The Practice of Quick Browsing of PJAX on HTML5

Xiaorui Yue

Hubei Urban Construction Vocational and Technological College, Wuhan, Hubei Province, China

Keywords: HTML5, AJAX, PJAX

Abstract: The traditional technology to access webpages without refresh by AJAX has some drawbacks. First, the address bar cannot show the different address of the different pages. Second, it cannot use the browser's forward and back buttons to go through the pages. The new APIs, the history. PushState and history. Replace State, which introduced by HTML5, can change the page URL while accessing the page without refreshing. Combined with AJAX technology, the above problems can be solved and the access efficiency will be significantly increased.

1. Introduction

When creating a web site with a large number of visits, in order to improve the user experience and reduce the time for customers to wait for the page to refresh, there are many server-side technologies while the client-side technology is relatively rare. The most famous one is to use AJAX to achieve partial refresh of the webpage. It can effectively reduce the data that needs to be transmitted on the network, thereby improving the speed of webpage access.

Users often want to get updated information about the web page, such as to get the latest mailing list, display the latest mailing list, replace the user's login verification code, verify the user name is available when the member registers, the number of online users of the website, and so on. However, traditional AJAX refresh page browsing often encounters the following problems:

(1) Ajax can change the local content of the page without refreshing, but does not automatically change the URL of the page.

(2) In order to improve the accessibility, after the content changes, we usually need to use a script to manually change the URL hash. But the URL hash cannot handle the browser forward, backward and other issues.

(3) Although some browsers introduced the on hash change interface to solve the problem of automatic hash update, the browser that is not supported can only periodically determine whether the hash changes, which affects the efficiency.

2. Introduction of New PJAX Technology in HTML5

In order to solve the problems caused by traditional Ajax, a new API is introduced in HTML5, namely history. pushState, history. replaceState. The browser history can be manipulated through the pushState and replaceState interfaces, and the URL of the current page can be changed.

The pushState is to add the specified URL to the browser history, and replace State replaces the specified URL with the current URL.

The specific javascript code using pushState is as follows:

```
var state = {
   title: title,
   url: options.url,
   otherkey: othervalue// other data
};
window.history.pushState(state, document.title, url);
```

In addition to the necessary title and url, the state object can also add other data, such as, to save some of the configuration to send ajax. replaceState and pushState are similar.

The onpopstate event is provided on the window object, and the state object passed above will be

  DOI: 10.25236/eeems.2018.048

the sub-event, so that the stored title and URL can be obtained.

```
window.addEventListener('popstate', function(e){
  if (history.state){
    var state = e.state;
    // Operate(state.url, state.title);
  }
}, false);
```

By using pushState to solve the problem of URL update and combining AJAX to transfer webpage data, it can better perform no refresh browsing.

3 The Basic Principles and Implementation of PJAX

3.1 The basic principles of PJAX

PJAX is a wrapper around ajax + pushState that makes it easy for programmers to use pushState technology. Both cache and local storage are supported, and local data is read directly during the next access without having to access it again.

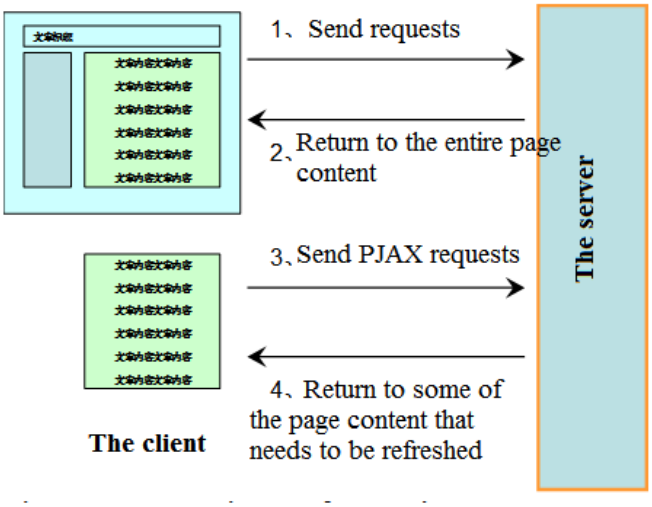

Figure 1 Comparisons of normal HTTP requests and PJAX requests

Figure 1 shows the differences in access between a normal full refresh page and a PJAX page. 1 and 2 show that when the user clicks on the hyperlink in the web page, the entire page is retrieved from the server; 3 and 4 show that when the user uses PJAX and the user clicks on the PJAX hyperlink in the web page, it is sent to the server. The request contains PJAX information, and the server returns information about the locally changed portion of the web page.

3.2 The implementation of PJAX

In specific implementations, the server first determines whether the client's request is a PJAX request. If it's "Yes", the returned content should be the relevant part of the container, and if it's "no", the entire page content is returned. The entire implementation process is described as follows:

(1) To add in the project's Gemfile:

gem 'rack-pjax'

(2) To add the property 'data-pjax-container' to the div container that needs to be partially refreshed in the view. For example, in the layout file application.html.erb, we can modify the div element with the id main:

```
<div id='main' data-pjax-contaioner>
  <%= yield %>
</div>
```

(3) To add the following in the js file of the page,

```
$(function() {
  $(document).pjax('a', '[data-pjax-container]');
});
```

The 'a' in the second line of the above code indicates that all hyperlinks on the page are wrapped in pjax, and '[data-pjax-container]' is the markup of the partially refreshed container.

If there are other hyperlinks in the page that do not need to use pjax, we can add different attribute tags, such as data-remote, data-behavior, data-skip-pjax. According to different requirements, we can modify the above code:

```
$(function(){
  $.pjax('a:not([data-remote]):not([data-behavior]):not([data-skip-pjax])',
'[data-pjax-container]')
});
```

At this point, all configurations have been completed. While using Chrome to access the webpage, debugging through the development tools, we can see that the entire webpage refreshes very quickly, and the images and js files outside the in the webpage are not transmitted over the network.

The following paper compares the number of requests sent by the client and the time the server responds when accessing web content when using PJAX technology and not using PJAX technology. We observe network transmission abilities in the developer tools that come with the Chrome browser to get information about network transmissions.

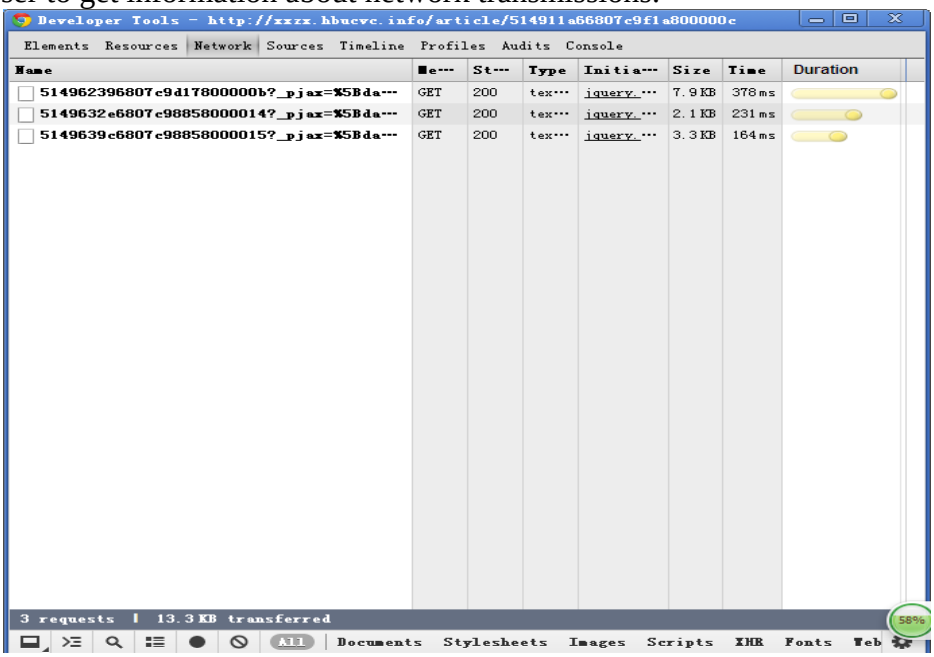

Figure 2 Network transmission results using PJAX to access three different pages

Figure 2 shows that only three partial refreshes are transmitted on the network when PJAX requests three different pages. The transfer sizes are 7.9KB, 2.1KB, and 3.3KB. Respectively, the response times are 378ms, 231ms, and 164ms.

Figure 3 shows that while accessing the same page without using PJAX technology, we not only need to return the content of the web page, but also request the CSS files and JS files needed to build the web page, which takes a long time. The test results are shown in Table 1.

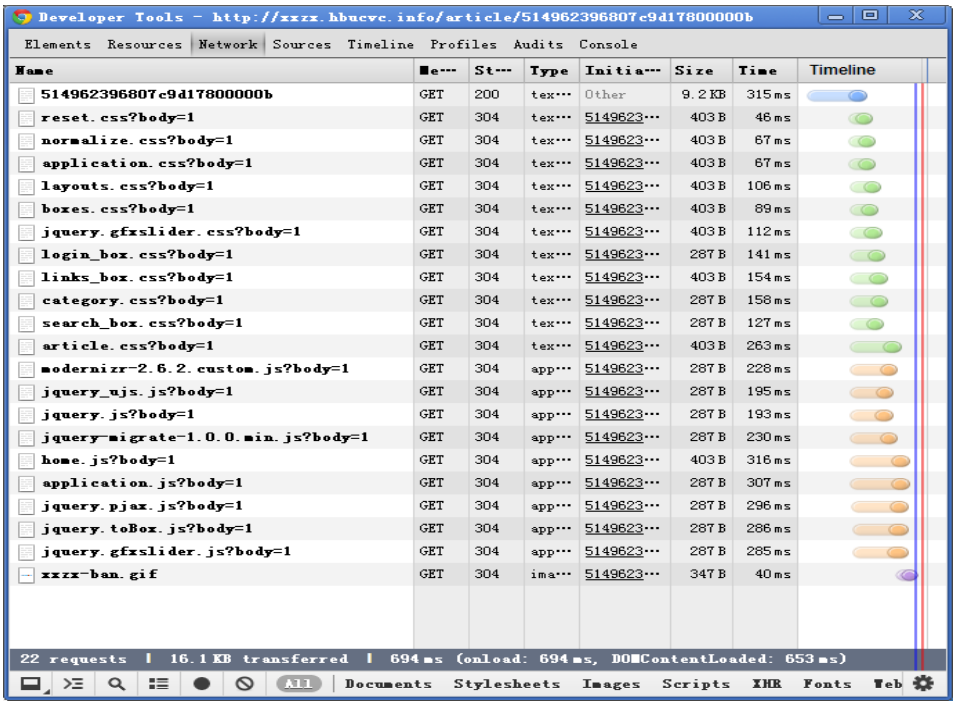

Figure 3 Network transmission results when using ordinary HTTP to request a web page

Table 1 Comparisons of assessing web pages by using PJAX technology and not using PJAX technology

Web page URL (http://xxzx.hbucvc.info/article/)	Using PJAX technology			Not using PJAX technology		
	Request number	Number of bytes returned	Response time	Request number	Number of bytes returned	Response time
514962396807c9d17800000b	1	7.9K	387ms	22	16.1K	1374ms
5149632e6807c98858000014	1	2.1K	231ms	22	11.3K	1184ms
5149639c6807c98858000015	1	3.3K	164ms	22	12.6K	1549ms

Through the comparisons of the above examples, it can be seen that using PJAX technology can complete partial page data update with only one request. Without PJAX technology, even if only updating local data, it requires up to 22 requests, including CSS and JS file data that does not need to be updated, so as to complete the update of the page data. The request time using the PJAX technology is one-quarter to one-fifth of the normal HTTP request time, which has significantly reduced the server's pressure.

4. Conclusions

Using AJAX to reduce the amount of network traffic when accessing web pages, and improving the user experience has been the main technology in web development. However, the shortcomings of AJAX technology are also very obvious. It is mainly reflected in the fact that once the user clicks the "back" button on the browser, undesired actions often occur; at the same time, the search engine cannot well understand and support AJAX. PJAX technology utilizes the latest pushState method provided by HTML5. Combined with AJAX, it can support the browser's back-up history function while implementing the local refresh function of the webpage. The page access efficiency is significantly improved, which greatly reduces the server pressure.

Acknowledgement

This research is a provincial project of Hubei Province. Project No.: 2017GA078.

References

[1] Sam Ruby, Dave Thomas, David Hansson, David Heinemeier Hansson. Agile Web Development with Rails Third Edition, Pragmatic Bookshelf, 2009, P19-25.

[2] Jonathan Chaffer, Karl Swedberg. Learning jQuery, Packt Publishing, 2007, P45-47.

[3] Bear Bibeault, Yehuda Kat. JQuery in Action, Manning Publications, 2008, P20-22.

[4] Manipulating the browser history https://developer.mozilla.org/en-US/docs/DOM/Manipulating_the_browser_history.

[5] Building Super-Fast Web Apps with PJAX http://ntotten.com/2012/04/09/building-super-fast-web-apps-with-pjax/.