Adaptation-based Web optimization

Suiyu Zhang, Cheng Yang, Chen Li
Science and technology Department
Communication University of China
Beijing, China
E-mail: ajsxzsy@126.com

Keywords: webpage optimization ; adaptation; webpage loading; webpage rendering.

Abstract. More and more terminals including mobile phone, Tablet PC, STB and PC browser are used to browse web. With the difference in network speed, screen size and interactive mode, there're many adaptation problems to be solved. When users access the Internet resources by mobile terminals, they face the problems such as nonfluency, latency and low-response. Based on page-load performance adaptation， we propose some improved models of loading and rendering.

Introduction

With the development of mobile telecommunication network and the intelligent terminal technology, people visit webpage on more and more different terminals. However, many webpage can't give the best experience to users. Bad adaptive capability to different screens and low performance in loading and rendering are terrible for users to browse the pages.

Webpage needs a better mode which is efficient but different from traditional mode. Through the way such as JS-merge, DOM-switch, priority loading we take them as practice on the different terminals, selecting the excellent mode that loading and rendering webpage.

Main problems of webpage-loading

By using Baidu, Taobao, iQiyi and other major mainstream sites on the mobile terminals, to sum up, the main problems found in the pages of mobile terminals are as follows:

(1) Long Loading time. Too many unnecessary requests are sent, which leads to the long loading time.

(2) Non-priority loading. User hasn't seen the element such as image, video or other rich media, but browser spends much time to load and render it.

(3) Long waiting time in page switching. When we click on a link, many pages use the URL jump. Page jump process has a bad user experience. The most important problem is terminal needs to refresh all DOM of the page before show a page.

(4) Too many JS requests. As the logic script for the webpage, JS is indispensable. For easy-maintenance, pages load many JS files, but it's terrible for users' view.

DOM-switch model

A. No URL- jump architecture

Traditional webpage use URL jump to switch different pages. But this model is very low performance for mobile terminal. After jumping, fully loading and rendering the new page is a great waste, although the new page is much similar to the previous page. Also, screen sizes of the mobile terminals limit the amount of information that can be displayed, in most cases, refreshing the all the elements is completely unnecessary.

No URL-jump architecture solves the problems described above. We use JS manipulating DOM elements to switch the page or change the dynamic element. So, the same elements don't need to refresh that lead to better user experience and loading optimization. For example, when the user clicks a button, we just manipulate the DOM of the modified block content, while other elements don't change.Fig.1. gives an example.

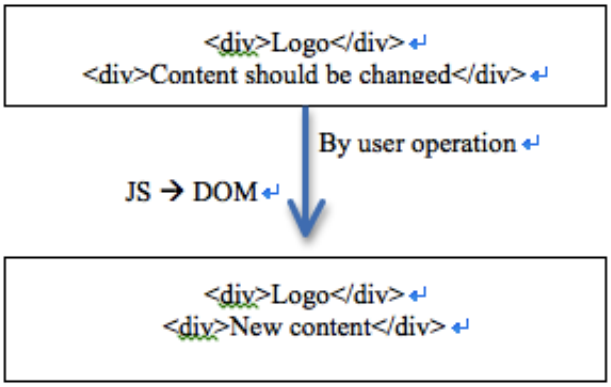

Fig. 1. No URL- jump architecture

B. No URL- jump architecture with AJAX

AJAX means Asynchronous JavaScript and XML (Asynchronous JavaScript and XML). AJAX allows web pages to implement asynchronous updates. We update a part of the page to replace reloading the entire page.

Without affecting the contents of the load, Ajax can exchange a small amount of background information, we can make the operation of the DOM elements through the JS to achieve update. This feature is well met in the demand for data exchange in mobile terminals. We can do the following by AJAX:

1) Partial DOM update. Update a small portion of the content to feedback the user as needed.

2) Submit interaction. Submit the form and give feedback without affecting the overall page. We also give a "submitting" tip to the user.

3) Delayed Loading. We show the main content firstly, when the user did some operation, through AJAX to load more content, which may include images, videos and so on. Via AJAX, we achieve the interaction between the users and the web servers, while not causing pressure to the mobile terminals or non-smooth experience.

Priority-based loading

Different browsers have different loading sequence to elements. We often use loading sequence to elements to measure browsers. Loading sequence to media and to DOM can improve pages' loading efficiency and user experience. But loading is just a basic strategy to the most mainstream browsers. Because browsers can not know each element's location and display time, so that they can only choose to load in turn. There are several models to improve loading performance by adaptation.

C. lazy-loading strategy

After testing, we obtain the following innovative strategies which can make pages display more efficient:

1) Setting meaningful indentifier in DOM to give media accurate loading sequence. When last element does not load successfully, next one will not be loaded. So that page frmae and prior elements have the fastest loading.

2) Some small icons synthesize a image, using CSS to display separately, thus reducing the number of requests sent to the servers, effectively reducing latency.

3) According to the size of the image to load the corresponding image.

D. Media loading based on terminal

According to devices' resolution and size of screen, it can load different resource. Because the media itself has a certain size, when the resolution is out of the size, it will not bring better experience and will reduce the loading efficiency. So vector is proposed to perform simple graphics and SVG is proposed to perform simple animation on mobile devices.

This model can well match with lazy-loading. We would not put resource's src in DOM when using lazy-loading. But we save all resources' addresses and load them as we wish. Before loading, all resources' address are put in media adaptation layer and choose address according to device. The main process of priority-based loading is shown in Fig. 2.

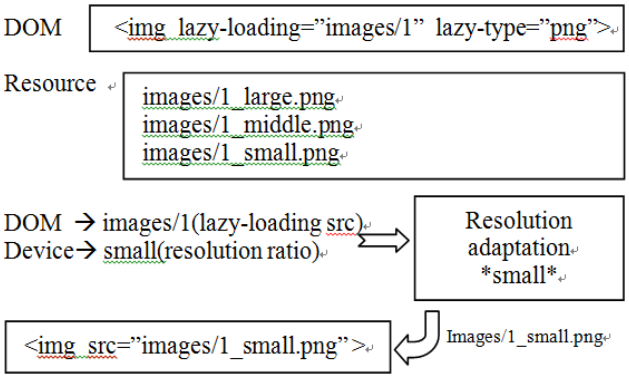

Fig. 2. lazy-loading strategy and loading based on terminal adaptation.

CSS&JS-merge model

E. Create CSS by JS

CSS provides style for pages and most pages would introduce several CSS files, but every introduction will generate a request, it will impair the loading efficiency. CSS files are compressed and stored in JS files in the form of string. Through JS files inserting CSS, it does not need generate a request. We can insert different styles to meet pages' need and reduce the pressure of rendering. An easy example is given in Fig. 3.

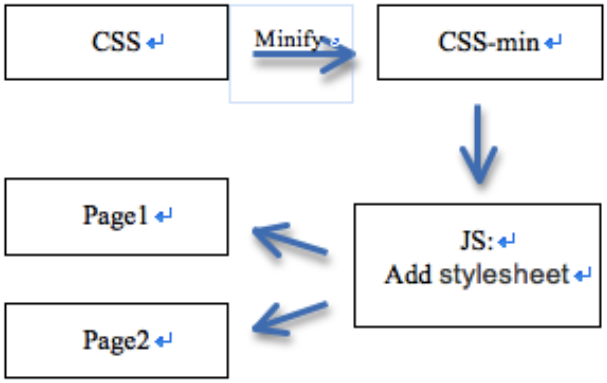

Fig. 3. Create CSS by JS according to different requirements of page

F. Merge JS files

Multiple JS requests would impair the loading efficiency as well. So after compressing, JS files are merged, so that a request could get all contents in JS files. Compression, merging and introduction could solve the problem for the lightweight frame. But for heavyweight frame, we can use gulp to manage projects, and compress and combine JS files automatically in building process, it would improve efficiency. At the same time, the server can provide interface of merging files and agree with the front, then the front delivers the request to server by parameters unite request, the server merges and combines several files, then return in one time. The only shortcoming is that code's readability and maintainability are impaired for merging, but gulp and other project tools can overcome this problem.

Performance test

G. Experiments and comparing on No URL-jump and traditional jump

Figure 4-1,4-2 show all elements from the two jumps need to reload. In the traditional page, all elements were re-rendered, which explains why traditional jump takes more time. While No URL-jump page only load new elements when most media don't need to be loaded again.

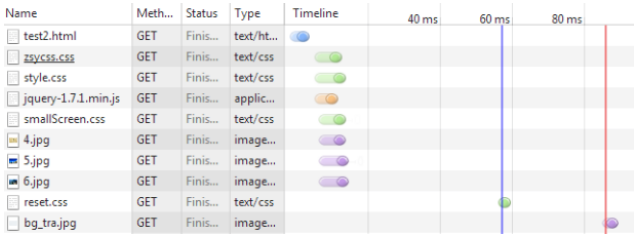

Fig. 4-1. Work load of traditional jump.

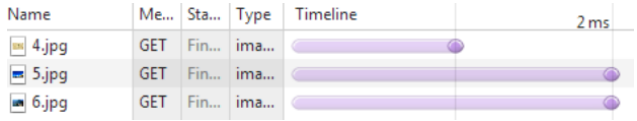

Fig. 4-2. Work load of No URL jump .

H. lazy-loading strategy performance test

When a page uses lazy-loading strategy to load media, the load speed of first sight area is the most important part of performance index.

Fig.5 gives the time of all the elements could be seen in traditional page and page used lazy-loading strategy.

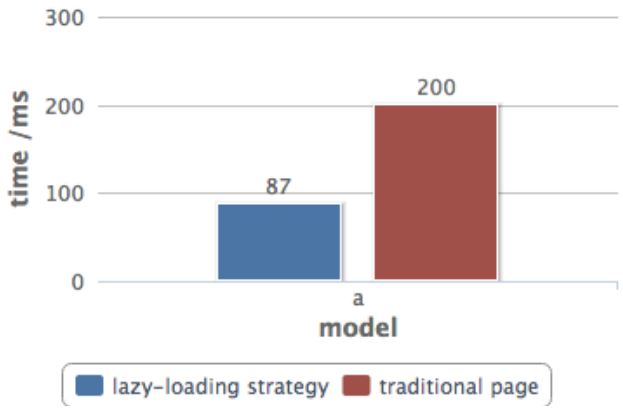

Fig. 5. Loading time of lazy-loading strategy and traditional page.

According to the figure above, obviously page used lazy-loading strategy has shorter loading time, because it only load the visible elements.

I. JS-merge model performance test

When a page uses CSS&JS-merge model to load JS files, the load speed of all JS files' completion is the most important performance index.

Fig.6 gives the test results.

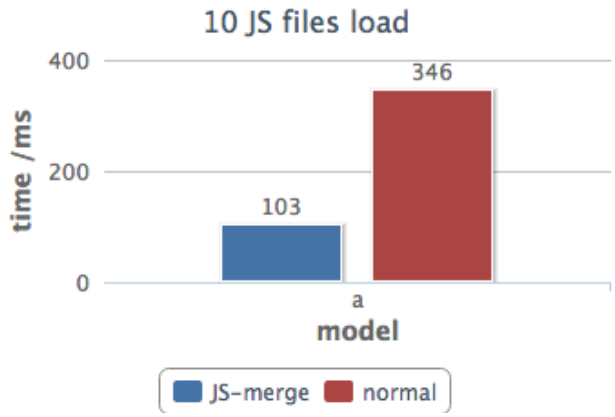

Fig. 6. Loading time of JS-merge model and traditional model.

Summary

With the popularity of smart devices, web adapter has a more and more important role. According devices' adaptation to choose loading models, it will improve efficiency of access and user experience, especially for mobile terminals. Although the above several models could make webpage's performance more excellent, it needs to choose the suitable model according to the pages' characteristics. Priority-based loading is suitable for mobile terminal's pages. No URL-jump architecture is suitable for mobile terminal's pages and the pages seeking better display effect. CSS and JS-merge are suitable for all pages if the efficiency and maintainability can reach a balance.

Acknowledgment

(1) National Science and Technology Supporting Program of China: The Research Of Streaming Media Service Based On Cloud Computing(2014BAH10F02)
(2) University Research Program of Communication University of China: The Study On Video Recommendation Service(3132015XNG1522)

References

[1] Lina Zhou; Bensal, V.; Dongsong Zhang, "Color adaptation for improving mobile web accessibility," *Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on* , vol., no., pp.291,296, 4-6 June 2014

[2] Haoyu Wang; Junjun Kong; Yao Guo; Xiangqun Chen, "Mobile Web Browser Optimizations in the Cloud Era: A Survey," Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium, pp.527,536, 25-28 March 2013

[3] Guirguis, S.K.; Hassan, M.A., "A smart framework for web content and resources adaptation in mobile devices," Advanced Communication Technology (ICACT), 2010 The 12th International Conference, vol.1, pp.487,492, 7-10 Feb. 2010

[4] Espada, J.P.; Garcia Diaz, V.; Gonzalez Crespo, R.; Pelayo Garcia Bustelo, B.C.; Cueva Lovelle, J.M., "An intelligent Mobile Web Browser to adapt the mobile web as a function of the physical environment," *Latin America Transactions, IEEE (Revista IEEE America Latina)* , vol.13, no.2, pp.503,509, Feb. 201

[5] Albasir, A.; Naik, K.; Abdunabi, T., "Smart mobile web browsing," Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference, pp.671,679, 2-4 Nov. 2013