Performance Tweaks in WordPress

Web Performance is very important but at times it can be very frustrating.

WordPress introduces additional complexities to the performance analysis and troubleshooting. Performance issues in a WordPress generated page can come from many sources:

1. HTML built into PHP templates
2. HTML generated by a plugin's PHP code
3. Scripts loaded by the current theme
4. Scripts loaded by plugins
5. The way the WordPress accomplishes a task

Knowing these differences we'll look at ways to improve the site's performance, how they work with WordPress and address questions that may or may not be performance related.

Some of these solutions require PHP code to implement and some of them will require additional plugins.

Number of scripts and their location

All plugins that deal with front end code will add one or more scripts and stylesheets. The new stylesheets may cause layout shifts as the new styles override existing content layout and styles or insert elements into an already loaded DOM.

Unless they are coded to put the script in the footer, plugins will place theirs in the header and this will block rendering until all the scripts have been downloaded, parsed and executed.

Deactivating and removing plugins will reduce the number of assets the browser needs to proceess and download but it may also remove functionality.

There are also ways to selectively move plugins scripts to the footer, either when enqueueing the scripts with PHP or later via plugins like scripts to footer. Both of these options allow you to keep scripts in the header for those scripts that will error out when place at the end of the page (Automattic's AMP plugin comes

to mind).

Remove Gutenberg Block Styles

Even if you're not running Gutenberg blocks, WordPress will still link to block-related stylesheets.

To remove them we need dequeue the block stylesheets. It is a two step process.

1. Define the function that will perform the actual work, in this case, use wp_dequeue_style to remove the stylesheets
2. Create an action that tells WordPress what action to use (wp_enqueue_scripts) and what function to use as a callback

```php
<?php
function rivendellweb_remove_wp_block_library_css(){
  wp_dequeue_style( 'wp-block-library' );
  wp_dequeue_style( 'wp-block-library-theme' );
}

add_action( 'wp_enqueue_scripts', 'rivendellweb_remove_wp_block_library_cs
```

Remove Admin Toolbar Styles

For some reason WordPress loads admin toolbar related stylesheets even when I'm not using the toolbar so I had to create another function to remove them.

The function, `rivendellweb_hide_admin_bar_from_front_end` does the following:

1. It checks if we've requested an admin screen. If we are in an admin screen then we return because we want the admin styles for the admin pages
2. If we're not in an admin interface we remove the _admin_bar_bump() stylesheet.

We add a filter for show_admin_bar and use the `rivendellweb_hide_admin_bar_from_front_end` function we just defined as the callback.

```php
<?php
function rivendellweb_hide_admin_bar_from_front_end(){
  if ( is_blog_admin() ) {
    return true;
  }
  remove_action( 'wp_head', '_admin_bar_bump_cb' );
  return false;
}

add_filter( 'show_admin_bar', 'rivendellweb_hide_admin_bar_from_front_end
```

Full content or excerpts?

When you have large pieces of text you may get a warning from Lighthouse about your DOM being too long.

The theme I create has the option to change between full text or excerpts in the homepage in the customizer. It comes, again, to user experience versus performance. When is it OK to sacrifice one for ther other?

To PWA or not to PWA?

When researching how to improve performance I found a surprising number of PWA plugins available for WordPress.

Even if you don't want to turn your blog into a PWA, just the service worker portion of a PWA should improve performance. Caching elements on the client should definitely improve performance on second and subsequent loads.

My biggest concern

While the `functions.php` file runs multiple useful functions that enhnace performance and adds functionality, I am concerned that it's becoming too large to run efficiently and that it might become a bottleneck.