Energy Efficient Mobile Web via Scripts&Stylesheets Based Transcoding

Hüseyin Ünlü and Yeliz Yeşilada

Middle East Technical University Northern Cyprus Campus
Guzelyurt, Mersin 10, Turkey
unlu.huseyin@metu.edu.tr,yyeliz@metu.edu.tr

Abstract. Mobile devices and also the web have become essential in our daily lives. Although mobile devices nowadays are technically much stronger than the desktops in the past, they still have some limitations regarding the battery size, processing power, and device memory. Additionally, the development of battery capacity is slow and increasing at around 3% per year. These limitations have effects on browsing web pages since they are not designed for mobile use, and it takes more power than necessary on the client side. To save energy and extend the battery life, there are some guidelines for web site programmers. However, most programmers are not aware of these guidelines, and therefore most web sites do not adhere to these guidelines. Important components of modern websites are the scripts that make them dynamic and stylesheets that are used for visual rendering. These two are external components of websites that are shown to effect the downloading time of web pages. This paper first investigates the effect of scripts and stylesheets on the energy consumption of web pages on mobile devices, and then presents techniques that can be used to transcode web pages based on these two components to improve energy consumption and therefore improve battery life. These transcoding techniques focus on saving energy on the client side, without changing the look&feel of the web pages and without adding extra load on the client side or the server.

Keywords: Transcoding, Energy Saving, Browsing, Mobile Web, Web Engineering

Web Sayfalarının Enerji Tasarrufu için Script ve Stil Şablonları Aracılığı ile Dönüştürülmesi

Özet. Mobil cihazların günlük hayatımızdaki yeri çok önemli yerlere ulaştı. Bunun sonucunda mobil platform gereksinimleri hergün artmaktadır. Günümüzdeki mobil cihazlar teknik olarak geçmişteki masaüstü bilgisayarlara göre çok daha güçlü olmasına ve batarya kapasitesi her yıl %3 artmasına rağmen hala batarya kapasitesi, cihaz hafızası ve işlemci kapasitesi konularında bazı kısıtlamalar ile karşı karşıyadır. Bu kısıtlamaların sonucunda web sayfalarına erişim kullanıcı tarafında bataryanın hızlı ve gerektiğinden fazla olarak tükenmesine neden olabilmektedir. Enerji dostu web sayfası için önerilen teknikler olmasına rağmen çoğu programcı bu tekniklerin farkında olmadan web sayfası yaratabiliyor. Günümüz modern web sayfalarının önemli dış bileşenlerinden stil şablonları sayfayı görsel olarak geliştirmek, script ise dinamik bir web sayfası yaratmak için gereklidir. Ancak, bu iki bileşenin sayfanın yüklenme süresine olumsuz yönde etkileri olduğu saptanmıştır. Bu çalışma, ilk olarak script ve stil şablonlarının web sayfalarının enerji tüketimine olan etkilerini ortaya koymayı ve sonrasında ise bazı teknikler önererek web sayfalarını daha az enerji tüketeceği bir şekile dönüştürmeyi ve bunların sonucunda batarya ömrünü geliştirmeyi amaçlamaktadır. Bu dönüştürme tekniklerinin amacı web sayfasının görünüşünü değiştirmeden ve kullanıcı tarafında veya sunucu tarafında ekstra yük oluşturmadan, kullanıcı tarafında enerji verimliliği sağlamaktır.

Anahtar Kelimeler: Dönüştürme, Enerji Tasarrufu, Tarama, Mobil Web, Web Mühendisliği

1 Introduction

In today's world, mobile devices have an important role in our lives. People are tend to use their mobile devices instead of desktop computers, as a result of evolved mobile technology. Thus, the number of mobile devices is increasing. To illustrate, the number of mobile devices is increasing rapidly while the number of desktop computers is decreasing over the years in Turkey [2]. Indeed, the number of connected devices is more than the population and according to a forecast the ratio will be 6.58 by 2020 [23].

Mobile devices are used for different purposes and web access is one of them. Today, people are more likely to use their mobile devices to access web rather than desktop computers. According to a forecast in 2011, it was expected that mobile web would overtake desktop web by 2014 [46]. In 2011, the number of people who access the web through their mobile phones was 900 million and this number was 1.4 billion for desktop Internet users. Their forecast estimates that there will be 1.7 billion mobile web users and 1.65 billion desktop web users. Today, when we compare the number of web access from desktop and mobile, it can be seen that mobile web overtook desktop web [65].

Alongside the increasing web access from mobile, web pages have also become more complex and technologies used also changed. For that purpose, they include more images and videos in high resolution and the size of the web pages is also increasing. Since 2011, the average web page size increased more than 2 MB [25]. By 2017 July, the average web page size increased to 3034 KB while it was 929 KB in 2011. Furthermore, the expectation is that the average size will be more than 4 MB by 2019 [25]. Furthermore, 16% of the web pages are greater than 4 MB today. The number of requests are also increasing by the time. The number of requests increased from 86 to 109 between December, 2011 and December, 2017 [6].

Web pages are evolving everyday, however, mobile device users may not have satisfying user experience as they expect since most of the web pages are not used in ideal conditions and they are not in average case. The reason is that mobile devices have some limitations and these limitations can be listed as screen size, limited bandwidth of the connection, the number of connections, device memory, processing power and the battery [24, 42, 41].

To sum up, mobile devices have some limitations compared to the desktop computers, but they are more likely to be used for browsing. However, the mentioned limitations cause more energy consumption while browsing and this leads battery to drain fast [28]. Web sites composed of different elements such HTML, scripts, stylesheets, images, videos and fonts. In a study, the energy consumption of the web page components are measured [64]. This study shows thatvthe energy consumption of images, JavaScript and CSS is high. Thus, these elements should be optimized to save energy.

There are two main ideas behind reducing the energy consumption of web pages: (1) reducing the number of HTTP requests and (2) reducing the size of the components [71]. There are different guidelines for web developers, however, most of them were not confirmed with respect to energy saving [1, 3, 61, 62, 24, 42, 71]. In this study, our aim is to evaluate the contribution of JavaScript and CSS related guidelines on energy saving.

In this proposed study, our goal is to transcode web pages without modifying their look&feel and without adding extra load to the client or the server side. Our goal is to save energy via optimizing scripts and stylesheets associated to a page. In order to do this, we propose two techniques in the proxy server: (1) concatenating external JavaScript and CSS files to reduce the HTTP request and (2) minification of CSS and JavaScript to reduce the size.

This paper has two goals: (1) to present the detailed literature review on mobile web and energy related studies alongside mentioning the guidelines (see Section 2) and (2) to introduce our work conducted so far to address the gap in the literature (see Section 3).

2 Related Work

In the past, performance was one of the most important issues for the web, however, as mobile web became the primary way to web access, energy efficiency started to be considered more seriously. This section includes a detailed literature review on energy related studies from hardware, network and software levels, transcoding, guidelines for performance improvement and energy saving and detailed information on CSS and JavaScript related guidelines with their implementations.

HTTP introduces the protocol of requests and responses to access web from both mobile and traditional devices and it is used by World Wide Internet since 1990. Briefly, requests and responses are generated by the client and the server in order to provide a communication between them and take the content of a web site or data [27]. When a user requests a web page, HTTP request is generated. Actually, more than one HTTP request is needed to show the content of a typical HTML page as web pages composed of different part such as CSS, scripts, images and videos and they are included as external files in general. For each external element, there should be an HTTP request. However, increasing number of HTTP requests means a waiting time for the client and this duration may be longer under low bandwidth and high latency. Thus, battery of the mobile device drains with the increased loading time. Around 80% of this loading time is used for loading the source and client side processing [24]. The remaining 20% of this period is used for displaying the content. This is same for both desktop and mobile devices. As a result, the increasing number of HTTP requests means higher energy consumption.

Mobile device users may access web via five different ways: native applications, web applications, hybrid applications, web sites and widgets. The summary of advantages and disadvantages of each is given in Table 1. Widgets and native applications uses Internet in their background but they are task specific. Moreover, they are not suitable for cross-platforms. Mobile web applications and hybrid applications are cross platforms but again they are designed to perform specific task. The last category, web sites, can be accessed from all of the browsers without any restriction. Responsive web sites manage the version of the web site. If it is requested from a mobile device, mobile version of the site is downloaded. Otherwise, desktop version is downloaded. Web sites are displayed with browsers. Browsers are only way to access the Internet which is not task specific. Another feature about browsers that makes them a good candidate for our study is that they are not blocked by applications security. Moreover, as they are not task specific, there are lots of web sites that can be included in our evaluation set.

2.1 Hardware Level

Mobile web sits between the hardware, network and software levels. In hardware level, energy related studies focus different areas such as reducing the energy consumption of the processor [31, 45], embedding renewable energy resources into mobile devices [44], analyzing the role of the processor in energy consumption of mobile web browser [72] and reducing the energy consumption of the screen [60]. Intel works on low power consumption with high performance in their processors, such as Intel Core Duo and Intel Core Duo 2 [31, 45]. Indeed, CPU plays an important role on the energy consumption of web browsers and it depends on the network latency. Some studies focus on using renewable energy resources to provide battery life improvements in mobile devices. [44] integrated thermoelectric generator into CPU heat pipe to use the waste heat of the processor and generate electricity. There are also studies that focus on saving energy from screens. [60] is an example for the studies that attempt to optimize the power consumption, focusing on the display. These studies are very important for energy saving in hardware level, however, it is not enough to focus on hardware level energy saving.

Table 1. Comparison of Ways to Access Web [32, 34, 40, 55, 58]

Approach	**Pros**	**Cons**
Native Applications	*High performance *Hardware access *Push notifications *Standalone application	*Compatible with only one platform *High development cost *One needs to maintain multiple applications
Mobile Web Applications	*Low development cost *Cross-platform *Straightforward development *Do not take up any memory or storage on the user's device	*Extra browser layer *Low performance *No hardware access *Internet connection is required to work
Hybrid Applications	*Cross-platform *Moderate development cost *Moderate performance *Hardware access	*Extra Webview layer
Web Sites	*Can be displayed in all platforms *Straightforward development *Low development cost	*No hardware access
Widgets	*Standalone applications	*Compatible with only one platform

2.2 Network Level

There are different studies about energy consumption in network level such as analyzing the energy consumption of the mobile network [22], the impact of different communication technologies on mobile devices [50, 68, 53, 52, 11], using localization [14] and caching [57] to improve energy saving in mobile networks. Network has an impact on energy consumption of mobile devices however the energy consumption of the network is very high compared to consumption of mobile devices. In [22], they show the energy consumption of mobile of a customer for a terminal and for mobile network. According to the their results, energy consumption of a customer for a terminal is 0.83 Wh/day while it is 120 Wh/day for the mobile network. When the energy consumption of the terminal is compared with mobile network, it may be negligible. However, they state that mobile devices are energy starving because of their battery limitations. Mobile device entities such as data communication, cellular link services, display, screen update, mobile tv and CPU may have different energy consumptions under different network technologies [68, 53, 52, 11]. In [14], they developed a system, which uses localization techniques but not only limited with Global Positioning System (GPS) and they achieved to increase the battery life from 9 hours up to 22 hours [14]. In [57], they propose a cooperative web caching system for ad-hoc networks, which enables mobile terminals to share web pages to decrease the energy consumption. Network is also an important level for energy saving but it is not enough to focus on network level energy saving.

2.3 Software Level

Software is another level which is related with the energy consumption and there are different studies that handle the software in order to reduce the energy consumption. These studies handle the effect of the operating system and the programming language [49, 48], refactoring [54, 51], code obfuscation [17, 56] and the effect of the networking protocol [4]. There are also some studies related with the energy consumption of web sites [64] and the effect of different JavaScript libraries on energy consumption [43]. Energy efficiency is also worked by the browsers [18, 35, 66, 67, 16]. [49] compares the energy consumption of two different operating system: Android and Angstrom Linux, with two different sorting algorithms. In another study [48], they compared Java, JavaScript and C++ in terms of energy consumption and performance in Android applications.

Code refactoring and code obfuscation also have an impact of energy consumption. In [54], they applied good programming practices and code refactoring to reduce battery consumption of scientific mobile applications on Android devices. On the other hand, refactoring may increase the energy consumption. [51] argues that god class refactoring has harmful effect on power consumption. Refactoring god classes derives excessive message traffic and this increases a system's power consumption. In [17], ten Android applications were analyzed with the impact of code level obfuscation executing number of scenarios and they achieved energy saving. There are also some studies about protocol. SPDY is an experimental application layer protocol developed by Google as a part of the "Let's make the web faster" initiative. Although the main aim of this protocol is not energy efficiency, it is a step for energy saving with reducing the latency. The comparison of HTTP and SPDY shows that SPDY achieves 64% reduction in page load duration.

Energy efficiency is also an important issue for the web browsers. Browsers provide users to choose energy saving mode or plug-ins, this feature is offered by most of the browsers with different techniques. Browser companies are doing tests to measure the energy efficiency of each other in every update [18, 35, 66, 67]. Apart from the sector, there are some scientific studies that aim to improve the energy efficiency of the browsers. [16] worked on reducing the energy consumption to load web pages on smartphones presenting three different techniques.

Alternative to aim of these studies, we also have transcoding which is the process of transforming web pages into alternative forms in order to have improvements for different purposes. To illustrate, it can be applied to provide web accessibility for disabled people, increase the performance or improve the usability of a mobile device. Transcoding can be done to improve the energy efficiency or loading time of the web page. There are three types of transcoding which are client-side, server-side and proxy-side [41]. The difference between the types is the location of the transcoding server. In client-side transcoding, modification is done in the device. After client receives the content, transcoding is applied on the device [41]. However, since the transcoding is done on by device itself, the device should be powerful. If the limitations of mobile devices considered, it can be said that client-side transcoding is not useful enough [69]. In server-side transcoding, the transformation is done on the server side and client receives the modified content from the server. This process is invisible to the client since all the process is done at the server-side [10]. However, it should be done by the owner of the web site. The third type of transcoding is proxy-side transcoding. Proxy is a specialized server that sits between the client and the web server. The transcoding process is done in the proxy, not in the client-side or server-side [41]. It is transparent to the user after the address of transcoding server is set by the client. When client requests a web page, the client sends the request to the proxy. Then, proxy sends a request to the server and the content is sent to the proxy. After transcoding, the web page content is sent to the client [69]. Proxy-side transcoding does not put extra work on client or server side but the connection speed between the client, proxy and server could be the drawback [69]. There are different purposes of transcoding such as text magnification [10, 7], color scheme changes [10, 33, 7], alternative text insertion [10, 33, 7],page re-arrangement [41, 10, 63, 59], simplification [10, 19, 70], summarization [10, 41, 9, 38, 39], image consolidation [21, 24, 7, 26, 36], responsive image [30, 24, 8, 42, 7], data compression [12, 24, 7, 20, 7], reducing the number of redirects [71, 1, 36], expiration header [24, 47], dynamic adaptation [29]. However, energy efficiency is validated in only few of them.

In the literature, there are different guidelines (both from scientific papers and sector) for performance improvement and energy saving in web sites [1, 3, 61, 62, 24, 42, 71]. Although most of these guidelines do not include any suggestion about energy efficiency, some of the suggestions may be also applicable for energy saving. There are two main ideas behind reducing the energy consumption of a website: (1) to minimize HTTP requests and (2) to reduce the size of the components [71, 24]. When we look at the guidelines for improving the speed of access to web sites, we can see that these two principles are behind them all.

These guidelines are related but not limited with caching, images, number of requests, redirects, scripts and stylesheets. From these guidelines we can see that there are lots of recommended techniques on CSS and JavaScript which are the core elements of dynamic web sites and their energy consumption is high. Very few of these recommendations are validated and some of them are implemented as tools that work on different sides such as client, proxy or server. Table 2 summarizes the CSS and JavaScript related guidelines and its related work and implementations. However, most of these techniques have not been evaluated in terms of energy saving or performance. Some techniques have some implementations so that they can be used while developing a web site.

Table 2 shows the recommended techniques on CSS and JavaScript but this table only mentions the techniques if they have implemented tools or validated research. From this table, we can see that most of the implementations work on the server side which means that they need to be used by the developers. Only one of them works on the client side but it puts extra load on the client's device. Moreover, there is no information on modifying the look&feel of the page in most of the implementations. Only three of them achieves battery life improvement

Table 2. Techniques on CSS & JavaScript

Technique	**Implementation**	**Ref.**	**Location**			**Modify the Look & Feel**	**Battery Life Improvement**
			Server	Proxy	Client		
Avoid CSS @imports	Google PageSpeed Module	[7]	✓			?	?
Avoid document.write	Google Lighthouse Tool	[7]	✓			?	?
Combine images with CSS sprites	Google PageSpeed Module	[7]	✓			?	?
	Twes+	[36]		✓		X	✓
Combine external CSS and Javascript	Google Apache Module	[71]	✓			?	?
	Google Closure Compiler	[5]	✓			?	?
	Google PageSpeed Module	[7]	✓			?	?
	Mobify Jazzcat	[7]	✓		✓	?	?
	IBM Worklight Studio	[7]	✓			?	?
	Grunt contrib-concat	[7]	✓			?	?
Minify JavaScript and CSS	YUICompressor	[7]	✓			?	?
	JSCompress	[7]	✓			?	?
	JSMin	[7]	✓			?	?
	Minify	[7]	✓			?	?
	CSSNano	[7]	✓			?	?
	csso	[7]	✓			?	?
	Microsoft Ajax Minifier	[7]	✓			?	?
Minification and concatenation	JS&CSS Script Optimizer	[37]	✓			?	?
	Merge+Minify+Refresh	[7]	✓			?	?
	Dependency Minification	[7]	✓			?	?
	Minqueue	[7]	✓			?	?
	Combine and Minify	[7]	✓			?	?
	Granule	[7]	✓			?	?
	Jawr	[7]	✓			?	?
	The Asset Pipeline	[7]	✓			?	?
	Codekit	[?]	✓			?	?
	Prepros	[7]	✓			?	?
Remove unused CSS	Firefox Dust-me selectors extension	[7]	✓			?	?
	Chrome Developer Tools	[13]	✓			?	?
	Gtmetrix	[7]	✓			?	?
	unused-css.com	[7]	✓			?	?
	mincss	[15]	✓			?	?
	uncss	[7]	✓			?	?
	CSS remove and combine	[7]	✓			?	?
	Who killed my battery?	[64]	✓			?	✓
Put scripts at bottom	JS&CSS Script Optimizer	[7]	✓			?	?
Remove duplicate scripts	uniq.js	[7]	✓			?	?
Write efficient JavaScript	Who killed my battery?	[64]	✓			?	✓

and the other techniques are not validated in energy aspect. Thus, analyzing the role of these techniques on energy saving would contribute to the literature.

Twes+ is the only example which is a proxy side transcoding system that achieves battery life improvement without modifying the look&feel of the web page [36]. It has two services: (1) to combine images with CSS sprites and (2) to reduce the number of redirects. In this study, Their results show that redirect service can success 4.6% cumulative processor energy reduction and image transcoding can success 7% reduction in cumulative processor energy, which is equal to between a 40 to 60 minutes saving in a battery of a mobile device.

The number of studies that attempt to reduce the energy consumption of web sites is very low. Twes+ is an example that attempts to reduce energy consumption of web sites based on proxy side transcoding but it did not evaluate JavaScript and CSS related techniques [36]. Thus, our proposed solution is to transcode web sites on proxy server based on two techniques on JavaScript and CSS: (1) concatenating external JavaScript and CSS files and (2) minification. The next section describes our proposed approach to reduce the energy consumption without modifying the look&feel on the web page and without putting extra load on the client or server side.

3 Methodology and Architecture

In this study, our goal is to reduce the energy consumption of the web pages and achieve battery life improvement on mobile devices without modifying the look&feel of the web pages and without putting extra load on the server or client side. Our focus is to implement (1) concatenating external JavaScript and CSS files to reduce the number of HTTP requests and (2) minification to reduce the size of the components. These techniques are explained as follow:

Combine external CSS and JavaScript

Each external resource means an extra HTTP request. It is suggested to make JavaScript and CSS external unless they are used in home pages that have few page view per session [1]. However, more external files mean more HTTP requests. As a solution, external JavaScript and CSS files should be concatenated [24, 71, 3]. In the development stage, multiple CSS and JavaScript files makes the process easier since it is better to follow [24]. However, an ideal web page should have one JavaScript file and one CSS file as external [71].

Traditionally, concatenation is done at build time. However, there are some services to make this process at runtime using content delivery network (CDN). Google Apache Module is a concatenation service which requires server support so you need to be owner. This module concatenates files dynamically at runtime. It uses special URL format to download multiple files using a single request.

http://www.example.com/assets/js/main.js

http://www.example.com/assets/js/utils.js

http://www.example.com/assets/js/lang.js

mod_concat in Apache Module combines these requests into one URL as follow:

http://www.example.com/assets/js??main.js,utils.js,lang.js

This URL concatenates main.js, utils.js, and lang.js into a single response in the order specified. In the combined URL, double question marks indicates to the server that this URL should use the concatenation behavior [71].

Minify CSS and JavaScript

Minification refers to removing unnecessary characters, such as white spaces and comments, from the code in order to reduce the size of a file. Therefore, it will improve the load times. According to a survey, in ten top U.S. web sites, 21% size reduction was achieved by minification [1]. Typically, minification achieves 20% size reduction for JavaScript [61]. Apart from reducing the bandwidth consumption and latency, minification makes difference for cacheable object by reducing the size of the object to be cached [24]. Gzip compression does not help in this regard since objects are cached after decompressed.

In our system, the location of transcoding is on proxy so that it does not put extra work on the server side and client side. The client, proxy, and the server are connected to each other locally by using a router. When a client requests a web page from the server, it goes through proxy, then to the server. When the server send the response, it goes to the proxy first and then transcoded here. After transcoding, the content is delivered to the client. In each step, communication is done by the router.

Our system is constructed as a proxy and in particular Squid Caching Proxy is used. It is an open source caching proxy and constructed on Linux OS in our system. Squid Caching Proxy is used an ICAP client and and GreasySpoon [36] is used as ICAP server, which is one of the suitable ICAP servers for Squid Proxy. Our system works over the GreasySpoon to modify the content to save energy for mobile devices.

4 Conclusion and Future Work

Energy efficiency of web sites is important when we consider the limitations of mobile devices, especially the battery size. Furthermore, web access from mobile devices is increasing. There are different studies that handle energy consumption in hardware, software and network level. Web sits between these three levels. There are different guidelines for web developers to design efficient web sites. Although most of these guidelines are not designed to improve energy efficiency, the main idea behind increasing the performance and energy efficiency may be the same: decreasing the number of HTTP requests and reducing the size of components. However, most of the developers are not aware of these guidelines. Indeed, some of the best web sites include some performance pitfalls. Thus, the transcoding should not be in the server side so that it can be applicable for all of the web sites. When the limitations of mobile devices are considered, it can be seen that client side transcoding is not a good candidate. The number of studies that attempts to reduce the energy consumption of web sites is very low. Thus, our proposed solution is to transcode web sites on proxy server based on two techniques on JavaScript and CSS: (1) concatenating external JavaScript and CSS files to reduce the number of requests and (2) minification to reduce the size of the contents.

Our system concatenates external JavaScript and CSS files into one JavaScript and one CSS file and minifies all the content on the proxy server. In fact, we will focus on demonstrating that these guidelines do improve energy saving or not. For the evaluation, we have two research questions to evaluate our system: *"Does the system allow saving energy by concatenating external files to reduce the number of HTTP connections?"* and *"Does the system allow saving energy by minifying scripts and stylesheets to reduce the size of the file?"*. The measurements in desktop and mobile are done to answer these questions. Finally, we are also working on modeling the sustainability of the proposed approach to show that we do in fact contribute to sustainability by saving energy.

References

1. Best practices for speeding up your web site, `https://developer.yahoo.com/performance/rules.html`, february, 2017

2. Information and communication technology (ict) usage in households and by individuals, http://www.turkstat.gov.tr/PreTablo.do?alt_id=1028
3. Make the web faster — google developers, https://developers.google.com/speed/
4. Spdy: An experimental protocol for a faster web, https://www.chromium.org/spdy/spdy-whitepaper
5. Sprite images, https://www.modpagespeed.com/doc/filter-image-sprite
6. Trends, http://httparchive.org
7. Url references, https://www.dropbox.com/s/r7jofqc5kg00dup/URL%20References%20-%20UYMS2018.pdf?dl=0
8. How apple.com will serve retina images to new ipads (Jul 2016), http://blog.cloudfour.com/how-apple-com-will-serve-retina-images-to-new-ipads
9. Ahmadi, H., Kong, J.: Efficient web browsing on small screens. In: Proceedings of the working conference on Advanced visual interfaces. pp. 23–30. ACM (2008)
10. Asakawa, C., Takagi, H.: Transcoding. In: Web Accessibility, pp. 231–260. Springer (2008)
11. Balasubramanian, N., Balasubramanian, A., Venkataramani, A.: Energy consumption in mobile phones: A measurement study and implications for network applications. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. pp. 280–293. IMC '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1644893.1644927
12. Barr, K.C., Asanović, K.: Energy-aware lossless data compression. ACM Transactions on Computer Systems (TOCS) 24(3), 250–291 (2006)
13. Basques, K.: What's new in devtools (chrome 59) — web — google developers, https://developers.google.com/web/updates/2017/04/devtools-release-notes
14. Ben Abdesslem, F., Phillips, A., Henderson, T.: Less is more: energy-efficient mobile sensing with senseless. In: Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds. pp. 61–62. ACM (2009)
15. Bengtsson, P.: peterbe/mincss (Nov 2017), https://github.com/peterbe/mincss
16. Bui, D.H., Liu, Y., Kim, H., Shin, I., Zhao, F.: Rethinking energy-performance trade-off in mobile web page loading. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking. pp. 14–26. MobiCom '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2789168.2790103
17. Bunse, C.: On the impact of code obfuscation to software energy consumption. In: From Science to Society, pp. 239–249. Springer (2018)
18. Cameron, C.: Chrome: Faster and more battery-friendly (Sep 2016), https://blog.google/products/chrome/chrome-faster-and-more-battery-friendly/
19. Chen, J., Zhou, B., Shi, J., Zhang, H., Fengwu, Q.: Function-based object model towards website adaptation. In: Proceedings of the 10th international conference on World Wide Web. pp. 587–596. ACM (2001)
20. Chi, C.H., Deng, J., Lim, Y.H.: Compression proxy server: Design and implementation. In: USENIX Symposium on Internet Technologies and Systems (1999)
21. Dhiraj: Image sprites how to merge multiple images, and how to split them (Sep 2013), https://dhirajkumarsingh.wordpress.com/2012/06/24/image-sprites-how-to-merge-multiple-images-and-how-to-split-them/
22. Etoh, M., Ohya, T., Nakayama, Y.: Energy consumption issues on mobile network systems. In: Applications and the Internet, 2008. SAINT 2008. International Symposium on. pp. 365–368. IEEE (2008)

23. Evans, D.: The internet of things: How the next evolution of the internet is changing everything. CISCO white paper 1(2011), 1–11 (2011)
24. Everts, T.: Rules for mobile performance optimization. Commun. ACM 56(8), 52–59 (Aug 2013), `http://doi.acm.org/10.1145/2492007.2492024`
25. Everts, T.: The average web page is 3mb. how much should we care? (Aug 2017), `https://speedcurve.com/blog/web-performance-page-bloat/`
26. Fainberg, L., Ehrlich, O., Shai, G., Gadish, O., Amitay, D., Berger, O.: Systems and methods for acceleration and optimization of web pages access by changing the order of resource loading (Feb 3 2011), uS Patent App. 12/848,559
27. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol–http/1.1. Tech. rep. (1999)
28. Firtman, M.: Programming the mobile web. " O'Reilly Media, Inc." (2010)
29. Flinn, J., Satyanarayanan, M.: Energy-aware adaptation for mobile applications. SIGOPS Oper. Syst. Rev. 33(5), 48–63 (Dec 1999), `http://doi.acm.org/10.1145/319344.319155`
30. Frain, B.: Responsive web design with HTML5 and CSS3. Packt Publishing Ltd (2012)
31. Gochman, S., Mendelson, A., Naveh, A., Rotem, E.: Introduction to intel core duo processor architecture. Intel Technology Journal 10(2) (2006)
32. Huy, N.P., vanThanh, D.: Evaluation of mobile app paradigms. In: Proceedings of the 10th International Conference on Advances in Mobile Computing &#38; Multimedia. pp. 25–30. MoMM '12, ACM, New York, NY, USA (2012), `http://doi.acm.org/10.1145/2428955.2428968`
33. Iaccarino, G., Malandrino, D., Scarano, V.: Personalizable edge services for web accessibility. In: Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A): Building the mobile web: rediscovering accessibility? pp. 23–32. ACM (2006)
34. Jobe, W.: Native apps vs. mobile web apps. International Journal of Interactive Mobile Technologies 7(4) (2013)
35. Kamierczak, B.: Why we challenge microsoft's battery test (Jun 2016), `https://www.opera.com/blogs/desktop/2016/06/over-the-edge/`
36. Koksal, E.: Twisting web pages for saving energy. In: International Wireless Communications Expo (IWCE) (2017)
37. Kotelnytskyi, Y.: Js & css script optimizer, `https://wordpress.org/plugins/js-css-script-optimizer/`
38. Lai, P.P.: Efficient and effective information finding on small screen devices. In: Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility. p. 4. ACM (2013)
39. Lam, H., Baudisch, P.: Summary thumbnails: readable overviews for small screen web browsers. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 681–690. ACM (2005)
40. Looper, J.: What is a webview? (Nov 2015), `https://developer.telerik.com/featured/what-is-a-webview/`
41. Lose, T., Thinyane, M.: A transcoding proxy server for mobile web browsing (09 2011)
42. Matsudaira, K.: Making the mobile web faster. Commun. ACM 56(3), 56–61 (Mar 2013), `http://doi.acm.org/10.1145/2428556.2428572`
43. Miettinen, A.P., Nurminen, J.K.: Analysis of the Energy Consumption of JavaScript Based Mobile Web Applications, pp. 124–135. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), `https://doi.org/10.1007/978-3-642-16644-0_12`

44. Muhtaroglu, A., Yokochi, A., Von Jouanne, A.: Integration of thermoelectrics and photovoltaics as auxiliary power sources in mobile computing applications. Journal of Power Sources 177(1), 239–246 (2008)
45. Naveh, A., Rotem, E., Mendelson, A., Gochman, S., Chabukswar, R., Krishnan, K., Kumar, A.: Power and thermal management in the intel core duo processor. Intel Technology Journal 10(2) (2006)
46. Norris, C.A., Soloway, E.: Learning and schooling in the age of mobilism. Educational Technology 51(6), 3 (2011)
47. Nottingham, M.: Caching tutorial (May 2013), `https://www.mnot.net/cache_docs/`
48. Oliveira, W., Oliveira, R., Castor, F.: A study on the energy consumption of android app development approaches. In: Proceedings of the 14th International Conference on Mining Software Repositories. pp. 42–52. MSR '17, IEEE Press, Piscataway, NJ, USA (2017), `https://doi.org/10.1109/MSR.2017.66`
49. Paul, K., Kundu, T.K.: Android on mobile devices: An energy perspective. In: Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. pp. 2421–2426. IEEE (2010)
50. Pentikousis, K.: In search of energy-efficient mobile networking. IEEE Communications Magazine 48(1) (2010)
51. Pérez-Castillo, R., Piattini, M.: Analyzing the harmful effect of god class refactoring on power consumption. IEEE software 31(3), 48–54 (2014)
52. Perrucci, G.P., Fitzek, F.H.P., Sasso, G., Kellerer, W., Widmer, J.: On the impact of 2g and 3g network usage for mobile phones' battery life. In: 2009 European Wireless Conference. pp. 255–259 (May 2009)
53. Perrucci, G.P., Fitzek, F.H.P., Widmer, J.: Survey on energy consumption entities on the smartphone platform. In: 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring). pp. 1–6 (May 2011)
54. Rodriguez, A., Mateos, C., Zunino, A.: Improving scientific application execution on android mobile devices via code refactorings. Software: Practice and Experience 47(5), 763–796 (2017)
55. Saccomani, P.: Native, web or hybrid apps? whats the difference? (Jan 2018), `https://www.mobiloud.com/blog/native-web-or-hybrid-apps/`
56. Sahin, C., Wan, M., Tornquist, P., McKenna, R., Pearson, Z., Halfond, W.G., Clause, J.: How does code obfuscation impact energy usage? Journal of Software: Evolution and Process 28(7), 565–588 (2016)
57. Sailhan, F., Issarny, V.: Energy-aware web caching for mobile terminals. In: Proceedings 22nd International Conference on Distributed Computing Systems Workshops. pp. 820–825 (2002)
58. Sin, D., Lawson, E., Kannoorpatti, K.: Mobile web apps-the non-programmer's alternative to native applications. In: Human System Interactions (HSI), 2012 5th International Conference on. pp. 8–15. IEEE (2012)
59. Song, R., Liu, H., Wen, J.R., Ma, W.Y.: Learning block importance models for web pages. In: Proceedings of the 13th international conference on World Wide Web. pp. 203–211. ACM (2004)
60. Sorber, J., Banerjee, N., Corner, M.D., Rollins, S.: Turducken: Hierarchical power management for mobile devices. In: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services. pp. 261–274. MobiSys '05, ACM, New York, NY, USA (2005), `http://doi.acm.org/10.1145/1067170.1067198`
61. Souders, S.: High-performance web sites. Communications of the ACM 51(12), 36–41 (2008)

62. Souders, S.: Even faster web sites: performance best practices for web developers. " O'Reilly Media, Inc." (2009)
63. Takagi, H., Asakawa, C., Fukuda, K., Maeda, J.: Site-wide annotation: reconstructing existing pages to be accessible. In: Proceedings of the fifth international ACM conference on Assistive technologies. pp. 81–88. ACM (2002)
64. Thiagarajan, N., Aggarwal, G., Nicoara, A., Boneh, D., Singh, J.P.: Who killed my battery?: Analyzing mobile browser energy consumption. In: Proceedings of the 21st International Conference on World Wide Web. pp. 41–50. WWW '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2187836.2187843
65. Titcomb, J.: Mobile web usage overtakes desktop for first time (Nov 2016), http://www.telegraph.co.uk/technology/2016/11/01/mobile-web-usage-overtakes-desktop-for-first-time/
66. Weber, J.: Get more out of your battery with microsoft edge (Sep 2016), https://blogs.windows.com/windowsexperience/2016/06/20/more-battery-with-edge/#MCyQIXD4topvxSdz.97
67. Weber, J.: Microsoft edge now gets even more out of your battery (Sep 2016), https://blogs.windows.com/windowsexperience/2016/09/15/edge-battery-anniversary-update/#C56OkPtxC3uPdrAy.97
68. Xiao, Y., Kalyanaraman, R.S., Yla-Jaaski, A.: Energy consumption of mobile youtube: Quantitative measurement and analysis. In: 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies. pp. 61–69 (Sept 2008)
69. Yesilada, Y., Harper, S., Eraslan, S.: Experiential transcoding: an eyetracking approach. In: Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility. p. 30. ACM (2013)
70. Yin, X., Lee, W.S.: Using link analysis to improve layout on mobile devices. In: Proceedings of the 13th international conference on World Wide Web. pp. 338–344. ACM (2004)
71. Zakas, N.C.: The evolution of web development for mobile devices. Queue 11(2), 30:30–30:39 (Feb 2013), http://doi.acm.org/10.1145/2436696.2441756
72. Zhu, Y., Halpern, M., Reddi, V.J.: The role of the cpu in energy-efficient mobile web browsing. IEEE Micro 35(1), 26–33 (Jan 2015)