Relazione progetto Produzione e Progettazione Multimediale

Mihail Teodor Gurzu
5940299

20 giugno 2020

1 Introduzione

L'elaborato svolto consiste nell'implementazione di un plugin wordpress per la creazione di un blog stile pinterest, integrato nel backend che utilizzi la libreria javascript Shuffle.js.

2 Funzionalità

Il plugin raccoglie per ogni post pubblicato l'immagine di copertina (se un post non ha l'immagine di copertina non verrà visualizzato) e le dispone in una galleria implementata con la libreria javascript Shuffle.js, grazie alla quale è possibile:

- ordinare le immagini in base alla data di pubblicazione del post oppure in base al titolo (default = date).
- filtrare le immagini in base alla categoria di appartenenza.
- è possibile inoltre ricercare un post preciso digitandone il nome.
- cliccando sul nome del post si aprirà una nuova finestra con il post corrispondente.

Il plugin genera uno shortcode **[ShuffleGallery]** che può essere inserito in una qualsiasi pagina o post. Inoltre possono essere utilizzati i seguenti shortcode per gestire quali post/categorie visualizzare o per inserire più gallerie personalizzate nella stessa pagina o post:

- **[ShuffleGallery category="id1,id2,..."]** - solo i post appartenenti alle categorie indicate verranno visualizzati. Se un post appartenente alle categorie indicate appartiene anche ad altre categorie, verranno mostrate nella sezione di filtraggio anche quest'ultime.
- **[ShuffleGallery hide_post="id1,id2,..."]** - i post indicati verranno ESCLUSI dalla galleria.
- **[ShuffleGallery posts="id1,id2,..."]** - verranno visualizzati solo i post specificati.

Per agevolare l'utente nel recupero dei vari ID delle categorie e dei post il plugin aggiunge una colonna nel pannello di amministrazione dei post, categorie e pagine che mostra l'ID di ogni singolo elemento.
All'attivazione del plugin viene aggiunto un menu denominato **ShuffleGallery** che contiene una semplice pagina HTML che elenca le funzionalità del plugin.

3 Implementazione

La funzione *output_ gallery($atts)* è la funzione principale del plugin, la quale in base al tipo di shortcode utilizzato crea l'array *$args* composto dai parametri utilizzati nella WP querry per ricavare i post che comporranno la galleria.

```
public function output_gallery( $atts ) {

     extract(shortcode_atts(array(
           'category'                              => '',
           'hide_post'                            => array(),
           'posts'                                 => array(),
           'taxonomy'                              => 'category',
     ), $atts));

     $cat          = (!empty($category)) ? explode(',', $category)   : '';
     $taxonomy       = !empty($taxonomy) ? $taxonomy          : 'category';
     $exclude_post    = !empty($hide_post)? explode(',', $hide_post)  : array();
     $posts        = !empty($posts)  ? explode(',', $posts)    : array();

   // Get the number of total posts
     $nr_posts = wp_count_posts() -> publish;

     // WP Query Parameters
     $args = array (
           'post_type'         => 'post',
           'post_status'     => array( 'publish' ),
           'order'             => 'DESC',
           'orderby'            => 'ID',
           'posts_per_page'    => $nr_posts,
           'post__not_in'    => $exclude_post,
           'post__in'      => $posts,
           'meta_query'      => array(
                 array(
                       'key'      => '_thumbnail_id',
                       'compare' => 'EXISTS'
                 )
           )
     );

     // Category Parameter
     if( $cat != "") {

           $args['tax_query'] = array(
                 array(
                       'taxonomy'     => $taxonomy,
                       'field'      => 'term_id',
                       'terms'      => $cat,
                 ));
    }

     $wp_query = new WP_Query($args);

     // If no posts, return empty content
      if ( $wp_query->found_posts < 1 ) return '';

      // Store found posts into new variable
      $_posts = $wp_query->posts;

      // Define global categorories array
      $categories = array();

      // Define global posts array
      $posts = array();

```

```
// Loop through each found post
    foreach ( $_posts as $post ) {
      // Set up "aspect", "span" and "img_src" for current post
      $data = $this->calculate_data( $post );

      // If above returns false, skip current post
      if ( ! $data ) continue;

      // Store above data into the current post, using property "spg_data"
      $post->spg_data = $data;

      // Define array to store current post's category slugs
      $post->category_slugs = array();

      // Get current post's categories
      $_categories = wp_get_post_terms( $post->ID, 'category', array( 'fields' => 'all' ) );

      // Store current post's categories into global $categories array and append it to current post's "category_slugs" property
      foreach( $_categories as $category ) {
        if ( ! isset( $categories[$category->slug] ) ) $categories[$category->slug] = $category->name;
        array_push( $post->category_slugs, $category->slug );
      }

      // Append current post to global $posts array
      array_push( $posts, $post );
    }
```

Listing 1: output_gallery

In base allo shortcode utilizzato vengono costruiti gli array *$exclude_posts*, *$post* e *$cat*, composti dagli *ID* indicati nello shortcode. Se viene utilizzato lo shortcode di base **[ShuffleGallery]** gli array indicati sopra saranno vuoti.
Una volta ricavati i post che soddisfano i requisiti contenuti nell'array *$args*, si procede ad applicare a ogni singolo post il metodo *calculate_data($post)* che ha il compito di estrarre l'immagine di copertina, il sorgente dell'immagine, e di calcolare l'aspect ratio di come verrà visualizzata l'immagine nella galleria.

```
// Function for calculate a post's "aspect", "span" and "img_src"
  private function calculate_data( $post ) {
    $post_thumbnail_id = get_post_thumbnail_id( $post );
    if ( ! wp_attachment_is_image( $post_thumbnail_id ) ) return false;
    $file = get_attached_file( $post_thumbnail_id );
    $size  = @getimagesize( $file );
    if ( ! $size ) return false;
    $ratio = $size[0] / $size[1];
    $img_src = get_the_guid( $post_thumbnail_id );
    switch( true ) {
      case $ratio <= 0.95 :
        return array( 'aspect' => 'aspect--9x80', 'span' => 'row-span', 'img_src' => $img_src );
      case $ratio >= 1.67 :
        return array( 'aspect' => 'aspect--32x9', 'span' => 'col-span', 'img_src' => $img_src );
      default:
        return array( 'aspect' => 'aspect--16x9', 'span' => 'normal-span', 'img_src' => $img_src );
    }
  }
```

Listing 2: calculate_data

Nella figura seguente viene mostrato il codice che aggiunge la clonna con l'ID dei post, pagine e categorie.

```
//Posts ID column
add_filter('manage_posts_columns', 'posts_columns_id', 5);
add_action('manage_posts_custom_column', 'posts_custom_id_columns', 5, 2);
add_filter('manage_pages_columns', 'posts_columns_id', 5);
add_action('manage_pages_custom_column', 'posts_custom_id_columns', 5, 2);

function posts_columns_id($defaults){
    $defaults['wps_post_id'] = __('ID');
    return $defaults;
}

function posts_custom_id_columns($column_name, $id){
    if($column_name === 'wps_post_id'){
        echo $id;
    }
}


// Category ID Column on Category Page
add_filter( 'manage_edit-category_columns', 'category_column_header' );

function category_column_header($columns) {
    $columns['header_name'] = 'ID';
    return $columns;
}

add_filter( 'manage_category_custom_column', 'category_column_content', 10, 3 );

function category_column_content($content, $column_name, $term_id){
    return $term_id;
}
```

Listing 3: id - column

3.1 Fade-in transition

E' stato implementato un effetto di transizione per le immagini che entrano nel viewport. Per permettere ciò, al contenitore interno (*aspect_inner*) di ogni immagine è viene assegnata inizialmente la classe *hidden*:

```
<div class="aspect__inner hidden"><img src="<?php esc_attr_e( $post->spg_data['
    img_src'] ); ?>" obj.alt="obj.alt"/></div>
```

Listing 4: row 47 file gallery.php

La transizione viene attivata cambiando la classe dell'elemento che entra nella viewport da *hidden* a *fade_in_element*

```
Demo.prototype.addTransition = function() {
  var elements;
  var windowHeight;

  function init() {
    elements = document.querySelectorAll('.hidden');
    windowHeight = window.innerHeight;
  }

  function checkPosition() {
    for (var i = 0; i < elements.length; i++) {
      var element = elements[i];
      var positionFromTop = elements[i].getBoundingClientRect().top;
```

```
14
15        if (positionFromTop - windowHeight <= 0) {
16          element.classList.remove('hidden');
17          element.classList.add('fade-in-element');
18        }
19        else{
20          element.classList.remove('fade-in-element');
21          element.classList.add('hidden');
22        }
23      }
24    }
25    window.addEventListener('scroll', checkPosition);
26    window.addEventListener('resize', init);
27    init();
28    checkPosition();
29  };
```

Listing 5: addTransition function

Il codice CSS risulta:

```
1  @keyframes fade-in {
2    from {opacity: 0; transform: scale(.7,.7)}
3    to {opacity: 1;}
4  }
5
6  .spg-main-container .fade-in-element {
7    animation: fade-in 1.4s;
8  }
9
10 .spg-main-container .hidden {
11   opacity: 0;
12 }
```

Listing 6: CSS

Per permettere a Shuffle.js di avere il tempo di finire le transizioni delle immagini quando effettuiamo un filtraggio, è stata creata una multycallback sull'evento *click* su un pulsante:

```
1  /**
2   * Function needed to give time to shuffle.filter to make transitions of images
        before invoking the addTransition() method.
3   */
4  Demo.prototype.filterMultyCallback = function (evt) {
5    this._handleFilterClick(evt);
6    sleep(4).then(() => {
7      this.addTransition();
8    })
9  }
10
11 Demo.prototype.addFilterButtons = function () {
12   var options = document.querySelector('.filter-options');
13
14   if (!options) {
15     return;
16   }
17
18   var filterButtons = Array.from(options.children);
19
20   filterButtons.forEach(function (button) {
21     button.addEventListener('click', this.filterMultyCallback.bind(this), false);
22   }, this);
23 };
```

Listing 7: multycallback

Ciò è stato implementato anche per i pulsanti di ordinamento e per la sezione di ricerca.