http://dx.doi.org/10.15576/GLL/2020.2.41

GLL

Geomatics, Landmanagement and Landscape No. 2 • 2020, 41–49

# THE IMPACT OF CODE MINIFICATION ON MAP APPLICATION PERFORMANCE

Karol Król, Dariusz Zdonek

**Summary**

There are many techniques available today for publishing maps in web browsers. The material is often created using geographical information systems (GIS). The performance, most often understood as the speed of loading the application into a web browser, is the determinant of the viewing experience. The performance of a map application can be improved through such process as minification. The purpose of the study is to measure the impact of minification on the performance of the map component.
Code minification was performed by selected web applications. The performance of two applications, GTmetrix and Dareboost, was tested. Two research questions have been posed: RQ1: *How great a reduction in the size of component files of an application can be achieved with minification?* and RQ2: *How will the minification affect the performance of a web browser map application?* The research has shown that the model applications were performing relatively poor, in particular, on mobile devices. The minification reduced the size of HTML, CSS, and JavaScript files by about 11%, which had a slight impact on application performance (in the employed research design). It has been demonstrated that minification was insufficient for improving significantly the performance of the tested applications. Additional compression of image files is recommended.

**Keywords**

minification • ad-hoc maps • performance • raster • code obfuscation • geo-visualization

## 1. Introduction

New website and web application technologies, evolution of web browsers, increased availability and performance of mobile devices, and greater network bandwidth: all this made it possible to publish maps and cartographic visualisations online [Farkas 2017, Netek et al. 2018]. There is an increase in number of visualisations as well as visualisation software [Lemos et al. 2016, Gotlib 2019].

Today, there are many tools and techniques available for publishing maps in web browsers. The maps are increasingly created with GIS tools (WebGIS, MobileGIS) and APIs [Gotlib 2012]. Users can choose from a growing number of expert or industry-specific websites such as localisation maps, environment and planning portals, travel map apps, and mobile applications [Siejka i Ślusarski 2014]. Most of them are so-called mashups. A web mashup is a web page or web application that combines information and services from multiple sources on the web [Lemos et al. 2016]. Map applications are also used as components embedded into the website or web application structure [Król and Prus 2018]. These components are most often sets of files in the form of hypertext using script languages, predominantly JavaScript [Król 2018].

In this paper, we explore the potential for HTML, CSS, and JavaScript size reduction of selected map applications, focusing on two reduction approaches, script minification and HTML/CSS minification. The purpose of the study is to measure the impact of minification on the performance of a map component. The two main research questions are: RQ1 – *How great a reduction in the size of component files of an application can be achieved with minification?* RQ2 – *How will the minification affect the performance of a web browser map application?* Three model map applications were prepared to address the questions. Two versions of the applications were made, basic and minified. Their performance was then measured. The results were correlated.

## 2. Code minification matters

Formatting rules, such as indents, line breaks, and comments ensuring readability for human users, are not necessary for the execution of web applications on the client side. In order to save data transfer, it is necessary to minify or optimize the script in the entire website [Sakamoto et al. 2015].

There are a few primary methods for optimising the loading time of web applications, including concatenation of resources, use of the HTTP/2.0 protocol, data compression, removal of redundant CSS rules, and minification of JavaScript files [Stępniak and Nowak 2017]. JavaScript is a scripting language that is commonly used to create sophisticated interactive client-side web applications [Lu and Debray 2012].

Minification (also minimisation) is a process of removing all unnecessary characters from the source code of markup languages or interpreted programming languages, without changing their functionality. These unnecessary characters usually include white space characters, newline characters, comments, and block delimiters (Table 1). Many other optimisation techniques can also be employed, such as removing block delimiters, inline functions, using implicit conditionals, and rewriting local variables. HTML, Cascading Style Sheet (CSS), or JavaScript files can be minified. JavaScript and CSS resources may be minified, preserving their behaviour while considerably reducing file size. Minification is usually accomplished by parsing the code and then outputting it in a compressed format. This code is generally unreadable with a naked eye. It is recommended to concatenate files of the same type before minification. Optionally, the process could include replacing variable names with their shorter forms [Stępniak and Nowak 2017]. Minification reduces the size of a source code, making its transmission over a network (such as the Internet) more efficient. Minification is a kind of code transformations that preserves the overall behaviour of a script while making it harder to understand and analyse [Skolka et al. 2019].

**Table 1.** An example of a source code that has been minified (spaces and other unnecessary characters have been removed)

```
(function($){$.fn.minimap=function($mapSource){var x,y,l,t,w,h;var $window=$(window);
var $minimap=this;var minimapWidth=$minimap.width();var minimapHeight=$minimap.
height();var $viewport=$(“<div></div>”).addClass(“minimap-viewport”);$minimap.
append($viewport); synchronize();$window.on(“resize”,synchronize);$mapSource.on(“scroll”,
synchronize);$mapSource.on(“drag”,init);$minimap.on(“mousedown touchstart”,down);function
down(e){var moveEvent, upEvent;var pos=$minimap.position();x=Math.round(pos.
left+l+w/2);y=Math.round(pos.top+t+ h/2);move(e);if(e.type===“touchstart”)
{moveEvent=“touchmove.minimapDown”;upEvent=”touchend”}else{moveEvent=“mousemove.
minimapDown”;upEvent=“mouseup”}$window.on
```

Source: Authors’ own work based on the ‘jquery-minimap’ file

In order to save wasteful data transfer, it is necessary to minify or optimise image files, HTML, CSS code, and scripts on the entire website. High-speed network infrastructure is becoming more and more popular in developed countries. However, crowded and low-speed Wi-Fi environments can still be found on airports, in supermarkets, cafés, international conferences, etc. The network environment of mobile devices requires particularly efficient usage of network bandwidth [Sakamoto et al. 2015].

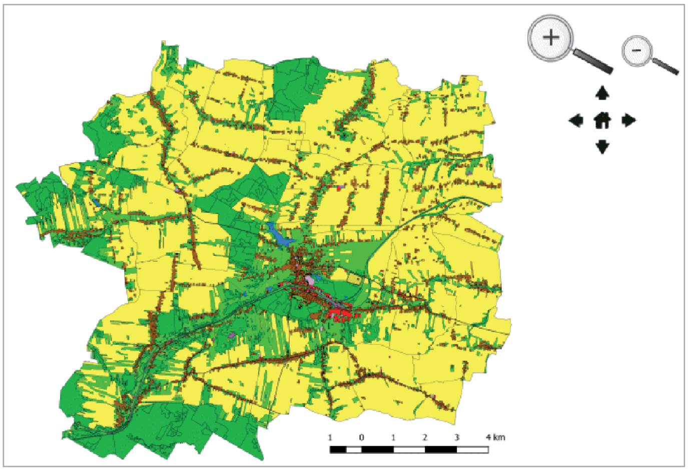
## 3. Materials and methods

Multimedia content, including image files, takes up a lot of the current web traffic, because their file size is relatively large compared to text-based Web content [Król 2019, Król and Bitner 2019]. It was, however, already a common practice to employ encoding or compression processes, whether lossy or lossless, for multimedia content. In contrast, size reduction techniques (e.g. minification, optimisation, or compression) for text-based content, including source code, are often neglected due to the relatively small size of such files. Therefore, the existing text-based Web content was assumed to still have certain reduction potential, especially HTML and CSS files.

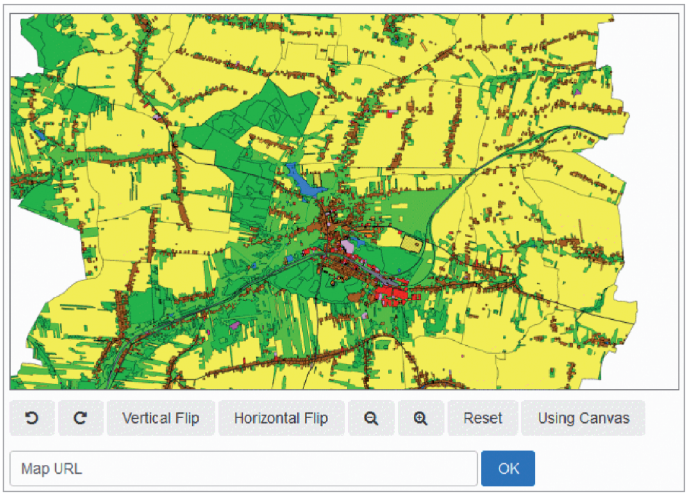
### Model applications

Component files of three model map applications were minimised (Fig. 1). The applications were created as functional extensions or plugins for any hypertext document. The core of the application was an image (raster) file; the applications acted as browsers [Król and Bitner 2019]. All the applications included HTML files (framework of the application), CSS files (appearance, design) JavaScript (functionality, range of interactions), and image files.

a)



b)



Source: Authors' own study

**Fig. 1.** Tested applications (screenshots): a) URL: http://bit.ly/WebApp-1, b) URL: http://bit.ly/WebApp-2

## Automatic minification

There are many tools available for reducing the size of scripts, usually by removing unnecessary whitespaces and comments and renaming symbols (Table 2). This technique is called code compression or minification. Although it makes the code difficult to read, the behaviour remains unaffected [Lu and Debray 2012]. There are also external tools such as grunt-uncss, which automatically remove unnecessary rules from the CSS file based on the static analysis of files.

**Table 2.** Online applications used in the research

| Analysis tool | Website | Application | Measurement unit (if applicable) |
|---|---|---|---|
| GTmetrix | gtmetrix.com | Performance tests | PageSpeed Score, YSlow |
| Dareboost: Website Speed Test (DaM) | dareboost.com | Performance tests | Speed Index |
| HTML minifier | minifycode.com | HTML minification | N/A |
| Online CSS Minifier/ Compressor | cssminifier.com | CSS minification | N/A |
| Minify – JavaScript and CSS minifier | minifier.org | JavaScript minification | N/A |

Source: Authors' own study

HTML and CSS minification was performed with HTML Minifier and Online CSS Minifier/Compressor. JavaScript files were reduced using Minify (Table 2). Next, the performance of regular and minified applications in a browser was measured.

### Automatic performance measurement

Website performance is often equated with the speed of loading the site in a browser window. The website's performance depends mostly on the design solutions adopted, including the techniques and components that were used to create it. Performance tests were informal, using selected online applications, including GTmetrix and Dareboost.

The GTmetrix application measures a website's performance, its loading time in a browser window, and the size of its components. The result of the performance measurement is presented using the PageSpeed Score and YSlow indices. The YSlow attribute is expressed with a synthetic point score, ranging between 0 and 100 points. Dareboost provides a module for testing applications in the mobile and desktop mode. The results of the measurements are presented in the form of a synthetic point score and the Speed Index. The faster the rendering, the lower the index value. Google recommends it does not exceed 1000 units [Król 2018].

## 4. Results

*RQ1 – How great a reduction in the size of component files of an application can be achieved with minification?*

The minification reduced the size of HTML, CSS, and JavaScript file by about 11% (Table 3). It is a relatively small reduction. Note that the model components included only a few files of 136.6 KB. As the application grows more complex, the number of kilobytes (megabytes) saved through minification increases.

**Table 3.** The size of map application component files before and after minification

| Map application | Size of component files* before minification (KB) | Size of component files* after minification (KB) | File size change (%) |
|---|---|---|---|
| Application 1 | 136 | 121 | 11 |
| Application 2 | 6.5 | 5.7 | 12.3 |
| Application 3 | 13.6 | 9.9 | 27.2 |
| Total | 156.1 | 136.6 | 10.8 |

Source: Authors' own study

*RQ2 – How will the minification affect the performance of a web browser map application?*

The model applications turned out to be performing relatively poorly. The GTMetrix PageSpeed Score was about 50% for all the investigated map applications. It is an average result at best [Król 2018]. The conclusion was confirmed by measurements for mobile devices (Table 4). The performance of the tested applications was several times worse here than Google recommendations. The performance of third application on mobile devices was at unacceptable level (the Dareboost Speed Index of 25,741 units, which is much more than the 1000 units recommended by Google). The architecture of the model applications was not suitable for mobile devices.

**Table 4.** Performance results before and after minification

| Map application | Performance measurement before minification | | | Performance measurement after minification | | |
|---|---|---|---|---|---|---|
| | Measuring tool | | | Measuring tool | | |
| | GTMetrix PageSpeed Score (%) | GTmetrix YSlow (%) | Dareboost Speed Index* | GTMetrix PageSpeed Score (%) | GTmetrix YSlow (%) | Dareboost Speed Index* |
| Application 1 | 48 | 77 | 3,281 | 48 | 79 | 3,096 |
| Application 2 | 50 | 94 | 8,870 | 48 | 79 | 8,791 |
| Application 3 | 50 | 89 | 27,047 | 50 | 90 | 25,741 |

* Mobile test, Galaxy S6, London

Source: Authors' own study

The minification generally had a slight impact on the performance of the application (in the employed research design). The performance improved by merely several per cent for most tests. It was mainly due to the nature of the test tools and the architecture of applications based on a relatively large image raster file.

## 5. Discussion

Despite the potential impact minification can have on the performance and security analysis, little is currently known about how real-world websites use such techniques. A better understanding of what kinds of code transformations are applied in the Internet ecosystem could guide future effort towards making the web more efficient and secure. In particular, awareness of the extent of use of minification can help future analyses to focus on relevant problems [Skolka et al. 2019].

Studies have shown that the best increase in performance can be ensured through the reduction of image file size. Data compression involves saving the original information using fewer bites. The size of text resources, such as HTML, CSS, and JavaScript, can usually be reduced by 60 to 80% using gzip compression. In contrast, images require more in-depth consideration. Image compression is lossy. It boils down to choosing the right format, determining the degree of compression and also the colour palette. Much depends on the subjective feeling of whether the image is still clear after applying compression [Stępniak and Nowak 2017]. Król [2019] has demonstrated that it is reasonable to categorise raster files as those intended for online publication (72 dpi) and those intended for printed publication (300 dpi and more). The dynamics of loading raster images into a browser window decreases as the image size increases. Moreover, the effective performance threshold – also referred to as the raster viewing comfort threshold or flexibility threshold – depends on the device used. The architecture of a map application should, therefore, be adapted to the device the application will be most often used on [Król and Bitner 2019].

JavaScript has been used for various attacks on client-side web applications. Skolka et al. [2019] have demonstrated that minification is sometimes used not to increase application's performance but to conceal 'malicious code' (so-called code obfuscation). They have investigated the occurrence of obfuscation and minification in 967,149 scripts on 100,000 websites. They have demonstrated code transformations were employed often and had a substantial impact on 38% of tested scripts. Most of the transformed code was minified, while advanced obfuscation techniques such as encoding parts of the code or fetching all strings from a global array were recorded in less than 1% of the scripts.

According to Sakamoto et al. [2015], about 40% of the total size of JavaScript files used on 500 websites could be reduced by minifying scripts. Stępniak and Nowak [2017] have demonstrated that SPA (single-page application) performance increases the most primarily through JavaScript code minification. Removal of unnecessary CSS rules also improved the performance of SPAs despite minor changes in CSS file sizes.

## 6. Conclusions

Minification may contribute to the improvement of a web application performance. The procedure is more effective in the case of complex applications. Minification results in tangible performance improvement if it is not inhibited by other factors such as image file size, data server configuration, or network bandwidth.

The performance of the model applications depended primarily on the amount of transmitted data. In this case, minification reduced the total size of the applications only slightly. The image file, which had the greatest impact on the performance, was unchanged. Minification alone turned out to be insufficient to significantly improve application performance, which is one of the most relevant results. It is possible that both minification and image file compression are necessary to improve map application performance. If this proves insufficient, the application architecture (design concept) needs to be amended.

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

## References

**Farkas G.** 2017. Applicability of open-source web mapping libraries for building massive Web GIS clients. J. Geogr. Syst., 19(3), 273–295. https://doi.org/10.1007/s10109-017-0248-z

**Gotlib D.** 2012. Mobile maps-modelling of cartographic presentation. Geoinformatica Polonica, 11, 37–47.

**Gotlib D.** 2019. Selected qualities of mobile maps for indoor navigation. Polish Cartographical Review, 51(4), 155–165. https://doi.org/10.2478/pcr-2019-0013

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

**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. https://doi.org/10.22616/ERDev2019.18.N002

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

**Król K., Prus B.** 2018. Application of interactive charts in the evaluation of socio-economic development of regions. The case of Poland. Acta Sci. Pol., ser. Formatio Circumiectus, 17(3), 141–151. https://doi.org/10.15576/ASP.FC/2018.17.3.141

**Lemos A.L., Daniel F., Benatallah B.** 2016. Web service composition: a survey of techniques and tools. ACM Computing Surveys (CSUR), 48(3), 33. https://doi.org/10.1145/2831270

**Lu G., Debray S.** 2012. Automatic simplification of obfuscated JavaScript code: A semantics-based approach. In: 2012 IEEE Sixth International Conference on Software Security and Reliability, 31–40. IEEE. https://doi.org/10.1109/SERE.2012.13

**Netek R., Vozenilek V., Vondrakova A.** 2018. WebGIS 2.0 as approach for flexible web-based map application. In: Proceedings of the International Conference on Geoinformatics and Data Analysis, 1–5. ACM. https://doi.org/10.1145/3220228.3220234

**Sakamoto Y., Matsumoto S., Tokunaga S., Saiki S., Nakamura M.** 2015. Empirical study on effects of script minification and HTTP compression for traffic reduction. In: 2015 Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC), 127–132. IEEE. https://doi.org/10.1109/DINWC.2015.7054230

**Siejka M., Ślusarski M.** 2014. Ocena geoportali internetowych powiatów na wybranych przykładach oraz według ustalonych kryteriów. Infrastruktura i Ekologia Terenów Wiejskich, II/2, 545–555.

**Skolka P., Staicu C-A., Pradel M.** 2019. Anything to Hide? Studying Minified and Obfuscated Code in the Web. In: Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA, eds. J.B. Sartor, Th. D'Hondt, W. De Meuter. ACM, New York, NY, USA. https://doi.org/10.1145/3308558.3313752

**Stępniak W., Nowak Z.** 2017. Performance Analysis of SPA Web Systems. In: Information Systems Architecture and Technology, vol. 521, eds. L. Borzemski, A. Grzech, J. Świątek, Z. Wilimowska, Springer, Cham, 235–247. https://doi.org/10.1007/978-3-319-46583-8_19

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
website: http://homeproject.pl
ORCID: https://orcid.org/0000-0003-0534-8471

Dr inż. Dariusz Zdonek
Politechnika Śląska w Gliwicach
Wydział Organizacji i Zarządzania
Katedra Ekonomii i Informatyki
ul. Roosevelta 26, 41-800 Zabrze
e-mail: dariusz.zdonek@polsl.pl
ORCID: https://orcid.org/0000-0002-6190-964