How Developer's Code Aids in Slower Rendering of Webpage

Shekhar Gupta[1]

[1]Department of Information Science
PESIT-Bangalore South Campus, Bangalore, India

Abstract: ***In today's era of Internet based society with slower browsing speed, everyone wanted webpage on the go. While researchers & communities are streamlining on performance optimizations at atomic level, its sole duty of developers to write efficient code for faster rendering of webpage & limited battery consumption. This paper presents hidden malicious code sections responsible for slower loading of webpage. Our survey from Chrome Tools done on various sites depicts how our proposed technique speeds up the webpage rendering by 21%-49%. Even though factors like selection of proper hosting plans, browser caching & facilitating CDN goes long way for seamless user experience. Apart from it, this paper brings out some business strategies for ranking your page so as to have more hits for monetization!***

Keywords: CDN (Content Delivery Network)

1. Introduction

In today's smartphone dominant society, unpleasant user experiences not only occur due to application, database, servers and infrastructures tuning but also due to the time it takes to render the web page and displaying its contents to the end-user. In 90's website performance optimization means optimizing server-side as most web sites were static (only HTML document) and almost all the computations was done on server side. With Web 2.0 technologies, wherein web applications are becoming more & more dynamic, client side processing are even more critical from performance perspectives because they have proven to be more impactful on user experience.

Researchers at Yahoo revealed that on average only 10-20% of total page loading time is spent on the back-end and remaining 80-90% time is spent on the front-end. Even the web analysts proposed that client (user) side should be prioritize as well besides server side processing. The complex architecture of Web application forces the performance engineers to rethink & rebuild the optimization strategies for performance testing. The statistics revealed while improving the back-end performance by 50% boosts the overall application performance up to 10% while application performance can be tuned by 40% by reducing the front-end time to half.

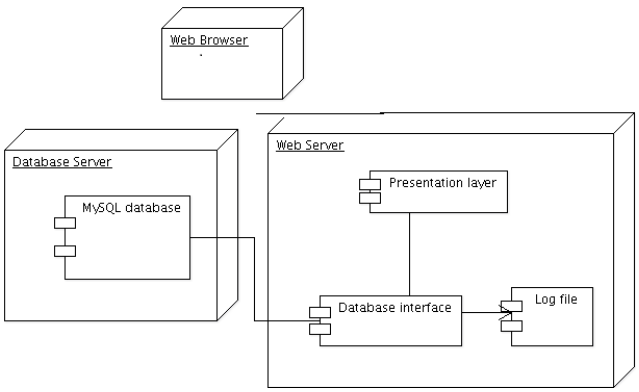

Figure 1: The block diagram showing the various components interaction in sandboxed mannerism [7]

The advantage for front-end performance optimization is that it's quite simple and cost effective as compared to back-end performance optimization wherein the developers are redesigning application architecture, adding or modifying hardware, distributing computations etc.

2. Identifying the Hidden Malicious Code Sections

What exactly are hidden malicious code sections? Well, there is no such set standard for the same, that's wherein the researchers & analysts come to play. To break it down these are modules of the code which hinders the loading of the webpage. This happens when the developer just make the things happen, without considering the after-effects. Let's look at some of the those critical sections.

A. CSS Expression

We know that CSS (Cascading Style Sheet) helps in rendering the plain boring HTML document dynamically. To elaborate let us take an example of the CSS expression which sets the background color after every 30 minutes. CSS expressions are evaluated very frequently and they are evaluated whenever any user action is performed. They are evaluated when the page is rendered and resized, page scroll down and even on mouse hover. They are so frequently evaluated even on moving mouse around the page can generate more than 10,000 evaluations. So one can predict the amount of computations one can pose if even single-lined CSS expression is not managed properly!!

B. JavaScript Libraries

With increasing number of websites using active code, the available programs are becoming larger and more complex. These programs make extensive use of JavaScript libraries and asynchronous communication, which results in larger programs thereby increasing workload on the browser. These applications provides greater interaction via asynchronous communication with a server. This allows programs to fetch and display new data without requiring full-page refresh.

For the quantification of use of asynchronous communication, we checks Alexa data sets for the website using XmlHttpRequest (XHR) class in JavaScript. The amount of code distributed with each map site while displaying the 560100 zip code shows 23.9 KB of code for traditional sites and 234 KB for interactive sites.

C. Amount of Additional Resources

Web pages are bigger than ever. According to the HTTP Archive, the average Web page contains more than 1 MB along with 80 resources such as images, sprites, JavaScript, CSS (Cascading Style Sheets) files, etc.

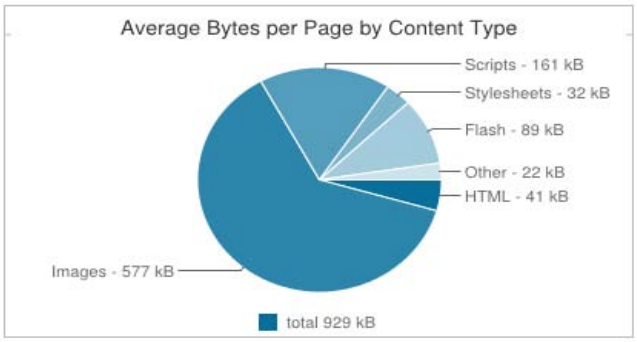

Figure 2: The average bytes distribution per page based on the various resources or contents present in any given webpage.

Only 20% of the time is required to display typical Web page by loading the HTML portion of the website. The remaining 80% is spent for loading the additional resources needed to render the page - including style sheets, script, files and images and performing client-side processing.

D. Browser's Concurrency

There's no assurance that even if our code is fully optimized & reliable, the user experience would be wonderful, even though all optimizations are done. It may be awful! This factor is attributed to the browser's three factors.

1) Poor isolation mechanism- One web application executing within the browser can interfere with another application with bug in that application.
2) Scheduling starvation- When one overly aggressive application prevents others from running.
3) Memory starvation- The memory consumption of one application prevents others applications from making good forward progress. Therefore, for us as developers how much pathetic it would be if we are crashing other's application.

3. Detection of Malicious Code Sections

For these never-ending problems web communities conducted loads of experiments, to conclude the importance of one parameter over the other, so that developers need not to worry about everything, but only something!!

Figure 3: The timing disribution of webpage rendering for serving frontend, backend & content download

A. Google's page reduction

Google experimented with showing 30 results/page instead of 10 results/page. In this experiment, Google's traffic and revenue dropped by 20 percent apparently since the pages with more results took just half-second more to load.

B. Threshold webpage response time

In 2009, the studies by Forrester Research by Akamai identified two seconds as the threshold value for acceptable webpage response times and proposed that 40 percent of consumers abandon a page that takes longer than three seconds to load.

C. Network strength

A study conducted over 200 leading e-commerce sites by Strange loop Networks found that the median load time was 11.8 seconds over 3G while performance over LTE only at 8.5 seconds.

D. DNS Lookups

According to Yahoo Developer Network Blog, it took about 20-120 milliseconds for DNS (Domain Name System) to resolve IP address for a given hostname or domain name and the browser cannot do anything until the process is properly completed.

4. Best Suggestions, Tips & Tricks

Let's look at the better side of the web development phase wherein best tips, tricks & suggestions are proposed. This portion suggests that how the placement of code sections within the webpage makes tremendous impacts!!

A. Placement of Style Sheet: Yahoo researchers discovered that putting the style sheets to the document HEAD enables the browser to render progressively and it makes the page loading faster. The core idea behind it's that rather than letting the user to wait for rendering all the page elements and glancing at white screen, it would bring out good user experience to display webpage to see the page components

gradually instead of waiting and then viewing all the components suddenly.

Few modern browsers including IE doesn't perform the progressive rendering on web page components on putting the style sheets at the bottom and frustrate the user with blank page!!

B. Minification: The removal of unnecessary characters & all the additional sources like comments, new line commands, metadata, white spaces, new line commands etc. from the code is called minification. It results in reduction of web page size and downloads time for that webpage.

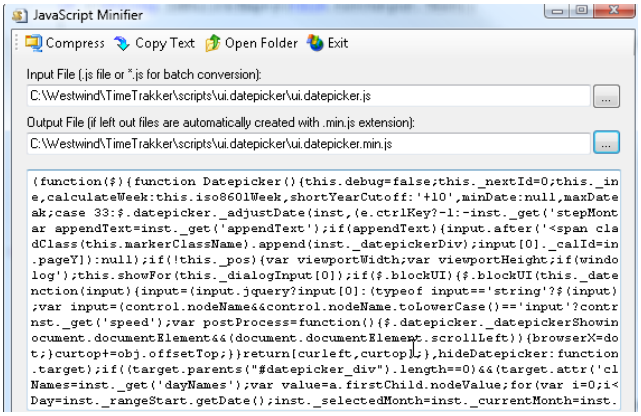

Figure 4: It shows the minification of Java Scripts with compression reduction 33.19% [8]

C. Image Optimization: Image loading time is one of the major performance issues affecting page load on mobile devices. The use of online image optimizers, such as smushit.com & choosing the appropriate file format would help. Normally JPG image format is used with high number of colors while PNG is best suited for rendering text and for images with alpha transparency.

D. Using Content Delivery Networks: A content delivery network (CDN) is a collection of web servers distributed across various locations all around the world to provide web contents in an efficient manner. Based on lesser number of network hope counts, user request should be entertained from the closest web server.
If the application is deployed on a single place, it can greatly affect users experience for accessing the application from longer distance due to network delays. User response time can be greatly improved by just redistributing static web contents over various locations instead of redesigning the application to distribute the dynamic contents.
Some large internet companies have developed their own CDN but it's not cost effective decision for smaller companies so there are various CDN service providers in market whose services can be used to optimize the end user response time.

E. Minimizing the HTML Requests: An HTTP request is used to fetch root HTML document that may refer to other page resources like images, scripts and style sheets. Each of these resources must be fetched with HTTP request. However, every HTTP request adds performance overhead since it creates network traffic between the client and server. Reducing the number of resources will decrease the HTTP requests required to render the web page and will improve the performance.

Figure 5: The timing distribution for miscellaneous activities [9]

F. Use image sprites: This technique enables to combine several images into one and use CSS to show only the part of the image that's needed. When you combine five or ten images into a single file, already you're making savings in huge overheads for requests & response.
The process of combining all the scripts and style sheets into a single script and style sheet respectively is a challenging task but would greatly help in achieving the desired goal on performance optimization.

G. Turnoff Entity Tags: Entity tags (ETags) are used for validation in updates for browser cache data. These tags compares the browser cached copy with the one on server cache side to make sure browser has updated data. These tags have limitations that it only compares the browser cache with unique server. All works fine when there is only one hosting server while it won't work in situation wherein application is hosted on multiple servers and browser gets the components from one server and validates the same on another server. The generation of ETags is achieved through IIS and Apache so the same components don't match from one server to another and user ends up in receiving 200 response code rather than small, fast 304 response. Therefore, the ETags are turned off when the application is hosted on multiple servers.

H. Use the Expires Header: The static assets in code should be exactly static for better performance. Therefore, there should be no dynamically generated scripts, styles or <img> tags pointing to scripts that generate dynamic images. For the dynamic generation of graphics containing visitor's username, result is cached as a static image. This image is replicated once when member signs up & then images gets stored on the file system and the path to the image in database is written. Static assets allows to set Expires header for those files to near-future date, so that once assets are downloaded, it's cached by the browser and never requested again.

The Expires header in Apache can be enabled by adding .ht access file containing the following directives:

Expires Active On: This directives enables generation of the Expires header.

Expires Default "modification plus 1 month": This directive sets the expiration date to one month after file's modification date.

I. Code compression: The processes of compression of JavaScript and CSS files potentially have significant impact on performance. For compression refresh-sh.com can be utilized as tool as all modern browsers support compression

& compressed resources. All HTML, JS, CSS and XML documents can be compressed on server side before transferring to the web browser wherein decompression of these documents happens before displaying them to end user. Compression can be easily enabled on most of the web server through some basic configurations. The binary files like images, PDF and SWF should not be compressed again because compressing the already compressed elements would waste CPU utilization and can also increase the file size as well.

J. Make JavaScript and CSS as External files: The JavaScript and CSS files are cached by the browser so enabling them as external files makes the page response time faster. These files are in lined with HTML document & because of external files caching is achieved with number of HTTP requests remains the same.

K. Use Post-load Pre-loading and Inline Assets: We use HTML for content, CSS for presentation and JavaScript for behavior. These assets are kept in separate files for better performance. However, homepage must be the fastest page on website as visitors may escape our site, irrespective of its contents, if homepage is slower to render. With empty caching on first-time visit, if only one request serves the purpose its optimized solution. This way of Post-load Pre-loading loads the components in the background after the home page has loaded.
Suppose **home.html** is homepage & **mystuff.js** being JavaScript file. Upon placing mystuff.js inline with home.html the request behind-the-scenes is achieved. Henceforth, when user hits one of the content pages, caching is done.

a) new Image().src = 'image.png'; For preloading of image wherein the image is requested but never used

b) Creating new <script> for preloading JavaScript files
```
var js = document.createElement('script');
js.src = 'merasite.js';
document.getElementsByTagName('head').appendchild(js);
```

5. Online Tools & Web Optimizers

With rapid advancement in tech are we going to do all these optimizations? Obviously no, it's not possible technically! Before few years it wasn't feasible for web developers to figure out what's happening after submission of user's request on browser.
There are 100's of tools & web-optimizers online wherein we can just put our code & everything is done for us!

1) They offer suggestions for improving page's performance.
2) They grades web page based on predefined rule sets or user-defined rule set.
3) It summarizes the web page's components.
4) It displays statistics about the web page.

The top-notch & most used ones includes Page Speed, FireBug, Yottaa.com, Y-Slow & Webpage Test

A. **Page Speed**: It is an open source Firefox/Firebug ad-on launched by Google for evaluating the web pages. It provides suggestions for minimizing web page loading time. In this service, webpages routes through Google server and algorithms are applied for making them more efficient and faster. This makes webpage retrieval faster when users access those pages through Google search engine

Figure 6: The various optimization categories available in tool

B. **Y-Slow**: Y-Slow is browser plug-in from Yahoo for testing the web page against various optimization rules defined by Yahoo performance team. It recommends best suggestions & tips for web page optimizations.

C. **Webpage Test**: It's a free online service providing the facilities of front-end speed test for websites. The speed of sites can be tested out on all the famous web and mobile browsers from different geographical locations thereby providing detailed information on all the application components which can be helpful in application optimization.

D. **Firebug**: Its browser plug-in providing various services including debugging of front end development, tracking of all the network requests and profiling JavaScript function calls. For most of the developers it's their most preferable tool for client side performance evaluation.

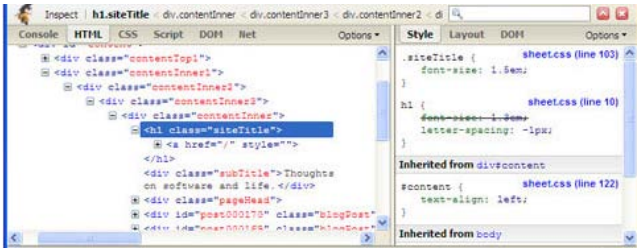

Figure 7. It shows the probabilistic malicious code section in HTML, CSS & JavaScript

E. **AgileLoad Script Editor**: It captures and analyses all the requests performed between user and application for building test scenarios. The validation of scripts generated by replaying and comparing each request with initial scenarios is done by Replay function. The graphical bar chart shows the time spent for primary requests and overall response times and details of all resources loaded, the time spent for each resources, the detailed HTTP response associated with each HTTP request.

F. **Yottaa**: Its web optimization solution providing web applications Yottaa performance score and identifies areas contributing most to the application performance.

6. The After-Effects of Browser

Web browser remains the most important applications on devices for connecting to Internet. Researchers and industry organizations propose various methods to build a better web browser.

A. Parallelization: This process helps in parallelizing the computation-intensive steps in webpage processing making browser more responsive and energy-efficient. The designing of parallel web browsers and algorithms for parallelizing each browser component, including parallelization of frontend, page layout and scripts.

B. Effective Caching: The caching mechanism used in web browsers typically is caching the web resources including HTML data, pictures, CSS and JS files. Smart Caching caches intermediate results for style formatting and layout calculation stages, henceforth the cached data can be applied in subsequent processing of same data for avoiding repetition of local computations. It reduces overhead for redundant calculations while revisiting the same webpage.

C. Webpage Prefetching.: It attempts for prediction of which webpages user may visit in near future and downloading the predicted webpages before actual visits so the browser can use resources preloaded locally on correct prediction. In Pocket-Web the visiting history of user is used for training the user access model via machine-learning approach.

D. Speculative Loading: It looks for those webpages in same website sharing common web resources. The predicted resources are loaded along with main resources, thereby saving round-trip-time. It utilizes similarities between webpages for speeding up resource loading

E. Cloud-Based Optimizations: The cloud acts as an agent for web browser and all requests between the web server and the mobile browser passes through the clouds thereby pre-processing and compression of web contents is done before sending the results to web browsers.

7. Conclusion

Well there seem no upper boundaries for optimizations! With advancement in technologies, increasing number of libraries & resources poses problem on client side & server side. Sometimes, only the difference in browsers makes huge impact on user experience. As developers, there are limitless numbers of hurdles which are mandatory on their part. Therefore, we as developers shouldn't just move on by coding endlessly rather should pose an optimized solution for bringing in the seamless user experience.

References

[1] Kaimin Zhang, Lu Wang & Aimin Pan, Bin B. Zhu Issue no. 11 "Smart Caching for Web Browsers"
[2] CSS tricks www.css-tricks.com
[3] Haoyu Wang, Mengxin Liu, Yao Guo & Xiangqun Chen "Similarity-based Web Browser Optimization"
[4] Front End Performance Testing & optimizations www.agileloads.com
[5] Trevor Jim, Nikhil Swamy &Michael Hick "Defeating Script Injection Attacks with Browser-Enforced Embedded Policies"
[6] Charles Reis, Brian Bershad, Steven D. Gribble and Henry M. Levy "Using Processes to Improve the Reliability of Browser-based Applications"
[7] Deployment Diagram: http://laurentparenteau.com/blog
[8] JavaScript Minification: https://blog.mastykarz.nl/content
[9] Timing distribution: http://volumatrixgroup.com/wp-content
[10] Kavindra Kumar Singh & Dr. Praveen Kumar "Optimizing the Performance of Mobile Web Application"