May 10, 2017

Algorithm To Enrich Display Of A Link In A Web Application

Erwan GEREEC
ALE International

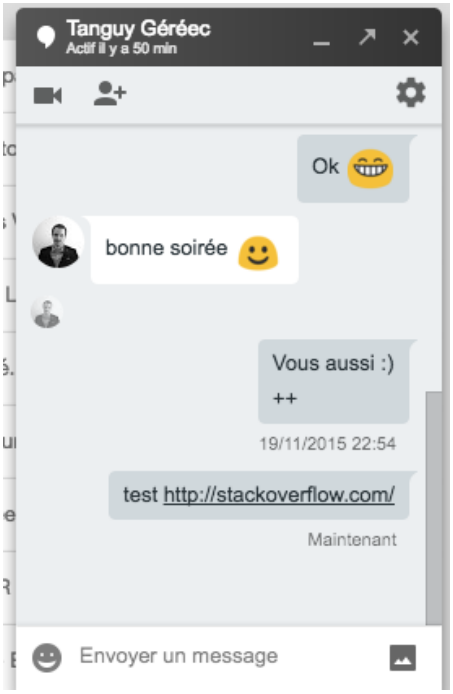
Docket Number	**FR82017001**
Title	**“Algorithm to enrich display of a link in a Web application”**
Contributor	**Erwan GEREEC**
Company	**ALE International**

Description of the technical solution:

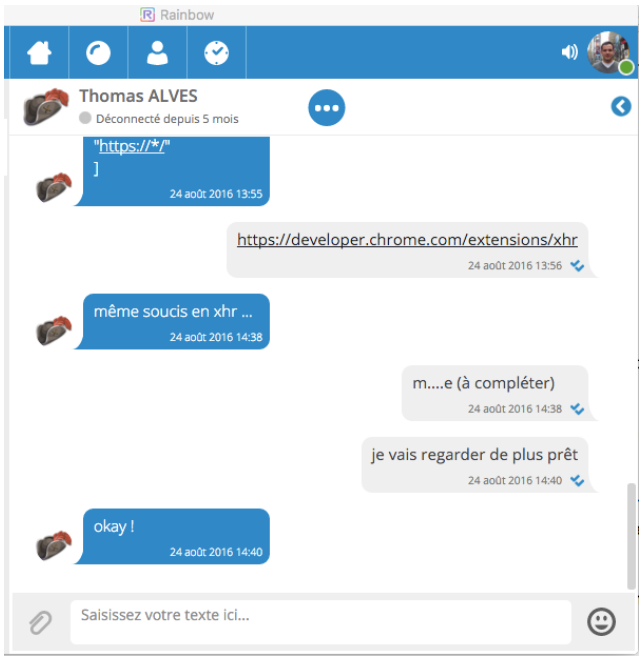
Many web applications provide a way to discuss with others members throw chat sessions. In Google Hangouts or Alcatel-Lucent Enterprise Rainbow clients for example, it’s possible for the user to send some link to others users.

In these web applications, these link are written using an HTML LINK tag (a href=”#”) as defined in RFC 1866 (Hypertext Markup Language - 2.0) (https://tools.ietf.org/html/rfc1866).

The drawback of this HTML tag, is that it’s only display the link written by the user. It’s a simple text which doesn’t contain any additional data like the website title, the website description or any website medias like pictures or videos. This is a disadvantage because if the user who receives the link does not know the website to which he refers, he will have no information to explain the contents of this site.



Google Hangout



Alcatel-Lucent Enterprise Rainbow

As you can see in the two screenshots, a user has sent a link. The link is displayed in the web application as a text. The user who receive the link can click on this link to open the dedicated website in a new window. There is no additional data about this website in the chat window.

This document describes a universal algorithm to enhance display of a link tag in web applications. The purpose is to display at least the title of the website, a little description and if exist, some preview of images which are displayed on the website. Thus, the user who receive a link will have relevant information to know what is the content of the website.

Before starting this study, no existing algorithms seem allowing to enrich the display of any HTML link tag. The reason is because all browsers doesn't allow to perform an outgoing HTTP request from a website A to a website B. (Cross domain restrictions).

The only thing that exists today and that we often see on many website, is to add a button "See more" to open the link in a new window in the browser. But as you can see, it doesn't enrich the display of the link tag directly into the web application.

This solution is different because it provides a single JavaScript library that could be used in any Web application. It also avoids cross domain restrictions.

Consequently, the solution allows to have a complete automation chain to build test cases:

Advantages:

- Easy-to-use: Easy to add in all Web application because it's a JavaScript library.
- Mobile compatible: A mobile application developed using Web components (no native code) can also use this algorithm.
- Universal/Generic: As it's a JavaScript library, algorithm is compatible with all browsers and all Webkit engines.
- No user action: Provides automatic way to convert a single text link to an enriched content.
- Time Saving: No configuration needed.

Disadvantages:

- This algorithm can be used only in web applications, not in native code (Java, C, etc.).

This solution is an algorithm divided in three different parts providing an easy way to convert a single HTML link tag to an enriched HTML content displaying relevant information for a given website. (see representation below): **PARSING PROCESS**, **EXTRACTING PROCESS** and **FORMAT PROCESS.**

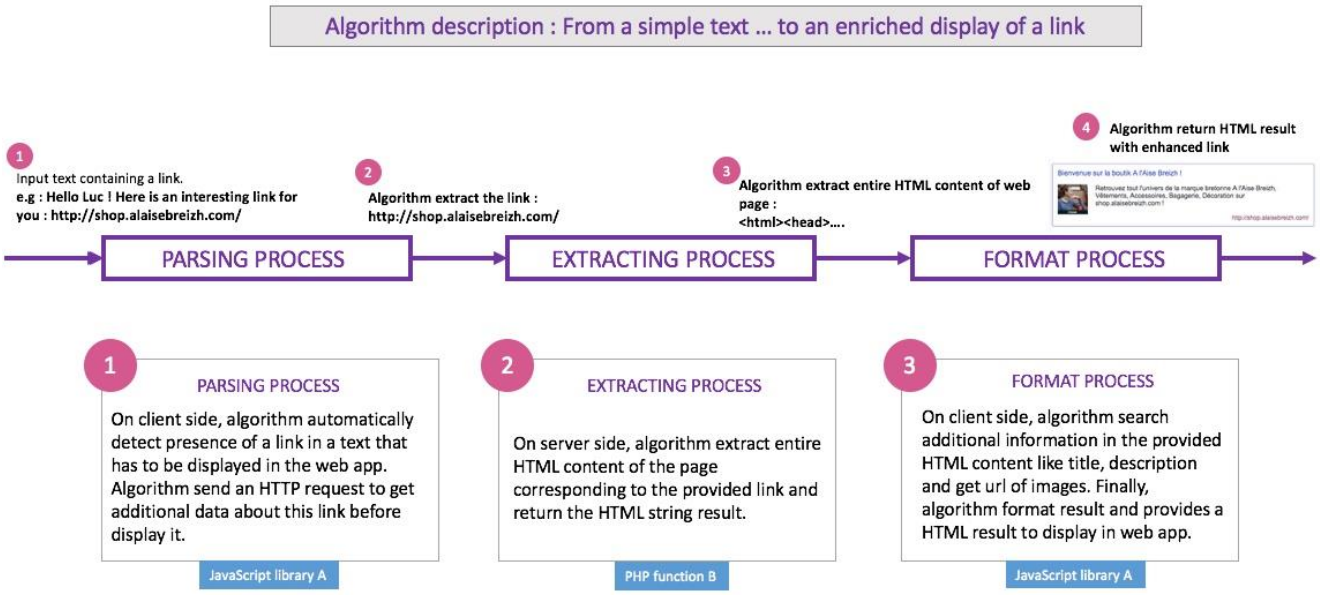


Figure 1: Global description of algorithm

Here is a technical description of each steps:

1. **PARSING PROCESS**: This step is implemented using a JavaScript code able to detect the presence of a link into a string. Once a link is detected, algorithm perform HTTP request to the PHP code on server side.

2. **EXTRACTING PROCESS:** The goal of this step is to extract all HTML content of a webpage using the provided link of the page. This step uses a PHP code that perform a Curl request to extract entire HTML content of the page.

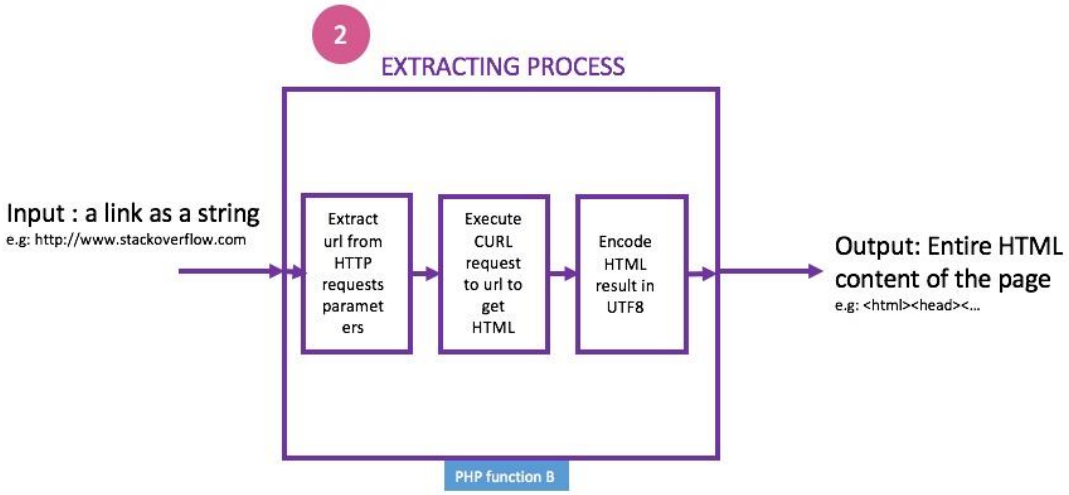


Figure 2: EXTRACTING process of algorithm

3. **FORMAT PROCESS:** The extract all relevant HTML content and create a new HTML div element containing pretty link information.

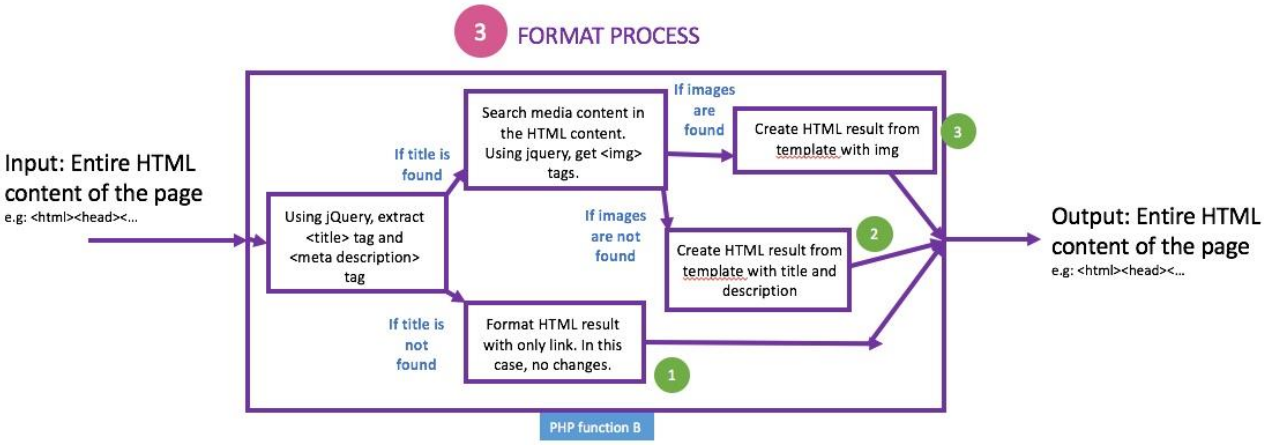


Figure 3: FORMAT process of algorithm

Here is an example of a result in a web test application for each case:

- Item 1 has not been changed because algorithm doesn't find any relevant information in HTML content.
- Item 2 display a first level of enriched content with the title and a little description of the webpage.
- Item 3 display the top level content with title, description and medias.

Figure 4: Output result in a web test application.

Technically, the algorithm is implemented in a JavaScript library on client side and a PHP function on a server side which can be added in any web page. Implementation of this architecture allows to avoid cross domain restrictions. Here is the entire code of the JavaScript library and the PHP function.

PHP function:

```php
<?php

    header('Access-Control-Allow-Origin: *');
    header('Access-Control-Allow-Methods: POST');
    header('Access-Control-Max-Age: 1000');
    // Init result json containing HTML content of desired web page.
    $outjson = array("htmlContent" => null);
    // Get url
    $url = $_POST["urlParam"];
    error_log("TEST = ".$url );
    $c = curl_init($url);
    curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
    $html = curl_exec($c);
    if (curl_error($c))
        die(curl_error($c));
    // Get the status code
    $status = curl_getinfo($c, CURLINFO_HTTP_CODE);
    curl_close($c);
    // Set html content result
    $outjson["htmlContent"] = mb_convert_encoding($html,'UTF-8','UTF-8');
    error_log(mb_convert_encoding($html,'UTF-8','UTF-8'));
    // Return data
    echo json_encode($outjson);
?>
```

JavaScript library:

```javascript
(function(window){

   // This function will contain all our code
   function myLibrary(){

      var _myLibraryObject = {};

      // Just create a property to our library object.
      _myLibraryObject.parseLink = function(url, targetId){

         console.log("Linko called !");

         var jsonDataObj = {
            urlParam : url
         };

         console.log('Reading html content from url = '+jsonDataObj.urlParam );

         jQuery.ajax({
            cache: false,
            type: "POST",
            crossDomain: true,
            dataType: "json",
            url: "http://path_to_php_function.php",
            data: jsonDataObj,
            success: function (jsonResponse) {
               console.log('Request is success ! | jsonResponse = ' +JSON.stringify(jsonResponse));
               _myLibraryObject.beautifyLink(url, jsonResponse.htmlContent, targetId);
```

```
            },
            error: function (jqXHR, textStatus, errorThrown) {
                console.log('Request failed ! | jsonError = ' +JSON.stringify(jqXHR)
);
                callbackError(jqXHR);
            }
        });

    };

    _myLibraryObject.beautifyLink = function (url, htmlStr, targetId) {

        var fakeDom     = $('<html></html>').append(htmlStr);
        var urlTitle    = $(fakeDom).find('title').text();
        var description = $(fakeDom).find('meta[name=description]').attr("content")
;

        console.log("title         = "+urlTitle);
        console.log("description   = "+description);

        // If there is a title
        if (urlTitle) {
           // If there is a description
           if (description) {
              // Extract images
              var arr = $(fakeDom).find('img');
              var images = [];
              for (var i=0; i<arr.length; i++) {
                 var src = $(arr[i]).attr("src");
                 if (src && src.indexOf("http") !== -1) {
                    images.push(src);
                 } else {
                    images.push(url+src);
                 }
              }

              console.log(images);

              if (images) {

                 var divId = "result"+new Date().getTime();
                 var result =   '<div class="result" id="'+divId+'"">'+
                                    '<a href="'+url+'" target="_blank" id="resultTit
le" class="resultTitle">'+urlTitle+'</a>'+
                                    '<p id="resultDescription" style="height: 15px;"
>'+
                                    '<div class="resultImg" style="background-
image: url('+images[0]+');"></div>'+
                                    '<span class="resultDescription" style="margin:0
px 15px 0px 0;">'+description+'</span></p>'+
                                    '<p id="resultUrl" class="resultUrl">'+url+'</p>
'+
                                '</div>';

                 $("#"+targetId).append(result);

                 var i = 0;
                 setInterval(function() {
                       var el = $("#"+divId).find(".resultImg");
                       $(el).css('background-image', 'url("' + images[i] + '")');
                       i = i + 1;
                       if (i == images.length) {
                         i =  0;
                       }
                 }, 1000);
```

```
88.                } else {
89.
90.                   var result =   '<div class="result">'+
91.                                     '<a href="'+url+'" target="_blank" id="resultTit
    le" class="resultTitle">'+urlTitle+'</a>'+
92.                                     '<p id="resultDescription" style="height: 35px;"
    ><span class="resultDescription">'+description+'</span></p>'+
93.                                     '<p id="resultUrl" class="resultUrl">'+url+'</p>
    '+
94.                                  '</div>';
95.
96.                   $("#"+targetId).append(result);
97.
98.                }
99.             } else {
100.                   var result =   '<div class="result">'+
101.                                     '<a href="'+url+'" target="_blank" id="resul
    tTitle" class="resultTitle">'+urlTitle+'</a>'+
102.                                     '<p id="resultUrl" class="resultUrl">'+url+'
    </p>'+
103.                                  '</div>';
104.
105.                   $("#"+targetId).append(result);
106.                }
107.             } else {
108.                // Display link as usual
109.                $("#"+targetId).append('<div class="result"><p id="resultUrl" sty
    le="text-align:left;margin-top:0px;" class="resultUrl">'+url+'</p></div>');
110.             }
111.
112.          };
113.
114.          return _myLibraryObject;
115.
116.       }
117.
118.       // We need that our library is globally accesible, then we save in the win
    dow
119.       if(typeof(window.linko) === 'undefined'){
120.          window.linko = myLibrary();
121.       }
122.
123.    })(window); // We send the window variable withing our function
```