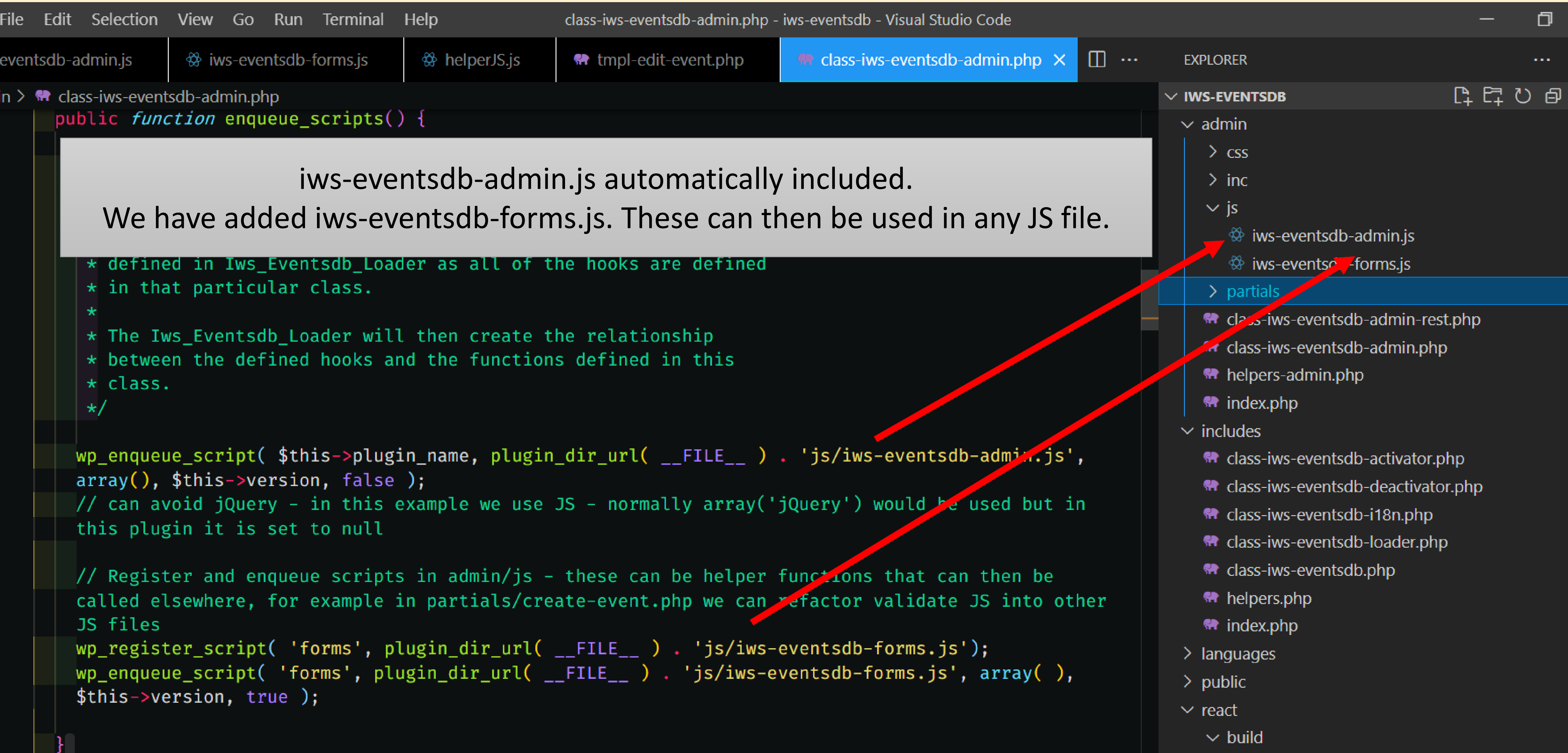
We can load in other CUSTOM helper files or classes…



```
 * - Iws_Eventsdb_Public. Defines all hooks for the public side of the site.
 *
 * Create an instance of the loader which will be used to register the hooks
 * with WordPress.
 *
 * @since    1.0.0
 * @access   private
 */
private function load_dependencies() {

  /** $CUSTOM php files that we can load in. */
  require_once plugin_dir_path( dirname( __FILE__ ) ) . 'includes/helpers.php';
  require_once plugin_dir_path( dirname( __FILE__ ) ) . 'admin/helpers-admin.php';

  /**
   * The class responsible for orchestrating the actions and filters of the
   * core plugin.
   */
  require_once plugin_dir_path( dirname( __FILE__ ) ) . 'includes/class-iws-eventsdb-loader.php';
```

admin/class-plugin-name-admin.php

Enqueues initial JS file and then our own.



```php
public function enqueue_scripts() {
```

iws-eventsdb-admin.js automatically included.

We have added iws-eventsdb-forms.js. These can then be used in any JS file.

```php
 * defined in Iws_Eventsdb_Loader as all of the hooks are defined
 * in that particular class.
 *
 * The Iws_Eventsdb_Loader will then create the relationship
 * between the defined hooks and the functions defined in this
 * class.
 */

 wp_enqueue_script( $this->plugin_name, plugin_dir_url( __FILE__ ) . 'js/iws-eventsdb-admin.js',
 array(), $this->version, false );
 // can avoid jQuery - in this example we use JS - normally array('jQuery') would be used but in
 this plugin it is set to null

 // Register and enqueue scripts in admin/js - these can be helper functions that can then be
 called elsewhere, for example in partials/create-event.php we can refactor validate JS into other
 JS files
 wp_register_script( 'forms', plugin_dir_url( __FILE__ ) . 'js/iws-eventsdb-forms.js');
 wp_enqueue_script( 'forms', plugin_dir_url( __FILE__ ) . 'js/iws-eventsdb-forms.js', array( ),
 $this->version, true );

}
```