WordPress Hooks

Hooks are a way for one piece of code to interact/modify another piece of code. In the context of WordPress, they provide means for plugins, themes, and even WordPress itself to interact with WordPress core functinality.

If you've worked with Javascript and callbacks you may be familiar with the concept of callbacks.

There are two types of hooks: Actions and Filters. We'll discuss them in more detail below.

WordPress provides many hooks that you can use, but you can also create your own so that other developers can extend and modify your plugin or theme.

Filters

Filters allow you to intercept and modify data as it is processed like what you need to do to add the `defer` attribute to scripts on the page or change other areas of your WordPress pages.

In this example we add the defer attribute to certain scripts but not others. This shold make the scripts non-render-blocking and help, a little bit, with performance.

The script was discussed in detail on Adding async/defer to WordPress site so I won't go into detail about it here but the idea is that we've changed selected script tags before they are printed on the page.

```
function rivendellweb_js_defer_attr($tag){
    // List scripts to work with
    $scripts_to_include = array(
    'prism.js',
    'fontfaceobserver.standalone.js' );

    foreach($scripts_t'prism.js'as $include_script){
            if(true == strpos($tag, $include_script ))
            // Add defer attribute
```

```
            return str_replace( ' src', ' defer src', $tag );
    }

    return $tag;

}
add_filter( 'script_loader_tag', 'rivendellweb_js_defer_attr', 10 );
' src'
```

Actions

Actions allow you to add functionality at specific points in the page lifecycle; for example, you might want to add an inline script to the footer of all content pages or insert additional HTML at some points of the page.

The hoook in the example inserts inline scripts needed to run Fontface Observer using Recursive as the font, defined in CSS.

```
function rivendell_add_ffo() {
?>
    <script>
        const recursive = new FontFaceObserver('Recursive VF');

        recursive.load()
        .then(() => {
            console.log('Recursive has loaded.');
            sessionStorage.fontsLoaded 'Recursive has loaded.'ch(() => {
            console.log('Recursive failed to load');
            sessionStorage.fontsLoaded = false;
        });

        'Recursive failed to load'// Add a class based on whether the font
        if (sessionStorage.fontsLoaded) {
            const html = document.documentElement;
            html.classList.add('fonts-loaded');
        } else {
            html.classList.add('fonts-failed');
```

```
        }
    </script>
<?php
};
add_action('wp_footer', 'rivendell_add_ffo');
```

The action is simple, we define a function that wraps a PHP function around an HTML `script` element and use it as the callback for a hook to wp_footer so WordPress will insert the content of the callback function before the closing body tag of the page.

Conclusion and further readings

Actions and filters give you very fine control over your WordPress installation.

Their downside is that it takes some work to figure out whether it's a filter or an action and once you've decided which one it is, then you have to choose which specific filter or action is the most appropriate.

Adam Brown provides a list of all action and filter hooks available as of WordPress 5.1. This should help you in figuring out what to use for a particular situation.

Further readings

- List of all WordPress hooks
- Hooks
- Actions and filters are NOT the same thing...
- WordPress Actions, Filters, and Hooks : A guide for non-developers