
Freeing the Source : The Story of Mozilla

Jim Hamerly and Tom Paquin with Susan Walton

On January 23, 1998, Netscape made two announcements. The first, as reported by C|Net: "In an unprecedented move, Netscape Communications will give away its Navigator browser, confirming rumors over the last several weeks."

The second: "It also will give away the source code for the next generation of its Communicator suite."

The decision to give away the browser came as no surprise, but the release of the source code stunned the industry. It hit the pages of newspapers around the world, and even the Open Source community was surprised at the move. Never before had a major software company opened up its proprietary code. What was Netscape up to now?

We had decided to change the playing field, and not for the first time. Always known for thinking outside the box, this time Netscape was taking the commitment to building a better Internet to a new level. When Netscape initiated unrestricted distribution of early versions of its browser over the Internet in 1994, people said "That's crazy!" When Netscape said "Free Source Code" they said the same thing.

The discussion period leading up to the Open Source announcement moved like a runaway train. After months of deliberation about whether or not to release the binary for free, critical mass was reached in the decision to free the source in an unbelievable twenty-four hours.

As fast and surprising as the announcement seemed to both insiders and outsiders, it reflected several converging tracks of thought. Netscape executives were discussing a whitepaper by Frank Hecker that expressed a view coming to the forefront. In it he advocated that Netscape free its source. Frank had done his homework, citing Eric Raymond's paper, "The Cathedral and the Bazaar," and talking to people in departments throughout the organization--from engineering to marketing to management. In a twenty-page opus that was widely circulated, he pled the case that was gaining momentum:

When Netscape first made Navigator available for unrestricted download over the Internet, many saw this as flying in the face of conventional wisdom for the commercial software business, and questioned how we could possibly make money "giving our software away." Now of course this strategy is seen in retrospect as a successful innovation that was a key factor in Netscape's rapid growth, and rare is the software company today that does not emulate our strategy in one way or another. Among other things, this provokes the following question: What if we were to repeat this scenario, only this time with source code?

In the engineering pit there was a similar view. Many Netscape employees had experience working with Open Source. And since Communicator's code was so tightly integrated with Java and HTML, most recognized an emerging truth: It wasn't such a huge jump to

make. The nature of Java invites a more open view of source distribution. Because it is cross-platform and can be compiled down to class files that are machine-independent executables, each binary is like a virtual machine. One effect of this is that programmers can decompile the executable and turn it back into source code. And the browser "view source" command made HTML a common vernacular. Rather than trying to block this, many believed Netscape should facilitate it, encourage it, and if possible, benefit from it.

The various grassroots schools of thoughts merged with unexpected suddenness. In meetings, reaction to the suggestion went from stunned shock to nods in minutes. Most of the discussions passed quickly from "should we?" to "when?" Most of the key people believed that we had to move fast, set a firm date, and make it happen. In January, Netscape made a promise to the Net: Communicator source will be released in the first calendar quarter of 1998. Netscape took this promise with deadly seriousness, and Project Source 331 came into being. This was the name for Netscape's all-out effort to have the source code out by March 31, 1998.

Then the reality set in.

Making It Happen

The body of Communicator source code at Netscape was called "Mozilla." Mozilla was a term initially created by Jamie Zawinsky and company during the development of Navigator. The team was working at a similarly frantic pace to create a beast vastly more powerful than Mosaic, and the word became the official code name for Navigator. Later the big green dinosaur became an inside joke, then a company mascot, and finally a public symbol. Now the name came into use as the generic term referring to the open-source web browsers derived from the source code of Netscape Navigator. The move was on to "Free the Lizard."

There was an amazing amount to be done to make the code ready for prime time. As issues surfaced, they separated themselves into categories and were claimed. The next three months were devoted to resolving issues at the fanatical pace that Netscapers knew well.

One of the largest issues was the disposition of the third-party modules included in the browser. Communicator contained over seventy-five third-party modules in its source, and all of the code owners needed to be approached. Teams of engineers and evangelists were organized to visit and sell each company on the concept of joining Netscape on the road to Open Source. All of them had heard Netscape's Open Source announcement, and now each company had a choice to make: their code could be removed or replaced, shipped as binary (kept in its compiled state), or shipped as source code along with Communicator. To complicate matters, many of the third-party contracts were unique and ran for different lengths of time. No one scenario would be appropriate as a solution for all situations.

Making the deadline for Project Source 331 was considered essential. And that required tough choices. This was surely the case when it came to the participation of the third-party developers. The rule was either you're in by February 24th, or your element will have to be scrubbed from the source. Those kinds of deadlines are not hard to set early on, but they became brutal when we hit the wall. When the time came, some code had to be removed.

Java was a proprietary language, so it had to be removed. Three engineers were assigned to perform a "Java-ectomy." The browser had to build, compile, and run--without Java. Since the overall code was so tightly integrated with Java, this was no small feat. The goal was to have the source code ready by March 15th so that the final two weeks could be devoted to testing. Engineers had to disentangle all Java code from the browser in an inconceivably short time.

Cleansing the code was a huge project. Early on, many felt it just couldn't be done in time for the deadline. But as steam gathered at meetings, strategies formed. The wheels began to turn. The Product Team dropped their entire workload (most were developing the next generation of the browser) and everyone got down to the business of surgery. Not only did the inclusion (or excision) of each third-party participant have to be resolved, all comments had to be edited from the code. Responsibility for each module was assigned to a team and they went in to scrub.

One of the great innovations that happened early on was the decision to use the Intranet bug-reporting system as a task manager. "Bugsplat" was the name for Scopus, a bug-reporting program fronted with an HTML interface. It was ideal as a workflow management system. New jobs were reported to the system as they came up, input in a simple HTML form. Just as with a bug that has been reported to the system, priorities were set, relevant participants were determined, and mailing lists grew up around each task. When the task (or bug) was resolved, all of the mailing lists and prioritization collapsed and disappeared from view. Engineers were able to track the progress of their modules and watch the project unfold by logging on to the Intranet.

The removal of the cryptographic modules was another tremendous task for the engineering team. Not only did the government insist that all cryptographic support had to be removed, but every hook that called it had to be redacted. One team's sole job was to keep in constant contact with the NSA and manage compliance issues.

Creating the License

Parallel to the Great Code Cleanup was the license effort. The first step was to resolve the big question: Would any of the existing licenses work with the open code? No one wanted to have to draft new licenses, but everyone realized it might be necessary to accommodate all of the third-party code and to make the project work on a corporate level. No existing proprietary software had ever been released under a free source license.

A group of Open Source community leaders, including Linus Torvalds, Eric Raymond, and Tim O'Reilly, were invited to visit the Mountain View campus. They spoke with audiences of executives, attorneys, and programmers about what they were in for, and met with small groups to talk about some of the issues they were likely to face. They spent a great deal of time with the Netscape legal team discussing the existing licenses--both their strengths and the problems they created. These advisors also acted as a sounding board for ideas.

One team dove into researching existing licensing agreements with the advice and guidance of the Netscape legal team, trying to determine whether one of them would work for Mozilla. Beginning with the GNU General Public License, the GNU Library General Public License (LGPL),

and the BSD license, we took long looks to outline exactly what problems they solved and created. Unlike the code to which these agreements had been applied in the past, Netscape's existing code base presented unique circumstances. One of the thorniest issues was the private licensing agreements that governed many of the third-party components used in the code. The license needed to create an environment where these and other new commercial developers could contribute their code to Mozilla while protecting their business interests.

The more permissive BSD license, which only requires that the copyright holder be referenced in unlimited changes to code, was deemed insufficient for Mozilla development. It would leave developers open to the risk that modifications to their work would not be returned to them or to the rest of the community. This point alone was a big issue, since it was crucial to the long-term viability of open source development efforts.

On the other hand, the requirements of the GPL made it undesirable in this project. The GPL is "viral"; when applied to an original piece of code, any other code with which the original is compiled must also be covered under the GPL. This aspect made it untenable for commercial software developers. For instance, the GPL would require that third-party components compiled into branded versions of Communicator also be released under the GPL, something outside of Netscape's reach, as Netscape does not control these third parties. And Netscape itself uses a portion of the Communicator code in its other products (such as servers). Since Netscape has no immediate plans to release that source code, the GPL viral effect on these products would present the same problem for Netscape as for other companies. The more open and less restrictive LGPL, a modification of the GPL, came closest to meeting Netscape's needs for use with commercial development, but it still contained too many of the same commercial pitfalls as the GPL.

After a frenzied month of research, discussion, meetings with experts and advocates from the free software community, and amidst public speculation, the team decided that a new license had to be crafted for this unique situation. The Netscape Public License (NPL) broke new ground in attempting to strike a compromise between promoting free source development by commercial enterprises and protecting free source developers. The process of fashioning a next-generation Open Source license took over a month.

In another extraordinary move, when the first draft of the Netscape Public License (NPL) was complete it was beta-tested publicly. On March 5, a draft was posted in a new newsgroup called `netscape.public.mozilla.license`, and a request was made for public comment. It was met with cheers and jeers. One section of the license acted as a lightning rod, catching most of the flames: the portion of the NPL that granted Netscape special rights to use code covered by the NPL in other Netscape products without those products falling under the NPL. It also allowed Netscape to issue revised versions of the NPL, and most controversially, to re-license code covered by the NPL to third parties under terms different from the NPL. Some of the people providing feedback went so far as to suggest that this fact alone would make the NPL unacceptable to the Open Source development community.

On March 11th, a status report appeared on

netscape.public.mozilla.license from jwz (Jamie Zawinsky). It read, in part:

First of all, THANK YOU for the incredible amount of cogent feedback you've been giving! It has been incredibly helpful, and rest assured, the opinions expressed here are being taken very seriously.

Next week, you can expect to see a dramatically reworked section 5. I probably shouldn't comment on it too much (wouldn't want to set expectations incorrectly) but the message that most of you hate it as it stands now has been received loud and clear.

On March 21st, the revision was posted. This was unprecedented. The reaction was incredulous: "I told them it was awful and they listened! I can't believe it!" People realized that this was a true open-source project, in spite of its unlikely birthplace. The discussions going on in the newsgroups were helping to guide the process, rather than providing commentary on its results. The continuing discussions took on a new tone and spirits were high.

The community criticism of the beta of the NPL had sent the license team back to the drawing board. They sought a solution that would allow Netscape to balance the goals of engaging free source developers while continuing to meet business objectives. The result was the release of a second license to work with the NPL, the Mozilla Public License (MozPL). The two licenses are identical, except that the NPL includes amendments granting Netscape additional rights.

All of the code initially issued on March 31, 1998 was released under the NPL, and all modifications to that code must be released under the NPL. New code developed can be released under the MozPL or any other compatible license. Changes to files contained in the source code are considered modifications and are covered by the NPL. And to resolve much of the concern expressed on the Net: new files that do not contain any of the original code or subsequent modified code are not considered modifications and are not covered by the NPL. This resulting code can be covered by any compatible license. The GPL is not compatible with the Netscape Public License or the Mozilla Public License. The GPL is by design incompatible with all other licenses, since it prohibits the addition of any restrictions or further permissions to its boundaries. All code developed to work with GPL software must in turn be covered by the GPL. Another minor point is that the GPL insists that when you distribute code covered under its terms, it must be complete and entire. The NPL does not have this condition.

The discussions on the newsgroups had brought an important issue into sharp relief: developers needed Netscape to allow a distinction between bug fixes and new code. Clearly, it's one thing to say, "I'm making a bug fix, a small modification to your program," and quite another to realize "I'm adding a new feature to your program." They provoke different feelings. Most people feel all right about giving away a bug fix, and the value of making a contribution is its own reward. But new code is a different story. A developer who has done a lot of new work doesn't want to see somebody else use it to make money for themselves.

The NPL and the MozPL were designed to encourage open development on the Mozilla code base, but from the beginning there

was also another goal in mind. Netscape was willing to be the first large corporation to open up its proprietary source, because it wanted to foster wider corporate interest in development in open source environments. Creating an atmosphere that made it possible for large, profit-making organizations to adopt this model and participate in the movement was paramount. The legal infrastructure in most Open Source licensing is a big hurdle to corporate cooperation. With Mozilla, the license was a project unto itself.

By giving away the source code for future versions, we hoped to engage the entire Net community in creating new innovation in the browser market. The idea that there would be talented programmers worldwide hacking on our code, infusing the browser with their creative energy, motivated everyone to keep going even when the going got tough.

Mozilla.org

People who had been involved in open-source projects before realized that the code had to have a place to live. The night after Netscape announced that it would free the source, Jamie registered a new domain name with Internic and drew up a chart on how distributed development projects work. Mozilla.org was born.

There's a pattern that all successful open-source projects follow, not necessarily by design. There tends to be one person or group that does coordination. People work on whatever aspect of the code they care about, scratching their own itches. At the end of the day, they have something that works a little better for them. But what happens a month later when a new version of the software comes out? Their fix is gone, and they're back to square one--or worse, because the software may have changed.

The result is that developers want to get their patch included in the main distribution. And if there's just a pile of source code floating around and a bunch of people working on it, eventually someone will stand up and say, "I might as well collect a bunch of patches and do a release." When the next person comes along wondering how to get his patch into the next release, he'll say, "I don't know who else to give my patch to, so I'll give it to that guy. He seems to be doing a good job of it." And as time goes by, that person becomes the maintainer.

For this open-source project, the horse was put in front of the cart. Mozilla.org was conceived and designed to fill the role of maintainer from the outset. Since the role would be filled one way or another, we decided to create the infrastructure to become the clearinghouse.

In the next months, mozilla.org began to set up an organization, getting funding and machines, posting mailing lists, and developing the underpinnings necessary to make it work. The mission was simply to get the organization off the ground and functioning. It was crucial that there be a central depot in operation as soon as the source code was released. And if we weren't prepared, in six months time, we'd be watching someone else do it. Netscape is not known for sitting around and watching the other guy.

Giving away the source code meant Netscape was collaborating with the Net. And there was a crucial concept that had to be accepted: the Netscape Client Product Development Group and mozilla.org were not the same organization. Mozilla.org's goal is to act as the coordinator for all of the people worldwide working on the software. Product

Development's purpose is to ship products--Netscape products based on the Mozilla code. Since both groups are working on the same product, interests can overlap. But the group behind mozilla.org knew that it would be disastrous for the Net to look at the organization and say, "These people only have Netscape's interests in mind and they're only about shipping Netscape products." This would mean that mozilla.org had failed in its goal to be a good maintainer. The separation had to be real and the Net had to know it.

Behind the Curtain

What happens when a developer makes a change and pipes up, "Hey, mozilla.org, please take this code?" One of mozilla.org's most important roles is to draw lines as to what code is accepted and what is not. We must factor in a number of issues. First and foremost is merit. Is it good? Second, is it under a license that is compatible with NPL? We decided not to accept contributions that were not under a license compatible with NPL. Otherwise there would have to be separate directories, Chinese walls, and lots and lots of legalese for everyone involved. The potential for error goes into the stratosphere.

Since Mozilla is a highly modular code base, each major module, such as the Image Library or the XML Parser, have a designated "owner." That person knows the code best and is the arbiter of what should go in to that module and what shouldn't.

Many module owners are Netscape engineers, but some are coming on board from the Net-at-large. When a module owner makes changes (for example, adding an API to the Image Library) the modifications are sent to mozilla.org for inclusion in distributions. If differences arise between a contributor and the module owner, mozilla.org performs as the arbitrator, making the final call--always aware that if it stops playing fair, it will be ignored and someone else will usurp the duties.

Mozilla.org had to contend with the fact that there would be both internal Netscape developers and people on the Net working on their code. The methods used to work on code internally had to accommodate the Web and be accessible on all platforms, in all time zones. This was done with "tree control" performed by the tools Bonsai and Tinderbox.

"Bonsai" is a tool that lets you perform queries on the contents of an archive. Like the front desk of a library, you can "check in" code you've worked on, or see what "checkins" have been made by others. In the background, it constantly runs the code, checking the code tree. If the tree breaks, it sends up a red flag, stopping further checkins until the problem can be identified. Logs can be pulled and problems traced to a particular time period. Previously used by Netscape developers in-house, it was erected on mozilla.org for use by developers around the world and could be used directly through the browser on any platform.

If you get more than ten developers together without tools, there is going to be an explosion. That's the theory behind "Tinderbox," a program that keeps this potentially explosive situation under control. Tinderbox is a detective tool. It allows you to see what is happening in the source tree. It shows who checked in what (by asking Bonsai), what platforms have built successfully, what platforms are broken, exactly how they are broken, and the state of the files that made up the build so you can track down who may have done the damage.

April Fool's Day, 1998

It was a week and a half before the end of March 1998, and the deadline was closing in fast. There was a general sense that there needed to be a party to celebrate the code release, but nothing had been done about it. In keeping with the rest of this project, the bash would become a groundbreaking event that invited the public into Netscape's world, shields down.

In a meeting Jamie laid out his plan to rent out a nightclub in San Francisco, invite the world, and broadcast it over the Net. "You mean invite non-employees to the party? But we've never done that before!" In character with the rest of the project, after a pause the reaction was . . . "Why not?"

The party will not soon be forgotten. Jamie rented out one of the biggest nightclubs in San Francisco, The Sound Factory, on the night of April 1st. DJs (including Apache founder Brian Behlendorf) gave away thousands of mozilla.org T-shirts, software, and items from NetObjects, Macromedia, Digital, Be, Wired, and unAmerican Activities.

When the doors opened for the "Mozilla Dot Party" at eight, there was already a line. An hour and a half later, the place was filled to its fire-code maximum of two thousand, and the line wrapped around the block. People were being waved in twenty at a time as others departed, and by the end of the night, over 3,500 had passed through the doors, including free software gurus like Brewster Kahle (founder of WAIS) and Eric Raymond. Hundreds more synched their watches and toasted Mozilla around the world. The virtual partygoers included a group of over a hundred at The Waag castle in Amsterdam, The Netherlands, and various individual groups in Norway, Montreal, Canada, Pennsylvania, North Carolina, Wisconsin, Colorado, and Alabama.

Inside, three projection screens scrolled the code at roughly sixty lines per second. (At that rate, the party would have had to linger more than seven hours to see the full million and a half lines of Mozilla code.) During the second of two sets played by the Kofy Brown Band (featuring a Netscape engineer), Eric Raymond, who had flown in from Philadelphia for the party, jumped on stage and surprised everyone with a flute solo. Toward the end of the night, a dozen CDs of the Mozilla Source Code, Signature Edition (CDs signed and numbered the night before by the Netscape Build Team and members of mozilla.org) were thrown to a lucky few in the crowd. The lizard was free!

The Authors :



Jim Hamerly is a Vice President in the Client Products Division of Netscape Communications Corporation. In June of 1997 Netscape acquired DigitalStyle Corporation, where Jim was a co-founder, president, and CEO.

Prior to founding DigitalStyle, he was Vice President, Engineering, of Pages Software, Inc. where he managed the development of Pages, a desktop publishing tool, and WebPages, the first WYSIWYG web authoring tool.

Jim spent 15 years with Xerox in various R&D and product development activities, most recently as Deputy Chief Engineer of XSoft, a software division of Xerox Corporation, where he was responsible for four software product lines.

Jim holds B.S., M.S., and Ph.D. degrees in Electrical Engineering and Computer Science from MIT, UC Berkeley, and Carnegie Mellon University.



Tom Paquin first joined IBM Research to work on a project involving parallel processors, but ended up doing a bitmapped graphics accelerator (AMD 29116-based) for the then-new PC. After tinkering on X6 and X9 at MIT and Brown University, he was part of the effort to ship the first-ever commercial X11 with Carnegie Mellon University.

Tom joined Silicon Graphics, Inc. (SGI) in May 1989, where he had the unlucky task of integrating the GL and X. He joined Jim Clark and Marc Andreesson at Netscape in April 1994. He was the very first engineering manager, guiding his team through the 1.0 and 2.0 releases of Mozilla. Now a Netscape fellow, he works on mozilla.org as the manager, problem arbitrator, and mysterious political leader.

First Published on O'Reilly.com

This pdf version is created by www.linuxjunkies.org