

Capturer une carte géographique

DomPDF, JS, PHP, Wordpress

Web Compétences : 4

Dans la section parcours à vélo, une carte est affichée selon le fichier GPX qui est joint à la publication. Si la carte récupérée provient bien d'un serveur sur Internet, cela pose problème pour en imprimer le contenu. Cette publication explique comment afficher la carte à l'écran, puis de la capturer et enfin de la sauvegarder.

Publié dimanche 23 août 2020, 17h17
Modifié lundi 26 août 2024, 10h05



By Olivier Paudex

Introduction

Les publications de la section **“Parcours à vélo”** contiennent une carte géographique, affichée dynamiquement en fonction du fichier GPX joint à celles-ci. Comme la carte issue du GPX n’est pas un fichier image statique, il est tout simplement impossible à la librairie **“DomPDF”**, utilisée sur ce site pour créer les PDF, de l’utiliser. Une capture d’écran de la carte est alors nécessaire pour que celle-ci soit disponible comme une vraie image.

Affichage de la carte

Les publications de type **"Parcours à vélo"** contiennent toutes un fichier GPX joint. Ce fichier GPX affiche une carte dynamique reprenant le tracé contenu dans celui-ci. Pour réaliser ceci, un plugin a été utilisé, **"WP-GPX-Maps"**.



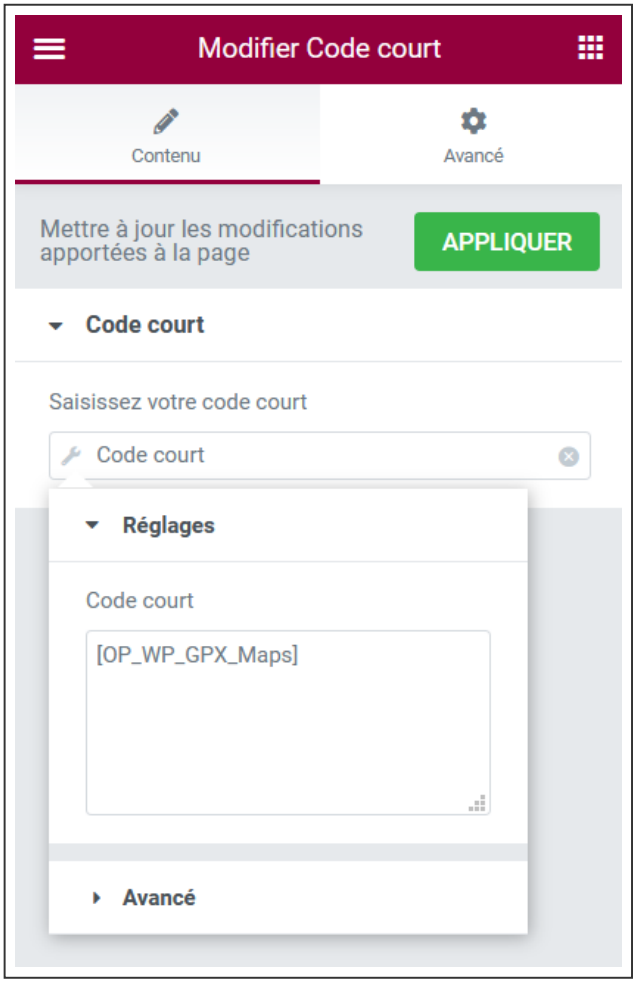
Le plugin WP-GPX-Maps

- Les fichiers GPX sont téléversés dans le dossier **"/wp-content/uploads/gpx/"**, lors de la création de la publication. Pour plus d'informations concernant cette partie, merci de lire la publication **"Liaison de pièces jointes"**.
- Le plugin **"WP-GPX-Maps"** affiche une carte à l'aide d'un code court contenant la syntaxe **'sgpx gpx="/wp-content/uploads/gpx/...."' (*)**.
- Les cartes sont affichées à l'aide du widget Elementor **"Code court"**. Le widget va appeler la fonction PHP **"OP_WP_GPX_Maps"**.

> * La syntaxe du code court se trouve entre [ ], mais impossible de l'écrire ici sans l'exécuter.



Le widget Elementor "Code court"



Les paramètres du widget

La fonction PHP ci-dessous retourne la syntaxe demandée par le plugin **"WP-GPX-Maps"**, depuis le champ fichier ACF.

```
// Get the GPX URL and tranform into a valid shortcode for the 'WP-GPX-Maps' plugin
function transform_GPX_URL_into_shortcode() {
  global $post;

  // Get URL from ACF 'track_gpx_upload' field
  $url = get_field('track_gpx_upload', $post->ID)['url'];
  // Keep only the folder name, without the domain
  $url = substr($url,strpos($url,'/wp-content'));
  // Add info to create the shortcode and return the url string
  // The syntax of the shortcode is between [ ], but impossible to write it down here without e
  return $url = 'sgpx_gpx="' . $url . '"';
}add_shortcode('OP_WP_GPX_Maps','transform_GPX_URL_into_shortcode');
```

La librairie “dom-to-image”

Pour parvenir à imprimer un PDF avec la carte géographique, une des solutions envisagées consiste à capturer l’image à l’écran. Ou plutôt à capturer son **“nœud DOM”**. Ici le nœud est notre widget Elementor **“Code court”**.

Une fois le nœud capturé, il faut encore le transformer en image avec une librairie appelé**“dom-to-image”**. Cette librairie a été créé par **Anatolii Saienko**, sur une idée originale de **Paul Bakaus**. Vous trouverez toute la documentation sur son **GitHub**.

Vous pouvez aussi télécharger la librairie en cliquant sur le lien **“pièce jointe”** en haut de cette publication.

Une fois la librairie entre vos mains, copiez-la dans un dossier **“js”** à la racine de votre thème enfant.

Créer un dossier pour chaque image

Pour plus de clarté dans la gestion des images capturées, elles seront enregistrées dans un dossier séparé.

- Toutes les images vont s’enregistrer dans le dossier **“wp-content/uploads/maps/”**, suivi d’un dossier portant l’ID de la publication et de son nom.
- Le dossier va se créer à l’ouverture de la publication, si celui-ci n’existe pas déjà.

Voici le code qui utilise le crochet **“template_redirect”**, pour l’exécuter à l’ouverture de la publication :

```
// Create map folder
function tracks_map_folder() {

  // Global post
  global $post;
  // Only on "tracks" single page
  if (is_singular('tracks')) {

    // Create folder 'maps' with post slug name into uploads folder
    $foldername = wp_upload_dir();
    $postname = $post->ID . '_' . $post->post_name;
    $foldername = $foldername['basedir'] .  '/' . 'maps' . '/' . $postname;

    // Test if folder already exists
    if (!is_dir($foldername)) {
      mkdir ($foldername, 0755, true);
    }
  }
}add_action('template_redirect', 'tracks_map_folder');
```

Suppresion d'une image

Si la publication venait à être supprimée, le fichier image serait lui aussi détruit grâve aux deux fonctions ci-dessous :

```
// Delete map image folder, when a track post is deleted
function delete_map_image($postID, $post) {
  // Get post type
  if ($post->post_type == 'tracks') {

    // Get map image file path
    $foldername = wp_upload_dir();
    $postname = $post->ID . '_' . $post->post_name;
    $foldername = $foldername['basedir'] .  '/' . 'maps' . '/' . $postname;
    // Remove '__trashed' from foldername
    $foldername = str_replace('__trashed', '', $foldername);
    // if exists, delete file
    if (is_dir($foldername)) {
      removeDirectory ($foldername);
    }
  }
}
add_action ('delete_post', 'delete_map_image',10,2);
/*****************************************************************************************************
function removeDirectory($path) {
  $files = glob($path . '/*');
  foreach ($files as $file) {
    is_dir($file) ? removeDirectory($file) : unlink($file);
  }
  rmdir($path);

  return;}
```

Capturer une image

Pour réaliser la capture du **"noeud DOM"** (la carte géographique), une fonction en Javascript s'exécutera lors de l'ouverture de la publication.

Comme dans tous les appels de code écrit en Javascript, le fichier de cette fonction sera mis en queue avec **"wp_enqueue_script"**, tout en limitant strictement son accès aux publications de type **"Parcours à vélo"**.

De plus, cette fonction va devoir enregistrer une image de la carte géographique. Pour réaliser ceci, l'emplacement et le nom de fichier devront être passés en paramètres avec **"wp_localize_script"**.

Voici le code et quelques explications :

- La limitation strictement réservée aux publications **"Parcours à vélo"** se fait grâce à la fonction **"is_singular('tracks')"**.

- Le dossier de destination est **"/wp_content/uploads/maps/"**.
- Chaque image est enregistrée dans un dossier portant son ID, suivi de son nom.
- Chaque image est enregistrée selon son ID, suivi de son nom, puis de l'extension**".jpg"**.
- Trois paramètres sont passés à la fonction Javascript, dont le chemin complet**($filename)** et l'url complet **($fileurl)**.

Copiez le code PHP ci-dessous et sauvegardez-le dans un fichier **"Enqueue_JS_Scripts.php"** ou un nom à votre convenance. Puis placez-le dans le dossier **"php"**, à la racine du thème enfant.

```
// Map To Image
if (!function_exists('MapToImage')) {
  function MapToImage() {
    // Only on single custom post "tracks" type
    if (!is_admin() && (is_singular('tracks'))) {
      $jquery = array ('jquery');
      $version = '1.0';
      $in_footer = true;

      // Global post
      global $post;

      // Get image filename & url
      $foldername = wp_upload_dir();
      $postname = $post->ID . '_' . $post->post_name;
      $fileurl = $foldername['baseurl'] .  '/' . 'maps' . '/' . $postname . '/' . $postname .
      $filename = $foldername['basedir'] .  '/' . 'maps' . '/' . $postname . '/' . $postname .

      // Parameters to javascript function
      $parameters = array(
        'filename' =>  $filename,
        'fileurl' => $fileurl,
        'stylesheet_directory_uri' => get_stylesheet_directory_uri()
      );

      // Enqueue script
      wp_enqueue_script('MapToImage', get_stylesheet_directory_uri() . '/js/MapToImage.js', $j
      wp_localize_script('MapToImage', 'parameters', $parameters);
    }
  }
}add_action('wp_enqueue_scripts', 'MapToImage');
```

La fonction "captureImage"

Il faut maintenant créer le fichier **"MapToImage.js"** dans le dossier **"js"** à la racine du thème enfant. Celui-ci va contenir la fonction **"captureImage"**.

L'appel se faisant à l'ouverture d'une publication de type **"Parcours à vélo"**, la fonction va devoir attendre que la carte soit chargée et affichée sur l'écran avant de pouvoir la capturer. Pour ceci, la fonction de capture **"captureImage"** s'exécutera après un délai de **500ms**.

Ci dessous le code, dont voici quelques explications :

- Une requête de type HTTP va vérifier que l'image de la carte n'existe pas encore. Le status **"HTTP 200"** retourne au navigateur que le fichier existe. Il suffit d'inverser ce status.
- La fonction va ensuite créer une variable **"phpfile"** qui va contenir le nom d'un fichier à exécuter, **"Save_Image.php"**.
- La fonction va ensuite récupérer le **"noeud DOM"** dont tous les ID contiennent **"wpgpxmaps_"** et placer le résultat dans une variable **"node"**.
- Ensuite, l'appel de la fonction **"domtoimage.toJpeg"** va nous donner les données de l'image JPEG au format **"base64-encoded"**, à partir de la variable **"node"**.
- La dernière étape consiste à ouvrir une requête Ajax en lui passant en paramètre '**POST"** le nom du fichier PHP à exécuter, les données de l'image, ainsi que son nom.

```
// Delay to give the map time to load
var delay = 500;
var timeout = 0;
// On window load, call captureImage with a delay
jQuery(window).on("load", function() {
  timeout = setTimeout(function() {
    captureImage (parameters.filename, parameters.fileurl, parameters.stylesheet_directory_uri
  }, delay);
});
/*****************************************************************************************
function captureImage(filename, fileurl, stylesheet_directory_uri) {
  // Check if map exist
  var xhr = new XMLHttpRequest();
  xhr.open('HEAD', fileurl, false);
  xhr.send();
  // Create map only if file doesn't exist
  if (xhr.status != 200) {

    // PHP File to call with the POST method
    phpfile = stylesheet_directory_uri + "/php/Save_Image.php";
    // Get the map
    node = document.querySelector('[id*=wpgpxmaps_]');
    // Save the map as JPEG
    domtoimage.toJpeg(node, {quality: 0.95}).then(function (dataURL) {
      // Call php file and attach the dataURL with the POST method
      var ajax = new XMLHttpRequest();
      ajax.open("POST", phpfile, true);
      ajax.setRequestHeader("Content-type","application/x-www-form-urlencoded");
      ajax.send("image=" + dataURL + "&filename=" + filename);
    });
  }}
```

La sauvegarde de l'image au format JPEG

Pourquoi ne pas enregistrer l'image directement avec la fonction **"domtoimage.toJpeg()"** ? Et bien, c'est impossible ! Ce code JS s'exécute en local et la sauvegarde se fait sur le serveur, d'où l'appel à la requête Ajax.

- Pour sauvegarder l'image au format JPEG, il faut créer le fichier '**Save_Image.php"**, dans le dossier **"php"**, à la racine du thème enfant.
- Ce fichier, appelé par la requête Ajax ci-dessus, va récupérer les données de l'image, la décoder, et enfin, la sauvegarder.

```
<?php
  /**
  * Plugin Name: Save_Image
  * Description: Save an image from the POST parameter
  * Author: Olivier Paudex
  * Author Web Site: https://www.fuyens.ch
  */
  // Global post
  global $post;
  // Transform DataURL to a decoded image
  $image = $_POST["image"];
  $image = explode(";", $image)[1];
  $image = explode(",", $image)[1];
  $image = str_replace(" ","+", $image);
  $image = base64_decode($image);
  // Get the image filename
  $filename = $_POST["filename"];
  // Save image
  file_put_contents($filename, $image);?>
```

En résumé

Si vous avez perdu le fil de l'explication, voici ce que vous devriez avoir :

- Un dossier **"wp-content/uploads/maps/ID+nom"**, créé automatiquement à l'ouverture d'une publication.
- Un fichier JS **"dom-to-image.js"**, contenant la librairie du même nom, qui doit se trouver dans le dossier **"js"**.
- Un fichier PHP **"Enqueue_JS_Scripts.php"**, qui doit se trouver dans le dossier **"php"**. Celui-ci contient une fonction **"MapToImage()"** qui va charger en queue le fichier JS **"MapToImage.js"**.
- Un fichier JS **"MapToImage.js"**, qui doit se trouver dans le dossier **"js"**. Celui-ci exécute, après un délai de **500ms**, la fonction **"captureImage"**. La fonction va ouvrir une requête Ajax, qui va ouvrir le fichier PHP **"Save_Image.php"**.
- Un fichier PHP **"Save_Image.php"**, qui doit se trouver dans le dossier **"php"**. Celui-ci va décoder les données de l'image et la sauvegarder au format JPEG.

Et voici le déroulement :

- A l'ouverture d'une publication de type **"Parcours à vélo"**, celle-ci va ouvrir le fichier GPX joint pour l'afficher dans un widget Elementor **"Code court"**. La carte est produite dynamiquement par le plugin **"WP-GPX-Maps"**.
- La carte affichée est alors capturer, puis sauvegarder au format JPEG dans un dossier **"/wp-content/uploads/maps/ID+Nom/"**.
- Le nom de l'image, qui porte le même nom que le dossier, est ajouté.
- A l'ouverture du fichier PDF, la librairie DomPDF utilise l'image pour créer le PDF.

Le mot de la fin

Cette manière de procéder n'est certainement pas la seule. Mais peu importe la voie choisie, la capture de l'image est la seule solution valable. La capture de la carte géopgraphique se faisant sur le navigateur, donc en local, posait le problème de la sauvegarde de l'image sur le serveur, résolu par la requête Ajax.

Amusez-vous bien.