Q: I'm here with Mike Shaver and it is June 5th, 2006. Mike, when did you first start using computers?

A: I think the first memory I have, probably photographically assisted, of using computers, was probably at my dad's work in Georgia. So I would have been two and a half maybe, going to the office with him and press buttons on things that turned colours. Sort of more under my own power would have been a couple of years later.

I remember saving up my Christmas allowance and birthday money to buy my half of a Commodore 64 when I was, I guess, five or six. Somewhere in that. So yeah, my mother and father are both in the software field. So I come by that somewhat honestly.

Q: And do you—did you have formal training as well, later?

A: Classes in elementary and high school. But no post secondary in it at all, no.

Q: Okay, I guess you were quite young, but do you remember the first programming project you worked on?

A: Yeah, we used to get these magazines for the Commodore which would have little programs in the back you would type in. Because people didn't really distribute software for it in the same way. Or I couldn't afford the software that was distributed that way.

So there were certainly a couple of occasions of, you know, changing the colour in something and so forth. Hacking on those bits as I typed them in. I think the first program I wrote sort of on my own was a little happy birthday thing for my—a happy mother's day thing, I think, for my mom on our little computer. Just put a little graphic up and make it beep and we were done. But—yeah.

Q: And when did you first start contributing to open source projects?

A: First work on—first contribution to open source was I think '94. I'd been using Linux for a while and at the—in our consultancy I was working with, and I was interested in contributing to it. Didn't really—I was a little intimidated by the sort of technical depth in the kernel and the community around there.

So the first thing I did was actually audit some of the network code against various standards and add comments to it. This has this, this is compliant with this section, or this is not compliant with that section. And I got most of them wrong on the first pass. But iterated through that with help from Alan Cox and some other people.

So that was probably my first contribution into the code, would have been the Linux kernel then and—yeah, certainly done other, you know, sort of local modifications of open source software. But not really—I think it was upstream and [nothing] that was really contributed back. So that was probably my first involvement in a community like that, in an active way.

Q: How did you first soft of connect to open source? Was it just—like, how did you hear about it? Did you get—did you have any magazines or—

A: Yeah, a friend of mine, Dan Mickery ran a bulletin board of his own writing out of his house. And I helped him administer that one summer when he was away and then he just—he told me at one point, he was going to replace that with a Linux system and I was—it could have been a microwave, it could have been a goat.

So that was my first sort of exposure to Linux, which was really the first, I guess the first open source software that I was aware was open source. And that would have been, yeah, probably '92 or '93. And so that was sort of the—that was the introduction there, was just word of mouth from a friend and then I enjoyed playing on it quite a bit. It was all very, sort of Lion, the Witch and the Wardrobe, wondrous and confusing and probably allegorical in its own way.

And so I helped him on that system as well and eventually ended up installing it at home and—the price point was right also for a lot of the stuff we were doing at the consultancy I was working at Ingenia. We were very much a bootstrap organization and so having the software to be able to do these Unix'y things without cost, was important. Less so being able to maintain them ourselves. But that was something that certainly appealed to a lot of the students and myself that were working on that stuff.

Q: And when did you first start working with Mozilla projects then?

A: I was at Netscape in '97 with the JavaScript team. I was there technically as an intern and I'd been—I'd been agitating for a while to get the JavaScript engine released under an open source license. At that time there was this program called the Netscape One Open Network Environment Program. And you could send in a fax and they would send you the password to get the JavaScript source engine and you could do what you wanted with it. You just couldn't re-distribute the source.

So I had been playing with that a little bit. We were a Netscape var. The company that I was with had grown by that point a little bit. So I was interested in that technology at that point and I got pretty burned out on what I was doing at the consultancy. And I was invited to come down and to joint Netscape.

So I said at first, I'm going to—I'll come down for six months and I'll sort of take a (inaudible) there and work on this stuff. But I'm—I want to come back and do more stuff with this company here. And then so when I arrived there, I was talking to people in the JavaScript group—Brendan Eich and our manager at the time, Clayton Louis—that I thought, you know, there was a real opportunity for JavaScript to be used in a lot more environments. People were imbedding Tickle and so forth into their—into their software in universities and I thought that was sort of repellent.

And thought, you know, JavaScript could have a great opportunity to be imbedded in these environments, if the licensing got fixed. And they were receptive to that and it sort of worked its way up the chain. There was a plan to release that under the LGPL at the time and then late in December, I guess maybe early January at that point, of '98, I was told we weren't going to be releasing the JavaScript engine that way but we were going to do—we were going to do a broader strategy around open source. And I thought at the time, that that meant there was going to—that, you know, someone in legal or someone in—on the business side, had freaked out a little bit and said, "Oh, we need a big strategy for this. And make sure it doesn't hurt us and do—"

And so I was little bit disappointed. But I was invited to a meeting with a bunch of people to talk about that strategy and that's where I'd heard that we were going to be releasing the source. And so I was involved from that point on in a lot of little things. It was sort of a mini, sort of a little start-up bubble within Netscape. Everybody kind of did all the little pieces that you do together, when you're doing just, you know, a bunch of us did system administration. And did, you know, setting up of tools and figuring out policies and I worked with Mitchell and others on the license. And bringing—making connections with the other open source communities that I had some contact with.

So really from, sort of day minus one, I've been involved and sometimes, you know, I was full time with AOL and Netscape for a while after that. And I worked as a consultant. I've been in other jobs as well. So always been at least a little bit involved and have—and am now back, you know, in the fold full time at the corporation.

Q: So you've—and during certain points during—within your career at Mozilla, you've been a volunteer?

A: Yeah, initially I was at Netscape on staff and it was a couple of months after the project got rolling Brendan moved over to be working on Mozilla full time, from the JavaScript group and I did the same thing a couple of—maybe a year later, when Jamie Zawinski was leaving. They needed somebody to do developer relations there.

But yeah, after I left AOL, I was at a company called Zero Knowledge doing open source strategy for them. And I continued to be active, to whatever my time permitted, in Mozilla. Either technically or from a sort of procedural, organizational perspective and that kind of pattern of more or less involvement, has continued.

Q: So what projects—what are you kind of specifically working on now?

A: Nowadays, I'm responsible for—or irresponsible for our—the software ecosystem around Firefox and Mozilla technology. So working—it includes developer relations and so all of the great work that Deb Richardson and others have done on Mozilla Developer Centre, reflects well on me now.

A lot around our extensions community and how to help users get more value out of that. How to make sure that people who are building extensions, have the right hooks and are in—plugged into the process in the right way. But also, in terms of helping Mozilla do work that we've always wanted it to do I think, which is sort of be ambassadors for the Web, as a technology platform and as a way for people to build applications and communities and share experiences and so forth.

And, you know, there are a lot of—certainly a lot of technologies you can choose to build an application on today. Even those that are very network related and I think Mozilla has an important role as sort of being the evangelists and the defenders and in some ways, sort of the central nexus for Web technology. For a lot of people it is, and I think we can do a lot of good there. So that's also on the edge of my plate.

Q: How do you generally communicate with those you work with?

A: Sarcastically. It varies a lot. Sometimes obviously face to face in the office here and I travel quite a bit to California. Often—sometimes as often as every third week I'm there. A lot of it is IRC and email, instant messaging. Sort of depends on the kind of conversation we're having. Some things are much easier to deal with in email where you can edit a little bit and present more structured positions.

But I spend a lot of time on the phone with co-workers and with partners and just sorting through stuff in a higher band length way. So really all the tools at my disposal. Not a lot of postal mail or fax, you know, when I have to sign something. But pretty much everything else, Skype and Voice over IP or the Voice over IP that our phones are. IRC is huge. It's probably, the bulk of the real time communication is that. And then little bits of pieces I need to—

Q: Do you ever find that working in IRC can be problematic if there's no sort of record of what was said and certain people are left out of certain discussions, that sort of thing?

A: Yeah, it can be. I mean, it—so many of our channels are logged. So people can go back and refer to them and I keep logs on my client for a shorter period of time to go back and refer to. It's sort of my collective memory, or personal, I guess, memory.

It can be—I think it's the same risk you run with any conversation. I mean, you can have that problem in an email thread. You can have that problem in a newsgroup. There is certainly something to be said for making it easier for people to catch up quickly. So you can forward someone a whole email thread. But you can't always—it's hard—it's harder to get the same feeling out of IRC by re-reading it.

But I think there are a lot of decisions and a lot of discussions that can happen that way that are quite productive. I don't know that you can sort of reach the perfection of everybody that could add value to it, is involved in it. At some point you have to trust that you got the right set of people. Or that, you know, if you didn't and someone comes in and says, "You know, really should be doing this instead." Then you decide then if you're going to bear the cost of making that change or not.

But it is an issue for—it's always been an issue for the project. It was an issue when we first created the project at Netscape with, you know, a hundred Netscape engineers and employees who were all in the same building and used to having conversations that way. And private mailing lists and how much of that becomes public and what are the terms for it? And what's the—do you need to involve everybody equally in these conversations? Do you need to a record of decisions? What are you trying to—what problem are you trying to solve with this communication?

And I think we're in a pretty good—pretty good place now. I think the recent newsgroup re-organization and a renewed focus on using those groups for active product design, especially in Firefox and technology design, has helped a lot in helping new people get into the community and finding good contact points there. But also in creating a textual record of the major discussions. Between that and Bugzilla, we have a lot of history there I think. Probably more than any other project of our scale—whether that's proprietary or open. That you can get to and dig through. Sometimes there's a lot to absorb for that reason. There's a lot of groups you could follow and the bug traffic can be very high. But I think that's the right problem for us to have right now.

Q: When you're working in groups, how, generally is the division of labor determined?

A: Yeah, that's a good question. It's usually some riff on self-selection. You know, within the corporation, within the project, there are certainly areas of work that tend to fall—or areas of involvement that tend to fall to one group versus another. Within those groups, I guess it varies group to group. But, you know, on the developer relation side, on extensions on product development and design, it's generally, you know, whoever says, "I'll do that," that has, you know, that we can sort of credibly believe they will.

There's load balancing that happens. Mike Tripp our VP of Engineering at the corporation, has been excellent in helping us get better at that. And so there's some of that that goes on. But really it's a, you know, it's self-selection. It's—sometimes someone is volunteered for a task by someone else. Usually in a pretty soft way, right? "You should talk to so and so. I think he could help you out with that." Versus, "He'll do this."

Which is a little bit—people who have come to the project, you know, and they—"I'd like to help out." And they don't really know what they want to help out on. Sometimes want to be directed pretty specifically. Those people tend to not have a fantastic experience with the project. Because it's not something that's really set up to give hands on, step by step, individual mentorship.

Sometimes you'll get that, it really comes out of a personal relationship more than a task that's being mentored in that case. But yeah, people who can't sort of find a piece they want to work on, people—we have—everybody has something they want someone to fix for them. So you can get those initial ideas. But to find something that suits what they want to do and, you know, be proactive about getting the help they think they need—those people are much more successful.

And I think that's one of the reasons that it tends to—it has so far sort of fed on itself there. The people who have been successful and become significant members of the community, tend towards that style of self-motivation and learning. And so there isn't as much of a culture of a more traditionally—like, that didactic approach to—or explicit approach to bringing people into the project.

I'm not sure we could really sustain it, in terms of resources anyway. Mentoring is hard work. It takes a different set of skills sometimes from doing the work itself. And with time zone and language and communication, pieces that are there as well, it's—there are a lot of things that make that complicated.

I think in the right cases, it can work well. But we tend to prefer more highly leveraged things where I can explain a problem. Someone can explain a problem and the tools that, you know, “this is what I would start looking at first,” and then sort of answer questions once and get, you know, have people do work and explore stuff and say, "This didn't work. Why do you think that is?" And move on. Rather than, "I did this piece, what's next?"

So maybe that's—that might be something that, you know, might be leaving contributors on the table, so to speak, that could be very valuable to us if we nurtured them up. But I think that taking that cost on would not be something the project could really bear in its current form.

We've been working actually with schools, the one chief to my mind right now is Seneca College here in Toronto, to get some of that leverage. To work with the faculty and key student project groups there. To bring them into the community and help educate them about how things work so they can pass that on and work it through the rest of their programs and the rest of their faculty. And they have a strong interest in Open Source as well and a similar approach to the important parts of how software is designed and built, and especially the social piece.

So that's been working out really well and that's the model I think if we're—if we continue to be this successful with Seneca, we want to find a good environment to roll that out in, in other areas. We get a lot of contacts from universities, as you might imagine, who want to do something with Mozilla. And that's, again, an area where coming and saying, "What should we do," is not really as productive a start as, "We want to do this with Mozilla. What do you recommend we do?"

And that's how it started with Seneca. They had a project that some students were working on as part of a group co-op effectively and they wanted to use Firefox for part of that. They had some questions and wanted to be connected with someone that could help answer them. And that was successful enough that we sort of established this baseline of communication and credibility there. So it was easier to say, I'm going to invest some more time in this and talk to them about how they could broaden the reach of Open Source and Mozilla specifically in their organization.

But doing that without the benefit of an initial sort of engagement, that shows that there's a good cultural fit between the organizations and we're all talking about the same kinds of things. And something to point out on both sides to say, “this worked and that's why we're doing more of this.” I think without those things it would have been hard to have been successful and we've seen that in some other relationships that were kind of stillborn with other universities.

Q: Who generally do they tend to go to initially? Or does it kind of vary—?

A: Yeah, it varies a lot. Some will come in through a partner's address. Some will come in through a mailing list or a newsgroup post. If they know generally the kind of technology they want to be working with, you'll sometimes see an out of the blue mail to one of the owners of that. Whoever's—in some cases, just whoever's email address they can find. It's not always clear, for better or for worse, it's not always clear who the right person to talk to is.

A lot of them end up on our developer documentation list now because it's relatively prominent on the documentation site. We need to actually change some of the text around that to people's expectations better. But a lot of them come in, especially if they're coming from a commercial context, come in through the partner's address.

Academic, can come in through there, can come in through Bug comments or newsgroups, even more now. But I still—yeah, any module owner gets a couple of message a week, I think, of, "I'm working on this project, what do you think I should do?" Or, you know, "Who can tell me more?" And sometimes, often it's not to the right person at all. So you bounce them around a little bit. But yeah, it's pretty spread out.

Q: Would you say that strict ownership of specific areas of code are enforced?

A: In some areas, very much so. In some areas, less so. One of the characteristics of the project confounds some attempts to make that more rigid, is that there's a lot of code that's shared, both in purpose and in—where it comes from in terms of who's doing the coding, between different projects and people move in and out and—or their focus will shift from, you know, they'll need to fix something in this module in order to solve a problem they're having with their application. And so they'll do a lot of work on that and they're sort of the de facto owner.

In some areas, there is a fair bit of ownership history and a fair bit of dogma around that. Not always bad dogma by any means. The JavaScript engine, the Netscape portable run time, portability layer, the lay out engine, to some degree, networking. Certainly the application, Firefox application itself, you know, Ben is the module owner of record for that. There are lots of different kinds of ownership there around what code goes in? Where does the product go? What's prioritized? How is it managed?

And as those modules become bigger and as the—especially as they become bigger in terms of the number of contributors that are working on it, we're trying to find, you know, ways to apply other techniques, right? What does product management mean to Firefox? How does that differ from module ownership or product design? How can that help? What do we need to be worried about? Can it help us scale? How do we find people who can help us but also understand the context we're in.

I think we're doing a pretty good job there. I think, you know, Mike Connor, Mike Beltzner, have contributed a lot to helping Ben and helping in general, and Chris Beard as well, obviously. To figure out where the product is going. I think if you asked, you know, five module owners what module ownership means, you'd get many, many different answers.

But I think that there's usually an idea of where the buck stops, for a given module, if there's some dispute that can't be—especially technical dispute that can't be solved through cold application of reason. Where there's just a preference issue or there are two goods to be traded off.

In some areas of code, you know, there isn't really any ownership. It's code that hasn't changed in a long time. Or—and that's the dominant case. Where there's code that hasn't changed in a long time, so there hasn't really been a need for ownership to be exerted there and so there's no real sense of, you know, of who's responsible for those final decisions.

And so those will tend to trickle up to our sort of, uber owner, Brendan. And if there's a need for an ownership change there, if somebody appears out of the woodwork, that, you know, is strong and committed to it and has established some credibility, then that's usually not a hard call to make.

Q: So would that person go to Brendan or would that--?

A: Yeah, owners are appointed, I guess, declared maybe, by Mozilla staff. Brendan very much leads that decision-making process for software areas. We're trying to figure out how the notion of module ownership applies to things like licensing. To user interface design, to the website, to different things we do that aren't just the production of source code.

So there's certainly a debate there. The staff at Mozilla that our group--was really vital for a long time and then kind of waned, I think, in its direct day to day involvement in how the project operates. Probably for the best, I think. I like to think that it, you know, it shows we set up a structure that worked pretty well. People could resolve disputes on their own, generally, and find a common direction to work towards.

And even where those--where there are some, you know, pretty big variances in direction, you know, between people who continue to want to work on the suite, and the Sea Monkey project and those that are focused on Firefox and different release timelines and so forth. People are generally still able to find commonality and trade off their needs against each other's.

And in that context, Brendan and staff tend to have more of a sort of moderating or mediating role than declarative. And sort of in setting policy, tends to--like all good law, right? It's codification of a practice and Brendan's done a great job with that. I think we'll see changes in how module ownership works. There's certainly issues we haven't had as much in the past or haven't had as many years around concentration of developer talent and module ownership at different companies. Whether that's the corporation or other companies that are contributing to it.

And what that means and what the role of a module owner is. What their—what they can be expected to do that's in--that's counter to the desires of their company. How much we can ask them to wear two hats and how they can abdicate themselves, if they feel they should, for a given decision.

I don't think we will likely have a, you know, congressional caucus and build some sort of constitution out of it. We don't really operate that way. But I think we will have to find things that work and get quickly away from things that don't in that area. I mean, it's a problem that we had historically under Netscape as well. Initially, all of the module owners were Netscape employees and they were certainly, as all of us were then, under a lot of pressure to ship and to be able to ship Netscape's product. And that was the source of quite a bit of tension in the early years of the project.

And a delicate tension, right? The engineers are being—were being paid by Netscape and so they, you know, they paid the piper and they got to call the tune. But there was also, you know, how much can we push on the Mozilla side, for things that are important to the project and that we think are—or keep us away from decisions that we think will hurt it.

Because that was “our job” and many of us were Netscape employees that were charged with making sure the project was successful.

Q: So do you think—or some modules operate quite a bit differently than others, at this point?

A: Yeah, I would say so. I think there's some commonality there around, you know, around how review works. Around who can vouch for somebody else to review in that area or representing their module direction within the context of a larger release. But that tends to be more, again, sort of self selection and, you know, if people are following you, then you're leading.

So people who say, this is where I think we're going to go and obviously have the time to back that up or can exert influence in other ways to get those things done in addition to being declared, that's sort of the modern form, I think, of module ownership for us and it's worked pretty well.

Q: How important are comments in the code to sort of, smooth development?

A: It varies a lot. I think I'm—and yeah, I think this is, again, one of those things you'll get a lot of different answers. I'm sort of the opinion that comments are necessary where, for whatever, you know, reason of externality, the expressiveness of the language or a need for a specific performance or something imposed by another API, you can't write code that really says what it means. There really is sort of self-documenting.

There have been many comment policy or coding style, on comment policies that have been proposed or used within different groups. I think one of the things that you need in order to make comments in code valuable, is a real discipline about keeping them up to date. And in some modules that's done well and in others, there are clear lies. And to my mind, that's a little worse than not having them at all.

I mean, a good comment that, you know, describes the—lets you think about the software at a higher level as you're reading through it and let you check your assertions and so forth through that, is—can be very valuable. I don't think that's been something that we've really invested in a lot and it's not clear that that kind of in-code documentation is really the main barrier to improving our quality of our software. I think the complexity of the code needs to be reduced and we're working on that.

And at that point, we'll be able to look and say, you know, “Are there pieces of this code that need some comments to describe it so people can come in and understand what it does and what they should know if they're going to change it?” Or, “Is the code clear or robust enough that it can accommodate those things?” And some of that's tied also into, you know, the state of test suites and the coverage of the tests that we do run. How confident are we that, you know, we can verify a change versus having to reason it out in priori or vice versa.

And that's where comments can help. But I don't think it's been—in some areas, in the layout module in JavaScript, in Necko, I guess, there's—there are often pretty good comments describing how things are expected to work. And if you find a discrepancy there, it's usually something you need to look more closely at. In other areas, especially the app front end there's often not as much explicit comment text.

The higher level languages, like, or the languages like JavaScript that let you express things a little differently, can make it a little easier, I think. To express some things. Sometimes they obscure your meaning in different ways but we don't—we certainly don't see as much in the way of explicit in-code documentation at those app levels. For whatever reason that is.

Q: Have you ever clashed with somebody else over a point of code and how do you resolve those problems generally?

A: Oh, yeah. Yeah, I mean, that happens all the time. It happens in small ways in every code review. "I think you should do X," "I did Y for a reason," etc. And that's an area where module ownership comes in, in a lot of cases and yeah, I mean, it's really case-by-case.

I'm trying to think of an example where we didn't just kind of pound it out and, you know, "This will happen if we do that." "This will happen if we do Y." "Well, I'll do this now and if that proves to be a problem, we'll do that later." Or, "Here's this code that does expect this." Or, "Here's how we did it in this other case." Or, "Here's what the spec actually says." Those kinds of things.

They are certainly stylistic deviations. But the rule there tends to be, when in Rome. So, you know, whatever code you're editing, you should look like--the code you put in should look like the, you know, the code that was already there. And in some areas, the code style is consistent enough that you can really do that, sort of lose your code in the rest of it and it blends in well. And in some it's—a lot of that code pre-dates that kind of discipline, or just wasn't written with it in mind. So there are some—there are occasional clashes there.

It seems to not be a huge, huge problem. There's—the most common form, I think, is a difference of opinion about when code is ready—when code is good enough to be put in and what can be fixed later, and sort of what the relative time value of that software is. And of course, as in all software projects, probably in all endeavors of man, often saying, I'll do it later, means it won't get done. Because it's, you know, it's not blocking this thing that you needed and there's lots of other stuff to do. So—and people are rightly wary of that in many cases.

But I think generally we're—whether it's always sort of agreed with spiritually or not, people sort of know what to expect in a given module about what's going to be accepted in and what's not. And so those things tend to not be a huge problem and new contributors that come in tend to be pretty receptive to learning how things should be. Sometimes frustrated by the review cycle and the number of details that are involved in it. But that's a cost of the complexity of the software we're using and wanting it to work well on all these platforms, and perform well and have good code size. So they tend to see the value of it, even if it's frustrating initially.

Q: Do you see any sort of repeated tensions between people who work on the front end and people who work on the back end?

A: Yeah, I think you see tensions between any module—any, you know, arbitrarily chosen pair of modules, where—and I mean, it can be in a form of, you know, I'm the caller of this code and I think it should do this for me. Or, versus I think the call—you know, the code that's calling in should make sure this is the case and I shouldn't have to defend against that or penalize all the other callers for it.

More often, we see a need to get people on the same page with respect to what's going to be compatible and what's not. What are the important use cases and what are the important calling patterns. The compatibility one is really important because especially when it comes to binary compatibility, you sort of have it at the whole system or you don't have it anywhere.

You can't--it's hard for us to say, you know, these pieces are staying the same and that's really valuable, but this other piece is going to change a lot and it's hard to describe to people which extensions will break for that. It's hard to know how to maintain those things in parallel. So we really all need to be on the same page about that stuff.

I think the biggest other source of tension, is just around scheduling. How late can features and changes come in? Who's responsible for those things? You know, if code goes into a module that the—on a—you know, on a branch for a product, that the module owner didn't like, are they—are they still sort of expected, in whatever volunteer sense, to help find and diagnose bugs in that code?

If they don't want to maintain something on two branches, do they have to care about that other branch at all? How much is that borne by people who take care of minority platforms or older versions or unofficial apps? And that's certainly an area that's very fruitful for discussion within the community.

Yeah, I mean, I think you get tension in the economic sense. There are definitely scarce resources of programmer time and software complexity and calendar time to be meted out, or doled out. And people advocate for their position, I think almost universally in good faith. And where there are differences of opinion, we, you know, only one of those things can go in the code.

Even, you know, choosing to do both, optionally, is something we try to avoid now because it imposes costs on both sides and leads to having a less pleasant project to work in. But yeah, those tensions tend to be around issues and not around, you know, a long-term sort of Hatfield and McCoy relationship between two modules. But enough of those issues can be cause for some pretty strained feelings.

We had this problem as we were releasing Firefox 1.0 off of what we called the "aviary branch," which was where Firefox and Thunderbird were doing their work. And we'd cut that branch for—and we expected it to be a short-lived branch. And we had expected to not take any core changes in it and then there were a number of things that wanted—we wanted to back port from the ongoing development. And it ended up lasting a lot longer than we thought it would, because it--we ended up shipping Firefox later than we thought it would, etc.

And how that was landed and that there was changes made in one side and not the other and regressions on those—when that code came back home to roost, caused some pretty strained relationships there, and I and others had to invest a fair bit of energy in getting people back at the table. I mean, no one was going to quit over it, I don't think. But it wasn't the most productive environment. It was hard to do sort of reasonable judgment based work after that. People were being pretty defensive of what they believed and in many cases, they were right—were reasonable requirements, you know, for the code that they were ostensibly being held responsible for. And that's certainly informed our work today on the parallel work on Firefox 2.0 and 3.0 and on Gecko 1.8 and 1.9, around patch discipline and landing on both branch and trunk and keeping these things in sync and having policies around that so people know what's expected and what they can and can't do in a given branch.

So some good came out of that. But that was—that's sort of—that's the one that comes to mind as being the most recent sort of major schism between front and back end. And that's calmed down quite a bit. There's a much better conversation there than there was a year and a half, even maybe a year ago, between those two groups.

Q: To what extent does Mozilla rely on the work of volunteers and maybe how has that reliance changed over time?

A: So in one sense, Mozilla has always been virtually a hundred percent dependent on volunteers, whether they're volunteered by a company or it's an individual that takes it themselves, or a corporation or a—or sorry, a school or some other organization. I mean, in a very real sense, Netscape was volunteering its continuing involvement in the project and its work.

We were, you know, pretty confident that that was going to last a while. They had good reasons to do so and those are the best relationships really, where it's not an act of charity. It's mutual assistance and whether that's very concrete, you know, they need some piece of technology or they need something improved in the browser for what they're doing, or trying to preserve the health of the Web.

So it was only, you know, relatively recently, a couple of years ago, that we had the foundation created, which gave a legal entity and some people who were paid by the Mozilla Foundation to work on the project. And that's—I guess that was—those were the first sort of sets of people that weren't volunteer in that sense.

And today, with the corporation, in some sense as an extension of that and that's—but in another sense, it's, again, you know, the corporation certainly has a lot of latitude about where its energy is devoted. And, you know, it's hard to imagine them working on something other than Mozilla, it's sort of nonsensical. But even within that, there's a lot of volunteered effort to work on things that are not, you know, directly related to the corporate—the products the corporation supports. Helping, you know, sustain the community and bringing new people in to it.

So certainly it's always been dependent on the kindness of strangers, as it were. I think that a lot of it though, when you talk about volunteers, it comes across as being charitable or requiring a pretty high level of altruism, or being fragile organizationally. That, you know, everybody could just not show up for work tomorrow. I think that's true for a lot of—for pretty much any software organization. People could just not show up tomorrow at IBM or Microsoft.

They have incentives to be there. They're all at will contracts. There's no real penalty. It's just the lack—you know, the removal of those incentives and Mozilla is interesting enough to enough people, that the success of

Mozilla is a significant incentive. And there's certainly a lot of organizations that see that and incent their employees financially. And there are organizations that—there are individuals that see their own, you know, see a way towards some result they want, through the Mozilla project.

So yeah, I like to think of them as motivated volunteers, or volunteers in the same sense as a, you know, a modern armed forces has all volunteer force, right? There are people who choose to be there and once they're there, then, you know, they have incentives to stay and they do it because they believe they'll get, you know, some return—personal or societal or professional out of it.

Q: Why do you think most individuals volunteer?

A: Yeah, it's an interesting question and it was one we struggled with for awhile, trying to bring more people into that. We wanted, especially early on in the project, we were concerned about the amount of the code that was controlled by Netscape. And it seemed like we—in order to be successful, we needed to have, you know, more of that being done elsewhere.

Looking back at that in hindsight, I might feel—might have felt about that different—a little bit differently about that if I'd thought about it differently. But I think people come in because it's an interesting technical challenge. There's certainly a lot of very interesting technical work that's happening there and it's visible on the Web and a lot of people are interested in the Web. And having, you know, we're sort of—we're, I think, by far the, you know, the premier open source client, for the Web.

So if you want to change the way something works, you can do that through us. People are interested in it, because they can scratch an itch. A lot of our contributors have come in that way. That they've been—they've wanted something to be different. They wanted something to be fixed for their own use.

In some cases, because they have a social or political preference for open source and they want to contribute to that. We're a pretty highly visible project. I think that's where a lot of the student interest comes from, and academic interest, is that you can get involved in this project and you can, you know, do valuable software that you can point at on a resume or in terms of increasing the prestige of your institution.

And so there's certainly a lot of that there. People—I'm sure some people do it as—today, as a way to maybe get a job, you know, being paid to work on it. It varies a lot, person to person. I think the technical challenge piece of it and the, you know, contributing to the health of the Web, are probably the two dominant ones that are not financial. And those are—those tend to be better served through companies that are employing people to work on various pieces of Mozilla.

Q: When you were a volunteer, volunteer for Mozilla, what was your primary reason?

A: Inertia. No! I mean, I—yeah, I had a lot of—part of the—I say that half jokingly. I mean, part of it is just that there was a lot of emotional energy invested in it and I really did believe that the Web was important for a lot of people. And to serve them well, it needed to—you needed to maintain choice there. You needed to make sure there was—it wasn't just the confluence of differently aligned corporate interests.

And that—those can get you a long way. A lot of great technology and even social advances that have come out of that, kind of commercial competition. But also because I saw a lot of promise in the Mozilla technology for making things easier or more possible on the Web and I like to build applications, and as a software developer, you know, I wanted to be able to use those things.

And there were a lot of people in that community that I had bonds with. It developed through my work in Mozilla and I wanted to help them be successful as well. Yeah, I mean, some of that is whatever neuro chemical imbalance that makes it hard for me to say no to those sorts of requests. But a lot of it is, I think there was a lot of promise in Mozilla and there obviously—I still believe there is. And to whatever extent I could contribute to that, I wanted to.

And yeah, it was fun and it was something I took quite a bit of pride from. And certainly during the period in which we were, you know, open source's greatest failure and, you know, people pointed, "Don't want my project to turn out like Mozilla. Nobody contributes to it, and it's big and slow and took years and years to get a release out."

I really did think we'd done a lot of work there. So pushing out Mozilla 1.0 was not hard at all to justify to myself. And, you know, I sort of teetered there between really wanting to be involved and being kind of burnt out on it for a couple of years because there was such a high level of emotional investment and a lot of overlap between my professional and personal life there.

But I'm in a much more balanced place now. And I think a lot of those motivations remain today, that it's a little easier to play to them now because it's easier to demonstrate them, it's easier to show what, you know, open source available cross platform, you know, user centric, you know, our technology can do—you can point to what Firefox and Thunderbird have done. As before, it really required a leap of faith to see that that could happen.

And, you know, it was by no means inevitable. We didn't—you know, we couldn't predict the lottery numbers ahead of time, and to some extent we got lucky with Firefox. But to some extent we were unlucky that we didn't have anything like Firefox for the preceding four or five years. I think we made that luck in a lot of ways.

And now, for a lot of us, I think, certainly for me, it's about using that leverage we have now and the resources we have at our disposal in terms of our brand, in terms of revenue, in terms of the reach of the community, in terms of the influences in the industry and other industries. And, you know, social organizations and policy groups to make sure that the Web moves the way we feel it should for the rest of the world.

And, you know, at the end of the day, we're—a lot of us, and certainly the foundation and corporation, are paid to use our judgment about where we should go. And that's a—you know, if you believe in the future of network software being something that's available and open and universal, it's hard to imagine where else you'd be, to work on this sort of thing. Be able to have this kind of incredibly leveraged impact on technology people use every day. Yeah, it's hard to imagine why you'd be somewhere else. Hard for me to imagine being somewhere else.

Today, I mean, I might get a great offer tomorrow and then we'll have to go back and edit this whole thing out but—

Q: Why do you think Firefox in particular, but other Mozilla products too, have been able to—has been able to attract so many users?

A: So yeah, I mean, Firefox is the—probably the easiest one to talk about there. Because it's been such a dramatic attraction of users in such a relatively short period of time and in a market that was so locked down. Part of it is that the—you know, it was addressing demand and there was a period there where the security record of Internet Explorer was bad and painful enough to people that they were looking for alternatives—and spy ware and pop-ups and so forth. And one of the things that manifested in Firefox, that's always been a part of Mozilla, and people were going to be able to see through Firefox, was that the software is really about putting the user in control. And it's not just about things like pop-up blocking and sort of explicit controls on limiting content, but it's also around making sure that the primary thing that the tool is doing, isn't, you know, selling you something through, you know, the old Netscape personal toolbar. The shop button. It's not about profiling specific, you know, partners through a revenue deal.

It's about focusing on what you want to do on the Internet. On being simple enough that anybody could use it. On being pleasant enough that you can expect more from the Web. And the fact that the Web browser industry had largely been stagnant for—for most people it had been totally stagnant for five years, at that point, four or five years.

In one sense we didn't have—we wouldn't have had to a lot in order to come in and be credible there. But I think in order to really achieve critical mass, to have sort of an escape velocity, for people switching browsers. Which of course all of these sort of layers of understanding underneath, that there is a browser, right? That a lot of—for a lot of people, the Internet is what they get when they click on the "e" and there's no real difference for them—they don't understand the difference between I'm using Google and I'm using a browser to use—to reach Google.

And there are lots of great analogies there between, you know, the difference between a TV channel and your television and that sort of thing. But having people understand that and having people understand what the effects of their choice could be, is huge. I think IE 7 will help us a lot in that regard. Microsoft's going to spend a lot of money talking to people about what the browser is and how the quality of the browser can impact their lives. I'm all for that.

But I think we had a good user experience. You know, it's hard to point at specific pieces there. Tab browsing and search and pop-up blocking—you can do feature checklist stuff. But I think we did what Joel Spolsky talks about software, consumer software needs to do, which is create an emotional contact. It was pleasant to use, it felt like it was on your side. It looked nice, it didn't get in your way.

We spent a lot of energy on making it faster and on making it smaller. So you're—you know, you go from, “I'd like to try that,” to, “I'm using it,” in as short a period of time as possible and I think people just found it refreshing. And I think some people switched away because they wanted to vote with their feet. Some of them switched away because they wanted open source. And I think a lot of people switched away because they had a better experience, and they heard from somebody they'd have a better experience with it.

And at the end of the day, I think that's the test for all of Firefox's future work, is, is the user going to have a better experience with this than without this, for anything we put in. You know, as we have more users and as we have, you know, more technical capabilities, we can build on the, you know, each version going forward, we—you know, we want to make sure it's still accessible to everybody. We want to make sure people understand how to do things with the browser that it doesn't get in their way or require them to learn a lot.

But I think we also have an important opportunity to help people become more demanding Web citizens. That they should expect to be able to get more of their content and their experiences on their terms. That we should be able to provide powerful tools for, you know, bookmark management, that lets them do more with their personal subset of the Web.

And we shouldn't be afraid of putting in features that are—that require the user to be more involved or more invested in their Web experience. Especially if they can get there incrementally and, you know, there's a friendly ramp up to it and they get some early value from it. I think we do ourselves, and I think we do the Web a service, by having people expect more from us and from their software and from the Web and to be more in control of that.

That's something that was missing for a long time. People were very disconnected from the development of the Web. They felt like it was something they would just consume. Even a lot of technologists. But certainly a lot of, you know, lay people didn't really feel like they could influence how the Web was. It was, you know, "Are there a lot of pop-up ads? That's just how the Web is," right?

"Does it not appear in my language? That's just how the Web is." " Can I use it with a screen reader? Not so much? That's just the nature of the Web," right? Vegetarians can't eat steak, I can't use the Web. And I think we want to get people away from that.

People say, "Why can't I do this? I want to build this thing, I should be able to do that on the Web? Why can't I? You know, what needs to happen for that to be possible? Why can't I get the full Web on my device or on my television or in my language?" And that will, of course, you know, "backfire," you can't see the air quotes on the tape, in that we'll have much more demanding users and they will demand things from us and we'll have to satisfy them or they'll go somewhere else.

But, you know, even users switching from IE 6 to IE 7, in some ways, is a victory for us. Not as huge a victory as it might be for them to come to Firefox. But for people to be consciously choosing IE 7 over Firefox versus falling, just sort of using IE 6 because it's there and they don't know they have a choice, so they don't have a choice, that's way better.

You know, if the world were very different and, you know, there were a number of browser choices that let people express their desires in a marketplace sense and that could help reach some of these niche, you know, smaller markets that don't get localizations in their language or work on their platform, then we might not have needed to do something like Firefox. We'd be able to focus our energies differently.

But for now, presenting that choice and, you know, creating demand, as Galbraith said, showing people what's possible and making them want it. Not only in our product, but in others, I think is an important part of what we do.

Q: How important was the—the sort of small team that worked on Firefox's UI, to the development of Firefox?

A: I think in one sense, it was vital. I think the nature of the Firefox application, sort of the core values that it—I mean, there's the sets of stuff we talk about in the FAQ. And that, you know, it's in the manifesto and so forth. But that's not as important to me as the values it sort of expresses in its conversation with the user.

And a lot of those were set out boldly by that original team. And they knew what they wanted and they were going to build it and if people didn't like it, then they didn't like it. But they were going to stick to their, you know, they really believed that there was a market for this in the commercial and social sense.

So I think that was very important. I think one of the things we've had a bit of a challenge with, is figuring how to scale that. We want to do more things and design more things than one or two people can do. And certainly, I don't think, you know, Ben wants to burn himself out, as he did on that Firefox 1.0 cycle.

So we know we need to broaden that. We need to broaden the number of minds that can contribute to solving these problems. But we don't want to dilute the design ethic. And I think, you know, that original team, Ben and Blake and Dave, would say that it's not necessarily about them making the decisions. It's not necessarily even that, you know, they can't make any bad decisions about UI or design, or that it has to be what they would prefer.

But that those core values about, you know, simplicity and approachability and putting the user in control and it being pleasant to use, need to be manifested in all the things we do. And, you know, that's where things like tab browsing and the search bar and so forth, come from, is, we think it makes it easier and more pleasant to use the Web and we want more people to use the Web for everything they do.

But I think that—from the perspective of kicking off, you know, a project that was so different from what they were doing and moving on it quickly and keeping it coherent—I don't know of any way to do that that doesn't involve a small team. And we're trying to figure out how to do that with more of the project. The extensions space has been a great example for this too. Where, you know, you can mix and match a bunch of these things, and some of them don't taste great together.

But a lot of them have great ideas and it's one or two people and they're able to focus on this piece. And then we have to figure out how they fit together in a coherent whole. But in terms of exploring the value around a given feature or a given way of interacting, you know, Firefox is a great platform for that now and we need to figure out how to do more of that. Bring more of that disparate energy and parallel innovation, from the extension space and people's research into the Firefox process as well.

And that's something I know, you know, Mike Beltzner and Mike Connor and Chris Beard and Ben and Shrep and myself, pretty much everybody that's involved in the product, you know, has in the back of their mind, how are we going to scale this? How are we going to do more of these things and bring more of that innovation to—that's appropriate, closer to our huge user base, so they all get the benefit of it?

Q: What role do you think marketing has in attracting the users to Firefox? And specifically what role do you think Spread Firefox plays? Like, do you think that professional marketers are crucial and necessary? Or do you think sort of developers can kind of create a certain base of volunteers and then just sort of let them run with it?

A: Well, yeah. I think that there's certainly—there are a lot of people out there who aren't using Firefox, not because it doesn't work well for them or better for them than what they're using, and not because they can't, but because they don't know about it. And for a lot—reaching a lot of those users, I think, traditional media channels, the perhaps not traditional marketing approach, is probably our best bet. It's an understood play. We don't have to invent everything from scratch. We don't have to figure out how that—we can get help.

But I think there—for a lot of the people we want to reach as well, they're not going to be as responsive to that, and I think we need to do stuff in both of those spaces. I think Spread Firefox was tremendous in—not only in raising awareness of Firefox, but in raising awareness of how what we were trying to do was different from just producing a software and just from producing a product we wanted users to be using.

We wanted to create a community of investment in the product and in the technology and in, you know, the open source project that wasn't just technologists, it wasn't just people building software and things on the periphery of software. It was about, you know, helping us understand how to reach and serve people in different regions or with different preferences or different affinity groups. How to apply the breadth of open source techniques and, you know, collaborative development and, you know, sort of right of self-determination to marketing. How to balance that stuff with trademark protection and protection of the brand and, you know, who's speaking for Mozilla and who's not?

I think we couldn't have got—it would be almost impossible for us to run a, you know, traditional professional or commercial marketing campaign without the benefit of the foundation of Spread Firefox and the sort of community involvement there. And it's attracted a lot of great people, many of whom are professionally involved in some facet of advertising or marketing.

But I think that that kind of—whether it's Spread Firefox or something like that, I think that kind of connection between the people who are doing—you know, actually speaking to another human about the product and are thinking about, "How can I reach the people who are near me who are in my context best, about Firefox?" I think those connections are very valuable, and those connections have probably a higher level of stickiness, a higher, you know, take-rate, whatever the measurement is, than a broadcast through, you know, the Super Bowl ad or, you know, traditional media buy. But to get to a lot of—you know, a lot of people watch the Super Bowl, and a lot of people who watch the Super Bowl would probably be interested in people that do different things on the Web.

The other really valuable thing about Spread Firefox and—you know, it's a value we did on the technical side as well is, there's a great story about the story there, right? Taking a two-page ad in the *New York Times* to talk about your product, you know, your little scrappy startup, that's kind of news. Having, you know, however many thousands of volunteers chip in money out of their pocket to do this, basically on faith, right? This guy said, "I'm going to run an ad," and he'd been around the community for a while and people said, "Great, I'd get in on that."

That's not just an ad. That's not just a product, right? And that's something that people talk about. And that's something that helps us make sure that there's stories and there's awareness of the fact that we're different, not just a software. And that people can get involved if they choose to and why they might want to be involved, and the kinds of people who are involved, right? It's not all the traditional open-source crowd, though I absolutely have nothing against my fellow traditional open-source crowd. You know, it's people who have never done anything like that before. It's people who've never told somebody else about a piece of software or, you know, put a sticker for a piece of software on their car.

We've had people come up and say, "You know, I really want to get Firefox." You know, they hear I work on it. But it's like, "What is Firefox?" They don't know, but they know that all the, you know, all the people in their office or their kids or, you know, people in their dorm are talking about it. In the dorms I think they tend to actually know what Firefox is. We have pretty good market coverage there.

But I think our Spread Firefox style of work is enhanced by having some of the professional tools and outreach out of it. I think that our traditional media channel marketing couldn't be anywhere near as effective as we need it to be without that base. I mean, Firefox Flicks is a great example of this, right? It's Spread Firefox in the way these things kind of get pulled together, and it's, you know, traditional commercial marketing in that we had an awards dinner and we had these, you know, celebrity judges and we have connections with the film festivals and so forth about airing these things.

But the film festivals wouldn't be interested if it didn't have a story behind it, and we wouldn't have as much a community interest there if they didn't—if there wasn't the ability for this stuff they're doing to reach more people. And I don't really see them in conflict. I think people who work on Spread Firefox work on it because they want to help Firefox succeed. You know, if the marketers are going to help with that, if we're going to get—you know, we're going to pay or we get partners to help pay for marketing that reaches new sets of people, I think everybody's on side with that. And I know everybody we've talked to that we've had any interest in at all on working with the professional marketing side, if you will, has been fascinated by the energy and breadth of the Spread Firefox community and other, you know, other marketing sort of communities in our world.

Q: And why have marketing efforts mainly been focused on Firefox as opposed to other Mozilla products?

A: The only other sort of real Mozilla product today is Thunderbird, and I think for a lot of—we're better at consumer software than we are at any other kind of software. And sometimes that's very frustrating for our enterprise partners and sometimes that's frustrating for the developers who want to make these things work better in the enterprise. For a lot of consumers, mail is Web mail. Everybody uses a browser. Many fewer people, especially individuals who control the software on their machine or could decide to try something new, are using mail clients.

I think it's also a harder transition to get right. There's a lot of value trapped in mail. It's harder to go back. They're more complex systems to configure and to get it right. And certainly, you know, the Thunderbird team's done a great job at ever improving the migration experience. But I don't know, I think it's—there's also a lot more stuff you can—a lot more interesting stuff you can do with the Web today than you can do with mail. You know, mail's sort of the Internet's killer app in the sense of, you know, making it happen, and the Internet couldn't survive without it.

But you can do a lot more different things on the Web than you can with mail, and I think that's why it's been more receptive—people have been more receptive to it. And I think that, you know, marketing focusing on that is really a matter of playing to our strengths. There's a lot of energy around it, and we want to amplify that. It's a lot harder for us to, you know, create that demand, certainly in the areas we've been marketing in. I think internationally the story may be different. Thunderbird's big in Japan, as they say, and in some parts of Europe. Much bigger than it is in North America where for a consumer it's usually Web mail is their mail. So that may be something that as we broaden our international focus we see more marketing efforts around.

But I think for now the other thing is just that, you know, we can only do so much, and Firefox is at least as much as we can handle from that perspective. We have to trade stuff off there already, and it's also, I think, personally, I don't think this is necessarily policy, but I think it's easier for us to convert additional Firefox success into Thunderbird success later than the other way around. I think it's easier to—the value proposition for the browser is clearer. It's easier to convert people that way than from a mail client to the Web. That's sort of my gut, anyway. I don't know that we will ever be able to run experiments to validate that, but yeah. And it's been working. There's a lot of energy around Firefox. There's a lot of people interested in making it more successful.

There's certainly energy around Thunderbird as well, but I think the scale is different and I think a lot of it just comes down to the fact that more people are interested in a browser and interested in the Web than are interested in mail. And we need to just sort of operate in that context.

Q: Okay. How would you list Mozilla's priorities? And since you've been with Mozilla for a while in one fashion or another, how do you think, or have priorities changed over time?

A: I think when we started out our priority was to be a healthy open-source project. That was really the only aim we had. You know, we wanted to produce good technology and, you know, conform to all the standards and run everywhere for everybody. But really what we were focusing a lot of our energy on and worrying about a lot was, are we going to be an open-source project that has its own life, or will we be, you know, something Netscape throws over the wall? And that was--quite a lot of our energy. It was a hard thing. There's lots of reason why—or maybe they're just excuses—why that was a hard nut for us to crack.

I think now we're a little more confident that we can do the right thing with the software and with the goal and what the results we want to have of that project are. And we're confident enough in it that we believe people will come along. That there's people out there who are like-minded and will contribute that way, whether for reasons of technical interest or sort of social alignment.

I think our priorities are the health of the Web. I mean, I think that really underlies pretty much all of what we do. I think a healthy Web is important to us because it touches a lot of people and because it can be an enabler for a lot of different kinds of experiences and improvements in, you know, global society. That there's a lot of potential there, and for that potential to be realized the Web needs to continue to be open and available. It needs to continue to be something where users, individual people, you know, the human at the keyboard, has choice and isn't excluded because of language, disability, choice of platform. You know, the $100 laptop versus the Alienware gaming rig. You should be able to experience to a large degree the same Web.

And so I think that, you know, that's sort of our—it's not even a goal, really. It's the virtue. It's the reason for being. And I think our goals around that are, you know, success of the—from the corporation's perspective, success of the product in helping users get the most out of the Web, and their mail in the Thunderbird case, and helping create and sustain a culture of continual improvement on the Web. We don't want the Web to fall back into a stagnant state. We don't want browsers to be frozen in stasis like Hans Solo for five years.

I mean, so that's—we wake up in the morning. How do you decide what you're going to work on that day? And for a lot of us I think that's what it is. It's how are we going to make sure that this—you know, Firefox reaching users gives us more leverage over the Web and improves, we believe, the lot of those users. You know, when we talk to potential partners and we talk to organizations that want to work with us, you know, explain to them our sort of, three priorities there are for determining what happens with the product.

First is user experience, right? Is this going to be better for the user? The second is, you know, we're interested in distribution. How do we get the product to more people? There are a lot of people in the world who can't or won't download software, so right now they're out of reach to us, with a couple of exceptions. The Google software pack and one OEM bundling deal we had as an experiment. And the third is revenue, which is really sort of a synonym for resources applicable to the project. Money is easy to convert in a lot of ways, into headcount, into marketing, into other kinds of resources, infrastructure for the project. But getting those things directly is also fine, too. And that's there because we can invest those things in improving the user experience and distribution, again, sort of in service of—we believe that a better Firefox in the hands of more people will help us improve the Web, even beyond the improvement of the Web that it will cause just by having that be the case. So we do a lot of things in service of that.

And we invest in other sort of longer or side elements, too, around helping people build things on the Web, helping people build things in Firefox, extensions in Web services. Making it easier to localize. Making it more accessible. Which are in some senses sort of longer term plays, making Web technology applicable to more people, helping people do more things on the Web. You don't want to wake up and have the next Gmail be a XAML app somewhere or a Cocoa app even, that can only run in this given environment and reduces the user's choice of how to interact with it.

So we have investments in the platform, we call it, in the technology that's ours and also that's sort of the core of the Web. But in some ways they're also very short-term investments for us and they reflect very quickly on the product, right? The breadth of the extension community and the number of different things you can get to plug into your browser to make it your browser, has been a huge selling point for us in a lot of areas, and it lets us keep the core small and focused. And where there is a feature that's significant to a significant number of people, but is, you know, too much of a user experience cost to bear for everybody if they're not using it, it can be provided well through an extension. And that's been very important for us, and, you know, people grow attached to their extensions and they're attracted into the Firefox community that way. And again, they expect more from the Web and from other browsers and technology. So we get a lot of short-term return on that as well, which is nice to see.

Q: Could you briefly describe the relationship between the corporation and the foundation?

A: Sure. The corporation has one shareholder, which is the foundation. The corporation's job is to build the products and technology that further the

Mozilla Foundation's mission. Mozilla Foundation has as a mission choice and innovation on the Internet, and they have a small staff that does project governance and [indiscernible] that stuff. And I think they like we, are trying to sort of figure out their best path forward. So Mozilla Corporation is about—perhaps obviously, wholly owned. It's taxable but it's not really for-profit in the normal use of that sense in that nobody gets paid out on a basis of that profit. The money goes back to our single shareholder, which is the foundation, where it goes anywhere. Virtually all of it's invested right back into the project in short term.

And it's the product, it's the software product wing or tool or platoon of the foundation. So we're where the software happens in service of the foundation's goals. And right now most of what the foundation is doing, sort of the Mozilla group of companies, right, the foundation in a larger sense, most of what we are doing is around building software and making sure the right software's built and gets to the right people. So most of the resources of the foundation that are being spent right now are being expended through the corporation.

But the foundation is to a large degree, in control there, and, you know, we have a fair bit of autonomy. We don't call the board up every time we want to make a bug fix. But our goals and targets are set by the corporation board and by the foundation board, and we do, I think, all of us see ourselves as doing a different facet of the foundation's work and making their tax lives easier by not trying to convince the IRS that there's a role for non-profit tax—non-profit software creation that generates a lot of revenue.

In a lot of ways it was an administrative change when the corporation was spun out, right, a lot of the same people working with the same people doing the same things. We were able to do a lot of things we weren't able to do before. We're able to do some things much more easily. And yeah, I think it's worked pretty well.

There's a cost to the level of interaction there. We need to coordinate those things, and we disagree sometimes about the relative value of different types of work or investment. Not usually fiscal investment, but dedication of resources. But that's all pretty standard fare. I think the corporation's an interesting model, too, for showing how what in some sense is a traditional software company. We're a software company. We're built on an open-source framework and so forth. But in a lot of ways, a lot of good ways I think, we're a software company. And how that can be done in a way that's in service of a social goal and still be quite effective at delivering good software and in focusing on that software. So I like to think we're setting a good example for people. And if not, they'll learn from our mistakes. So a cautionary tale or a guiding light, we can really win either way.

Q: When Chase Phillips left the corporation, he said that he basically was lacking knowledge of where the place as a whole was headed. How would you respond to that characterization, and do you feel like—have other people sort of raised those types of concerns?

A: Yeah. I think—the project goes through this periodically where there's a major change in context or in community disposition. There's a lot of stuff going on, and it's not always clear who's talking to who, who's making what decisions. I think that certainly within the corporation operationally there was a huge amount of growth that was going on, and there were a huge number of additional things we were trying to do. Every time we'd add one more person we'd add two more people worth of work. And communication certainly doesn't flourish in that environment.

I think—I was a member of the management team then. I think we certainly could have done a better job of communicating what we were doing. Not in the sense of what the corporation's doing but what the people who were in those management positions were doing and trying to solve and so forth. I think there was, you know, a need to improve communication down the organization, if you will, but also up and also laterally. And I think in a lot of cases people were concerned because they weren't sure they were doing the right thing. If you don't know sort of what other people—what the end goal is or what you're trying to achieve. And some people are confident and will just do the right thing or what they believe to be is the right thing. And in some cases they'll want some sort of validation of that, some direction of that.

Certainly I think, to a person in the corporation and the foundation there was a huge amount of emotional investment in the work people were doing. And that raises the stakes for a lot of these things, where it's not just "I don't know—I don't feel like I know enough to make good decisions about my work," but it becomes to a large degree, "I don't know enough to make good decisions about who I am and whether I should be working on this and whether I should be making any sacrifices." And Chase certainly sacrificed a lot for the good of the project. And yeah, I think he'll always be thought of fondly in Mozilla.

So there's certainly some truth to that. I think different people saw different paths through—I mean, a lot of people shared the desire to know what was going on better. And we've improved that I think quite a bit since that point. You know, it would have been, you know, good if we could have figured out how to, you know, save Chase, as it were. Absolutely. He was fantastic to work with, and it was certainly a loss. I don't think it indicated a terminal condition or critical condition in the organization. There was certainly inefficiency that came out of it. There was certainly a lot of energy and discomfort that was—energy that was, you know, wasted or misdirected. And a lot of people weren't as confident and comfortable as they could have been with what was going on.

And part of it was just there was a huge number of new variables. There was the corporation-foundation split. There was this, you know, we had a product that had tens of millions of users that we had to support. We were in a serious competitive environment. We had revenue coming in we had to spend or figure out what to do. We didn't have to spend it, but we had an opportunity to figure out how to use that money to further the goals of the project. We had an organization that was growing and, you know, how do people come into that and what does it mean to be an employee with respect to the relationship with the project? What are our relationships with these other projects that are going on that aren't products? I mean, a huge number of decisions.

And any one of those things can just consume you, right? You can end up on the rocks of any of those pieces pretty easily. And to me it's amazing in some ways that we didn't end up in more—you know, we came through that as well as we did with the rate of change that was there. And there wasn't really a model for us to follow, right? There was no—we couldn't say, you know, "We're going to do what these guys did." You know, you're doing a start-up and you're pitching to people, whether they're people you want to hire or people you want to have invest, you say, you know, "It's like this with a bit of that," you know. But it's a consumer play. And you can sort of build in terms of these models.

But we've never really been extremely comfortable in determining what we're doing by analogy. We're getting more comfortable with it now and there are more models to choose from, but to a large degree people had to every day, you know, make those decisions from scratch. And just because of the demands on people's time and the rate of growth and just people wanting different things, there was a lot of opportunity there for people to feel lost or certainly to get burned out. And I think some of that was in play with Chase. I can't imagine he wasn't burned out given, you know, what he was doing.

And I think also, you know, if the environment he was in wasn't one that was healthy for him, then it was absolutely the right thing for him and for the project, to do something else. And that's a decision that, yeah, is only his to make, and that he's willing to talk about why and what the characteristics were there, and what the issues were for him, and what it cost him and what he would have wanted instead, I'm glad he was able to do that. And I'm glad that we as a community and a project were able to talk openly about a lot of those things as well.

You know, it was news for a little while that we were in huge trouble, we were going to hemorrhage people, was it the end of the road. And you worry that it might be because you never really know what's going to happen tomorrow. But I think most of us were pretty confident that we could solve these problems, and maybe not as well as we might have in hindsight or maybe not as early as we might have liked to, but we've improved things a lot since then. I think Chase would agree. And what didn't kill us made us stronger, so. Yeah, it's unfortunate. I liked working with him a lot. He's a great guy. I have no reason to believe he's not still a great guy.

Q: How would you define a successful open-source project, just sort of broadly if you want to just talk about Mozilla or just kind of broadly? What elements and practices do you think are sort of necessary to building a successful project?

A: I think the most important aspect of succeeding in an open-source project is managing expectations. I think that there are lots of different and legitimate ways to run an open-source project. I've been lucky enough to work with a bunch of different kinds of them in the past, and some of them are, you know, very much like Mozilla where it's open source but it's also a very collaborative development and it's very distributed and there's a lot stakeholders and so forth.

And I've worked in others where it's really, you know, a group of people working as they might in a proprietary software, and they put the code over the wall, their most recent release, or a previous one, under open-source terms. And they do that because there's a market demand for it, or it lets them take advantage of some other tools.

I think those are both successful in meeting the goals of the project itself. I think from the perspective, if you want to build a broad, you know, sort of Mozilla-like project or Apache-like or Eclipse or Linux kernel, that have a large set of contributors and build a common open-source artifact that's the primary place of development and so forth. I think again a lot of it comes into managing expectations of, you know, how are decisions made here, what are the goals of the project, what happens if you disagree, right? That's sort of the—those are the things from which you can I think, derive if not a lot of the behavior of the project, at least how that behavior will be determined. Is it expected that it's very democratic? Are you trying to build something that's perfectly general? Are you trying to build something that's really focused? Are you trying to solve a specific problem or to, you know, do something entirely new or exploratory?

And I think a lot of those characteristics are shared with, you know, how do you create a successful software project? How do you create a successful project of any kind? And, you know, in deciding to do an open-source project, an open-development project even, the first piece there is, you know, “Are the things I'm trying to do made easier or better by doing it in open source?”

Some things—I don't believe that all software needs to be, or should be currently, built as open source. The game industry is one classic example. And, you know, as software becomes more economically similar to a service, we'll probably see more and more of that change where there's value in building this infrastructure and selling a service on top of it and letting people maintain their own pieces of it. And it will become a competitive issue as, you know, open standards and protocol and interoperability have been for some time.

But I don't think it's necessarily the case that you sort of—you should start from the point of, "I want to create an open-source project to do X." I think you need to start from, "I want to do X, and I want to have these be the characteristics of what I'm going to get, and does open source fit that?" I mean, I take as a value in a lot of the stuff that I do that it'll be open source, whether that's Mozilla or otherwise. And I prefer to work on those things. It will be easier to get me to take a job that involved doing stuff in open source than doing stuff that wasn't, all else being equal. So that, you know, might be a choice you make. "I'm going to start this company. I want to build this system. If I make this core piece of it open source it'll be easier for me to attract some of the talent I want or to build a development community around it," and so forth.

But open source as its own sort of primary good, I think is probably a—even if it—it may well be, you know, a very virtuous thing to pursue. I think starting from that point is probably pretty risky, especially people who haven't done open source before and don't really understand all the different knobs you can twist there and the ways you can work. I think people often come in and say—when they have open-source project in their mind, they have something probably very similar to Mozilla or the Linux kernel, GNOME, that sort of thing, and they imagine all these contributors working together and all this software happening and so forth. But they don't really think about why they want that to be the case or how much cost they're willing to bear to get that or to know what costs there are in, you know, decision-making bandwidth or in, you know, showing all of what you're doing to the various competitors. Or making it easier for people to make modifications to the software you might not want.

The GPL is going through stuff related to this in its Version 3, which could be a whole other hour of tape. But one of the things that's been proposed for that is that nothing under the GPL, the third version of the GPL, can have characteristics that limit the users' rights to, you know, fair use of content and so forth. And I think that's a pretty dangerous slope, a risky move for a copyright-based license. But I think it's also a case where it may be that you have to give up something you can't afford to give up in the context of your project in order to get open source to happen.

And there's certainly I think, a lot of underestimation of how much energy goes into coordinating open source. It's probably on par with proprietary software development without the ability to fall back on the "I'm paying your salary" stick as often. So you need I think, slightly different kind of leadership sometimes. But proprietary software development's no piece of cake, either, and people routinely underestimate that. So I'm not sure it's really an open source specific thing, though your failures would tend to be more public. You need to be sort of able to—you need to have sort of the emotional constitution to deal with that as well.

Q: Do you consider open source software a public service?

A: That's a good question. I think people work on open source software for a variety of reasons, one of which is that they feel it's a public service, it's a good thing to do. There are lots of jokes to be made here about some open source software as a public disservice, but some of the people might hear this interview and that's just not a good scene.

I don't know that it necessarily has to be. I mean, open source doesn't necessarily even end up always being publicly available or useful. But I think it certainly—I think you can do a lot of stuff with open source that makes it easier for software to be a public good in a way that a park might be or a way that, you know, a poem or some kind of invention or, you know, recommendations for health can be a public good.

I think it's interesting because software is so critical to a lot of what people do day to day, that people having some fluency in that is important. I think it's important for people to understand economics. I think it's important for people to understand, you know, the way that markets operate, the way that democracy operates, the way that, you know, basic medicine, a lot of these things. And I think software is something that's pervasive enough and powerful enough, both powerful in the sense that it can affect what you're doing a lot, but also powerful in that people can—it's highly leverageable. You can do a lot with software just by, you know, thinking about it in the right way, basically. You're making stuff out of pure thought stuff. And I think that's—the open source culture isn't just about altruism there and it's not just about providing the resulting software. I think it's also about providing the value of the process and the ability to understand those things in a different way.

And, you know, it's a long way from, you know, where we are today to people learning about, you know, software the way they learn about history and geography in schools. I think I'd probably have them do economics in the sort of philosophical sense first. But I think it is a side effect of—I think it has been a side effect of free software and of open source that it's made software more visible. It's let people have better expectations and understanding of what software can and can't do and of how it's put, and how it's a reflection of the people who build it the same way, you know, any other artifact is.

Some of the original manifestos behind free software were around 'not trapping' the user. That if you had software to use some piece of hardware, a printer driver is a common example, you should be able to get the source for that if you got the software itself so that you could maintain control over it. It wasn't always about, you know, putting it in the public or being a common good or, you know, a basis for some sort of neutral software foundation for the Internet or whatever else. And it sort of evolved into that. And I think it's much more valuable in that sense than just in consumer protection. But it's certainly not—I think for a lot of people it's not just about providing a public good. I think it's nice that you can get a good out of it in a public sense even when people contribute for what could be very selfish motives. I guess that's, you know—Adam Smith would tell you the same thing, right? It's not to their charity that we owe the software we have, but to their own enlightened self-interest.

Q: What, if anything, do you think the popularity of Firefox will do for open source as a whole?

A: I think the most important thing—so there are two things. One is that it will expose people to the concept of open source, whether they're, you know, aware of it. Whether they see it as a good thing or a bad thing or they don't care. But just that there's a difference between software you have the source for and software you don't have the source for. So you might call the latter "hardware."

But I think one of the biggest things and one of the more short-term things is, it's showed that open source can produce software that's for real people. There's been this sort of albatross around the neck of open source software, especially stuff that is user facing, that open source software can only be built for power users, for geeks. It can't be built in a way that—you know, you'll never get good design out of it, even if you get good technology. It just can't be built that way. It can't market itself. It can never reach these people.

And existence proof is pretty powerful. So you can argue a lot whether Firefox's, you know, user experience is better than some other program's user's experience. But I think it's showed that being open source isn't the barrier there. Turns out it's hard to make good software that is pleasant to use. And it might be harder to do in an environment that's filled with the kinds of people who were originally attracted to open source, just because of their sensibilities. Not that they're lesser folk for it, but because it's not where—not from the perspective from which they come toward building technology, around usability and so forth.

And what we've seen, you know, in the last couple years in the open-source community, a real interest in the usability of their software. It's become the holy cry that that's what matters, is the software's useable and simple and so forth. And the pendulum can swing too far and people will self-correct there. But I think that's been a signal value for open source projects everywhere to demonstrate that it can be good software and that it can be marketed and that it can reach real people. And that it can compete with, you know, one of the most entrenched majorities, monopolies, in all of modern software, and it can affect their product, right? We, in a lot ways, made IE7 happen. So, you know, not a lot of commercial software has been able to do that.

And yeah, I think, you know, certainly there's lots of soft introductions to building open source that come around Firefox, around extensions, around being able to redistribute it and make little modifications and get it—you know, it's into schools. It's into environments where it wouldn't be if you couldn't do your own thing with it. But I think just people being aware that similar to, you know, knowing that there's a choice of browser and what that means. Knowing there's a choice of—or it's a characteristic of software. People understand, you know, how long is it supported? People understand does it run on Windows or Mac? People understand, you know, what does it cost? Or do I pay now, or do I pay—you know, is it a service like an online game? Do I get it upfront? Discounts? Upgrades?

They understand a lot of things about how software operates and interoperability and so forth, but a lot of people don't understand that there is this difference in different kinds of software. And it may not matter to them at all. It may not matter to somebody that their software can also run on a Mac. They use Windows; they're happy with that. But they're aware that this choice is there, and when it does matter to them it'll be that much more likely that they'll be able to take advantage of it and make the right decision for themselves, which may well not be open source but would be an informed decision. And that's a pretty big step forward, I think.

Q: Do you think that open-source techniques can be applied to other means of production? Or do you see examples of that in today's society?

A: Yeah. I think, one classic example is Wikipedia, which applies a lot of the same open source principles to collection. I can't really say "creation of knowledge," I don't think. I have a philosophical problem with that. But with collection, aggregation. Certainly adding a lot of value to human knowledge. That's one example. And that might be enough of one.

I think the areas where open source thrives are where the cost of goods are pretty low, especially the cost of goods for someone to become involved. Which isn't to say that's the only case, but there are people who work on open source software together who, you know, require some expensive MIDI keyboard or some rare piece of hardware or service that's expensive and that kind of thing. But I think we see less of that.

And then sort of one question that sort of follows into that is what are open source principles? One, you sort of take it to the degenerate case, and open source is about licensing. Open source is something only lawyers do. Programmers don't really do it. It's about the terms of use of that software. And so from that perspective we see, you know, user-serviceable parts. We see a return to those. We see people being able to, you know, mix and match parts on their bikes. You know, picking their provider. Being able to do these sort of—and you see it in deregulation of industry, you know, power and telephone. You know, that's similar in that the user has a meaningful choice there. The right to modify things. We're not seeing as much where it sort of reaches the consumer. Take the cover off your Xbox and do something to it. And it depends on what jurisdiction you're in whether you can actually get away with that.

But in a lot of areas I think it will make people want to have that control and that flexibility over other things that are in their lives. I think we'll of course, see it—I say "of course" because I'm obviously a futurist in control here—in things that are, you know, like software. You see it in things that are effectively software wrapped in a box. The Linksys router, people will load their own software onto. And the production of that system is, you know, built on open source principles there.

You see some of it also in, you know, community networking. And, you know, whether that's open source principle or that's just, you know, good old-fashioned community spirit. It's not as clear what the right—some of the terms of art in open-source software are around the right to modify and the preferred form of modification, making source available at the same time. Is that, you know, sending the schematics along? Is that—you know, how much detail is needed there? And to be fair we haven't really locked that down for software. Everybody basically understands what it means and the hard cases we just ignore.

So I think we'll see some of that. I think again that will come out of people being more aware that, you know, it is a meaningful choice to decide whether you're going to get something you can tinker with yourself or have control over, or pay somebody other than the vendor to fix, to work on. And that's an area that, you know, has had a lot of profound competitive impact from PC versus Mac to different types of, you know, [indiscernible] power supply and AC versus DC and those kinds of things.

Q: Last question, and it's extremely broad.

A: Oh, good.

Q: What do you think the future of open source is?

A: Well, yeah, that is pretty broad. I think the future of open source is people picking and choosing more of these techniques and elements into different things they do. I think the future of open source is people who are not software developers learning how open source affects them. I think to some extent, or maybe I just hope to some extent, the future of open source is people demanding more of its characteristics in other things they do. [Users] [indiscernible] parts in people deploying complex systems or—which will likely be software at first. But having enough information about them and the rights to modify them. Maybe to share those results.

I think the future of open source will also be things like Wikipedia, which are a collaborative development of non-software. Probably even tangible sort of soft-goods things at first because again the cost of goods and replication are low enough to reach there. I think open source is also, even more than it is today, going to be an important tool for the sovereignty of people's computer experiences. We talk about the sovereignty of the Web experience, that it should be in my language, right? Whether I'm, you know, living in California or Papua, New Guinea, I should be in control of the software that's there. It should be in my language. It should adhere to my cultural customs.

You know, most of the content on the Internet, yeah, that might still be in English. But there's an important element of sovereignty and control there, and I think open source allows a lot of that. It allows people and nations and organizations to take control. Not a zero cost. You know, it's not a perpetual motion machine. You don't get this stuff for free in the sense that you can't wish software into being. But you have the option of making it what you want you it to be, and you get to decide about the economics of making it happen or not.

And people choosing to collaborate more on different types of development and improvement. The interesting thing about a lot of open-source licenses is that they protect the rights to—a lot of them do—protect some rights to derive value from things that are derivative. So you make changes to what I do. I can use your changes. They don't all do that. Depends on what the goals of the software license are. But you see—you know, it sort of echoes some of the original tenets of the Western patent system for all it may or may not have gone off the rails, that I will share this and I'll give this contribution and if you're going to use that contribution then I get some rights to what you do. In this case, in the patent case, usually a royalty example.

We're starting to see that in patent pooling. We might see it in pharmaceuticals. We might see it in a lot of educational areas. And I think the other sort of thing that I hope open source continues to do is to demonstrate the different ways people of different backgrounds and interests can collaborate to produce something of mutual value. And that they can do that in a way that still lets them compete in a lot of ways. I mean, Google and Yahoo are both Mozilla partners, and valuable ones. Are they competitors? Absolutely. Do they understand the value of making sure that the Web continues to be where the fight is? Absolutely.

And they can collaborate on this software together and they both get value out of that, and they get value out of each other's work and they're able to—and a lot of companies probably wouldn't be able to, you know, sort of put that aside. But they're able to participate in that, and I think they get a lot of value from that. I think showing those models and showing the value of transparency of that process, that people should—you know, people want to be able to know, you know, how that sausage was made.

And we talk about that in the context of security a lot, right? A lot of people—very, very few of our users read the source code to Firefox, and even fewer of them could find security bugs that way. But they get benefit from everybody who does. And I think that's a model that can apply to a lot of—could apply to policy-making. It could apply to, you know, collective bargaining. It could apply to any number of collaborative human endeavors. And so I hope the future of open source is beyond just software, and I hope that the software that it does produce is interesting to more and more people, and that people discover ways to be involved that don't involve writing code, and that more of those ways are created with things like Spread Firefox, with things like localization communities. And, you know, nobody knows where the next one of those will come from.