ACE: An Adaptive CSS Engine for Web Pages and Web-based Applications

Luis A. Leiva
ITI/DSIC – Universitat Politècnica de València
Camí de Vera, s/n - 46022 (Spain)
llt@acm.org

ABSTRACT

ACE is a system that tailors web interfaces to the users' behavior without requiring end user intervention. By leveraging implicit interactions (e.g., tracking mouse or touch events), the visual appearance of page elements is subtly modified in an unsupervised and incremental manner. Such page elements (accessed by means of CSS selectors) and their alterable parts (defined as CSS properties) are both specified by the webmaster via JSON notation. ACE remembers the adapted styles for a given user, and consequently reapplies them when the user returns to the website, being also able to populate them to other non-browsed pages that share a similar structure.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Input devices and strategies, Interaction styles, User-centered design; H.5.4 [**Hypertext/Hypermedia**]: Navigation

General Terms

Experimentation, Design, Human Factors

Keywords

Adaptive Interfaces, Event Tracking, Implicit Interaction

1. INTRODUCTION

Approaches like 'universal design' or 'design for all' attempt to create technologies that have properties suitable to as many people as possible [4]. On the Web, however, the one-size-fits-all design cannot accommodate the broad range of abilities and skills of the potential visitors. Moreover, most web pages are visually-oriented and assume that users do not have functional impairments or special requirements. These problems are more aggravating when moving to the domain of tablets or mobile phones [7].

As such, proposals that actively involve end users (e.g., [3]) have been considered to adapt a particular web page to their needs. However, user-driven customization can be cumbersome, as it requires to perform additional activities that are not directly related to their main purpose for visiting the site [2]. Atterer et al. [1] observed that *knowing the user's every move* can provide meaningful statements about how users interact with web pages. Therefore alternative adaptation approaches without burdening the user can be derived.

On the other hand, websites often use template-based designs for keeping consistency across pages, using Cascading Style Sheets (CSS) to control their visual appearance. While the content can be frequently updated, this is not the case for the HTML structure, which can remain stable for weeks or even months [2]. Based on these notions, I developed an Adaptive CSS Engine (ACE) that works on the Document Object Model (DOM) of web pages (Figure 2).

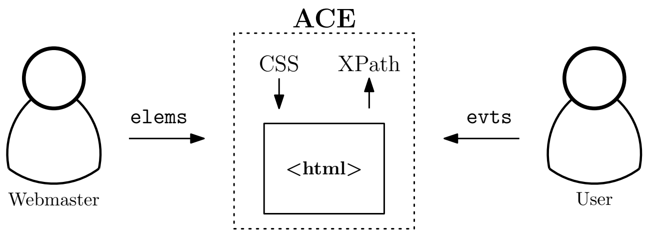
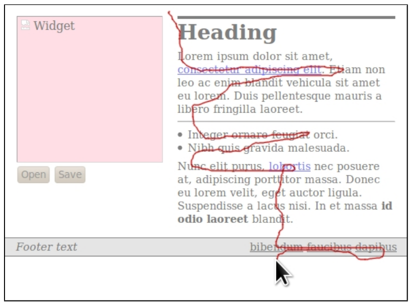


Figure 2: Workflow diagram. ACE tracks the elements indicated by the webmaster. When the user access a page, triggered browser events translate interacted DOM elements to XPath notation for later storing. On returning to the page, the CSS properties of such stored elements are restyled.

1.1 Method Rationale

This work lies in the Attentive User Interfaces paradigm, a recent interaction and visualization style that aims to focus (rather than distract) the user [10]. ACE leverages information that users perform with little or no awareness (e.g., mouse movements, taps, clicks) to incrementally mutate the appearance of interacted DOM elements. The idea is to introduce ephemeral changes that can be easily incorporated and do not alter the web design in a way that it might confuse the user.

The importance of an interaction towards a specific DOM element is measured as the proportion of UI-generated events on that widget between consecutive sessions [8]. Motivated by the fact that exertions are preceded by some intentionality, ACE measures these page-level interactions and use them as a proxy of user attention.



Heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam non leo ac enim blandit vehicula sit amet eu lorem. Duis pellentesque mauris a libero fringilla laoreet.

- Integer ornare feugiat orci.
- Nibh quis gravida malesuada.

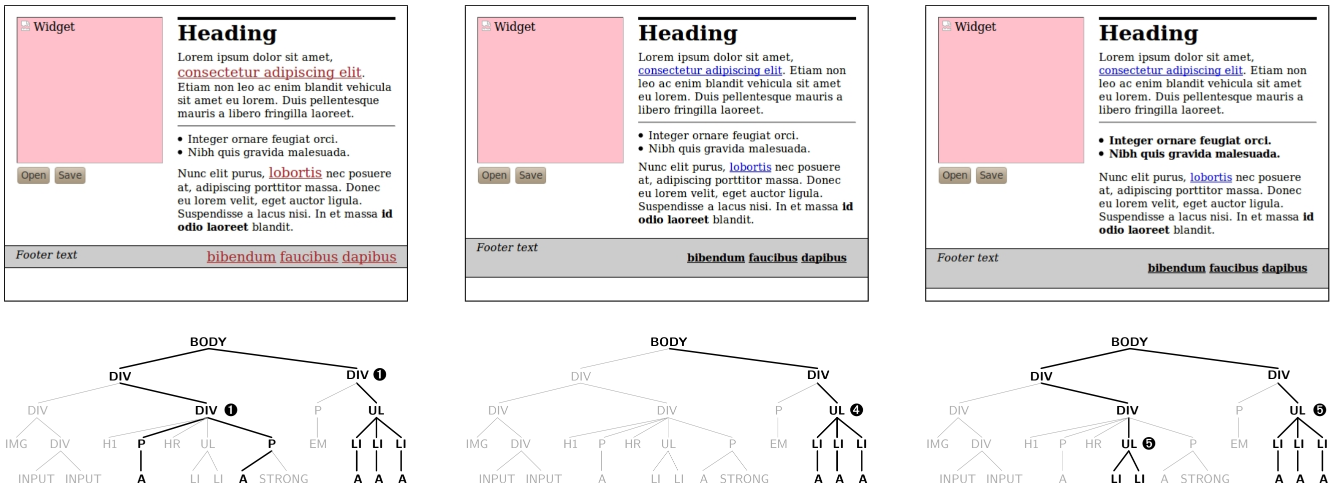
Nunc elit purus, lobortis nec posuere at, adipiscing porttitor massa. Donec eu lorem velit, eget auctor ligula. Suspendisse a lacus nisi. In et massa **id odio laoreet** blandit.

(a)

```
ACE.adapt({
❶   "div a": ["font-size", "color"],
❷   "p ul" : ["font-weight", "margin"]
});
```

```
ACE.adapt({
❸   "div + a": ["font-size", "color"],
❹   "p + ul" : ["font-weight", "margin"]
});
```

```
ACE.adapt({
❺   "div ∼ a": ["font-size", "color"],
❻   "p ∼ ul" : ["font-weight", "margin"]
});
```



(b) pattern: `E F` **(c)** pattern: `E + F` **(d)** pattern: `E ∼ F`

Figure 1: Original page design (1a) with an overlaid mouse behavior that may cause different adaptation possibilities, according to the following CSS combinator patterns: (1b) F elements that are descendants of E elements; (1c) F elements immediately preceded by E elements; (1d) F elements preceded by E elements. *Top row*: Sample JSON syntax. *Middle*: Corresponding page changes. *Bottom row*: DOM tree traversals, highlighting in bold the matched paths. Note that any combination of CSS selectors is supported, e.g., `"div + p.foo > span a:first-child"`.

This paper is based on the framework introduced in [7, 8], which has been specifically devoted to web pages in a browser environment; c.f. Features and Figure 1.

The main difference with other state-of-the-art interface adaptation techniques is that ACE relies on the webmaster's control to accommodate the appearance of the pages to the users in a transparent way (see Related Work for a brief discussion).

2. SYSTEM OVERVIEW

ACE is a totally self-contained JavaScript (JS) program that restyles *numerical* CSS properties, i.e., related to:

- **Dimensions** (e.g., `font-size`, `margin-top`.) These properties often do have a unit of measure[1], e.g., `16px`, `2.5em`, or `20%`, which is preserved once they are adapted.
- **Colors** (e.g., `background-color`, `border-color`.) These properties have an hexadecimal representation[2], which is specified either by a keyword (e.g., `"red"`) or by a numerical RGB specification (e.g., `#RRGGBB` or `rgb(R,G,B)`).

2.1 Features

ACE's main features are summarized in the following list:

- Does not require end user intervention.
- Supports desktop, touch, and mobile web clients.
- Any combination of CSS selectors can be used.
- Modifications are incrementally applied, ensuring that they are not intrusive for the user.
- Adaptation can be performed once the DOM is parsed or the page is fully loaded, so that third party or JS-controlled modifications are also supported.
- Since the system has a user's *interaction history*, it can populate the adaptation to other pages on the site that share a similar structure.

2.2 User Interaction Protocol

The webmaster indicates which elements and which CSS properties can be modified via a `<script>` tag, by using a straightforward JSON notation (see sample code snippets in Figure 1). Then:

1. Users access a page and such elements are tracked.
   (a) If users are new visitors, no modifications to CSS of page elements (i.e., DOM objects) are performed.
   (b) Otherwise, their previous interaction data are loaded, if available, and DOM elements are restyled.
2. Users leave the page and their tracked interaction data are stored on a local database (i.e., on the client side).

2.3 Implementation

ACE has been developed without relying in other JS libraries or frameworks, to ensure portability across different platforms and/or devices. A very simple JS API can be accessed via the global `ACE` namespace to invoke the system. The system exposes the public method `adapt()`, which takes two arguments: a configuration object (see script snippets in Figure 1) and a context (`window.document` by default). Notice that such syntax is dramatically less verbose than the notation envisioned in [7], and that it can support any combination of CSS selectors. This allows ACE more flexibility to adapt page elements, as shown in Figure 1.

Under the hood, the DOM elements that were specified in the configuration object as CSS selectors are retrieved by means of the `querySelectorAll()` method[3]. Then, if there were previous interaction data available, the matched DOM elements are restyled according to Equation 3. Also, in any case, event listeners are attached to these elements, in order to keep record of the (new) user interactions.

Interaction data are classified into different event lists, e.g., hovered, typed, scrolled, or tapped elements; where each list member is a DOM fragment (see bottom rows of Figure 1) in XPath notation — to allow retrieving them later on subsequent user visits. Such a list members are scored according to the number of browser-generated events, or, in other words, how many times the user has interacted with those DOM elements. The Interaction Scoring Scheme is described in the next section.

Finally, data are persistently stored on the client side by means of the `localStorage` API[4] or a `document.cookie` as a fallback mechanism, so that the users' privacy is completely under their control; e.g., they may opt to configure their browsers to restrict access to the storage context, or automatically delete stored data after some time.

2.4 Interaction Scoring Scheme

As commented above, each interacted element is assigned a score s, which depends on the type of events. Let n_i be the number of times an event of type i was fired for a certain DOM element, and let N be the number of all fired events during browsing. The assigned score for that event is

$$s_i = \zeta(n_i/N) \tag{1}$$

where $\zeta(\cdot)$ is a symmetric sigmoid function. The idea is to impose scores to follow a non-linear distribution, in order to ensure that adaptation is smoothly applied.

Note that if an element receives different types of interactions (e.g., an `input` text field can listen to `click`, `focus`, or `keydown` events) then its scores need to be fused in order to compute a single value. ACE uses the weighted mean as a fusion scoring method:

$$s = \sum_{i=1}^{m} w_i s_i \quad \text{with} \quad \sum w_i = 1 \tag{2}$$

where m is the number of computed scores for that element.

The value v of a CSS property is then modified this way:

$$v = v(1 + s) \tag{3}$$

On subsequent user visits, the new scores s'_i and how they will affect the CSS properties are both updated as follows:

$$\begin{aligned} s'_i &= \zeta(n'_i/N) - s_i \\ v' &= v(1 + s') \end{aligned} \tag{4}$$

According to equations (3) and (4), when a user enters a site for the first time, elements are rendered as they were designed, as the system has no information about previous interactions ($s_i = 0 \;\; \forall i$). Then, in successive visits to the same page (or pages that share the same DOM structure) the system will react accordingly, i.e., modifying the value of those CSS properties specified by the webmaster based on the amount of user's interactions. Given that scores are bounded to the interval $(-1, 1)$, a score of, say, 0.05 for a `margin-top` property will be interpreted as "increasing by a 5% the value of the top margin." Conversely, a score of -0.1 for a `color` property will be interpreted as "decreasing by a 10% (the contrast or saturation of) the font color."

3. EXPERIMENTAL EVALUATION

In terms of system performance, ACE takes a few milliseconds to complete the adaptation process. A series of JavaScript benchmarks were performed on the mock-up shown in Figure 1 with different configuration objects and CSS properties. The machine was an i686 @ 2 GHz with 1 GB of RAM. The adaptation code was executed 100 times and benchmark results were averaged. Concretely, for 10 items (that were specified by different CSS level 3 selectors[5]) having at most 5 properties each, in all tested browsers (Firefox 7, Chrome 15, Opera 11, Internet Explorer 9, and Dolphin 2.2) the average times were below 20 ms, with standard deviations below 0.1 in all cases. A demonstration example is available at `http://personales.upv.es/luileito/ace/`, so it can be easily inspected how the system can be configured.

Regarding human evaluation, the most suitable evaluation method is still not clear. As a preliminary approximation, an informal study involving 12 users was carried out, in which participants where asked to browse a mock-up website [7]. At the end of the test, users answered three questions (see Figure 3); **Q1**: Do you think page elements are well laid out? **Q2**: Did you notice any change on the page, regarding the first time you visited it? **Q3**: If so, did you find distracting those changes?

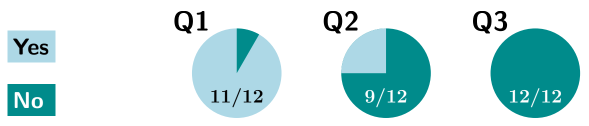


Figure 3: Results of the user questionnaire.

Overall, users' acceptability towards the method was perceived as positive. As observed, nine of them did not notice the automatic modifications, and none found distracting those changes while browsing. The informal pilot study, although being not conclusive, revealed that this adaptation technique has an interesting potential in building client-side adaptive user interfaces.

4. RELATED WORK

Currently, with the exception of customizing font preferences, browsers do not provide end users with substantial control over how web pages are rendered. This way, researchers have proposed different approaches to layout adaptation that mainly involve user's manual work.

Ivory et al. [5] employed learned statistical profiles of award-winning websites to suggest improvements to existing designs; however, changes would be manually implemented. Tsandilas and Schraefel [9] introduced an adaptive link annotation technique, although it required the user to perform direct manipulation of a middleware application. Notable approaches in this direction include the work of Bila et al. [2], where the user must actively modify the layout contents. Kurniawan and co-workers [6] proposed to override the visual layer of a web page with custom CSS, but unfortunately updates had to be performed by hand. Now that web standards have minimized browser inconsistencies, I foresee this approach as an alternative to automate the adaptation of web page design without disrupting users' navigation habits.

5. DISCUSSION

When browsing, there is a lot of information that is not provided on purpose by the user. As such, page-level interactions can help to identify what characteristics of pages add benefit (or not). The main advantage of leveraging implicit input from the user is that every interaction on the website can contribute to enhance its utility. Additionally, such an input removes the cost of having to interrupt the user to submit explicit feedback about the website.

A known limitation of ACE is that currently it can adapt only properties that vary in a numerical range. However, in a future it is expected to be able to map semantic properties; e.g., to adapt the `text-align` property of a text paragraph one could use:

$$v = \begin{cases} \texttt{"left"} & \text{if } s \in (-1, -0.5], \\ \texttt{"center"} & \text{if } s \in (-0.5, 0.5), \\ \texttt{"right"} & \text{if } s \in [0.5, 1). \end{cases}$$

Also, due to ACE's simplicity, more advanced adaptation strategies such as re-arranging several page elements (beyond alignment) or inserting/removing page content would require a technically more sophisticated approach.

All in all, this technology enables a straightforward means to invisibly enhance the utility of web pages and web-based applications; e.g., in terms of usability, accessibility, readability, interactivity, or performance. Systems like ACE may allow websites to be flexible enough to meet different user needs, preferences, and situations.

6. CONCLUSION AND FUTURE WORK

Dynamic and continuously changing environments like the Web demand new means of building UIs that are aligned to the skills of the users. ACE is my contribution to help people enhance the way they create websites.

In addition, it is possible to easily extend the `ACE` namespace with custom methods and/or properties, since it is assigned to the global `window` object of every web browser. This way, for instance, developers could use ACE as a base system for developing other applications; e.g., logging the tracking data in an external database to generate aggregated statistics, or present the users with detailed in-page analytics about how they interact on the site. Moreover, having a user model based on page-level interactions would allow webmasters to deploy customized design guidelines that impact page presentation.

Finally, I believe that this work has barely scratched the surface of a wealth of applications that can be developed by tracking the user activity and dynamically changing the browsing experience in response. For instance, the technology described in this paper could be a good place to start creating new types of self-adapting websites. Also, integrating ACE with an eye-tracker would provide a finer-grained and potentially more focused analysis of user interactions. Even more, other biometric inputs such as electrocardiogram signals would allow developers create "organic" pages that are able to react to the emotions of the users.

Notes

[1] Most browsers internally normalize computed CSS properties to equivalent `px` values.
[2] Also, in most browsers, hexadecimal colors are converted to equivalent `rgb(R,G,B)` values.
[3] http://www.w3.org/TR/selectors-api/
[4] http://dev.w3.org/html5/webstorage/
[5] http://www.w3.org/TR/css3-selectors/

Acknowledgements

Work partially supported by the Spanish MEC/MICINN under the MIPRCV "Consolider Ingenio 2010" (CSD2007-00018) project.

7. REFERENCES

[1] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move — user activity tracking for website usability evaluation and implicit interaction. In *Proc. WWW*, 2006.

[2] N. Bila, T. Ronda, I. Mohomed, K. N. Truong, and E. de Lara. PageTailor: reusable end-user customization for the mobile web. In *Proc. MobySys*, 16–29, 2007.

[3] M. Bolin, M. Webber, P. Rha, T. Wilson, and R. C. Miller. Automation and customization of rendered web pages. In *Proc. UIST*, 163–172, 2005.

[4] K. Z. Gajos, J. O. Wobbrock, and D. S. Weld. Automatically generating user interfaces adapted to users' motor and vision capabilities. In *Proc. UIST*, 2007.

[5] M. Y. Ivory and M. A. Hearst. Statistical profiles of highly-rated web sites. In *Proc. CHI*, 367–374, 2002.

[6] S. Kurniawan, A. King, D. Evans, and P. Blenkhorn. Personalising web page presentation for older people. *Interacting with Computers*, 18(3):457–477, 2006.

[7] L. A. Leiva. Restyling website design via touch-based interactions. In *Proc. mobileHCI*, 91–94, 2011.

[8] L. A. Leiva. Interaction-based user interface redesign. In *Proc. IUI*, 163–172, 2012.

[9] T. Tsandilas and M. C. Schraefel. User-controlled link adaptation. In *Proc. HT*, 152–160, 2003.

[10] R. Vertegaal. Attentive user interfaces. *Commun. ACM*, 46(3):31–33, 2003. Editorial note.

APPENDIX

Video available at http://vimeo.com/luileito/ace-www.
Code and demo: http://personales.upv.es/luileito/ace/