# Automatic Discovery of Performance and Energy Pitfalls in HTML and CSS

Adrian Sampson, University of Washington
Călin Caşcaval, Qualcomm
Luis Ceze, University of Washington
Pablo Montesinos, Qualcomm
Dario Suarez Gracia, Universidad de Zaragoza

***Abstract*—WebChar is a tool for analyzing browsers holistically to discover properties of HTML and CSS that lead to poor performance and high energy consumption. It analyzes a large collection of Web pages to mine a model for their performance based on static attributes of the content. An evaluation on two platforms, a netbook and a smartphone, demonstrates that WebChar can yield actionable conclusions for both content developers and browser implementors.**

## I. Introduction

Web browsing is an increasingly important part of the end-user computing experience. But the performance and energy consumption of Web technologies limits their growth in the mobile space where these factors are most crucial. Tools for understanding browsers' energy and performance characteristics are essential to creators of efficient Web content and to developers of mobile browsers.

Some studies have measured specific parts of the browser that are known to be bottlenecks, but Web technologies are complex, browsers change rapidly, and real Web content often uses the technologies in unexpected ways [1], [2]. Moreover, while JavaScript is amenable to traditional tools like performance profilers, fewer tools exist to analyze the declarative languages HTML and CSS—even though they represent a large portion of browsers' execution time [3]. A complete understanding of performance and energy on the Web requires empirical analysis of specific HTML and CSS implementations in the context of real-world usage. Web optimization tools should give broad, up-to-date answers to these central questions: What makes some Web pages slower than others? Why do some sites seem to guzzle battery life while others sip it?

This paper describes WebChar (for *Web characterization*), a model-mining system for analyzing browsers holistically. WebChar uses a large body of HTML and CSS content to build a model relating static page features to browser performance and energy consumption. It then mines this model to discover detailed, nonintuitive potential performance and energy problems reflective of common usage. Web content developers can use WebChar results as recommendations to avoid certain content-authoring pitfalls; to browser developers, WebChar is a tool for discovering new high-level optimization opportunities.

A prototype end-to-end WebChar implementation, along with an extended technical report, is available online at:
http://sampa.cs.washington.edu/sampa/WebChar

## II. Approach

The WebChar system consists of two main components. A data collection module takes snapshots of a large set of popular Web sites and measures the page load time and energy for each site. Then, the analysis step builds a model that predicts browser performance (or energy) based on page features. WebChar mines this model to produce a ranked list of likely expensive features.

### A. Data Collection

The data collection tool downloads raw data from popular Web sites for analysis and measures the performance and power consumption of Web browsers while loading each page.

To obtain this input data, it would not suffice to download individual HTML documents from popular Web sites; instead, we need full *snapshots* of Web pages along with all linked linked content including embedded images and CSS stylesheets. A page snapshot includes all information necessary to replicate the experience of loading the page.

Using snapshots, the system collects performance or energy metrics for a particular browser, OS, and hardware setup. The measurement component consists of an HTTP server capable of replaying snapshots while simultaneously collecting timing and power data. From the browser's perspective, the replay server is indistinguishable from a "real" remote server; the HTTP responses are identical to those of the original host.

### B. Analysis

Using the collected page data and performance measurements, WebChar's analysis component first summarizes the page data into a set of numerical *features*. It then correlates these feature values with the power and performance measurements to produce recommendations.

Features are Web page metrics that could potentially correlate with performance or energy usage. In other words, each feature is a candidate for identification as an "expensive" aspect of Web content. WebChar's features consist of metrics of each page's HTML and CSS abstract syntax trees, including tag, property, and selector frequencies. In all, the present implementation calculates 253 features per page.

The correlation step learns the relationship between a page's feature vector and its performance or energy consumption. Each page's feature vector and performance/energy data together constitute an example that WebChar uses to train its model. Any function estimation technique may be used to

produce this model; WebChar uses support vector regression. Once the model is trained, we apply an optimization heuristic, simulated annealing, to find a maximum of the trained function—a feature vector that leads to maximally poor performance or high energy consumption. The resulting optimized feature vector represents a set of feature values that, according to the trained model, result in bad browser behavior.

## III. Results

To evaluate WebChar, we analyzed data from 200 popular Web sites. We measured browsers on two systems: a mobile phone running Android 2.3.2 and an Atom-based netbook running Chromium 12.0.742.100. The phone's wireless communication hardware and display backlight were disabled to avoid measuring their power draw.

Using WebChar's output rankings, we distilled eight testable hypotheses regarding the performance and power consumption of HTML and CSS features. Of these, two hypotheses (the cost of images and CSS descendant selectors) reflected previously known performance pitfalls. We tested the remaining six hypotheses using microbenchmarks designed to exercise the feature that was hypothesized to be expensive. Of these, five effects were shown to be statistically significant and represent recommendations to browser and content authors:

- Laying out tables can be expensive on the Chrome desktop browser. Content developers should avoid placing content into tables where not strictly necessary.
- CSS opacity controls carry a significant performance impact, especially on the mobile browser we measured: translucent elements rendered 28% more slowly than opaque elements in our mobile phone tests. Web developers should avoid opacity effects; meanwhile, mobile browser developers should investigate using hardware-accelerated compositing for this feature.
- Across both platforms we measured, "floating" layout is expensive. Content developers should avoid it where possible; however, since modern Web page layouts frequently depend on these elements for their structure, browser implementors should optimize for this common pattern.
- Background fills cost significant energy on Android. Even when pages do not seem to load slowly on that platform, they may spend a large amount of energy drawing backgrounds. Mobile content developers should avoid using unnecessary background fills.
- The Android browser exhibits a performance and energy penalty when using <SPAN> HTML elements. The developers should investigate the element's unnecessary inefficiency as its presence does not affect the page's appearance.

Several of these findings are surprising and nonintuitive. The energy (but not performance) cost of element backgrounds, for instance, represents an unexpected discrepancy between performance and energy that can be exploited to conserve battery life on mobile devices. The cost of <SPAN> elements in the Android browser is also counterintuitive and is likely the result of a performance bug in the platform's implementation. These unexpected results are where WebChar is most useful: the technique can identify costly factors without relying on human intuition.

## IV. Related Work

Existing performance- and energy-related resources for content developers generally consist of best practices provided by experts [4]–[6]. Tools like WebChar can make such best-practice advice more complete and allow it to grow and change along with the Web. Case studies [7] can also provide some insight into Web application workloads but do not scale to capture large and diverse portions of the Web.

Some work has focused on new implementation techniques for Web technologies [3], [8]–[11]. Even improved Web browsers, however, will likely exhibit hard-to-predict performance and energy pitfalls.

WebChar's approach is similar to analyses for distributed systems that use data mining techniques to debug failures [12] and pathologies [13]. WebChar treats browsers as complex black-box systems and, like the prior work, analyzes externally observable metrics to infer internal problems.

## V. Conclusion

WebChar is a new technique for developing understanding of performance and energy in Web browsers, filling a crucial role in the improvement of the browsing experience on resource-constrained mobile devices. WebChar eliminates the slow process of manually identifying bottlenecks from among the thousands of features that browsers implement. As the Web continues to grow as a platform and as browser implementations continue to evolve, automated analyses like WebChar will be necessary to help identify new pitfalls when they appear.

## References


[1] G. Richards, S. Lebresne, B. Burg, and J. Vitek, "An analysis of the dynamic behavior of JavaScript programs," in *PLDI*, 2010.

[2] P. Ratanaworabhan, B. Livshits, and B. G. Zorn, "JSMeter: comparing the behavior of JavaScript benchmarks with real web applications," in *WebApps*, 2010.

[3] L. A. Meyerovich and R. Bodik, "Fast and parallel webpage layout," in *WWW*, 2010.

[4] S. Souders, *High performance web sites: essential knowledge for frontend engineers: 14 steps to faster-loading web sites*. O'Reilly, 2007.

[5] Yahoo. Best practices for speeding up your web site. http://developer.yahoo.com/performance/rules.html.

[6] Google. Web performance best practices. http://code.google.com/speed/page-speed/docs/rules_intro.html.

[7] S. Xu, B. Huang, J. Ding, and J. Dai, "Browser workload characterization for an Ajax-based commercial online service," in *IISWC*, 2009.

[8] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd, "Power browser: efficient web browsing for PDAs," in *CHI*, 2000.

[9] J. Mickens, J. Elson, J. Howell, and J. Lorch, "Crom: faster web browsing using speculative execution," in *NSDI*, 2010.

[10] M. Dong and L. Zhong, "Chameleon: a color-adaptive web browser for mobile OLED displays," in *MobiSys*, 2011.

[11] E. Fortuna, O. Anderson, L. Ceze, and S. Eggers, "A limit study of JavaScript parallelism," in *IISWC*, 2010.

[12] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: problem determination in large, dynamic Internet services," in *DSN*, 2002.

[13] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen, "Performance debugging for distributed systems of black boxes," in *SOSP*, 2003.