Olivia Ryan: It's June 26, 2006, and could you state your name please?

David Baron: David Baron.

OR: And David, when did you first begin using computers?

DB: When did I first—I don't know exactly how young I was, but I was pretty small, like probably somewhere in the range from three to five.

OR: Do you remember the first programming project that you worked on?

DB: Not specifically. I mean, it also depends what you count as a programming project because, you know the first computer I used when I was little was a Commodore 64, where, to some extent, anything you did was a program. You'd, you know, write a few lines of Basic, and it's not really a programming project in much of a sense, but I guess it was—the transition to working on larger things was so gradual that I don't know where to draw the line.

OR: Right. Were you—do you also have formal training in computers, or is it that you are all sort of self taught?

DB: I do have formal training. I mean, I was a computer science major in college, but--. I mean, I did learn a lot of things, but the actual—I guess I didn't learn a whole lot about things like how to maintain a large program. That I learned from actually doing it rather than being taught it.

OR: Mm-hmm. And when did you first begin turning to Open Source projects, or how did you kind of get involved in the Open Source community?

DB: I guess Mozilla really was the first Open Source project I was involved in because I actually—I got involved with Mozilla through my interest in web standards, which had been around—which I'd been interested in for a few years before that. So, I was involved in a group that was part of the web standards project in 1998 that was writing essentially reviews of CSS support in browsers and other things like that. And I learned about the new layout engine that Mozilla was working on, although they hadn't actually switched to it by that point, and started testing it and filed bunch of bugs and, I guess, started off with just a few, testing a few little things.

But I guess one weekend I sat down and went through a bunch of tests I had and filed like 15 bugs in a row. That was back when it was Bug No. 900 something or 1000 and something, rather than Bug 350,000 like we have now. So, you know, one, I sat down for a few hours and I'd file more than one percent of the bugs in the bug system.

OR: So what do you do here at Mozilla now? Kind of take us through the different projects that you've work on.

DB: Well, I guess, what I've worked on actually hasn't changed all that much. It's—I mean, I've been working on the layout engine in various ways, so, how we handle things like CSS and layout. Some things in other areas, domain scripting and memory leak work and things like that. But I wouldn't split it in—I don't see where you just split it into projects so much. I mean, it's a lot of maintenance and development of new features and debugging of various things and, of course, lots of reviewing code and other things like that.

OR: Do you generally work alone or within groups?

DB: It's—I think it's some sort of weird combination of the two. A lot of programming is, you know, you sit down and figure out why something is happening and then maybe write up a patch for it. Then, you know, once—I think there's a lot more work in groups once you've figured out what's wrong with something and there--you have to decide what the right way to fix it is.

OR: And how does that decision-making process happen? How do you sort of figure out who's going to work on what?

DB: To some extent, there's not really a decision-making process. There's, I mean, for many contributors, they, people work on what they think is important and, to some extent, I do that too. Forgetting what else I was going to say. What was the question again?

OR: Just generally, kind of maybe how the division of labor is determined?

DB: Yeah, I mean, a lot of it is what people choose, what people think is important. Sometimes, there's more—in some areas, there's more organized management of saying that a particular feature is a priority. I think that's more the case in the front end than in the back end. And then a lot of stuff is responding to things other people are doing. So, somebody else is working on something and they need advice or they need code review, and even if you don't think it's that important, you still kind of have to do it eventually.

OR: Mm-hmm. How much sort of communication and coordination is there among people working on the front end to those working on the back end?

DB: Sometimes not a whole lot, although I don't know that there often needs to be a whole lot. Though, in the cases where it does need to exist, I think it sometimes doesn't exist there either. I mean, it also depends what—there

are some features that involve both front end and back end components, and if they're not being done by the same person, then, yeah, there would be much more detailed communication. But communication between the people who work on how we display web pages and the people who work on writing an application on top of XUL and JavaScript, it isn't that huge some of the time.

OR: How do you generally communicate with those who you're writing with?

DB: Probably the biggest form of communication is comments in bugs, as bizarre as that may seem. Also, e-mail; also, IRC. It tends to be very easy to use IRC, even when it's not the right form of communication to use, and I sometimes try and force myself not to because, you know, before I used IRC, I would frequently send e-mail messages back and forth to a few other people at a rate of one per minute. And, you know, that worked actually, and it also works when that person goes away. And with IRC, IRC isn't very tolerant to somebody just walking away from the computer 'cause you don't know if they're going to see what you wrote, whereas e-mail is considerably more reliable.

OR: Because there's a record of it?

DB: Yeah, because you know the message is going to get there.

OR: Yeah, 'cause nobody is going to miss it.

DB: I think these days people sometimes tend to use IRC too much and not use e-mail as much and not use mailing lists as much. The advantage of people communicating over mailing lists and it's—to some extent, this is how I learned a lot about Mozilla, is that if the developers are communicating on a mailing list that other people can read, then somebody might just decide to read that and gradually pick up what they're talking about and understand what they're talking about.

So, when I was involved in testing web standard stuff in Mozilla, I ended up reading the mailing lists where a lot of the original developers were discussing things, and I kind of read that and didn't necessarily understand all of it, but gradually picked up what they were talking about. And then when a bunch of them all left very near the same time, suddenly I was the only person who remembered it.

OR: Mm-hmm. Do you think that there should be an effort to sort of standardize what ways you communicate with one another because there is kind of like, stuff that gets lost? Or do you think that's kind of an impossibility?

DB: I mean, I'd—there have been efforts at various times to get people to communicate more in the mailing, in mailing lists, which are mirrored to newsgroups rather than other things, because it's public, and it allows other people to get in. For example, many of the reasons I just described and also for other reasons, too, in that other forms of communication have the tendency to leave people out who should be involved, especially if some people are working in the same building or the same time zone and others aren't.

So, I think there should be efforts to encourage people to use certain forms of communication over others, but not to force them. And there are just some things where it's much easier to do in person. I think, actually, reviewing code is one of those. I find it's much easier to review patches when I can just ask the person who wrote the patch to explain it. Then, given that explanation, it's much easier to understand what the code changes they have are.

Now, working remotely, sometimes people do that in a bug when they request review, which, you know, they make a change in Bugzilla and it sends me e-mail; but, sometimes they don't, and it's often hard to figure out what happened. I try to do that when patches are complicated. I'm not necessarily very good about it either, but--.

OR: What about comments written in the code? Do people generally comment using areas of the code or--?

DB: Some areas of code are well commented, some are not. I think there's a tendency—commenting confusing areas in code is one of the things that comments are really useful for. One of the problems there is that it's hard to know what other people will find confusing, and I don't think too many people are good at that at least until somebody gets confused by code that they wrote six months earlier. And that's the point where it's obvious that something needs a comment.

But I think trying to comment everything is way too much work. It's both way too much work, and it leads to comments ending up disagreeing with the code because somebody fixes a bug in the code and, if the comment is bigger than the code, they might not find the equivalent bug in the description of the code, hidden in the comment.

So, writing too many comments both means that it's hard to find the important ones and that they get out of sync with the code. It is often useful to have general descriptions of what things do and descriptions of invariants that people might not think about. Although, to some degree, invariance are better described with assertions, which are in some sense

another form of commenting, but one that actually catches when people ignore them.

OR: Would you say that strict ownership of specific areas of the code is something that's enforced?

DB: Sometimes more than other times. If there's a strong owner or a set of strong owners, who actually are involved in the project on a day-to-day basis, I think ownership often is reasonably well enforced, although not completely. In areas where there isn't anybody currently working on the code, there might be nominal owner who's not as involved anymore, I don't think ownership is enforced all that well.

OR: And if somebody, if like an owner of a specific area leaves, leaves a project or whatever, is that person necessarily replaced or does somebody just kind of become the owner through a particular process of—or kind of organically, or something?

DB: It's varied between both of those. When something is considered a critical feature that needs work generally by some level of management in a company working on--that's contributing to Mozilla and building products off of Mozilla, then often they will immediately put someone else to work on the same area, and that person will effectively become the owner, at least if they're competent in what they do. Sometimes it is much more organic in that other people will start fixing bugs around the edges and eventually understand the code and kind of just officially, in the end, officially become recognized as owners.

We have an official list of module owners, which is generally maintained only after the fact. In other words, there's not an active effort to maintain the list but every so often somebody looks at the list and says, "That's wrong. This person doesn't own that any more. It's so and so." And, in fact, every, maybe year or so for the past few years, or maybe even six months, I actually try to go through the list and propose changes to Brendan, and he goes through them generally in consultation with the people who are going to become the owners, although not always. Sometimes, people just become owners of things without their advance permission or knowledge, but essentially recognize, just recognizing reality, that they're the person who maintains the code.

OR: And how is that person then informed? They're just sort of told, "By the way, here, you're the new owner now."

DB: Yeah, by the way, we changed the page. You're now the official, officially the owner of DocShell. Or I think there might have been cases where we

forgot to tell them and, you know, they looked at the page a few months later and noticed that they own something else.

OR: But they're, in effect, already functioning as the owner?

DB: Yeah, or as one of the people responsible for it. Now some of these people might be people who work on different areas of the code and if they work on one area, even if they work more on another area that has other people working on it and less on an area that has nobody else working on it, they might end up the owner of something that they didn't work on all that much. I think that, like, maybe happened with your uriloader or, I think, Boris Zbarsky, well, no, I don't think Boris is the owner of that. Boris Zbarsky is one of those people who works on just a lot of different areas, some of which were effectively unowned, so he kind of ended up, I think, being owner of some things that he doesn't work on a whole lot. I think that might have also happened with things like PostScript handling and some other graphics and platform-related stuff.

OR: How would you, how would you sort of describe your programming style in relation to other people's style? Or maybe you'd put it another way—have you ever clashed with another person, another developer over a particular point of code and, if so, how did you sort of resolve your differences?

DB: Clash kind of sounds like a strong word. We disagree a lot, but I wouldn't use the word "clash" to describe minor disagreements. I think, in general, I tend to be one of the people who's a little bit more perfectionist in that I tend to work more slowly on things and not—and try to land things that are tested by the time I'm landing them.

Lately, I think I've also developed a strong bias towards having less code rather than more because I think it's very easy to fix bugs by adding code even if they could also be fixed by removing code. And you end up much better off in the future if you fix them by removing code. I think, I mean, probably the types of things I've had disagreements with other people over were, you know, whether it was worth adding a lot of code to do something. In other words, is adding 200 lines—is it worth adding 200 lines of code to fix some bug, even given that we have the 200 lines of code written and tested right in front of us.

I sometimes think that we shouldn't take the fix if the bug is obscure enough because that 200 lines of code is going to need to be maintained in the future, and people are going to need to understand what it does repeatedly every time they have to make a change in code that's related to it. I think a lot of other core contributors actually think the same way, but sometimes people contributing patches don't necessarily realize how high

the maintenance costs of additional code are and, therefore, expects that if it fixes the bug, therefore, it's good and should become part of Mozilla, even if that's not necessarily the way we see it.

OR: To what extent would you say Mozilla relies on the work of volunteers?

DB: It's hard to know how to answer that, but we rely on volunteers for tons of things. And what would happen if the volunteers disappeared, a lot of things would kind of fall apart, especially testing. We rely on volunteers a huge amount to just, to find, especially things dealing with web compatibility because there's so many pages on the web, and when you break something, somebody will usually notice. If we didn't have thousands of people testing nightly builds, we'd have to have better regression testing, and we probably should have better regression testing anyway so that we don't break the nightly builds so much and so that we have more confidence in making changes at the last minute.

But, to some extent, having so many users who are also occasional bug reporters has kind of prevented us from needing to do things that we probably really should do anyway. But we also rely on volunteers for a huge number of other things. Significant parts of the code are maintained by volunteers. There are also significant parts of the code maintained by people who work for various companies, Mozilla, Google, IBM, Sun, Red Hat and other companies.

Ken Albers: Did you start out as a volunteer when you originally were submitting bugs then?

DB: Yeah, I started out filing bugs. I ended up doing a summer internship. Essentially, after working, after filing bugs a good bit for about a year, and then I did a bunch of summer internships at Netscape. I also worked there for a semester off and then worked there briefly before I—briefly after I graduated, before they laid off the entire browser team.

KA: What sort of led you—why did you go and volunteer? What lead you to it, to do that?

DB: So, I initially started filing bugs because I was interested in web standards and interested in improving CSS support in web browsers. So, I essentially found the things that I thought were important to fix and filed bugs on them. And when I realized how responsive the developers were, then that encouraged me to keep filing more bugs because I realized that if I filed something it would actually have a pretty good chance of getting fixed. You know, this was back when there were a thousand bugs in Bugzilla and, you know, filing another bug—there weren't that many people testing things, so if you filed another bug then the developers knew about that

bug, and if they knew about it, there was a much higher chance they'd fix it than if they didn't know about it.

As there have been more and more bugs in the bug system and more people filing bugs, it's, essentially, that's become less effective because now there are so many bugs in the system that I can't keep track of all the bugs in the areas that I work on. They're in the system, many of them probably have—they're often invalid. They're often not clearly enough described, not well enough described to understand what the bug is. But essentially, when I started it was just kind of like walking over to somebody and saying, "Hey, did you notice this?" And they could respond quickly 'cause there weren't a lot of other bugs that they were getting e-mail about. So, seeing how quickly things improved when I filed bugs on them really just encouraged me to keep going and eventually get more involved.

OR: Do you think that the process now for volunteers sort of—gaining entry has changed because the community is so much larger because there are so many more bugs, because it's easier to kind of get lost in the shuffle or something?

DB: I think some things about it have changed a lot, but I think other things about it are really still very similar in that a lot of it is based on doing work that people notice and recognize so that other people start to trust you to do things. Maybe today you need to be a little noisier about it sometimes, but I think it's still—and the mechanisms for getting through to people have changed. But I think, fundamentally, the process for getting involved is still—do something to prove that you know what you're doing and let people know about it.

The problem is that, you know, for every one person who does know what they're doing, there are also a bunch who try to get involved and try to work on things but don't necessarily know what they're doing, maybe because they're trying to work on code that they don't understand well enough, or they're trying to work on--. I think some, in the areas I work on, it's more often that they're working on things where they don't understand the requirements well enough, because the requirements are this whole complex mesh of web standards that are all related to each other in complex ways and that it really takes a lot of reading and a lot of working with to understand.

So, there is always, there's the issue for, you know, new people getting involved. There are always the people who are trying to get involved but aren't actually doing work that we like and, you know, saying that the thing to do is be persistent is maybe a little misleading because other

people, people can be persistent when they're not doing good work just as much as if they are doing good work.

OR: Right. You mentioned the mechanisms for getting involved have changed. How so?

DB: I think people, well, people have started to use Bugzilla differently as Bugzilla has evolved. A lot of the features that we rely on every day in Bugzilla didn't exist eight years ago, so, you know. Now, code review is done through these request flags where you set a flag in a drop-down and then type somebody's e-mail address. Six years ago, if you wanted code review, you sent somebody an e-mail, just wrote an e-mail message. You didn't use some magical field in Bugzilla. So, to some extent, we've started to use tools for a lot of things that used to be done with e-mail, as people have seen the need for tools to make very common tasks easier. But it also makes it harder for new people to get involved because they have to learn this complex set of tools that we rely on.

So, just knowing what to do if you have a patch and want to get it included is, perhaps, a little harder today. Although it's also, because patches get attached to bugs, easier for other people who notice the patch being there and the process not necessarily being followed quite correctly. It's become easier for other people to notice and correct because it's also, I guess, more public. Although, again, that would depend on whether the communication six years ago was done via private e-mail or via a public mailing list.

OR: I understand that during that early development of Firefox that CVS access was limited to a small team. Can you sort of discuss why that was and how our access has—sort of, the restrictions have changed over time?

DB: So, you're talking about the early development of Firefox, meaning 2002, 2003?

OR: Right.

DB: To some extent, I'm not really the person to ask about that because I was never that involved in the front end stuff, which was the area that I guess was restricted for a while.

OR: Were you involved that development of the back end during those years?

DB: Yeah, I was mostly working on stuff that was independent of whether it was Firefox or The Suite, so I wasn't all that involved in a lot of those issues.

OR: Do you think that having kind of a small team of people working on the UI enabled them to sort of work at a faster pace and produce the successful browser that it is now, or how you think that that may have shaped the development?

DB: I think to some extent it does because what can happen to people is that if you're opening up development to a large number of people, especially in something like the front end where the code is written in JavaScript and XUL and it's relatively easy for people to understand, the problem that can happen to the lead developers is that they get so many contributed patches that they spend most of their time reviewing code written by other people rather than doing their own development. And in many cases it's more work to review a patch than to write it yourself. And the reason to want to review it is to get new contributors who will eventually be able to write it themselves much more quickly and be able to make larger design decisions about the code. So, there's a tradeoff there.

I think sometimes the lead developers do need to just say, "I really need to get this big chunk of work done, and I'm going to ignore other people for a bit." And if that's what it takes to have the people who really do understand the system writing code, which I think is often good for maintenance certainly, because you end up—they're more likely to fix things that are broken than to work around them, I think, and to understand things like what patterns or interfaces have caused problems repeatedly in the past and fix the root cause of problems more rather than working around the specific instance.

So, I think it is important to have—I've kind of wandered off onto another topic, but I think it is important to have the, kind of the lead engineers who really do, who have been working on the project for a while and understand what problems things have—why things are the way they are and where things are supposed to be going, actually working on code a good bit of the time.

I think what the Firefox group did at the beginning was, maybe, too extreme in that direction, and they did end up backtracking on it a good bit, maybe even too far in the other direction today. But I think, to some extent that has to be done. You have to—if you want to be the maintainer of something, you actually have to work on it some and not just review other people's work on it. Although, there are exceptions to that, I'm sure, but--.

OR: Why do you think Firefox has been able to attract so many users?

DB: I think--there's a bunch of reasons, I think. It has been designed to provide a better user experience and doing things like pop-up blocking and tab

browsing that users really do want and that make browsing the web easier, attracts users and encourages them to switch. I think there are also a series of security problems with Internet Explorer that also encouraged users to switch. And I think both of those are good reasons to switch.

I think the security problems there's actually two underlying issues. One is that I think there actually are some security advantages in Firefox, but I think the other one is that it's better to have a more diverse set of browsers in that if you're using the browser that 99 percent of people use, you're more vulnerable than the browser that 1 percent of people use even if the browsers have identical, or not identical security problems, but identical levels of security problems just because people are more likely to go after the 99 percent than the 1 percent. That's not to say that's the only reason, but I think diversity of the web browsers that people use is also good because it helps keep the web open.

And I think when, from a web standards perspective, seeing IE have 95 percent market share was a big problem because it meant that web authors were writing only to IE, and anyone who wanted to write a viable web browser had to reverse engineer things from IE effectively by looking at whatever feature—but not necessarily all of IE, but looking at which features from IE sites we're really relying on and then implementing those features. And the more of a monopoly it has, the bigger the pressure to do that becomes, and developing a web browser just becomes catching up rather than doing anything new.

OR: What do you think sort of sets apart Firefox from other open source software projects?

DB: I don't know how specifically you're asking about Firefox versus Mozilla as a whole, but I think—

OR: You can answer both ways or you can or distinguish Firefox from other Mozilla projects, or talk about Mozilla.

DB: Yeah, I'm probably more qualified to talk about Mozilla as a whole or the layout engine part of Mozilla than Firefox, but I think, I mean, there aren't that many other open source projects that are as big as Mozilla in terms of size of code, in terms of number of people working on them, and also in terms of how noticeable they are to users. A web browser is a pretty important part of a lot of what people do with their computers, although I think many users don't actually understand that it's there, which is kind of interesting.

But, anyway, I think one of the other things that's distinguished it, maybe more in the early days than today, was that it was an open source project

that successfully managed both significant amounts of contribution from a company that was interested in producing products, and from volunteers. There are other projects like that, but I think it was unusual in that there was a very large percentage of the contribution from one company, from Netscape or AOL. But there was still, despite that significant contribution from other places, and I think that's one of the things that was interesting about Mozilla as a project.

In the past few years, that's kind of become less interesting just because the Netscape group was gradually moved off to other projects or laid off, and we ended up with about 10 people at the Mozilla Foundation, who essentially couldn't be as big a part of the project because they were only 10 people compared to the 200 some people Netscape had working on it at one point.

OR: This is sort of a broad question, but how would list Mozilla's priorities? And how do you think that compares to its priorities in 1998, even if you weren't there?

DB: Yeah, I mean, I was just starting to get involved peripherally about six months after it started, so I don't know a whole lot about the priorities then. But I think, I think some of the things are—well, I think at the beginning, maybe the priorities weren't all that clear at the beginning. Just because, to some extent, it was significantly driven by what Netscape wanted. So Mozilla as a project developing a set of priorities separate from the company that was providing, was paying three-quarters of the contributors was a little hard to do.

I think the reasons that a lot of the non-Netscape contributors became involved in the project are similar to the priorities of the project today, but I'm not sure how much those priorities could—how much, to what extent you could say that those were priorities of the Mozilla project as a whole in '98. Although, again, it's a little hard for me to judge.

I guess the priorities I'm thinking about when I'm saying this are, I guess, the top one is keeping the Internet open. In other words, keeping it based on open standards, keeping it—preventing it from being dominated by one vendor. I think something that's kind of a newer priority is, I think, trying to improve the experience for end users, which I certainly think should be an important priority. But, essentially, being user driven is to some extent a business model, and that really wasn't so much Netscape's business model. They were—they made money off of advertising, so they were trying, I guess, to—this, you know, is really mostly an outsider's perspective on that, although I was an intern there for a bunch of summers.

But I think Netscape was trying to maximize the amount of money they got from their advertising revenue. And I think in many cases what Firefox was about was actually trying to produce the best browser for users. There are other incentives that we have, although I think the Mozilla Corporation's financial incentives are probably better aligned with making the product better for users than most other situations I could think of.

I guess those two priorities, well, the first one is kind of coming out of the mission statement of the foundation, that is keeping—I think that the official statement is something like promoting choice and innovation on the Internet. But we've had some discussions recently about what our big priorities should be, and I think the one that grabbed me the most was that we should be focused on what's better for the end user. And what's better for the end user is often what's better for other—when I say that, I'm thinking about that making things easier for authors is good for the end user because it means that authors have more time to produce whatever content the users are interested in and less time to worry about technical details of how to make it, and similar things for a lot of other areas.

But I think, actually, it's one of the things that was brought up in discussions and one of the ideas that I really like about how we should think about our priorities. But I don't—I think that one actually has evolved more recently, and I think was a big part of what originally drove Firefox, even at that time without any business model associated with it at all.

OR: How—do you think that open source principals have affected the way that Firefox is marketed? Do you see that the things like Spread Firefox and trying to get the community involved in marketing is sort of a similar movement as the open source software movement?

DB: I think there are some connections, but I don't know. I think open source software is, to a significant extent, about meritocracy. In other words, you learn to trust people who you think are good at what they do and listen to them and ignore the others. I think a lot of the market—some of the marketing things that are open marketing projects or, you know, community-driven marketing projects, are to some extent more about just getting the largest number of people involved, and that is the goal of marketing, so that's a good thing in that case. But I think it's—that does make it a good bit different from open source where volume doesn't necessarily help.

There's a lot of discussion of the concept of the mythical man-month, the idea that adding more people to a software project doesn't make it happen faster. I think, with marketing, that's based on tell your friends; having more friends, telling more friends does actually make it happen faster.

OR: Do you consider open source software as a public service?

DB: I don't know if public service is quite the right word, but I think, in many cases, open source software is a public good at least in that having people be able to understand how their computers work is, you know, it allows people to become interested in computers. It allows people to learn how things work, and I think letting people understand how things work is often a good thing.

I think, on the web in particular, I think there is some connection between open standards and open source. It's not a hard connection, but I think having open source implementations of standards allows people to see that they really are implementable and allows people to see maybe what's required that's not explicitly written in the standard and allows people to know that they're going to have a way of using the open standard even if some company stops supporting it later on.

I think one issue with data formats is that as you move to different machines or different operating systems, you may or may not have software that can operate on those data formats. And having open source software that deals with data formats means that you're much more likely to be able to get to that format on a different platform, on a different operating system, on a different piece of hardware, when those things become popular, even if the company that originally developed the software doesn't care about that platform. Because if it's important enough, somebody will care about that platform enough to make sure the program works, at least if the format is important enough.

OR: How would—this is sort of broad and you've kind of already touched upon it, but how would you define a successful open source software project, and what elements and practices do you think are absolutely necessary in order for a successful project to take off?

DB: I mean, it depends on what your measure of success is.

OR: Well, you define from that too.

DB: Yeah. I think the measure of success depends on what the project is trying to do and, for us, it's, success is being an important part of the web, which is the whole, I guess, people use the word ecosystem sometimes. It's the best word I can think of for it right now, and being a part of that requires continual maintenance and continual evolution. Success doesn't necessarily require continual activity, depending on what the project is.

But I think in many cases—well, I think in many cases, if there are improvements to be made—success, to some extent, is maybe that, if there are improvements to be made, people can figure out how to make them. That said, for something that's a very small program that does a simple job well, there aren't necessarily improvements to be made, and success can simply be being used and not necessarily being improved. But I guess, and I haven't really thought about this a whole lot before, but maybe the idea I just said, that maybe success should be, for open source, in general, should be measured by whether people who want to improve it can improve it. Maybe that's one measure. It's hard to know how well we're doing on that. I think we're—in many cases we're doing pretty well. There are some areas of code that are pretty hard to work on.

OR: What, if anything, you do you think the popularity of, in particular, Firefox will do for open source as a whole? Or has done?

DB: I think, well, I think one of the things it does is that the popularity encourages people to contribute to it, in that having a lot of users and having a lot of users who, hopefully will upgrade to newer versions, encourages people to work on things for the newer versions because they can make an improvement that really helps a lot of people.

Is it going to increase the number of people who know about open source? I think, at this point, not necessarily, because the difference between having 5 percent market share and having 50 percent market share—most of the people who know about open source software are going to be in the 5 percent rather than the 50 percent and, you know, as we increase market share, the users are going to be a lot of people who don't necessarily know what open source is and don't necessarily care. But the fact that somebody who is interested in computers and does want to improve Firefox or Mozilla or whatever is capable of doing that is valuable to that 50 percent. Or, and, that's you know, wild future projections, but--.

OR: Do you think that's a plausible future projection?

DB: I don't know. I mean, I'd like to see web browser market share not be dominated by one browser. Right now, we're kind of the second place in the market, in most market share data that I've seen. So, you know, I don't know if that's reasonable or not, but I think it would be nice.

Frankly, I don't think it would be nice to go too much higher than that because then you'd get Mozilla causing the same monopoly problems that other web browsers have caused. And I think that IE has, well, IE and, before that, Netscape have caused, and I think it's better to have more diversity. That said, I'm not sure if a situation like that is really stable, but it never has been in the past.

OR: Just the last—do you think that open source techniques can be used beyond software development? You know, like in other modes of—in other areas of production in society?

DB: I think some of them can. I mean, I think one of the things that we've gotten good at is working with people who live in other parts of the world, who are in different time zones and, you know, people who work on the Mozilla project for a while just learn to deal with the fact that some things take a day to go around 'cause somebody in New Zealand isn't awake at the same time as somebody in London.

And I think some of the things we learn from working in a very distributed way could apply to other areas. I think, you know, I think it's probably in some other areas people are telecommuting more and doing things like that more, but, I mean, there are obviously some areas where that's never going to work. But I think that's one area where things that open source projects are doing are, I mean, not that open source projects necessarily did them first, but can be applied to other areas other than software. There are probably a whole bunch of others, but I can't think of them right now.

OR: Okay, great. Thanks very much.