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 #200

February 12, 2010

Jon Skeet - World's Greatest Living Programmer, or just a nice English guy?

I tease! It's show 200, and we've got Jon Skeet. Jon writes Java at Google, but he's also got a new book out called C# in Depth, Second Edition. Jon is also well-known for his answers on StackOverflow, gaining him the title "The Chuck Norris of Programmers." Listeners can get 40% off with code "HanselC40" at http://manning.com/skeet2 until March 15th.

(Transcription services provided by PWOP Productions)

Our Sponsors

Lawrence Ryan: From hanselminutes.com, it's Hanselminutes, a weekly discussion with web developer and technologist, Scott Hanselman. This is Lawrence Ryan, announcing show #200, recorded live Friday, February 12, 2010. 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. In this episode, Scott talks with developer Jon Skeet.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes. In fact, this is episode 200 of Hanselminutes so it maybe a little bit more than just another episode and I thought about who should I have on this episode. I said maybe I should get Bill Gates or Ray Ozzie, and in fact I realize the person I wanted most to have on my 200th show representing nearly four years of weekly podcasts was the Chuck Norris of programming, Jon Skeet himself, and he is on the phone with us now from England. How are you, sir?

Jon Skeet: I'm fine. Thank you very much. How are you?

Scott Hanselman: I'm very well. I'm thrilled, I'm thrilled to meet you over the phone here. People believe that you are, in fact, the Chuck Norris of programming and this has been kind of a controversial thing over on Stack Overflow. About a year and a half ago someone opened a question about facts, about Jon Skeet and Chuck Norris. Can you give us a little background about what exactly happened and why was it a little mini controversy?

Jon Skeet: Basically it was a bit of fun. So I've been on Stack Overflow for a few months and the reason to be quickly due to having just enough knowledge to be dangerous and having quick typing fingers; and someone opened this question saying Jon Skeet sucks which is clearly not programming related and various people disliked the idea of celebrating the cult of Jon, etc, which I can completely sympathize with but then Jeff Atwood started with Stack Overflow said no, Stack Overflow is about to use this so this question is fine and it's been opened and then closed however many times, lots of times but at the same time it got hundreds of answers. It's extremely silly to be honest. It's not about me at all. It's about let's see what silly things we can come up with it that a super hero programmer would have.

Scott Hanselman: Certainly, certainly. Yeah, it's not about Jon Skeet who we're talking to. It's about the mythical, mythical, mythical piece of Jon Skeet that we have never met. If you don't mind I'm going to read a few of these because they're truly amazing. Jon Skeet is immutable. If something is going to change it's going to have to be the rest of the universe. Jon Skeet code doesn't follow a coding convention. It is the coding convention. And here's the best one I think that is the most Stack Overflow-ish. Users don't mark Jon Skeet's answers as accepted. The universe accepts them out of the sense of truth and justice. I think that's pretty fantastic. There are 279 answers on that question and it continues to go on. It has a life of itself. But you are very well respected as the technologist and you work for Google.

Jon Skeet: I do indeed.

Scott Hanselman: You've written C# in depth which is out now.

Jon Skeet: Yeah. So I work at Google in the London office on the mobile team which means I write Java most of the time and I should point out that I don't speak for Google in any way, shape, or form so don't regard anything that comes out from my mouth as the voice of Google. But yeah, so I write in Java, I write about C# and I mostly program C# for fun and I find it a lovely language to work in. C# in Depth has been out for about a year and a half now. Ironically it actually was published about the same time that I came over to the States for a fortnight for a regular orientation as they call it and so just as I was officially leaving a career in C# the book came out and by the time I got back there was a lovely carton with my 25 true copies. The C# in Depth, Second Edition is what I'm working on at the moment and that will come out more probably June, maybe July.

Scott Hanselman: Did you have a problem moving from C# to Java, or was it more about working for Google regardless of the language that you would be using there?

Jon Skeet: Well, I certainly decided to work for Google whatever the language, and it's a fabulous place to work. I haven't had very much problem doing Java because I've always been flipping between the two. I started off in Java. Well, my career started in '94, '95 sort of in university vacation working Digital and I started off in Java. That was back in Java 1.0 when it's completely interpreted so it was a really slow piece back then. Then my first job started off in Java and I did experiment with C# just near the end. Then I went to a job that I thought would be about C# and it ended up being in Java. After a while then I went back to another job with C#, then I'm at Google that's doing Java although I do different projects while I'm at Google and in C# when I can.

Scott Hanselman: Are these two languages really that close that someone can move back and forth between them with apparently no major problem, or is this just that you're a good programmer?

Jon Skeet: Oh, certainly not the latter. I wonder whether actually the problem might be too close to be comfortable rather than too far back, and

people use different languages all the time. There are some people who will write a bit of JavaScript, then a bit Haskell or crazily our cup of things. If you're doing some enterprise integration, you might have to maintain some credible system and then write in C# or whatever. The frustrating thing sometimes is that Java and C# are very similar in many ways, but Java has just been evolving that much slower than C# so I reckon that when C# originally came out it was a little bit better than Java and Java was I think 1.4 at that stage. It hadn't yet got generics. The Java got its version of generics before C# did and you would have thought they would be in sort of favoring each other all the time but Java really hasn't gone very far in sense of the language since then. We're sort of pinning our hopes on Java 7.0 to really get a lot of the features that C# already have and maybe a few other ones as well, but yeah, C# has just been taking off incredibly fast.

Scott Hanselman: It does seem to be a kind of thumb war between the two languages. How many more features can we add to a language? I think that LINQ is pretty extraordinary the way that it was added to C#. Is there something coming down the line in Java that's going to be LINQy, or in the sense adding a huge DSL to this language?

Jon Skeet: I don't know anything quite on the scale of LINQ and I suspect that if those come it would be a little bit later. The main talking point about Java 7.0 from my point of view is whether Java will get closures and it was announced a few months ago that it now looks like it will. For a while it was no one can work out what closure should look like in Java, something equivalent to Lambda Expressions in C# and there's no trace of which. It looks pretty promising and we'll see whether it happens but it's meant to happen and that's for me is the huge point because there are a lot of things that are like LINQ. If you imagine the LINQ Query expression then take this in C# but we still have extensions method and we still have Lambda Expressions and we still have all the type in front improvements that came in C# 3.0, you could still be pretty damn productive with that. A Query expression is the icing on the cake in a way. So I think now for sure what the Java 7.0 looks like. It's going to get extension methods or something similar and everything can be slightly different that achieve the same kind of things. But just having closures will make a lot of things much, much better.

Scott Hanselman: What do you use from an OS and IDE perspective while you're at work at Google versus what do you code C# in at home?

Jon Skeet: So if I'm doing C# at home I'll use Visual Studio either 2008 or 2010. I've just moved to the release candidate of 2010 and it's looking lovely. At work I use Eclipse to Java and often in fact for most languages I'll use just a plain Text Editor particularly for questions on Stack Overflow items like I tend to like to write small consul app, 15 lines long. I'd build a TestBook.java and the TestBook.cs in the directory that a consul brings me up in or I edit that and run the command-line compiler in a way and that's considerably quicker than to bring up Visual Studio and find the sandbook project trying to get rid of what you have before, finding it very quick. Sort of EmEx like Text Editor is considerably quicker.

Scott Hanselman: Yeah. For that I've been using a program called Snippet Compiler. It's basically like Notepad but it compiles. So it does exactly what you just described.

Jon Skeet: Right.

Scott Hanselman: You know, you may do a quick edit and you go see it at the command line. But I understand that you definitely don't want to litter the world with the, you know, file a new consul application in '96 if you're just trying to do a repro.

Jon Skeet: Yeah.

Scott Hanselman: So what is the OS you're using though at Google?

Jon Skeet: We use Linux as well. I use Linux as my main desktop and most people do like engineers do. I do also have a Windows laptop.

Scott Hanselman: Is it okay to ask which distribution?

Jon Skeet: I don't know whether it's okay to ask or not so I'm going to not say anything.

Scott Hanselman: All right. But it is Eclipse. So you've been using Visual Studio 2010 RC. This is the one that we just release to the public a couple of days ago.

Jon Skeet: Yeah.

Scott Hanselman: And we release it to MSDN subscribers a little bit before that. I just installed it on three different machines, fast, medium, and slow and I'm going to try doing some tests. What kind of machine are you running on it and how is it working out for you?

Jon Skeet: So I have a couple of years old Dell Inspiron 1720 which has got I think two gigahertz and the dual core for my main laptop frame, and also I've got a Samsung NT10, because I've got about half an hour long community train, or an hour and a half I'm using. I can try to get some work done on the train so I've been working the book and writing some code for the book on the train. So that really sort of put Visual Studio throughout places in certain speed. The feature too was not a pleasant experience for the

network. But with the release candidate, it's actually pretty reasonable.

Scott Hanselman: Yeah. The general sense is that the RC has a huge number of performance and improvements. I'm going to be really interested to dig into it and find out if that's something that's specific to the IDE that they made faster or if it's underlying things like making WPF faster or making other libraries faster in order to enable the IDE to be faster. Because I think if someone at Microsoft has learned a great deal about speed, the rest of us should learn about it as well to take advantage of that.

Jon Skeet: Yes. It's difficult to know exactly which will be due to Visual Studio itself. I know Rick and Naomi has looked about things that they have found while tuning the heck out of Visual Studio and some of those would be quite specific where it's interoperating with COM in plugins that most program has probably done to very, very much common integration. I sort of personally have not because I don't particularly enjoy using COM myself. The other things I will be interested, there are a couple of strange performance at work that I found with nullable clients for instance that will be really kind of deep in the CLR and I have no idea what really are that changes between Beta 2.0 and the release candidate. But yeah, I don't have unfortunately a good performance benchmark other than eventually my protocol buffers and the trend that it tend to project that I've mentioned earlier on which is sort of a serialization and de-serialization framework. I do rotation on even performance benchmarks at that site and see whether that's any faster in .NET 4.0 than .NET 3.5 now.

Scott Hanselman: Hi, this is Scott Hanselman from another Parallel universe. I've got to tell you about some of our things our sponsors are doing. They make this free podcast possible. If you're developing a new line of business application, you've probably have tried the latest version of Silverlight. Now you can get even better results by combining the functionality in Silverlight 4.0 beta with some of the richness of the third party controls from our friends at Telerik. They're the first vendors to offer native support for the Silverlight 4.0 beta and their RadControls. They've got a new Silverlight 4.0 CTP suite of these controls. They let you tap into the framework's great potential. You've got native right mouse click and all the new features of Silverlight. There are 38 controls that give you all these features. You can start building those compelling line of business applications right away. I encourage you to check those products out at www.telerik.com/silverlight, and, you know, thank Telerik for supporting .NET Rocks!, supporting Hanselminutes on their Facebook fan page at www.facebook.com/telerik. Now back to the show.

Is this new -- this framework that you're working on, is that Noda Time, the open source date/time library that you're working on or is that something else?

Jon Skeet: That's something else. So Noda Time is a completely personal project. It's got nothing to do with Google as it were, whereas Protocol Buffers is a Google framework serialization that's why it effectively just pulls it from the Java. Again they're both Java ports really, but the idea of the Protocol Buffers is you can have a common portable serialization framework so that someone could send my framework and object serialized from Python or from Perl. I think there's a Perl implementation also in C++ and Java and I can incorporate with that, send then something back. So it's a lot less virtual than the sort of Java and C# binary serialization that is very, very specific to that platform. Obviously it becomes less flexible at that point as well so you have to specify your serialization messages made out to various sort of primitive case string, here's a binary blob, here's an integer, etc. You really have to design your client server in that but equally you certainly get portability designs because obviously Google will have lots of different versions and bits of code running and you can't upgrade all of your virtual servers to the same version just like that. So you have to have a lot of tolerance for well this was message version two; can we read it from version 3.0, yes, we can. If we send the message from version 3.0 back to version 2.0, can we read that? Yes, we can and it's specifically designed for that sort of thing, whereas, they're design is a completely different project that I really started after talking at the Stack Overflow Dev Days conference and I gave a very lighthearted talk called "Humanity: Epic Fail" where I describe some of the problems that really we brought upon ourselves in terms of computing. So you would have thought three of the things that should be easy dealing with text, dealing with numbers, and dealing with dates and time. Those are all sorts of run of the mill things that we should make fairly easy, but all of them are insanely difficult. One of the problems in .NET is while the BCLs have certainly been improving and claims an intro and they claim also that all are very, very welcome, they don't really spend everything that I think should be in a date and time library and there's a fabulous library by a chap called Stephen Colebourne in Java called Joda Time and that is widely regarded as the library that you use if you're going to use date and time stuff on Java. You don't bother with Java. You tell the date. You then talk to Java, you tell calendar that's too error-prone, too difficult to use, but you use their design. It seems to make sense to me to put that in .NET and tries to do a real good job of making it feel like it's a .NET library, not just support. At the same time, because it's an interesting open source project therefore it will be a good idea to try to take a record of the problems that they faced when porting it, the problems in the sense of what does .NET having to write build systems, unit test systems,

etc, how do you use one for an open source project. So I'm going to try to pull that together in a sort of guide to so if you think you want the stuff and they convert into those projects, here are the things to think about.

Scott Hanselman: That's a great idea. Is this something that we can watch? Is there a Get Hub or a subversion repository that we could kind of snoop-in on?

Jon Skeet: There's a mercurial repository at code.google.com/p/noda-time. We can put the link in the show notes. We've got loads of people interested so I treated that app as an important app for the design and last time I looked I think there were 80 or 90 people who were interested in it and interested enough to have chipped in on the mailing list and various others committed and have definitely contributed plenty of code. It's all from a one-man project. Unfortunately everyone is busy. I'm busy writing the book, etc. So I started it in September and my aim was to have something that could be useful and you wouldn't feel worried about looking at and using in a real project in one year's time from that September. I still got about seven months to make it production-ready. We've got some useful stuff already, but it's definitely early days and I'm hoping that as soon as I finish C# In-Depth Second Edition I can start to search some real-time and, they're design.

Scott Hanselman: Will it be as complete as Joda Time? I mean, Joda Time is not just complete. It's like encyclopedic. It supports like six different kinds of other calendars. It's not just the standard Gregorian calendars, but it goes out to Islamic calendars, Coptic calendars.

Jon Skeet: Yes.

Scott Hanselman: Plus it has a pretty interesting kind of a fluent API. Are you going to have similar functionality?

Jon Skeet: Certainly. So we'll definitely support various different calendars like we're restricting some of it because in some ways Joda Time is a bit too flexible. It allows you to create your calendar fields. So if you have some idea that rolls a third of a week, which third of the week are we in at the moment? You could add that as your field type in Joda Time and if you hook it into every single spot that you needed to then you could start asking what third of the week today is, etc. Now, we thought that was a little bit too flexible. Just in a sense it made the inner phase harder to understand, harder to implement, etc. So we've tried to simplify a bit. We will have a fluent interface in the same place as Joda Time does although actually there's a lot to be gained from the fact that we've got software overloading in .NET. So Joda Time tries to stick to immutable plates in most places and in fact where they do have immutable plates so far we've managed to mostly remove them from native time, and immutable plates are ideal for operators adding appearance to an instant whatever and so it's really trying to take the benefits of all the stuff that you can do in .NET that you'd like to be able to do in Java but it just doesn't quite supports it. And likewise in Java they're all reference type, they're all classes because that's all you can specify, whereas I think I find myself writing test instructions in C# but in fact we've already got three different construct which are just wrappers around a single one; but they're really, really useful to have as different wrappers and in fact we've got one slightly strange scenario and the offset of a date/time from UTC. So for time zone it's got a particular offset at a particular time and we've got strange situations that you can only add it to a UTC time and you can only subtract it from a local plane to get back to UTC time. So we've got a sort of asymmetric situation which actually helps you keep time safe. We can't accidentally use it the wrong way around and end up with your sec backwards which is strange but it has worked enormously well.

Scott Hanselman: Wow. Are you going to deal with time zones?

Jon Skeet: Oh yes. So we've got plans and support. One of the things that's missing as far as I'm aware from the .NET BCL is support for what's called the open date space in time zones. So time zone insight has all the Windows main forms for language which is entirely reasonable given .NET Windows nature, but also the rest of the world users also identifies so Europe/London for example has a time zone that's climbing at the moment and we support that real, it supports historical changes to the time zone. So time zones are changing all the time. One example I with a vendor when I was giving at Stack Overflow a thought was when one of my unit test has just broken and thinking that Greenland was in Argentina because the Argentine government has just announced that they were stopping daylight savings and they gave us 11 days notice that they were doing this which is kind a mess as far as the software in Argentina is concern but it does mean you need to have the idea of a time zone that changes overtime so we can accurately represent what happened in history and we can accurately represent what it looks like if it will happen in the future and as this is updated so there's a whole blog to do as date and time zone is changing. As things changed, then also date space gets updated and then we'll be able to ship either a new and complete version of Noda Time or just a sort of supplementary here's the time zone date space that you don't need to change your binary, just the time zone information.

Scott Hanselman: This really seems like it will be an exercise in unit testing. I mean, the date/time

libraries and date/time calculations are only as good as their test coverage.

Jon Skeet: Absolutely. It's an ideal thing to unit test because there's no extra on input. We don't have to worry about what does it look like on the screen. We don't have to worry about how is going to react when we use it to type something in because they're not typing anything in it. It's just the library. It's plain, old put some numbers in, get some numbers out, are they the right numbers. Now there are various internationalization issues so we'll have to make sure that the unit test either don't assume anything about the culture they're in or specify the culture for formatting for instance. One of the nice things that we've got that I can, so long as its Joda Time, is the idea of you build a formatter that knows what pattern it's using which means that user have to call it all the time. So every time you call date/time, try it to call exactly to whatever you're calling, if you specify the format pattern then the BCL is having to support that and say, all right, you're expecting some years to solve within, some months and some dates, etc. What a waste of time. You're going to use the same format again and again and again. So we've got the idea of a formatter that got that pattern and build into it and then it can just scan through the actual input text which will change all the time and hopefully that will be considered to be faster than the BCL in completion. We are checking performance against the BCL and I'm expecting that other things will be slower just because we're not spending all our professional lives tuning simulator. I'm sure the BCL teams are. It will be interesting to see whether or not that's specific teacher. We can actually have a form of BCL. I'm really hinting that we can.

Scott Hanselman: That sounds like if you're willing to put this much effort into improving .NET, that you definitely think that there's a future for .NET.

Jon Skeet: Oh yeah, yeah. It's something I think that occasionally we all talk about the global legacy but I think that's going to be a huge legacy in the future for all these platforms, but .NET is not going away. Java is not going away. Ruby and Python aren't going away. I really feel for people who graduate in say 30 years time and have all these systems which by then maybe sort of legacy in the same way that COBOL is legacy. Yeah, maybe we'll still be writing new C# code in 30 years time. Probably not, but there will be a heck of a lot of code to maintain. So I see a Ruby feature for .NET itself in the short term and in a very long term I'm sure it will still be a realm to be maintained for a very long time indeed.

Scott Hanselman: That's comforting. As somebody that kind of makes a living on community, that's pretty important to me. Do you spend most of your time on Stack Overflow? Is that what community is to you?

Jon Skeet: That's first and most of the plan as I think of community, yeah. So I do write my own blog and in some ways I guess you could call writing a book a community activity. It feels solitary at some of the time although for this second edition I've actually tried to get quite a lot of feedback and by asking questions in blog post saying, hey, how would people feel about teaching this chapter, or changing the focus of another chapter, and so on but trying to get as much feedback as I can early on in the process. Other than that, yeah, I spend an amount of time on Stack Overflow, yeah. I claim that I learned an awful lot just from trying to answer the people's question. I've done that for, well, for as long as I've been programming in Java or in C# certainly.

Scott Hanselman: You're still the number one person on Stack Overflow, right?

Jon Skeet: Yeah. I was talking about it. They stayed at breakfast the other day. It's actually slightly tricky because I'm now sufficiently far ahead. If someone joins Stack Overflow today and if stop posting today then because I've been posting for nearly a year and a half, it would take them a year and a half if they posted the same app that I have been doing in order to catch up. Sometimes I wonder whether there ought to be some sort of dictator reputation but equally that doesn't make sense in other ways. It really depends on what you gather your reputation is and Miguel de Icaza was talking about this just yesterday saying "You really doesn't have chances beating me." Well, beating me suggest a competition which I guess it is in some ways but I think it would do us all good to just regard reputation as that happy point.

Scott Hanselman: Well, yeah but when you have a main page that sorts by happy points it's pretty...

Jon Skeet: Oh yeah, yeah. I maybe think of it as a game and I'm competitive when I'm gaming. Particularly I play ballgames quite a lot at lunchtime and so yes, I certainly play to win when I'm playing a game but at the same time you can't take it too seriously. It would be absolutely pointless to try to gain the system to get more reputation playing just for the sake of getting more reputation points. At that point you've really lost the whole nature of why you're on Stack Overflow to start with. If I've answered a question and someone else answers it better and includes everything else besides all already written, then I will just delete my own. The main point in keeping it there is just to keep that reputation and the point is ultimately to help people as much as possible and reputation is a fun game to play along with really.

Scott Hanselman: I think for me the thing that's most significant about the kind of the "Stack Overflow experience" is the idea that reputation enables you to do other things, not that it's most interesting to be

number one, but that it's interesting to have some amount of effort into it because then you get the ability to moderate and the ability to self-maintain and that is not just a bit flipping. You're either a newbie or you've been there. There's a gradient of various things that you can unlock and those things may the experience better and then you would continue on and it's kind of a self-maintaining system.

Jon Skeet: Yes and even when you go and look at all the source special abilities there, still Apaches and particularly with the tag batches and when you've had 400 upgrades to a particular tag then you get the silver badge for that, and if you get a thousand in a particular tag you get a gold badge for that. Not that it's particularly amusing but I was the first person to get a silver badge for Visual Basic which, yeah, recent. I like it personally. I regard myself as particularly fluent in and yet I got a silver badge and I think I may now, I'm just checking, yeah, I now have a gold badge in VB. If you want me to write from VB, I would completely fail -- I would have to compose in C# code and then decompose and reflect through all of that together.

Scott Hanselman: Uh-hmm.

Jon Skeet: So it does amuse me, but that's just because most questions that people think of, VB questions, are actually just .NET questions with a VB accent.

Scott Hanselman: I like that one, with a VB accent. Let me ask you this kind of in closing. How much time are you spending on this really?

Jon Skeet: Oh, I couldn't measure it because I use it -- I don't spend it in idle playing doing it. How much of the time do you spend breathing? Not that I'm equating Stack Overflow to breathing today. I'm not quite that addicted, but you know, it's just something to do while you're waiting for other things. So if I'm waiting for a compile to happen, I'll just quickly check. If I'm just taking a mental breather, I will check and then often usually even there are many questions we kind of have to look at, if there is I'll take a quick look at it and maybe answer it for a few minutes. Sometimes I'll answer a single question for haft an hour or an hour. Occasionally I put in quite a lot of methods. So it's something that I spend a lot of time looking at but without wasting much of my time if you see what I mean and I don't think anything will particularly take its place and make me more productive or more useful to humanity in some other way, but equally it does take a fair amount of my time, yeah.

Scott Hanselman: Okay but it's not like you sit down and you think of this as a piece of work that needs to be done like, all right, got to pound through the Stack Overflow today. It's more of a it's your time filler.

Jon Skeet: Yes, yes, exactly. So where some people would sort of have a game of solid Stack, it's you've got three minutes for whatever reason, you have a game to pull it there or you check, check also whatever, I tend to check Stack Overflow and it's just one place that takes my time.

Scott Hanselman: Well, I for one am glad that you're spending time over there because it's a great community and I enjoy reading your very thorough and complete questions and answers.

Jon Skeet: Yeah. It really is completely different as community goes and to anything else I've seen. So I know it looks a bit like expert exchange, it text the exchange without evil, without having to look at experts to change myself much. I don't know if it even really is but there you go, but even compared with news groups the reaction time is so fast. It used to be on news groups. We would say, okay, if you don't hear back within the day then it's reasonable to key in and say have I missed something and did I not give enough information, etc. On Stack Overflow, if you haven't surf back within half an hour then I'd be surprised. Normally when I asked questions and if they see it, if I'm asking a question it tends to be very confrontational or something about C# 4.0 that not many people will come to a particular corner, but even so I get answers very, very quickly. It really is an amazing result. I can get really changing how people compare them.

Scott Hanselman: Absolutely.

Jon Skeet: And the fact that it's so white that it doesn't just write this on .NET, that they doesn't just take it on Java. Yeah, I've got some Ruby questions. I've seen a question and so, yeah, with a bit of research then I could whip it out. I would go ahead. It turned out a certain operated dash or help something saved and I've learned a little bit of Ruby. I'm not a Ruby programmer too. I would have never picked it out if I had seen that question. Obviously I couldn't have done it. It's like going out of my way. I'm sure there are a lot of great Ruby programmers, but the fact that Stack Overflow just throws them all together it means that we've all got just that bit more incentive to learn something beyond that comfort zone.

Scott Hanselman: Yup. I love it, I love it. Thank you so much, Jon Skeet, for talking to us today on our 200th show of Hanselminutes. I really appreciate it a lot.

Jon Skeet: My pleasure. It's great talking.

Scott Hanselman: Well, this is show 200, another episode of Hanselminutes and I'll see you again next week.