From: Brendan Busch
Sent: Monday, January 27, 1997 9:55 AM
To: Steven Sinofsky; Andrew Kwatinetz; Michael Mathieu
Subject: FW: Thoughts from the Word 9 offsite -- applying it to all of Office 9

Some PPT debate raging in response to the Word 9 offsite summary:
--brendan

-----Original Message-----
From: Brendan Busch
Sent: Monday, January 27, 1997 9:42 AM
To: John Tafoya; Robert Parker; Hannes Ruescher
Cc: Imran Qureshi; Ralf Harteneck; Manish Vij; Roz Ho
Subject: RE: Thoughts from the Word 9 offsite -- applying it to all of Office 9

Viewing: As JohnT says, long term strategy for viewing should be to rely on the browser. Short term (PP9) the browser should be a great place for someone to review a PPT presentation, but not a great place for a presenter to *give* a presentation. We need to continue to support our great, market-leading presentation tools (including pres conferencing) in the PP9 timeframe. After that, hopefully the browsers + extensions will be good enough for us to get out of the viewing *and* presenting business and focus solely on creation/editing.

File Formats: In PP9, our HTML support should focus on accomplish the viewing task mentioned above: someone should be able to review a presentation (and it should remain beautiful and compelling) within an HTML browser. Certain presentation features will not be possible with NS/IE3, so we shouldn't beat ourselves to death trying to support these in HTML. We should keep in mind what the HTML version of the document will be used for in each release, and focus our energies on supporting those activities in HTML. In PP9, some functionality of PowerPoint will only exist in the .exe + our binary format.

Printing: I don't think we can expect browsers to print PPT data with the number of options and high quality that we can do directly (even in IE4 or 5). We support very intricate B/W options per-shape. We support multiple printed layouts, headers/footers, etc. We do *lots* of tricks to get good PCL and PostScript output. Printing from the browser should print what you see in the browser well (a problem for the NS/IE folks); we should push for hooks, so if you have PowerPoint, the browser can launch it and use PowerPoint to do the more high-end printing.

Editing: *We must excel here!* As mentioned above, in the next release we get out of the viewing business. The release after, we may get out of the presenting business. Eventually, maybe we even get out of the printing business. Long term, the **ease of creating a document**--the level of automation, intelligent formating, help with content and organization, etc--is the **only reason** that people will buy Microsoft PowerPoint (Office). We must start **now** building a huge lead in this area. Currently we are not way ahead of our competitors in the ease of editing--we are incrementally better in *some* aspects of editing, viewing, printing. Some would argue we are actually behind in this area.

--brendan

-----Original Message-----
From: John Tafoya
Sent: Monday, January 27, 1997 8:57 AM
To: Robert Parker; Hannes Ruescher
Cc: Imran Qureshi; Ralf Harteneck; Brendan Busch; Manish Vij; Roz Ho
Subject: RE: Thoughts from the Word 9 offsite -- applying it to all of Office 9

Since I just sent mail about it and I haven't read all your comments, I'll comment on one item: Viewing.

So are you saying that we shouldn't strive for pushing the browser to be the viewer of Office docs? If so, I disagree. HTML should absolutely be able to render Office data w/ full fidelity, and Office+Trident+IE should absolutely be able to support our "viewing features". I'm not convinced it (100% fidelity of *both* data and viewing features) can happen in the Office9 time frame, but we need to be moving in that direction. There are indeed tons of issues and current shortcomings - both technical and due to standards - but there is also a strong desire for end users and corporations to move towards a view-everywhere solution.

We should absolutely strive towards the browser as our viewing component. Imagine the cool stuff we could provide if we spend the time working on viewers and slide show instead on code that dramatically enhances the viewing experience for IE users. Navigator would still show the data just dandy but IE users will have more viewing features (but look the same since it's HTML). Corporations and ISVs could add browser addins to do all kinds of cool stuff. An Office/IE addin pack with gobs of extra builds and animations is an obvious one.

John

-----Original Message-----

From: **Robert Parker**
Sent: Saturday, January 25, 1997 7:26 PM
To: Hannes Ruescher
Cc: Ralf Harteneck; Brendan Busch; Manish Vij; Roz Ho; John Tafoya
Subject: FW: Thoughts from the Word 9 offsite -- applying it to all of Office 9

Though it is not a surprise, I fundamentally disagree with what Word is trying to put into HTML (and so would the W3C). The browser that they are suggesting would support viewing every feature of every application (which is ludicrous).
An quick analysis of their thinking:

1) HTML is a logical description, hence we can't easily build more complex pieces out of less complex parts. This line of thinking would lead you to believe that you have to have keys in the browser for everything. Already for V4 browsers -- Ihammer (for IE4), MacroMedia (for NetScape) are trying to create leveragable technologies (like filters, direct draw etc.) in order to add small physical building blocks. Much of the industry is focusing on how to plug their technology into the browser to give people building blocks because its clear that the browser can't be a universal viewer/navigator.
2) Writing HTML natively does not seem as important as reading it and round tripping it. The issue is that there is not one clear editor which does everything well so people demand that they can edit with multiple editors. Really this issue is just the clipboard/interop issue.
3) Browsers already only provide a part of the user's experience in HTML - Plugins, ole controls, Doc Objects, Java, VBScript are the designed mechanisms for building/enhancing the experience. Object model support, Filters, Layers etc. are technologies which allow even more extension of the browser. The browser teams know that they cannot be everything for everyone. Browsers have recognized that uniform physical descriptions that have been attempted in the past have failed. (Display PostScript, etc.) The reason for this is the additional layer involved (App->Common->Output). O/S such as windows has developed multiple direct APIs to handle this. Browsers came to the same conclusion - Graphics/Rendering for them is either JPEG, GIF (standard BMPs), or they delegate to a control, java app, java bean which will do the job directly. Its conceivable for them to manage high level objects, but for them to render everything would be inconceivably slow. It should be noted that no internet app that I have seen uses HTML as its file format. They are trying to provide a user experience which is compelling. (It **is** to their detriment that they do not use HTML more effectively [e.g. support indexing etc.], but its important that HTML be recognized for the tool that it is.).
4) HTML/Internet is not our only scenario - Its important but traditional uses for our product still exist and must be supported.
5) Their theories on printing are amusing. I believe that its important to be able to have something like a draft mode printing ability, but full-bore document typeset printing belongs in our apps. There are a number of reasons for this: First of all it is worth a lot to the customer (who should pay for it). Printing is really hard (to support all configurations, and complex graphics [just ask Tuan, JohnBo, JGibss etc.)). Print in Write v.s. Word, its pretty clear which document you would send to your boss (on paper). However if you want to review something its okay to print a doc from Write that was created in Word. I think its the same issue in printing on the Web, we need draft quality output to be supported (from HTML). Technically it will be a challenge just to get this. It should be noted that real printing is a huge chunk of code that browsers won't want to have. One thing I can say with absolute certainty - the browsers which ship Q1-Q2 (NS, IE) this year will **not** print reasonably.

To summarize, I think that I agree with all their suppositions and none of their conclusions.
0. Its difficult to influence or expect support for new HTML keys.
1. HTML support and roundtrip are important. (Must support all HTML features where possible, and roundtrip those we don't to be good editors.)
2. Must work well in the browser to make a compelling web page. We know that viewers don't do this for the obvious reasons.
3. Need to support some kind of printing in the browser.
4. Need to be able to script documents for web.

Where we disagree
0. We must focus on key areas where we need support and make sure that we at least get it in IE. Every investment in this area is worth it.
1. This does not imply that the apps use HTML as their native format any more than our in-memory usage of metafiles as a pictoral representation means that this would be our disk-based representation. We will need solid performance on read/write but we should be no worse than FrontPage in this area.
2. We need display solutions that work on the Web - Ihammer-ish 50K solutions are good, 1.5 Meg viewer bad. Ihammer-leverage HTML good, HTML graphics bad.
3. Its unrealistic to assume that browsers will support reasonable printing anytime soon. However some things which we can/should push for: A better hook so that we could launch the app for printing if you have it. Warn users that don't that they are only getting a draft and much better quality can be achieved with the app itself.
4. I think that we should try and come up with the best possible Web OM (which works in NS and IE4), then make this work in Office. It is important to realize that this is much more than simply separating out editing/viewing. Some editing can be done on the web, other stuff can't, some objects are supported (like groups) etc. etc. Basically to succeed we make a great Web OM and then pull some tricks to make this work in our mono-lithic apps.

5. Our target platform may have already shipped. It is quite likely that NetScape 3, IE 3 are **very** important targets. They **don't** have these kind of capabilities and we need solutions for these platforms. In-so-far as possible we must leverage these solutions for IE4 and the future. We don't want to be writing out content for every browser configuration since this would explode our file size, performance, and indexing.

Our Principals

Support LowCost Viewing Everywhere (through the browser)
Leverage Editing Environment to really raise the bar web page output
Leverage HTML
Maximize Sharing - Information only in one place wherever possible

Eaxmples of raising the bar are date/time metachars are easy to add/edit in PowerPoint. If your webpage has the right date/your name etc. these are cool features that raise the bar on other editors. Other examples are intelli-download. Our escher graphics control notices that the graphic is being downloaded slowly and switches to a no-frills draft mode which needs far less info (or better yet does background progressive rendering). The no-frill rendering is already supported by our apps and is another feature which we can dump on the web at low cost to raise the bar. Progressive rendering is something we've talked about but never implemented as a result it does not fit into this catagory.

Examples of leveraging HTML are:
Have output which is easily indexed. Make all text available to indexors. Avoid duplicate content where possible. Don't try and do things in HTML that its not meant for.

I don't want to dilute the message by going on-and-on about technical issues so I have left the overly technical stuff out of this email message. If you want more information please come and talk to me.

-----Original Message-----
From: **Ralf Harteneck**
Sent: Saturday, January 25, 1997 5:53 PM
To: PowerPoint 9.0 Planning
Subject: FW: Thoughts from the Word 9 offsite -- applying it to all of Office 9

These are interesting remarks, validating the planning we have been doing for Powerpoint 9.

Ralf

-----Original Message-----
From: **Michael Mathieu**
Sent: Saturday, January 25, 1997 4:51 PM
To: Ralf Harteneck; Brendan Busch; Peter Pathe; Andrew Kwatinetz; Antoine Leblond; Steven Sinofsky; Jon DeVaan; Duane Campbell; Chris Peters; Craig Unger; Richard McAniff; Daniel Bien; Eric Michelman; Brian MacDonald (Xenix); Bill Bliss (Exchange); Nathan Myhrvold; Bill Gates; Jon Reingold; Dean Hachamovitch; Brad Silverberg; Manish Vij; John Ludwig; Mark Walker (Word); Paul Maritz
Subject: Thoughts from the Word 9 offsite -- applying it to all of Office 9

On Thursday I spent all day at the Word 9 pm offsite. I think we made some pretty big breakthroughs there (at least in the ways that I've been thinking about Word and FrontPage.) I think we came up with some important things that also impact the way the other apps might think about their plans for Office 9.

I'm not sure where everyone is in their thinking right now, so I'll just put the basic flow of thinking down below, and just let me know if you don't buy into various pieces or need more explanation, etc. We went through something like the following discussion:

1 - HTML is important to our apps business (we talked about this just to make sure everyone was really bought in. We are.)
2 - To be a player, you have to write HTML natively (this is *playing* vs. *dabbling* that we do today)
3 - There's only one HTML, and it's defined by the browser (i.e. no inventing your own tags)
4 - Since it's HTML, it's the browser's responsibility to view the documents (A **key insight** from AndrewK)
5 - Since people print what they view, it's the borwser's responsibility to print the documents
6 - How do you get all of our existing features into the browser then? No consensus here, but maybe you don't get them all into the browser. Possible alternatives: a) revert to Office97 formats if you hit a feature you can't render; b) don't support all the features (this would probably require use to make a new product, for marketing reasons, even if it were from the same code base (this would be more like "WebOffice" than the "New Internet Application", I think); c) add the features to the browser (three options there -- get Trident team to do it; Office devs party on the Trident codebase to add features; figure out an architecture that lets us install low level extensions to the browser to give us what we need (no one knows how to do this today on either side of the equation.))

4+5 imply important things like:

- Apps need a way to preserve editing information within valid HTML (FP WebBot trick w/comments)
- File Open/d-click from shell of Office apps works just like any other HTML page -- it comes up in the browser

- If you want to edit the doc then open in the browser, and then hit Edit button (viewing outnumbers editing)
- Might have smarts to edit right away if we see that you're the author of the doc
- Need a new meta tag to indicate the "preferred editor" even though it's just an HTM file (e.g. don't want PPT files being editing in FrontPage, b/c we won't understand all of your WebBot junk, and won't have the ideal UI for editing it. But technically it would be possible.)
- All printing smarts from today's apps should migrate to the browser. e.g. footnotes would be displayed in one way on screen, but browser should be smart enough to print them at the bottom of the page. Same goes for all the PPT color printing controls and smarts about how to divide things up into speaker notes, slides per page, etc. Users should get all of this in the browser. We'd need to figure out a new architecture for doing this -- and yes, it probably wouldn't be as good in the beginning as it is in the standalone apps, but we'll fix that with a few turns of the crank.
- Viewing and Printing are just two more examples of things which get "horizontalized" (to quote Andy Grove), when the file format for apps get horizontalized (Manish's insight)
- [Just came up with this one while I was writing this --] Of course this all has a big impact on our programmability story. The object model for all of our runtime capabilities should be aggregated onto the *browser* object model. The portion of the object model that affects the editing environment is really entirely separate and distinct. That's not to say that editing isn't part of the browser object model. It's the editing *environment* that keeps it's own OM outside of the browser. So, the runtime object model is yet another thing that gets horizontalized by the browser with the common HTML format. Of course, the individual apps could provide runtime OM's which have redundant functionality to the built in Trident OM, but that makes sense b/c it's much easier to code with task-specific OM's, rather than just very low level control.

What does all of this mean for PowerPoint?

- Save natively as HTML (don't know if this is already in your plans)
- File Open goes to the browser. Edit goes to PowerPoint
- Transitions are built into Trident. You'd be less feature rich with Netscape, but we ship IE with our products.
- Need to figure out how to get required viewing and printing functionality into the browser. Short term hack way might be via Java applets, but you're more likely to want to put this into our browser -- that really improves our platform story, b/c now it becomes a great platform for people to target with presentation graphics packages.

What does all of this mean for Excel?

- Save natively as HTML
- File Open goes to browser. Edit goes to Excel
- Excel's special printing knowledge needs to work somehow in the browser. Will browsers ever handle the 2-D scrolling region as well as Excel? That might be a particular investment area we want to look at for the future. Or the 2-D-ness might just be particular to the online editing environment that Excel provides, rather than the viewing, which takes place in screen/page-size chunks.
- Need to decide level of functionality that goes into browser vs. addons. e.g. sorting, pivoting(?), filling in forms, etc. Today you can get Java applets that do a lot of what you want for basic list management. Data entry is the huge terrible part.

What does all of this mean for Access?

- DBC, Reports, and Forms saved as native HTML
- Table creation, query building, and programming are still native to Access (as is the MDB format for tables, indices, etc.)
- File Open goes to browser, as does everything outside of Design mode
- Need to build in some intelligence into the browser for how to handle large data sets, and how to do all of the data access remotely; also all of the banded report printing -- probably very different for printing reports than how they'd ideally show up in the browser (e.g. report header in a frame at the top. Same for page header (what does that mean for a bottomless report in a browser?) and all the footer stuff.)

What does all of this mean for Outlook?

- HTML as native format. My understanding is that they're already going fullbore on this.
- File Open goes through Browser -- that's not just for email and news, but views and view elements as well.
- Move to a web UI for viewing. Seamless integration for editing tools. This is in contrast to today's model where Outlook is almost like a wrapper of its own (besides the browser) and has it's own non-standard views. This is longer term, but basically all of the great views features in Outlook should move *into* the browser, rather than the browser becoming just another view in Outlook. Despite that it's what Netscape is doing, it's not the right thing to have this inside-out model. The browser is the one thing that should control views, and provide the runtime functionality for interacting with them (e.g. forms.)
- More file printing intelligence moves to the browser.

What does all of this mean for FrontPage?

- We're already HTML native format
- File Open through the browser is a model that we've been using for one area of FP3 improvements. You end up browsing around and then hitting the Edit button. We're making changes here in FP3 so that you can edit the page directly without having to first open the web in the FP Explorer. That makes things way faster.
- Printing – well our only printing is through MFC. It's got tons of bugs, but we don't really get complaints. IE already does a better job than FP at printing pages. It just makes sense to put it in IE.
- We do need to think about hosting our Explorer views in the browser somehow. This is similar to the Outlook case where the views really belong *inside* the browser, rather than outside.

In a sense, this whole email can be summarized as "What changes when you don't have your own file format?" That's a consequence of HTML. AndrewK's insight that the browser should do all viewing is really crucial, and I don't think it's something that we've ever thought about before (witness our Word, Excel, and PPT Viewers.) It makes so much sense. And thinking through the implications of that for our apps will make us all work better in a world where don't have our own file format. This also lets us think a lot more about how the browser could become a platform for "real" applications, with a whole range of sophisticated needs that wouldn't necessarily be built into the browser. This is just a starting point.

Thanks,
-Mike