From:	bens
Sent:	Thursday, August 03, 1995 9:57 PM
To:	'johnlu@microsoft.com'; 'paulma@microsoft.com'
Subject:	FW: some thoughts for the record

we're thinking critically about the web over here in building 5! good comments from jeremys, and a good reply from chrisjo.

chris and I and thomasre met with ericr a few days ago and asked him when he would build a save VB(A). The bottom line he gave was: 1) he doesn't think saftey is very important; 2) he thinks safety is hard to do; 3) even if he wanted to do safety, he doesn't have any bandwidth.

I must observe that this is the lamest, most "big company" response I've heard in a long time. If we don't get the VB group focused yesterday on building a safe scripted execution environment, I agree with jeremys, you can just kiss VB away on the Internet. VB will quickly come to be known as "Virus Basic"...

just my $0.02
--bens

From:	chrisjo@exchange.microsoft.com[SMTP:chrisjo@exchange.microsoft.com]
Sent:	Saturday, July 29, 1995 3:40 PM
To:	bens@microsoft.com; chrisfra@microsoft.com; jcordell@microsoft.com; thomasre@microsoft.com
Subject:	RE: some thoughts for the record

I agree with your principles, but I think that you are combining issues. Our team is chartered with two things:
1) Build a standard set of runtime services that all applications can use, which are optimized for a narrow pipe and targeted at Web Browsing, Help, Workgroup Forms, and MultiMedia Titles (BlackBird and MediaView).
2) Build a competitive browser and integrate it as a part of Windows.

For (1), the core question is "What do we need to provide to ISV's so they can develop the best tools to run on the Web, and develop them on Windows first?" The proposal on the table is to use OLE/COM, OCX, and other components to build *services* from which people can build *interpreters (or apps)* for the different data formats that appear on the Web. OLE/COM and OCX are connection models that we will choose to use, and these can interpret a wide variety of data formats. This framework must be extensible, language independent, and provide enough value that people who write custom applications will use it. These services will be used not only by Web browsers, but also by other applications and tool vendors. We could just as easily use DLL's or Windows API's to do this -- these wouldn't be platform independent either. We are choosing COM/OLE because our company has a lot of technology that exists which has solved a number of these problems, and it is architected using COM/OLE. I am interested in hearing additional ways that we can get to this goal, because I think the proposed solution is extensible, lightweight, and satisfies client needs.

For (2), the proposal is that we have a group of people who take the services above and build a Web browser that competes with NetScape and evangalizes Web standards. This team must figure out how to build the best HTML interpreter and renderer using these services -- and if the services don't satisfy the teams needs, they need to push back to get the right things. Java is a threat to us, and we will need to embrace controls written in Java. But we should figure out how to do this is a way that allows all languages to exist, not just things scripted in Java. Just because we have runtime services based on OLE/COM *does not* mean that you can't host Java objects -- in fact, the Java folks have been inquiring about how to write these controls already. ThomasRe owns figuring out what we will do w/ Java, how it works, and how we can embrace it and then extend what it does to support any type of object written in any language.

There are hard questions and issues: VB and DocObj are two. My perspective on these is as follows:
1) DocObj. First, Office is evangalizing this to NetScape and whoever else comes around, and my assumption is that NetScape will support it, which means that most people will agree we have to as well. That said, even if NetScape didn't support it, having a browser that can plug in different viewers for different data types is a good thing -- it opens up the market for different people to extend your browser, and makes it hard for anyone to catch you. DocObj is a plug in architecture that our Office applications support today, and has been evangalized to all Office compatible developers -- it would be silly not to support it in some way, even if the plug in for HTML was lighter weight.
2) VB. There are two clients (Exchange and BlackBird) who need scripting support in their applications, and want to

choose VB. There are thousands of VB developers in the world today. True, VB is not safe, nor cross platform, but it is in our companies best interested to get it out there if only on the Windows platform. We should also allow any scripting to plug in, and then let the market decide. If Java starts kicking the pants off of VB, the VB team will wake up and make the changes they need -- they are aware of them already. All we are doing is providing them as a choice in our box, providing a Java alternative (even though it doesn't do the same thing exactly), which some set of people will use.

From: Jeremy Stone (Xenix)
Sent: Friday, July 28, 1995 2:19 AM
To: bens; chrisfra; jcordell; thomasre; Chris Jones (Exchange); victors
Subject: some thoughts for the record

"Every time history repeats itself, the price goes up."
- Samuel Butler

I know I'm pissing in the wind here, but I thought I'd take a minute
to express as coherently as I can some doubts I have about our new mission.

Basically, I think that trying to make any of the following: [
OLE/OCX, VBA, .DOC, .XLS, docObj] into an Internet standard or any
sort of prevalent data type on the Net is doomed to failure. I think
it's sticking our head in the sand. Not the work we discussed today
is not worth doing-- but we have to give it an appropriate priority.

The Internet is held together by standards. Let's list some of the
important standards that have withstood the test of time and become
THE standard in each department:
TCP/IP
FTP
SMTP
HTTP
HTML
....

Now here are some of the properties of these standards:
- They are *platform-independent*
- They are *publicly documented*, *open* (I use that term loosely),
and *any vendor can make an implementation of them*;
- They are tightly *focused*

Why are these properties important? Why do I think any standard (at
least at this point in time) is not going to succeed unless it has all
of these properties?

- Platform independence. Internet content authors (which today
primarily means web page authors) will be faced over the next year
with a potential audience on at least four major platforms: Win 95,
Mac, Unix in its various flavors, Win NT. We are presumably going to
blow Unix off no matter what we do and hope it withers and dies. But
I claim we absolutely must have a platform-independent solution for
Win 95, NT and the Mac otherwise other competing technologies (notably
Java) will be more attractive to author in. We won't have ISVs (web
authors) for our platform (VBA/OCX in HTML), we won't get critical
mass. Java already runs on Unix boxes, we expect to see it on Win 95
and the Mac in Netscape soon. We have a huge hurdle just to match
Java's platform independence, we're not even in the game until then.

- Vendor independence. Standards that can be implemented by only one
vendor not only stifle platform independence, but innovation. Think
of how much VB has changed in a year. Now think about how much HTML
and web browsers have changed in a year. (And is it a concidence that
as web browsers got cooler and content got more compelling [with

MSS 0037409
CONFIDENTIAL

features like background images], the web grew at an enormous rate?) The reason, of course, is competition. We don't like competition, but competition breeds innovation, and innovation breeds growth. And we like growth. Think about HTML-like technologies at Microsoft and how long they've been struggling to ship. Now think about the leaps and bounds HTML has made in a short time span, with Netscape and Iexplore enhancements. Here's an example where a lightweight public implementation has beaten Microsoft's heavyweight, proprietary approach bloody. Think ahead to a potential battle between Java and VB. Which technology will be more agile and able to respond faster with new compelling and enabling features? What reasons can anyone offer why history will not repeat itself?

- Focus. I guess my point is that standards that succeed on the Internet don't carry a lot of extra baggage. They don't have political entanglements. They're lightweight. They're not build into other products. How quick on their feet to we expect the proprietary MS technologies we've talked about to be in the face of a competing technology given their ship schedules, schedules of other products they depend on, and other commitments. Also remember that MS technologies are not optimized for the Internet... for instance, we'd want VB to support "safe VB". Adding "safe VB" to a product the size of VB, and maintaining it over the product lifetime, is a big deal. Not a recipe for success in a highly competitive environment.

Can anyone name just one proprietary standard that has become accepted on the Internet? Or even come close? I can't think of any. John Cordell mentioned an interesting case study, .GIF. As soon as one vendor claimed it as proprietary all we heard was people shouting about how quickly they were going to abandon it. So given our experiences of the past, how can we possibly expect that platform-specific, vendor-specific standards such as [OLE/OCX, VBA, .DOC, .XLS, docObj] possibly stand a chance of becoming prevalent? Can anyone present a convincing argument or scenario where this might happen?

I'm not saying that we should abandon work on supporting OCX's and making our browser more componentized-- we need to enable these other technologies and who knows, something like mediaview may come along and plug in and surprise us. But I think we should reconsider our priorities, and consider an alternative path to make sure we are in a position to be the leading (or a leading) Internet client without assuming VBA will take the Internet by storm. The alternative position is fairly obvious.

Yes, I'm turning to the subject of Java. Now the jury's still out; Java might suck, it might never catch on, it might not meet our needs. We need to learn more about it, see if we can figure out which way the wind is blowing, and decide. If it sucks then we go back to square one. However, let's assume it doesn't suck and is in a position to catch on. (There's an OK article on Java in the August Dr. Dobbs journal that John pointed out, by the way, if you want to learn more.)

Java is a lot like HTML. It has all the properties of HTML; multi-platform, multi-vendor (e.g. lots of critical mass), extensible, safe, focused. In fact, the potential battle we face if we enter the Java fray is very analagous to our web browser efforts vis a vis netscape and others. With Java, we don't have control of the standard (although we can gain defacto control by being a product leader, just as netscape has defacto control of HTML today)-- but in return we have the potential for much greater critical mass, which means more Web clients, which means selling more copies of windows, which means selling more copies of NT (assuming we cleverly build in synergy), and so on. Remember that market share and critical mass was key to MS' current success in other businesses (like Windows).

Possible complaints about Java, and why they shouldn't distract you:
"We don't own the standard!"
-- well, we don't own the HTML standard either, but that's not stopping us from participating that battle. If we owned the HTML standard earlier, then we never would have gotten critical mass to start the Web as it exists today, much less have it achieve its current growth rate. I predict Java will be like HTML in the sense that it is "owned" by whoever extends it in the most compelling way.
"With VBA, we get to sell copies of VB as the authoring tool! With Java, we don't sell VB."
-- with HTML we get to sell copies of Word + Internet Assistant as an authoring tool. Is there any reason that we couldn't make a Java authoring environment with Word or VC++ as the authoring tool? We feel fine about producing an HTML tool, we should feel fine about producing a Java tool. The stock price goes up whether we sell Word or VC++ or VB.
"There are no authoring environments for Java!"
-- don't expect that to last. (A good analogy: recall that the original motivation behind MSN was to push the state of the art by having text and graphics on the same page. It's easy to get overtaken by events.)

Unless we decide Java is terrible, then we should hop right in the ring and fight the same fight with Java that we're currently fighting with HTML, for all the same reasons. It's a good fight, we'll build mass and market share, $$ along the way, and leave ourselves in a good position for what comes next. If we enable other MS technologies like OCX's, VBA and what have you to plug in then they can have their run at it.

The biggest single mistake I feel we could make at this juncture is to not support Java because it would compete with similar MS technologies like VBA. I claim that proprietary MS standards are destined to fail as widespread standards on the Internet (I think history is on my side here) and we risk being left with nothing. The second biggest mistake I think we could make would be to become obsessed with the idea of enabling these MS technologies that we spend an undue amount of effort on it and miss out on other avenues. What constitutes an undue amount of effort? What are the other avenues? I don't know yet.

But from now on if somebody says, "we have to make [insert heavyweight MS technology of your choice here] THE standard for the Internet", you'll know why I have difficulty keeping a straight face.