fundamentally on this syntactic uniformity; the many attempts that have been made to leave it behind generally haven't fared well. With experience, many Lisp programmers come to feel that the syntax is a reasonable price to pay for the language's power and flexibility.

In Emacs, a command is simply an Emacs Lisp function that the programmer has marked (with a brief annotation at the top of the definition) as suitable for interactive use. The name of the command that the user types after Meta+x is the name of the Lisp function. *Keymaps* bind key sequences, mouse clicks, and menu selections to commands. Core Emacs code takes care of looking up the user's keys and mouse gestures in the appropriate keymaps and dispatching control to the indicated command. Along with the automatic display management provided by Emacs's View, the keymap dispatching process means that Lisp code is almost never responsible for handling events from an event loop, a welcome relief for many user interface programmers.

Emacs and its Lisp have some critical characteristics:

- Emacs Lisp is light on bureaucracy. Small customizations and extensions are easy: one-line expressions placed in the *.emacs* file in your home directory, which Emacs loads automatically when it starts, can load existing packages of Lisp code, set variables that affect Emacs's behavior, redefine key sequences, and so on. You can write a useful command in under a dozen lines, including its online documentation. (See the "The Five-Minute Lisp Tutorial" sidebar for the complete definition of a useful Emacs command.)
- Emacs Lisp is interactive. You can enter definitions and expressions into an Emacs buffer and evaluate them immediately. Revising a definition and reevaluating it puts your changes in place; there is no need to recompile or restart Emacs. Emacs is, in effect, an integrated development environment for Emacs Lisp programs.
- The Emacs Lisp programs you write yourself are first-class citizens. Every nontrivial editing feature of Emacs itself is written in Lisp, so all the primitive functions and libraries Emacs itself needs are equally available for use in your own Lisp code. Buffers, windows, and other editing-related objects appear as ordinary values in Lisp code. You can pass them to and return them from functions, store them in data structures, and so on. Although Emacs's Model and View components are hardcoded, Emacs arrogates no special privileges to itself within the Controller.
- Emacs Lisp is a full programming language. It is comfortable to write reasonably large programs (hundreds of thousands of lines) in Emacs Lisp.
- Emacs Lisp is safe. Buggy Lisp code may cause an error to be reported, but it can't crash Emacs. The damage a bug can do is contained in other ways as well: for example, buffers' built-in undo logging lets you revert many unintended effects. This makes Lisp development more pleasant and encourages experimentation.