

From the Editor in Chief...

Taking Back the Web

Robert E. Filman • filman@computer.org



The original commercial Web browsers seemed more oriented toward pleasing Web page creators than their consumers. I found the ability of scripts to take over browser processing, generate a desktop full of pop-up windows, and execute arbitrary code on my machine offensive. (I was particularly offended when Internet Explorer stymied my self-educational efforts by occasionally refusing to let me see a Web page's source. A Web page's source is clearly no deep secret — after all, it's revealed by a few lines of Java. I suspected a dark conspiracy between Microsoft and the page-creation establishment, but that limitation turned out to be a bug, provoked, among other things, by having an alias to the text viewer on the desktop.)

But in a funny sense, there was a dark conspiracy. Early on, it wasn't a foregone conclusion that the Web would take off; commercial organizations had to be convinced to put content out there. (That changed.) Browser creators weren't eager to put obstacles in Web page creators' paths. Holders of large quantities of intellectual property often have a more expansive interpretation of intellectual property rights than consumers do. (Or not, depending on which way they think they might make more money.) Browsers and the Internet exposed operating systems to environments that were far more malicious than their designers had anticipated, particularly those operating systems in which — how can I say this politely — a seemingly innocuous action like adding a link to an application can cause another application to misbehave in unexpected ways.

In any case, it's my computer and I don't understand why purveyors of Web pages get to control what I see. Actually, I do understand why — for the same reason that commercial television broadcasts include commercials: the ability to get attention for selling soap is the incentive for hiring production crews to make the show. I'd like to get rid of the Web page commercials in the same

way that users of digital video recorders (DVRs) want to skip the commercials (I care about the content and not the soap). DVRs are a threat to the business model of television. If I want to control what my browser presents (and informs the sending Web site), well, the traditional answer has been simple: write my own browser. Although I think I could figure out how to do that, and as much as I enjoy the Web, that's probably a bit more effort than it's worth.

Greasemonkey

That could be changing. Greasemonkey is a Firefox extension that lets you load and execute JavaScripts on Web pages. That is, you can define JavaScripts and specify a pattern for pages that are to execute these scripts. Once loaded, these scripts can manipulate the page's content, to, for example, remove particular elements, rearrange the page, or insert additional HTML. You can build scripts that are specific to particular sites or generic to all sites.

Examples of what this technology generically enables include suppressing the advertisements on a page, altering links that redirect through a tracking site so that they go directly to the desired target, putting up comparative prices from other sites when shopping, changing fonts and formats to make text more comprehensible, restoring parts of the interface (for example, context menus and status bars) on sites that disable them, automatically reloading pages to prevent auto-logouts or to always display more current data, skipping the registration process demanded by some sites by using shared logins, suppressing offensive language, allowing mouse-free navigation, converting currencies by mousing over them, and removing wrapper frames. Site-specific actions include saving standard searches in Gmail, presenting the prices at alternative booksellers on Amazon, redoing Google's entire interface, or presenting the

IEEE Internet Computing

IEEE Computer Society Publications Office
10662 Los Vaqueros Circle
Los Alamitos, CA 90720

EDITOR IN CHIEF

Robert E. Filman • filman@computer.org

ASSOCIATE EDITORS IN CHIEF

Fred Douglass • f.douglass@computer.org
Doug Lea • dl@cs.oswego.edu

EDITORIAL BOARD

Helen Ashman • hla@cs.nott.ac.uk
Jean Bacon • jean.bacon@cl.cam.ac.uk
Elisa Bertino • bertino@cerias.purdue.edu
Scott Bradner • sob@harvard.edu
Junghoo Cho • cho@cs.ucla.edu
kc claffly • kc@caida.org
Siobhán Clarke • siobhan.clarke@cs.tcd.ie
Ian Foster • foster@cs.uchicago.edu
Li Gong • li.gong@ligong.com
Monika Henzinger • monika@google.com
Michael N. Huhns • huhns@sc.edu
Leonard Kleinrock • lk@cs.ucla.edu
Samuel Madden • madden@csail.mit.edu
Daniel A. Menascé • menasce@cs.gmu.edu
Chris Metz • chmetz@cisco.com
Charles J. Petrie* • petrie@nrc.stanford.edu
Krithi Ramamritham • krithi@cse.iitb.ac.in
Michael I. Schwartzbach • mis@brics.dk
Munindar P. Singh* • singh@ncsu.edu
Craig Thompson • cwt@uark.edu
Andrew Tomkins • atomkins@yahoo-inc.com
Steve Vinoski • vinoski@ieee.org
Dan S. Wallach • dwallach@cs.rice.edu
Jim Whitehead • ejw@soe.uscs.edu

* EIC emeritus

IEEE COMMUNICATIONS SOCIETY LIAISON

G.S. Kuo • gskuo@ieee.nccu.edu.tw

CS MAGAZINE OPERATIONS COMMITTEE

Bill Schilit (Chair), Jean Bacon, Pradip Bose, Arnold (Jay) Bragg, Doris L. Carver, Kwang-Ting (Tim) Cheng, Norman Chonacky, George Cybenko, John C. Dill, Robert E. Filman, David Grier, Warren Harrison, James Hendler, Sethuraman (Panch) Panchanathan, and Roy Want

CS PUBLICATIONS BOARD

Jon Rokne, Bill N. Schilit, Roger U. Fujii, Steven Tanimoto, Michael R. Blaha, Frank Ferrante, Linda Shafer, Wenping Wang, Phillip Laplante, and Mark Christensen

TECHNICAL COSPONSOR:



“printer-friendly” (that is, ad-free) version of the *New York Times*.

Having It Your Way

So how can you bring this power to your desktop? Broadly, there are two ways: you can download scripts that someone else has written or write your own.

There are many directories full of Greasemonkey scripts. For example, the generic examples I mentioned come from <http://dunck.us/collab/GreaseMonkeyUserScripts>. I've tried various scripts I've found on the Internet, with mixed results — that is, many of them don't work. This can be expected, given that I haven't paid for this freeware, that it needs to run on a variety of hardware, operating systems, and browser versions, and that particular Web pages can be moving targets. However, most people have only a limited tolerance for things that don't work (except, perhaps, for that operating system mentioned earlier). Widespread acceptance requires widespread reliability.

Of course, we're engineers here, and we can roll our own scripts. I set out to learn how to do this. My starting point was Mark Pilgrim's “Dive Into Greasemonkey,” a free, online book (<http://diveintogreasemonkey.org>). Pilgrim has also recently published a conventional (that is, physical) book on Greasemonkey programming.¹ Greasemonkey relies on your ability to understand two key technologies: JavaScript and the structure of Web pages. JavaScript, despite the ugly semicolon and curly-brace syntax, is intellectually a descendant of Scheme and Self. It borrows from Scheme notions of interpretation, functions as objects (including closures), and a dynamic environment. It borrows from dynamic object languages such as Self and certain AI systems the idea of objects as dynamically extendable maps from names to values. Of course, part of the power and attrac-

tion of Lisp and its descendants comes from their fluid programming and debugging environments — something more difficult to achieve for Web pages, but for which rudimentary tools are appearing.

The rub with respect to effective programming is that these technologies are being applied to the Web page data object, an often messy structure that intermixes structural elements (headings, anchors, and lists) with a variety of presentation elements (not only directly with font and style commands, but also indirectly with style sheets). A Web page creator has many ways to encode the page's content. Scripts must often rely on pattern expressions in XML query (XPath) expressions, but pattern matching can be an easily disrupted technique. Web page formats evolve. Good Greasemonkey scripts need to consider all these possibilities — a computationally and intellectually intractable problem.

Scripting Scripts

Early technology adopters wrote their own HTML, but GUIs were necessary before every grandmother could have her own custom Web page. Now grandmothers can build Web pages without knowing of the syntax or even the existence of HTML. Skilled JavaScript programmers can build Greasemonkey scripts, but learning JavaScript, XPath, and Web page formatting details is too much for most. To be transformative, this technology needs to become more accessible.

One such initial effort is Platypus (<http://platypus.mozdev.org>), which advertises itself as providing a graphical interface for creating Greasemonkey scripts. This is roughly on the level of keyboard macros — I'll watch what you do and repeat it for you next time. The fly in this ointment (as experienced keyboard macro writers know) is that it takes foresight and cleverness to write macros that

don't stumble on minor variations of structure. That is, the macro that moves over a character on a numbered list to skip the "number" falters when the list suddenly turns up with 10 items.

Platypus is a good start, but much more effort (and creativity) is needed. Ideally, a user-capable Greasemonkey scripting environment needs to present Web pages in a conceptual model that is straightforward and easy for the user to understand, as opposed to one that's close to the textual metal of presented pages. This will likely blend elements of the watch-what-I-do macros with wizards that disambiguate possible interpretations of gestures ("When you said you didn't want to see this image, did you mean to suppress all images, just images that come from another site, just images from the site of this image, or just this particular image?") and provide a more sensible vocabulary of actions. It will not be able to demand that the user know a volume of function calls or field names, while nevertheless making a rich library of additional functionality easily accessible. Excel, to say something nice about a product from that large operating system company, is a fairly successful example of enabling such user programming — spreadsheets were the original transformative technology that enabled laypersons to define useful computations without having to train for the computational priesthood. The challenge for Web page manipulation is to make such manipulation Excel-transparent without the underlying data model being Excel-simple.

Will Greasemonkey transform the Web? Perhaps. Greasemonkey and DVRs both have the potential to distort the economic model underlying the way the technology is used. Television producers are fighting back against DVR technology on several

fronts: politically, trying to make "transforming" a television show an illegal violation of copy- or some other fancied right; technologically, by trying to make the "to-be-skipped" commercials less obviously commercials; and semantically, by moving from a model in which the commercial is a separate entity to one in which the product endorsements are part of the text and visuals of the entertainment itself.

Faced with the Greasemonkey threat of removing the parts of Web pages that pay the bills, we can expect similar responses from Web page creators. We can certainly expect the usual political efforts, with the usual one-sided polemics about ethics. Technologically, we'll have more complex presentation mechanisms (for example, mixing the advertising with the context in executable form, or Web pages that check to see that they haven't been modified before they'll present their information), more useful advertising (the genius of Google is to present ads that are twice as likely to be clicked and correspondingly only half as annoying), frequent changes of Web page formats to catch naïve scripts, and more complex intermixing of desired information with the economically sustaining information.

We have seen a continuing arms race between malware writers and detectors and between Web page creators and search engine writers. If this technology moves beyond high-tech early adopters to actually impact the way the Web is used, we should expect a similar continuing escalation of more clever Web page hacks and more cleverly generated Web pages. I just can't tell whether that particular arms race will produce a détente or mutually assured destruction. □

Reference

1. M. Pilgrim, *Greasemonkey Hacks*, O'Reilly Media, 2005.

IEEE Internet Computing

STAFF

Lead Editor: Rebecca L. Deuel
rdeuel@computer.org

Group Managing Editor: Steve Woods

Staff Editors: Kathy Clark-Fisher,
Jenny Ferrero, and Brandi Ortega

Production Editor: Monette Velasco

Magazine Assistant: Hazel Kosky
internet@computer.org

Graphic Artist: Alex Torres

Contributing Editors: Cheryl Baltes,
Greg Goth, Kerl Schreiner, and Alison Skratt

Business Development Manager:
Sandy Brown

Publisher: Angela Burgess,
aburgess@computer.org

Associate Publisher: Dick Price

**Membership/Circulation Marketing
Manager:** Georgann Carter

Advertising Supervisor: Marian Anderson

HOW TO REACH IC

Writers: Access www.computer.org/internet/author.htm. Articles are peer reviewed for technical merit and copy edited for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society.

Letters to the Editors: Send letters to Rebecca Deuel, Lead Editor, rdeuel@computer.org. Provide an email address or daytime phone number with your letter.

On the Web: Access www.computer.org/internet/ or our online resource, *IEEE Distributed Systems Online* at www.dsonline.computer.org.

Subscribe: Visit www.computer.org/subscribe/.

Subscription Change of Address: Send requests to address.change@ieee.org.

Missing or Damaged Copies: Contact help@computer.org.

Reprints of Articles: For price information or to order reprints, send email to internet@computer.org or fax +1 714 821 4010.

Reprint Permission: To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at copyrights@ieee.org.

Copyright and reprint permission: Copyright © 2006 by the Institute of Electrical and Electronics Engineers. All rights reserved. Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of U.S. copyright law for patrons' private use those articles that carry a code at the bottom of the first page, provided the per-copy fee in the code is paid through the Copyright Clearance Center, 222 Rosewood Dr., Danvers, Mass. 01923. For copying, reprint, or republication permission, write to Copyright and Permissions Dept., IEEE Service Center, 445 Hoes Ln., Piscataway, NJ 08855-1331.