input locally, and then communicates with the underlying servers as needed. The JavaScript code uses a standard interface, the *Document Object Model*, to navigate and modify the web page's contents, and further standards dictate the page's visual appearance. All modern web browsers implement these standards to some degree.

Although a web browser is not a text editor, there are some striking resemblances between Emacs's architecture and that of a browser:

- Although Emacs Lisp and JavaScript don't resemble each other much at the syntactic level, their semantics have many essential traits in common: like Emacs Lisp, JavaScript is interpreted, highly dynamic, and safe. Both are garbage-collected languages.
- As with Emacs Lisp, it's very practical to begin with a small fragment of JavaScript on a page to improve some minor aspect of its behavior, and then grow that incrementally into something more sophisticated. The barrier to entry is low, but the language scales up to larger problems.
- As in Emacs, display management is automatic. JavaScript code simply edits the tree of nodes representing the web page, and the browser takes care of bringing the display up to date as needed.
- As in Emacs, the process of dispatching input events to JavaScript code is managed by the browser. Firefox takes care of deciding which element of the web page an event was directed at, finds an appropriate handler, and invokes it.

However, Firefox takes the ideas behind these modern web applications a bit further: Firefox's own user interface is implemented using the same underlying code that displays web pages and handles their interactions. A set of packages known as *chrome* describe the interface's structure and style, and include JavaScript code to bring it to life.† This architecture allows third-party developers to write *add-ons* that extend Firefox's user interface with new chrome packages. Taking the same techniques even further, developers can replace the standard Firefox chrome altogether and radically reshape the entire user interface—to adapt it for use on mobile devices, for example.

Like Eclipse plug-ins, Firefox chrome packages include a significant amount of metadata. And, resembling Eclipse's Plug-in Development Environment plug-in, there is a Firefox extension to help people write Firefox extensions. So there is still a significant amount of work required up front before one can extend or fix Firefox. However, Firefox's automatic display management and simplified event handling mean that the effort required is still not as high as that needed to write an Eclipse plug-in.

† Naturally, JavaScript code used in chrome can read and write preference files, bookmark tables, and ordinary user files—privileges that would never be granted to code downloaded from a web page.