Time for Comet?

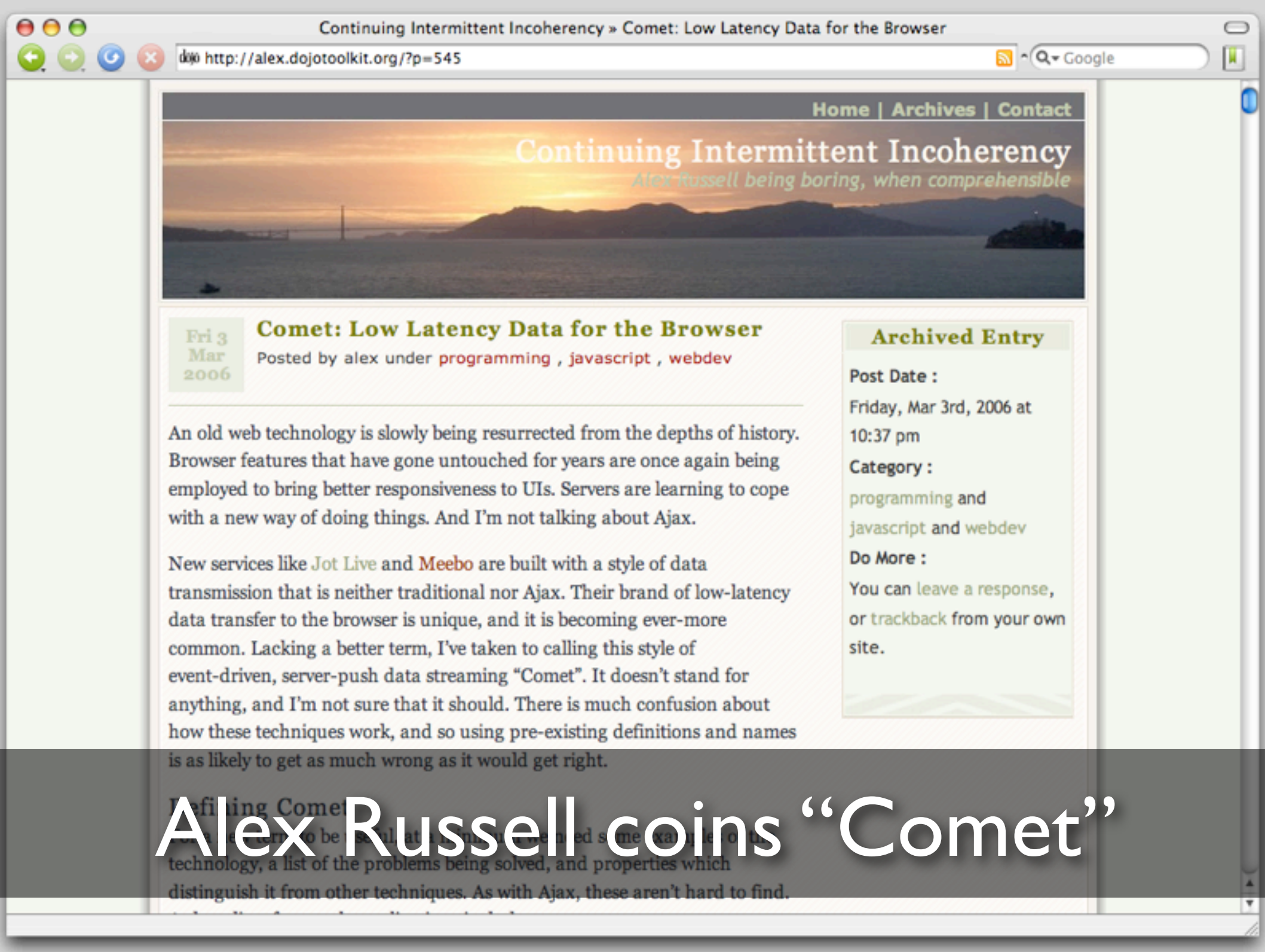
Simon Willison

Yahoo! Web Development Summit

5th December 2007

Header image: http://svn.xantus.org/shortbus/trunk/cometd-perl/html/

A new name for a very old technique

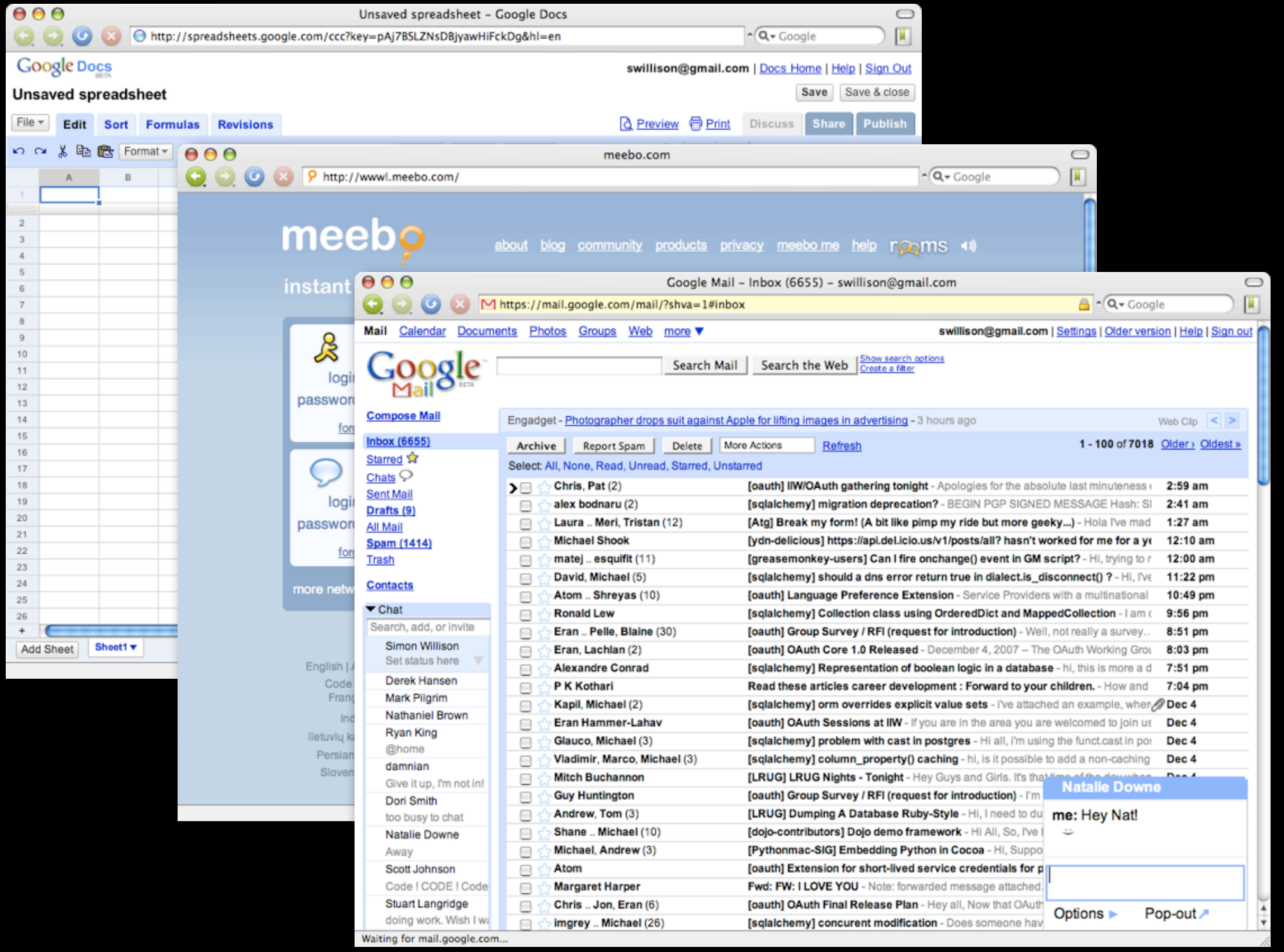


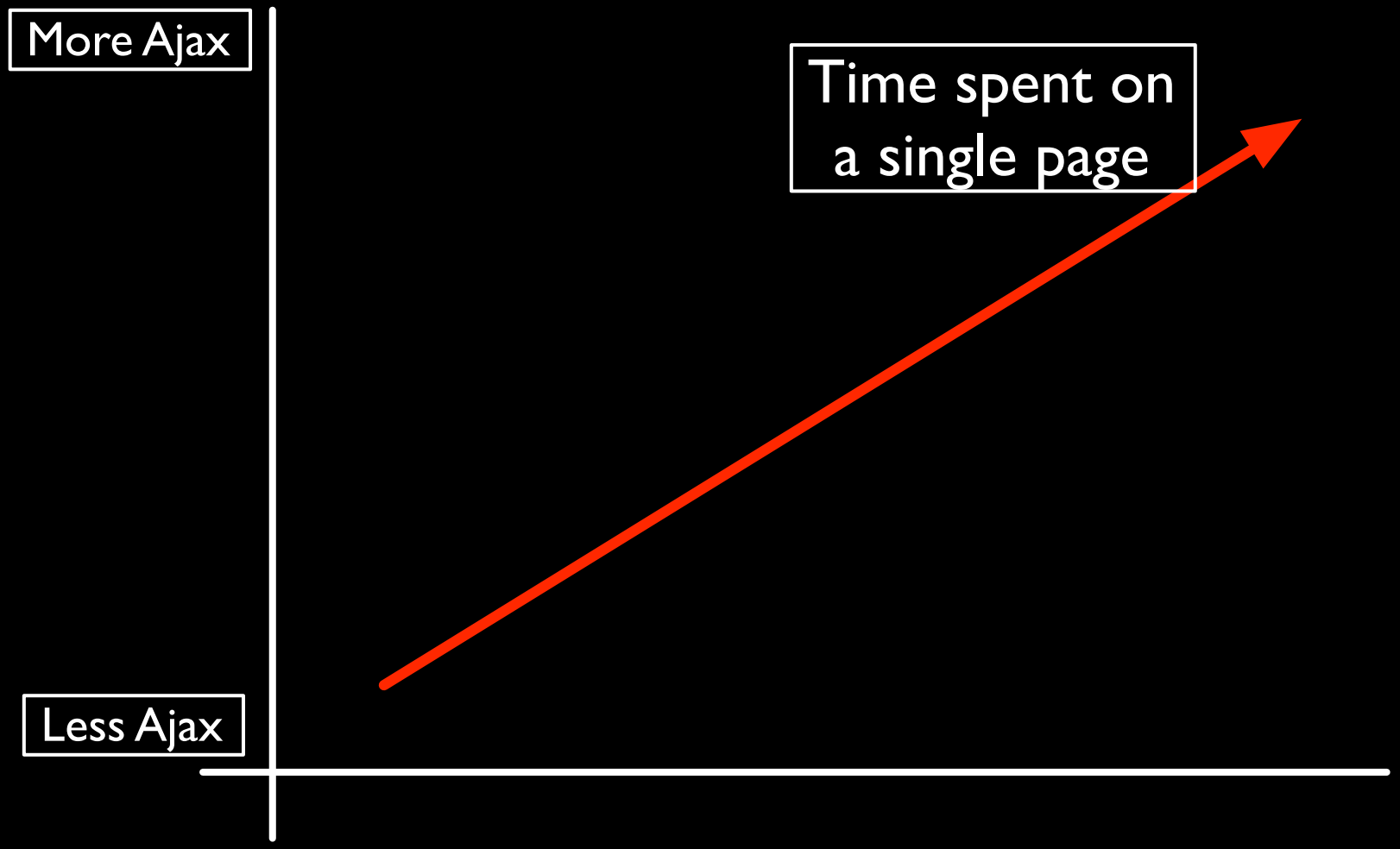
Alex Russell coins "Comet"

Deliberately named after a kitchen cleaner

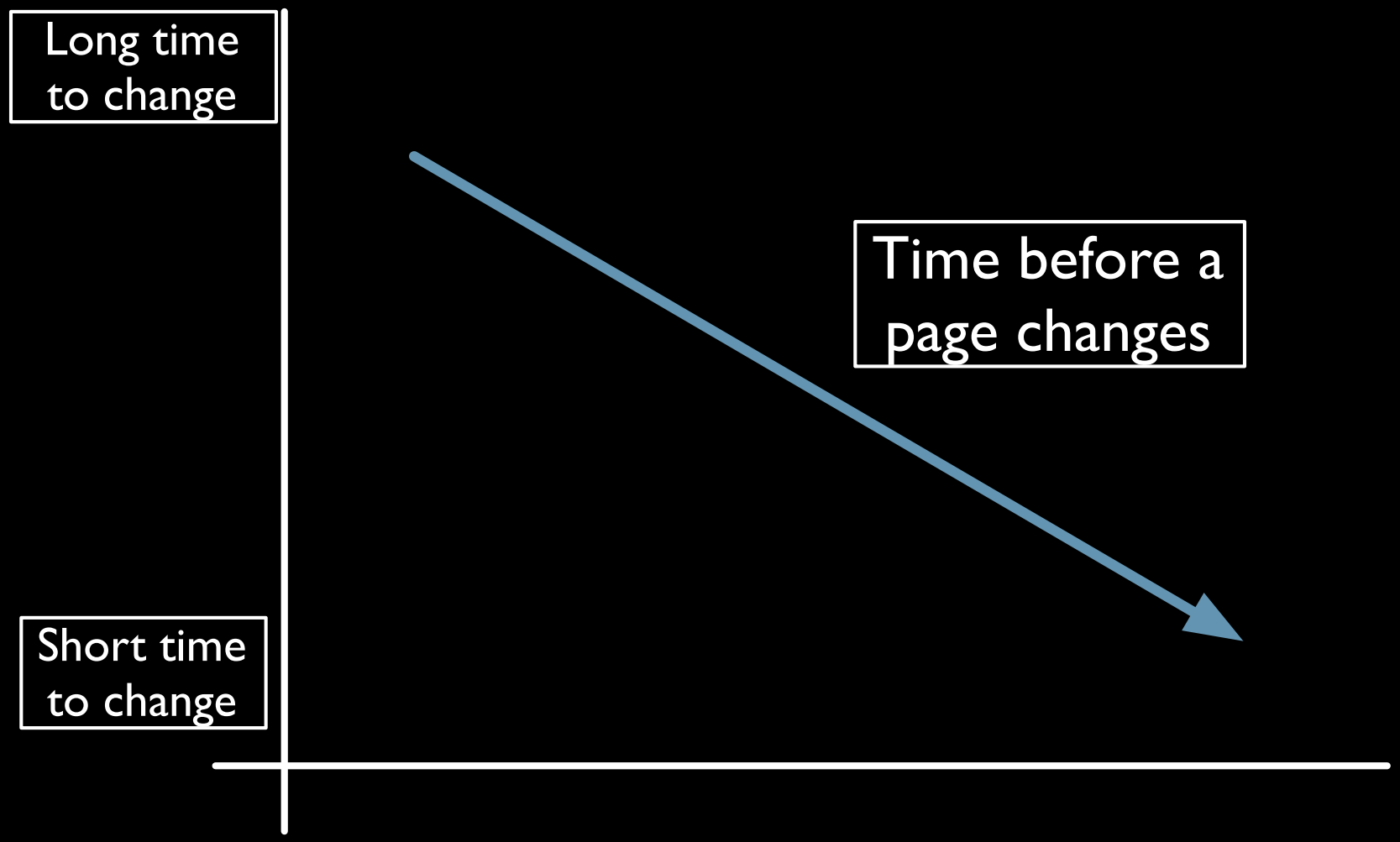
Fundamentally, [*Comet applications*] all use long-lived HTTP connections to reduce the latency with which messages are passed to the server. In essence, they do not poll the server occasionally. Instead the server has an open line of communication with which it can **push** data to the client.

Who’s using it?

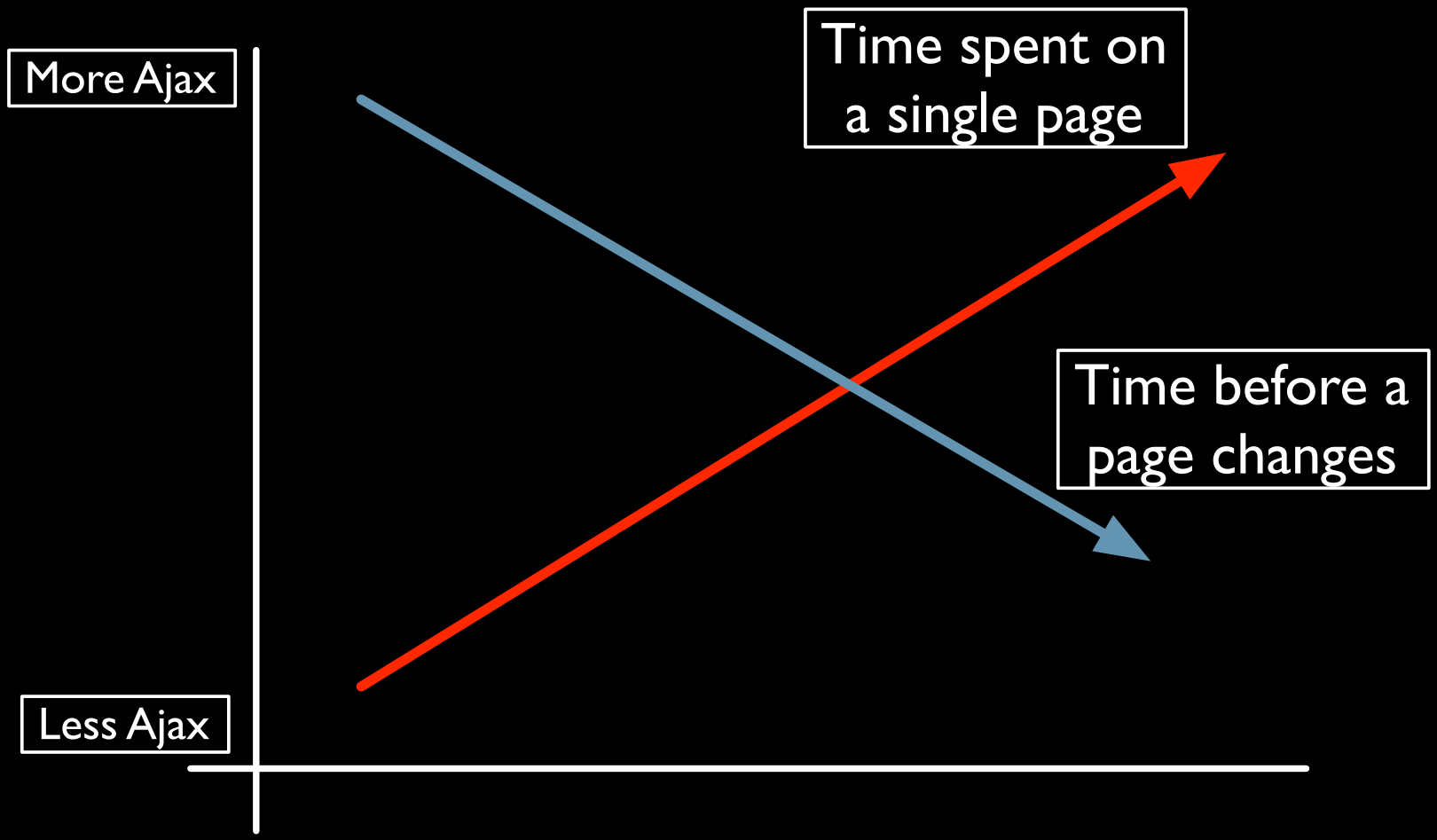

Unsaved spreadsheet - Google Docs
http://spreadsheets.google.com/ccc?key=pAj7BSLZNsDBjyawHiFckDg&hl=en
Google Docs
swillison@gmail.com | Docs Home | Help | Sign Out
Unsaved spreadsheet
Save
Save & close
File
Edit
Sort
Formulas
Revisions
Preview
Print
Discuss
Share
Publish
Format
Add Sheet
Sheet1
meebo.com
http://www.meebo.com/
meebo
about
blog
community
products
privacy
meebo me
help
rooms
instant
Google Mail - Inbox (6655) - swillison@gmail.com
https://mail.google.com/mail/?shva=1#inbox
Mail
Calendar
Documents
Photos
Groups
Web
more
swillison@gmail.com | Settings | Older version | Help | Sign out
Search Mail
Search the Web
Show search options
Create a filter
Compose Mail
Inbox (6655)
Starred
Chats
Sent Mail
Drafts (9)
All Mail
Spam (1414)
Trash
Contacts
Chat
Search, add, or invite
Simon Willison
Set status here
Derek Hansen
Mark Pilgrim
Nathaniel Brown
Ryan King
@home
damnian
Give it up, I'm not in!
Dori Smith
too busy to chat
Natalie Downe
Away
Scott Johnson
Code ! CODE ! Code
Stuart Langridge
Engadget - Photographer drops suit against Apple for lifting images in advertising - 3 hours ago
Web Clip
Archive
Report Spam
Delete
More Actions
Refresh
1 - 100 of 7018
Older ›
Oldest »
Select: All, None, Read, Unread, Starred, Unstarred
Chris, Pat (2) [oauth] IIW/OAuth gathering tonight - Apologies for the absolute last minuteness 2:59 am
alex bodnaru (2) [sqlalchemy] migration deprecation? - BEGIN PGP SIGNED MESSAGE Hash: SI 2:41 am
Laura .. Meri, Tristan (12) [Atg] Break my form! (A bit like pimp my ride but more geeky...) - Hola I've mad 1:27 am
Michael Shook [ydn-delicious] https://api.del.icio.us/v1/posts/all? hasn't worked for me for a ye 12:10 am
matej .. esquifit (11) [greasemonkey-users] Can I fire onchange() event in GM script? - Hi, trying to r 12:00 am
David, Michael (5) [sqlalchemy] should a dns error return true in dialect.is_disconnect() ? - Hi, I've 11:22 pm
Atom .. Shreyas (10) [oauth] Language Preference Extension - Service Providers with a multinational 10:49 pm
Ronald Lew [sqlalchemy] Collection class using OrderedDict and MappedCollection - I am 9:56 pm
Eran .. Pelle, Blaine (30) [oauth] Group Survey / RFI (request for introduction) - Well, not really a survey.. 8:51 pm
Eran, Lachlan (2) [oauth] OAuth Core 1.0 Released - December 4, 2007 -- The OAuth Working Grou 8:03 pm
Alexandre Conrad [sqlalchemy] Representation of boolean logic in a database - hi, this is more a d 7:51 pm
P K Kothari Read these articles career development : Forward to your children. - How and 7:04 pm
Kapil, Michael (2) [sqlalchemy] orm overrides explicit value sets - I've attached an example, wher Dec 4
Eran Hammer-Lahav [oauth] OAuth Sessions at IIW - If you are in the area you are welcomed to join us Dec 4
Glauco, Michael (3) [sqlalchemy] problem with cast in postgres - Hi all, i'm using the funct.cast in po Dec 4
Vladimir, Marco, Michael (3) [sqlalchemy] column_property() caching - hi, is it possible to add a non-caching Dec 4
Mitch Buchannon [LRUG] LRUG Nights - Tonight - Hey Guys and Girls. It's tha
Guy Huntington [oauth] Group Survey / RFI (request for introduction) - I'm
Andrew, Tom (3) [LRUG] Dumping A Database Ruby-Style - Hi, I need to du
Shane .. Michael (10) [dojo-contributors] Dojo demo framework - Hi All, So, I've
Michael, Andrew (3) [Pythonmac-SIG] Embedding Python in Cocoa - Hi, Suppo
Atom [oauth] Extension for short-lived service credentials for p
Margaret Harper Fwd: FW: I LOVE YOU - Note: forwarded message attached.
Chris .. Jon, Eran (6) [oauth] OAuth Final Release Plan - Hey all, Now that OAuth
imgrey .. Michael (26) [sqlalchemy] concurent modification - Does someone hav
Natalie Downe
me: Hey Nat!
Options
Pop-out
Waiting for mail.google.com...



(Slides courtesy of Joe Walker)

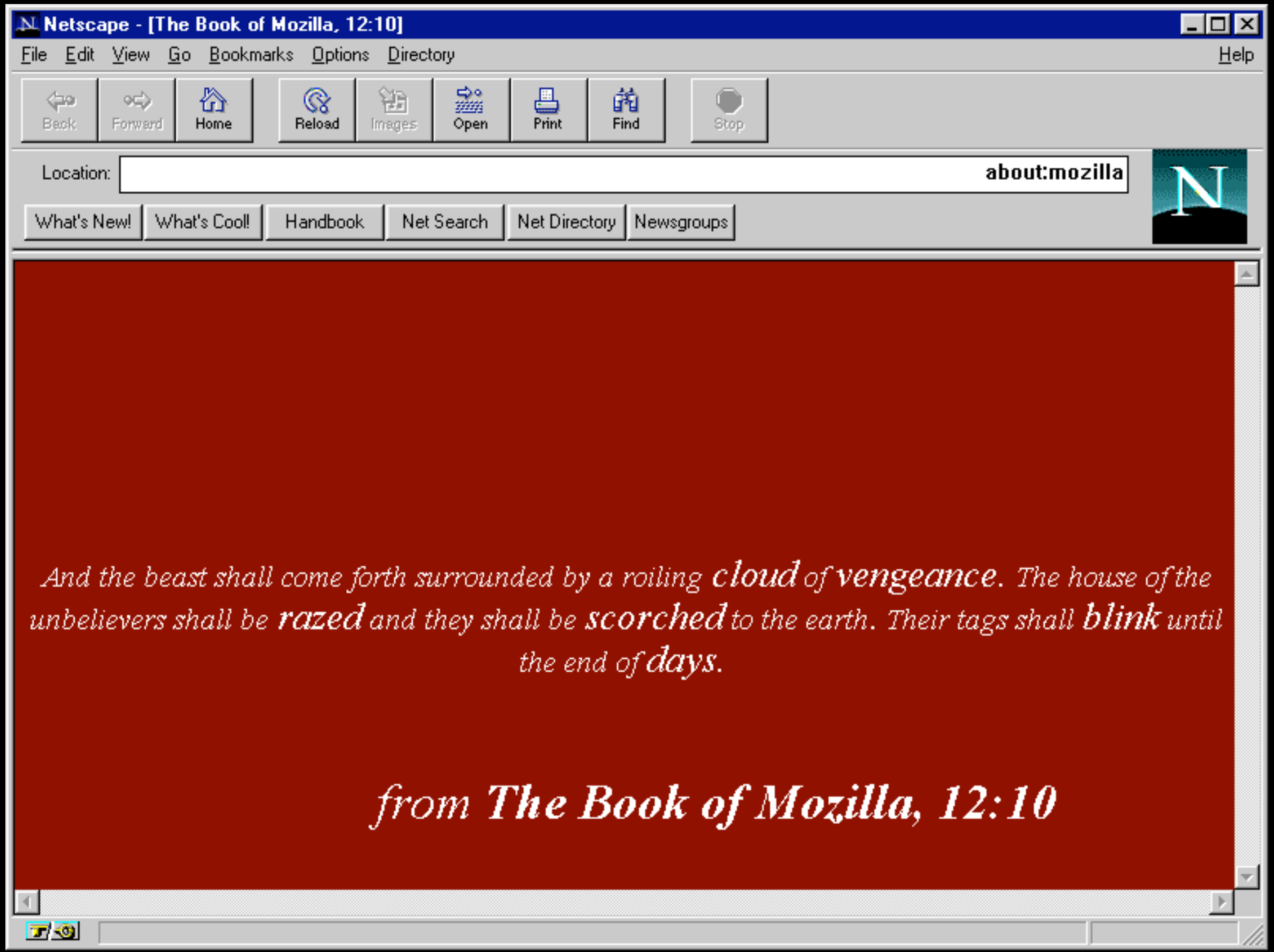


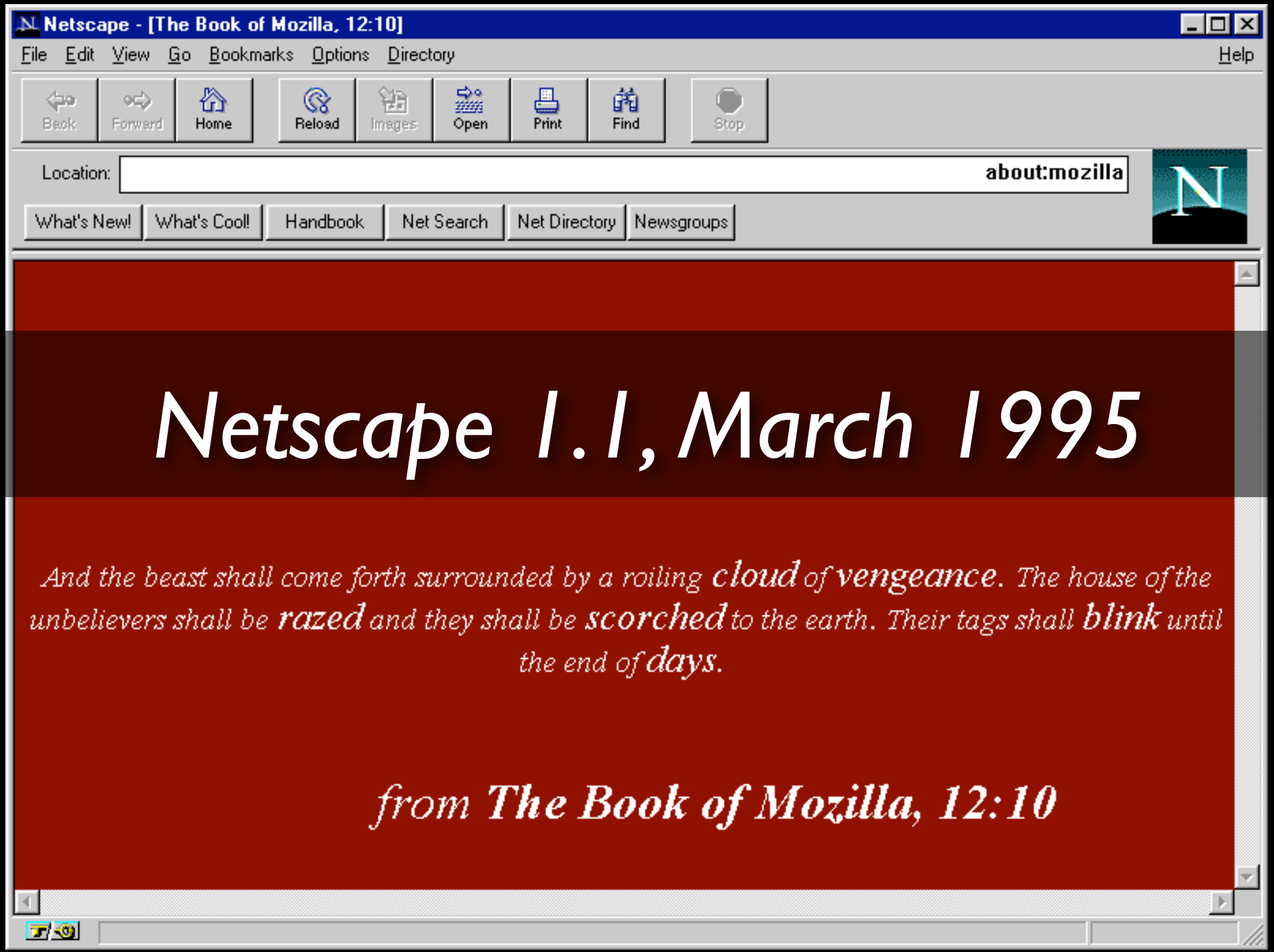
(Slides courtesy of Joe Walker)



(Slides courtesy of Joe Walker)

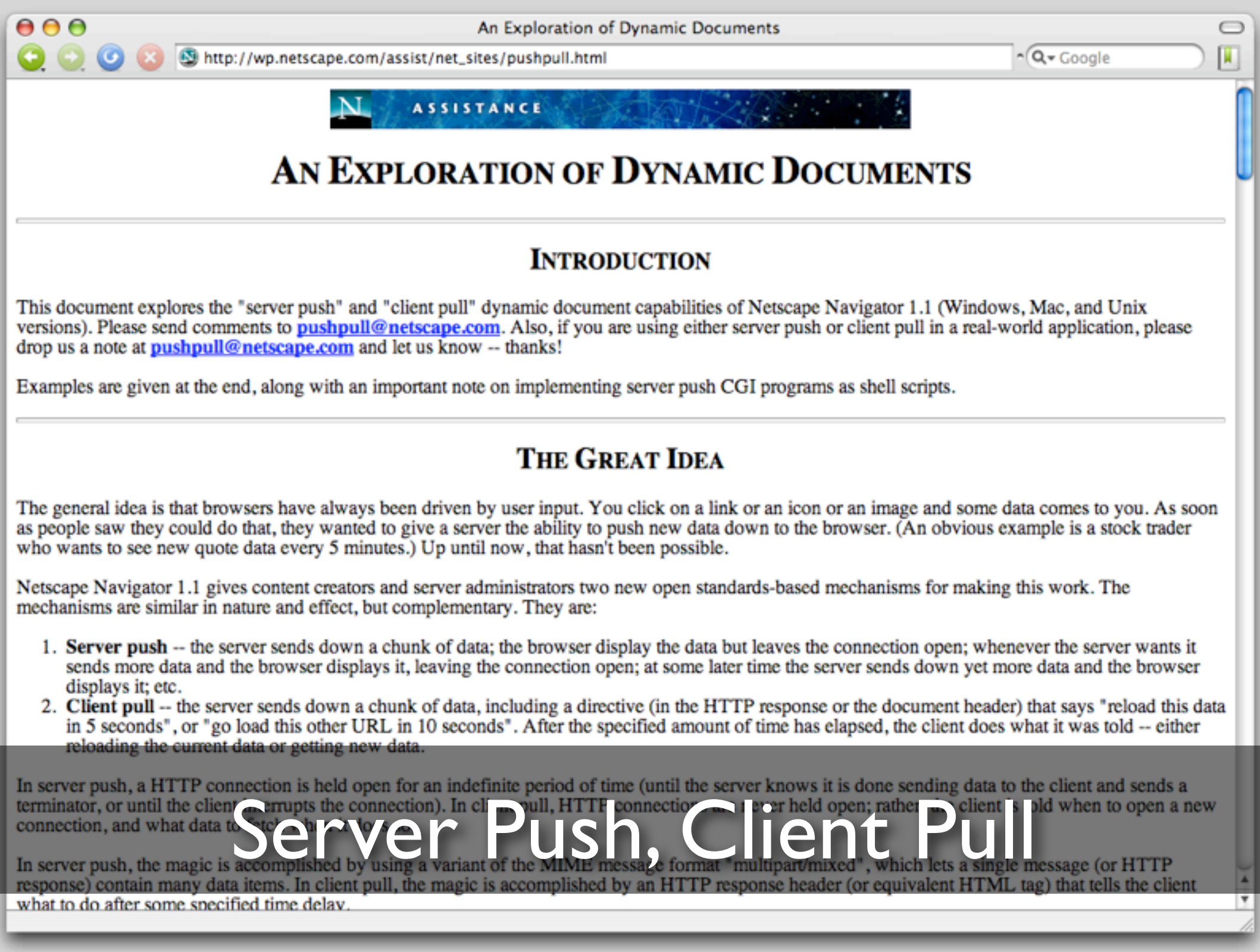
Early Comet


Netscape - [The Book of Mozilla, 12:10]
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
Location: about:mozilla
What's New! What's Cool! Handbook Net Search Net Directory Newsgroups
And the beast shall come forth surrounded by a roiling cloud of vengeance. The house of the unbelievers shall be razed and they shall be scorched to the earth. Their tags shall blink until the end of days.
from The Book of Mozilla, 12:10


Netscape - [The Book of Mozilla, 12:10]
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
Location: about:mozilla
What's New! What's Cool! Handbook Net Search Net Directory Newsgroups
Netscape 1.1, March 1995
And the beast shall come forth surrounded by a roiling cloud of vengeance. The house of the unbelievers shall be razed and they shall be scorched to the earth. Their tags shall blink until the end of days.
from The Book of Mozilla, 12:10

Netscape 1.1 new features

- Tables
- Background images
- “Dynamic Documents”
  - Client Pull
  - **Server Push**
- This predates JavaScript by an entire year


An Exploration of Dynamic Documents
http://wp.netscape.com/assist/net_sites/pushpull.html
Google
ASSISTANCE
AN EXPLORATION OF DYNAMIC DOCUMENTS
INTRODUCTION
This document explores the "server push" and "client pull" dynamic document capabilities of Netscape Navigator 1.1 (Windows, Mac, and Unix versions). Please send comments to pushpull@netscape.com. Also, if you are using either server push or client pull in a real-world application, please drop us a note at pushpull@netscape.com and let us know -- thanks!
Examples are given at the end, along with an important note on implementing server push CGI programs as shell scripts.
THE GREAT IDEA
The general idea is that browsers have always been driven by user input. You click on a link or an icon or an image and some data comes to you. As soon as people saw they could do that, they wanted to give a server the ability to push new data down to the browser. (An obvious example is a stock trader who wants to see new quote data every 5 minutes.) Up until now, that hasn't been possible.
Netscape Navigator 1.1 gives content creators and server administrators two new open standards-based mechanisms for making this work. The mechanisms are similar in nature and effect, but complementary. They are:
1. Server push -- the server sends down a chunk of data; the browser display the data but leaves the connection open; whenever the server wants it sends more data and the browser displays it, leaving the connection open; at some later time the server sends down yet more data and the browser displays it; etc.
2. Client pull -- the server sends down a chunk of data, including a directive (in the HTTP response or the document header) that says "reload this data in 5 seconds", or "go load this other URL in 10 seconds". After the specified amount of time has elapsed, the client does what it was told -- either reloading the current data or getting new data.
In server push, a HTTP connection is held open for an indefinite period of time (until the server knows it is done sending data to the client and sends a terminator, or until the client interrupts the connection). In client pull, HTTP connections are never held open; rather, the client is told when to open a new connection, and what data to fetch (usually the same data).
In server push, the magic is accomplished by using a variant of the MIME message format "multipart/mixed", which lets a single message (or HTTP response) contain many data items. In client pull, the magic is accomplished by an HTTP response header (or equivalent HTML tag) that tells the client what to do after some specified time delay.
Server Push, Client Pull

“Client Pull”

<META HTTP-EQUIV="Refresh" CONTENT=1>

“Server Push”

Content-type: **multipart/x-mixed-replace**;boundary=XXooXX

--XXooXX
Content-type: text/plain

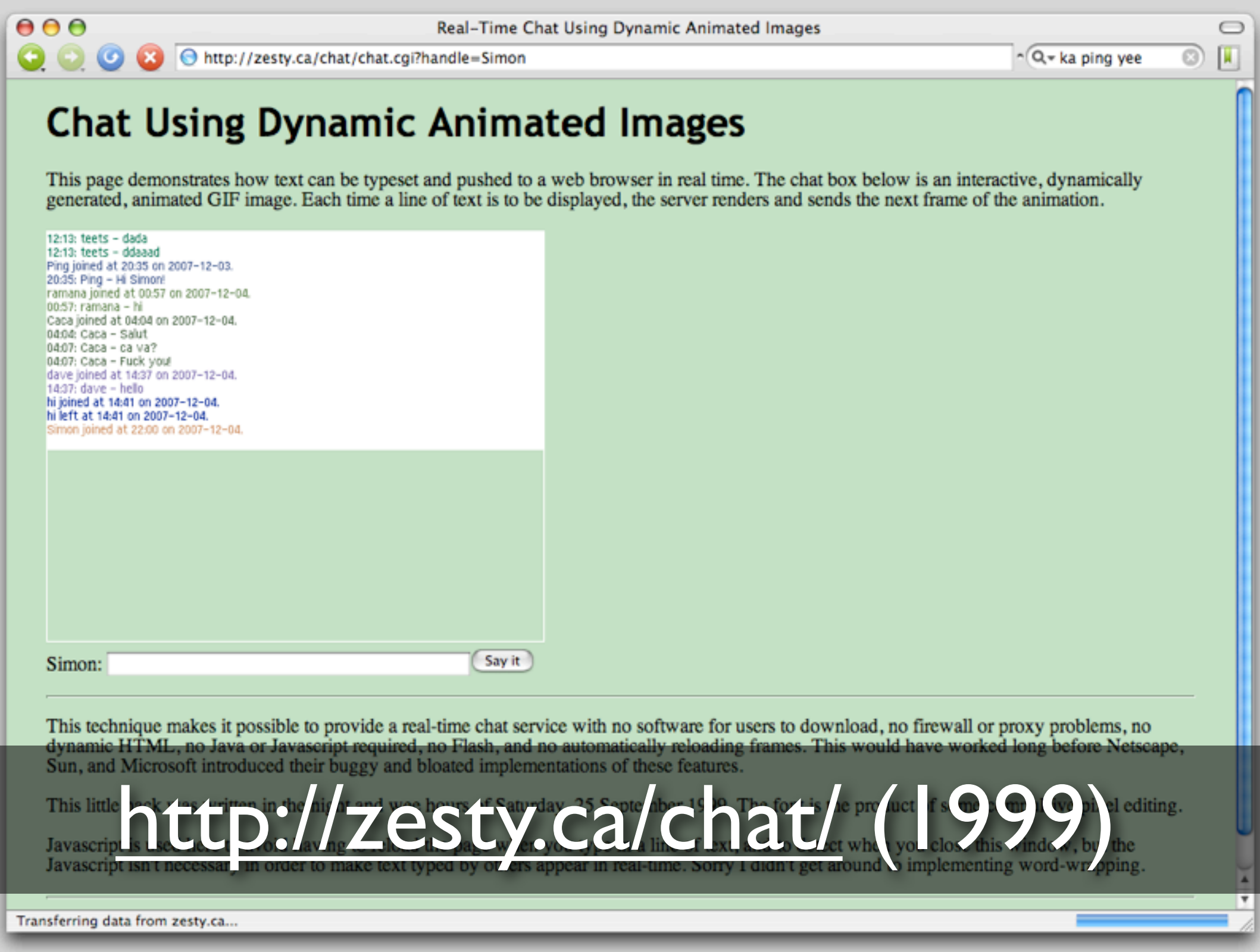
Data for the first object.

--XXooXX
Content-type: text/plain

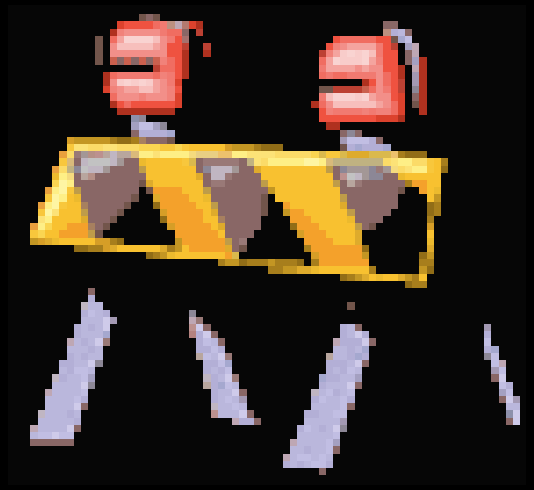
Data for the second and last object.

--XXooXX--

multipart/x-mixed-replace
still works today!


Real-Time Chat Using Dynamic Animated Images
http://zesty.ca/chat/chat.cgi?handle=Simon
ka ping yee
Chat Using Dynamic Animated Images
This page demonstrates how text can be typeset and pushed to a web browser in real time. The chat box below is an interactive, dynamically generated, animated GIF image. Each time a line of text is to be displayed, the server renders and sends the next frame of the animation.
12:13: teets - dada
12:13: teets - ddaaad
Ping joined at 20:35 on 2007-12-03.
20:35: Ping - Hi Simon!
ramana joined at 00:57 on 2007-12-04.
00:57: ramana - hi
Caca joined at 04:04 on 2007-12-04.
04:04: Caca - Salut
04:07: Caca - ca va?
04:07: Caca - Fuck you!
dave joined at 14:37 on 2007-12-04.
14:37: dave - hello
hi joined at 14:41 on 2007-12-04.
hi left at 14:41 on 2007-12-04.
Simon joined at 22:00 on 2007-12-04.
Simon:
Say it
This technique makes it possible to provide a real-time chat service with no software for users to download, no firewall or proxy problems, no dynamic HTML, no Java or Javascript required, no Flash, and no automatically reloading frames. This would have worked long before Netscape, Sun, and Microsoft introduced their buggy and bloated implementations of these features.
http://zesty.ca/chat/ (1999)
Transferring data from zesty.ca...

Modern Comet



WARNING

Excessive browser hacks
coming up

We need a solution that...

- Works through proxies
- Doesn't "click"
- Doesn't throb
- Let's us detect a lost connection
- Works in every browser

Three principle techniques

- Regular polling
- Streaming
  - XMLHttpRequest
  - Forever frame
- Long polling

Streaming XHR

- Good browsers let you open up a permanent XHR request
  - onreadystatechange is called periodically
  - If readyState == 3, data has arrived
- Need to split data on a delimiter and keep track of how many delimiters have been seen
- But... IE won't let you read responseText until the document has completely loaded!

Forever frame

- A hack that takes advantage of progressive rendering
- Send <script> blocks down the wire one at a time in to an iframe
  - Pad them with enough junk in between to clear the buffer in Safari
- Refresh the iframe occasionally to clean up

Killing the throbber

- Streaming techniques cause Firefox to continue to display the loading bar
- If you create an empty iframe and add and remove it from the DOM when an event is received, that problem goes away

```
if (load_kill_ifr === null) {
  load_kill_ifr = document.createElement('iframe');
  hide_iframe(load_kill_ifr);
}
document.body.appendChild(load_kill_ifr);
document.body.removeChild(load_kill_ifr);
```

Forever frame in IE

- Regular forever frame throbs, and causes “clicks” in IE when the iframe is refreshed...
- ... unless you embed the iframe in an ActiveXObject("htmlfile") (!)

```
var htmlfile = new ActiveXObject('htmlfile');
htmlfile.open();
htmlfile.write('<html><script>' +
               'document.domain="' + document.domain + '";' +
               '</script></html>');
htmlfile.parentWindow.Orbited = this; htmlfile.close();
var iframe_div = htmlfile.createElement('div');
htmlfile.body.appendChild(iframe_div);
iframe_div.innerHTML = '<iframe src="' + this.url + '"></iframe>';
```

htmlfile issues

- htmlfile gets upset if you try to perform too many DOM manipulations from inside it
  - This appears to relate to JavaScript garbage collection
- One solution is using a queue to shuttle messages between htmlfile and the real page
- Another is to use an empty setInterval() to move the DOM manipulations to another thread

Confuse IE with enough nested iframes and it forgets to click

Cross-domain Comet

- The 2 connection limit, combined with the need to run a specialist server stack, means it's extremely useful to be able to run your Comet server on a separate domain
- This requires even more hacks...

Double iframes

- To do streaming cross-domain in Safari and Firefox you need to use iframes to work around the same-domain policy
- This causes unwanted history entries
- If you nest two iframes you can navigate the inner iframe without affecting browser history

Long polling

- The most effective method for punching through proxies
- You make a request (via iframe or XHR), the server hangs until an event occurs, then it sends a response and your client instantly reconnects
- Can be scary for broadcast: 10,000 clients all re-connecting at the same time

Enough hacks yet?

- http://meteorserver.org/browser-techniques/
- http://cometdaily.com/2007/10/25/http-streaming-and-internet-explorer/
- http://cometdaily.com/2007/11/18/ie-activexhtmlfile-transport-part-ii/
- http://orbited.org/svn/orbit/trunk/daemon/orbited/static/orbited.js

HTML 5

http://www.whatwg.org/specs/web-apps/current-work/#event-source

```
<event-source src="/comet">

<script type="text/javascript">
document.getElementsByTagName("event-source")[0]
        .addEventListener("server-time", function(event) {
            alert(event.data);
        }, false);
</script>
```

```
Content-Type: application/x-dom-event-stream

Event: server-time
data: [time on the server]
```

Support added in Opera 9.5

“Wow, client-side
Comet sucks”...
that’s not the half of it

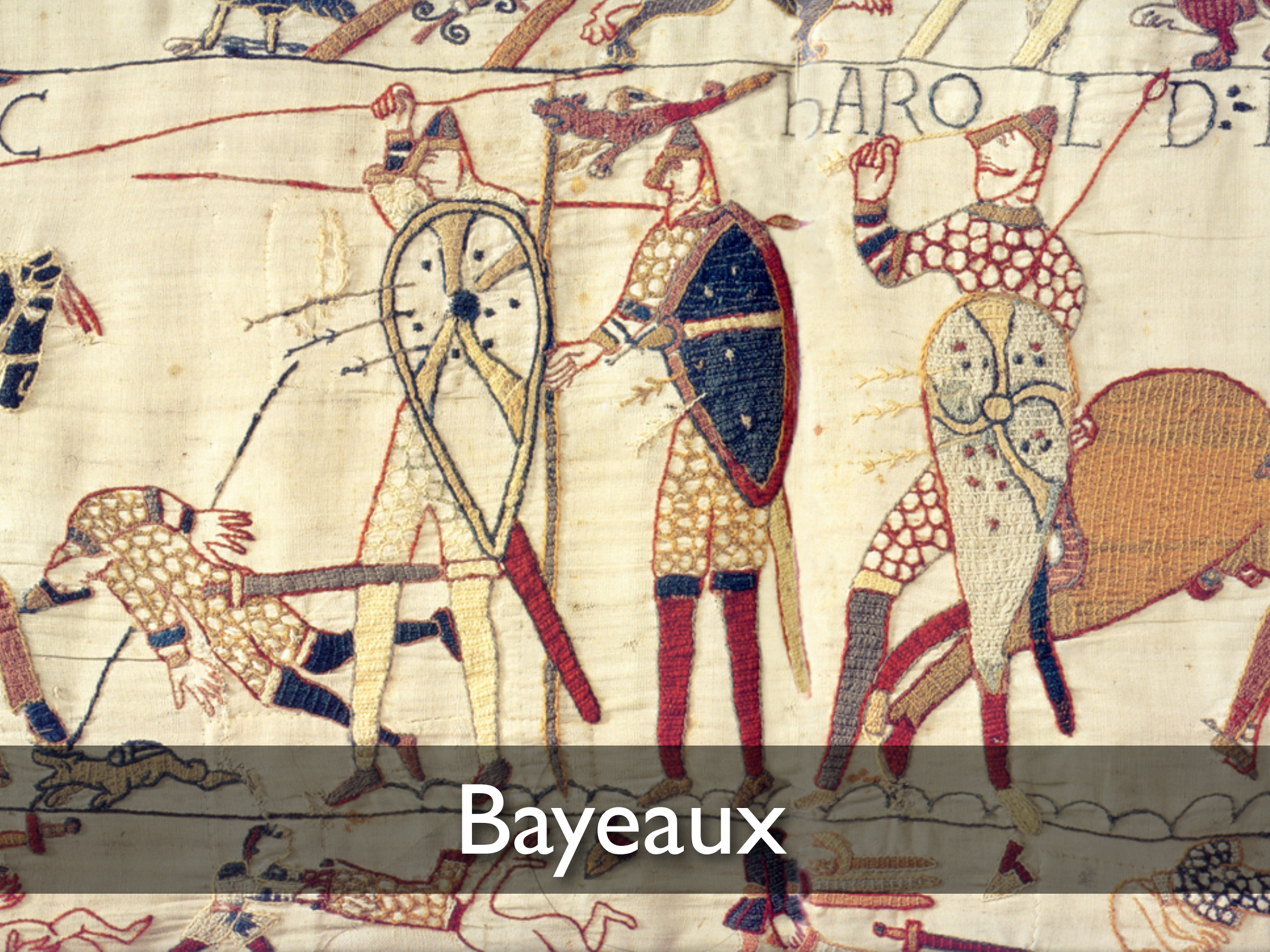
Scaling the server

http://flickr.com/photos/adrianblack/254968866/ craig1black

- Comet requires potentially tens of thousands of simultaneous HTTP connections
- Apache and other mainstream servers are designed to handle a single request as quickly as possible
  - Requests are generally served by a worker process or thread
- This absolutely will not scale to Comet

Event-based IO

- Rather than a thread or process per connection, have one process that loops through hundreds of connections checking if any of them are ready to send or receive data
- This approach scales, and can even be implemented in high level scripting languages (such as Perl, Python and Ruby)



Bayeaux

Bayeaux

- Bayeaux is a *protocol* for Comet
- Any Bayeaux client can talk to any Bayeaux server
- The protocol is based on clients publishing and subscribing to channels
- Data is encoded using JSON
- Clients can create a permanent connection using a handshake, or send simple one-off messages

Dumb Comet servers

- The principle advantage of Bayeaux is that your Comet server can be **stupid -** it doesn't need any application logic, it just needs to route messages around
  - It's a black box which you simply drop in to your architecture
- This means your application logic can stay on your favourite platform

Cometd

- Cometd (Comet Daemon) is an umbrella project for a number of Bayeaux implementations
  - cometd-python (in Twisted)
  - cometd-perl (also a Perlbal plugin)
  - cometd-java (on Jetty)
  - dojox.cometd (JavaScript client)
- http://cometd.com/

Still to solve...

- There's no clear standard method for implementing authentication against Bayeaux servers
- There's a need for a way to authorise specific clients to post messages on specific channels

Related Comet projects

- Meteor - Perl Comet server
  - http://meteorserver.org/
- Orbited - Python Event Daemon
  - http://orbited.org/
- Lightstreamer - commercial Comet server
  - http://www.lightstreamer.com/

So despite all of that...
Comet applications
are *easy* to build

How to build a Comet application (in 5 minutes)

Set up Jetty

- Download and unzip Jetty-6.1 from www.mortbay.org
- Install Maven from http://maven.apache.org/
- cd jetty-6.1.6/contrib/cometd/demo/
- mvn jetty:run
- Browse to http://localhost:8080/
- Mess around with the JS in src/main/webapp/examples/

dojox.cometd

```
dojo.addOnLoad(function() {
    dojox.cometd.init("http://127.0.0.1:8080/cometd");
    dojox.cometd.subscribe("/mychannel", function(comet) {
        alert(comet.data); // a JSON object
    });









});
```

dojox.cometd

```
dojo.addOnLoad(function() {
    dojox.cometd.init("http://127.0.0.1:8080/cometd");
    dojox.cometd.subscribe("/mychannel", function(comet) {
        alert(comet.data); // a JSON object
    });

    dojo.byId('send').onclick = function() {
        dojox.cometd.publish("/mychannel", {
            'msg': "A message sent by Comet"
        });
        return false;
    };
});
```

Here's my slideshow code

```
<script type="text/javascript">
dojo.require("dojox.cometd");
jQuery(function($) {
    dojox.cometd.init("http://127.0.0.1:8080/cometd");
    dojox.cometd.subscribe("/slideshow/change", function(comet) {
        $('#currentSlide').attr('src', comet.data.src);
    });
});
</script>
```

Conclusions

- Comet requires hacks, but they're reasonably well understood
- Bayeaux and Cometd abstract away the nastiness and make Comet accessible to sane developers
- If we survived CSS, Comet should be a cinch
- We're going to see a lot more sites built with Comet over the next year

Thank you