

Relazione progetto Produzione e Progettazione Multimediale

Mihail Teodor Gurzu
5940299

22 maggio 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 query per ricavare i post che comporranno la galleria.

```
1 public function output_gallery( $atts ) {
2
3     extract(shortcode_atts(array(
4         'category'           => '',
5         'hide_post'          => array(),
6         'posts'              => array(),
7         'taxonomy'           => 'category',
8     ), $atts));
9
10    $cat      = (!empty($category)) ? explode(',', $category) : '';
11    $taxonomy  = !empty($taxonomy) ? $taxonomy : 'category';
12    $exclude_post = !empty($hide_post)? explode(',', $hide_post) : array();
13    $posts     = !empty($posts) ? explode(',', $posts) : array();
14
15    // Get the number of total posts
16    $nr_posts = wp_count_posts() -> publish;
17
18    // WP Query Parameters
19    $args = array (
20        'post_type'      => 'post',
21        'post_status'    => array( 'publish' ),
22        'order'          => 'DESC',
23        'orderby'        => 'ID',
24        'posts_per_page' => $nr_posts,
25        'post__not_in'   => $exclude_post,
26        'post__in'       => $posts,
27        'meta_query'     => array(
28            array(
29                'key'      => '_thumbnail_id',
30                'compare' => 'EXISTS'
31            )
32        )
33    );
34
35    // Category Parameter
36    if( $cat != "" ) {
37
38        $args['tax_query'] = array(
39            array(
40                'taxonomy' => $taxonomy,
41                'field'    => 'term_id',
42                'terms'    => $cat,
43            )
44        );
45    }
46
47    $wp_query = new WP_Query($args);
48
49    // If no posts, return empty content
50    if ( $wp_query->found_posts < 1 ) return '';
51
52    // Store found posts into new variable
53    $_posts = $wp_query->posts;
54
55    // Define global categories array
56    $categories = array();
57
58    // Define global posts array
59    $posts = array();
```

```

60 // Loop through each found post
61 foreach ( $_posts as $post ) {
62     // Set up "aspect", "span" and "img_src" for current post
63     $data = $this->calculate_data( $post );
64
65     // If above returns false, skip current post
66     if ( ! $data ) continue;
67
68     // Store above data into the current post, using property "spg_data"
69     $post->spg_data = $data;
70
71     // Define array to store current post's category slugs
72     $post->category_slugs = array();
73
74     // Get current post's categories
75     $_categories = wp_get_post_terms( $post->ID, 'category', array( 'fields'
=> 'all' ) );
76
77     // Store current post's categories into global $categories array and
append it to current post's "category_slugs" property
78     foreach( $_categories as $category ) {
79         if ( ! isset( $categories[$category->slug] ) ) $categories[$category->
slug] = $category->name;
80         array_push( $post->category_slugs, $category->slug );
81     }
82
83     // Append current post to global $posts array
84     array_push( $posts, $post );
85 }

```

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.

```

1 // Function for calculate a post's "aspect", "span" and "img_src"
2 private function calculate_data( $post ) {
3     $post_thumbnail_id = get_post_thumbnail_id( $post );
4     if ( ! wp_attachment_is_image( $post_thumbnail_id ) ) return false;
5     $file = get_attached_file( $post_thumbnail_id );
6     $size = @getimagesize( $file );
7     if ( ! $size ) return false;
8     $ratio = $size[0] / $size[1];
9     $img_src = get_the_guid( $post_thumbnail_id );
10    switch( true ) {
11        case $ratio <= 0.95 :
12            return array( 'aspect' => 'aspect--9x80', 'span' => 'row-span', '
img_src' => $img_src );
13        case $ratio >= 1.67 :
14            return array( 'aspect' => 'aspect--32x9', 'span' => 'col-span', '
img_src' => $img_src );
15        default:
16            return array( 'aspect' => 'aspect--16x9', 'span' => 'normal-span', '
img_src' => $img_src );
17    }
18 }

```

Listing 2: calculate_data

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

```

1 //Posts ID column
2 add_filter('manage_posts_columns', 'posts_columns_id', 5);
3 add_action('manage_posts_custom_column', 'posts_custom_id_columns', 5, 2);
4 add_filter('manage_pages_columns', 'posts_columns_id', 5);
5 add_action('manage_pages_custom_column', 'posts_custom_id_columns', 5, 2);
6
7 function posts_columns_id($defaults){
8     $defaults['wps_post_id'] = __('ID');
9     return $defaults;
10 }
11
12 function posts_custom_id_columns($column_name, $id){
13     if($column_name == 'wps_post_id'){
14         echo $id;
15     }
16 }
17
18
19 // Category ID Column on Category Page
20 add_filter( 'manage_edit-category_columns', 'category_column_header' );
21
22 function category_column_header($columns) {
23     $columns['header_name'] = 'ID';
24     return $columns;
25 }
26
27 add_filter( 'manage_category_custom_column', 'category_column_content', 10, 3 );
28
29 function category_column_content($content, $column_name, $term_id){
30     return $term_id;
31 }

```

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*:

```

1 <div class="aspect_inner hidden"></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*

```

1 Demo.prototype.addTransition = function() {
2     var elements;
3     var windowHeight;
4
5     function init() {
6         elements = document.querySelectorAll('.hidden');
7         windowHeight = window.innerHeight;
8     }
9
10    function checkPosition() {
11        for (var i = 0; i < elements.length; i++) {
12            var element = elements[i];
13            var positionFromTop = element.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
3  * before invoking the addTransition() method.
4  */
5 Demo.prototype.filterMultyCallback = function (evt) {
6     this._handleFilterClick(evt);
7     sleep(4).then(() => {
8         this.addTransition();
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.