AJAX for Designers

Why are we talking about this now?
Why are we talking about this at all?
What new paradigms for browser-based interfaces are we facing?

by

David Heller

published on *--Engage @*
www.synapticburn.com

October 11, 2005
Brooklyn, NY, USA

Introduction

For some AJAX is marketing speak for something that has been around for years. For others, it is the salvation they have been looking for, for seemingly centuries. I would tell the former group that the latter group finally caught up, and you should revel in the growth of your new community and accept them with open arms. Snobbery really won't help anyone.

Recently Jesse James Garrett from Adaptive Path wrote an article for the duo at Ok-Cancel called "Why AJAX Matters Now". As usual Jesse is insightful and dead on about why AJAX has arrived. More aptly I would say that Jesse describes well why the world is now ready for AJAX where earlier uses of the same technology had really no chance of taking off. Jesse does still speak of AJAX's arrival in his article, and while it is not such an important distinction, I would like to correct him and say that AJAX (as our former group notes above) has been around for a while, and that other environmental variables have been put in place to make AJAX consumable.

What makes AJAX consumable now are 3 things:

- Jesse gave the technology XMLHTTPRequest (created by Microsoft for IE 4.0) a great catchy name that was non-intimidating: Asynchronous JavaScript and XML-AJAX.
- The general web-browsing community had reached a point where older browsers that no longer supported this technology were not being supported. This is VERY recent. Netscape 4.x users were holding back everyone. Specifically two big leaps helped this happen:
    - Safari was standard on all MacOS X systems.
    - Firefox has had rave success as an alternative to IE on both Mac & Windows AND it runs on Linux as well with relatively little differences in the way that it's engine translates HTML, JavaScript and CSS.
- Someone with mainstream clout put their money where their mouth is and developed an application for the world to see that it not only works, but makes a HUGE difference to the user experience of their product—Gmail and Google Maps. Google also used it in smaller ways as well, like Google Suggest that are still in their Google Labs area. Other companies quickly followed suit like Flickr and Basecamp.

The article mentioned above has had so far a small back and forth comment battle going on. There have been 2 distinct themes in the conversation:

- There are better technologies out there: Java, XUL, XAML and Flash (not in the thread, but why it isn't there is beyond me.
- AJAX kills the "back-button" and thus the stated page.

What follows is an attempt to answer those concerns in as real a way as possible, and also to further elaborate on the relevance and importance of AJAX at a more tangible and tactical level. Core pieces that will come out of this paper are:

- Why other technologies are not as good as AJAX for MOST web-based (in the browser applications).
- Why AJAX by itself doesn't kill the "back-button", but rather how AJAX changes the paradigm for what a page really is, and also why this still differs from desktop software and SHOULD.
- Lastly, a set of suggestions as to how we might need to change the way we craft user interfaces that utilize AJAX.

Richness and Choosing the right Technology

The question of richness has been discussed here and there. I have an article to be published in the inaugural edition of UXmatters sometime soon on the topic. The short of it though is that richness of any type creates a more engaging experience, by creating controls and methods for increasing the context between interconnected activities throughout the workflow of the completion of a task.

Lots of technologies are able to achieve this: Java, XAML, XUL, Flash, ActiveX, .Net, and oh, yea! AJAX. In this article, it would be too much to do an in depth comparison of all these and other technologies. Frank Ramirez and Luke Wroblewski did a pretty good job with this already in their article Web Application Solutions: A Designers Guide.

What I would like to discuss is what qualities of your design you need to consider before choosing a Rich Internet Application technology and why companies like Google have made a pretty good choice with AJAX. To even begin to do this we have to understand the various levels of possible richness necessary in a design as well as what is there to be gained or lost with various technologies. To do this analysis requires a set of analysis questions, which I'll attempt to go over here.

What are my deployment issues?

There are a series of questions that need to be answered around the issue of deployment of an application. I'll list a set off (by no means is this comprehensive, but should set you off on your own hunt):

What type of audience is my application for? Consumer? Business? Both? A subset of one? What techno-savvy assumptions can I make about them?

This question is very important as it can determine right away which technologies you can't use. For example, many corporations prohibit specific technologies, or they lock down user's machines and don't allow them to do installations. This issue will come up a lot. The issue of level of savvy is also important, in that many users are not comfortable "installing" things.

Do I plan on "tweaking" rapidly, or will my product follow a more standard release cycle?

Even with today's desktop applications that can self-update over the Internet there is a price to pay for even this level of updating of binary applications as opposed to ASCII-based code like AJAX, XAML, or XUL.

Am I willing to use closed or otherwise non-ubiquitous technologies?

Closed in this case means, only works on a limited number of platforms. For example Java is a language and engine that is not open, but it runs on a number of platforms, while XUL is an open-source language that has limited environments that it can run on.

What are my team's skills?

Hey! It's gotta be built, right? Yes, you can bring in new talent, but new technologies have learning curves of how to manage and deploy and if you don't have in-house talent and you have limited learning curve scope, you might be limited by the number and types of technologies you can use.

In looking at the answers to some of these questions I find that XAML and XUL while interesting are irrelevant technologies. XAML **will** be relevant in about 1-2 years, especially in business environments where you can count on MS only environments. XUL is just on too limited a platform (Mozilla). I find though that the real battle of technologies is between the following: Java, Flash, and AJAX. These "richening" agents are powerful and useful. They are ubiquitously available and have good deployment methods. Besides the questions above, you also need to look at the requirements of the design.

What issues in my design will help me choose a technology?

So like the above questions regarding deployment, but there may be further cues and clues from your design that guide you in your technology choice:

How much integration do I need to add between the browser and the rest of a user's local desktop environment?

Many applications require some type of local file access which could be greatly improved over the more limited browser capabilities. Selecting multiple files, dragging and dropping between the desktop and the browser (or representation of the server content), access to sound controls for a microphone, a security card reader or other peripheral device, application integrations like maybe to synchronize data between a PIM (Outlook, Entourage, iSync), or finally application "sharing" like that done in LiveMeeting, Raindance, or WebEx.

Does my design require cinematic effects between scenes, or just within them?

Cinematic is a term that the people at Laszlo use to describe the quality of RIAs that is most important to their technology. My paraphrasing of this, is whether or not, motion and sound are going to be used to increase the context of the interactions taking place in an application.

At what point do I need to manage calls to and from a remote server in my design?

> Again, these are either intra- or inter- pages. This is also a very big value question. Will increasing the appearance of synchronicity in my application between the client and server add enough value to the user to warrant the added "expenses" in building a rich application?

To cut to the chase, AJAX is NOT good if your answer to question #1 is fairly complex and deep. AJAX still plays by the rules of the web browser. It just creates a wormhole in a web-page so that it can pretend a subsection of a page is well a page of its own like a frame. Flash also has the same problems. Java, ActiveX and .NET are the only solutions here.

AJAX is also not so good when you want to have highly cinematic richness in your application. Flash is really built for this, as is XAML.

I think you get the gist of the types of questions you need to be asking yourself as you think about your audience, your designs and other constraints. The important piece to take away, is that there is no perfect technology. You have to choose a technology relative to the design of the application and the constraints offered by the problem you are designing for in the first place.

What about AJAX and the "page"?

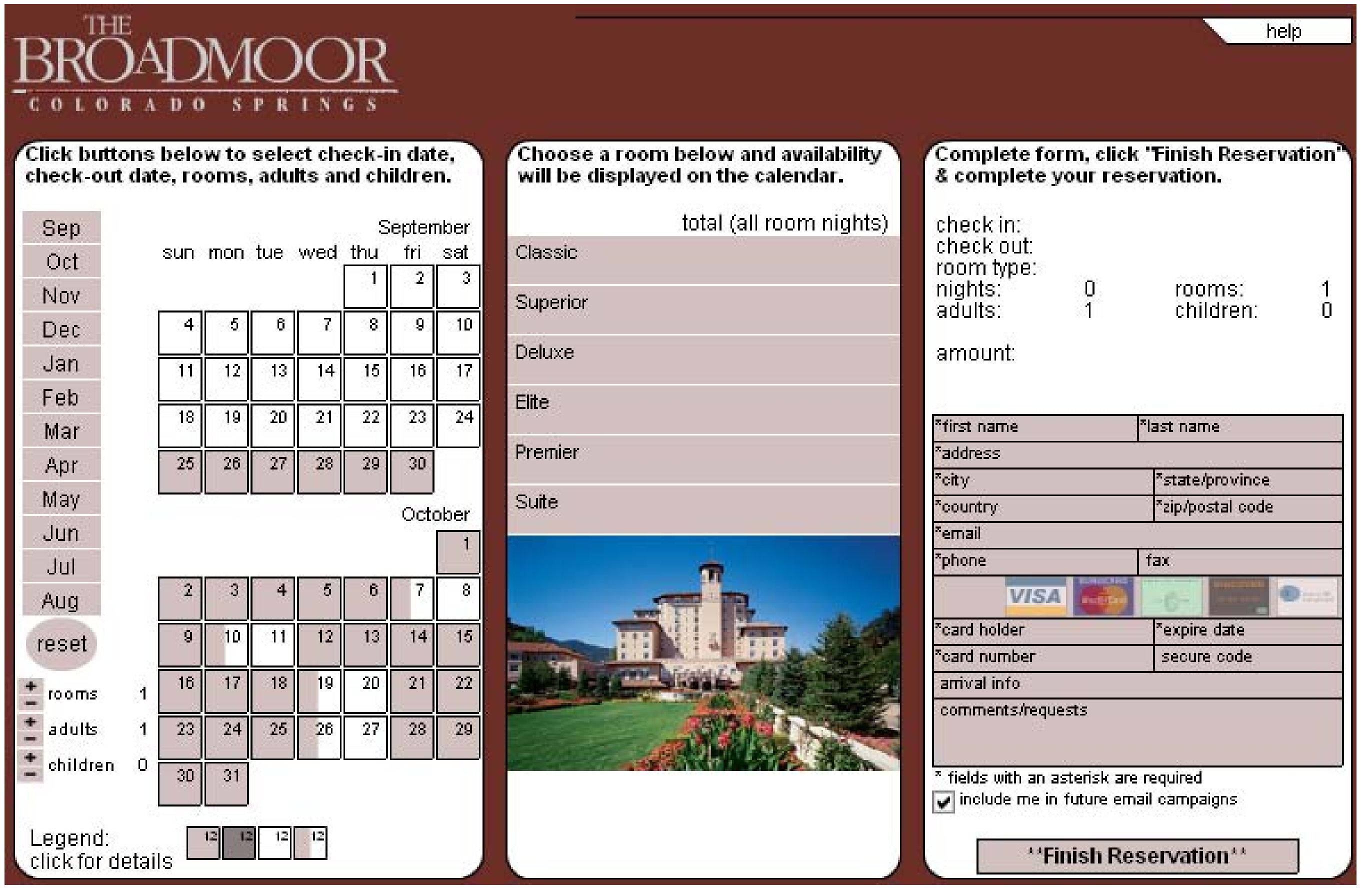
People seem to have such weird criticisms of AJAX solutions many of which I don't quite understand. It sounds like the same "99% Bad" criticism of Flash. People looked closely at bad, early attempts instead of realizing that there is a new design language being developed which in turn requires some time to experiment with it. Most technologies don't really force bad designs. They lean people towards a period of early serendipity through which designers learn and explore.

Good criticisms of AJAX do exist. One good criticism of AJAX is around accessibility. Layering and dynamic server calls don't seem to be well supported by accessibility tools like screen readers. These tools change and deploy slowly and AJAX has not been on their radar yet.

Another criticism is that AJAX incorporates new or untested conventions by creating custom controls, and creating new navigation styles. I don't buy this one. The old conventions are fairly well tested, and often fail during those tests. The conventions are there, but are they really all that valuable. Some are, yes, but some are not. And those are worthy of further exploration. Heck! Even the ones that are "good" should be explored some more. I can't imagine anyone saying there isn't room for improvement.

So the page itself has been around since the very beginning of the web. While it was clear we were talking about HTML files and not the number of pages we would get in a printed version, it didn't matter. Each hyperlinked item returned another HTML file that we called a page. The metaphor also derives from what was being delivered in that page-usually textual content that would come from white papers, brochures, magazines, books, or other forms of publications. When a user moved from file to file, it had the same effect of turning a page. There is a moment when both the page you were on and the page you are going to are both unreadable at the same time. So the "blink" or flash that occurs when I click on a hyperlink or graphical button. The last derivation of the "page" comes from how we have traditionally designed web sites and web applications-paper static prototypes usually known as wireframes.

The real qualities of the "page" that are being challenged with new technologies is its static nature and the maintenance of its state by the browser through its "history"— navigation-back and forward buttons. An example of the purely dynamic view (instead of page) is the classic Flash example of the Broadmoor Hotel reservation site. But a newer AJAX application doing something similar is the new IM client (and aggregator) on the web, Meebo.



help

THE BROADMOOR
COLORADO SPRINGS

Click buttons below to select check-in date, check-out date, rooms, adults and children.

Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
reset

September

sun	mon	tue	wed	thu	fri	sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

October

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

rooms	1
adults	1
children	0

Legend: click for details 12 12 12 12

Choose a room below and availability will be displayed on the calendar.

total (all room nights)

Classic
Superior
Deluxe
Elite
Premier
Suite

Complete form, click "Finish Reservation" & complete your reservation.

check in:
check out:
room type:
nights: 0 rooms: 1
adults: 1 children: 0

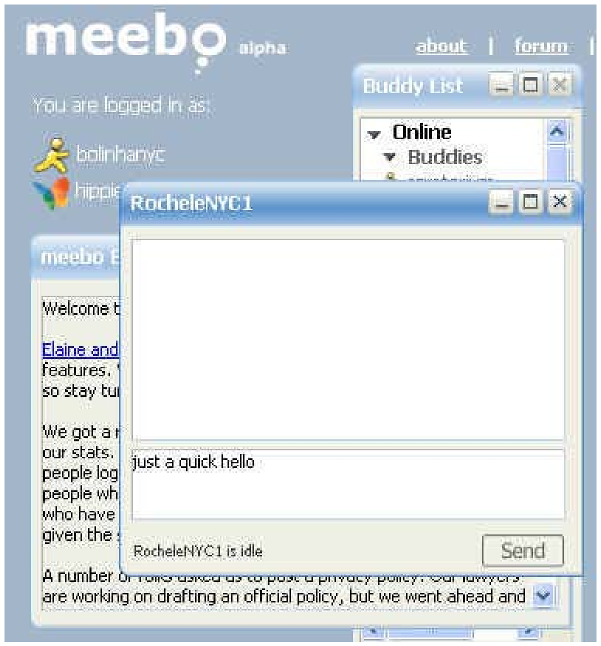
amount:

*first name | *last name
*address
*city | *state/province
*country | *zip/postal code
*email
*phone | fax
VISA
*card holder | *expire date
*card number | secure code
arrival info
comments/requests

* fields with an asterisk are required
✓ include me in future email campaigns

Finish Reservation

The Broadmoor Hotel tries to do everything in one view. This includes searching, information, selection, and fulfillment.



meebo alpha
about | forum |
You are logged in as:
bolinhanyc
hippie
meebo
Welcome t
Elaine and
features.
so stay tu
We got a
our stats.
people log
people wh
who have
given the
A number of
are working on drafting an official policy, but we went ahead and

Buddy List
Online
Buddies

RocheleNYC1
just a quick hello
RocheleNYC1 is idle
Send

The Meebo application uses the browser as a second desktop where you can in that browser window/tab/frame have an aggregated buddy list with the ability to open and manage multiple IM sessions.

But these are either new, or unproven examples of Rich Internet Applications. The examples that have been touted as successes for AJAX do not do this "death of the page" structure. In fact, many rely on the old page structure, but have made some level of innovation on top of it to create a seemingly completely new user experience.

There are two primary innovations to the page paradigm that are so far definable:

Inline Updating/Form Submission

In this type designers have realized that there are moments when the type of data being updated or submitted back to the server doesn't necessitate going to a next page, and in fact the experience is made better by NOT going to a next page.

Page as new category or grouping

This is just a simple restructuring. Before "the page" meant a moment in time when you had to go back to the server for more information. But this first new form, has defined a page to mean a grouping of like elements, usually that share a view, or a single point of categorization, and the contents also share a common relationship with each other.

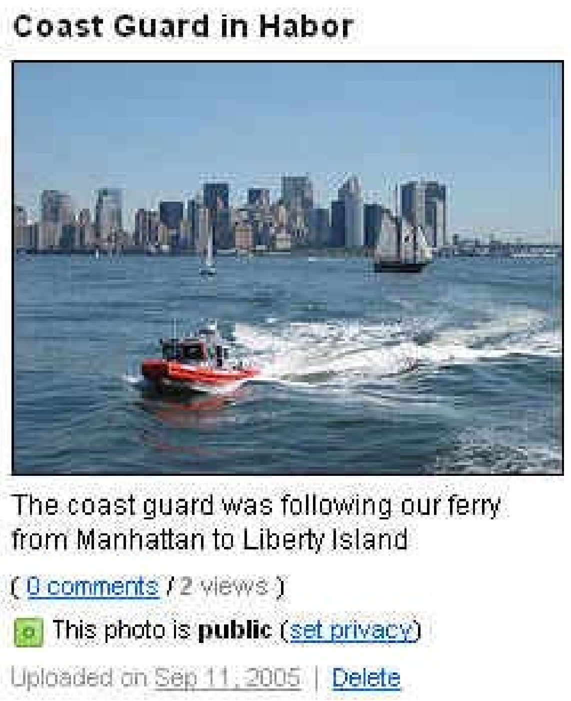
In both cases the history and back-button still have a function, but there exists a flow between historically recorded moments by the browser.

Examples of AJAX

Inline Updating: Flickr, Google, Gmail

This next section I want to use examples to demonstrate how the page is maintained why richer a more engaging interaction is added. By using AJAX this richness is achieved without any plug-ins whatsoever, so there are no further technologies to install besides the ubiquitous browser. Further, the three major browsers IE, Firefox/ Netscape/Mozilla and Safari all support it.

As you can see here in this image, there is already a description. But I can click on the existing text ...



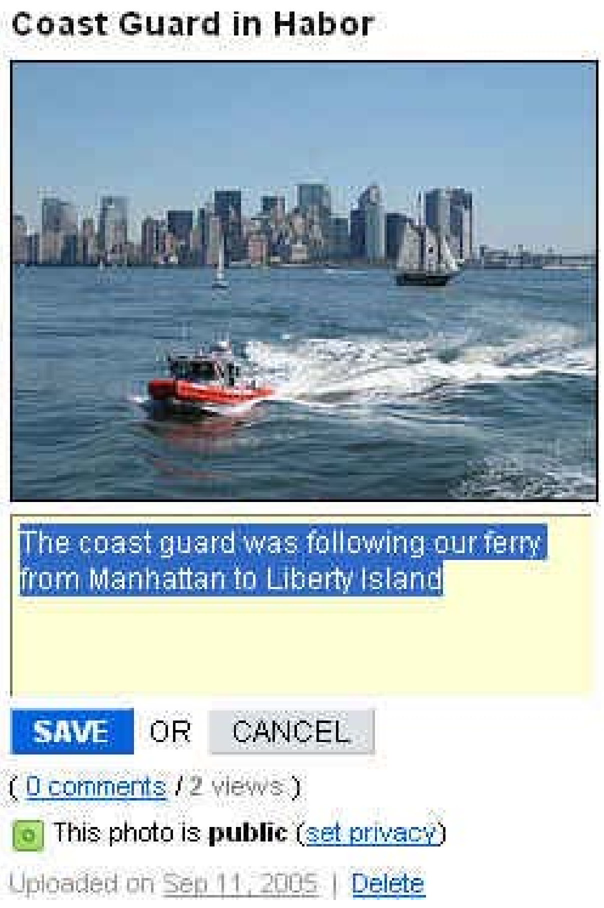
Coast Guard in Habor

The coast guard was following our ferry from Manhattan to Liberty Island

(0 comments / 2 views)

This photo is **public** (set privacy)

Uploaded on Sep 11, 2005 | Delete



Coast Guard in Habor

The coast guard was following our ferry from Manhattan to Liberty Island

(0 comments / 2 views)

This photo is **public** (set privacy)

Uploaded on Sep 11, 2005 | Delete

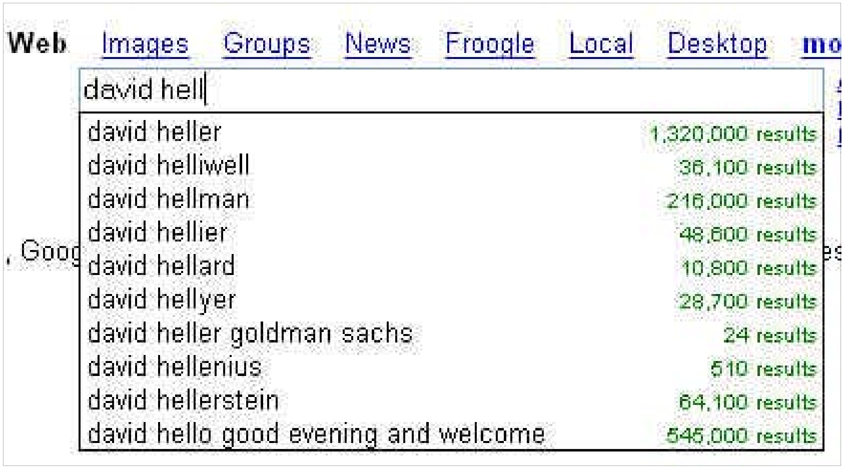
And as you see here, the text is instantly and inline placed i a text area and the appropriate submission triggers are made available as well. Clicking on "SAVE" will return me to the previous image with the changes to the description.

The above scenario all happens without ever loosing the context of that image as being listed on a page with other images. The screen never blinks at all, and further I am given very good feedback from the system while data is being submitted.

Now the page metaphor is indeed maintained, but this element is not part of that metaphor. Why would it need to be? In fact, by not having a page metaphor we manage details like hitting the back-button to see old data, and have accidental re-submissions. Here, hitting the back-button has the appropriate effect of moving me to the previous view that got me to this view—i.e. The selection of a set, or a search results screen of tags.

I think this level of additional richness has been wrongly overlooked by the greater RIA thought leaders, and has now been championed correctly by AJAX folks. Previous choices of technology could do this sort of thing by having a flash plug-in doing it, but these were much harder to manage, and weren't really promoted by Macromedia, who wants people to think bigger and use the entirety of what Flash as a platform offers, instead of thinking to scale.

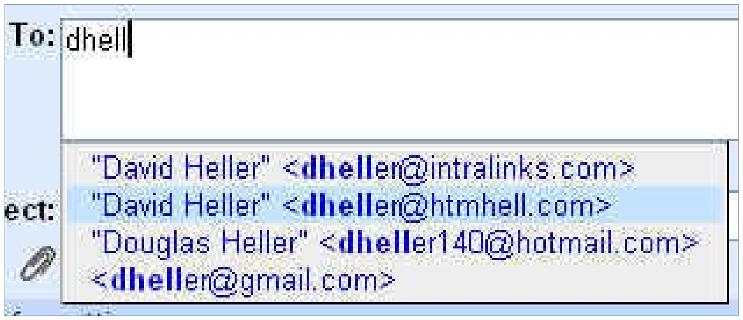
Another version of a widget like this comes from Google. The Google Suggest feature is still in what they call their Google Labs area, so many people do not get to take advantage of it as it is not part of the primary Google search interface, but it has been appropriated into Gmail. Other follower companies like Yahoo have brought it into its own applications (as has Zimbra, MSN Hotmail [beta], etc.).



david hell

david heller	1,320,000 results
david helliwell	36,100 results
david hellman	216,000 results
david hellier	48,600 results
david hellard	10,800 results
david hellyer	28,700 results
david heller goldman sachs	24 results
david hellenius	510 results
david hellerstein	64,100 results
david hello good evening and welcome	545,000 results

As you can see as I type in something into the search text field, the system goes to the server and fills in a short selection area with even further information about the number of results. A user can use their arrow keys to tap down to a final selection.



dhell

"David Heller" <dheller@intralinks.com>
"David Heller" <dheller@htmhell.com>
"Douglas Heller" <dheller140@hotmail.com>
<dheller@gmail.com>

Here we see the same thing, but now Gmail is using suggestions from both my address book, and from all the e-mail I ever sent and received. This was the first time we saw this happen on a server, hosted application.

Re-defining the page itself.

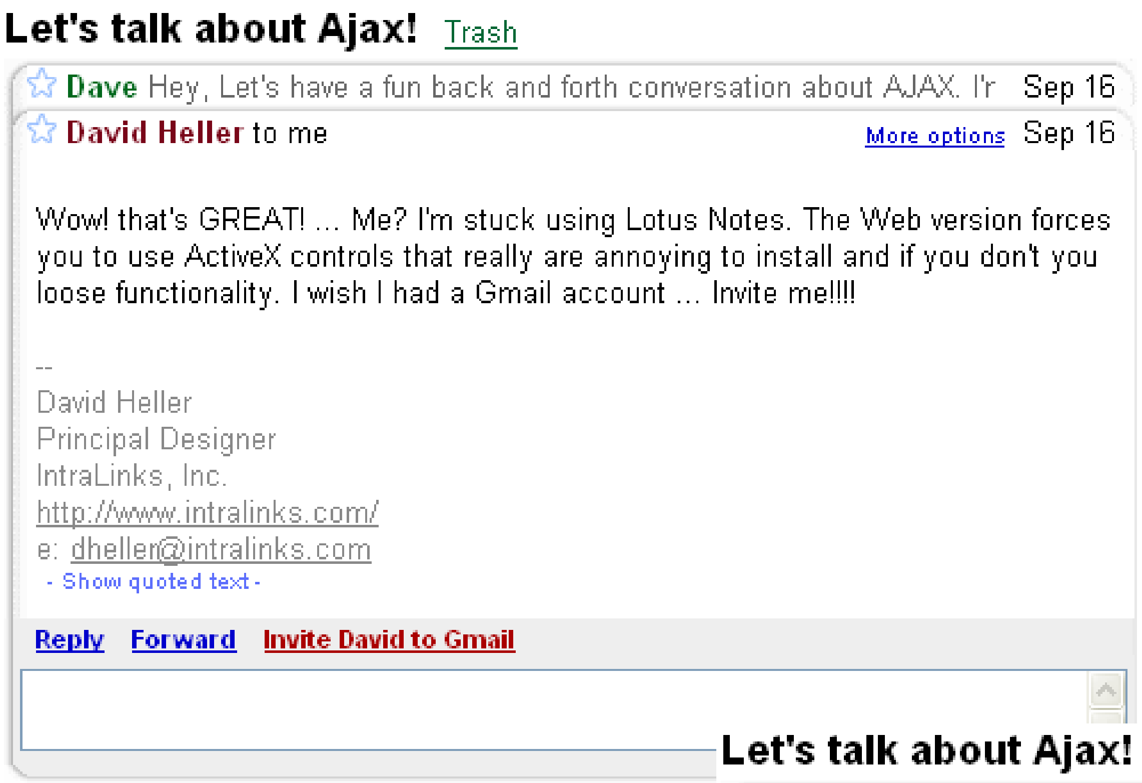
Probably the poster-child of AJAX is Gmail. We've already looked at an AJAX widget with the address book suggestion tool. Now, we are going to look a little more broadly at the structure of the very Gmail application.

Before we look at Gmail, let's first look at what the old version of e-mail used to be. Yahoo Mail and Hotmail are very popular tools. Yahoo Mail is about to get re-launched with its own innovations based off of a not so recent purchase, Odd Post. Since this is still in closed Beta, I cannot directly address this upcoming version. Old Yahoo and Hotmail though both use a very similar metaphor, which is based off of the old page metaphor. A page is meant for a single primary piece of content surrounded by a navigation system. Any messages that are "related" to each other have that relationship maintained only in the folder system. Desktop applications have had the ability to view messages according to thread or conversations for years, but this has not been done by many if any web-based e-mail programs. Well, until Gmail came out.

The creation of the conversation, as a unified grouping in a single page innovated the concept of the page for e-mail programs. Notice what I said here. "it innovated the concept of the page". I didn't say it threw it away, In fact moving from conversation list view—i.e. the inbox—to the conversation view itself and back again, is still easily maintained by most web browsers. I for one use my back button (well <alt>-left arrow) to go from conversation to conversation list view all the time, and it works great.

What we'll see below is that the AJAX innovation that makes this possible, useful, and usable is that I don't have to have a long scrolling page of the entire conversation (subject thread). There is an expand collapse mechanism in place. If AJAX wasn't being used here (or other richening engine) then it would be interminably slow to move from collapse to expand and visa versa through all the messages in a conversation. Further, when replying or forwarding a message within a conversation, I don't loose the context of the conversation and have full access to the expand collapse of the messages while writing the new message, which when done and sent, I still don't leave the conversation, but my message is converted from a writing form to a read-only message.

Ok, stop describing it and show it to me …



Let's talk about Ajax! Trash

Dave Hey, Let's have a fun back and forth conversation about AJAX. I'r Sep 16

David Heller to me More options Sep 16

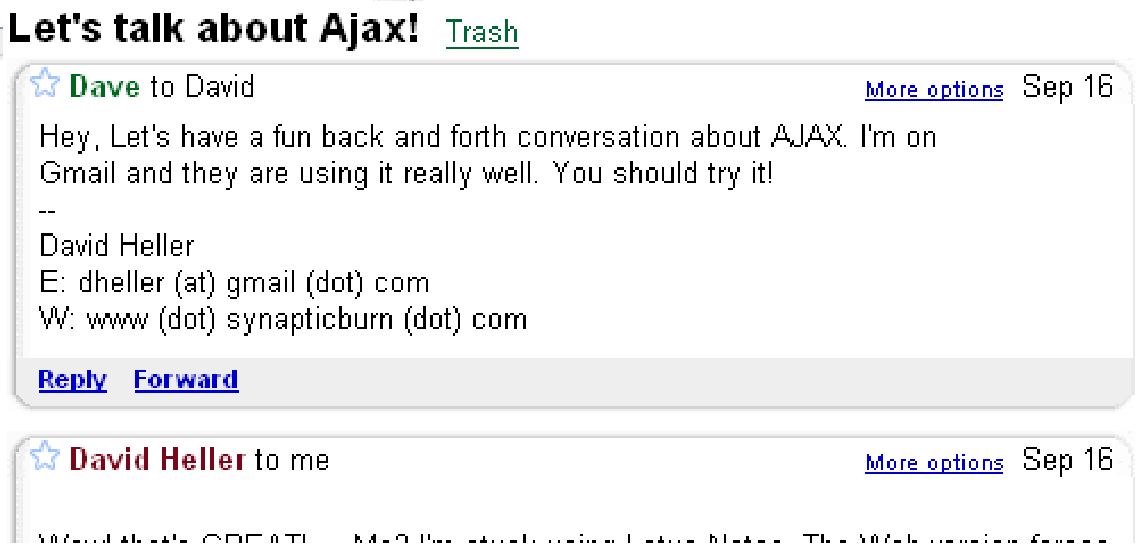
Wow! that's GREAT! ... Me? I'm stuck using Lotus Notes. The Web version forces you to use ActiveX controls that really are annoying to install and if you don't you loose functionality. I wish I had a Gmail account ... Invite me!!!!

--
David Heller
Principal Designer
IntraLinks, Inc.
http://www.intralinks.com/
e: dheller@intralinks.com

- Show quoted text -

Reply Forward Invite David to Gmail

Here is a 2 message conversation, where the originating message is collapse behind the most recent message of the thread. Notice how the quoted material is automatically recognized and placed in under a collapsed area.



Let's talk about Ajax! Trash

Dave to David More options Sep 16

Hey, Let's have a fun back and forth conversation about AJAX. I'm on Gmail and they are using it really well. You should try it!
--
David Heller
E: dheller (at) gmail (dot) com
W: www (dot) synapticburn (dot) com

Reply Forward

David Heller to me More options Sep 16

Here we see the message being replied to. By its position directly under the message that is being replied to, it communicates a clear context of the flow of reading and then replying to a message. (I must admit that sometimes it is difficult to get this to work properly all the time. Sometimes the message I am replying to is not the message in the conversation I want to be replying to.)

What does this all mean to our practice?

To put it simply, we need to innovate our methods of design to keep up with the innovations around the page metaphor itself. Since we have barely come to agreement about the old way of designing solutions and creating deliverables, I'm not sure I can with confidence supply any methods that would not be as equally controversial as the old methods.

As designers there are two distinct tasks that we have to achieve. We need to conceive, innovate, sketch, brainstorm, or otherwise create solutions and we then need to communicate those solutions. Obviously, an idea conceived that cannot be communicated through to execution is not much of an idea. But also, how we communicate needs to be tailored to different purposes, goals and motivations of the audiences at any given stage of design.

On top of this complication it is increasingly clear that the tools we use do indeed make designers gravitate toward different types of solutions. A tool is a focal point, and focus is a blinder in reverse. Meaning, when we focus tightly, our peripheral blurring space tends to increase. Some tools have a tight focus, and limit not because an idea can't be communicated through that tool, but because the tool itself pulls our attention away from such ideas. I believe that 2-d tools such as Illustrator, Visio, and Photoshop while valuable, do not allow us to envision beyond the 2-d. In this case the 3rd dimension is not depth, but rather time. Again, some great designers work in these tools, and have been quite successful. But people have also been successful with designs without doing any research, generative or evaluative, but it doesn't mean those design methods don't add further, measurable value to a total design process.

For me working on interactive products, I believe that an iterative studio approach is most appropriate. How we convert this to documentation later, is a different story. By using interactive tools like Dreamweaver/GoLive or Flash we have at our disposal not just the drawing tools of the 2-D solutions (why I prefer Flash to HTML editors, as it has better drawing tools), but also the ability to add behaviors, and envision how these behaviors will play out over time and within the appropriate context.

By using Flash you can create pages using primary "screens" but you can create movie clips or use nested screens (or more likely both) to bring to life the dynamic nature of the applications you want. You can also create array objects to hold data for you to use so that there is even a greater "realism". Of course this gets done through iterations, where you first create frameworks that get refined toward a complete designed solution.

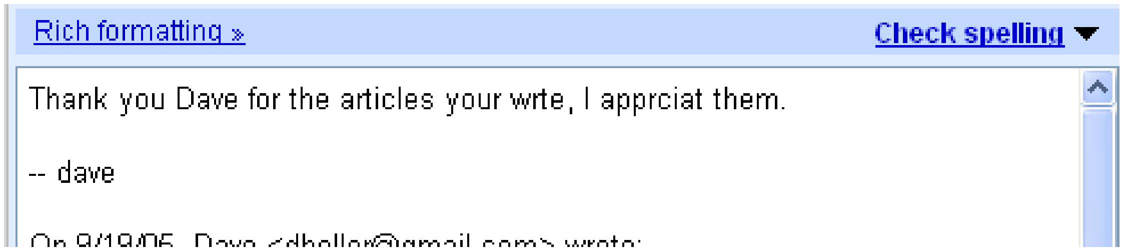
Conclusion

Jesse James Garrett has been a great spokesperson for the utility of AJAX-based web applications. His mere coining the term AJAX has done more than the creation of the technology itself did over a half-decade ago. But touting AJAX and pointing to successes of others is just the beginning. We need to better articulate the patterns being created and how old patterns are evolving through great innovations. In so doing, we can create a better repeatable common language, so that new conventions can be built and shared, and so appropriate processes and tools are utilized.

What we don't need any more of, is nay-saying. There are indeed some technologies that are 99% bad, but these are few and far between. Now, many technologies have been abused from Flash to the theory of relativity, but that does not mean that there is not further value in them beyond its criminal uses. That being said, nay-sayers do give us caution, as we move forward with innovations. They are good at pointing to areas of concern, so we can adjust our practice over time. We must be cautious that we never throw out the baby with the bath water.

Appendix A: Gmail Spell Check

Gmail in my opinion not only brought conversations and labels to web-mail, but also made spell-check really work. It's in context spell checker is just a superb piece of design and worthy of demonstrating here.

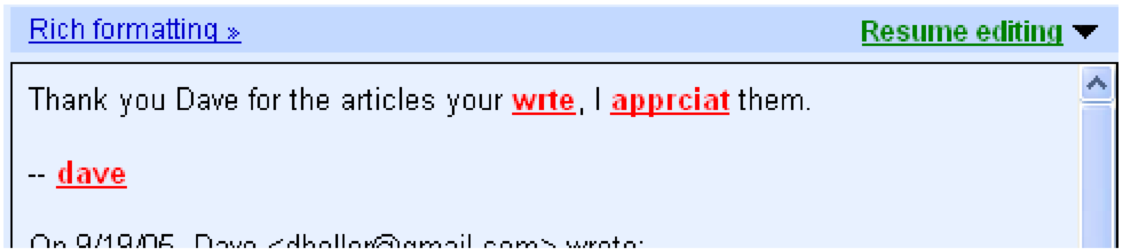


Thank you Dave for the articles your wrte, I apprciat them.

-- dave

On 9/19/05, Dave <dheller@gmail.com> wrote:

Notice above the text box on the far right "check spelling". Clicking that will change the text area from writeable to read-only.

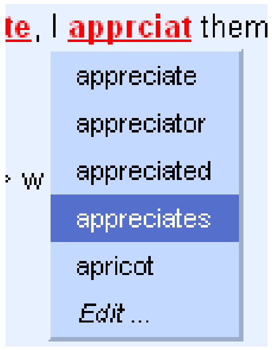


Thank you Dave for the articles your wrte, I apprciat them.

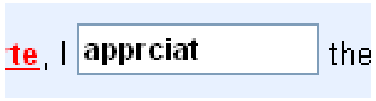
-- dave

On 9/19/05, Dave <dheller@gmail.com> wrote:

The read-only version highlights in red and underline which items are spelled wrong. Does the underline do enough to tell you it is clickable? I think so.



By clicking the red, underlined text, you get a dropdown of suggestions and the option to edit inline.

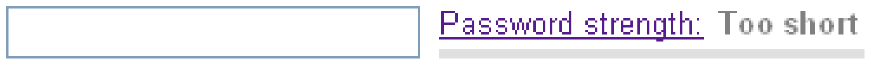


Selecting "edit" turns that word into a pre-populated text input field.

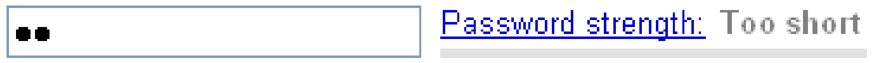
As you can see this spell check takes on the expected qualities of a desktop application, while still maintaining some tried and true conventions determined by a web browser.

Appendix B: Is My Password Good Enough? (Gmail)

I came across this functionality by accident, when trying to set up an account. I found the process to be a little weird, but this discovery made the whole thing worthwhile. Basically, when you are creating a password for a Google account, there is a meter widget next to where you type the password that tells you how "strong" your password is. Strength is determined by several factor sthat are explained in a popup window (not shown).

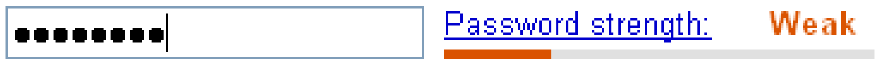


When you arrive you are presented with a link to a popup. See next page.



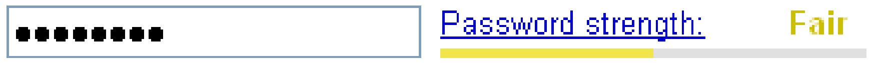
typed: pa

When first typing your password, it will tell you that you haven't met the minimum requirements.

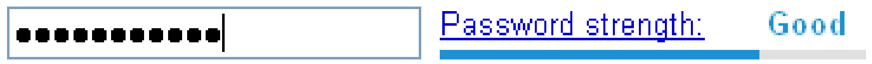


typed: password

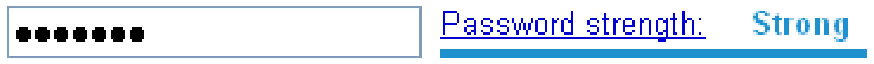
Then as you type it will based on what you type, change its meter as is appropriate.



typed: pa$$word



typed: password001



typed: pa$$wrd

I'd like to point out that the "Fair" value is actually more characters than the "Strong" one. The reason for this is that when I was typing along I eventually reached a point where the system could tell that this was a real word. Real words are easy to guess and so not as strong. AND it can recognize the dollar signs as S's.