Running Javascript in Wordpress

JS, Wordpress

Web Skills : 3

How to perform a function that has not been expected by Wordpress and/or Elementor. Everything depends on whether this task must be done on the server side or directly on the client's browser. One of the main missions of the client navigator is to display the information received by the server. But how to format this data? That's where Javascript comes in. Here is a small overview of the implementation of Javascript with Wordpress.

Published Saturday November 2nd 2019, 09:19
Modified Monday August 26th 2024, 10:06

By Olivier Paudex

Introduction

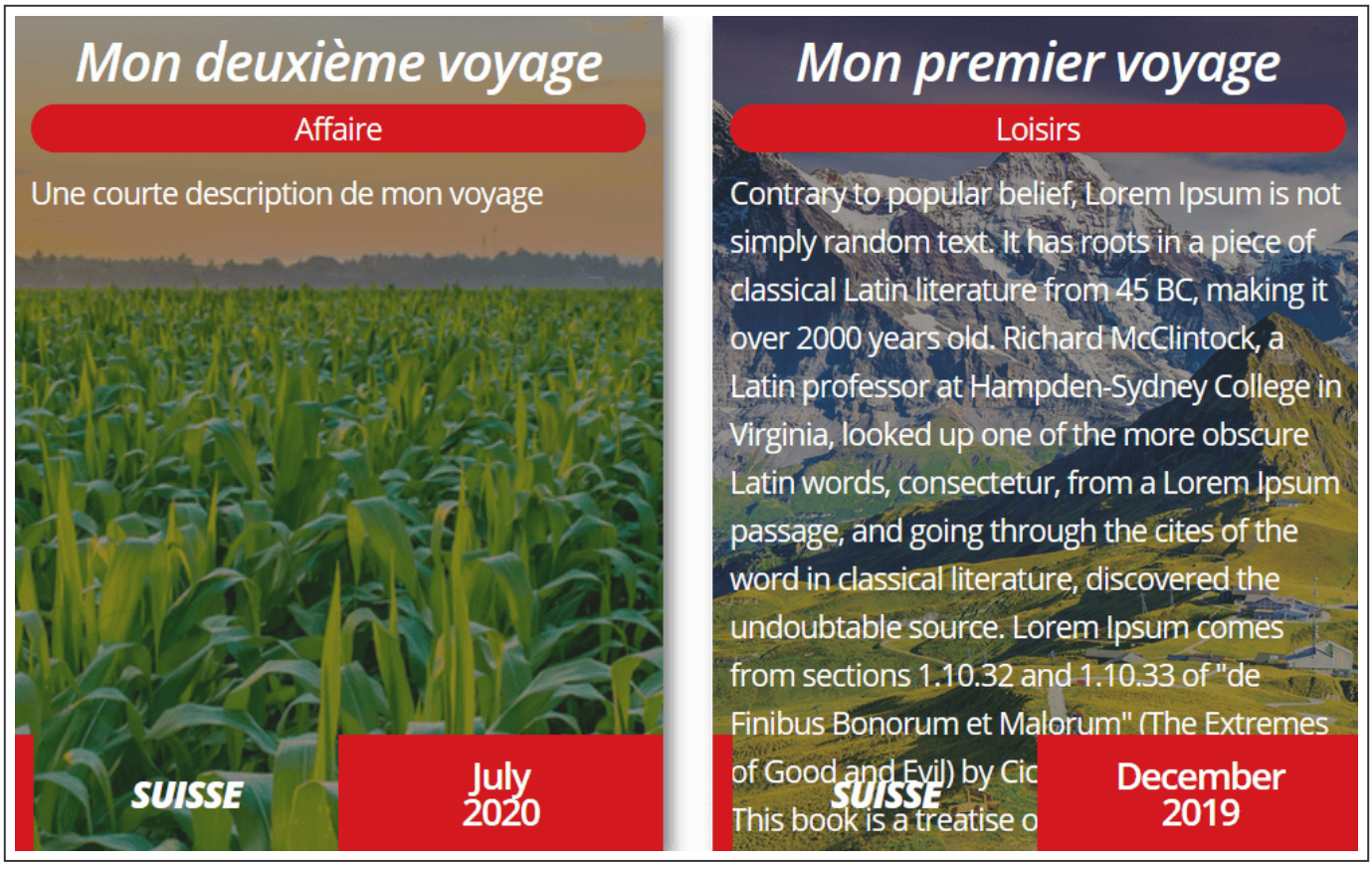
A website, even if you are not a developer, in which you use Elementor to create your pages, will most likely use Javascript to perform tasks. Javascript is a reference in the web world. **95%** of websites use it and it is the most used language in the web world, at more than **67%**. It was invented in **1995**, by the company **Netscape**, then pioneer of the web. It is a scripting language, which means that it is executed on the fly. It is oriented to run on the client machine, unlike PHP, which is a language that runs on the server. PHP and Javascript are the two omnipresent languages in WordPress and are inseparable. There is recently, a new syntax, adapted and web-oriented, which is called **Jquery**, but it works identically to Javascript (JS). We can even mix the two syntaxes.

Implement Javascript in your child theme

As for PHP, Javascript will dock in WordPress through the child theme. In principle, the two languages work in the same way, they queue up and run on demand.

This is not a Javascript course. I am not an experienced developer and the code could certainly be improved. The code below is a real example used on this site. The design of the archive page, displays a block representing a post. This one presents an introduction in the form of text, a kind of teaser that is not limited in the number of characters. At the beginning of the design, the goal was to contain it in a text zone delimited by the margins, but this implementation quickly became inadequate because the text of the post had to be dynamic, to leave space for the title, if it had to extend over more than one line. To accomplish this, the main task of this function was to count the maximum number of lines that could be displayed in the archive block to keep the text contained within and truncate it, if necessary.

Below, the problem is clearly demonstrated. The excerpt exceeding the space provided.



The text of the description exceeds the dedicated area

Like in the PHP chapter, start by declaring a PHP file in which you will reference your Javascript functions.

- In the "**functions.php"** file, add a file named here **"Enqueue_JS_Scripts.php"**.

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

- Create a **"js"** folder, at the root of your child theme (where your **"php"** and **"css"** folders are also located).
- Create the file **"Enqueue_JS_Scripts.php"** in your **"php"** folder.
- Create the JS file named here **"CountLine.js"** in your **"js"** folder.
- Copy and paste the code below into the **"Enqueue_JS_Scripts.php"** file.

> Be careful, the **"CountLine.js"** file will run each time a page is loaded. To limit this, you can use the test functions provided by WordPress (**is_admin, is_archive, is_search, is_page, etc...**). The functions speak for themselves. The code below will only run if the visitor is on a page of the site (not on the admin console) and this one is an archive page, a search page or the home page.

```
// 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');
```

- Copy and paste the JS code into the **"CountLine.js"** file

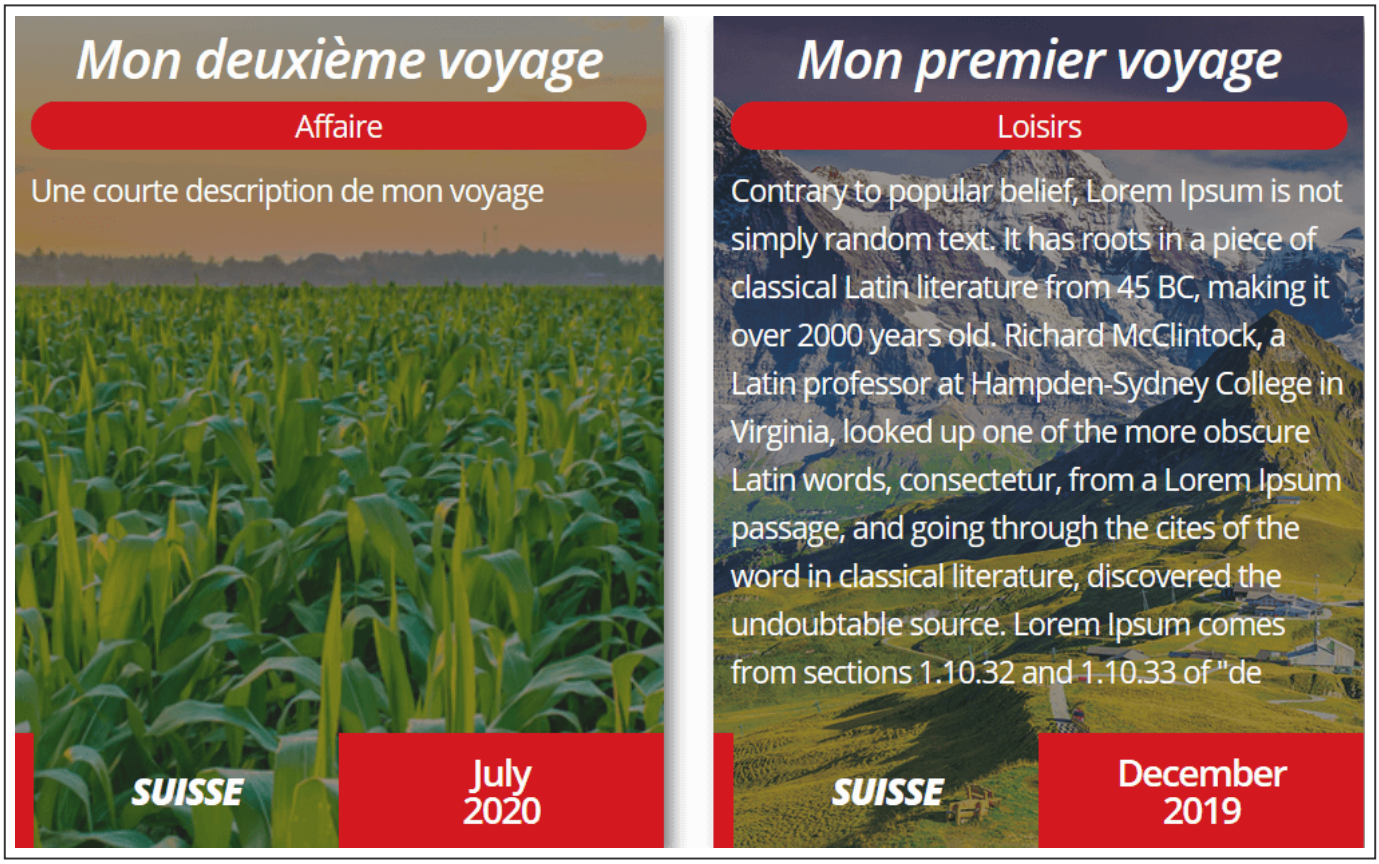
The purpose is not to explain in detail what the code does, but its function is to calculate the number of lines of text that an archive block can write according to the available space.

```
/**
* 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');
      }
    }
  }
}
```

And here is the result.



The text of the description remains in the limit available thanks to Javascript

Now this archive page is starting to look like something. But an archive page without a search engine isn't really an archive page, is it ? The next chapter will lead you to the realization of**a search filter** created with Elementor tools only.