Foreword

As the Hypertext Markup Language / Extensible Markup Language / Cascading Style Sheets (HTML/XML/CSS) parsing and rendering engine for the Firefox, Mozilla, and Netscape browsers, Gecko is one of the most widely deployed cross-platform rendering engines in the world.

As both a Netscape engineer and later as the development manager of the Mozilla Gecko team, I had the privilege to work on the Gecko engine from its inception.

Gecko was born out of the desire to create a cross-platform, small-footprint, fast, state-of-the-art, embeddable Web browsing engine that would leapfrog our competition in the "browser wars." It had become painfully apparent that it was too difficult to add full CSS2, CSS3, and XML Web standards support to the lumbering Netscape 4.x engine. The idea was to start from scratch using only a few libraries from the original engine. Early in the Gecko project, there were discussions about using Java rather than C++ to leverage Java's cross-platform capabilities. It was ultimately decided that C++, along with special development processes, tools, and design techniques, would yield the best solution. Many of those processes, tools, and design techniques are described as best practices throughout this book.

Before coming to Netscape, I worked at several companies that produce cross-platform software. However, the Mozilla project took it to a whole other level. We utilized and developed software architectures, tools, and processes that enabled cross-platform development on a wide scale.

My first task was to port Gecko from Microsoft Windows to Motif/Xlib. As anyone who has written cross-platform software knows, the early ports are the most challenging. That's where you discover just how portable your software really is. Even though Gecko was designed from the beginning to be portable, small differences between platforms and compilers

came back to bite us. This is why it's so important to have a continuous build system such as the Mozilla Tinderbox that verifies every source code check-in for portability and a software development process that requires engineers to verify their new code on at least two platforms before they check into the source code repository.

As the Gecko engine development gained momentum, the decision was made to re-create the Netscape Communicator user interface experience on top of it. This would require a cross-platform user interface solution because Netscape Communicator worked on multiple platforms with diverse graphical user interface environments. I was given the opportunity to develop a strategy to solve the thorny cross-platform user interface problem. I authored a document that described how to achieve a cross-platform user interface by combining an XML meta description of the user interface elements and JavaScript as control/event logic within the Gecko rendering engine. This document was the seed document for what later became the XUL (XML User Interface Language) language. Later, the Firefox developers took advantage of XUL and the Gecko engine to build a small, fast, cross-platform, Web browser that has become wildly popular. Chapter 9 describes how you can also build on top of XUL to create your own cross-platform user interfaces.

As an original member of the W3C SVG (Scalable Vector Graphics) working group, I am particularly excited to see that Gecko continues to evolve and solve additional cross-platform challenges. The recent addition of SVG native support marks yet another chapter in Gecko's portability success.

The information Syd Logan is presenting here is the collective insight of the myriad engineers who ironed out the special problems associated with creating cross-platform production software. Although it is written with C++ in mind, many of the techniques can be adapted to non-C++ software projects, too. I hope you will be able to use some of the tools, techniques, and processes described in these pages to avoid some of the pitfalls of cross-platform development and make your project a huge success.

—Kevin McCluskey