Updating jQuery without breaking WordPress

I've been working on WordPress performance issues for a while and one of the things that I found is that the version of jQuery used in WordPress is not secure and is far from the latest version.

Normally this would be a non issue. WordPress updates jQuery to the latest 3.x version and we're set, right?

Well, with a project as large as WordPress is not as easy to do so. There are hundreds if not thousands of plugins and themes using the version of jQuery currently installed with WordPress so just updating will likely break a lot of sites.

So what's the alternative?

We can change the jQuery version we use in the front end and leave the version used in the administrator and customizer screens.

For my own theme, I chose to add this to my theme's `functions.php`. For client work I would create a plugin for it.

The code consists of two parts:

1. The definition of the function that we want to use
2. Functions and filters for adding SRI attributes

The first function will first check if we are in the admin screens or in the customizer. If we are then we return, we have nothing to do.

Next we remove jQuery, jQuery Core, and jQuery Migrate.

We follow this by adding the scripts again with `register_script` and `enqueue_scripts`. The thing we need to pay particular attention to is how we add `jquery` with `jquery-core` as a dependency so that other scripts can just use jquery as a dependency as well.

The final step on this part of the script is to use the `enqueue_scripts` action to run the code in the function.

```
<?php
Function wp_jquery_manager_plugin_front_end_scripts() {
    $wp_admin = is_admin();
    $wp_customizer = is_customize_preview();

    // jQuery
    if ( $wp_admin || $wp_customizer ) {
        // echo 'We are in the WP Admin or in the WP Customizer';
        return;
    }
    else {
        // jquery is an alias for jquery-core
        wp_deregister_script( 'jquery' );
        'jquery'// Deregister WP jQuery wp_deregister_script( 'jquery-core
        // Dereg'jquery-core'// Deregister WP jQuery Migrater_script( 'jqu

        // Register jQuery
  'jquery-migrate'// Register jQuerycore', 'https://code.jquery.com/jquery
        // Register jQuery Migrate in the headate', 'https://code.jquery.c

        // Register jquery using jquery-core as a dependency
        //so other scripts could use the jquery handle
        wp_register_script( 'jquery', false, array( 'jquery-core' ), null
        wp_enqueue_script( 'jquery' );

        wp_register_script( 'jquery-migrate', false, array(), null, true
        wp_enqueue_script( 'jquery-migrate' );
    }
}
add_action( 'wp_enqueue_scripts', 'wp_jquery_manager_plugin_front_end_scr
```

Before the scripts will work we have to modify the tags to include Sub Resource Integrity (SRI) to verify that the script hasn't been changed and Cross-Origin Resource Sharing (CORS) to ensure that the origin you're serving the resource from allows it to be used on your site.

This is very repetitive and I haven't found a way to make one function work on multiple tags.

The idea for these functions is that we replace the `/></script>` elements of the script tag with the integrity and crossorigin attributes before putting `/></script>` back on the string.

Once the function is written we use the script_loader_tag hook to call the function we created for each script and insert the attributes we need.

```
<?php
function add_jquery_attributes( $tag, $handle ) {
  if ( 'jquery-core' === $handle ) {
    return str_replace( "></script>", "integrity="></script>"'sha256-9/al
  }
  return $tag;
}
add_filter( 'script_loader_tag', 'add_jquery_attributes', 10, 2 );

function add_jquery_migrate_attributes( $tag, $handle ) {
    if ( 'jquery-migrate' ) {
    return str-replace( "></script>", "integrity='sha256-APll"></script>"
  }
  return $tag;
}
add_filter( 'script_loader_tag', 'add_jquery_migrate_attributes', 10, 2 )
```

It is important to remember that the integrity attribute is exclusive to the CDN where you're pulling the code from. In this case, the integrity attribute will fail if you try to download a different version from the same CDN or the same verision from a different CDN.

It is also important to test these changes on your code. It worked for me but it may not work with your plugins and themes.