

Exécuter du Javascript dans Wordpress

JS, Wordpress

Web Compétences : 3

Comment réaliser une fonction qui n'a pas été prévu par Wordpress et/ou Elementor. Tout dépend si cette tâche doit être réalisée du côté du serveur ou directement sur le navigateur du client. Une des grandes missions de celui-ci, est d'afficher les informations reçues par le serveur. Mais comment formater ces données ? C'est là que Javascript rentre dans l'arène. Voici un petit tour d'horizon de l'implémentation de Javascript avec Wordpress.

Publié samedi 2 novembre 2019, 09h19
Modifié lundi 26 août 2024, 10h06



By Olivier Paudex

Introduction

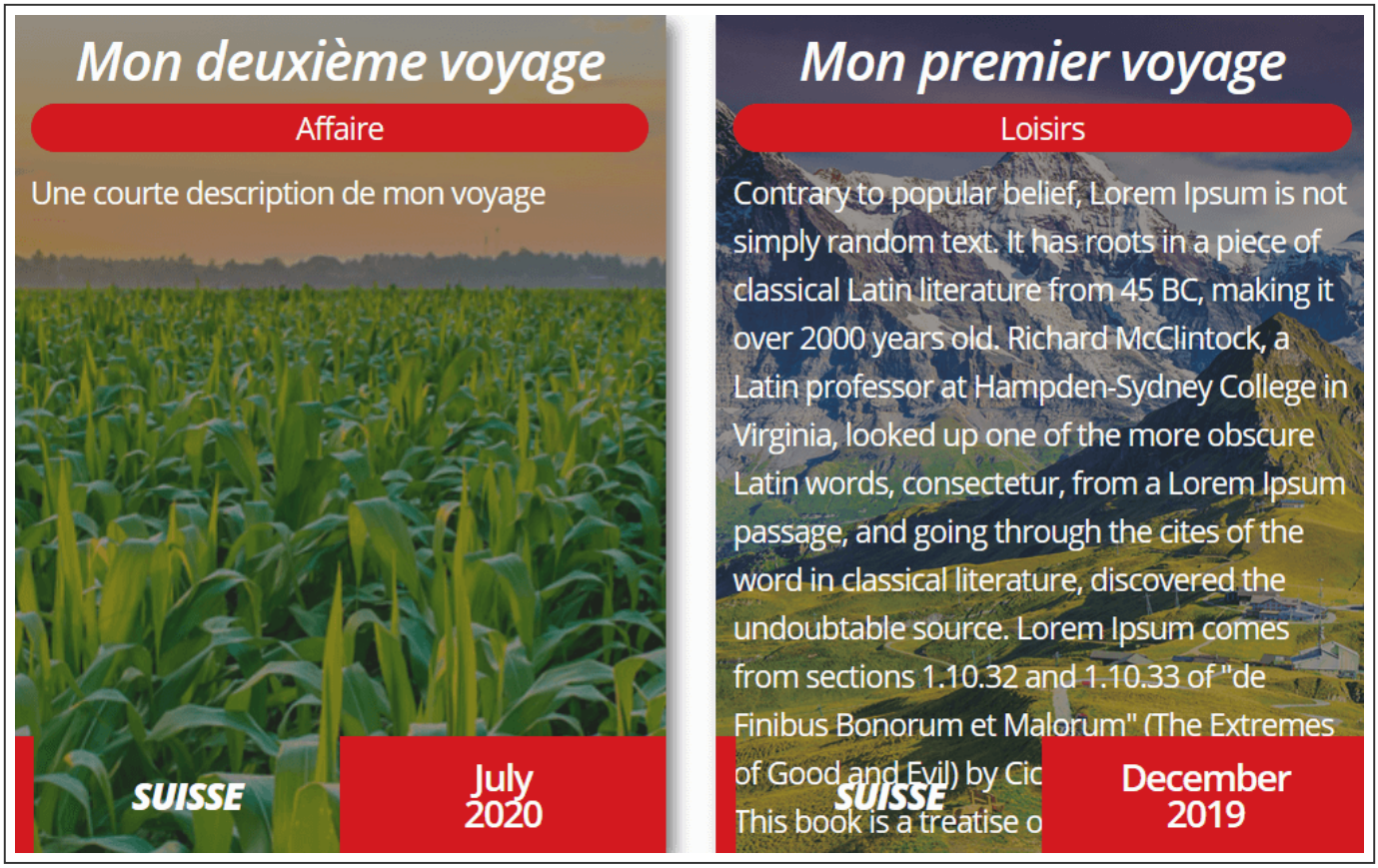
Un site web, même si vous n'êtes pas développeur, dans lequel vous utilisez Elementor pour créer vos pages, va très certainement utiliser le langage Javascript pour exécuter des tâches. Le Javascript est une référence dans le monde du web. **95%** des sites web l'utilise et c'est le langage le plus utilisé dans le monde du web, à plus de **67%**. Il a été inventé en **1995**, par la société **Netscape**, alors pionnière du web. C'est un langage de scripts, à savoir qu'il s'exécute à la volée. Il est orienté pour s'exécuter sur la machine cliente, au contraire de PHP, qui est lui un langage qui s'exécute sur le serveur. PHP et Javascript sont les deux langages omniprésent dans WordPress et sont indissociables. Il existe depuis peu, une nouvelle syntaxe, adapté et orienté web, qui porte le nom de **Jquery**, mais ce dernier fonctonne de manière identique au Javascript (JS). On peut même mélanger les deux syntaxes.

Implémenter Javascript dans son thème enfant

Comme pour le PHP, le Javascript va s'arrimer dans WordPress par le biais du thème enfant. En principe, les deux langages fonctionnent de la même manière, ils se mettent en queue et s'exécutent à la demande.

Ceci n'est pas un cours de Javascript. Je ne suis pas un développeur expérimenté et le code pourrait être très certainement amélioré. Le code ci-dessous est un exemple réel utilisé sur ce site. Le design de la page d'archives, affiche un bloc représentant une publication. Celui-ci présente une introduction sous forme de texte, une sorte d'accroche qui n'est pas limitée dans le nombre de caractères. Au début de la conception, le but à atteindre était bien de le contenir dans une zone de texte délimitée par les marges, mais cette implémentation est vite devenu inadéquate car le texte de la publication se devait d'être dynamique, pour laisser la place au titre, si celui-ci devait s'étendre sur plus d'une ligne. Pour effectuer ce tour de force, la tâche principale de cette fonction devait compter le nombre de lignes maximum pouvant s'afficher dans le bloc d'archive pour que le texte reste circonscrit à l'intérieur et le tronquer, si nécessaire.

Ci-dessous, le problème est clairement démontré. Le texte de l'accroche dépassant l'espace prévu.



Le texte de la description dépasse la zone dédiée

A la manière du chapitre PHP, commencez par déclarer un fichier PHP dans lequel vous allez référencer vos fonctions Javascript.

- Dans le fichier "**functions.php"**, ajoutez un fichier nommé ici "**Enqueue_JS_Scripts.php**".

```
include_once ($dir . '/php/Enqueue_JS_Scripts.php');
```

- Créez un dossier **"js"**, à la racine de votre thème enfant (là où se trouve également votre dossier **"php"** et **"css"**).
- Créez le fichier "**Enqueue_JS_Scripts.php**" dans votre dossier "**php**".
- Creez le fichier JS nommé ici "**CountLine.js**" dans votre dossier "**js**".
- Copiez-Collez le code ci-dessous dans le fichier "**Enqueue_JS_Scripts.php**".

> Attention, le fichier "**CountLine.js**" va s'exécuter à chaque fois qu'une page est chargée. Pour limiter ceci, on peut utiliser les fonctions de test fournies par WordPress (**is_admin, is_archive, is_search, is_page, etc...**). Les fonctions parlent d'elles-mêmes. Le code ci-dessous va s'exécuter uniquement si le visiteur se trouve sur une page du site (pas sur la console d'admin) et que celle-ci est une page d'archive, une page de recherche ou la page d'accueil.

```
// Count Lines in archives widget posts
if (!function_exists('CountLine')) {
  function CountLine() {
    // Only on archives, search and home pages
    if (!is_admin() && (is_archive() || is_search() || is_page('home'))) {
      $jquery = array ('jquery');
      $version = '1.0';
      $in_footer = true;

      // Enqueue script
      wp_enqueue_script('CountLine', get_stylesheet_directory_uri() . '/js/CountLine.js', $jque
    }
  }
}add_action('wp_enqueue_scripts', 'CountLine');
```

- Copiez-Collez le code JS dans le fichier "**CountLine.js**".

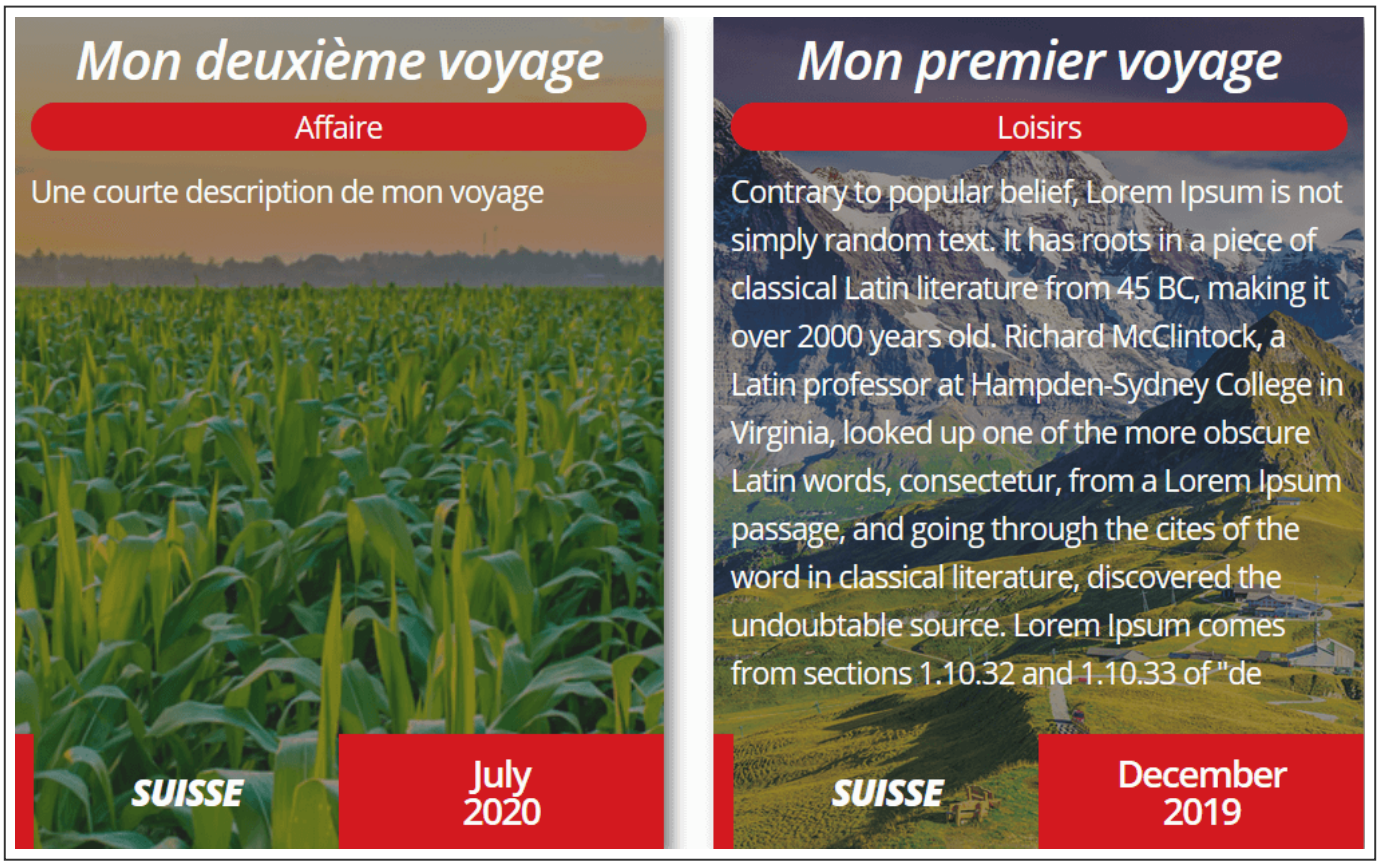
Le but n'est pas d'expliquer en détail ce que fait le code, mais sa fonction est de calculer le nombre de lignes de texte qu'un bloc d'archive peut écrire selon la place disponible.

```
/**
* Plugin Name: CountLine
* Description: Count number of text lines and chop textbox
* Author: Olivier Paudex
* Author Web Site: https://www.fuyens.ch
*/
// Global var (to avoid resize on scrolling)
var wwidth = window.innerWidth;
var timeout = 0;
var delay = 250;
// On document load
jQuery(document).ready(cut_text);
// On window resize
jQuery(window).resize(function() {
  clearTimeout(timeout);
  timeout = setTimeout(cut_text, delay);
});
/*****************************************************************************************
// Cutting text
function cut_text() {
  // Init
  var container_height = 500;

  // Avoid resize on scrolling
  if (timeout == 0 || window.innerWidth != wwidth) {
    wwidth = window.innerWidth;
    // Loop through archives posts
    let archives = document.getElementsByClassName('archives');
    for (var index = 0; index < archives.length; index++) {
      // Reset on each archive
      var archive_title_height = 0;
      var archive_termes_height = 0;
      var archive_infos_height = 0;
      // Get height of title
      if (archives[index].getElementsByClassName('title').length > 0) {
        archive_title_height = archives[index].getElementsByClassName('title')[0].offsetHeight
      }
      // Get height of termes
      if (archives[index].getElementsByClassName('termes').length > 0) {
        archive_termes_height = archives[index].getElementsByClassName('termes')[0].offsetHeigh
      }
      // Get height of infos
      if (archives[index].getElementsByClassName('infos').length > 0) {
        archive_infos_height = archives[index].getElementsByClassName('infos')[0].offsetHeight
      }
      // Calculate height left over for textbox
      var textbox_height = container_height - archive_title_height - archive_termes_height - ar
      // Calculate the float value of "p" line-height -> line-height (em) * font-size (px)
      if (archives[index].getElementsByTagName('p').length > 0) {
        let textbox = archives[index].getElementsByTagName('p')[0];
        var line_height = window.getComputedStyle(textbox).getPropertyValue('line-height');

        // Check if Apple devices or others
        if (navigator.userAgent.match(/(iPod|iPhone|iPad)/)) {
          line_height = parseInt(line_height);
        } else {
          line_height = parseFloat(line_height);
        }
        // Calculate the int number of lines for the textbox
        lines = Math.floor(textbox_height/line_height);
        // Adjust the textbox height, cutting off the lines which are incomplete
        textbox_height = Math.floor(lines*line_height);
        textbox.style.setProperty('height',textbox_height + 'px');
      }
    }
  }
}
```

Et voici le résultat.



Le texte de la description reste dans la limite disponible grâce à Javascript

Voilà une page d'archives qui commence à ressembler à quelque chose. Mais une page d'archives sans moteur de recherche n'est pas vraiment une page d'archives, non ? Le prochain chapitre vous conduira vers la réalisation d'**un filtre recherche** créé avec les outils d'Elementor uniquement.