# Why We Customize the Web

Lydia B. Chilton, Robert C. Miller, Greg Little, Chen-Hsiang Yu
MIT CSAIL

Traditionally, customization has been associated with desktop applications. Many desktop applications have customizable menus and tool bars, with options that the user can pick from to customize their environment. End user programming is a more powerful kind of customization tool, which exists in some desktop applications in the form of macros and scripting languages, and on the web in various incarnations. On the web, customizable menus are rare; end user programming seems to be the main form of customization.

In order to make generic software more efficient and usable, end users should be able to customize it to fit their work habits and preferences. Unfortunately, that has been a limited reality for desktop applications. In desktop applications, it's hard to know what can be changed, how to select only specific parts of an interface to keep, and how to incorporate data and functionality from other applications. Yet the web has offered new possibilities in the realm of customization. Web page components are easy to identify, and almost any component can be altered in behavior, style or placement.

The modern web now offers not only a wealth of information and media but also applications. The combination of private data, public data and desktop-quality applications has created a powerful computing environment. As people use the web to shop, plan events, consume media, manage their personal information and much more, they want to customize the sites they use to shape around their preferred work flow. Fortunately, the openness and flexibility of the web platform enables customizations that would not have been possible on the desktop.

This chapter looks at part of the landscape of web customization, particularly around the Mozilla Firefox web browser. We use *customization* in a broad sense to refer to modifications to a web site that are not hosted on the site itself. Customizations may automate interaction with the site -- clicking links, filling in forms, and extracting data automatically. They may change the site's appearance by adding, removing, or rearranging page components. Or they may combine a site with other web sites, commonly called a *mashup*.

Like most of this book, this chapter focuses on customizations hosted by the web browser, at the risk of glossing over the rich world of mashups and web services hosted out in the web. Examples include specific mashups like the classic HousingMaps[1] mashup of craigslist and Google Maps, as well as programming systems like Yahoo Pipes[2], Google Mashup Editor[3], and Microsoft Popfly[4]. Although some of these web-hosted customizations may be aimed at or created by end users, browser-hosted customizations are more general, and this is where we will focus our main attention.

1. housingmaps.com
2. pipes.yahoo.com
3. code.google.com/gme
4. popfly.com

The rest of this chapter is organized as follows. We start by looking back at previous research on desktop customization, which uncovered motivations and practices that continue to be true on the web, and on web mashups. We then survey some examples of customizations for Firefox, drawing from public repositories of Firefox extensions, Greasemonkey, Chickenfoot, and CoScripter scripts. We follow with a discussion of motivations for customization revealed by these examples, some of them unique to the web platform.

## Related Work

Early research on customization was exclusively on desktop customization. In many ways, web customization achieves the same basic goals as traditional customization, making an interface better suit personal needs. Yet the web's unique platform and culture bring about several differences from traditional customization. Some of the barriers to desktop customization aren't as prevalent on the web; the types of changes that user make on the web are different, and the customization on the web is driven by a wider set of the motivations.

In a study of the WordPerfect word processor for Windows, Page *et al.*[5] describe five types of customization found on the desktop. Most common was (1) setting general preferences; then (2) using or writing macros to enhance functionality of the application; (3) granting easier access to functionality, typically by customizing the toolbar with a shortcut; (4) "changing which interface tools were displayed," for example, showing or hiding the Ruler Bar; and (5) changing the visual appearance of the application. These categories are certainly relevant to web customization. In fact, the customizations encouraged by end user programming are most strongly associated with enhancing functionality and granting easier assess to existing functionality.

Wendy Mackay studied[6] a community of customization around a shared software environment, MIT's Project Athena, and identified four classes of reasons that users customized. The top technological reason cited was that "something broke," so that the customization was actually a patch. The top individual factors cited were first that the user noticed they were doing something repetitively and second that something was "getting annoying." All of these reasons can be found in web customization as well.

In the same study, Mackay describes barriers to customization reported by users. The most reported barrier overall was lack of time, but the chief technical reasons were that it was simply too difficult and that the application programming interfaces (APIs) were poor. End user programming systems for the web have endeavored to ease the technical burden. Additionally, customizable web sites all use the same technology, and have the same structure -- the Document Object Model of HTML, modifiable with JavaScript -- which curbs the basic reliance on APIs.

Mackay defines customization as changes users can make without writing code. In this chapter, we regard end user programming as one kind of customization. Some end user web programming systems don't require writing code at all (notably CoScripter), but many rely on the user to do at least some coding. The goal of end user programming is to lower the threshold of traditional programming and thus alleviate some of the technical barriers to customization.

---

5. S.R. Page, T.J. Johnsgard, U.Albert, C.D. Allen. "User customization of a word processor." CHI '96.
6. Wendy Mackay. "Triggers and barriers to customizing software." CHI '91.

In a study of users' sharing customizations stored in text files[7] Mackay highlights how important it is that authors of customizations share their work with nonprogrammers who want customizations but don't know how to write them.  The same culture of sharing is just as important and widely prevalent on the web. Mackay describes a group of "translators" who introduce the non-programmers to existing customizations.  The web has its own channels for distributing customizations that rely not just on repositories of customizations (like blogs and wikis) but also on personal interactions (email, forums, comments, and so on).

Mackay recommends that users "want to customize patterns of behavior rather than lists of features."  Desktop applications today still concentrate on customizing lists of features.  End user programming on the web, however, leans heavily toward what Mackay recommends -- adding features, combining features, and streamlining the process of completing tasks.

MacLean *et al.*'s classic end-user programming system Buttons[8] is a customization tool built around macros that can be recorded and replayed by pressing a single button.  MacLean *et al.* point out that there is a "gentle slope" of customizability where technical inclination governs ability and willingness to customize.  Web customization exhibits a similar slope.  At the very basic level, users have to know about and trust browser extensions in order to even run browser-hosted customizations. To start writing customizations, a user must become familiar with an end-user programming tool.  At the programming level, it may be necessary for customizers to know about the HTML DOM, JavaScript, and browser APIs such as Firefox's XPCOM components.

More recently, researchers have studied practices of customization on the web, particularly mashup creation and scripting.  Zang *et al.*[9] found through surveys of mashup developers that mapping sites (specifically Google Maps and Yahoo Maps) were among the most popular to mash up with other sites, and that many programming obstacles still exist to for mashup creation.  Wong & Hong[10] surveyed mashup artifacts rather than the people who created them, examining two public directories of web-hosted mashups (ProgrammableWeb and MashupAwards.com) in order to discover interesting characteristics and dimensions for classifying existing mashups. Although aggregation of data from multiple sites was a very common feature, Wong & Hong also found mashups designed to personalize a site, filter its contents, or present its data using an alternative interface.  This chapter takes a similar approach to Wong & Hong, surveying customizations in four public repositories, but focusing on browser-hosted customizations.

The CoScripter system in particular has been the subject of two studies of web customization practice. Leshed *et al.*[11] studied how CoScripter was used by internal IBM users, while Bogart *et al.*[12] looked at much wider use on the web after CoScripter was released to the public.  Some of findings from these two studies will be cited in the next section.

---

7. Wendy Mackay, "Patterns of sharing customizable software." CSCW '90.
8. A. Maclean, K. Carter, L. Lövstrand, T. Moran. "User-tailorable systems: pressing the issues with buttons." CHI '90.
9. q.v. Zang et al. chapter in this book
10. Wong & Hong, "What do we 'mashup' when we make mashups? ", WEUSE 200.
11. G. Leshed, E. Haber, T. Matthews, T. Lau. "CoScripter: Automating and sharing how-to knowledge in the enterprise." CHI 2008, 1719-1728.
12. C. Bogart, M. Burnett, A. Cypher, C. Scaffidi. "End-User Programming in the Wild: A Field Study of CoScripter Scripts," VL/HCC 2008.

## Examples of Web Customization

We now turn to a brief, informal survey of examples of web customization found on today's web. These examples are mostly drawn from the public repositories of four web customization systems, all designed for the Mozilla Firefox web browser. Firefox extensions[13] add new user interface elements and functionality to the browser, generally using Javascript and HTML or XUL (Firefox's user interface language). Greasemonkey[14] allows user-provided Javascript code to be injected into a web page and customize the way it looks or acts. Chickenfoot[15] is a Javascript programming environment embedded as a sidebar in the browser, with a command library that makes it easier to customize and automate a web page without understanding the internal structure of the web page. Finally, CoScripter[16] allows an end user to record a browsing script (a sequence of button or link clicks and form fill-in actions) and replay it to automate interaction with the browser, using a natural-language-like representation for the recorded script so that knowledge of a programming language like Javascript is not required.

Each of these systems has a public repository of examples, mostly written by users (not the system developers). Firefox extensions are hosted by a Mozilla web site[17]. Greasemonkey has a public scripts wiki[18], as do Chickenfoot[19] and CoScripter[20]. CoScripter is worth special mention here, because *all* CoScripter scripts are stored on the wiki automatically. The user can choose to make a script public or private, but no special effort is required to publish a script to the site, and the developers of CoScripter have been able to study both public and private scripts.

Other customization techniques exist. *Bookmarklets* are small chunks of Javascript encoded into a javascript: URL, which can be invoked when a web page is showing to change how that page looks and acts. Bookmarklets have the advantage of being supported by all major browsers (assuming the Javascript code in the bookmarklet is written portably). Other browser-specific systems include User Javascript[21] for the Opera browser, which is similar to Greasemonkey, and Browser Helper Objects[22] for Internet Explorer, which are similar to Firefox extensions.

Since this survey focuses mainly on customizations for Firefox that users have chosen to publish in a common repository, it may not reflect all the ways that web customization is being used in the wild, but these examples show at least some of the variety, breadth, and in particular varying motivations that drive users to customize the web.

The rest of this section describes some interesting dimensions of web customizations:

- the kinds of sites customized;
- the nature of the customization (shortcut, simplification, mashup, etc.);
- generic customizations (for any web site) vs. site-specific ones;
- customizations for one-shot tasks vs. repeated use;

---

13. developer.mozilla.org/en/Extensions
14. greasemonkey.mozdev.org
15. q.v. Chickenfoot chapter in this book
16. q.v. CoScripter chapter
17. addons.mozilla.org
18. userscripts.org
19. uid.csail.mit.edu/chickenfoot/scripts
20. coscripter.research.ibm.com
21. www.opera.com/browser/tutorials/userjs
22. msdn.microsoft.com/en-us/library/bb250436.aspx

- the creator of the customization (not necessarily a user of the site);
- the relationship between the customization and the targeted site (sometimes breaking the rules, sometimes collaborative).

**Kinds of web sites customized.** Many scripts target web sites for personal information management (PIM), such as email, calendars, and todo lists, probably because users of these sites spend much time using them. GMail alone has spawned a rich community of customizers. One popular Firefox extension, Better GMail, bundles up several Greasemonkey scripts that change how GMail works, among them keyboard macros, saved searches, and forcing use of a secure connection. Folders4GMail is a Greasemonkey script that allows GMail's flat labels to be organized into a hierarchy of folders. Google Account Multi-Login speeds up switching between different GMail accounts with different usernames and passwords.

Other customizations target media sites, particularly for video and photo sharing. Since YouTube is widely used for posting and watching music videos, YouTube Lyrics adds a box to video pages that searches for and displays the lyrics to the song. Other scripts and extensions support downloading videos from YouTube, and combining IMDb (a movie database) with Bittorrent (a popular technology for downloading movies online). Better Flickr bundles up several scripts that enhance the usability of the Flickr photo-sharing site, including a photo magnifier, shortcuts for replying to other users' comments, and a rich text editor for comments. GMiF (embeds Google Map in Flickr page)

A third major category of customized sites are search engines. The GooglePreview extension inserts webpage thumbnails into Google and Yahoo search results, and SurfCanyon reranks search results (Google, Yahoo, CraigsList, LexisNexis) and filters out undesirable sites.

PIM, media, and search may be important areas of customization for two reasons: first, because users' interaction with these sites may be highly idiosyncratic and personalized, and second, because the developers of these sites sometimes have their hands tied for legal or practical reasons, preventing them from implementing features that end users may demand. Customization is thus forced to pick up the slack. This idea will be elaborated in more detail later in the chapter.

Customization is certainly not limited to sites in these areas, however, and it seems likely that the tail of the distribution is long[23].

**Kind of customization.** Many scripts are *shortcuts*, making a frequent or repeated task in the site more efficient. A classic example of a shortcut is GMail Delete button, a Greasemonkey script that added a Delete button to GMail's interface. Deleting a message was already possible in GMail by opening the More Options pulldown menu and selecting Trash This Message. This script simply makes the function accessible with one click. Another simple example in this category is a script that changes links to GMail and Google Calendar so that they simply switch to an existing GMail or Google Calendar tab rather than opening a new one (initially prototyped in Chickenfoot, and subsequently a popular Firefox extension). Other notable Chickenfoot examples include a script that add a recently-viewed-pages box to MediaWiki, and one that forces all Youtube videos viewed into high-quality mode whenever available. Many CoScripter scripts surveyed by their developers also fall into this category. Since CoScripter was widely deployed and used within IBM

23. D. F. Huynh, R. C. Miller, and D. Karger. "Exhibit: Lightweight Structured Data Publishing." WWW 2007, pp. 737-746.

before being made public, Bogart *et al.* found that many popular shortcuts automate internal IBM systems, such as voicemail, telephony, and business processes.

A particular kind of shortcut is *automatic form fill-in*, which fills in a form with defaults. Major web browsers already provide this ability for login forms, including Firefox, but generally don't have good support for users who must manage multiple accounts per site. The Google Account Multi-Login mentioned previously is one script for this problem. Another is a Chickenfoot script for the Mailman mailing list system, which automatically selects the right password to use for administering a mailing list. For general form fill-in, several Firefox extensions exist, the most widely-installed probably being Google Toolbar. Many CoScripter scripts and Chickenfoot scripts also do form fill-in for specific sites and purposes, under more control by the user than a generic extension.

Another category of customization is *simplification*, which reduce clutter or distraction or eliminate unnecessary features. One Chickenfoot script simplifies iGoogle (Google's home page portal) by removing the logo and search box, which otherwise occupy half the screen. Another script used by one of the authors on GMail hides the fact that there are unread messages in the Drafts folder (turning off the boldface and removing the unread-message count). For general web browsing, two very popular Firefox extensions are Adblock Plus and Flashblock, which remove advertisements and Flash objects from web pages.

Another category is *mashups*, which combine two or more web sites or services together. The YouTube Lyrics script mentioned previously is a mashup between YouTube and a number of lyrics search engines. The GMiF extension embeds a Google Map into a Flickr page to show where the photo was taken (assuming geographical metadata is found in the photo). The Delegate to Remember the Milk extension allows a GMail message to be forwarded to the Remember the Milk to-do-list site with a single click. Chickenfoot mashups include a script that posts the currently-viewed item on Amazon to a Mag.nolia wishlist and a price comparison script for the online used textbook exchange, mit412.com, and Barnes & Noble. Finally, Chickenwire, a Chickenfoot-based mashup created by two of the authors, supports connecting YouTube and Wikipedia together, so that Wikipedia pages about songs or movies can be linked to YouTube music videos or movie clips.

**Generic vs. site-specific customization.** The examples so far have targeted specific web sites, such as GMail, YouTube, Flickr, and Wikipedia, or specific web-based systems, such as MediaWiki and Mailman. But other customizations are designed to change the overall experience of browsing the web. Generic simplifiers like Adblock Plus, and generic form fill-in tools like Google Toolbar, certainly fall into this category. A generic shortcut is the Interclue extension, which pops up a preview of a hyperlink's destination when the mouse hovers over the link.

Other generic scripts focus on reading on the web. A vocabulary-builder script for Chickenfoot highlights words from a vocabulary list wherever they happen to be found in web pages, and pops up definitions on mouseoever, so that students can learn words in context. The Lookitup Greasemonkey script can pop up the dictionary definition of any selected word. For foreign languages, the Globefish extension helps a reader by translating selected text, and helps a writer with idioms and awkward phrasing by measuring the popularity of a phrase with web searches. Finally, the Froggy extension developed by one of the authors reduces distraction and splits paragraphs into sentences, to help nonnative English readers increase comprehension of the text.

Other scripts help with text entry. The Virtual Keyboard Interface Greasemonkey script displays an onscreen keyboard under a textbox, for entering special characters, avoiding keyloggers, and accessibility. Chickenfoot scripts allow any textbox to be resized and add commands for joining paragraphs (removing hard linebreaks) and splitting them (inserting hard linebreaks).

One more kind of generic script concatenates a series of web pages together, such as a multipage article or a list of search results, using a table of contents or Next link to discover the subsequent pages. An early Chickenfoot example did this, as do the Greasemonkey scripts GoogleAutoPager and AutoPagerizer.

Since generic customizations essentially add functionality to the overall browser experience, they raise questions about whether the browser itself should evolve to include these features, and if not, why not. Similar questions arise about site-specific customizations; why doesn't the site itself implement these features, particularly when a customization has proved popular? Sometimes the site eventually does, but sometimes it can't, for legal or practical reasons.

**One-shot vs. repeated use.** Most of these examples have been designed for repeated use over time by the same user, and have little value if used only once. Another kind of script is intended for a *one-shot task* that may never be needed by the user again. One-shot scripts generally use automation or scraping to do a large task. These scripts are worth creating when the size of the task is large enough that the effort put into writing the script pays off in reduced manual effort[24].

An example of a one-shot task is a Chickenfoot script that clears the bounce flag on all subscribers to a Mailman mailing list. Mailman's interface makes this task tedious to do manually, first because subscribers are divided up into pages by initial letter (A, B, C, ...), forcing the user to visit 26 or more pages for a large mailing list; and second, because each bounce flag is a checkbox next to the subscriber's email address, with no option to select or deselect all. The Chickenfoot script automates stepping through the pages, clearing all the checkboxes.

Other examples of one-shot automation have included clean-up tasks for a wiki; transferring course grades from a college information system to a grad school application; scraping school tuition data into a spreadsheet; and calculating scores for a school contest. These examples were Chickenfoot scripts written by one or more of the authors. One-shot automation is hard to find in public repositories, since it rarely seems worth publishing. Also, the programming system used must have a low threshold to make it practical to create the one-shot script in the first place. One-shot Firefox extensions seem highly unlikely, because building a Firefox extension is serious investment. Chickenfoot and CoScripter are easier to use for that purpose.

**Creator of the customization.** Most customizations are created by *users* of the web sites they target. Granted, these creators may have more programming skills than the average user. Certainly this is true for Firefox extensions, Greasemonkey, and Chickenfoot, since those systems require knowledge of Javascript. CoScripter draws in a broader swath of end-users, since it has no such requirement. Whether the customizer has programming skills or not, however, the *role* they play in the customized system is generally an end-user.

But the customizer is not always an end user. For example, several popular Firefox extensions have been published by web sites themselves, including Google Toolbar, Yahoo Toolbar, eBay Sidebar, and RememberTheMilk for GMail. It may be hard to call these customizations. In a sense they extend the web site deeper into the browser, to run code

24. R. Potter, "Just-in-time programming." In A. Cypher, ed., *Watch What I Do: Programming by Demonstration*, MIT Press, 1993.

with stronger permissions and more functionality (e.g., storing data locally or mashing up data with other sites), or to provide a more persistent, browser-level interface that follows the user around while they browse elsewhere on the web.

Another situation arises when the customizer is not just an end-user but also the *owner* of the site. Open-source systems like Mailman and MediaWiki may be locally installed in an organization, and the source code for the system may be technically available to be changed, and yet web customization may still be a more viable route. The authors have written many Chickenfoot scripts for systems that their organization owns and controls, including the Mailman bounce-flag script, a script for MediaWiki editing that provides a save-and-continue-editing command, and several scripts for the Flyspray bug database that provide simplification and shortcuts[25]. Many CoScripter scripts created for internal IBM use may also fall in this category, since IBM owns and controls many of the systems that IBM's own employees find it necessary to customize.

Finally, the customizer may be neither an end-user nor a developer, but a researcher exploring new ideas in web user interfaces. The four customization systems surveyed here have proven to be fertile ground for research prototypes, many of which are described in this book. Firefox extensions created for research purposes include Web Summaries[26], Zoetrope[27], Transcendence[28], and MashMaker[29]. IE Browser Helper Objects were used by Creo[30]. Notable research uses of Greasemonkey include Accessmonkey[31], PrintMonkey[32], Spartag.us[33], Kalpana[34], and table browsing for small screens[35]. In our own group, Chickenfoot has been used in Kangaroo[36], Smart Bookmarks[37], and Inky[38]. CoScripter has contributed to Highlight[39] and Vegemite[40].

**Is it an arms race?** A final consideration is the relationship between the customization and the targeted web site. Some customizations bend the rules of a site. In a survey of public CoScripter scripts, Bogart *et al* found that 18% of scripts sampled were designed to circumvent assumptions made by the site that a human user was clicking on a button or hyperlink. One example is Automated Click for Charity, a script that visits charity web sites

25. q.v. Chickenfoot chapter in this book, section "Making a Bug Tracker More Dynamic".
26. q.v. Web Summaries chapter
27. q.v. Zoetrope chapter
28. J.P. Bigham, A.C. Cavender, R.S. Kaminsky, C.M. Prince, and T.S. Robison. "Transcendence: enabling a personal view of the deep web." IUI 2008.
29. q.v. MashMaker chapter
30. q.v. Creo chapter
31. J.P. Bigham and R.E. Ladner. "Accessmonkey: a collaborative scripting framework for web users and developers." W4A 2007.
32. J. Baldwin, J.A. Rowson, Y. Coady. "PrintMonkey: giving users a grip on printing the web." DocEng '08.
33. L. Hong, E.H. Chi, R. Budiu, P. Pirolli, L. Nelson. "SparTag.us: a low cost tagging system for foraging of web content." AVI '08.
34. A. Ankolekar, D. Vrandecic. "Kalpana - enabling client-side web personalization." Hypertext '08.
35. K. Tajima, K. Ohnishi. "Browsing large HTML tables on small screens." UIST '08.
36. q.v. "Adding Faces to Webmail" section in Chickenfoot chapter
37. D. Hupp and R.C. Miller. "Smart Bookmarks: Automatic Retroactive Macro Recording on the Web." UIST 2007, pp. 81-90.
38. q.v. "Sloppy Programming" chapter
39. q.v. Highlight chapter
40. J. Lin, J. Wong, J. Nichols, A. Cypher, T. Lau. "End-user programming of mashups with Vegemite." IUI '09.

and clicks on links to trigger donations without viewing the ads that pay for those donations. Other examples include ballot stuffing for online polls and automatic players for online lotteries or multiplayer games.

Many web sites have Terms of Service agreements for their users, and these agreements may forbid certain kinds of automation. Sites may defend against customizations by technical means (such as CAPTCHA tests that distinguish humans from scripts) or legal means. One very recent case concerned a Firefox extension that injects a button into Amazon music and video pages, linking to downloadable content on the illicit file-sharing site The Pirate Bay. The site hosting the extension was reportedly taken down by a cease-and-desist order from Amazon[41].

Conversely, sites can welcome and support customizations. GMail is noteworthy in this regard, providing the GMail API for Greasemonkey to make Greasemonkey scripts easier to write and more robust to future changes.

Sites can also pay attention to what customizers are saying with their customizations, and respond with improvements. GMail is notable here too. The GMail Delete button mentioned as our first example has become obsolete, because GMail listened to its users and now includes a Delete button.

## Why We Customize

The basic reason users make customizations is that users want applications that were written for "just anybody" to be optimized for their work habits and preferences. On the web, this means that users want individual sites to have the features they need, they want sites to be able to work together for certain tasks, and they want to make improvements to their browsing in general - customizations that many involve all sites that they visit. In addition, web sites tend to be less developed than most desktop software. Some sites get launched before they are fully functional, and some sites don't get regularly updated and have interfaces that could be improved from the users' standpoint.

From surveying existing web customizations, we can that some of the prevalent reasons Mackay found for users making desktop software customizations in her 1991 paper are still true on the web today.

Mackay identified one of the top reasons to customize being that the user noticed themselves doing something repetitive. Many customizations on the web have been written to eliminate redundant actions. For users that are constantly clicking away ads, there are ad blocking customizations. For users who spend time looking up uncommon words, there are customizations that add definitions for certain words in a tool tip. For users who commonly search for icons, there is a customization that in the click of one button on Google that will search for small (icon-sized) images. For users who repetitively look at the enlarged product images while shopping online, there are customizations to load all the full sizes images automatically.

Additionally, Mackay reported that feeling annoyed was a reason for customization. The web has seen customizations to work around the annoyance of not being able to bookmark pages such as airline fare results. There are customizations to reduce the saliency of ads on

---

41. D. Kravets. "Amazon.com Tossed Into Pirate Bay Jungle," December 4,2 008. blog.wired.com/27bstroke6/2008/12/amazoncom-tosse.html

a page to make reading the content less annoying. For users who find it annoying that Ctrl+S doesn't "Save-and-Continue" in MediaWiki edit pages, there is a customization to bind the shortcut to that action on all MediaWiki edit pages.

Many repetitive tasks span multiple pages: shopping comparisons, looking up words in an online dictionary, and looking up airfare for example. Web customizations can navigate through a sequence of pages or mash up multiple web sites, while it is uncommon for desktop customizations to use more than one application. GooglePreview inserts web page thumbnails in Google and Yahoo search results. SmartBookmarks automates the process of entering data and clicking through pages within a site to get to dynamically generated price results such as airfare.

Some of the customizations we have seen are changes to individual sites that fix general usability problems, and the sites could fix (and in some cases have fixed) these problems eventually. Users who make these changes don't want to wait, they want to stay ahead of the curve. This includes simple changes like moving the login interface to the top of the page where it's more useful, and adding a delete button to GMail, which is a change GMail eventually made.

One way in which web customization is unique is that for some sites, customization is the only way to get certain features because the site will never provide them. Legal restrictions prevent some sites from adding features and some commercial sites have an dis-incentive to provide certain features. Scaffidi *et al.* refer to customizations who circumvent the intensions of a site as "changing the rules." Legal restrictions prevent Wikipedia from housing copyrighted content; one customization keeps it's own list of links to YouTube videos relevant to each Wikipedia page and includes those videos everytime the user visits a Wikipedia page. YouTube doesn't allow users to download its video content largely for legal reasons, but there are several customizations that make it trivial to download the videos as they stream. There are customizations to recommend torrent sites for particular movies on The Internet Movie Database (IMDB). Among the CoScripter customizations designed to "change the rules," the most devious may be a script to automate voting on a poll whose winner receives money. Other "changing the rules" customizations include injecting one shopping site with price comparisons to other sites, and scrubbing ads out of applications such as GMail or sponsored search results in search engines.

In summary, users customize the web for some of the same reasons users usually customize: to better suit their needs by making tasks less repetitive or aspects of browsing less annoying. On the web, some of these problems arise because tasks span multiple sites and some of them arise because web sites can be underdeveloped and have usability problems or missing functionality. Lastly, an important reason to customize the web is to "bend the rules" - circumvent the legal restrictions or the intentions of the site.

## Conclusion and Future Directions

This chapter has shown a variety of motivations for customization in the web browser. Customizations were fundamentally enabled by both the inherent openness of web technology and the particular decision by Mozilla developers to make Firefox highly extensible. The threshold of customization was lowered still further by Greasemonkey, Chickenfoot, and CoScripter, all of which were built on top of Firefox's extension mechanism.

Although one can expect the world of web customization to continue to grow in size and richness, challenges remain. One continuing problem is robustness. Web sites change over time, in appearance and organization and internal structure. Customizations that target a changing web site will decay unless maintained and updated. If the web site commits to supporting customizations, as GMail does through its Greasemonkey API, then this problem can be mitigated, but it seems unlikely that more than a tiny fraction of web sites will ever do this. Customization systems that target the rendered user interface of the site (as Chickenfoot and CoScripter do) may prove more robust to change than those that operate on internal HTML/Javascript interfaces (like Greasemonkey), but this remains to be proven.

Another perennial challenge comes from programming platforms that run inside the browser but do not provide the same degree of reflection and modification as HTML and Javascript. Java applets are probably the oldest example, but Adobe Flash/Flex and Microsoft Silverlight are growing in popularity. Currently these platforms are used more for embedding components (like video players or ads) in a mostly-HTML site, rather than implementing the entire site, but as the trend toward richer interactive experience continues, site developers may find the new platforms very enticing. A site whose entire functionality is provided by Flash or Silverlight can't be customized using the systems described in this chapter. New approaches may be required, lest we slip back toward the closed and less customizable world of the desktop.

We have focused on browser-hosted customizations in this chapter, but a serious limitation of this approach is that the user's customizations don't easily move with them as they use different browsers on different computers. CoScripter has an advantage here, because it stores all scripts on a wiki so that any Firefox browser with CoScripter installed can access them. Mozilla Weave[42] is an effort to solve this problem for Firefox extensions and preferences in general.

Finally, although we made a distinction in this chapter between "desktop" and "web," in fact the web platform we considered might better be called the "web desktop" -- the web as seen by a conventional web browser like Mozilla Firefox running on a conventional desktop or laptop computer with a big screen, keyboard, and pointing device. The future of the web platform is much more diverse. Web browsers will turn up in a variety of different devices and contexts, including cell phones, netbooks, TV set-top boxes, home media servers, and wall displays. Mobile web customization is already an active area of research, as demonstrated by Creo[43] and Highlight[44], but much work remains to be done to give users the power to customize the web of the future.

42. labs.mozilla.com/projects/weave/
43. q.v. Creo chapter
44. q.v. Highlight chapter