

master ▾

...

[visualizer](#) / [classes](#) / [Visualizer](#) / [Module](#) / [Chart.php](#) / [<> Jump to ▾](#)

girishpanchal30 Fix PHP error when polylang plugin activated #960 ✖

[History](#)

5 contributors



1597 lines (1434 sloc) | 56.1 KB

...

```
1  <?php
2  // +-----+
3  // | Copyright 2013 Madpixels (email : visualizer@madpixels.net) |
4  // +-----+
5  // | This program is free software; you can redistribute it and/or modify |
6  // | it under the terms of the GNU General Public License, version 2, as |
7  // | published by the Free Software Foundation. |
8  // |
9  // | This program is distributed in the hope that it will be useful, |
10 // | but WITHOUT ANY WARRANTY; without even the implied warranty of |
11 // | MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the |
12 // | GNU General Public License for more details. |
13 // |
14 // | You should have received a copy of the GNU General Public License |
15 // | along with this program; if not, write to the Free Software |
16 // | Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, |
17 // | MA 02110-1301 USA |
18 // +-----+
19 // | Author: Eugene Manuilov <eugene@manuilov.org> |
20 // +-----+
21 /**
22  * The module for all stuff related to getting, editing, creating and deleting charts.
23  *
24  * @category Visualizer
25  * @package Module
26  *
27  * @since 1.0.0
28  */
```

```

29 class Visualizer_Module_Chart extends Visualizer_Module {
30
31     const NAME = __CLASS__;
32
33     /**
34      * The chart object.
35      *
36      * @since 1.0.0
37      *
38      * @access private
39      * @var WP_Post
40      */
41     private $_chart;
42
43     /**
44      * Constructor.
45      *
46      * @since 1.0.0
47      *
48      * @access public
49      *
50      * @param Visualizer_Plugin $plugin The instance of the plugin.
51      */
52     public function __construct( Visualizer_Plugin $plugin ) {
53         parent::__construct( $plugin );
54         $this->_addAjaxAction( Visualizer_Plugin::ACTION_GET_CHARTS, 'getCharts' );
55         $this->_addAjaxAction( Visualizer_Plugin::ACTION_DELETE_CHART, 'deleteChart' );
56         $this->_addAjaxAction( Visualizer_Plugin::ACTION_CREATE_CHART, 'renderChartPages' );
57         $this->_addAjaxAction( Visualizer_Plugin::ACTION_EDIT_CHART, 'renderChartPages' );
58         $this->_addAjaxAction( Visualizer_Plugin::ACTION_UPLOAD_DATA, 'uploadData' );
59         $this->_addAjaxAction( Visualizer_Plugin::ACTION_CLONE_CHART, 'cloneChart' );
60         $this->_addAjaxAction( Visualizer_Plugin::ACTION_EXPORT_DATA, 'exportData' );
61
62         $this->_addAjaxAction( Visualizer_Plugin::ACTION_FETCH_DB_DATA, 'getQueryData' );
63         $this->_addAjaxAction( Visualizer_Plugin::ACTION_SAVE_DB_QUERY, 'saveQuery' );
64
65         $this->_addAjaxAction( Visualizer_Plugin::ACTION_JSON_GET_ROOTS, 'getJsonRoots' );
66         $this->_addAjaxAction( Visualizer_Plugin::ACTION_JSON_GET_DATA, 'getJsonData' );
67         $this->_addAjaxAction( Visualizer_Plugin::ACTION_JSON_SET_DATA, 'setJsonData' );
68         $this->_addAjaxAction( Visualizer_Plugin::ACTION_JSON_SET_SCHEDULE, 'setJsonSchedu
69
70         $this->_addAjaxAction( Visualizer_Plugin::ACTION_SAVE_FILTER_QUERY, 'saveFilter' )
71
72         $this->_addFilter( 'visualizer_get_sidebar', 'getSidebar', 10, 2 );
73
74     }
75
76     /**
77      * Generates the HTML of the sidebar for the chart.

```

```

78      *
79      * @since ?
80      *
81      * @access public
82      */
83      public function getSidebar( $sidebar, $chart_id ) {
84          $chart      = get_post( $chart_id );
85          $data        = $this->_getChartArray( $chart );
86          $sidebar      = '';
87          $sidebar_class = $this->load_chart_class_name( $chart_id );
88          if ( class_exists( $sidebar_class, true ) ) {
89              $sidebar      = new $sidebar_class( $data['settings'] );
90              $sidebar->__series = $data['series'];
91              $sidebar->__data   = $data['data'];
92          } else {
93              $sidebar = apply_filters( 'visualizer_pro_chart_type_sidebar', '', $data );
94              if ( $sidebar !== '' ) {
95                  $sidebar->__series = $data['series'];
96                  $sidebar->__data   = $data['data'];
97              }
98          }
99          return str_replace( '"', "'", $sidebar->__toString() );
100      }
101
102      /**
103       * Sets the schedule for how JSON-endpoint charts should be updated.
104       *
105       * @since ?
106       *
107       * @access public
108       */
109      public function setJsonSchedule() {
110          check_ajax_referer( Visualizer_Plugin::ACTION_JSON_SET_SCHEDULE . Visualizer_Plugin::
111
112          $chart_id = filter_input(
113              INPUT_POST,
114              'chart',
115              FILTER_VALIDATE_INT,
116              array(
117                  'options' => array(
118                      'min_range' => 1,
119                  ),
120              )
121          );
122
123          if ( ! $chart_id ) {
124              wp_send_json_error();
125          }
126

```

```

127         $time = filter_input(
128             INPUT_POST,
129             'time',
130             FILTER_VALIDATE_INT,
131             array(
132                 'options' => array(
133                     'min_range' => -1,
134                 ),
135             )
136         );
137
138         if ( Visualizer_Module::is_pro() ) {
139             $is_woocommerce_report = filter_input(
140                 INPUT_POST,
141                 'is_woocommerce_report',
142                 FILTER_VALIDATE_BOOLEAN
143             );
144
145             if ( $is_woocommerce_report ) {
146                 update_post_meta( $chart_id, Visualizer_Plugin::CF_IS_WOOCOMMERCE_
147             } else {
148                 delete_post_meta( $chart_id, Visualizer_Plugin::CF_IS_WOOCOMMERCE_
149             }
150         }
151
152         delete_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_SCHEDULE );
153
154         if ( -1 < $time ) {
155             add_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_SCHEDULE, $time );
156             // Update schedules.
157             $schedules = get_option( Visualizer_Plugin::CF_JSON_SCHEDULE,
158             $schedules[ $chart_id ] = time() + $time * HOUR_IN_SECONDS;
159             update_option( Visualizer_Plugin::CF_JSON_SCHEDULE, $schedules );
160         }
161         wp_send_json_success();
162     }
163
164     /**
165      * Get the root elements for JSON-endpoint.
166      *
167      * @since ?
168      *
169      * @access public
170      */
171     public function getJsonRoots() {
172         check_ajax_referer( Visualizer_Plugin::ACTION_JSON_GET_ROOTS . Visualizer_Plugin::
173
174         $params = wp_parse_args( $_POST['params'] );
175

```

```

176         $source = new Visualizer_Source_Json( $params );
177
178         $roots = $source->fetchRoots();
179         if ( empty( $roots ) ) {
180             wp_send_json_error( array( 'msg' => $source->get_error() ) );
181         }
182
183         wp_send_json_success( array( 'url' => $params['url'], 'roots' => $roots ) );
184     }
185
186     /**
187      * Get the data for the JSON-endpoint corresponding to the chosen root.
188      *
189      * @since ?
190      *
191      * @access public
192      */
193     public function getJsonData() {
194         check_ajax_referer( Visualizer_Plugin::ACTION_JSON_GET_DATA . Visualizer_Plugin::V
195
196         $params = wp_parse_args( $_POST['params'] );
197
198         $chart_id = $params['chart'];
199
200         if ( empty( $chart_id ) ) {
201             wp_die();
202         }
203
204         $source = new Visualizer_Source_Json( $params );
205         $source->fetch();
206         $data = $source->getRawData();
207
208         if ( empty( $data ) ) {
209             wp_send_json_error( array( 'msg' => esc_html__( 'Unable to fetch data from
210         }
211
212         $data = Visualizer_Render_Layout::show( 'editor-table', $data, $chart_id, 'viz-j
213         wp_send_json_success( array( 'table' => $data, 'root' => $params['root'], 'url' =>
214     }
215
216     /**
217      * Updates the database with the correct post parameters for JSON-endpoint charts.
218      *
219      * @since ?
220      *
221      * @access public
222      */
223     public function setJsonData() {
224         check_ajax_referer( Visualizer_Plugin::ACTION_JSON_SET_DATA . Visualizer_Plugin::V

```

```

225
226     $params = $_POST;
227     $chart_id = $_GET['chart'];
228
229     if ( empty( $chart_id ) ) {
230         wp_die();
231     }
232
233     $chart = get_post( $chart_id );
234
235     $source = new Visualizer_Source_Json( $params );
236     update_post_meta( $chart->ID, Visualizer_Plugin::CF_EDITABLE_TABLE, true );
237     $source->fetchFromEditableTable();
238
239     $content = $source->getData( get_post_meta( $chart->ID, Visualizer_Plugin::CF_E
240     $chart->post_content = $content;
241     wp_update_post( $chart->to_array() );
242     update_post_meta( $chart->ID, Visualizer_Plugin::CF_SERIES, $source->getSeries() );
243     update_post_meta( $chart->ID, Visualizer_Plugin::CF_SOURCE, $source->getSourceName
244     update_post_meta( $chart->ID, Visualizer_Plugin::CF_DEFAULT_DATA, 0 );
245     update_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_URL, $params['url'] );
246     update_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_ROOT, $params['root'] );
247
248     delete_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_HEADERS );
249     $headers = array( 'method' => $params['method'] );
250     if ( ! empty( $params['auth'] ) ) {
251         $headers['auth'] = $params['auth'];
252     } elseif ( ! empty( $params['username'] ) && ! empty( $params['password'] ) ) {
253         $headers['auth'] = array( 'username' => $params['username'], 'password' =>
254     }
255
256     add_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_HEADERS, $headers );
257
258     delete_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_PAGING );
259     if ( ! empty( $params['paging'] ) ) {
260         add_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_PAGING, $params['pag
261     }
262
263     if ( Visualizer_Module::is_pro() ) {
264         if ( ! empty( $params['vz_woo_source'] ) ) {
265             update_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_WOOCOMMER
266         } else {
267             delete_post_meta( $chart->ID, Visualizer_Plugin::CF_JSON_WOOCOMMER
268         }
269     }
270
271     $time = filter_input(
272         INPUT_POST,
273         'time',

```

```

274         FILTER_VALIDATE_INT,
275         array(
276             'options' => array(
277                 'min_range' => -1,
278             ),
279         )
280     );
281
282     delete_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_SCHEDULE );
283
284     if ( -1 < $time ) {
285         add_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_SCHEDULE, $time );
286     }
287
288     // delete other source specific parameters.
289     delete_post_meta( $chart_id, Visualizer_Plugin::CF_DB_QUERY );
290     delete_post_meta( $chart_id, Visualizer_Plugin::CF_DB_SCHEDULE );
291     delete_post_meta( $chart_id, Visualizer_Plugin::CF_CHART_URL );
292     delete_post_meta( $chart_id, Visualizer_Plugin::CF_CHART_SCHEDULE );
293
294     $render      = new Visualizer_Render_Page_Update();
295     $render->id    = $chart->ID;
296     $render->data  = json_encode( $source->getRawData( get_post_meta( $chart_id, Visu
297     $render->series = json_encode( $source->getSeries() );
298     $render->render();
299
300     defined( 'WP_TESTS_DOMAIN' ) ? wp_die() : exit();
301 }
302
303
304 /**
305  * Fetches charts from database.
306  *
307  * This method is also called from the media pop-up (classic editor: create a post and add
308  *
309  * @since 1.0.0
310  *
311  * @access public
312  */
313 public function getCharts() {
314     $query_args = array(
315         'post_type'      => Visualizer_Plugin::CPT_VISUALIZER,
316         'posts_per_page' => 9,
317         'paged'          => filter_input(
318             INPUT_GET,
319             'page',
320             FILTER_VALIDATE_INT,
321             array(
322                 'options' => array(

```

```

323                                     'min_range' => 1,
324                                     'default'   => 1,
325                                     ),
326                                 )
327                             ),
328                        );
329        $filter      = filter_input( INPUT_GET, 's', FILTER_SANITIZE_STRING );
330        if ( empty( $filter ) ) {
331            // 'filter' is from the modal from the add media button.
332            $filter = filter_input( INPUT_GET, 'filter', FILTER_SANITIZE_STRING );
333        }
334
335        if ( $filter && in_array( $filter, Visualizer_Plugin::getChartTypes(), true ) ) {
336            $query_args['meta_query'] = array(
337                array(
338                    'key'      => Visualizer_Plugin::CF_CHART_TYPE,
339                    'value'    => $filter,
340                    'compare' => '=',
341                ),
342            );
343        }
344        $query = new WP_Query( $query_args );
345        $charts = array();
346        while ( $query->have_posts() ) {
347            $chart      = $query->next_post();
348            $chart_data = $this->_getChartArray( $chart );
349            $chart_data['id'] = $chart->ID;
350            $chart_data['library'] = $this->load_chart_type( $chart->ID );
351            $css              = '';
352            $settings         = $chart_data['settings'];
353            $arguments        = $this->get_inline_custom_css( 'visualizer-chart-' .
354                if ( ! empty( $arguments ) ) {
355                    $css      = $arguments[0];
356                    $settings = $arguments[1];
357                }
358            $chart_data['settings'] = $settings;
359            $chart_data['css'] = $css;
360            $charts[]          = $chart_data;
361        }
362        self::_sendResponse(
363            array(
364                'success' => true,
365                'data'    => $charts,
366                'total'   => $query->max_num_pages,
367            )
368        );
369    }
370
371    /**

```



```

372     * Returns chart data required for rendering.
373     *
374     * @since 1.0.0
375     *
376     * @access private
377     *
378     * @param WP_Post $chart The chart object.
379     *
380     * @return array The array of chart data.
381     */
382     private function _getChartArray( WP_Post $chart = null ) {
383         if ( is_null( $chart ) ) {
384             $chart = $this->_chart;
385         }
386         $type = get_post_meta( $chart->ID, Visualizer_Plugin::CF_CHART_TYPE, true );
387         $series = apply_filters( Visualizer_Plugin::FILTER_GET_CHART_SERIES, get_post_meta
388         $data = self::get_chart_data( $chart, $type );
389         $library = $this->load_chart_type( $chart->ID );
390
391         $settings = get_post_meta( $chart->ID, Visualizer_Plugin::CF_SETTINGS, true );
392         $settings = apply_filters( Visualizer_Plugin::FILTER_GET_CHART_SETTINGS, $settings
393         if ( ! empty( $atts['settings'] ) ) {
394             $settings = apply_filters( $atts['settings'], $settings, $chart->ID, $type
395         }
396
397         $css = '';
398         $arguments = $this->get_inline_custom_css( 'visualizer-' . $chart->ID, $settings
399         if ( ! empty( $arguments ) ) {
400             $css = $arguments[0];
401             $settings = $arguments[1];
402         }
403
404         $date_formats = Visualizer_Source::get_date_formats_if_exists( $series, $data );
405
406         return array(
407             'type' => $type,
408             'series' => $series,
409             'settings' => $settings,
410             'data' => $data,
411             'library' => $library,
412             'css' => $css,
413             'date_formats' => $date_formats,
414         );
415     }
416
417     /**
418     * Sends json response.
419     *
420     * @since 1.0.0

```

```

421     *
422     * @access private
423     *
424     * @param array $results The response array.
425     */
426     public static function _sendResponse( $results ) {
427         header( 'Content-type: application/json' );
428         nocache_headers();
429         echo json_encode( $results );
430         defined( 'WP_TESTS_DOMAIN' ) ? wp_die() : exit();
431     }
432
433     /**
434     * Deletes a chart from database.
435     *
436     * @since 1.0.0
437     * @uses wp_delete_post() To delete a chart.
438     *
439     * @access public
440     */
441     public function deleteChart() {
442         $is_post      = $_SERVER['REQUEST_METHOD'] === 'POST';
443         $input_method = $is_post ? INPUT_POST : INPUT_GET;
444         $chart_id     = $success = false;
445         $nonce        = wp_verify_nonce( filter_input( $input_method, 'nonce' ) );
446         $capable      = current_user_can( 'delete_posts' );
447         if ( $nonce && $capable ) {
448             $chart_id = filter_input(
449                 $input_method,
450                 'chart',
451                 FILTER_VALIDATE_INT,
452                 array(
453                     'options' => array(
454                         'min_range' => 1,
455                     ),
456                 )
457             );
458             if ( $chart_id ) {
459                 $chart = get_post( $chart_id );
460                 $success = $chart && $chart->post_type === Visualizer_Plugin::CPT_
461             }
462         }
463         if ( $success ) {
464             global $sitepress;
465             if ( Visualizer_Module::is_pro() && ( function_exists( 'icl_get_languages'
466                 $trid      = $sitepress->get_element_trid( $chart_id, 'post' ) .
467                 $translations = $sitepress->get_element_translations( $trid );
468                 if ( ! empty( $translations ) ) {
469                     foreach ( $translations as $translated_post ) {

```

```

470         wp_delete_post( $translated_post->element_id, true
471     }
472     } else {
473         wp_delete_post( $chart_id, true );
474     }
475     } else {
476         wp_delete_post( $chart_id, true );
477     }
478 }
479 if ( $is_post ) {
480     self::_sendResponse(
481         array(
482             'success' => $success,
483         )
484     );
485 }
486 wp_redirect( remove_query_arg( 'vaction', wp_get_referer() ) );
487 exit;
488 }
489
490 /**
491  * Delete charts that are still in auto-draft mode.
492  */
493 private function deleteOldCharts() {
494     $query = new WP_Query(
495         array(
496             'post_type'      => Visualizer_Plugin::CPT_VISUALIZER,
497             'post_status'    => 'auto-draft',
498             'fields'          => 'ids',
499             'update_post_meta_cache' => false,
500             'update_post_term_cache' => false,
501             'posts_per_page'  => 50,
502             'date_query'     => array(
503                 array(
504                     'before' => 'today',
505                 ),
506             ),
507         )
508     );
509
510     if ( $query->have_posts() ) {
511         $ids = array();
512         while ( $query->have_posts() ) {
513             wp_delete_post( $query->next_post(), true );
514         }
515     }
516 }
517
518 /**

```

```

519     * Renders appropriate page for chart builder. Creates new auto draft chart
520     * if no chart has been specified.
521     *
522     * @since 1.0.0
523     *
524     * @access public
525     */
526     public function renderChartPages() {
527         defined( 'IFRAME_REQUEST' ) || define( 'IFRAME_REQUEST', 1 );
528         if ( ! defined( 'ET_BUILDER_PRODUCT_VERSION' ) && function_exists( 'et_get_theme_v
529             define( 'ET_BUILDER_PRODUCT_VERSION', et_get_theme_version() );
530     }
531     // Set current screen for the render chart.
532     set_current_screen( 'visualizer_render_chart' );
533     // check chart, if chart not exists, will create new one and redirects to the same
534     $chart_id = isset( $_GET['chart'] ) ? filter_var( $_GET['chart'], FILTER_VALIDATE_
535     if ( ! $chart_id || ! ( $chart = get_post( $chart_id ) ) || $chart->post_type !==
536         if ( empty( $_GET['lang'] ) || empty( $_GET['parent_chart_id'] ) ) {
537             $this->deleteOldCharts();
538             $default_type = isset( $_GET['type'] ) && ! empty( $_GET['type'] )
539             $source       = new Visualizer_Source_Csv( VISUALIZER_ABSPATH . DI
540             $source->fetch();
541             $chart_id = wp_insert_post(
542                 array(
543                     'post_type'      => Visualizer_Plugin::CPT_VISUALIZE
544                     'post_title'     => 'Visualization',
545                     'post_author'    => get_current_user_id(),
546                     'post_status'    => 'auto-draft',
547                     'post_content'   => $source->getData( get_post_meta(
548                 )
549             );
550             if ( $chart_id && ! is_wp_error( $chart_id ) ) {
551                 add_post_meta( $chart_id, Visualizer_Plugin::CF_CHART_TYPE
552                 add_post_meta( $chart_id, Visualizer_Plugin::CF_DEFAULT_DA
553                 add_post_meta( $chart_id, Visualizer_Plugin::CF_SOURCE, $s
554                 add_post_meta( $chart_id, Visualizer_Plugin::CF_SERIES, $s
555                 add_post_meta( $chart_id, Visualizer_Plugin::CF_CHART_LIBR
556                 add_post_meta(
557                     $chart_id,
558                     Visualizer_Plugin::CF_SETTINGS,
559                     array(
560                         'focusTarget' => 'datum',
561                     )
562                 );
563
564                 do_action( 'visualizer_pro_new_chart_defaults', $chart_id
565             }
566         } else {
567             if ( current_user_can( 'edit_posts' ) ) {

```

```

568     $parent_chart_id = isset( $_GET['parent_chart_id'] ) ? fil
569     $success = false;
570     if ( $parent_chart_id ) {
571         $parent_chart = get_post( $parent_chart_id );
572         $success = $parent_chart && $parent_chart->post_ty
573     }
574     if ( $success ) {
575         $new_chart_id = wp_insert_post(
576             array(
577                 'post_type' => Visualizer_Plugi
578                 'post_title' => 'Visualization',
579                 'post_author' => get_current_user
580                 'post_status' => $parent_chart->p
581                 'post_content' => $parent_chart->p
582             )
583         );
584
585         if ( is_wp_error( $new_chart_id ) ) {
586             do_action( 'themeisle_log_event', Visualiz
587         } else {
588             $post_meta = get_post_meta( $parent_chart_
589             $chart_id = $new_chart_id;
590             foreach ( $post_meta as $key => $value ) {
591                 if ( strpos( $key, 'visualizer-' )
592                     add_post_meta( $new_chart_
593             }
594         }
595     }
596 }
597 }
598 do_action( 'visualizer_pro_new_chart_defaults', $chart_id );
599 }
600 wp_redirect( esc_url_raw( add_query_arg( 'chart', (int) $chart_id ) ) );
601
602 if ( defined( 'WP_TESTS_DOMAIN' ) ) {
603     wp_die();
604 }
605 exit();
606 }
607
608 $_POST['save_chart_image'] = isset( $_POST['save_chart_image'] ) && 'yes' === $_PO
609 $_POST['lazy_load_chart'] = isset( $_POST['lazy_load_chart'] ) && 'yes' === $_POS
610
611 if ( isset( $_POST['chart-img'] ) && ! empty( $_POST['chart-img'] ) ) {
612     $attachment_id = $this->save_chart_image( $_POST['chart-img'], $chart_id,
613     if ( $attachment_id ) {
614         update_post_meta( $chart_id, Visualizer_Plugin::CF_CHART_IMAGE, $a
615     }
616 }

```

```

617 $lib = $this->load_chart_type( $chart_id );
618
619 // the alpha color picker (RGBA) is not supported by google.
620 $color_picker_dep = 'wp-color-picker';
621 if ( in_array( $lib, array( 'chartjs', 'datatables' ), true ) && ! wp_script_is( '
622     wp_register_script( 'wp-color-picker-alpha', VISUALIZER_ABSURL . 'js/lib/w
623     $color_picker_dep = 'wp-color-picker-alpha';
624 }
625
626 // enqueue and register scripts and styles
627 wp_register_script( 'visualizer-chosen', VISUALIZER_ABSURL . 'js/lib/chosen.jquery
628 wp_register_style( 'visualizer-chosen', VISUALIZER_ABSURL . 'css/lib/chosen.min.cs
629
630 wp_register_style( 'visualizer-frame', VISUALIZER_ABSURL . 'css/frame.css', array(
631 wp_register_script( 'visualizer-frame', VISUALIZER_ABSURL . 'js/frame.js', array(
632 wp_register_script( 'visualizer-customization', $this->get_user_customization_js()
633 wp_register_script(
634     'visualizer-render',
635     VISUALIZER_ABSURL . 'js/render-facade.js',
636     apply_filters( 'visualizer_assets_render', array( 'visualizer-frame', 'vis
637     Visualizer_Plugin::VERSION,
638     true
639 );
640 wp_register_script(
641     'visualizer-preview',
642     VISUALIZER_ABSURL . 'js/preview.js',
643     array(
644         $color_picker_dep,
645         'visualizer-render',
646     ),
647     Visualizer_Plugin::VERSION,
648     true
649 );
650 wp_register_script( 'visualizer-editor-simple', VISUALIZER_ABSURL . 'js/simple-edi
651
652 do_action( 'visualizer_add_scripts' );
653
654 if ( Visualizer_Module::is_pro() && Visualizer_Module::is_pro_older_than( '1.9.0'
655     global $Visualizer_Pro;
656     $Visualizer_Pro->addScriptsAndStyles();
657 }
658
659 // dispatch pages
660 $this->_chart = get_post( $chart_id );
661 $tab = isset( $_GET['tab'] ) && ! empty( $_GET['tab'] ) ? $_GET['tab'] : 'visua
662
663 // skip chart type pages only for existing charts.
664 if ( VISUALIZER_SKIP_CHART_TYPE_PAGE && 'auto-draft' !== $this->_chart->post_statu
665     $tab = 'settings';

```

```

666     }
667
668     if ( isset( $_POST['cancel'] ) && 1 === intval( $_POST['cancel'] ) ) {
669         // if the cancel button is clicked.
670         $this->undoRevisions( $chart_id, true );
671     } elseif ( isset( $_POST['save'] ) && 1 === intval( $_POST['save'] ) ) {
672         $this->handlePermissions();
673         // if the save button is clicked.
674         $this->undoRevisions( $chart_id, false );
675     } else {
676         // if the edit button is clicked.
677         $this->_chart = $this->handleExistingRevisions( $chart_id, $this->_chart );
678     }
679
680     // Clear existing chart cache.
681     if ( isset( $_POST['save'] ) && 1 === intval( $_POST['save'] ) ) {
682         $cache_key = Visualizer_Plugin::CF_CHART_CACHE . '_' . $chart_id;
683         if ( get_transient( $cache_key ) ) {
684             delete_transient( $cache_key );
685         }
686     }
687
688     switch ( $tab ) {
689         case 'settings':
690             $this->_handleDataAndSettingsPage();
691             break;
692         case 'type': // fall through.
693         case 'visualizer': // fall through.
694             $this->_handleTypesPage();
695             break;
696         default:
697             // this should never happen.
698             break;
699     }
700     defined( 'WP_TESTS_DOMAIN' ) ? wp_die() : exit();
701 }
702
703 /**
704  * Load code editor assets.
705  */
706 private function loadCodeEditorAssets( $chart_id ) {
707     global $wp_version;
708
709     $wp_scripts = wp_scripts();
710
711     // data tables assets.
712     wp_register_script( 'visualizer-datatables', VISUALIZER_ABSURL . 'js/lib/datatable
713     wp_register_style( 'visualizer-datatables', VISUALIZER_ABSURL . 'css/lib/datatable
714     wp_register_style( 'visualizer-jquery-ui', sprintf( '//code.jquery.com/ui/%s/theme

```

```

715 wp_enqueue_script( 'visualizer-datatables' );
716 wp_enqueue_style( 'visualizer-jquery-ui' );
717
718 if ( ! Visualizer_Module::is_pro() ) {
719     return;
720 }
721
722 $table_col_mapping = Visualizer_Source_Query_Params::get_all_db_tables_column_map
723
724 if ( version_compare( $wp_version, '4.9.0', '<' ) ) {
725     // code mirror assets.
726     wp_register_script( 'visualizer-codemirror-core', '//codemirror.net/lib/co
727     wp_register_script( 'visualizer-codemirror-placeholder', '//codemirror.net
728     wp_register_script( 'visualizer-codemirror-matchbrackets', '//codemirror.n
729     wp_register_script( 'visualizer-codemirror-closebrackets', '//codemirror.n
730     wp_register_script( 'visualizer-codemirror-sql', '//codemirror.net/mode/sq
731     wp_register_script( 'visualizer-codemirror-sql-hint', '//codemirror.net/ad
732     wp_register_script( 'visualizer-codemirror-hint', '//codemirror.net/addon/
733     wp_register_style( 'visualizer-codemirror-core', '//codemirror.net/lib/cod
734     wp_register_style( 'visualizer-codemirror-hint', '//codemirror.net/addon/h
735
736     wp_enqueue_script( 'visualizer-codemirror-hint' );
737     wp_enqueue_style( 'visualizer-codemirror-hint' );
738 } else {
739     wp_enqueue_code_editor(
740         array(
741             'type' => 'sql',
742             'codemirror' => array(
743                 'autofocus' => true,
744                 'lineWrapping' => true,
745                 'dragDrop' => false,
746                 'matchBrackets' => true,
747                 'autoCloseBrackets' => true,
748                 'extraKeys' => array( 'Ctrl-Space' => 'aut
749                 'hintOptions' => array( 'tables' => $table_c
750             ),
751         )
752     );
753 }
754
755 return $table_col_mapping;
756 }
757
758 /**
759  * Handle permissions from the new conslidated settings sidebar.
760  */
761 private function handlePermissions() {
762     if ( ! Visualizer_Module::is_pro() ) {
763         return;

```



```

764     }
765
766     // we will not support old free and new pro.
767     // handling new free and old pro.
768     if ( has_action( 'visualizer_pro_handle_tab' ) ) {
769         do_action( 'visualizer_pro_handle_tab', 'permissions', $this->_chart );
770     } else {
771         do_action( 'visualizer_handle_permissions', $this->_chart );
772     }
773 }
774
775 /**
776  * Handle data and settings page
777  */
778 private function _handleDataAndSettingsPage() {
779     if ( isset( $_POST['map_api_key'] ) ) {
780         update_option( 'visualizer-map-api-key', $_POST['map_api_key'] );
781     }
782
783     if ( $_SERVER['REQUEST_METHOD'] === 'POST' && isset( $_GET['nonce'] ) && wp_verify_nonce(
784         $this->_chart->post_status === 'auto-draft' ) {
785         $this->_chart->post_status = 'publish';
786
787         // ensure that a revision is not created. If a revision is created
788         // we do not want any difference in data so disable revisions temp
789         $this->disableRevisionsTemporarily();
790
791         wp_update_post( $this->_chart->to_array() );
792     }
793     // save meta data only when it is NOT being canceled.
794     if ( ! ( isset( $_POST['cancel'] ) && 1 === intval( $_POST['cancel'] ) ) ) {
795         update_post_meta( $this->_chart->ID, Visualizer_Plugin::CF_SETTING
796
797         // we will keep a parameter called 'internal_title' that will be s
798         // this will help in searching with the chart id.
799         $settings = get_post_meta( $this->_chart->ID, Visualizer_Plugin::C
800         $title = null;
801         if ( isset( $settings['title'] ) && ! empty( $settings['title'] ) )
802             $title = $settings['title'];
803         if ( is_array( $title ) ) {
804             $title = $settings['title']['text'];
805         }
806     }
807     if ( empty( $title ) ) {
808         $title = $this->_chart->ID;
809     }
810     $settings['internal_title'] = $title;
811     $settings_label = isset( $settings['pieResidueSliceLabel'] ) ? $se
812     if ( empty( $settings_label ) ) {

```

```

813         $settings['pieResidueSliceLabel'] = esc_html__( 'Other', '
814     } else {
815         $settings['pieResidueSliceLabel'] = $settings_label;
816     }
817     update_post_meta( $this->_chart->ID, Visualizer_Plugin::CF_SETTING
818 }
819 $render      = new Visualizer_Render_Page_Send();
820 $render->text = sprintf( '[visualizer id="%d"]', $this->_chart->ID );
821 wp_iframe( array( $render, 'render' ) );
822 return;
823 }
824 $data      = $this->_getChartArray();
825 $sidebar    = '';
826 $sidebar_class = $this->load_chart_class_name( $this->_chart->ID );
827 if ( class_exists( $sidebar_class, true ) ) {
828     $sidebar      = new $sidebar_class( $data['settings'] );
829     $sidebar->__series = $data['series'];
830     $sidebar->__data   = $data['data'];
831 } else {
832     $sidebar = apply_filters( 'visualizer_pro_chart_type_sidebar', '', $data );
833     if ( $sidebar !== '' ) {
834         $sidebar->__series = $data['series'];
835         $sidebar->__data   = $data['data'];
836     }
837 }
838 unset( $data['settings']['width'], $data['settings']['height'], $data['settings']['
839 wp_enqueue_style( 'wp-color-picker' );
840 wp_enqueue_style( 'visualizer-frame' );
841 wp_enqueue_script( 'visualizer-preview' );
842 wp_enqueue_script( 'visualizer-chosen' );
843 wp_enqueue_script( 'visualizer-render' );
844
845 if ( Visualizer_Module::can_show_feature( 'simple-editor' ) ) {
846     wp_enqueue_script( 'visualizer-editor-simple' );
847     wp_localize_script(
848         'visualizer-editor-simple',
849         'visualizer1',
850         array(
851             'ajax'      => array(
852                 'url'      => admin_url( 'admin-ajax.php' ),
853                 'nonces' => array(
854                     ),
855                 'actions' => array(
856                     ),
857             ),
858         )
859     );
860 }
861

```

```

862 $table_col_mapping = $this->loadCodeEditorAssets( $this->_chart->ID );
863
864 wp_localize_script(
865     'visualizer-render',
866     'visualizer',
867     array(
868         'l10n' => array(
869             'invalid_source' => esc_html__( 'You have entered an inval
870             'loading'       => esc_html__( 'Loading...', 'visualizer'
871             'json_error'    => esc_html__( 'An error occured in fetchi
872             'select_columns' => esc_html__( 'Please select a few co
873             'save_settings' => __( 'You have modified the chart\'s
874         ),
875         'charts' => array(
876             'canvas' => $data,
877             'id' => $this->_chart->ID,
878         ),
879         'language' => $this->get_language(),
880         'map_api_key' => get_option( 'visualizer-map-api-key' ),
881         'ajax' => array(
882             'url' => admin_url( 'admin-ajax.php' ),
883             'nonces' => array(
884                 'permissions' => wp_create_nonce( Visualizer_Plu
885                 'db_get_data' => wp_create_nonce( Visualizer_Plu
886                 'json_get_roots' => wp_create_nonce( Visualizer_
887                 'json_get_data' => wp_create_nonce( Visualizer_P
888                 'json_set_schedule' => wp_create_nonce( Visualiz
889             ),
890             'actions' => array(
891                 'permissions' => Visualizer_Plugin::ACTION_FETCH
892                 'db_get_data' => Visualizer_Plugin::ACTION_FETCH
893                 'json_get_roots' => Visualizer_Plugin::ACTION_JS
894                 'json_get_data' => Visualizer_Plugin::ACTION_JS
895                 'json_set_schedule' => Visualizer_Plugin::ACTION
896             ),
897         ),
898         'db_query' => array(
899             'tables' => $table_col_mapping,
900         ),
901         'is_pro' => Visualizer_Module::is_pro(),
902         'page_type' => 'chart',
903         'json_tag_separator' => Visualizer_Source_Json::TAG_SEPARATOR,
904         'json_tag_separator_view' => Visualizer_Source_Json::TAG_SEPARATOR,
905         'is_front' => false,
906         'rest_base' => get_rest_url( null, 'wc/v3/reports/' ),
907     )
908 );
909
910 $render = new Visualizer_Render_Page_Data();

```

```

911     $render->chart    = $this->_chart;
912     $render->type     = $data['type'];
913     $render->custom_css = $data['css'];
914     $render->sidebar  = $sidebar;
915     if ( filter_input( INPUT_GET, 'library', FILTER_VALIDATE_BOOLEAN ) ) {
916         $render->button = filter_input( INPUT_GET, 'action' ) === Visualizer_Plugin
917             ? esc_html__( 'Save Chart', 'visualizer' )
918             : esc_html__( 'Create Chart', 'visualizer' );
919         if ( filter_input( INPUT_GET, 'action' ) === Visualizer_Plugin::ACTION_EDIT
920             $render->cancel_button = esc_html__( 'Cancel', 'visualizer' );
921     }
922 } else {
923     $render->button = esc_attr__( 'Insert Chart', 'visualizer' );
924 }
925
926 do_action( 'visualizer_enqueue_scripts_and_styles', $data, $this->_chart->ID );
927
928 if ( Visualizer_Module::is_pro() && Visualizer_Module::is_pro_older_than( '1.9.0' )
929     global $Visualizer_Pro;
930     $Visualizer_Pro->enqueueScriptsAndStyles( $data, $this->_chart->ID );
931 }
932
933 $this->_addAction( 'admin_head', 'renderFlattrScript' );
934 wp_iframe( array( $render, 'render' ) );
935 }
936
937 /**
938  * Handles chart type selection page.
939  *
940  * @since 1.0.0
941  *
942  * @access private
943  */
944 private function _handleTypesPage() {
945     // process post request
946     if ( $_SERVER['REQUEST_METHOD'] === 'POST' && wp_verify_nonce( filter_input( INPUT
947         $type = filter_input( INPUT_POST, 'type' );
948         $library = filter_input( INPUT_POST, 'chart-library' );
949         if ( in_array( $type, Visualizer_Plugin::getChartTypes(), true ) ) {
950             if ( empty( $library ) ) {
951                 // library cannot be empty.
952                 do_action( 'themeisle_log_event', Visualizer_Plugin::NAME,
953                     return;
954             }
955
956             // save new chart type
957             update_post_meta( $this->_chart->ID, Visualizer_Plugin::CF_CHART_T
958             update_post_meta( $this->_chart->ID, Visualizer_Plugin::CF_CHART_L
959             // if the chart has default data, update it with appropriate defau

```

```

960         if ( filter_var( get_post_meta( $this->_chart->ID, Visualizer_Plug
961             $source = new Visualizer_Source_Csv( VISUALIZER_ABSPATH .
962             $source->fetch();
963             $this->_chart->post_content = $source->getData( get_post_m
964             wp_update_post( $this->_chart->to_array() );
965             update_post_meta( $this->_chart->ID, Visualizer_Plugin::CF
966         }
967
968         Visualizer_Module_Utility::set_defaults( $this->_chart );
969
970         // redirect to next tab
971         // changed by Ash/Upwork
972         wp_redirect( esc_url_raw( add_query_arg( 'tab', 'settings' ) ) );
973
974         return;
975     }
976 }
977 $render = new Visualizer_Render_Page_Types();
978 $render->type = get_post_meta( $this->_chart->ID, Visualizer_Plugin::CF_CHART_TYP
979 $render->types = Visualizer_Module_Admin::_getChartTypesLocalized( false, false, f
980 $render->chart = $this->_chart;
981 wp_enqueue_style( 'visualizer-frame' );
982 wp_enqueue_script( 'visualizer-frame' );
983 wp_iframe( array( $render, 'render' ) );
984 }
985
986 /**
987  * Renders flattr script in the iframe <head>
988  *
989  * @since 1.4.2
990  * @action admin_head
991  *
992  * @access public
993  */
994 public function renderFlattrScript() {
995     echo '';
996 }
997
998 /**
999  * Processes the CSV that is sent in the request as a string.
1000  *
1001  * @since 3.2.0
1002  */
1003 private function handleCSVasString( $data, $editor_type ) {
1004     $source = null;
1005
1006     switch ( $editor_type ) {
1007         case 'text':
1008             // data coming in from the text editor.

```

```

1009         $tmpfile = tempnam( get_temp_dir(), Visualizer_Plugin::NAME );
1010         $handle = fopen( $tmpfile, 'w' );
1011         $values = preg_split( ' /\n\r+/', stripslashes( trim( $data ) ) )
1012         if ( $values ) {
1013             foreach ( $values as $row ) {
1014                 if ( empty( $row ) ) {
1015                     continue;
1016                 }
1017                 $row = explode( ',', $row );
1018                 $row = array_map(
1019                     function( $r ) {
1020                         return '' === $r ? ' ' : $r;
1021                     },
1022                     $row
1023                 );
1024                 $row = implode( ',', $row );
1025                 // don't use fpucsv here because we need to just d
1026                 // minus the empty rows
1027                 // because fputcsv needs to tokenize
1028                 // we can standardize the CSV enclosure here and r
1029                 // we can assume that if there are an even number
1030                 // but that will screw up words like fo'c'sle
1031                 // so here let's just assume ' will NOT be used fo
1032                 fwrite( $handle, $row );
1033                 fwrite( $handle, PHP_EOL );
1034             }
1035         }
1036         $source = new Visualizer_Source_Csv( $tmpfile );
1037         fclose( $handle );
1038         break;
1039     case 'table':
1040         // Import from Chart
1041         // fall-through.
1042     case 'excel':
1043         // data coming in from the excel editor.
1044         $source = apply_filters( 'visualizer_pro_handle_chart_data', $data
1045         break;
1046     }
1047     return $source;
1048 }
1049
1050 /**
1051  * Parses the data uploaded as an HTML table.
1052  *
1053  * @since 3.2.0
1054  *
1055  * @access private
1056  */
1057 private function handleTabularData() {

```

```

1058     $csv      = array();
1059     // the datatable mentions the headers twice, so lets remove the duplicates.
1060     $headers   = array_unique( array_filter( $_POST['header'] ) );
1061     $types     = $_POST['type'];
1062
1063     // capture all the indexes that correspond to excluded columns.
1064     $exclude   = array();
1065     $index     = 0;
1066     foreach ( $types as $type ) {
1067         if ( empty( $type ) ) {
1068             $exclude[] = $index;
1069         }
1070         $index++;
1071     }
1072
1073     // when N headers are being renamed, the number of headers increases by N
1074     // because of the way datatable duplicates header information
1075     // so unset the headers that have been renamed.
1076     if ( count( $headers ) != count( $types ) ) {
1077         $to = count( $headers );
1078         for ( $i = count( $types ); $i < $to; $i++ ) {
1079             unset( $headers[ $i + 1 ] );
1080         }
1081     }
1082
1083     $columns    = array();
1084     for ( $i = 0; $i < count( $headers ); $i++ ) {
1085         if ( ! isset( $_POST[ 'data' . $i ] ) ) {
1086             continue;
1087         }
1088         $columns[ $i ] = $_POST[ 'data' . $i ];
1089     }
1090
1091     $csv[]      = $headers;
1092     $csv[]      = $types;
1093     for ( $j = 0; $j < count( $columns[0] ); $j++ ) {
1094         $row = array();
1095         for ( $i = 0; $i < count( $headers ); $i++ ) {
1096             $row[] = sanitize_text_field( $columns[ $i ][ $j ] );
1097         }
1098         $csv[] = $row;
1099     }
1100
1101     $tmpfile = tempnam( get_temp_dir(), Visualizer_Plugin::NAME );
1102     $handle  = fopen( $tmpfile, 'w' );
1103
1104     if ( $csv ) {
1105         $index = 0;
1106         foreach ( $csv as $row ) {

```

```

1107         // remove all the cells corresponding to the excluded headers.
1108         foreach ( $exclude as $j ) {
1109             unset( $row[ $j ] );
1110         }
1111         fputcsv( $handle, $row );
1112     }
1113 }
1114 $source = new Visualizer_Source_Csv( $tmpfile );
1115 fclose( $handle );
1116 return $source;
1117 }
1118
1119 /**
1120  * Parses uploaded CSV file and saves new data for the chart.
1121  *
1122  * @since 1.0.0
1123  *
1124  * @access public
1125  */
1126 public function uploadData() {
1127     // if this is being called internally from pro and VISUALIZER_DO_NOT_DIE is set.
1128     // otherwise, assume this is a normal web request.
1129     $can_die = ! ( defined( 'VISUALIZER_DO_NOT_DIE' ) && VISUALIZER_DO_NOT_DIE );
1130
1131     // validate nonce
1132     if ( ! isset( $_GET['nonce'] ) || ! wp_verify_nonce( $_GET['nonce'] ) ) {
1133         if ( ! $can_die ) {
1134             return;
1135         }
1136         status_header( 403 );
1137         exit;
1138     }
1139
1140     // check chart, if chart exists
1141     // do not use filter_input as it does not work for phpunit test cases, use filter_var
1142     $chart_id = isset( $_GET['chart'] ) ? filter_var( $_GET['chart'], FILTER_VALIDATE_INT ) : 0;
1143     if ( ! $chart_id || ! ( $chart = get_post( $chart_id ) ) || $chart->post_type !== 'chart' ) {
1144         if ( ! $can_die ) {
1145             return;
1146         }
1147         status_header( 400 );
1148         exit;
1149     }
1150
1151     do_action( 'themeisle_log_event', Visualizer_Plugin::NAME, sprintf( 'Uploading data for chart %d', $chart_id ) );
1152
1153     if ( ! isset( $_POST['vz-import-time'] ) ) {
1154         apply_filters( 'visualizer_pro_remove_schedule', $chart_id );
1155     }

```



```

1156
1157     if ( ! isset( $_POST['chart_data_src'] ) ) || Visualizer_Plugin::CF_SOURCE_FILTER !=
1158         // delete the filters in case this chart is being uploaded from other data
1159         delete_post_meta( $chart_id, Visualizer_Plugin::CF_FILTER_CONFIG );
1160         delete_post_meta( $chart_id, '__transient-' . Visualizer_Plugin::CF_FILTER
1161         delete_post_meta( $chart_id, '__transient-' . Visualizer_Plugin::CF_DB_QUE
1162
1163         // delete "import from db" specific parameters.
1164         delete_post_meta( $chart_id, Visualizer_Plugin::CF_DB_QUERY );
1165         delete_post_meta( $chart_id, Visualizer_Plugin::CF_DB_SCHEDULE );
1166         delete_post_meta( $chart_id, Visualizer_Plugin::CF_REMOTE_DB_PARAMS );
1167     }
1168
1169     // delete json related data.
1170     delete_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_URL );
1171     delete_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_ROOT );
1172     delete_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_PAGING );
1173     delete_post_meta( $chart_id, Visualizer_Plugin::CF_JSON_HEADERS );
1174
1175     // delete last error
1176     delete_post_meta( $chart_id, Visualizer_Plugin::CF_ERROR );
1177
1178     // delete editor related data.
1179     delete_post_meta( $chart_id, Visualizer_Plugin::CF_EDITOR );
1180
1181     // delete this so that a JSON import can be later edited manually without a proble
1182     delete_post_meta( $chart_id, Visualizer_Plugin::CF_EDITABLE_TABLE );
1183
1184     $source = null;
1185     $render = new Visualizer_Render_Page_Update();
1186
1187     $remote_data = false;
1188     if ( isset( $_POST['remote_data'] ) && function_exists( 'wp_http_validate_url' ) )
1189         $remote_data = wp_http_validate_url( $_POST['remote_data'] );
1190     }
1191     if ( false !== $remote_data ) {
1192         $source = new Visualizer_Source_Csv_Remote( $remote_data );
1193         if ( isset( $_POST['vz-import-time'] ) ) {
1194             apply_filters( 'visualizer_pro_chart_schedule', $chart_id, $remote
1195         }
1196         // phpcs:ignore WordPress.PHP.StrictComparisons.LooseComparison
1197     } elseif ( isset( $_FILES['local_data'] ) && $_FILES['local_data']['error'] == 0 )
1198         $source = new Visualizer_Source_Csv( $_FILES['local_data']['tmp_name'] );
1199     } elseif ( isset( $_POST['chart_data'] ) && strlen( $_POST['chart_data'] ) > 0 ) {
1200         $source = $this->handleCSVasString( $_POST['chart_data'], $_POST['editor-t
1201         update_post_meta( $chart_id, Visualizer_Plugin::CF_EDITOR, $_POST['editor-
1202     } elseif ( isset( $_POST['table_data'] ) && 'yes' === $_POST['table_data'] ) {
1203         $source = $this->handleTabularData();
1204         update_post_meta( $chart_id, Visualizer_Plugin::CF_EDITOR, $_POST['editor-

```

```

1205     } else {
1206         do_action( 'themeisle_log_event', Visualizer_Plugin::NAME, sprintf( 'CSV f
1207         $render->message = esc_html__( 'CSV file with chart data was not uploaded.
1208         update_post_meta( $chart_id, Visualizer_Plugin::CF_ERROR, esc_html__( 'CSV
1209     }
1210
1211     do_action( 'themeisle_log_event', Visualizer_Plugin::NAME, sprintf( 'Uploaded data
1212
1213     if ( $source ) {
1214         if ( $source->fetch() ) {
1215             $content = $source->getData( get_post_meta( $chart_id, Visualiz
1216             $populate = true;
1217             if ( is_string( $content ) && is_array( unserialize( $content ) )
1218                 $json = unserialize( $content );
1219             // if source exists, so should data. if source exists but
1220             // if we populate the data even if it is empty, the chart
1221             if ( array_key_exists( 'source', $json ) && ! empty( $json
1222                 do_action( 'themeisle_log_event', Visualizer_Plugin
1223                 update_post_meta( $chart_id, Visualizer_Plugin::CF
1224                 $populate = false;
1225             }
1226         }
1227         if ( $populate ) {
1228             $chart->post_content = $content;
1229         }
1230         wp_update_post( $chart->to_array() );
1231
1232         update_post_meta( $chart->ID, Visualizer_Plugin::CF_SERIES, $sourc
1233         update_post_meta( $chart->ID, Visualizer_Plugin::CF_SOURCE, $sourc
1234         update_post_meta( $chart->ID, Visualizer_Plugin::CF_DEFAULT_DATA,
1235
1236         do_action( 'themeisle_log_event', Visualizer_Plugin::NAME, sprintf
1237
1238         Visualizer_Module_Utility::set_defaults( $chart, null );
1239
1240         $settings = get_post_meta( $chart->ID, Visualizer_Plugin::CF_SETTI
1241         if ( isset( $settings['series'] ) && ! ( count( $settings['series'
1242             $diff_total_series = abs( count( $settings['series'] ) - c
1243             if ( $diff_total_series ) {
1244                 foreach ( range( 1, $diff_total_series ) as $k =>
1245                     $settings['series'][] = end( $settings['se
1246                 }
1247                 update_post_meta( $chart->ID, Visualizer_Plugin::C
1248             }
1249         }
1250
1251         $render->id = $chart->ID;
1252         $render->data = json_encode( $source->getRawData( get_post_meta(
1253         $render->series = json_encode( $source->getSeries() );

```

```

1254         $render->settings = json_encode( $settings );
1255     } else {
1256         $error = $source->get_error();
1257         if ( empty( $error ) ) {
1258             $error = esc_html__( 'CSV file is broken or invalid. Pleas
1259         }
1260         $render->message = $error;
1261         do_action( 'themeisle_log_event', Visualizer_Plugin::NAME, sprintf
1262             update_post_meta( $chart_id, Visualizer_Plugin::CF_ERROR, $error )
1263     }
1264 } else {
1265     do_action( 'themeisle_log_event', Visualizer_Plugin::NAME, sprintf( 'Unkno
1266 }
1267 $render->render();
1268 if ( ! $can_die ) {
1269     return;
1270 }
1271 defined( 'WP_TESTS_DOMAIN' ) ? wp_die() : exit();
1272 }
1273
1274 /**
1275  * Clones the chart.
1276  *
1277  * @since 1.0.0
1278  *
1279  * @access public
1280  */
1281 public function cloneChart() {
1282     $chart_id = $success = false;
1283     $nonce     = isset( $_GET['nonce'] ) && wp_verify_nonce( $_GET['nonce'], Visualizer
1284     $capable   = current_user_can( 'edit_posts' );
1285     if ( $nonce && $capable ) {
1286         $chart_id = isset( $_GET['chart'] ) ? filter_var( $_GET['chart'], FILTER_V
1287         if ( $chart_id ) {
1288             $chart = get_post( $chart_id );
1289             $success = $chart && $chart->post_type === Visualizer_Plugin::CPT_
1290         }
1291     }
1292     $redirect = remove_query_arg( 'vaction', wp_get_referer() );
1293     if ( $success ) {
1294         $new_chart_id = wp_insert_post(
1295             array(
1296                 'post_type'   => Visualizer_Plugin::CPT_VISUALIZER,
1297                 'post_title'  => 'Visualization',
1298                 'post_author' => get_current_user_id(),
1299                 'post_status' => $chart->post_status,
1300                 'post_content' => $chart->post_content,
1301             )
1302     );

```

```

1303         if ( is_wp_error( $new_chart_id ) ) {
1304             do_action( 'themeisle_log_event', Visualizer_Plugin::NAME, sprintf
1305         } else {
1306             $post_meta = get_post_meta( $chart_id );
1307             foreach ( $post_meta as $key => $value ) {
1308                 if ( strpos( $key, 'visualizer-' ) !== false ) {
1309                     add_post_meta( $new_chart_id, $key, maybe_unserial
1310                 }
1311             }
1312             $redirect = esc_url(
1313                 add_query_arg(
1314                     array(
1315                         'page' => 'visualizer',
1316                         'type' => filter_input( INPUT_GET, 'type'
1317                         'vaction' => false,
1318                     ),
1319                     admin_url( 'admin.php' )
1320                 ),
1321                 null,
1322                 'db'
1323             );
1324         }
1325     }
1326
1327     if ( defined( 'WP_TESTS_DOMAIN' ) ) {
1328         wp_die();
1329     }
1330     wp_redirect( $redirect );
1331     exit;
1332 }
1333
1334 /**
1335  * Exports the chart data
1336  *
1337  * @since 1.0.0
1338  *
1339  * @access public
1340  */
1341 public function exportData() {
1342     check_ajax_referer( Visualizer_Plugin::ACTION_EXPORT_DATA . Visualizer_Plugin::VER
1343     $capable = current_user_can( 'edit_posts' );
1344     if ( $capable ) {
1345         $chart_id = isset( $_GET['chart'] ) ? filter_var(
1346             $_GET['chart'],
1347             FILTER_VALIDATE_INT,
1348             array(
1349                 'options' => array(
1350                     'min_range' => 1,
1351                 ),

```

```

1352         )
1353     ) : '';
1354     if ( $chart_id ) {
1355         $data = $this->_getDataAs( $chart_id, 'csv' );
1356         if ( $data ) {
1357             echo wp_send_json_success( $data );
1358         }
1359     }
1360 }
1361
1362 defined( 'WP_TESTS_DOMAIN' ) ? wp_die() : exit();
1363 }
1364
1365 /**
1366  * Handles chart data page.
1367  *
1368  * @since 1.0.0
1369  *
1370  * @access private
1371  */
1372 private function _handleDataPage() {
1373     $data = $this->_getChartArray();
1374     $render = new Visualizer_Render_Page_Data();
1375     $render->chart = $this->_chart;
1376     $render->type = $data['type'];
1377
1378     if ( $data && $data['settings'] ) {
1379         unset( $data['settings']['width'], $data['settings']['height'], $data['set
1380     }
1381     wp_enqueue_style( 'visualizer-frame' );
1382     wp_enqueue_script( 'visualizer-render' );
1383     wp_localize_script(
1384         'visualizer-render',
1385         'visualizer',
1386         array(
1387             'l10n' => array(
1388                 'invalid_source' => esc_html__( 'You have entered an inval
1389                 'loading' => esc_html__( 'Loading...', 'visualizer'
1390             ),
1391             'charts' => array(
1392                 'canvas' => $data,
1393             ),
1394         )
1395     );
1396
1397     do_action( 'visualizer_enqueue_scripts_and_styles', $data, $this->_chart->ID );
1398
1399     if ( Visualizer_Module::is_pro() && Visualizer_Module::is_pro_older_than( '1.9.0'
1400         global $Visualizer_Pro;

```

```

1401         $Visualizer_Pro->_enqueueScriptsAndStyles( $data, $this->_chart->ID );
1402     }
1403
1404     // Added by Ash/Upwork
1405     $this->_addAction( 'admin_head', 'renderFlattrScript' );
1406     wp_iframe( array( $render, 'render' ) );
1407 }
1408
1409 /**
1410  * Returns the data for the query.
1411  *
1412  * @access public
1413  */
1414 public function getQueryData() {
1415     check_ajax_referer( Visualizer_Plugin::ACTION_FETCH_DB_DATA . Visualizer_Plugin::V
1416
1417     $params      = wp_parse_args( $_POST['params'] );
1418     $chart_id    = filter_var( $params['chart_id'], FILTER_VALIDATE_INT );
1419
1420     $source      = new Visualizer_Source_Query( stripslashes( $params['query'] ), $char
1421     $html        = $source->fetch( true );
1422     $error       = $source->get_error();
1423     if ( ! empty( $error ) ) {
1424         wp_send_json_error( array( 'msg' => $error ) );
1425     }
1426     wp_send_json_success( array( 'table' => $html ) );
1427 }
1428
1429 /**
1430  * Saves the query and the schedule.
1431  *
1432  * @access public
1433  */
1434 public function saveQuery() {
1435     check_ajax_referer( Visualizer_Plugin::ACTION_SAVE_DB_QUERY . Visualizer_Plugin::V
1436
1437     $chart_id    = filter_input(
1438         INPUT_GET,
1439         'chart',
1440         FILTER_VALIDATE_INT,
1441         array(
1442             'options' => array(
1443                 'min_range' => 1,
1444             ),
1445         )
1446     );
1447
1448     $hours = filter_input(
1449         INPUT_POST,

```

```

1450         'refresh',
1451         FILTER_VALIDATE_FLOAT,
1452         array(
1453             'options' => array(
1454                 'min_range' => -1,
1455                 'max_range' => apply_filters( 'visualizer_is_business', fa
1456             ),
1457         )
1458     );
1459
1460     if ( ! is_numeric( $hours ) ) {
1461         $hours = -1;
1462     }
1463
1464     $render = new Visualizer_Render_Page_Update();
1465     if ( $chart_id ) {
1466         $params      = wp_parse_args( $_POST['params'] );
1467         $source      = new Visualizer_Source_Query( stripslashes( $params['query'] )
1468         $source->fetch( false );
1469         $error       = $source->get_error();
1470         if ( empty( $error ) ) {
1471             update_post_meta( $chart_id, Visualizer_Plugin::CF_DB_QUERY, strip
1472             update_post_meta( $chart_id, Visualizer_Plugin::CF_SOURCE, $source
1473             update_post_meta( $chart_id, Visualizer_Plugin::CF_SERIES, $source
1474             update_post_meta( $chart_id, Visualizer_Plugin::CF_DB_SCHEDULE, $h
1475             update_post_meta( $chart_id, Visualizer_Plugin::CF_DEFAULT_DATA, 0
1476             if ( isset( $params['db_type'] ) && $params['db_type'] !== Visuali
1477                 $remote_db_params = $params;
1478                 unset( $remote_db_params['query'] );
1479                 unset( $remote_db_params['chart_id'] );
1480                 update_post_meta( $chart_id, Visualizer_Plugin::CF_REMOTE_
1481             }
1482
1483             $schedules      = get_option( Visualizer_Plugin::CF_DB_SCH
1484             $schedules[ $chart_id ] = time() + $hours * HOUR_IN_SECONDS;
1485             update_option( Visualizer_Plugin::CF_DB_SCHEDULE, $schedules );
1486
1487             wp_update_post(
1488                 array(
1489                     'ID'           => $chart_id,
1490                     'post_content' => $source->getData( get_post_meta
1491                 )
1492             );
1493             $render->data = json_encode( $source->getRawData( get_post_meta(
1494             $render->series = json_encode( $source->getSeries() );
1495             $render->id = $chart_id;
1496         } else {
1497             $render->message = $error;
1498         }

```

```

1499     }
1500     $render->render();
1501     if ( ! ( defined( 'VISUALIZER_DO_NOT_DIE' ) && VISUALIZER_DO_NOT_DIE ) ) {
1502         defined( 'WP_TESTS_DOMAIN' ) ? wp_die() : exit();
1503     }
1504 }
1505
1506
1507 /**
1508  * Saves the filter query and the schedule.
1509  *
1510  * @access public
1511  */
1512 public function saveFilter() {
1513     check_ajax_referer( Visualizer_Plugin::ACTION_SAVE_FILTER_QUERY . Visualizer_Plugi
1514
1515     $chart_id = filter_input(
1516         INPUT_GET,
1517         'chart',
1518         FILTER_VALIDATE_INT,
1519         array(
1520             'options' => array(
1521                 'min_range' => 1,
1522             ),
1523         )
1524     );
1525
1526     $hours = filter_input(
1527         INPUT_POST,
1528         'refresh',
1529         FILTER_VALIDATE_INT,
1530         array(
1531             'options' => array(
1532                 'min_range' => -1,
1533                 'max_range' => apply_filters( 'visualizer_is_business', fa
1534             ),
1535         )
1536     );
1537
1538     if ( 0 !== $hours && empty( $hours ) ) {
1539         $hours = -1;
1540     }
1541
1542     do_action( 'visualizer_save_filter', $chart_id, $hours );
1543
1544     if ( ! ( defined( 'VISUALIZER_DO_NOT_DIE' ) && VISUALIZER_DO_NOT_DIE ) ) {
1545         defined( 'WP_TESTS_DOMAIN' ) ? wp_die() : exit();
1546     }
1547 }

```



```

1548
1549 /**
1550  * Save chart image.
1551  *
1552  * @param string $base64_img Chart image.
1553  * @param int    $chart_id Chart ID.
1554  * @param bool   $save_attachment Save attachment.
1555  * @return attachment ID
1556  */
1557 public function save_chart_image( $base64_img, $chart_id, $save_attachment = true ) {
1558     // Delete old chart image.
1559     $old_attachment_id = get_post_meta( $chart_id, Visualizer_Plugin::CF_CHART_IMAGE,
1560     if ( $old_attachment_id ) {
1561         wp_delete_attachment( $old_attachment_id, true );
1562     }
1563
1564     if ( ! $save_attachment ) {
1565         return 0;
1566     }
1567
1568     // Upload dir.
1569     $upload_dir = wp_upload_dir();
1570     $upload_path = str_replace( '/', DIRECTORY_SEPARATOR, $upload_dir['path'] ) . DIRE
1571
1572     $img          = str_replace( 'data:image/png;base64,', '', $base64_img );
1573     $img          = str_replace( ' ', '+', $img );
1574     $decoded      = base64_decode( $img );
1575     $filename     = 'visualization-' . $chart_id . '.png';
1576     $file_type    = 'image/png';
1577     $hashed_filename = $filename;
1578
1579     // Save the image in the uploads directory.
1580     require_once ABSPATH . '/wp-admin/includes/file.php';
1581     \WP_Filesystem();
1582     global $wp_filesystem;
1583     $upload_file = $wp_filesystem->put_contents( $upload_path . $hashed_filename, $dec
1584
1585     // Insert new chart image.
1586     $attachment = array(
1587         'post_mime_type' => $file_type,
1588         'post_title'     => preg_replace( '/\[^\.]+' . $/, '', basename( $hashed_file
1589         'post_content'   => '',
1590         'post_status'    => 'inherit',
1591         'guid'           => $upload_dir['url'] . '/' . basename( $hashed_filename
1592     );
1593
1594     $attach_id = wp_insert_attachment( $attachment, $upload_dir['path'] . '/' . $hashe
1595     return $attach_id;
1596 }

```

