http://dx.doi.org/10.15576/GLL/2020.2.119

GLL

Geomatics, Landmanagement and Landscape No. 2 • 2020, 119–129

COMPARATIVE ANALYSIS OF SELECTED ONLINE TOOLS FOR JAVASCRIPT CODE MINIFICATION. A CASE STUDY OF A MAP APPLICATION

Karol Król

Summary

The performance of some map applications can be improved not only through the compression of raster files or appropriate data server configuration, but also by using source file minification. Minification can be more or less effective. The objective of the paper is to perform a comparative analysis of selected online tools for minifying JavaScript code and to measure the impact of such minification on the performance of a map application. Minification and performance tests were conducted on a prototype map application. The application was developed as a ZoomLens component extending the functionality of any website. Various tools yielded similar results of the JavaScript file minification, and it did not affect the values of aggregate performance indices. In most cases, it reduced the JavaScript file size by over a half. It has been demonstrated that minification of JavaScript code alone may not be sufficient to improve the application performance noticeably.

Keywords

boosting minification • bloating files • compression • performance • optimisation • ad hoc testing

1. Introduction

Broad access to communication networks caused a dynamic growth of spatial information sharing technologies and the geoinformation society [McCall and Dunn 2012]. Advanced geoinformatics technologies introduced map services that offer remote access to metadata sets, including reference data about the environment. Location and navigation portals are growing increasingly popular and serve a similar purpose to tourist, road, or city maps. Location is a very important metadata object as it conveys data on objects and events that is as precise as possible [Tsou 2014]. Official geographic information portals provide vast sets of land surveying and cartographic data, which slowly replace printed topographic or reference maps. The most significant advantage of such data is their validity and completeness. They can also be used as reference data. Additionally, more and more of them are available for free [Neis and Zielstra 2014].

Web applications for advanced geovisualisation help create maps for individual use, improving the efficiency of geographic information procedures. The appeal of a web application is related to the efficiency of information communication, which in turn depends on the usability of the map and application alike. A map is usable when it is an appropriately developed cartographic product. Usability of a web application hinges on such factors as its performance [Król 2018].

Websites have grown over tenfold in the last decade [Zhu and Reddi 2013]. This generates a demand for high-performance computer systems and mobile devices for comfortable viewing of websites that offer increasingly sophisticated functionalities. From the programmer's perspective, performance is a technical, engineering measure. It may result from design solutions or techniques and components used. From the user's perspective, performance is a measure of usability. It is the decisive parameter for website viewing comfort [Takalikar and Joshi 2016]. Map applications should exhibit high performance, especially when they are complex and highly specialised [Celentano and Dubois 2017].

The performance of map applications can be improved, among other things, through the compression of raster files or appropriate data server configuration [Król and Bitner 2019], but also by using source file minification. The objective of the paper is to perform a comparative analysis of selected online tools for minifying JavaScript code and to measure the impact of such minification on the performance of a map application.

2. What is code minification?

Yahoo! and Google have often investigated the issue of improving website performance over the recent decade. This is due, inter alia, to the fact that page speed is one of the search engine optimisation (SEO) ranking factors, and minification of source files can help improve a site's load speed. Most of the research did not involve single HTML files but all resources necessary for the appropriate viewing of the whole website. Although both CSS (Cascading Style Sheets) and JavaScript (JS) codes can be an integral part of a hypertext document, the best practice is to keep this code in external files that are downloaded and buffered separately [Farkas 2017]. The performance tests answer the question: How can these external resources be downloaded and used most efficiently? The first approach is to limit the number of external requests since the overhead of each HTTP (Hypertext Transfer Protocol) request is high. The other involves a reduction of the source code size [Zakas 2010].

When creating HTML, CSS, and JS files, developers tend to use spacing, comments, and clearly-named variables to make code and mark-up readable for themselves. However, web servers and browsers can parse file content without comments and well-structured code, both of which create additional network traffic without providing any functional benefit. Minification is a major component of front-end optimisation, a set of tools and techniques that reduce file sizes and the number of associated web page requests. It is one of the main methods used to reduce load times and bandwidth usage on websites [Imperva 2020].

It is only recently that Internet users started to pay greater attention to problems caused by poorly optimised source code. The less code is transferred, the lesser 'friction' among the server, application, and the user. More code can mean greater application complexity, which makes maintenance more difficult and hampers performance. Hence, the focus now turns to reduction of CSS and JS file sizes through such means as splitting the code into reasonable parts and their minification.

To compress, minimise or minify code means to remove such unnecessary characters as white spaces, comments, block delimiters, and other redundant data from the source code without actually affecting how the browser processes the code or resource as a whole. This is an effective technique otherwise called code minimisation. Thanks to improved application load time, it enhances the overall web performance because of a smaller file size [Lotanna 2018]. Minification (also uglification) in computer science is the process of removing all unnecessary characters from the source code without changing its functionality. Minified source code is especially useful for interpreted languages deployed and transmitted on the Internet (such as JavaScripts) because it reduces the amount of data that needs to be transferred.

2.1. The history of JavaScript byte savings

Douglas Crockford introduced JSMin in 2001 as a tool for reducing the size of JavaScript files before they are released. He called this size reduction minification. JSMin is a minification tool that removes comments and unnecessary whitespaces from JS files. It typically reduces file size by half, resulting in faster downloads. It also encourages a more expressive programming style because it eliminates the download cost of clean, literate self-documentation [Crockford 2019]. JSMin removes spaces, tabs, and comments from JavaScript files, effectively reducing their size. Three years later, a Yahoo! engineer, Julien Lecomte presented the YUI Compressor. The purpose of the YUI Compressor was to reduce the size of JavaScript files through smart source code optimisation. Apart from the removal of comments, spaces, and tabs, the YUI Compressor removes line breaks to reduce the file even more [YUI Compressor 2020]. The most significant byte savings, though, come from replacing local variable names with one- or two-character names. Zakas [2010] enhanced the capabilities of the YUI Compressor. Meanwhile, many new online applications that compress JavaScript code in the browser window were produced.

Today, minification has become the standard practice for page optimisation. All major JavaScript library developers (Bootstrap, JQuery, AngularJS, etc.) provide minified versions of their files for production deployments, usually denoted with a min.js name extension [Imperva 2020].

2.2. Methods of code minification

Most production websites use JavaScript minification but the way it is carried out varies greatly. From simple online converters to more comprehensive GUI tools, to command-line interfaces. Source code (HTML, CSS, JS, or other) can be minified manually or

automatically with tools available on the Internet, for example. With online tools, it is usually a matter of pasting the code and copying the result immediately. Nevertheless, manual minification is a bad practice and becomes virtually impossible when large files are concerned [Imperva 2020]. Although it can be gratifying to minimise code by hand (for a programmer who has control over the whole process), it can be arduous, in particular in the case of large files. GUI tools often contain many more functionalities; JS minification is just a small part of what they do [Pataki 2017]. Command-line tools are also usually more comprehensive, offering minification as a module. Moreover, content management systems (CMS) such as WordPress can manage JavaScript file optimisation in the CMS panel.

3. Materials and methods

Minification and performance tests were conducted on a prototype map application. The application was developed as a ZoomLens component extending the functionality of any website. A ZoomLens or interactive magnifying glass is a specific tool for navigation. Its primary function is to present an image file, usually a raster, with high object density [Król 2019]. The component was developed as a fileset so that it can be implemented in a structure of any hypertext document.

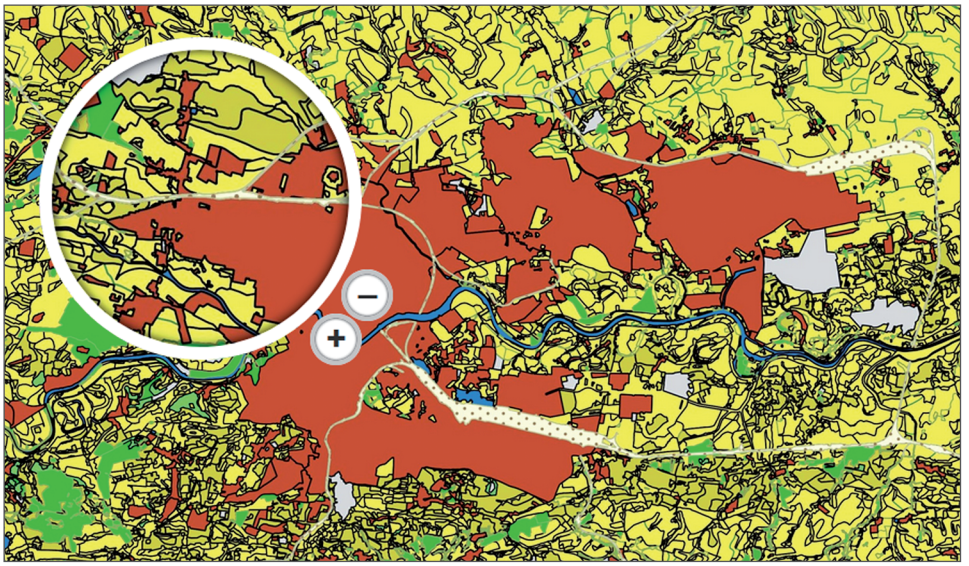
Most portals today are called mashups. In web terminology, a mashup is a website that combines content from more than one source (from multiple websites) into an integrated experience. Mashups are content aggregates that leverage the power of the Web to support worldwide sharing of content that would not have been easily accessible conventionally or reusable in different contexts or from different locations [Murugesan 2007]. Thematic maps are popular as well (also called special-purpose maps, single-topic maps, statistical maps). Many Internet users work with thematic maps and use them as a source of information or a method of data presentation.

The application employs jfMagnify (Fig. 1). jfMagnify is a jQuery plugin that creates a magnifying glass effect. This plugin will magnify HTML content as well, not just images. It does this by cloning an identified element and its children, scaling it to the configured specification, and then appending to an identified container element [Fahnestock 2020].

Performance tests were conducted on the application before and after JavaScript code minification. Code minification was performed using selected online tools (Table 1).

The JavaScript minification applications were selected so as to meet usability criteria: a) easy to use – code is minified in a web browser window as a thin client. Infrastructure and software necessary for the application to work are ensured by the service provider. The code to be minified is simply pasted into the minifier window. Whole files can be uploaded as well. Minification takes from several to over a dozen seconds depending on the file size; b) completely free – the tool is free to use and no user account has to be created. No programmes or components have to be installed on the client machine or server; c) safe and secure – communication with servers should be SSL-encrypted (https). Uploaded files are removed from servers of the service provider immediately after minification and the resulting JavaScript file is removed immediately after the first

download attempt or after 15 minutes of idle. Files sent for minification are not stored or explored in any way. Application's principles of operation are described in terms and conditions and privacy policy.



Source: Author's own work using jfMagnify [Fahnestock 2020]

Fig. 1. A ZoomLens-type component interface, an agricultural soil map of Kraków (print screen)

Table 1. Online tools used for minification

ID	Tool	Website address	Full name (original spelling)
1	JavaScript Minifier	javascript-minifier.com	Online JavaScript Minifier/Compressor
2	Minify	minifier.org	Minify – JavaScript and CSS minifier
3	JSCompress	jscompress.com	JSCompress – The JavaScript Compression Tool
4	JS Minifier	minify.one	JavaScript minifier
5	JS Minify	uglifyjs.net	JS Minify and Beautify

Source: Authors' own study

The performance tests were conducted using selected online applications (Table 2). The most important factor in keeping first-time visitors on the website after the initial click is the page load speed. Pingdom and GiftofSpeed can identify bottlenecks, meaning components that adversely affect the website load time.

Table 2. Online tools used in the performance tests

Performance tool	Website	Key performance indicators (KPIs)
Pingdom Website Speed Test	pingdom.com	1. Performance grade 2. Page size (MB) 3. Load time (s)
GiftofSpeed Website Speed Test	giftofspeed.com	1. Content visible (ms) 2. Fully loaded (ms) 3. Optimisation Score

Source: Authors' own study

Pingdom's performance grade is the general final note, usually a concise test summary. It is a general note, cumulative and rounded off. The page size index measures the size of the website's components and load time shows the general load time of the website (according to Pingdom). GiftofSpeed's content visible (ms) shows the time it takes to load the DOM (Document Object Model). This is the most important load time to consider. In the vast majority of cases, when the DOM content was loaded, at least some content will be visible on the screen of the user who is loading the page. GiftofSpeed Optimization Score is the accumulation of the scores of all performance metrics (0/100 = worst, 100/100 = best).

4. Results

The largest components of the prototype application were raster (image) files. Scripts, HTML, and CSS files were next. They were very small, only a small percentage of the whole (Table 3).

Table 3. Content size by content type

File type	Per cent	File size
Image	95.78%	3.4 MB
Script	4.15%	147.3 KB
HTML	0.04%	1.4 KB
CSS	0.04%	1.3 KB
Total	100.00%	3.6 MB

Source: Authors' own study (Pingdom Website Speed Test)

The performance testing with Pingdom showed that the server response time for an attempt to open the application in a browser window was 0.7 s on average. Next, individual components of the application were loaded (requests), and the longest process was

image file download (Table 4). The size structure of the source files and the browser load and read times justified the minification of JavaScript files and compression of image files.

Table 4. Averaged measure of the wait time for application files

Component file (JS, CSS, JPG)	File size (KB)	0.8 s	1.4 s	2.1 s	2.8 s
jquery-2.2.0.min.js*	30.8				
jquery-ui.js*	115.1				
jfMagnify.css	1.3				
jquery.jfMagnify3.js	1.4				
mapa.jpg	3.4				
* Files downloaded from an external 'code.jquery.com' server					
Wait – Web browser is waiting for data from the server					
Receive – Web browser is receiving data from the server					

Source: Authors' own study (waterfall charts, Pingdom Website Speed Test)

Minification reduced the JavaScript file by 57.7% on average. The most effective was the minification with JS Minify. The program reduced the size of the JavaScript file by almost 66% while retaining code functionality. The effectiveness of the minification is proven by the number of characters in the processed source code. The largest number of characters was left after minification with Minify. The smallest number of characters and file size was rendered by JS Minify (Table 5).

Table 5. JavaScript code minification results

ID	Tool	JavaScript file size before minification*	JavaScript file size after minification*	Number of characters in the code (including spaces)	File reduction per cent
1	JavaScript Minifier	5,297	2,073	2,071	60.86
2	Minify	5,297	3,185	3,169	40.17
3	JSCompress	5,297	2,073	2,071	60.86
4	JS Minifier	5,297	2,084	2,082	60.69
5	JS Minify	5,297	1,807	1,805	65.92
		Mean	2,244.4	2,239	57.7

* File size in bytes (B) and measured with Total Commander

Source: Authors' own study

The minification of the JavaScript code did not affect the Pingdom Performance grade (in the employed research design). The size of the JavaScript file was too small for the change of its size to be registered by Pingdom's measuring scripts (relatively large round-off). The change was, however, reflected in the application load time, which was the shortest for the smallest JavaScript file (Table 6).

Table 6. Performance results as measured with Pingdom

ID	Performance grade*	Page size (MB)	Minified JavaScript file size (B)	Load time (s)
1	94	3.6	2,073	3.02
2	94	3.6	3,185	2.73
3	94	3.6	2,073	2.75
4	94	3.6	2,084	3.03
5	94	3.6	1,807	2.71

* Measurement location Asia – Japan – Tokyo

Source: Authors' own study (Pingdom Website Speed Test)

The minification of the JavaScript code did not affect the GiftofSpeed Optimization Score (in the employed research design). The change in the JavaScript file size was reflected in indices of application load dynamics. The values were the most advantageous for the file minified using JS Minify (Table 7). These results correspond to the ones from Pingdom.

Table 7. Performance results as measured with GiftofSpeed

ID	Content visible (ms)*	Fully loaded (ms)	Optim. score
1	384	803	74
2	356	766	74
3	385	783	74
4	350	762	74
5	348	673	74

* Measurement location: London (UK).

Source: Authors' own study (GiftofSpeed Website Speed Test)

Various tools yielded similar results of the JavaScript file minification. In most cases, it reduced the JavaScript file size by half. The Minify web application gave the poorest results of minification, while JS Minify produced the best effects. Each operation

involved pasting JavaScript code into the minifier window, but minification attributes could be configured only for JS Minify.

5. Discussion

Effective reporting of test results is one of the holy grails of SEO. If done correctly, it improves the project's quality and helps focus on the real issues. However, if poorly done, it adds confusion and reduces the value testers bring. Reporting results of functional tests is relatively simple because these tests have a clear pass or fail conditions. Reporting results of performance testing is much more nuanced [Stahl 2018].

Speeding up websites is important not just to site owners, but to all Internet users. Not only do faster sites improve user experience, but they also reduce operating costs, according to recent data. Eric Schurman (Bing) found that a 2-second slowdown changed queries/user by –1.8% and revenue/user by –4.3%. Jake Brutlag (Google Search) found that a 400-millisecond delay resulted in a –0.59% change in searches/user. Marissa Mayer shared several performance case studies from Google. One experiment increased the number of search results per page from 10 to 30, with a corresponding increase in page load times from 400 to 900 milliseconds. This resulted in a 25% drop-off in first result page searches. Phil Dixon, from Shopzilla, offered the most interesting statistics about the impact of performance on the bottom line. A year-long performance redesign resulted in a 5-second speed up. This resulted in a 25% increase in page views, a 7 to 12% increase in revenue, and a 50% reduction in hardware [Souders 2009].

Research by various institutions demonstrated that even a seemingly minor improvement in performance can boost target conversion. A 100-millisecond delay in a website load time can hurt conversion rates by 7%. A two-second delay in a web page load time increases bounce rates by 103%. 53% of mobile site visitors will leave a page that takes longer than three seconds to load [Akamai 2017].

6. Conclusions

Although performance optimisation brings tangible results, recent years have seen a growing, often little excessive pressure on this aspect of the web. This often leads to the form beating the substance. Optimisation focuses on going one kilobyte further at all costs. The goal is noble but only if achieved with reasonable effort and optimisation is perceptible for users.

The study has shown that it is difficult to interpret single, isolated, random results of measurements. They can be unreliable or inaccurate. Application performance can significantly vary in time. The difference in website load time can reach up to several seconds within a minute. The website development technique is a factor here as well. The more external resources are requested, the greater the performance variability.

Measurements from different locations and measurements done at different moments were noted to yield different results. Actual information about website

performance can be obtained only from continuous monitoring and measurements in unit time. At the same time, individual measurements pinpoint bottlenecks, which may be responsible for performance issues and should be observed closely during tests in unit time. Advantages of unit testing include low costs, quick results, and repeatability. Ad hoc unit tests can provide information necessary to prepare unit time tests.

Minification of JavaScript code alone may not be sufficient to improve the application performance noticeably. It may be necessary to minify HTML and CSS files as well. Note that the tested component was based on a raster file (a JPG image file). It may be the compression of the image that could bring the best performance improvement.

Therefore, code minification is not a goal in and of itself. It is a means to an end. The end being such code optimisation that the user downloads and uses only the code performing the current tasks. This ultimate objective is not achieved if the user has to download the whole code of the application, even minified, when they need only a piece of it at the moment. The removal of unnecessary characters is the right direction, but it is only important to diagnose which pieces of code are not necessary for the application to work.

Funded with a subsidy of the Ministry of Science and Higher Education for the University of Agriculture in Kraków for 2020.

References

Akamai 2017. Akamai Online Retail Performance Report. Milliseconds Are Critical. http://bit.ly/akamairap [accessed: 15 May 2020].

Celentano A., Dubois E. 2017. A layered structure for a design space dedicated to rich interactive multimedia content. Multimedia Tools and Applications, 76(4), 5191–5220.

Crockford D. 2019. JSMin. Douglas Crockford Blog. https://www.crockford.com/minify.html [accessed: 15 May 2020].

Fahnestock J. 2020. jfMagnify plugin. GitHub. https://github.com/fonstok/jfMagnify [accessed: 15 May 2020].

Farkas G. 2017. Applicability of open-source web mapping libraries for building massive Web GIS clients. Journal of Geographical Systems, 19(3), 273–295.

Imperva 2020. Minification. Imperva Blog. http://bit.ly/3aKAY1s [accessed: 15 May 2020].

Król K. 2018. Performance threshold of the interactive raster map presentation – as illustrated with the example of the jQuery Java Script component. Geographic Information Systems Conference and Exhibition GIS ODYSSEY 2018, 321–327.

Król K. 2019. Zoomlens – graphic form of data presentation on a web map, comparison of chosen tool and usage examples. Engineering for Rural Development, 18, 1641–1648.

Król K., Bitner A. 2019. Impact of raster compression on the performance of a map application. Geomatics, Landmanagement and Landscape (GLL), 3, 41–51.

Lotanna N. 2018. 10 Javascript Compression Tools and Libraries for 2019. Bits and Pieces Blog. https://bit.ly/3bEUw6s [accessed: 15 May 2020].

McCall M.K., Dunn C.E. 2012. Geo-information tools for participatory spatial planning: Fulfilling the criteria for 'good' governance?. Geoforum, 43(1), 81–94.

Murugesan S. 2007. Understanding Web 2.0. IT Professional Magazine, 9(4), 34–41.

Neis P., Zielstra D. 2014. Recent developments and future trends in volunteered geographic information research: The case of OpenStreetMap. Future Internet, 6(1), 76–106.

Pataki D. 2017. 14 Tools For Minifying Javascript. Hongkiat.com (HKDC). https://www.hongkiat.com/blog/javascript-minifying-tools/ [accessed: 15 May 2020].

Souders S. 2009. Velocity and the Bottom Line. Radar. http://bit.ly/2U1LmvR [accessed: 15 May 2020].

Stahl M. 2018. A Better Way of Reporting Performance Test Results. StickyMinds Blog. TechWell. http://bit.ly/37A9TvT [accessed: 15 May 2020].

Takalikar V., Joshi P. 2016. Inter-page access metrics for web site structure and performance. In: 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), IEEE, 196–203.

Tsou M-H. 2014. Big data: techniques and technologies in geoinformatics. Annals of GIS, 20(4), 295–296.

YUI Compressor 2020. YUI Compressor. https://yui.github.io/yuicompressor/ [accessed: 15 May 2020].

Zakas N.C. 2010. Better JavaScript Minification. JavaScript, Workflow & Tools. A List Apart Blog. https://alistapart.com/article/better-javascript-minification/ [accessed: 15 May 2020].

Zhu Y., Reddi V.J. 2013. High-performance and energy-efficient mobile web browsing on big/little systems. In: 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), IEEE, 196–203.

Dr inż. Karol Król
Uniwersytet Rolniczy w Krakowie
Katedra Gospodarki Przestrzennej i Architektury Krajobrazu
al. Mickiewicza 24/28, 30-059 Kraków
e-mail: k.krol@onet.com.pl
http://digitalheritage.pl
ORCID: https://orcid.org/0000-0003-0534-8471