Structural Profiling of Web Sites in the Wild

Xavier Chamberland-Thibeault and Sylvain Hallé

Laboratoire d'informatique formelle
Université du Québec à Chicoutimi, Canada

Abstract. The paper reports results of a large-scale survey of 708 websites, in order to measure various features related to their size and structure: DOM tree size, maximum degree, depth, diversity of element types and CSS classes, among others. The goal of this research is to serve as a reference point for studies that include an empirical evaluation on samples of web pages.

1 Introduction

Over the past years, several tools and techniques have been developed to analyze, debug, detect errors, or otherwise process the output produced by web applications. Many of these tools focus on an analysis of the Document Object Model (DOM) of a page, and accessorily to the Cascading Stylesheet (CSS) declarations associated to its elements. For example, X-PERT [2] and $\mathcal{X}$Fix [5] attempt to fix cross-browser issues; Cornipickle [3] is a general purpose interpreter for declarative specifications over DOM elements and their rendered attributes; ReDeCheck [6] performs an analysis of a page's rendered DOM to detect responsive web design (RWD) bugs.

A common point to these approaches, and to many others, is that their scalability –and ultimately, their success– is dependent on features of a page that are typically related to its *size*. Hence, the running time for ReDeCheck scales according to the number of DOM nodes in the target page; Cornipickle scrapes a page in time proportional to the number of DOM nodes, and evaluates a declarative property in time proportional to the number of elements matching any of the CSS selectors found in that property; some RWD constraints scale proportionally to the number of nodes and the maximum number of direct children they have; etc.

Most of the aforementioned works duly provide an empirical evaluation of the proposed tools on a sample of pages or websites. However, it is hard to assess where these samples lie across the whole spectrum of web pages that may exist "in the wild". For example, the experimental analysis in [6] is run against documents of up to 196 DOM nodes: is this typical of a large web page, or a small one? Without data making it possible to situate such values with respect to a larger population, the authors, readers and reviewers alike are left speculating, with often conflicting viewpoints, to what extent the tested samples can be accepted as reasonably "real".

The present paper aims to address this issue. It reports on results of a large-scale analysis of 708 websites, with the goal of measuring various parameters related to the structure and size of their pages, such as the size of the DOM tree, degree distribution of its nodes, depth, distribution of various element types, diversity of CSS classes, etc. The goal of this research is to provide an objective (albeit partial) reference point allowing practitioners to quantitatively situate the samples used in research works that include an empirical evaluation. To summarize, the paper's contributions include a description of the methodology used to harvest and process data, a freely-available interactive package that can be used to explore the results, and a repository of all the raw data used in the analysis.

2 Methodology

To accomplish such an analysis, a few steps had to be followed. At first, we had to collect a large enough sample of websites for the analysis to be meaningful, then we had to find a way to collect the data from the previously found websites and, at last, process the recovered data. In this section, we shall present how each of those steps were fulfilled.

2.1 Website Collection

The first step of the process was to collect the list of websites to be included in our sample. We opted for a combination of two methods. First, we considered the first 500 most frequented websites in the world, by retrieving data from the latest list found on Moz [1]. However, this list contains many duplicates made of country-specific versions of the same platform. These duplicates have been removed, keeping only the first occurrence of each site. The remaining sites from this list (approximately 300) accounted for around 40% of our total sample.

This first sampling step provides us with a set of sites that are visited by the most users. However, this notion is orthogonal to the sample of sites most visited by an individual user. To illustrate this, consider a set of websites $\{s_0, \ldots, s_n\}$ and n users. Site s_0 is visited once by each user, and for each $i > 0$, site s_i is visited $n - 1$ times by user i. With $n = 10$, s_0 accounts for 53% of all traffic (compared to less than 5% for each of the remaining sites), but only accounts for 10% of all visits for any single user. That is, the fact that a site impacts the *most people* does not necessarily imply it impacts people *most significantly*.

In order to address this issue, we added a second part to our sampling step, by informally asking people around to provide us with the list of websites they use daily. Therefore, by combining these two data collection methods, we got the most commonly used ones in the world and various day-to-day websites. The complete dataset and scripts can be downloaded[1].

[1] DOI: `10.5281/zenodo.3718598`

2.2 DOM Harvesting

After the list of websites was established, the next step was to collect data on the DOM for each of these sites. This was done by creating a JavaScript program which is designed to run when the DOM of a page has finished loading.

The script starts at the `body` node of a page and performs a depth first preorder traversal of the integral DOM tree of that page. For every node, the script records and/or computes various features:

- *tag name*; this is used to record the usage proportion of elements
- *CSS classes* associated to the element; this is used to measure the diversity and distribution of CSS classes in a document
- *visibility status*; elements in the page can be made invisible in various ways: setting their position outside the viewport, setting their dimensions to 0, using the `display` or `visibility` CSS attributes; the script records if any of these techniques is applied on the rendered element
- structural information, such as degree and depth from the root.

For a given page, the output of the script is made of two files: the first is a JSON document containing a nested data structure that mirrors the structure of the page's DOM tree, where each node is a key-value map containing the features mentioned above. The file also contains global statistics computed on the DOM, such as the number of elements for each tag name, class and method of invisibility. The second document is a text file in the DOT format accepted by the Graphviz[2] library. It can be used to produce a graphical representation of the DOM, where each tag name is given a different color. An example of such a tree (for the Zippyshare.com website) is given in Figure 1.

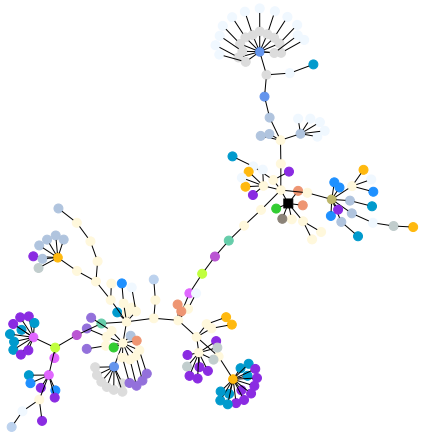


Fig. 1: An example of a DOM tree represented graphically, as produced by our harvesting script. Each color represents a different HTML tag name. The root of the tree is the black square node.

In order to actually harvest the data from a web page, we used the TamperMonkey[3] extension, which allows users to inject and run custom JavaScript code every time a new web page is loaded in the browser. This extension also presents the advantage that versions exist for multiple browsers,

[2] `https://graphviz.org`
[3] `https://www.tampermonkey.net`

including Chrome, Firefox, Edge and Safari. For each website in our sample, the home page URL was loaded and our TamperMonkey script was run on this page.

It shall be emphasized that the harvesting step on each web page is performed in the browser, and hence operates on the DOM tree and on properties of the elements *as they are actually rendered* by the browser. That is, our script does not perform a simple scraping of the raw HTML code returned by the server. Doing so would miss all the elements that are dynamically inserted or modified by client-side scripts at load time. Overall, running the scripts over the 708 sites takes approximately 6.5 hours and generates 62 MB of raw data.

2.3 Data Processing

As a second step, we aggregated these various measurements to compute statistics over the whole sample of pages, so as to get information such as distributions for various numerical parameters, using the LabPal experimental processing framework [4]. This library makes it possible to load, process, transform, and display empirical data in the form of tables, macros and plots. Our LabPal instance is designed such that every website is an experiment instance, whose task is to read the corresponding raw JSON files and compute various additional statistics on the structure of the DOM tree. These results are then collated in various ways into tables and plots.

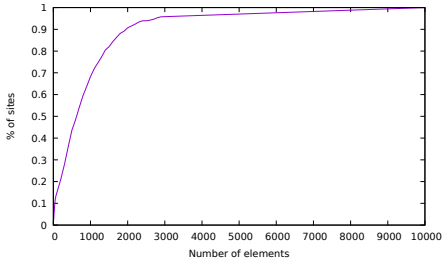
It is to be noted that some of the recovered files were not used in the analysis. As was said earlier, the automated loading of web pages made us retrieve a lot of advertisement pop-ups –which arguably do not really count as "real" web pages, in the sense meant by most papers concerned with analyzing the DOM. Manually inspecting each file to judge whether it is an advertisement page or a normal web page is a tedious process. We therefore opted for a more general rule that would remove most of these pages: we took away from the analyzed data each page that contained fewer than 5 DOM nodes, or whose URL belonged to a list of domains that are known to be advertisement pages.

3 Results and Discussion

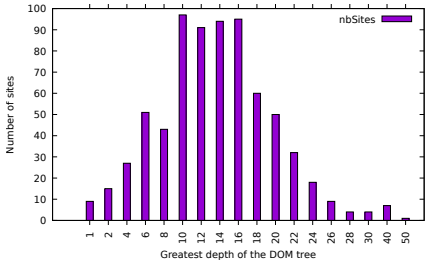
A full presentation of the results of the study cannot be done exhaustively in this paper due to a lack of space. In this section, we shall present a summary of the most important features, as well as a few interesting highlights and take-home points that should be taken from this study. The combined size of all websites visited amounts to a total of 623100 DOM nodes.

3.1 Website Profiles

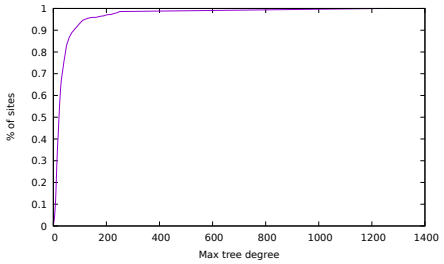
DOM Tree Structure A first statistic we computed is the number of elements in a web page across all sites. Figure 2a shows the cumulative distribution, where the line represents the fraction of all sites that have fewer than some number of elements. From this, we can observe that half of all websites have fewer than



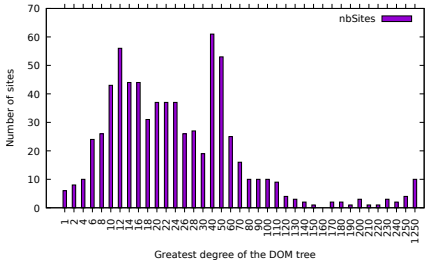
(a) Cumulative distribution of websites based on the size of DOM tree



(b) Distribution of websites based on DOM tree depth



(c) Cumulative distribution of websites based on maximum node degree



(d) Distribution of websites based on maximum node degree

Fig. 2: Graphical summary of DOM profiling

700 nodes, and 90% have fewer than 2000. The distribution of the number of elements in a web page very closely matches an inverse exponential function. In our dataset, the fraction of the number of pages having x elements or less is given by the function $1 - \frac{0.83}{e^{0.0011x}}$; the coefficient of determination is $R^2 = 0.999$, which indicates a surprisingly strong fit with the regression.

We also computed the depth of the DOM tree for each website; this corresponds to the maximum level of nesting inside the top-level `body` element. The distribution across websites is shown in Figure 2b.The distribution assumes a relatively smooth bell shape; 39% of all sites have a maximum depth between 10 and 16.

A complementary measurement to the depth of the DOM tree is the maximum degree (i.e. number of direct descendants) that a DOM node can have. This is represented in Figure 2c. 50% of websites have a maximum degree of at most 22, while 90% of websites have a maximum degree of at most 80. This time, a good fit for the data is the function $1 - \frac{4.91}{x^{0.92}}$, with a high coefficient of determination of $R^2 = 0.894$.

A fourth statistic related to tag usage across websites. We measured the relative proportion of each HTML element name, excluding tags corresponding to inline SVG markup. Unsurprisingly, `div` is the most frequent tag, representing

26% of all elements in a page. It is followed by `a` (19%), `li` (11%) and `span` (10%). Combined, these four elements account for two-thirds of all tags found in a page. No other tag has a frequency greater than 4%. Given that `div` and `span` have no special meaning, and are only used to enclose elements for display purposes, we can conclude that more than one third of all HTML markup is not semantic.

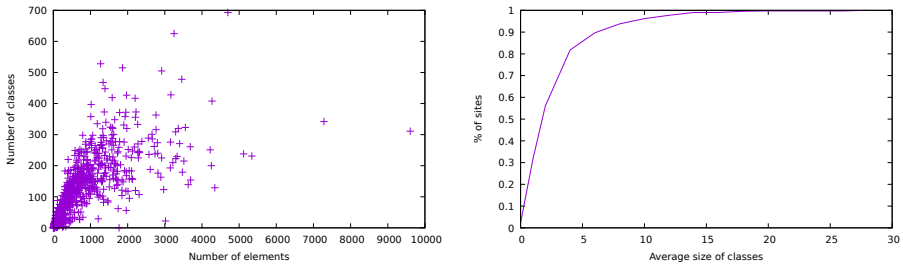
Visibility Status Of all DOM nodes in the websites analyzed, 54% were invisible to the user by one of the available techniques. This is a rather surprising finding, which means that more than half of a page's markup corresponds to elements that are not immediately displayed to the user, such as scroll-down menus or pop-ups. Figure 4 shows the distribution of websites according to the fraction of nodes that are invisible. One can see that this distribution is fairly uniform, with a roughly equal number of websites for each 10% interval of invisible nodes.

An interesting finding is that the techniques used to make elements invisible are not uniformly distributed. Of all invisible elements across the analyzed pages, 60% are made so by assigning them a negative position. Surprisingly, *none* of the websites analyzed used `display:none` or a dimension of zero to make nodes invisible.

nbElement	InvisibleType
0	Display
	widthOrHeight
131334	visibility
213840	outsidePosition
536700	negativePosition

Fig. 3: Total number of elements using each visibility.

CSS Classes Another aspect of our study is concerned with the CSS classes associated with elements of the DOM, either directly with the `class` attribute, or programmatically using JavaScript.



(a) Size of the DOM tree vs. number of CSS classes

(b) Cumulative distribution of websites based on the average size of a CSS class

Fig. 5: Statistics about the use of CSS classes.

We first checked whether there is a correlation between the size of the DOM tree and the number of distinct CSS class names occurring in the tree. This can be represented graphically with the plot of Figure 5a. As one can see, there is a relatively loose dependency between the size of a website and the number of classes it contains. We also calculated, for each website, the average size of each CSS class (i.e. the average number of DOM nodes belonging to each distinct CSS class present in the document). The function $1 - \frac{1.72}{x^{1.93}}$, which represents the fraction of all sites having an average CSS class size of x or less, fits the experimental data with a coefficient of determination of $R^2 = 0.931$.

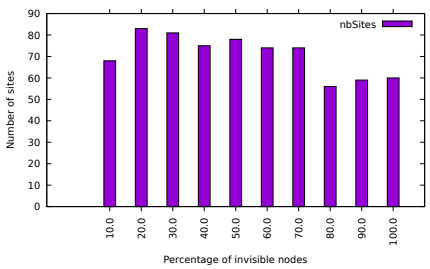


Fig. 4: Distribution of websites according to the fraction of all DOM nodes that are invisible.

3.2 Threats to Validity

Website Sample All the distributions and statistics computed in this study obviously depend on the sample of websites used for the analysis. Different results could be obtained by using different selection criteria. We tried to alleviate this issue by including in our sample a good fraction of sites selected using an objective and external criterion (the Moz top-500 list, which ranks sites according to their traffic). However, we also balanced this selection by including lesser known sites suggested by people based on their daily usage of the web. Finally, the relatively large size of our study (708) lessens the odds that our selection fortuitously picked only outliers in terms of size or structure.

Variance Due to Browser A single browser was used in our study, namely Mozilla Firefox. Since there sometimes exists a discrepancy between the pages rendered by different browsers, the actual DOM trees obtained could differ when using a different browser. However, most compliance violations affect the way elements are *graphically* rendered on the page, but not the actual contents of the DOM tree from which the page is rendered; hence these discrepancies do not affect the statistics we measure in our study. It shall also be noted that our use of Firefox has been made out of commodity: nothing technically prevents the same scripts from running in other browsers, thanks to TamperMonkey's wide support.

Homepage Analysis For all the sites considered, only the homepage has been analyzed. This is deliberate, so that the same methodology could be applied uniformly to all the sites in our study. Although many websites allow users to access to a different section after logging in, most of them have a relatively complete and usable homepage. There are, however, some exceptions; the most

notable is Facebook, which shows nothing but a login form to non-authenticated visitors. This has repercussions on the reported page size and structure.

To minimize the impact of this phenomenon, it could be possible to choose a different page in each website, in order to pick one that is representative –for example, one of the pages shown to the user after logging in. This, however, would introduce considerable complexity: valid credentials would need to be provided for all these sites, and specific instructions on how to reach the desired page (automatically) would have to be defined. More importantly, the choice of the "representative" page would introduce an additional arbitrary element that could in itself be a threat to validity.

4 Conclusion

In this paper, we have presented various statistics about the structural content of web pages for a sample of 708 websites, which includes the 500 most visited sites according to Moz [1]. Our analysis has revealed a number of interesting properties of websites in the wild. For example: the distribution according to their size closely follows an inverse exponential; most pages have fewer than 2000 nodes and a tree depth less than 22; more than half of all elements are hidden, and the majority of them are concealed by setting them to a negative position; most websites have CSS classes containing on average 10 elements or less.

It is hoped that these findings, and many more included in our online experimental package, can be used as a reference point for future research works on website analysis. They can help situate a particular benchmark or sample used in an empirical study, with respect to a larger population of websites "in the wild". As future work, we plan to expand the amount of data collected on each page, and intend to periodically rerun this study in order to witness any long-term trends over the structure of websites.

References

1. The Moz top 500 websites. `https://moz.com/top500`, Accessed October 20th, 2019.
2. S. R. Choudhary, M. R. Prasad, and A. Orso. X-PERT: accurate identification of cross-browser issues in web applications. In D. Notkin, B. H. C. Cheng, and K. Pohl, editors, *Proc. ICSE 2013*, pages 702–711. IEEE Computer Society, 2013.
3. S. Hallé, N. Bergeron, F. Guerin, G. L. Breton, and O. Beroual. Declarative layout constraints for testing web applications. *J. Log. Algebr. Meth. Program.*, 85(5):737–758, 2016.
4. S. Hallé, R. Khoury, and M. Awesso. Streamlining the inclusion of computer experiments in a research paper. *IEEE Computer*, 51(11):78–89, 2018.
5. S. Mahajan, A. Alameer, P. McMinn, and W. G. J. Halfond. Automated repair of layout cross browser issues using search-based techniques. In *Proc. ISSTA 2017*, pages 249–260, 2017.
6. T. A. Walsh, P. McMinn, and G. M. Kapfhammer. Automatic detection of potential layout faults following changes to responsive web pages (N). In M. B. Cohen, L. Grunske, and M. Whalen, editors, *Proc. ASE 2015*, pages 709–714. IEEE Computer Society, 2015.