



Hanselminutes

Hanselminutes is a weekly audio talk show with noted web developer and technologist Scott Hanselman and hosted by Carl Franklin. Scott discusses utilities and tools, gives practical how-to advice, and discusses ASP.NET or Windows issues and workarounds.

Text transcript of show #91

December 6, 2007

Eclipse with Bjorn Freeman-Benson

In this episode Scott discusses Eclipse, Open Source and both the history and future of software with Bjorn Freeman-Benson. Bjorn is the Technical Director for Open Source Process and Infrastructure for the Eclipse Foundation.

(Transcription services provided by [PWOP Productions](#))



Our Sponsors

 **telerik**
deliver more than expected
<http://www.telerik.com>

 **nsoftware**
<http://www.nsoftware.com>

NET 
DEVELOPER'S JOURNAL
<http://dotnet.sys-con.com>





Lawrence Ryan: From hanselminutes.com, it's Hanselminutes, a weekly discussion with web developer and technologist, Scott Hanselman, hosted by Carl Franklin. This is Lawrence Ryan, announcing show #91, recorded live Monday, December 3, 2007. Support for Hanselminutes is provided by Telerik RadControls, the most comprehensive suite of components for Windows Forms and ASP.NET web applications, online at www.telerik.com, and by .NET Developers Journal, the world's leading .NET developer magazine, online at www.sys-con.com. In this episode, Scott discusses Eclipse with Bjorn Freeman-Benson.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes and I am privileged to be sitting down here with Bjorn Freeman-Benson, the Director of Committer Community at the Eclipse Foundation. Thank you, sir, for taking the time out of your busy schedule.

Bjorn Freeman-Benson: Yeah, no problem.

Scott Hanselman: Most people who listen to this podcast are Microsoft developers, although the audience is starting to expand a little bit. They know that Eclipse is a thing that's out there, but for the most part they may not necessarily experience it in their everyday life, but Eclipse is a huge part of a great deal of developer's lives from day to day. What should a Microsoft developer know about Eclipse?

Bjorn Freeman-Benson: Well, it's sort of a large question.

Scott Hanselman: It's a big question.

Bjorn Freeman-Benson: Eclipse is probably the most popular -- I think we actually have stats somewhere that demonstrate that it is the most popular Java IDE. Now, Eclipse has expanded to be more than just a Java IDE, but it's best known for being a Java IDE. So, since a lot of Microsoft developers work in C# and that sort of thing, they don't work in Java, they don't have that familiarity with Eclipse, whereas, people who build Java applications, web servers, rich clients and so on all know about Eclipse even if they don't use it, even if they use one of the other tools out there. As a developer, you might consider Eclipse to be sort of the Java version of Visual Studio. Visual Studio is the 800-pound gorilla. It's the main development environment that you use if you're a windows programmer even if you happen to use one of the other tools that are out

there. There's a wide variety of them. Everybody knows about Visual Studio. Same thing applies in the Java world. Even if you use one of the other Java tools, you know about Eclipse as the main programming environment.

Scott Hanselman: And where did it come from originally? Is there a Linus for Eclipse who is like the guy who made it and then it became a foundation?

Bjorn Freeman-Benson: The history of Eclipse started at this little company, Object Technology International, or known as OTI, in Ottawa. I worked for them for a while and we made Smalltalk programming environments basically. We made a deal with IBM even when we were an independent company and they sold it as IBM Smalltalk, which was known as VisualAge Smalltalk at the time. That same group after being bought by IBM went on to produce the first IBM Java programming environment, which was called VisualAge for Java, which was innovative, but fairly weak in its execution. It had a number of flaws. Then they went on to make a new version of that, which became Eclipse. Then IBM decided to take that proprietary code that they had in that and they were actually selling that for multiple thousands of dollars and so on and open source that for a number of reasons, but that became the Open Source Eclipse product that's out there. After a few years of that being just open under IBM's label, they decided to spin it out to a separate foundation, which is where I work today, which is a non-IBM thing. It's funded by the member companies of the Eclipse foundation of which they are currently 120-member companies, maybe 130, I'm never quite sure exactly what the number is, and they all put in varying amounts of money to help run the foundation, maintain the servers, provide the bandwidth, pay the salary of myself and the few other employees there are of the foundation, etc.

Scott Hanselman: So, a foundation like this, an open source foundation, isn't a behemoth. It's not a huge thing. I mean I'm sitting here in your offices in downtown Portland, Oregon, and with all due respect, I was surprised that it was so small. Somehow I assumed it would be mahogany walls and big desks and it's just you and another gentleman right now.

Bjorn Freeman-Benson: Yeah, there are actually four of us here in Portland and then there's another I think eight or nine in Ottawa. We live in the locations we live in because we like living there. I lived in Ottawa for a while and decided it was a bit cold, so I



prefer Portland. There are people who like living there for family reasons or whatever, but the foundation itself don't do the work. It's not like Microsoft building Visual Studio where Microsoft builds all of the code and there are -- gosh, I don't even know how many people there are working on the Visual Studio product, but I'm sure it's in the least 100, if not 1000. The work of writing the code for Eclipse is done by people out there who typically work for our member companies. There are 800 to 900 of those committers out there who work on Eclipse either part time or full time.

Scott Hanselman: Wow.

Bjorn Freeman-Benson: So, those are the people who are the committers on the project who have the right access. There are more people than that who contribute patches and code and so on, but I don't have a count for those. The 12 or so employees in the foundation do things like system administration, keeping the servers up, making sure that elections are run fairly and things like that. Carl who is sitting over there to my right is one of our system administrators and he's busy fixing a problem with I think the website at this very moment, but he and I don't actually write the code. Those are done by the 800 or 900 developers who are out there.

Scott Hanselman: Of those 800 or 900, are there leads? I mean how is that structured? Is there a hierarchy?

Bjorn Freeman-Benson: There is a hierarchy, but it's a meritocratic hierarchy. So, we've currently divided up into I think 80 different subprojects around the entire Eclipse philosophy. I could go over the web browser and look it up. We have a page that lists them all. For instance, there is a subproject that deals with Java tooling and there's a subproject that deals with C tooling and a subproject that deals with UML diagramming and so on and each one of those is run by between 2 and 20 people and those 2 and 20 people self-select a leader among them that they want to follow. So, each one of those projects has a project lead or sometimes co-leads who sort of define where the project is going to go and what things are going to be worked on, how to triage the bugs and that sort of thing, but it's all done by, you know, if you and I are sitting together working on some project and we decide that you're better to lead the project, we go, "Okay, now you're the leader and off you go," and then, you know, maybe later you get tired of doing that and this has happened at Eclipse. You've got

other things to do in your life and so then I become the leader for a while because everybody says, "Well, you're the guy who knows the architecture. You go off and do it."

Scott Hanselman: And for the most part, these are people who are using Eclipse and they're fixing Eclipse and they're using it in their everyday life. Their job may not necessarily be full time to be a committer on Eclipse.

Bjorn Freeman-Benson: Yeah, everybody who is working on the Eclipse environment is also using the Eclipse environment. I think it would be really hard to be a developer on Eclipse if you weren't using it in everyday life. I suppose it's possible.

Scott Hanselman: Are they full time? I assume a member company becomes a member company because they have some deep investment in Eclipse. They're using it. They're not just doing it to be philanthropic.

Bjorn Freeman-Benson: Well, it could be. I haven't actually spoken with them and asked each one of them "why are you doing this?" although I have to believe that most of them are doing it because they have some corporate market-based reason for doing that. It wouldn't make a lot of sense for that many companies to just go out and donate time for no particular reason, so I'm positive, my belief without having done any interviews that they're all doing that for some corporate reason. They're trying to make money. One of the interesting things about Eclipse as an open source foundation versus other open source foundations is that we're an explicitly commercial open source group. Our goal is for all of our member companies to be able to make money with the open source that's out there. Our goal isn't necessarily to give away free software, although there's sort of a -- I was going to say a vicious cycle, but it's not really vicious, a reinforcing cycle of, you know, if we have enough users who are using the free stuff that provides a market, which enables the companies to sell additional products on top of that market or services or something, but we're definitely aiming at -- our goal is for the companies to make money on top of the things that are out there.

Scott Hanselman: Forgive me of my ignorance here because I'm trying to get my head around this. Let's say that I want to start a company around this new LOL code that had been released. This is the funny cats that speak funny languages and I've



developed this new language and we have no IDE though. All the LOL code is being written in Notepad. I could take Eclipse and extend it, add plug-ins, add understanding of this environment.

Bjorn Freeman-Benson: Right, so you would want to add the plug-ins that would make an IDE for LOL code. In fact, we have a new project at Eclipse that helps you do that. You basically define the syntax and the semantics of language and it generates an IDE for you around whatever language you've typed in. Obviously, the IDE is not as polished as, say, you might find in Visual Studio where you've had hundreds of people working on all the little details, but it whips out the basic structure for you in fairly quick order so then you could make your LOL code IDE on top of Eclipse.

Scott Hanselman: And does it say Eclipse or does it say Scott's LOL code IDE? I mean how much I can brand this really and then I can go and sell it?

Bjorn Freeman-Benson: You can brand it as much as you want and some people do. We found that generally in the embedded space, companies tend to build a branded IDE on top of Eclipse that you know it's Eclipse if you're good at it because you sort of look at it and go, "That looks like Eclipse windows," but it doesn't say Eclipse anywhere. In the server side space, people tend to build on top of Eclipse and then add extensions to that, so it says Eclipse, and then with their extensions on top of that.

Scott Hanselman: Probably because I think in the server space, the name Eclipse holds some weight and some sense of quality and we're getting something that we can count on. So, the developer who is consuming that end product feels comfort in that this is built on Eclipse.

Bjorn Freeman-Benson: Yeah, that could be. The other thing we found is that companies, when they first moved to using Eclipse as an open source platform feel that they need to put a brand around the whole thing so they work very hard to make it their own thing, it just happens to be built on top of Eclipse, and then relatively quickly, their customers say, "Oh, I see it's built on top of Eclipse. I'd like to use this other tool, which is also built on top of Eclipse in combination with your tool." The first company will say, "Well, we didn't anticipate that, so there's not really a way to do that," and then their customers drive them into being more open and allowing the customers to combine different tools from different

vendors and that turns out to be a great attraction for the customers and then they try and buy more tools that are based on Eclipse because they have this ability to combine them.

Scott Hanselman: So, is there a distinct difference between the auto generation of the branded IDE on the one side and the extending Eclipse proper through extensions?

Bjorn Freeman-Benson: What it ends up being is do you have an installer and it creates an EXE that you double click to start it up or do you take an existing Eclipse image and install plug-ins into that? That's sort of the difference.

Scott Hanselman: I see. I see. What are the licensing requirements? If I was going to go and make my LOL code Eclipse IDE, do I have to pay you? Who do I pay?

Bjorn Freeman-Benson: All of the Eclipse open source code is licensed under the Eclipse Public License, which is one of these licenses which requires that if you modify the code, you distribute the modified code, but it's not viral in the GPL sense where it doesn't attach itself to all the rest of your code. This means that you can write commercially licensed plug-ins and ship them with Eclipse and that your commercially licensed plug-ins don't become polluted by the EPL.

Scott Hanselman: I see, but if I had to modify Eclipse to do it...

Bjorn Freeman-Benson: Right. If you modify the base Eclipse code, the EPL says that you then have to release that base Eclipse modifications out to the world, to help the rest of the world.

Scott Hanselman: And I wouldn't have to ship the source code to my extensions?

Bjorn Freeman-Benson: Absolutely not.

Scott Hanselman: One of the things that I thought would be of interest to some of the Microsoft developers that are listening is this really amazing IDE based on Eclipse called Aptana as a JavaScript IDE and I didn't know it was Eclipse until I noticed some of the windows and the buttons and I said, "It really looks like Eclipse." I went into the preferences and noticed that it was built as an add-on. Now, they've built both a Community Edition and a Pro



Edition. Is that a real common way for a commercial company to still have a foot in open source by offering a free version and then by offering a commercially licensed version all based on Eclipse?

Bjorn Freeman-Benson: Well, I think it is and I actually think that when you go that route, you go back to when we were building tools 10 to 15 years ago. Even Visual Studio had a Personal Edition, a Professional Edition and an Enterprise Edition I think what they were called and the Personal one was low cost and the Professional one was more cost and so on. Now, with this open source model that Aptana is using and various other people are using, the Personal Edition has become free and then you have these more expensive editions. I mean IBM still sells a version of Eclipse called Rational Application Developer. It's Eclipse with a whole bunch more tools built on top of it so you can get Eclipse, which IBM has distributed now through the foundation, it's no longer an IBM thing, or you can buy Rational Application Developer, which is this full-featured suite of tools for doing enterprise-sized development.

Scott Hanselman: And all these are written in Java, all these plug-ins? That's the primary plug-in architecture for Eclipse is to write JAR files and write Java extensions?

Bjorn Freeman-Benson: Yes. Currently, you have to write all your extensions in Java. I wish it were different. I wish you could write your extensions in any programming language, but we have yet to actually see that technology take over and there are certain technical reasons why that turns out to be difficult, which if you have plenty of time in your podcast we can go into or if you have another podcast we could go into, but I think it would be great for instance if I could be able to write extensions in, say, JavaScript or PHP, which I happen to be doing a lot of or if I'm a C# programmer, if I were to write my extensions in C# and have them work on Eclipse because there are people out there who want to use Eclipse who don't have necessarily a lot of Java experience. The Aptana example is great. Those people are really sharp people. They have a great community built around that product, but a lot of those people are either Ruby programmers, because Aptana also does Ruby, or JavaScript programmers neither of which translates directly to being a Java programmer. So, it would be great if those people could write extensions to Eclipse in their own language, but we don't have that technology yet.

Scott Hanselman: Yeah. It seems like it would be good if there was some kind of -- and I'm thinking of the Firefox kind of XUL method of extension where there's a manifest of some kind, the XML manifest, and then the system will handle the instantiation of whatever those objects and the bridging such that one could build an abstraction layer that was generic across all IDEs so that one could build a plug-in that would work in both Visual Studio and in Eclipse and in whatever, but perhaps that's a view of a Utopian future that will never...

Bjorn Freeman-Benson: Well, I have that same Utopian view, so I'm glad to hear you express that because I think there's a lot of things that IDEs do that are common across all the IDEs that it would be handy if we didn't have to reinvent them over and over again. One of the reason Eclipse exists in the first place is because a lot of the vendors in the Java IDE space looked around and said, "Hey, we're building the same tools for connecting to, say, CVS, and you're building the same tools and this other guy's building the same tools. Why don't we get together and just build them once and then we can spend our energies building the product quality things on top of it rather than reinventing the same base stuff over and over again?" I think that as IDEs get more sophisticated and Visual Studio gets more sophisticated and Eclipse tries to keep up or wherever you want to make that dividing line, there are a larger and larger base of those things that we could do in common between not just Visual Studio and Eclipse, but Eclipse and some of the other -- there's IntelliJ, which is another popular Java environment, which has a lot of great stuff in it, but they must also do a lot of the same basic things and have plug-ins.

Scott Hanselman: It's really interesting to watch the community kind of self-correct because I feel like on one really far side of the extreme view are the people who believe that everything should be open sourced and that free software completely and then of course on the other side is that everything should be commercial, but somewhere in the middle, the community decides where the current water level is and where things blow that water level should be free. There's just kind of an expectation of free that is kind of in the ether. You can't quite get your finger on it, but the community ultimately decides that, you know, I think syntax highlighting, IntelliSense editors, I think we can all agree are something that we shouldn't have to pay for much for and there should be value added on top of that before I will pay my 100 bucks or whatever.



Bjorn Freeman-Benson: Right. When you bring up the concept of free, there's free as in it's no cost to me because it's such a common idea that everybody has implemented it now. When I went to university, syntax highlighting was a big deal. It's the sort of thing we studied and now, syntax highlighting, 12-year-olds do that in their spare time. Then there's also the concept of free as in free choice and one of the things that Eclipse does, which I think is really remarkable is that it allows you to choose whether you want to pay for something or not pay for something. Let me give you the example of support. There are companies in the Eclipse ecosystem that you can pay them to support Eclipse for you or if you choose not to buy support, you could fix the bugs yourself because all of the source code is available or if you choose not to do that, you could submit bugs through Bugzilla to Eclipse and hope the people on the projects fix them. So, you can choose time waiting for people to fix them, you can choose doing it yourself, or you can choose money where you pay someone to do it. So, you have the opportunity to choose which way that's going to be and I think that's one of the real powers of open source is that it gives you that choice to how to spend your time and money. It doesn't force you into a particular vendor's model of *you have to pay for support or you have to wait two years for the next version* or whatever it is.

Scott Hanselman: I think that that's about the best definition of the essence of what open source means that I've heard in a long time. I mean that really resonates to me. I think that the perspective that it should be free and that the committers should jump on every support thing while they were working on the next version is an unreasonable expectation. It appears that some proponents of open source would have that kind of an opinion.

Bjorn Freeman-Benson: Right and we at Eclipse definitely don't. We acknowledge that each one of our member company supplies people to work on the various projects. Typically, those people who work on the projects have a belief in open source or they wouldn't be working on it, but at the same time they work for some corporation that has particular goals in mind about where the next generation of modeling tools should go, for instance. So, if your needs corresponds to their needs and their belief in the system, then you're in great shape. If they don't, you have the opportunity to choose some other path, well, staying with the Eclipse foundation. You can either

make a new plug-in. You can modify the plug-in. You could join the project yourself and help guide it.

Scott Hanselman: As a person who has worked for Microsoft for just about two months now, it's interesting to watch some of the discussions internally because some of the younger people (and younger is in quotes) people who haven't been at Microsoft for 15 to 20 years, in different meetings, at every opportunity someone says, "Well, how can we get that source out?" which is really unexpected. I did not expect to be in these meetings where someone would say, "We're gonna write XYZ and we're gonna figure out some way to get the source out," and it turns out that the things that are preventing us from doing that are all of the infrastructure and the legal and the this and the that and the 20 years of Microsoft-ness working against it. They just released the .NET Framework as a reference license and apparently it took like two years of legal wrangling to okay that. I wanted to get just a little widget, just a little scroll bar benefit, a little add-in that a guy at Microsoft is working on released and I've been there on and off working on this for two months to get this silly little DLL, this silly little feature out and it's been release meetings and legal review and writing EULAs and back and forth, but the fact that people at Microsoft are willing to work for that, I think will move Microsoft hopefully more towards a company that looks like some of these member companies that you're describing.

Bjorn Freeman-Benson: Yeah and I think a key thing there, I think that developers who just walk into this without thinking about the deeper issues don't realize is that we still need to make money somehow in the end. You need to be paid, I need to be paid. The price of gas is going up, I need to be paid more. So, there needs to be some model where even if we're making something open, there's still a way for revenue to flow in one way or another and different companies have chosen different models around that, around open source, but just giving everything away for no cost is not going to work. There are many good reasons to make your software open source and obviously, since I work for an open source foundation, I believe in them, but I don't believe in giving everything away. I was just giving a talk last week to another company and I was saying, "Well, one thing you don't want to give away is your core competency, the driving factor that makes all the money for your company. You don't want to give that away." I worked for a company for a while that contracted with a large shipping container firm. Their core



competency is scheduling shipping containers to minimize inventory in warehouses. That's a really interesting problem, but basically the inventory of their customers floats around the ocean in these shipping containers and can arrive at destinations at the right time. That algorithm is where they make all their money. The rest of it is not actually interesting, but there's no way that they would ship that algorithm out as open source because then their competitors could all have that exact same algorithm.

Scott Hanselman: Exactly, exactly. That's the algorithm that the chief architect has on a plaque with a patent underneath it hanging above his desk.

Bjorn Freeman-Benson: Right and he's handcuffed to it.

Scott Hanselman: But maybe 25 years from now when 12-year-olds start scheduling shipping containers, then there's a new problem that we want to crack and that does get open sourced.

Bjorn Freeman-Benson: Exactly.

Scott Hanselman: Yeah. I want to take a little bit of a 90-degree left turn because whenever I get the opportunity to talk to someone who's been in the industry for decades (*plural*) like yourself who has not only a lot of experience, but also an advanced degree in these kinds of things, I'm interested in your perspective of where you think language is going because as a person who has a computer science background, but not an advanced degree, I'm always interested in some of the language innovations like around C# adding LINQ and Visual Basic adding XML, some of the arguments that are happening between the Java purists and the Java pragmatists as languages are being developed and you've gotten into OOPSLA many times. Do you think that we're breaking some rules? Are there absolute rules like kind of the Butcha rules that shouldn't be touched or is it okay that we're changing these languages in ways that aren't pure in the kind of '80s OO kind of sense of the word.

Bjorn Freeman-Benson: The interesting thing is I am both a pragmatist and a purist, an interesting combination. I think that languages should provide us with the pragmatic bits to allow us to express our algorithms as quickly and as succinctly as possible. I program in Java a lot because Eclipse is written in Java and I find it at times frustrating and how verbose I have to be to do what appear to be simple things.

The language shouldn't be doing that to me. It should make it as easy as possible to do things. I think LINQ is great because it makes it so that I don't have to write a lot of verbosity to get the simple concepts that I want in terms of database queries. I like the Visual Basic XML syntax also, meaning if you've ever tried to do XML processing in Java, it's just a nightmare. I do a lot of XML processing PHP these days because they're simple XML and you just use it as a data structure, exactly what I want to do. At the same time, I find that a lot of the new language designs are not paying attention to some of the really great work that was done even before I was around in university having to do with closures and full objects that way and so they're reinventing things sort of badly. That bothers me because I think some of those concepts like languages without closures in the 2000 is just a mistake. I mean why would you invent a new language that didn't have closures today? Everybody knows how to implement them and knows how useful they are. It just boggles my mind. I'm speechless.

Scott Hanselman: Let's not assume that every listener knows what a closure is. Maybe you could just very briefly...

Bjorn Freeman-Benson: Right. So, what's the quick definition of a closure? A closure is a function without a name that I can pass around as a first class object. So, I can make a function and I can pass it to you and then you can call that function. I think the modern languages that are doing things like adding SQL and XML are making it easier and easier for us to write -- I was looking at a demonstration of the Adobe Flex stuff last spring and they've done some amazing things with how to call web APIs without having to do a lot of mechanism around it, which is built into the languages how you do that. That's great, we should have more of that, but while we do that we should also use some of these fundamental concepts that were invented back before they were really able to be implemented well. I think that language design typically says, "Well, I have this existing language and then I'm going to incrementally make it better." People who do that don't go back and go, "Well, I'm gonna make it better, but why don't I use all the power I've got now?" If we look at computers today, disk space is free. I mean we're recording this podcast onto a disk, I assume.

Scott Hanselman: Yeah.

Bjorn Freeman-Benson: Or maybe even in memory because memory is free, right?



Scott Hanselman: It's actually in a 2 gig flash card.

Bjorn Freeman-Benson: A 2 gig flash card?

Scott Hanselman: That cost me \$14.

Bjorn Freeman-Benson: Right. You know, when I first started writing programs, memory was fabulously expensive and so you had to be very careful how you wrote your algorithms and so things like closures and languages didn't make a lot of sense because we didn't have the raw CPU horsepower or the memory to spend on those data structures and we have that now. So, rather than having you or I do essentially in our heads what the computer could do, we should have the computer be doing that work for us and allow us to write the simplest, clearest statement of our algorithms that we're looking for.

Scott Hanselman: I've had some really interesting discussions with some people internally at Microsoft and we did a podcast on this concept of Parallel LINQ, which is going to be basically LINQ with an extra keyword that will recognize how many cores you have on your machine and then scale appropriately. So, if you had a four-loop, if you have *for each* over a million things and you got four cores, it will automatically handle all the threading, the parallel execution, and the coming back together of all that kind of stuff. It seems like all of us, Microsoft included, the languages are playing catch up a little bit because no one ever expected -- I mean three years ago, I would never have said that I would be running a quad-core machine and now they're saying 16 core within a year?

Bjorn Freeman-Benson: Right and it used to be you'd have to write Fortran and then you'd have to manually unroll all the loops based on the number of levels of parallels in your head. So, for instance, if you were programming a Cray, you unroll the loop in one way, and if you were programming on a different version of a Cray, it's an X-MP instead of a Y-MP, you unroll it a different way. Now, we can basically almost do that in hardware. Now, you're talking about doing that in the interpreter or the compiler, but some of these modern Pentium chips almost do that unrolling in the hardware itself.

Scott Hanselman: Yeah, yeah.

Bjorn Freeman-Benson: And so that's great because it removes that level of thinking that I have to do as a programmer. Now, I still have to, as a programmer, write my four loops in a way that doesn't create cross iteration dependencies because otherwise they won't unroll properly. The same thing when I'm writing a really high performance piece of code, I need to think about how the cache lines in the processor work because if I access my array in the wrong order, I can actually cause it to do a cache miss on every access, whereas, if I rewrite it slightly differently, I can cause it to do a cache hit on every access. That can affect the speed of my program dramatically.

Scott Hanselman: But that's exactly stuff we shouldn't have to think about. I should be able to express my intent in a dry and a way that it's not always I kind of always have to repeat myself, like I just did, and in a simple mechanism that will be optimized for whatever environment I've got. If I have a low memory environment, it should automatically optimize. I feel like just the fact that someone could be writing in a language and then think about what's going on in the processor cache is just kind of an uncomfortable...

Bjorn Freeman-Benson: Yeah.

Scott Hanselman: It shouldn't have a foot in each world.

Bjorn Freeman-Benson: I can imagine that that's our goal to eventually have that work that way. It's been my experience over a couple of decades of doing this that while we continue to get better abstractions and we can require less thinking of those details, in the end, to get the maximum efficiency, you always have to think about those details. The exact details we have to think about change. But even today when we're thinking about having, say, wide area storage networks where I'm using Google or Amazon S3 for my storage instead of my local disc, yeah, that's great because it removes a certain level of needing to do backups. For instance, Amazon, some guy runs all the backups for me I guess.

Scott Hanselman: Yeah, and they're up in the cloud.

Bjorn Freeman-Benson: And they're up in the cloud, but on the other hand, it takes more time to get to that for every access than it does to get to my local disk for every access.



Scott Hanselman: Because if it didn't, you'd use it as primary storage.

Bjorn Freeman-Benson: If it didn't, I'd use it as primary storage. Plus, the sheer speed of electrons is going to prevent me from having, you know, on your laptop here, the distance between the memory and the CPU is only a couple of inches. Even if there was a direct wire between here and Seattle, the electrons aren't going to travel that fast, right? I mean after a while, you approach physical limits. I'm just saying that I think you're always going to have to consider some constraint when you're writing code. It's just we'd like to think of fewer and fewer of them.

Scott Hanselman: Yeah, although I didn't expect myself to be writing C# and thinking about, "Darn, that speed of light limitation." If I could just get around C, then maybe I could get some work done.

Bjorn Freeman-Benson: Yeah, although I think that for majority of your programs, you don't have to think about that, right? I mean we're talking about the smallest, lowest power of things you can work on. I worked for a while for a chip startup company that was building chips for cell phones. There, it was really important to optimize the programs to use the lowest possible power because the battery only has 800 mA hours in it. So, it wasn't so much speed as we were programmed for, it was reduced power consumption and so we would know how much power every instruction would take and our compiler would tell us every function how much power that function would take because that was the constraint when running inside a cell phone.

Scott Hanselman: I heard that the Linux guys right now are deeply involved in a kind of a system wide power kind of an audit and they have an application now that will basically look at a program and give you an idea of how friendly that program is. For example, I guess they discovered that the volume slider was polling in like a really, really tight loop. So, when you were changing the volume, your power consumption went way, way up. They're doing some kind of a system wide thing to look at all the different applications so that an application could be labeled green or not green and one could say that when doing a system wide audit that we've lowered the power consumption by 20% through good programming practices. So, then power has become yet another thing to optimize for.

Bjorn Freeman-Benson: That's really interesting.

Scott Hanselman: Yeah, I find it to be pretty crazy. My last question for you back on Eclipse before we go is that have you heard any demand for .NET on Eclipse or C# on Eclipse? Are there any projects that are involved or like any member companies that are interested in providing a .NET environment such that I could use Eclipse for my .NET work?

Bjorn Freeman-Benson: So far, there is no member company I know of that's providing a C# environment, but the people who work on the C, C++ tooling have made a prototype of a C# tooling and they actually have it and it's working and they've actually written some programs on top of it. So far, there's not a lot of demand for C# on Eclipse because there are great C# tools out of the Microsoft environment.

Scott Hanselman: Sure.

Bjorn Freeman-Benson: But it's definitely been proven to work. You can write your programs and run them and debug them, the whole thing.

Scott Hanselman: Well, I have definitely found the Aptana application to be a good compliment to Visual Studio even though the new version of Visual Studio has JavaScript debugging. One of the things I thought was amazing about Aptana is that they'll be able to tell you in JavaScript, HTML or CSS which browsers will be supported for that particular thing. So, if you're using some feature, it has a beautiful IntelliSense that has a column with a grayed out or not grayed out icon of the browser that you're about to use and you say, "Well, this won't work in Opera, but it will work in this and it will work in that."

Bjorn Freeman-Benson: Yeah.

Scott Hanselman: It's a gorgeous IDE.

Bjorn Freeman-Benson: The Aptana guys are really sharp. I think that's an example of where they've chosen to make the right level of abstraction to solve your problems so that you don't have to think about them. If you look at our more traditional JavaScript editor, it just edits JavaScript. Maybe it does syntax highlighting and IntelliSense and so on, but it just sort of a pure JavaScript editor. It doesn't go further and say, "Well, if you're editing JavaScript that uses the Dojo framework, what else do you want to know? What are the little idioms that you would use when



doing that?" The Aptana guys have gone a little bit farther and done that bit of it.

Scott Hanselman: Well, I think that's a really great example of what can be done on Eclipse and I really appreciate you taking the time to talk to me today. This has been another episode of Hanselminutes and we will see you again next week.