

831 PH Ex. 8

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re reexamination application of:

DOYLE et al.

Application No.: 90/006,831

Filed: October 30, 2003

For: DISTRIBUTED HYPERMEDIA
METHOD FOR AUTOMATICALLY
INVOKING EXTERNAL
APPLICATION PROVIDING
INTERACTION AND DISPLAY OF
EMBEDDED OBJECTS WITHIN A
HYPERMEDIA DOCUMENT

Examiner: Caldwell, A. T.

Art Unit: 2151

Interview Summary

OFFICE INTERVIEW OF APRIL 27, 2004

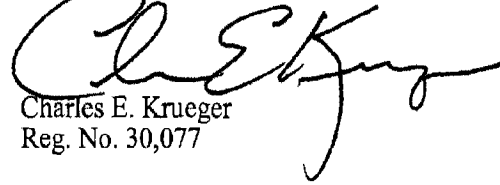
Attending the interview representing the assignee and exclusive licensee were Dr. Michael D. Doyle, one of the inventors, and Charles E. Krueger, the attorney of record, and representing the Patent Office were Examiners A. Caldwell and P. Laufer and Ms. Elizabeth Dougherty from the Office of Patent Legal Administration.

The subject matter discussed related to the rejection of claims 1 and 6 over the Applicants' Admitted Prior Art, Berners-Lee, and Raggett I and II. The issues were discussed in connection with a set of slides which are attached hereto. Further, pages from the Microsoft Computer Dictionary, Third Addition, were left with Examiners. These pages are also attached to this interview summary. Examiner Caldwell stated that he would not make a decision on the allowability of the claims discussed until he had received a written submission.

Charles E. Krueger delivered the original copy of the January 28, 2004, letter from Mr. Peter Wong, Group Director, Technology Center 2100, Computer Architecture, Software, and information Security, forwarding the following attachments: October 24, 2003, letter from the Law Firm of Pennie and Edmunds representing the WWW; October 14, 2003, letter signed by in-house counsel of America Online, Macromedia, and Microsoft; October 15, 2003, letter from Adobe Systems; October 22, 2003, letter from the law firm of Sidley Austin; and a binder of attachments. The purpose of delivering the original copy and attachments was to assure that they were included in the file of U.S. Patent No. 5,838,906.

THE OFFICE OF THE PATENT AND TRADEMARK OFFICE

Respectfully submitted,


Charles E. Krueger
Reg. No. 30,077

LAW OFFICE OF CHARLES E. KRUEGER
-P.O.Box 5607
Walnut Creek, CA 94596
Tel: (925) 944-3320 / Fax: (925) 944-3363

THE STATE OF CALIFORNIA

'906 Patent Reexamination

Interview with Examiner Andrew Caldwell
April 27, 2004

Michael D. Doyle, Ph.D.
Charles E. Krueger, Attorney

906 Patent Reexamination

• A decade ago, before ease of interactivity had become a key ingredient to the popular success of the Internet, the World Wide Web was in transition from laboratory to dormitory. Far from today's easy-to-use browser technology with seemingly ubiquitous interactivity, the World Wide Web then consisted of a large collection of static pages through which a user could navigate using a Web browser. As the technology progressed, still images were added to the Web collection; however the user was still only able to access the information, not interact with it. While early Web participants struggled to implement helper applications, researchers at the University of California were already examining the potential of the Web to become a platform for fully interactive embedded applications. The '906 invention was born.

'906 Patent Reexamination

- Claims 1 and 6

- Scope of the claim

- Executable application is automatically invoked, when an embed text format is parsed by the browser, in order to display the object and allow in-place interaction while the web page is being displayed

- Animation of claim 6

The References

- **Berners-Lee**
 - Provides a specification for the HTML document language
- **Raggett I and II**
 - Proposed use of a tag called EMBED for specification of static inline images
- **Mosaic**
 - Early web browser that supported helper applications

The Grounds of Rejection

- States that Raggett I's embed text format, type information, and automatic invocation are equivalent to 906 teachings
- States that external editors provide interactive control of embedded data
 - "These external editors that create or revise the embedded data would work the same way as the simple example of providing equation support"
- States that the claimed invention would have been obvious over Mosaic in view of Berners-Lee, Raggett I and Raggett II

Summary of References

Arguments

- Rendering application and external editor operate in different ways to perform different functions
- The external rendering application of Raggett I and II would cease execution as it returned a static image to the browser, prior to the image being displayed to the user
- The use of EMBED within the FIG tag requires that EMBED return a static and non-interactive image
- The rendered image of the source HTML+ document would not change if the end user modified the locally-downloaded copy of the embedded image
- The statement in Raggett I that a browser could be made to link to an external editor teaches away from the claimed element of automatically invoking an executable application in order to display the object and to enable in-place interaction

HTML+ Specification

Inline graphics

- "treated like characters"

- Therefore they are **static** pixmaps
- "Sophisticated HTML+ editors should allow authors to modify images using an external editor. Larger images should be specified with the FIG tag"
- Raggett teaches here that only the web page author would need to modify an inline graphic image.

As Berners-Lee teaches, it is the web page author who creates and publishes the page content for retrieval by the end-user. Only the **author** can change the source data.

HTML+ Specification

EMBED tag

- "for mathematical equations and simple drawings"
- "Images and complex drawings are better specified using the FIG or IMG elements."
- It should be noted that 906 technology is used by modern browsers for complex datatypes that browsers can't handle on their own
- Raggett teaches away from this use
- This is because use of EMBED for larger or more complex graphics would have a negative impact on page display speed – because the rendering application would have to **finish** computation **before** the page is displayed

Eolas v. Microsoft Summer of 2003

Raggett I and Raggett II were exhibits at trial, and Dave Raggett, himself, testified about them. Princeton Professor Edward Felten also testified, giving an expert opinion about the meaning of the Raggett documents.

Edward Felten testimony:

Q Now, does the work that Mr. Raggett did with the embed text have any relationship to what the embed text is used for in the '906 patent?

A No, it's an **entirely different** thing. If you are looking for similarities between them, it doesn't go much beyond having the text called "embed."

"...And so really what's happening here with HTML Plus is a slightly fancier way of putting **static** images into web pages. There's **no interactivity** here, and some of the other elements required in the '906 claims are also absent."

HTML+ Specification

EMBED tag -- Filter

- Raggett I and II's filter application renders data and then **returns** a pixmap
- Execution ends **before** browser uses returned data to render page
- Raggett I gives two examples which result in static images in the web page
- Filters are **non-interactive**
- Raggett I and II teach implementing the rendering filter applications through UNIX pipes

HTML+ Specification

EMBED tag -- Pipes

- UNIX pipes are treated as files by the calling program
- In this context, reading the data stream from a pipe is just like reading from a file stream
 - The src attribute specifies a **static** graphic file
 - The ability to substitute an EMBED tag for the src attribute in the FIG tag shows that, to the FIG tag code, EMBED would have behaved like a **static** graphic file

HTML+ Specification

FIG tag

- Teaches that you can use the EMBED element in place of the src attribute in order to define the image data

You can substitute the EMBED-defined pipe, for the src-defined file stream because, to the FIG tag code, they look the same

- FIG tag is clearly intended for use with **static data**

- Image maps are a feature of FIG

- They provide pre-defined active areas that can be associated with hypertext links

- A user's click on one of these active areas would cause the browser to fetch a new web document

- If the image data in an image map changes, the active areas lose their semantic correspondance, they lose their meaning

- Since Raggett teaches that EMBED should work with image maps, it **cannot** refer to a method for specifying dynamically-changing image data

HTML+ Specification

FIG tag

- A mouse click can **only** mean **one** thing at a time
- The image map feature of the FIG tag would have obviated any ability to interact with **EMBED**-based images beyond the simple clicking of an image map
- Any mouse clicks on an **EMBED**-based FIG-tag image would have been captured by the image map code of the FIG tag. The **EMBED**-based image, itself, would have to be **dead**.
- This means that the use of the proposed **EMBED** tag, itself, was appropriate only for the **non-interactive** display of image data.

HTML+ Specification

FIG tag

- If Raggett I had meant **EMBED** to support image data that can be dynamically changed and interactively controlled during the viewing of the Web page, this would have created a logical inconsistency that would have required discussion in the section of the FIG tag specification relating to image maps
- Since the reference was actually referring to **static** and **non-interactive** image data, no logical inconsistency existed, so no discussion was given

HTML+ Specification

EMBED tag -- Editor

- "Sophisticated browsers can link to external editors for creating or revising embedded data"
 - . In the context of Raggett 1, a browser that supported helper apps would be a sophisticated browser
 - . It is important to note that "Sophisticated" modifies browser, not the external editor
- "linked to"
 - . Means hyperlinked
 - . Therefore the editor is **not automatically invoked**
 - . Combination with Mosaic teaches that the helper application paradigm should be used
 - . External application would not be automatically invoked and would open in a **separate window**
 - . **No ongoing communications** between browser and external app

Eolas v. Microsoft Summer of 2003

Raggett testimony:

Q Let me direct your attention to the fourth line from the bottom where it says, "Sophisticated browsers can link to external editors for creating or revising embedded data." Do you see that, sir?

A Yes.

Q What does that mean?

A In the example of the mathematical equation, you might want to be able to **pop up** a kind of like an editor for mathematics which might have menus. So it's a simple thing. You might **pop up a separate window** with a pallet with different kinds of mathematical symbols.

HTML+ Specification

EMBED tag – Editor

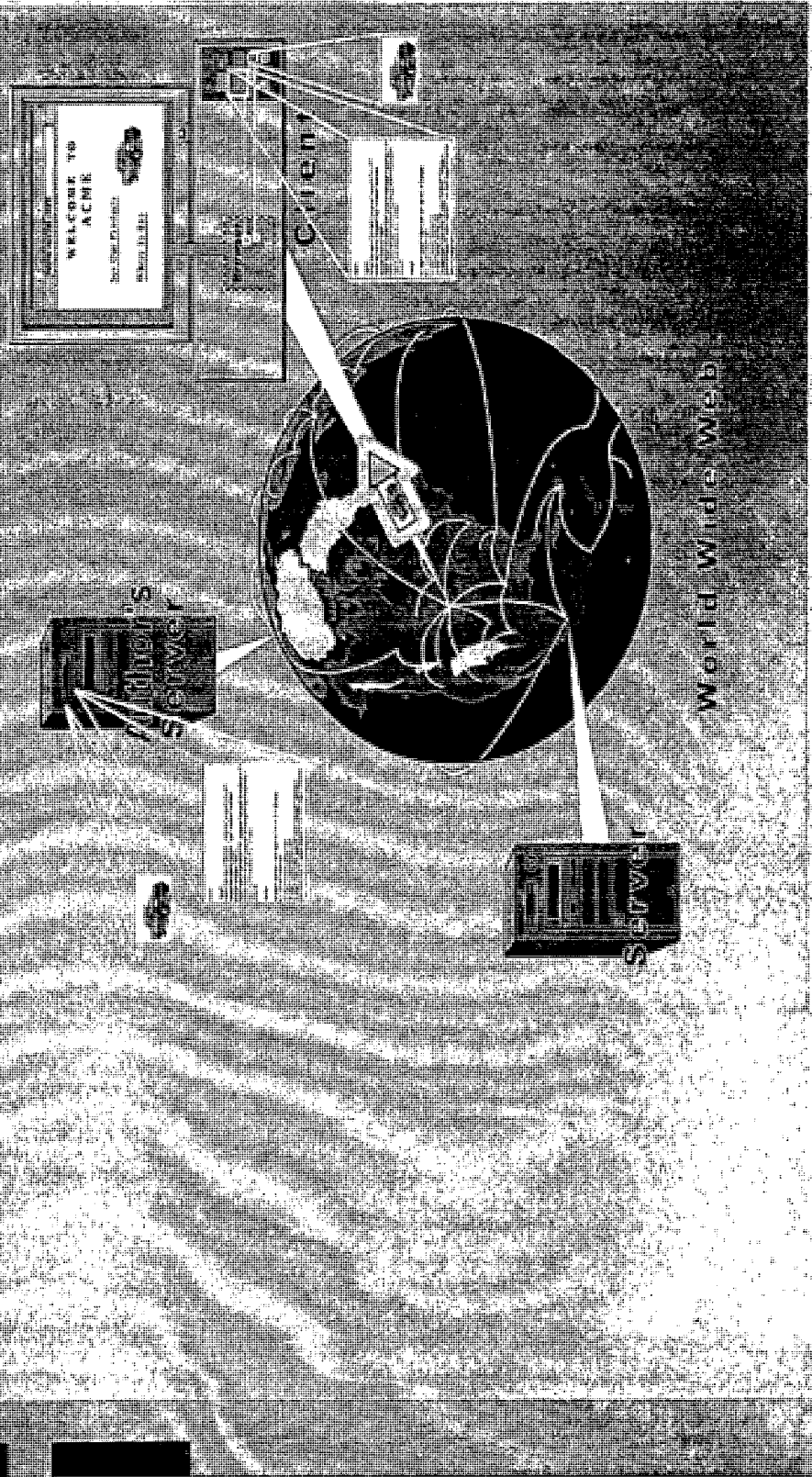
- "create or revise"

- Can only be done by the web page **author** prior to publishing the page on the author's server.
- The rendering application on the end-user's computer cannot be used to "create or revise"
 - The rendering application is not even active while the creating or revising is occurring

It would be self-defeating for remote browser to try to edit a locally-downloaded image

- Raggett I and II provide **no teaching** for how any editing program would work
- Since the user would only be editing the **local** temporary file in the cache
- The end user can't upload changes back to the server, **only** the web page **author** can do that
- The first time the page is refreshed or returned to the changes would be **overwritten. The page display would be unchanged, and would not reflect any changes made by the end-user.**

Editor vs. Client



HTML+ Specification

EMBED tag – Editor

- "Sophisticated" modifies the term "browser", not "external editors"
- There is no teaching or suggestion in Raggett I or II of creating new editors or modifying existing editors
 - "It allows authors to continue to use familiar standards, such as TeX and eqn."
- No existing external editor at the time of the filing of the 906 patent was able to communicate with a browser to dynamically change the rendered view of a web page.

Summary of References

Arguments

Rendering application and external editor operate in different ways to perform different functions

The external rendering application of Raggett I and II would cease execution as it returned a static image to the browser, prior to the image being displayed to the user

The use of EMBED within the FIG tag requires that EMBED return a static and non-interactive image

The rendered image of the source HTML+ document would not change if the end user modified the locally-downloaded copy of the embedded image

The statement in Raggett I that a browser could be made to link to an external editor teaches away from the claimed element of automatically invoking an executable application in order to display the object and to enable in-place interaction

Real World Considerations

The early demonstrations of the '906 invention were **enthusiastically received** by the scientific visualization community, and Dr. Doyle was invited to present it, in 1993 and 1994, at many prestigious institutions and at several highly-regarded conferences.

The EMBED tag of Raggett I and II was **abandoned** by Raggett, after the www-talk group asked him to drop it, and it was never implemented by others

Authors continued to use the IMG and FIG tags of Berners-Lee to embed inline graphics and **never** adopted the proposed EMBED tag of Raggett I and II

Real World Considerations

• Raggett testimony in *Eolas v. Microsoft*,
2003

CROSS EXAMINATION

Q Netscape plug-ins had the ability to interact with an embedded program object in a web page, right, sir?

A That is correct.

REDIRECT EXAMINATION

Q And you envisioned that, didn't you?

A I can't say I did. ...

831 PH Ex. 9

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re reexamination application of:

DOYLE et al.

Application No.: 90/006,831

Filed: October 30, 2003

For: DISTRIBUTED HYPERMEDIA
METHOD FOR AUTOMATICALLY
INVOKING EXTERNAL APPLICATION
PROVIDING INTERACTION AND
DISPLAY OF EMBEDDED OBJECTS
WITHIN A HYPERMEDIA
DOCUMENT

Examiner: Caldwell, A. T.

Art Unit: 2157

Declaration Under 37 C.F.R. §132

DECLARATION OF MICHAEL D. DOYLE

I, Michael D. Doyle, declare as follows:

1. The following references to "906" mean the subject matter recited in claims 1 and 6 of U.S. Patent No. 5,838,906.

2. The earliest demonstrations of the 906 technology, given in late 1993 and early 1994, were very enthusiastically received by the scientific and technical community. These demonstrations included both private presentations to influential experts in the field, as well as public demonstrations to large technical and scientific audiences. The following are some examples.

3. Dr. Donald Lindberg

One of the earliest demonstrations we made was to Dr. Donald Lindberg, the Director of the National Library of Medicine (NLM) and the Director of the National Coordination Office for High Performance Computing and Communications (HPCC). The HPCC program was an important source of funding for many of the early Web innovators, including the original developers of the Mosaic browser, who worked in the software development group at the National Center for Supercomputing Applications, in Illinois. Therefore, Dr. Lindberg was keenly interested in the state of the art in Web technologies at the time, and in innovations that could push that state of the art forward.

Dr. Lindberg was visiting the University of California San Francisco (UCSF) in mid-November, 1993, for a conference regarding the Red Sage Project, which I was directing at the time. We pulled him aside during the conference to come down to my Center in order to view and interact with a demonstration of the 3-dimensional visualization of Visible Embryo Project data via our 906-enhanced web browser.

He was so enthusiastic as a result of this demonstration that he invited me to go to the NLM Lister Hill Center three weeks later in order to do a live demonstration of the system during a platform presentation at an NLM technical conference, in early December, 1993. I made that demonstration, and received an enthusiastic response from the audience.

4. SIGWEB

Shortly after the first Lindberg demo, also in mid-November 1993, my UCSF group presented an on-stage live demonstration of our 906-enhanced browser system to a large group of early Web developers at the second monthly meeting of SIGWEB, held at Xerox PARC on November 19, 1993. SIGWEB was a special interest group for the World Wide Web that was founded, at my direction, by several members of my UCSF staff. Prominent members included Xerox PARC, Sun Microsystems, O'Reilly Publishers, Pacific Bell, NASA Ames, SCO, Lockheed, Amdahl, Netcom, Silicon Graphics, as well as researchers from eight different University of California institutions. (Later SIGWEB conferences featured presentations by early Web developers such as Marc Andreessen, the creator of the Mosaic browser, and Tony Johnson, the creator of the Midas browser.)

Several individuals from Silicon Graphics Corporation (SGI) who were at that November '93 SIGWEB meeting saw the 906 demonstration, and became very enthusiastic about the innovative character of our technology. As a result, John Flynn, a key individual at SGI responsible for health care technology applications, invited us to do a demonstration of the 906 system for Silicon Graphics technical management, in mid-January, at the SGI corporate headquarters.

5. Silicon Graphics

Following up on Mr. Flynn's invitation, we gave an on-site demonstration of the 906 system at the SGI Corporate Briefing Center in mid-January 1994. That demonstration was videotaped by my staff. A portion of that demonstration videotape is included in the video transcript included in the file history.

6. MMVR II

I had submitted an abstract in late November 1993 for a conference called Medicine Meets Virtual Reality II, which was being sponsored by the University of California San Diego (UCSD) Medical School, the IEEE, and the ACM. Video clips of a 906 demonstration had been captured by my UCSF staff in December of '93. A videotape of a 906-demonstration was shown during my platform presentation at the MMVR conference, in late January 1994, in San Diego, California. A major topic of the conference was the possibility of using the Internet to provide "telepresence" for virtual reality style applications for a variety of specialized medial applications. Since the World Wide Web was beginning to become very well known, there was tremendous interest among the attendees concerning the World Wide Web and whether it might have any relevance to the virtual reality community. After making my platform presentation and showing the 906-demo videotape, I received immediate and enthusiastic responses from a variety of researchers present in the conference hall at the time.

7. AAAS

One of the conference chairs at the MMVR conference, the NLM's Dr. Michael Ackermann, was also to be chairing a session on scientific visualization the following month at the annual meeting of the AAAS in San Francisco, in February '94. He was so excited about my MMVR 906-demonstration video that he requested that I show the same videotape at the AAAS

conference. I did, and my presentation was very well received from the large audience present at that AAAS presentation.

9. University of Michigan

Another person who was present at the MMVR conference was Dr. Brian Athey, from the University of Michigan (UMich) Digital Libraries Project. He became quite excited about the 906 technology and invited me to travel a few weeks later to Ann Arbor to present a seminar at UMich on our new browser innovations and their applicability to the Visible Embryo project and other large-scale digital-library-like informatics projects. I gave that presentation to a very enthusiastic response from the audience of 10-20 digital library researchers.

10. University of Pennsylvania

Also as a result of the MMVR conference, I was invited to give a seminar at the University of Pennsylvania (UPenn). That invitation was extended by Dr. Jayram Udupa, the Director of the UPenn Medical Image Processing Group. I showed the MMVR 906-demo videotape to a group of approximately 20 medical visualization and informatics researchers. The audience was very enthusiastic about our 906 results. One of the attendees of that demonstration, Dr. Scott Baldwin, then proceeded to spend several months trying to recruit me to join the UPenn's faculty, as a part of their prestigious Wistar Institute.

Shortly after my group gave additional private 906 demonstrations at UCSF in late Summer of 1994, the 906 co-inventors decided to co-found Eolas Technologies Inc., and so I declined the various academic recruiting efforts I was receiving from institutions such as UPenn, and my team went on to launch Eolas.

11. Dr. Dobbs Journal

In August of 1995, Eolas secured the exclusive licensing rights to the 906 technology from the University of California. This allowed us to release shortly afterward a non-commercial-use version of our 906-enhanced Web browser, which we had recently named Eolas WebRouser. That announcement received quite a bit of coverage in the press, which led to my being invited, by Jonathan Erickson, the Editor-in-Chief of Dr. Dobbs Journal, a very well-known industry-leading technical publication with a worldwide readership, to submit an article about our innovative 906-based browser technology for inclusion in their upcoming issue on "Data Communications and Internet Development." The article that I co-authored with the 906 co-inventors was entitled "Proposing a Standard Web API." Mr. Erickson received the article with great enthusiasm, and Dr. Dobbs made it the cover story for the February 1996 issue, featuring on the magazine cover a screenshot of our WebRouser browser, with a molecular modeling application shown embedded in a Web page. This article is attached hereto as exhibit A.

12. I declare that all statements made herein of my knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001, and that such willful false statements may jeopardize the validity of the patent.

Dated: May 16, 2004


MICHAEL D. DOYLE

Dr. Dobb's

JOURNAL

SOFTWARE
TOOLS FOR THE
PROFESSIONAL
PROGRAMMER

Data Communications

AND

Internet Development

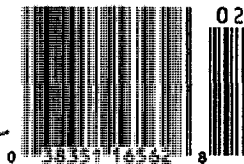
- JAVA COMMAND-LINE ARGUMENTS
- IMPROVING KERMIT PERFORMANCE
- DEBUGGING CGI APPS
- NETWORKING WITH WINSOCK 2.0

IMPLEMENTING MULTILEVEL
UNDO/REDO

NETWORKING INTELLIGENT DEVICES

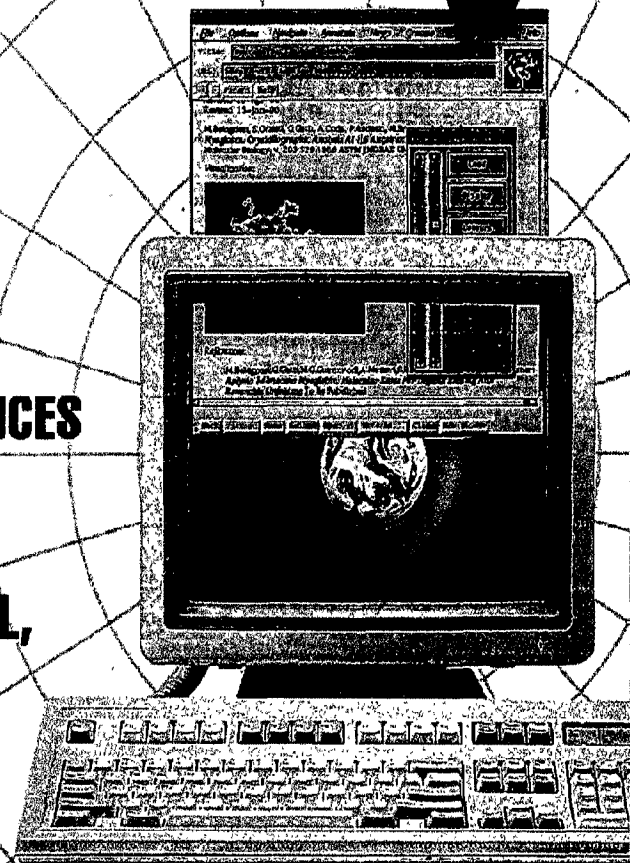
EXAMINING TOOLS.H++

\$3.95 (\$4.95 CANADA)



A Miller Freeman Publication

BOOKS ON HTML,
ALGORITHMS,
AND MORE



PROPOSING A STANDARD WEB API

by Michael Doyle, Cheong Ang, and David Martin

At last count, there were nearly a dozen APIs vying for hearts and home pages of Web developers. Our authors propose a standard API that leverages the concept of embedded executable content for interactive application development and delivery.

18

IMPROVING KERMIT PERFORMANCE

by Tim Kientzle

Tim compares the error-handling strategies of a variety of popular protocols, then presents heuristics that improve the performance of Kermit's windowing strategy.

28

CGI AND THE WORLD WIDE WEB

by G. Dinesh Dutt

The Common Gateway Interface (CGI) makes it possible for Web servers to interact with external programs. Dinesh presents a program that reports gateway-execution errors.

42

USING SERVER-SIDE INCLUDES

by Matt Kruse

Server-side includes are commands embedded inside HTML documents that enable your page to do something different each time it is loaded. Matt describes the format of these commands and shows how to write programs that work with your Web pages.

52

JAVA COMMAND-LINE ARGUMENTS

by Greg White

Greg introduces a package of Java classes that parse the command-line parameters for HtmlXlate, an application that converts HTML to RTF. Because HtmlXlate doesn't require display graphics, Greg made it an "application" instead of an "applet."

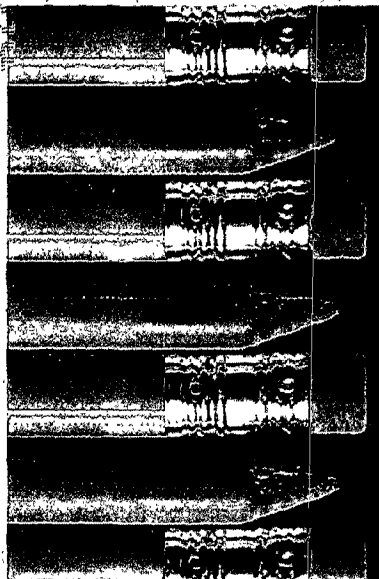
58

IMPLEMENTING MULTILEVEL UNDO/REDO

by Jim Beveridge

The Undo/Redo mechanism Jim presents here is based on a history length limited only by available memory. Because it is implemented in Visual C++ and MFC, this mechanism can easily be added to your applications

64



EMBEDDED SYSTEMS

NETWORKING INTELLIGENT DEVICES

68

by Gil Gamero

Novell's Embedded Systems Technology (NEST) lets you incorporate network protocols and client services into embedded systems. Gil uses NEST to put an intelligent coffee maker online, then controls it with a Windows-hosted menu program.

NETWORKED SYSTEMS

FAST NETWORKING WITH WINSOCK 2.0

76

by Derek Brown and Martin Hall

Derek and Martin show how you can get maximum performance from WinSock 2.0 applications by taking advantage of two features new to the spec—event objects and overlapped I/O.

EXAMINING ROOM

EXAMINING ROGUEWAVE'S TOOLS.H++ 80

by P.W. Scherer

RogueWave's Tools.h++, a C++ library consisting of more than 100 classes, has been the cornerstone of Perry's development efforts ever since he ported over 30,000 lines of C++ code to an equivalent app that was only 6000 lines long.

PROGRAMMER'S WORKBENCH

LEX AND YACC 86

by Ian E. Gorman

Ian describes how he used traditional compiler-development techniques and the MKS Lex & Yacc Toolkit to build a keyword-query compiler for a CD-ROM database.

COLUMNS

PROGRAMMING PARADIGMS 117

by Michael Swaine

Before continuing his examination of little languages for the Macintosh, Michael looks at a number of books devoted to HTML coding.

C PROGRAMMING 121

by Al Stevens

Al launches "Quincy 96," a C/C++ interpreter that runs under Windows 95 and is based on GNU C and C++. Among other features, Quincy 96 manages two kinds of documents—project documents and text source-code documents.

ALGORITHM ALLEY 135

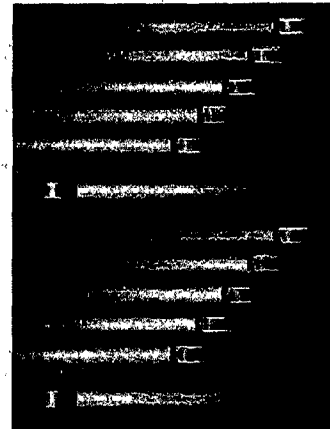
edited by Bruce Schneier

Binary searches are algorithmic staples that can be used in just about any program. Micha Hofri sees how efficient he can make a basic binary-search algorithm.

PROGRAMMER'S BOOKSHELF 139

by Dean Gablon

Dean compares *Practical Algorithms for C Programmers*, by Andrew Binstock and John Rex, and *Practical Algorithms in C++*, by Bryan Flaming.



FORUM

EDITORIAL 6

by Jonathan Fricben

LETTERS 10

by you

SWAINE'S FLAMES 144

by Michael Swaine

PROGRAMMER'S SERVICES

OF INTEREST 142

by Monica E. Berg

SOURCE CODE AVAILABILITY

As a service to our readers, all source code is available on a single disk and online. To order the disk, send \$14.95 (California residents add sales tax) to Dr. Dobb's Journal, 411 Borel Ave., San Mateo, CA 94402, call 415-655-4100 x5701, or use your credit card to order by fax, 415-358-9749. Specify issue number and disk format. Code is also available through the DDJ Forum on CompuServe (type GO DDJ), via anonymous FTP from site ftp.mv.com (192.80.84.3) in the /pub/ddj directory, on the World Wide Web at <http://www.ddj.com>, and through DDJ Online, a free service accessible via direct dial at 415-358-8857 (14.4 bps, 8-N-1).

NEXT MONTH

Little languages are a big deal to most programmers—and they're the focus of our March issue.

Proposing a Standard Web API

Short circuiting the API wars

Michael Doyle, Cheong Ang, and David Martin

The World Wide Web has matured from a relatively limited system for passive viewing of hypermedia-based documents into a robust framework for interactive application development and delivery. Much of this progress is due to the development of embedded executable content, also known as "inline Web applets," which allow Web pages to become full-blown, compound-document-based application environments. The first Web-based applets resulted from research begun in the late 1980s to find a low-cost way to provide widespread access for scientists and educators to remote, supercomputer-based visualization systems.

The Visible Human Project

In the late 1980s, the National Library of Medicine began a project to create a "standard" database of human anatomy. This "Visible Human Project" was to comprise over 30 GB of volume data on both male and female adult human anatomical structures. It was one of the original Grand Challenge projects in the federal High-Performance Computing and Communications initiative, the brainchild of then Senator Al Gore. As a member of the scientific advisory board for this project, one of us (Michael Doyle) became interested in the software issues involved in working with such a large database of the most detailed image information on human anatomical structure yet available. His group in the Biomedical Visualization Lab (BVL) at the University of Illinois at Chicago realized at the time that much research would have to be done to make such a vast resource both functional and accessible to scientists all around the world.

Until that time, medical visualization systems were designed to work on 3-D datasets in the 15-30 MB range, as produced by the typical CT or MRI scanner. High-end graphics workstations had adequate memory capacity and processor power to allow good interactive visualization and analysis of these routine

The authors are cofounders of Eolas Technologies Inc. and can be contacted at <http://www.eolas.com>.

datasets. The Visible Human data, however, presented an entirely different set of problems. To allow widespread access to an interactive visualization system based upon such a large body of data would require the combined computational power of several supercomputers, something not normally found in the typical biomedical scientist's lab budget.

Doyle's BVL group immediately began to work on solving the information-science problems related to both allowing interactive control of such data and distributing access to the system to scientists anywhere on the Internet. Our goal was to provide ubiquitous access to the system, allowing any user connected to the Internet to effectively use the system from inexpensive machines, regardless of platform or operating system.

The Promise of the Web

We saw Mosaic for the first time when Larry Smart, director of the National Center for Supercomputing Applications, demonstrated it at an NSF site visit at BVL in early 1993. We became immediately intrigued with the potential for Mosaic to act as the front end to the online visualization resource we had been designing. Immediately after Michael Doyle left the University of Illinois to take the position of director of the academic computing center at the University of California, San Francisco, we began enhancing Mosaic to integrate it with our system. We designed and implemented an API for embedded inline applets that allowed a Web page to act as a "container" document for a fully interactive remote-visualization application, allowing real-time volume rendering and analysis of huge collections of 3-D biomedical volume data, where most of the computation was performed by powerful remote visualization engines. Using our enhanced version of Mosaic, later dubbed "WebRouter," a scientist using a low-end workstation could exploit computational power far beyond anything that could be found in one location.

This work was shown to several groups in 1993, including many that were later involved in projects to add APIs and applets to Web browsers at places such as NCSA, Netscape, and Sun. Realizing our group's work enabled the transformation of the Web into a robust platform for the development and deployment of any type of interactive application, in 1994 the University of California filed a U.S. patent application covering embedded program objects in distributed hypermedia documents. Eolas Technologies was then founded by the inventors to promote widespread commercialization and further development of the technology.

Enhancing the Web

Once the concept of the Web as an environment for interactive applications was initiated, the question was how to further develop it. Toward the end of 1993, we discussed the relative merits of building an interpreted language, such as Basic or Tcl, directly into the browser, versus enhancing browsers through a "plug-in" API. We chose the API approach, believing that the best way to add language support would be by adding interpreters as external inline plug-in modules, which we called "Weblet applications". This would enable us to add a new language or other feature merely by developing a new Weblet application, without having to reengineer the browser itself.

Figure 1 is a Weblet-based version of the public domain RASMOI visualization program that lets users view, analyze, and visualize 3-D protein structure from within the Web page. A single programmer converted the original RASMOI source code into Weblet form in only ten hours.

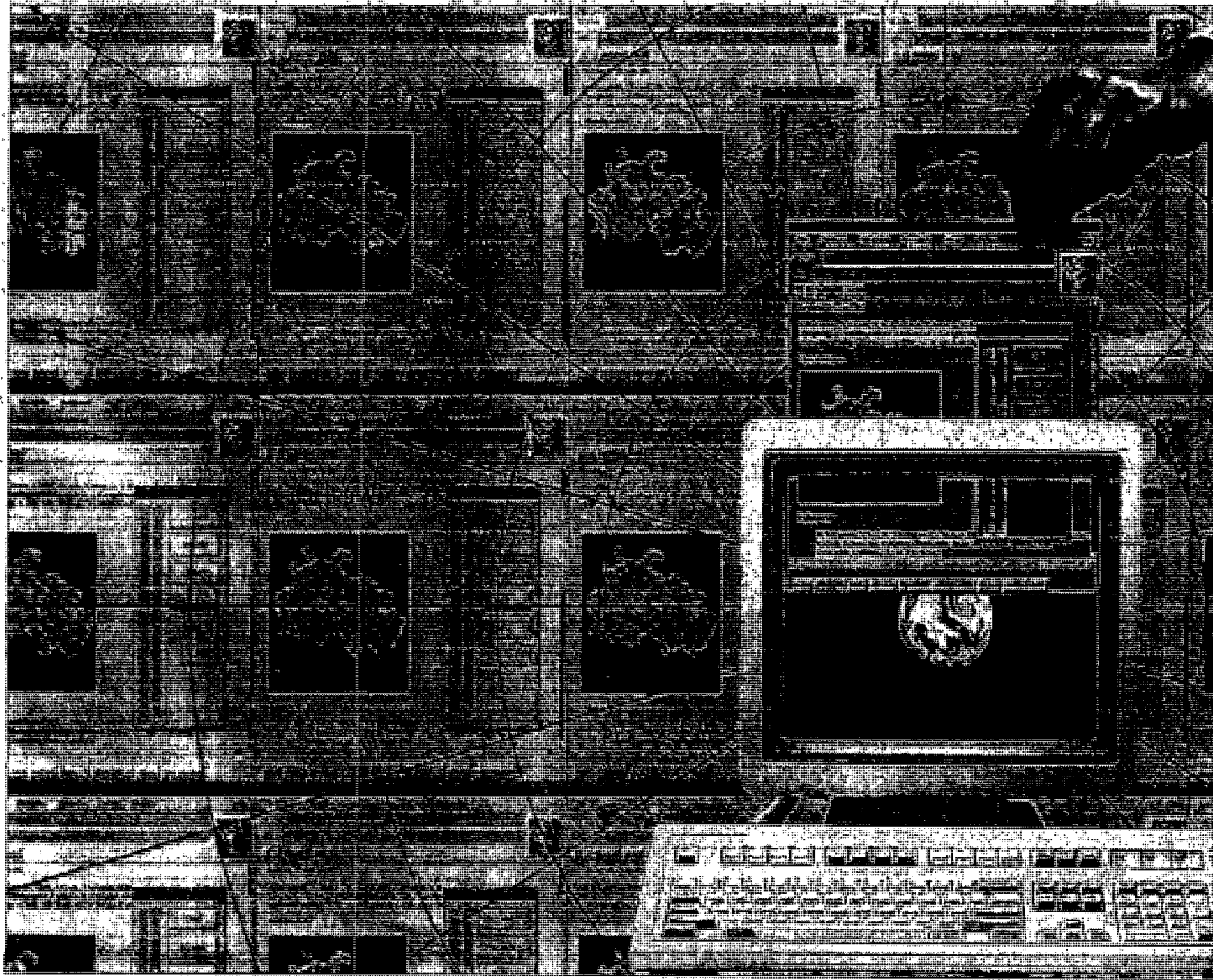
Some time later, both limited API support, such as MMS, CGI, and embedded language support, such as Java, began to appear in various Web browsers (the EMBED-> tag, which we first implemented in mid-1993, appeared in beta versions of Netscape's product by summer of 1995. Still, it wasn't until October of 1995 that the Netscape implementation began to approach the functionality of EMBED-> used in WebRouser. The enormous effect of these developments in accelerating the commercialization of the Internet industry prompted us to release the first (free, for noncommercial-use) distribution version of WebRouser for UNIX platforms in September 1995 (<http://www.csis.com/csias/Webrouser/>).

The WebRouser Approach

Our general philosophy with WebRouser was to allow enhancements of the browser's functionality through object-oriented modular application components that conform to a standard API, rather than turning the browser into a monolithic application with an ever-increasing code base. This encourages Web developers to take a document-centric approach to application development. The Web page itself becomes the mechanism for doing work through collections of small, efficient Weblet building blocks, rather than the manager of a top-heavy application found on the common desktop PC.

The first release of WebRouser also included other enhancements aimed primarily at improving the interactivity of Web pages. These included client-side image maps, a document-driven button bar, and document-driven modification of the browser's menu area.

Client-side image maps are supported through the Polymap format. Polymap files are essentially GIF files with polygon and line information stored in the image's comment fields. To prevent complex polygon information from bloating the file, all of the comment fields are compressed and decompressed using the GIF LZW algorithm. Polymap files require no special treatment of the HTML code. WebRouser automatically detects the presence of Polymap data when it reads inline GIFs. If a server-side (SMAP) image map points to a Polymap GIF then WebRouser will ignore the SMAP data and give the Polymap data priority. Hotspots are decoded in real time and highlighted as the mouse moves over the image, and the associated URL is displayed at the bottom of the screen, providing users the same style of interactivity that hotwords have in HTML text. The Polymap for-



mat specification is open and freely available for use. You can find the spec at <http://www.eolas.com/papers/Papers/Polymap/>.

The <LINK...> and <GROUP...> tags allow Web pages to dynamically customize elements of the browser's GUI. The LINK tag allows the creation of a document-driven button bar implemented by placing tags in the document header, with the syntax <LINK ROLE="button label" HREF="http://...">. Several of these tags in sequence result in buttons below the URL window, similar to Navigator's What's New or What's Cool buttons, but they are dynamically defined by the page currently being viewed. Similarly, the GROUP tag allows the Web page to modify the browser's GUI; however, this tag differs by defining a hierarchical menu that reflects an entire tree of Web pages. In Example 1, a typical GROUP menu trigger, the text string "Click here to view the WebRouser slide show" appears as a conventional anchor on the Web page, but selecting it, brings up the "slide_1.html" and activates the GROUPS menu option on WebRouser's menu bar. Slide Show is the first menu option, with a submenu whose options are Slide 1, Slide 2, and Slide 3. This allows the user to easily navigate through, for example, the "year, issue, article" hierarchy of online magazines.

The Web API

Of course, the key feature of WebRouser is the implementation of the <EMBED...> tag, through which inline plug-in Weblet applications are supported in Web pages. X Window applications that conform to the Eolas distributed hypermedia object embedding (DHOE) protocol can run—inline and fully interactive—within Web pages in the WebRouser window. WebRouser also supports the NCSA common client interface (CCI), which allows the Weblet to "drive" the browser application. DHOE and CCI collectively make up the Eolas Web API (WAPI) as supported in WebRouser.

WAPI is minimalist, combining the functionality of DHOE and CCI to exploit both the efficiency of X-events for communication of interaction events and graphic data and the flexibility of socket-based messaging for browser remote control and HTML rendering of Weblet-generated data. We are currently working on a cross-platform API, in the form of an OpenGL-style com-

mon-function library. The current minimalist WAPI specification will allow us greater flexibility in creating a cross-platform API, while maintaining compatibility with Weblets developed under the UNIX WAPI specification.

Eolas' primary objective with respect to the pending Web-applet patent is to facilitate the adoption of a standard API for interactive, Web-based application development, and then to develop innovative Weblet-based applications for the growing Internet software market. For an example of such a Weblet application, see the accompanying text box entitled "WebWish: Our Wish is Your Command." We intend to short circuit the API wars brewing between the major Web-browser competitors. In addition to creating a universal standard API, we are also instituting a mechanism for ensuring continued evolution of the WAPI spec on a regular timetable. Royalty-free licenses for browser-side implementation of Web applets under the pending patent have been offered to the major browser companies, and are in various degrees of negotiation. The primary condition of these licenses is that each licensee must conform to the WAPI protocol, and no other applet-integration protocol. A consortium of Eolas licensees is being formed to set the continuing WAPI specification and update it at regular intervals. The widespread acceptance of the developing WAPI standard will allow application developers to concentrate on the functionality of their applets without worrying which Web browser their customers will use.

Creating a WebRouser Weblet

WebRouser communicates with Weblet applications through a set of messages called the DHOE protocol. DHOE messages are relatively short character strings, which allow convenient, efficient use of existing interprocess-communications mechanisms on various platforms. We have implemented DHOE systems on several X Window platforms, including IRIX, SunOS, Solaris, OS/F 1, Sequent, and Linux. Implementations for both Microsoft Windows and Macintosh are planned for release by the end of the first quarter of 1996.

Listing One (listings begin on page 91) is a skeleton program for Weblet-based applications that can work with WebRouser. The current DHOE protocol defines a set of messages that synchronize the states on the DHOE clients and DHOE servers. The first four messages are used by the server to set up the DHOE system at startup, refresh/resize the client, and terminate the system on exit. The rest of the messages are sent by the browser client to the data server. They include messages about the client drawing-area visibility, and mouse and keyboard events.

Programming with DHOE involves initializing DHOE by installing a message-handling function, registering the DHOE client with the DHOE server, and registering various callbacks with their corresponding messages. The DHOE client and server may, at any time after client/server registration, send messages to each other. The messages (see Table 1) are character strings, and may be followed by different types of data. DHOE also supports buffer sharing (that is, bitmaps and pixmaps) between DHOE clients and servers.

Adding the DHOE mechanism into an existing data handler creates a DHOE server. The DHOE library kit consists of protocol_lib.h (the declaration file) and protocol_lib.c (the implementation file). To follow the Xt programming conventions, the DHOE strings are #defined with their Xt equivalents (DHOEKeyUp

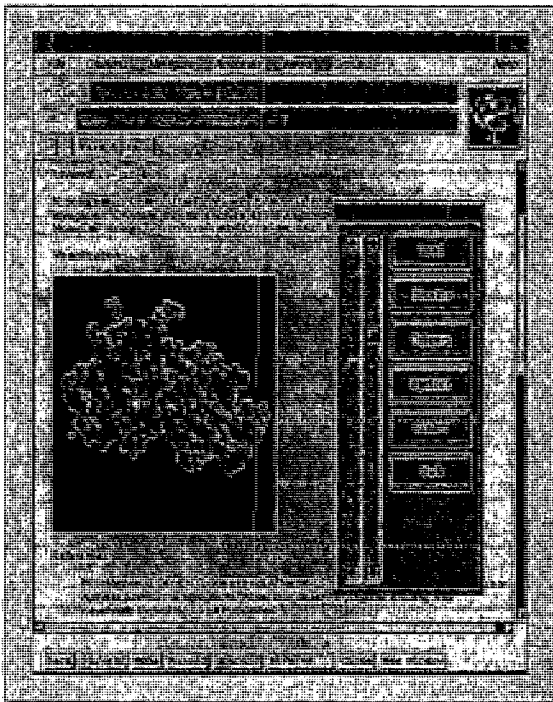


Figure 1: Typical Weblet application.

```
ALPHA: ROLE="Slide Show"  
LINK ROLE="Slide 1" HREF="slide_1.html"  
LINK ROLE="Slide 2" HREF="slide_2.html"  
LINK ROLE="Slide 3" HREF="slide_3.html"  
Click here to view the WebRouser slide show (2200)
```

Example 1: Typical GROUP menu.

(continued from page 20)
is mapped to *XtNkeyUp*, and so on). Messages from the DHOE server to the DHOE client (for example, external app→hypermedia browser) are:

- *XtNrefreshNotify*, server updating.
- *XtNpanelStartNotify*, server ready.
- *XtNpanelExitNotify*, server exiting.

Message	Description
DHOEServerUpdate	Tells a client to update data
DHOEServerReady	Tells a client the server is ready
DHOEServerExit	Tells a client the server is exiting
DHOEServerConfigureWin	Tells a client to resize/reposition the DHOE window
DHOEClientAreaShown	Tells the server the DHOE area is exposed
DHOEClientAreaHidden	Tells the server the DHOE area is being hidden
DHOEClientAreaDestroy	Tells the server the DHOE area is being destroyed
DHOEButtonDown	Sends mouse-pointer coordinates to the server on button down
DHOEButtonUp	Sends mouse-pointer coordinates to the server on button up
DHOEButtonMove	Sends mouse-pointer coordinates to the server on button move
DHOEKeyDown	Sends the corresponding keyname to the server on key down
DHOEKeyUp	Sends the corresponding keyname to the server on key up

Table 1: DHOE messages.


Messages from the DHOE client to the DHOE server (for example, hypermedia browser→external app) are:

- *XtNmapNotify*, DHOE area shown.
- *XtNunmapNotify*, DHOE area hidden.
- *XtNexitNotify*, DHOE area destroyed.
- *XtNbuttonDown*, DHOE area button down.
- *XtNbuttonUp*, DHOE area button up.
- *XtNbuttonMove*, DHOE area button move.
- *XtNkeyDown*, DHOE area key down.
- *XtNkeyUp*, DHOE area key up.

You can name these messages differently as long as the names are merely aliases of the original DHOE strings. These messages are defined in *protocol_lib.h*, which must be included in your program.

The following DHOE fundamental functions are provided in *protocol_lib.c*:

- *void handle_client_msg(Widget w, caddr_t client_data, XEvent *event)*, a function called back by *XtAddEventHandler* when it sees a message from the DHOE client (the hypermedia browser). To register this function with Xt, your program (DHOE server) should call *XtAddEventHandler(Widget app_shell, NoEventMask, True, handle_client_msg, 102)*. Here, *handle_client_msg* will be called with parameters *w=app_shell*, *client_data=102*, an *event* pointing to an X-event structure generated by Xt when it sees the message. The *app_shell* variable is usually the application shell returned by *XtInitialize*, *XtAppInitialize*, or *XtVaAppInitialize*.



neologic Now! <http://www.neologic.com/~neologic>
Download the NeoAccess Database Engine

neoAccess™

Cross-Platform Object Database Engine

Power Too Abundant to Meter

Powerful

NeoAccess is the most powerful object-oriented database engine available. It displays electrifying performance—up to ten times that of its competitors. Behind its elegant programming interface is a high performance query engine utilizing: extended binary trees and binary search algorithms tuned for short access times, dynamically combined, collapsed, and compressed indices, and object caching for nearly instantaneous access to previously used objects.

No Runtime Fees

Get the power of NeoAccess, and avoid the expense and administrative hassle of feeding the runtime fees meter. You pay one affordable price no matter how many copies of your application you sell or use.

Cross Platform

Others may promise cross-platform development tools—NeoLogic delivers. NeoAccess is a set of C++ classes designed for use with popular compilers and application frameworks on Windows®, Macintosh®, and Unix™ platforms. Full source code is included so it can even be used with custom frameworks.

Proven

Thousands of commercial and in-house developers have already found that NeoAccess enabled them to build fast, powerful applications in record time. That's why NeoAccess based applications are already operating on millions of computers. Tap into the power!

neo logic™

Powering Development of Object-Oriented Applications

NeoLogic Systems 1450 Fourth St., Suite 12 W. 510.524.5897
neologic@neologic.com Berkeley, CA 94710 f. 510.524.4501

CIRCLE NO. 239 ON READER SERVICE CARD

(continued from page 22)

- `void register_client(Widger w, Display *remote_display)`, which registers your program with the DHOE client.
- `void register_client_msg_callback(char *msg, void (*function_ptr)())`, which registers a function to be called back when Xt sees a string that matches `msg`. This function may appear anywhere in your program. You do not need to handle the `XtNmapNotify/XtunmapNotify` pair because DHOE servers deiconify/iconify when they receive these messages. You must specify a "quit" function to shut down your application gracefully on `XtNexitNotify`. Button- and key-message handling are optional. To obtain mouse coordinates, call `get_mouse(int *x, int *y)` for button-handling functions and `get_keysym(KeySym *keysym)` for key-handling functions. `Keysym` is defined by X11 (in `keysymdef.h`) for cross-platform compatibility.
- `void send_client_msg(char *msg, Display *remote_display, Window remote_window)`, which sends a message with the value `msg` to the DHOE client at a `display=remote_display` and has an X window ID of `remote_window`. The `remote_display` and `remote_window` must be provided. This function may appear anywhere in the program after `register_client`.

A Weblet CAD-File Viewer

WT is an applet that allows interactive rotation and zooming of a 3-D CAD file stored in NASA's neutral file format (NFF). The source code for the sample Weblet application is available electronically (see "Availability," page 3) and at <http://www.eolas.com/eolas/webrouse/wtsrc.tar.Z>. What follows is a brief walk-through of the weblet-enhancing sections of the code (illustrated in the code listing just mentioned as a "simplified sample program outline").

1. The outline starts with a `typedef` and some global declarations. The new type, `ApplicationData`, defines a structure common to all Xt Weblets. Together with the `myResources` and `myOptions` static variables, `myAppData` (which is of type `ApplicationData`) is used with `XtGetApplicationResources` in `main()` to extract the command-line arguments flagged with `win`, `pixmap`, `pixmap_width`, `pixmap_height`, and `datafile`. This is how Xt extracts command-line arguments and is unnecessary if the program has alternatives to decode command-line arguments. The aforementioned global variables and `XtGetApplicationResources` nicely store the information in a line such as `w1 -win 1234 -pixmap 5678 -pixmap_width 400 -pixmap_height 300 -datafile frame into myAppData`.

2. In `main()`, `app_shell` is first initialized the Xt way by using `XtInitialize`, which opens a connection to the X server and creates a top-level widget. `XtGetApplicationResources` gets the application resources as in step 1. The next section conveniently uses the `myAppData.win` variable to find out if the Weblet should run as a DHOE server or a stand-alone program. For a DHOE server, the program adds the `handle_client_msg` function from the DHOE implementation, `protocol_lib.c`, as the handler of the X client message event. The subsequent lines call three more DHOE functions: `register_client`, to initiate a handshake with the DHOE client; `register_client_msg_callback`, to register `myQuit()` as the callback function of the message `XtNexitNotify`; and `send_client_msg`, to send a `XtNpanelStartNotify` message, telling the DHOE client that the server is ready. The program then enters the conventional `XtMainLoop()`.

3. Two more functions must be modified. The drawing routine (`myDraw`) needs to copy the drawn picture (`myPixmap` in this case) onto `myAppData.pixmap`, the client's `pixmap`. The function then should send an `XtNrefreshNotify` message to the client, informing it of the change. The `myQuit()` function registered in

WebWish: Our Wish is Your Command

Sun's announcement of the adaptation of the Java language to the Web in 1995 was received enthusiastically by the entire Internet community as a welcome means for increasing the interactivity of Web-based content. Despite much of the publicity surrounding Java, which described it as an "interpreted" language, Java code must be compiled to a "virtual machine," which is then emulated on various platforms. A Web browser that supports the Java emulator is not enough to develop Java-based applications—the applet developer must purchase a compiler from Sun or its licensees at considerable cost.

Fully interpreted languages like Tcl/Tk or Basic are extremely useful, partly because they don't require a compiler for application development, just the language interpreter and any ASCII text editor. In choosing a programming language to adapt to the Web API, we decided early on that a fully interpreted programming language would be vital to quick, widespread Weblet implementation. We chose Tcl/Tk because of its robust capabilities and widespread use.

By the time you read this article, Eolas' WebWish Tcl/Tk interpreter should be available for both WebRouser under UNIX and Netscape Navigator 2.0 on Windows and Macintosh (see <http://www.eolas.com/eolas/webrouse/tcl.htm>). It supports Tcl 7.5 and Tk 4.1, as well as the Tcl DP and EXPECT extensions. A new security feature has been added that exploits PGP-style digital signatures in order to authenticate scripts from trusted sources and to prevent unwanted execution of scripts from untrusted sources. This Weblet application turns WebRouser and Navigator into complete application-development

environments, without the need for expensive compilers. All that is needed to develop a WebWish-based application is WebWish, a WAPI-compliant Web browser, and a good text editor. Developers can draw upon the vast existing resources of freely downloadable Tcl/Tk program source code, and the expertise of thousands of experienced programmers.

WebWish provides an easy-to-use, rapid prototyping environment, with built-in support for socket-based communications, remote procedure calls (RPCs), and the ability to "remote control" existing text-based server systems without reengineering the server. WebWish can run either as a Weblet in a Web page, or in stand-alone mode on either the client or a server machine. WebWish running in a Web page can communicate directly with other copies of WebWish running on remote servers, either through sockets or RPCs. This allows WebWish to act as "middleware" for the Web, allowing Web-based interfaces to create state-aware graphical front ends to existing text-based legacy systems, without changing the operation of the legacy server application.

Last November, Chicago's Rush Presbyterian St. Luke's Medical Center surgical department created both client and server WebWish applets for just such a purpose. The applets allowed physicians using WebRouser to interactively query and browse Rush's (Informix) SQL-based Surgical Information System, consisting of medical records on over 1.5 million patients. The entire project took one programmer exactly 12 hours from start to finish. Try that with Java!

—M.D., C.A., and D.M.

(continued from page 24)

`main()` needs to send an `XtPanelExitNotify` message to the client, telling the client that the server is terminated.

This Weblet can be tested by putting it in your path and pointing your copy of WebRouser to <http://www.eolas.com/eolas/webrouse/office.htm>.

The Eolas Web OS

In addition to the WebWish applet described in the text box, a Java interpreter Weblet application is planned for release by the end of March 1996. Java is a compiled language that produces binaries for a "virtual machine." The binaries are downloaded to the client and run on virtual-machine emulators that run on Macintosh, Windows, and UNIX platforms. Java applications tend to be smaller and more efficient than WebWish interpreted code, but they are far more difficult to develop. Eolas is developing a virtual operating system, the Web OS (planned for release late in 1996) that will allow far more robust, compact, and efficient compiled applets to be developed than is possible with Java. The Web OS is key to Eolas' long-term goal to transform the Web into a robust, document-centric, distributed-component application environment. It is a real-time, preemptive, multitasking, multithreaded, object-oriented operating system that will run efficiently on low-end platforms, even on 80286-based systems and handheld PDAs.

The Web OS can run within Windows, Macintosh, and UNIX environments, or in stand-alone mode on machines with no pre-installed operating system. It supports dynamic memory management and linked libraries, and is both graphical and object oriented at the OS level. The OS kernel includes fully defined object classes, inheritance, and direct messaging. The OS includes sev-

eral building-block objects that allow sophisticated applications—WYSIWYG word processors, spreadsheets, databases, e-mail systems, and the like—to be developed with a minimum of code. These applications are created primarily by subclassing and combining various Web OS component objects. Since new applications are created by defining differences and additions to the constituent objects, this results in tiny, robust, efficient binaries that optimize both bandwidth usage and server storage requirements. This platform is so efficient that a complete WYSIWYG word processor can be created in less than 5K of compiled code. Applications developed for the Web OS are likely to be smaller than most of the inline GIF images found on average Web pages today.

The operating system employs a single imaging model for screen, printer, fax, and other output devices; an installable file system, for both local and remote file access; direct TCP/IP and socket support; distributed objects; and security through public-key encryption and "ticket-based" authentication.

As the Internet pervades more of our work environments, the Web OS will allow the Web to become the preferred environment for new and innovative productivity, communications, and entertainment applications for all hardware platforms. The concept of a machine-specific operating system will become irrelevant, since any application will be available to the average user, regardless of hardware platform. Much of the computational load for applications will be pushed off to remotely networked computational engines, allowing low-cost Web terminals to act as ubiquitous doorways to potentially unlimited computational resources. The Web will be your operating system and the Internet will be your computer.

DDJ

(Listing begins on page 91.)

IF YOU KNOW OS/2 YOU SHOULD KNOW INDELIBLE BLUE

Post Road Mailer

by Innovel Systems

Make sending and receiving email easier with an editor, quoted replies, MIME attachments, and dropping, printing and sending just the words at your fingertips.
INO 001 PRM (Green, Internet Edition) MSRP \$79.00
INO 01 PRM Blue (PROFS) MSRP \$79.00

Kopy Kat

by Innovel Systems

Anything you can do on a PC with OS/2, you can do remotely by modem, serial cable or LAN. See the full desktop of the remote PC on your screen, control mouse and keyboard.
HGR48 KopyKat 2-user pack MSRP \$199.00
HGR44 KopyKat 10-user pack MSRP \$795.00

Your Single Source for OS/2 Solutions is also Your Single Source for Internet Solutions!

OS/2 WarpConnect IBM's Internet Connection Server
AntiVirus Hyperwise
DB2 World Wide Web Connection
Call, or check our web pages for all the info!

GammaTech Utilities

by Softouch Systems

Buy them before you need them! Recover deleted files, files from a corrupted volume or partition, optimize EFTS and FAT partitions, mass telephone edit sectors and much more.
GTU20 GammaTech Utilities MSRP \$149.00
Site Licensing Available

Corporate Accounts

Special Incentive Program

Special pricing on IBM software for your business! Call 1-800-776-8284 for details.
Visit our web site:
<http://www.indelible-blue.com/ib>

YOUR SINGLE SOURCE
FOR OS/2 SOLUTIONS

YOUR NEW 64 PAGE
FREE SERVICE CATALOG!

1-800-776-8284

CIRCLE NO. 229 ON READER SERVICE CARD

Listing One

```

#include "protocol_lib.h"

/* X may be to define resources and parse the cmdline args */
/* WebRouser 2.4-b2 gives the embedded window information through these args */
typedef struct {
    int win;
    int pixmap;
    int pixmap_width;
    int pixmap_height;
    char *datafile;
} ApplicationData, *ApplicationDataPtr;
static XResource myResources[] = {
    {"win", "Win", XtRInt, sizeof(int),
     XtOffset(ApplicationDataPtr, win), XtRImmediate, 0},
    {"pixmap", "Pixmap", XtRInt, sizeof(int),
     XtOffset(ApplicationDataPtr, pixmap), XtRImmediate, 0},
    {"pixmap_width", "Pixmap_width", XtRInt, sizeof(int),
     XtOffset(ApplicationDataPtr, pixmap_width), XtRImmediate, 400},
    {"pixmap_height", "Pixmap_height", XtRInt, sizeof(int),
     XtOffset(ApplicationDataPtr, pixmap_height), XtRImmediate, 400},
    {"datafile", "Datafile", XtRString, sizeof(char*),
     XtOffset(ApplicationDataPtr, datafile), XtRImmediate, NULL},
};
static XrmOptionDescr myOptions[] = {
    {"-win", "*win", XrmOptionSepArg, 0},
    {"-pixmap", "*pixmap", XrmOptionSepArg, 0},
    {"-pixmap_width", "*pixmap_width", XrmOptionSepArg, 0},
    {"-pixmap_height", "*pixmap_height", XrmOptionSepArg, 0},
    {"-datafile", "*datafile", XrmOptionSepArg, NULL},
};
ApplicationData myAppData;

void myDraw()
{
    /* do your drawing... */
    /* if you draw into your own drawables (myPixmap in this case) */
    if (myAppData.win) {
        /* copy from myPixmap to the "shared" pixmap */
        XCopyArea(display, myPixmap, myAppData.pixmap, myGC, 0, 0, WIN_WIDTH,
                 WIN_HEIGHT, 0, 0);
        /* tell WebRouser to update the drawing window */
        send_client_msg(XtRefreshNotify, display, myAppData.win);
    }
}

void myQuit()
{
    /* tell WebRouser you are exiting... */
    if (myAppData.win)
        send_client_msg(XtPanelExitNotify, display, myAppData.win);
    /* Motif way of exiting */
    XtCloseDisplay(XtDisplay(any widget));
    exit(1);
}

main()
{
    Widget app_shell;
    /* XtInitialize does XOpenDisplay, as well as creates a toplevel widget */
    app_shell = XtInitialize("w", "W", myOptions, XtNumber(myOptions),
                           target, argv);
    /* This time fill up myAppData with user specified values/default values */
    /* We get the embedded window's info this way */
    XtGetApplicationResources(&app_shell, myAppData, myResources,
                              XtNumber(myResources), NULL, 0);
    /* if we have an external window to display the image... */
    if (myAppData.win) {
        XtAddEventHandler(app_shell, NoEventMask, True, handle_client_msg, NULL);
        register_client(app_shell, display);
        /* register the func to be called when WebRouser exits */
        register_client_msg_callback(XtNextNotify, myQuit);
        /* tell WebRouser you have started fine */
        send_client_msg(XtPanelStartNotify, display, myAppData.win);
    }
    XtMainLoop(); /* Motif's event loop */
}
/* End of program listing */

```

KERMIT

Listing One

```

STATIC int KSendPacketFromCache(KERMIT_PRIVATE *pk, long sequence, int addToList)
{
    int slot = sequence & 63;
    long prev, next;

    if ((pk->exchange[slot].myPacket.type == 0)
        || (pk->exchange[slot].myPacket.data == NULL))

```

```

        return OK;
    prev = pk->exchange[slot].previousPacket; /* Unlink from list */
    next = pk->exchange[slot].nextPacket;
    if ((pk->lastPacket & 63) == slot) pk->lastPacket = prev;
    if (prev >= 0) pk->exchange[prev & 63].nextPacket = next;
    if (next >= 0) pk->exchange[next & 63].previousPacket = prev;
    pk->exchange[slot].nextPacket = -1;
    pk->exchange[slot].previousPacket = addToList ? pk->lastPacket : -1;
    if (addToList) {
        if (pk->lastPacket >= 0) /* Add to end of list */
            pk->exchange[pk->lastPacket & 63].nextPacket =
                pk->exchange[slot].sequence;
        pk->lastPacket = pk->exchange[slot].sequence;
    }
    pk->exchange[slot].tries++; /* Count number of sends */
    pk->exchange[slot].sendTime = SerialTime(pk->initTime);
    return StWarn (KSendPacket (pk, slot, pk->exchange[slot].myPacket.type,
                                /* Stamp time of send */
                                pk->exchange[slot].myPacket.data,
                                pk->exchange[slot].myPacket.length));
}
STATIC int KSendPacketReliable(KERMIT_PRIVATE *pk, BYTE type,
                                const BYTE *pSendData, unsigned long sendDataLength,
                                unsigned long rawDataLength)
{
    int blocked = FALSE;
    int err;
    int slot = pk->sequence & 63;
    int timeout = pk->my.timeout;
    [ /* Put packet into cache */
    EXCHANGE *pThisExchange = &(pk->exchange[slot]);
    if (pThisExchange->myPacket.data == NULL) {
        if (pk->minCache < pk->minUsed) {
            SwapSlots (pk->minCache, slot);
            /* Move free exchange to end of window */
            pk->minCache++;
            pThisExchange->myPacket.type = 0;
        } else return StWarn (kFail); /* Internal consistency failure */
    }
    if (pSendData == pk->spareExchange.myPacket.data) {
        /* In the reserved slot? */
        BYTE *pTmp = pThisExchange->myPacket.data; /* Just swap it in */
        pThisExchange->myPacket.data = pk->spareExchange.myPacket.data;
        pk->spareExchange.myPacket.data = pTmp;
    } else /* copy it */
        memcpy (pThisExchange->myPacket.data, pSendData, sendDataLength);
    if (pk->sequence > pk->maxUsed) pk->maxUsed = pk->sequence;
        /* Update end of window */
    pThisExchange->sequence = pk->sequence;
        /* Finish initializing this exchange */
    pThisExchange->myPacket.length = sendDataLength;
    pThisExchange->myPacket.type = type;
    pThisExchange->rawLength = rawDataLength;
    pThisExchange->tries = 0;
    pk->rxPacket.data = pk->spareExchange.myPacket.data;
    pk->txPacket.length = 0;
    ]
    StRet (KSendPacketFromCache (pk, pk->sequence, TRUE)); /* Send packet */
    if (pk->minUsed <= pk->minCache) blocked = 1; /* Are we blocked? */
    if (pk->maxUsed - pk->minUsed + 1 >= pk->currentWindowSize)
        /* How blocked are we? */
        blocked = (pk->maxUsed - pk->minUsed + 1) - pk->currentWindowSize + 1;
    err = KReceivePacketCache (pk, 0); /* Got a packet if one's ready */
    do { /* Until we're not blocked and there are no more packets pending */
        switch (err) {
            case kBadPacket: /* Didn't get a packet */
                break;
            case kTimeout:
                default: /* Unrecognized error, pass up to caller */
                    return StWarn (err);
            case OK: /* Got one! */
                EXCHANGE *pThisExchange = &(pk->exchange[pk->rxPacketSequence & 63]);
                switch (pk->rxPacket.type) {
                    case 'N': /* Got a NAK */
                        if (pThisExchange->myPacket.type != 0) /* Resend packet */
                            StRet (KSendPacketFromCache (pk, pk->rxPacketSequence, FALSE));
                        if ((pk->currentWindowSize > 1) || (pk->maxUsed > pk->minUsed))
                            break; /* Don't generate implicit ACKs for large windows */
                        pThisExchange = &(pk->exchange[(pk->rxPacketSequence - 1) & 63]);
                        pThisExchange->myPacket.type = 'Y';
                    case 'Y': /* Got an ACK */
                        if (pThisExchange->rawLength > 0) { /* ACK'd before? */
                            if (pThisExchange->tries == 1) { /* Update round-trip stats */
                                long now = SerialTime (pk->initTime);
                                long thisDelay = now - pThisExchange->sendTime;
                                if (pk->roundTripSamples == 0) { /* First sample? */
                                    pk->roundTripDelay = thisDelay;
                                    pk->roundTripDelayVariance = 0;
                                } else {
                                    long oldAverage = pk->roundTripDelay;
                                    long diffSquared;
                                    if (pk->roundTripSamples > 30) /* Average first 30 */
                                        pk->roundTripSamples = 30;
                                    /* Then decaying average */
                                    pk->roundTripDelay += (thisDelay - pk->roundTripDelay)
                                        / pk->roundTripSamples;
                                    diffSquared = (thisDelay - oldAverage) *
                                        (thisDelay - oldAverage);
                                    pk->roundTripDelayVariance += (diffSquared -
                                        pk->roundTripDelayVariance) /
                                        pk->roundTripSamples;
                                    pk->roundTripDelaySD = (pk->roundTripDelaySD +
                                        pk->roundTripDelayVariance /
                                        pk->roundTripSamples) / 2;

```

(continued on page 92)

831 PH Ex. 10

15
PL
5-13

Application of: Doyle et al.

Application Num.: 90/006,831

Filed: October 30, 2003

For: Distributed Hypermedia Method for Automatically Invoking External Application Providing Interaction and Display of Embedded Objects Within a Hypermedia Document

Examiner: Caldwell, A.T.

Art unit: 2157

Declaration of Edward W. Felten

I, Edward W. Felten, declare as follows:

1. I have been retained by Eolas and the Regents of the University of California to serve as an expert in the field of computer science and Internet software. My Curriculum Vitae, which recites my technical expertise, is attached hereto to as Exhibit A.

I. Qualifications

2. I graduated with Honors from the California Institute of Technology in 1985, with a B.S. degree in Physics. I received an M.S. in Computer Science in 1991, and a Ph.D. in Computer Science in 1993, both from the University of Washington.
3. I am currently a Professor of Computer Science at Princeton University, where I have taught since 1993. I was originally hired at Princeton as an Assistant Professor, in 1993. I was promoted to Associate Professor in 1999, and to Professor in 2003.
4. I am the author or co-author of numerous publications relating to computer science and Internet software. These publications are listed in my CV.
5. I have been asked to address the arguments presented in the Office Action mailed March 12, 2004 ("the Office Action") in connection with the reexamination of United States Patent No. 5,838,906 ("the '906 patent") that the claims of the '906 patent are unpatentable as being "obvious". For the reasons described in this declaration, I disagree with the arguments presented in the Office Action and, instead, believe that the claims of the

'906 patent fully meet the requirements for patentability over the cited references, as those patentability arguments have been described to me.

6. To familiarize myself with the issues involved in the rejection of the claims, I have reviewed numerous documents, including the following: the '906 Patent and its file history, the documents cited in the Office Action, and all other documents referenced or cited in this declaration.
7. Before specifically addressing the cited references and unpatentability arguments raised in the Office Action, I believe that it is important to discuss the relevant state of the browser art as it existed in 1994. My discussion is based on my experience as a computer science researcher and teacher, and as a Web user and network software developer. From this experience, I have gained an independent understanding of how the browser art developed.

II. Relevant State of the Art in 1994

8. In 1994, the Web was young, and browsers were a relatively new technology. Browsers offered only a very limited form of interactivity. A page could contain hyperlinks, on which the user could click to view another page. A page could be a form to be filled out by the user, with a "submit" button which, when clicked, caused the user to see another page.
9. Another technology, known as "helper applications," was implemented in the Mosaic browser. This technology allowed the browser to link to an external program, in cases where the browser encountered a file whose format the browser did not understand. For example, if the user clicked on a hyperlink that pointed to a file in .mpeg format (i.e., a movie in MPEG format), then the browser would launch an external MPEG-viewer program and pass the .mpeg file to that program. The result would be that the MPEG program ran, in a separate window from the browser.
10. Helper applications allowed the browser to link to an external program, but that program could not provide interactivity within the browser window. The helper application was just an external program that ran on the same computer, in a separate window.
11. None of these methods allowed a Web page author to place fully interactive objects within the confines of a Web page's display.
12. These methods are all implemented in today's browsers, and they are all in use on the Web today.

III. Response to the Unpatentability Arguments Raised in the Office Action

13. I have been told by patent counsel for Eolas and the Regents that a patent may not be obtained, even though the invention is not anticipated, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made, to a person having ordinary skill in the art to which the subject matter pertains. I have further been told that I need to make a four step inquiry to evaluate "obviousness" in which the scope and content of the prior art are to be determined; the level of ordinary skill in the pertinent art resolved; and against this background, the obviousness or nonobviousness of the subject matter is determined. I have also been told that such secondary considerations as commercial success, long felt but unresolved needs, failure of others etc. might be utilized to give light to the circumstances surrounding the origin of the subject matter sought to be patented.
14. As a "useful general rule" I have been told that references that "teach away" cannot serve to create a meritorious case of obviousness. Also, I have been told that proceeding contrary to the accepted wisdom is strong evidence of nonobviousness. In addition, I have been told that the prior art must "suggest" or "motivate" one of ordinary skill in the art to combine the prior art to make the claimed invention and must further have taught that such a combination would have a "reasonable expectation of success".

A. The Level of Skill In the Art

15. My benchmark for what ordinary skill in the art means is a person who is just graduating from a good computer science program at a college or a university – not a star student but just an average student – or a person who has gained an equivalent level of knowledge through experience in the industry. This person knows how to do things in conventional ways but does not exhibit an unusual level of innovative thinking.
16. In 1994, those of ordinary skill in the art were just becoming familiar with the Web and Web browsers. One of ordinary skill would have had a general idea of how the Mosaic browser worked, and would have been familiar with hyperlinks, forms, and helper applications.

B. The Grounds of Rejection

17. Claims 1 and 6 of the '906 patent have been rejected by the United States Patent Office as being obvious under 35 U.S.C. Sec. 103(a); as being unpatentable over the admitted prior art in the '906 patent and teaching of Berners-Lee, Raggett I, and Raggett II. While I understand that the patent attorneys for Eolas and the Regents are challenging whether Raggett I and Raggett II are really "prior art" to the '906 patent, I have been asked to

assume for the purposes of my analysis that both the Raggett I and Raggett II references would have been "prior art".

B. The '906 Patent

18. The claims of the '906 Patent describe a technology that allows web page authors to include, within the boundaries of a web page, interactive objects. This is done (briefly stated) by including in the web page's HTML text an embed text format, that provides information about where to get the object's data, along with information to identify and locate an executable application that will be invoked on the client computer to display the data and to provide interactivity with it, and by providing a web browser that knows how to parse the HTML to extract the embed text format, how to use type information to identify and locate the executable application, how to invoke the executable application, to execute on the client computer, and how to interface to the executable application so as to allow the user to interact with it within the boundaries of the browser window.

C. Prior Art Browsers

19. The Office Action cites the applicants' admitted prior art. I have reviewed all prior art references referenced in the '906 Patent's file history. It appears that the Office Action's discussion of this prior art focuses on the Mosaic browser, which was the most advanced prior art browser.
20. Mosaic, and other prior art browsers, executed on a client computer, and operated by downloading copies of web pages (and other files, such as embedded static images) over a network from web servers. After downloading a copy of a file, Mosaic would sometimes keep a copy of that file in a local cache, on the user's client computer. Caching allowed the file to be referenced more quickly if it was needed again later.
21. After downloading a file, Mosaic would parse that file (i.e., analyze its structure) to determine how the file should be displayed on the screen. Mosaic would then paint the contents of the file into a browser window.
22. When Mosaic, or another prior art browser, was used to view web pages, several steps stood between the author of the web page and the user who was viewing it. First, the file would be copied, at least once and perhaps more times, while in transit between the web server and the user's browser. Second, the file would be written in one format (typically, HTML) but displayed in another form, by rendering the HTML into a visual representation that would actually be presented to the user.
23. Because these steps stood between the author and the user, there was no realistic way for the user to edit the web page on the client workstation. The user did not have access to the version of the page that was distributed – that version lived on the server, and it wouldn't make sense to let an arbitrary user edit the contents of somebody else's web page.

24. In addition, because web pages were written in one format (HTML) and viewed in another (visual representation), it did not make sense to talk about editing and viewing a document in the same window. Web page authors would typically work with two separate windows open, one (a browser) to see what the visual representation looked like, and another (an external editor) to actually modify the page's HTML representation. An author would fiddle with the HTML, then click the save button in the editor and the refresh button in the browser to see what the visual representation of the page looked like, then fiddle with the HTML some more, and so on until he was satisfied with the page's appearance.

D. The Berners-Lee Reference

25. The Berners-Lee reference is a specification for the HTML markup language. HTML is a language used by Web page authors to describe the structure and desired contents of their pages. A browser parses an HTML document to determine its structure and then displays the visual representation of the specified items within a browser window.
26. The Berners-Lee reference teaches a model in which Web pages are written by an author, then distributed by a Web server to a browser, and viewed as a static item by the browser's user. The user views a page, and then clicks a hyperlink or a button, or enters some text, to select another page to view.
27. In the model taught by Berners-Lee, a user interacts with the Web by moving from one static page to another. Thus Berners-Lee teaches away from the provision of rich interactivity within a page.
28. Berners-Lee teaches a language for authoring web pages, but it does not teach how to build a browser or how a browser works.

D. The Raggett I Reference

29. Raggett I suggests some modifications to the HTML system taught in Berners-Lee. The overall teaching of Raggett I is very similar to that of Berners-Lee.
30. Like Berners-Lee, Raggett I does not teach how to build a browser or how a browser works.
31. Raggett I teaches the use of the same model as Berners-Lee, in which Web pages are essentially static, and the user interacts with the Web by moving from page to page. Accordingly, Raggett I teaches away from the provision of rich interactivity within a page.
32. Raggett I is motivated by the problems of Web page authors. Authors want to be able to include in their pages information in a wide variety of formats. For preexisting content, an author wants to be able to use the content in the format in which it was originally created. For new content, an author wants to be able to choose a format well suited to a particular type of content. For example, if the content consists of mathematical

equations, the author wants to be able to use a format designed for describing equations.

33. At the time of Raggett I, browsers such as Mosaic could handle only a limited set of data formats. Web page authors had noted a need for the display of static pages in more, and more varied, data formats.
34. One known method for displaying more formats was to do server-side translation. In this method, a web page author would take a document in some format, and generate a static image file from it. For example, an author might take a file describing a diagram, and generate from that file a static image, in GIF format, depicting the diagram. The web server could then deliver the GIF file to the browser, which would know how to render it within a web page.
35. Another known method to enable the display of more formats was to build support for displaying additional formats into the browser itself. Among the disadvantages of this approach were that it made the browser larger and more complicated, and that it required a new version of the entire browser to be distributed to a user before that user could view the new format.
36. Raggett I proposed a slight extension of this method, in which, rather than receiving an image, the browser receives information in some foreign format, and then uses an external program to render that information into an image, which the browser displays within the web page. This is a simple and natural extension of the browser's ability to display static images.
37. This extension is described in the following paragraph, which is also cited in the Office Action:

The EMBED tag provides a simple form of object level embedding. This is very convenient for mathematical equations and simple drawings. It allows authors to continue to use familiar standards, such as TeX and eqn. Images and complex drawings are better specified using the FIG or IMG elements. The type attribute specifies a registered MIME content type and is used by the browser to identify the appropriate shared library or external filter to use to render the embedded data, e.g. by returning a pixmap. It should be possible to add support for new formats without having to change the browser's code, e.g. through using a common calling mechanism and name binding scheme. Sophisticated browsers can link to external editors for creating or revising embedded data. Arbitrary 8-bit data is allowed, but `&`, `<` and `>` must be replaced by their SGML entity definitions. For example `<embed type="application/eqn">2 pi int sin (omega t) dt</embed>` gives [image of equation appears here].

(Raggett I at p. 6)

38. This paragraph teaches a method for displaying new types of *static* information within a Web page. The teaching of the use of static information is evident for several reasons.
39. First, the use of static information is consistent with the teaching of the remainder of Raggett I and with the teaching of Berners-Lee that preceded it.
40. Second, Raggett I motivates its proposed embed tag by referring to two types of data that one might want to display: “mathematical equations and simple drawings”. These are types of data that one would want to display statically.
41. Third, Raggett I says that Raggett’s proposed embed tag “allows authors to continue to use familiar standards, such as *TeX* and *eqn*.” (italics in original). These are well-known formats for describing the display of static data. TeX is used to specify the typesetting of textual documents; it is still widely used to format scientific publications. Eqn is used to specify the typesetting of mathematical equations. The TeX format is conventionally used with a program called “tex” or “latex” that produces as output a static document. The eqn format is conventionally used with a program called “eqn” that produces as output a static image or description of an equation. (For information on TeX, see Donald E. Knuth, *The TeXbook*, Addison-Wesley, 1986. For information on eqn, see Brian W. Kernighan and Lorinda L. Cherry, “A System for Typesetting Mathematics,” *Communications of the ACM* 18:3, March 1975; attached as Exhibit B.)
42. Fourth, Raggett I refers to the invocation of a “shared library or external filter to render the embedded data, e.g. by returning a pixmap”. This passage uses several terms of art (in the art of computer science) in ways that teach non-interactivity. “Filter” is a term of art that refers to a type of non-interactive program that translates data from one format to another. “Render” as used by Raggett I is a term of art that refers to the generation of a static image that is to be displayed. “Pixmap” as used by Raggett I is a term of art for a data structure describing an image. “Return” is a term of art that refers to the information produced by a program when that program terminates. A program that has returned something cannot do anything else; for example it cannot provide interactive processing. The use of these four terms of art further teaches the use of static images.
43. Fifth, the only specific example of the use of Raggett’s proposed embed tag that is given in Raggett I involves the use of a non-interactive filter which renders static data and then returns. The example depicts the use of the “eqn” program to translate the description of an equation into a static image.
44. Sixth, the discussion of the FIG and ISMAP features in Raggett I is inconsistent with the proposition that Raggett’s proposed embed tag

allowed interaction with an embedded object. In Raggett I, an instance of Raggett's proposed embed tag can be placed within a FIG element:

Instead of the *src* attribute, you can include an EMBED element immediately following the <fig> tag. This is useful for simple graphs, etc. defined in an external format.

(Raggett I at p. 12, emphasis in original) When the FIG element is used in conjunction with the ISMAP parameter (as described in the "Active areas" section of Raggett I, p. 13), the FIG element's display area becomes an image map: any mouse clicks made by the user within the visual depiction of the embedded data will be interpreted by the browser as pertaining to the image-map feature, and will therefore be intercepted by the browser and sent by the browser to the web server. This section of Raggett I teaches that the browser may intercept mouse clicks within the depiction of the embedded data, thereby contradicting the proposition that the embedded data itself can react to mouse clicks.

45. To my knowledge Raggett's proposed embed tag was never implemented. This is confirmed, for example, by Mr. Raggett's trial testimony:

Q. Sure. I'm sorry. I think you mentioned on direct exam that Mr. Martin's work and Mr. Ang's work and Dr. Doyle's work weren't part of the HTML Plus specification [i.e., of Raggett I].

A. Their work was not part of the specification.

Q. Okay. Now, you understand that they wrote to you in 1994 to describe their use of the embed tag and in fact suggested that you use their version of the embed tag in your upcoming HTML specification, correct?

A. They wrote to me saying that they'd obviously been looking at the HTML Plus specifications, and they were proposing something similar, and I responded to them that at that time there'd been a discussion in the summer of 1993, and at that time the consensus was that the group felt that there were higher priorities and so recommended that we drop the embed mechanism for that moment.

(Eolas v. Microsoft trial transcript at 1884:9-24; see Krueger declaration, Exhibit A)

46. However, if one of ordinary skill in the art (at the time) were asked to implement the Raggett I feature, he would do so by starting with the existing code for handling IMG tags, and modifying that code. The existing IMG code was able to paint static images into the body of a page, based on an input file that described the image. This code would be modified to invoke an external program, which would return a static image that would then be pasted into the web page in the same manner as in an IMG tag. Such an implementation would not support interactivity within a web browser window.

PH_001_0000785444

47. The sentence about “linking to external editors for creating or revising embedded data” refers to the use of external programs by a Web page’s author to edit or revise the external data before it is published on the author’s Web server.
48. There is nothing in Raggett I to suggest that the “external editors” would provide any display within a web browser window. The editors that were (and still are) conventionally used to create or revise data all run in their own windows; nothing in Raggett I suggests that they would be modified to run within a browser window, or that a browser would be modified to allow the editors to operate in that way. The reference to “linking” to an “external” program refers to the use of a hyperlink or button that the user can click to launch a separate program, as is done with helper applications. (Having the browser automatically invoke an editor wouldn’t make sense anyway, since only the page’s author would be in a position to edit a copy of the page that anybody else would see, and it wouldn’t make sense to invoke an editor automatically when ordinary users had no reason to want to invoke it.)
49. There is nothing in Raggett I that suggests how to provide an interactive program within a browser window – nothing about how to modify a browser to provide such a feature, and nothing about how to modify an editor to work with such a modified browser. No method for doing these things would have been obvious to one of ordinary skill.

D. The Raggett II Reference

50. Raggett II is a brief email message, written in response to requests for “equation support,” “eqn support,” and support for “embedded Postscript” in browsers. Equations, eqn data, and embedded postscript are all formats for specifying static data. The requesters ask for support for two rendering programs, eqn and ghostscript, both of which produce static images as output.
51. Raggett II responds by referring to the same functionality described in Raggett I.
52. Raggett II reiterates the teaching of Raggett I about the embedding of static images into Web pages. Raggett II refers to the use of external programs that “render[] foreign formats, e.g. as functions that take a sequence of bytes and return a pixmap.” Here again the term of art “render” is used, referring to the creation of a static image.
53. Additionally, the programs are said to “return a pixmap.” “Return” is a term of art that refers to the provision of information by a program when that program completes its execution. Therefore, once one of these programs has “return[ed] a pixmap”, the program is no longer running and cannot do anything more. In particular, the program cannot provide any interactive functionality, since the program would have stopped running

before the browser even painted the returned static pixmap onto the screen.

54. Raggett II mentions the possibility of implementing the external program as a DLL or dynamically linked library. A DLL is just another way of packaging an executable software application.
55. Raggett II teaches that the programs could be “driven via pipes and stdin/stdout”. This refers to a method by which one program invokes another, in such a way that the invoking program can provide input to the invoked program, and can receive any output produced by the invoked program. In this instance, the browser would invoke the external program, would provide the foreign data to the external program, and would receive the external program’s output, as a static image.

E. The Unpatentability Arguments in the Office Action Are Unpersuasive.

56. From my knowledge of the field, my own personal experience, and the state of the art in 1994, to the extent that a person of ordinary skill in the art was familiar with the teachings of the art cited in the Office Action, I find that the rejection of claims 1 and 6 as obvious is incorrect.
57. For example, the Office Action concludes, incorrectly, that Raggett I teaches interactive processing within a browser window. As described above, Raggett I teaches the use of static content within a browser window, coupled with the use of external editors that appear in separate windows.
58. The core of the Office Action’s argument on this point appears in this passage:

Although Raggett I describes an example where the browser calls a program for rendering an equation in ASCII character format into a pixmap image of the equation, Raggett I does also recognize that more sophisticated browsers can link to external editors for creating or revising embedded data. These external editors that create or revise the embedded data would work in the same way as the simple example of providing equation support. (See Raggett I: p. 6) However, the ability to create and revise the embedded data allows the user to interactively process the data within the browser window.

(Office Action at 5:42-6:5)

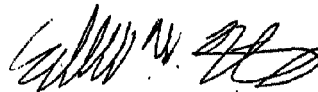
59. The Office Action is incorrect to say that an editor could work “in the same way” that the external rendering programs of Raggett I work. Raggett I’s external rendering programs operate by rendering external data to a static image, such as a pixmap, and then returning. Having produced a static image, and having returned (that is, having completed their work), they could not provide interactivity. A program that worked “in the same

way” could not provide editing functionality, or any other form of interactivity.

60. In any case, the editor programs available at the time were incapable of operating in the manner suggested by the Office Action. (They were, of course, capable of being invoked in a separate window.) Raggett I does not suggest the possibility of modifying any editor program. I am not aware of any such description in the prior art of how such modifications might be done; nor does the Office Action point to such a description.
61. If anything the Raggett I and II references teach away from the combinations recited in claims 1 and 6 of the '906 patent. These references teach the use of static web pages, with which the user interacts by moving from page to page, as opposed to the model of the '906 patent where a page can contain a fully interactive object. The two Raggett references teach the inclusion of static images, in various formats, into web pages, but they do not teach interactive processing within a browser window.
62. Finally, I have been told by the patent attorney for Eolas and the Regents that I should consider as part of my obviousness analysis “secondary considerations” such as copying, long felt but unresolved need, properties of the claimed invention, licenses showing industry acceptance of the invention and skepticism of skilled artisans before the invention.
63. I believe there is exceptionally strong “secondary consideration” evidence demonstrating non-obviousness in the case. This evidence includes the failure of others to duplicate the invention. I know of no evidence that either Mr. Raggett or anyone else tried to implement the purportedly obvious combination. In fact, I understand that the “HTML+” syntax described in Raggett I was never implemented.
64. For these reasons, I conclude that the rejection of claims 1 and 6 as being unpatentable is incorrect. The claims of the '906 patent would not have been obvious in view of the references cited in the Office Action.

I declare that all statements made herein of my knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both under 18 U.S.C. Section 1001, and that such willful false statements may jeopardize the validity of the patent.

Dated: May 7, 2004



Edward W. Felten



PH_001_0000785448

Edward W. Felten

Dept. of Computer Science
Princeton University
35 Olden Street
Princeton NJ 08540
(609) 258-5906
(609) 258-1771 fax
felten@cs.princeton.edu

Education

Ph.D. in Computer Science and Engineering, University of Washington, 1993.
Dissertation title: "Protocol Compilation: High-Performance Communication for Parallel Programs." Advisors: Edward D. Lazowska and John Zahorjan.
M.S. in Computer Science and Engineering, University of Washington, 1991.
B.S. in Physics, with Honors, California Institute of Technology, 1985.

Employment

Professor of Computer Science, Princeton University, 2003-present.
Associate Professor of Computer Science, Princeton University, 1999-2003.
Assistant Professor of Computer Science, Princeton University, 1993-99.
Senior Computing Analyst, Caltech Concurrent Computing Project, California Institute of Technology, 1986-1989.

Director, Secure Internet Programming Laboratory, Dept. of Computer Science, Princeton University, 1996-present.

U.S. Dept. of Justice, Antitrust Division: consulting and testimony in Microsoft antitrust case, 1998-2002.

Robins Kaplan, Miller & Ciresi. Consulting and testimony in patent lawsuit, 1998-2003.

Keker & Van Nest. Consulting in intellectual property / free speech lawsuit, 2002.

Electronic Frontier Foundation. Consulting in intellectual property / free speech lawsuits, 2001-present.

Certus Ltd.: consultant in product design and analysis, 2000-2002.

Cigital Inc.: Technical Advisory Board member, 2000-present.

Cloakware Ltd.: Technical Advisory Board member, 2000-present.

Propel.com: Technical Advisory Board member, 2000-2002.

NetCertainty.com: Technical Advisory Board member, 1999-2002.

FullComm LLC: Scientific Advisory Board member, 1999-2001.

Sun Microsystems: Java Security Advisory Board member, 1997-present.

Finjan Software: Technical Advisory Board member, 1997-2002.

International Creative Technologies: consultant in product design and analysis, 1997-98.
Bell Communications Research: consultant in computer security research, 1996-97.

Honors and Awards

Scientific America 50 Award, 2003.
Alfred P. Sloan Fellowship, 1997.
Emerson Electric, E. Lawrence Keyes Faculty Advancement Award, Princeton University School of Engineering, 1996.
NSF National Young Investigator award, 1994.
Outstanding Paper award, 1997 Symposium on Operating Systems Principles.
Best Paper award, 1995 ACM SIGMETRICS Conference.
AT&T Ph.D. Fellowship, 1991-93.
Mercury Seven Foundation Fellowship, 1991-93.

Research Interests

Computer security, especially relating to consumer products. Technology law and policy. Internet software. Operating systems. Interaction of security with programming languages and operating systems. Distributed computing. Parallel computing architecture and software.

Professional Service

Professional Societies and Advisory Groups

ACM Advisory Committee on Security and Privacy, 2002-2003.
DARPA Information Science and Technology (ISAT) advisory group, 2002-present.
Co-chair, ISAT study committee on "Reconciling Security with Privacy," 2001-2002.
National Academy study committee on Foundations of Computer Science, 2001-present.

Program Committees

USENIX General Conference, 2004.
Workshop on Foundations of Computer Security, 2003.
ACM Workshop on Digital Rights Management, 2001.
ACM Conference on Computer and Communications Security, 2001.
ACM Conference on Electronic Commerce, 2001.
Workshop on Security and Privacy in Digital Rights Management, 2001.
Internet Society Symposium on Network and Distributed System Security, 2001.
IEEE Symposium on Security and Privacy, 2000.
USENIX Technical Conference, 2000.
USENIX Windows Systems Conference, 2000.
Internet Society Symposium on Network and Distributed System Security, 2000.

IEEE Symposium on Security and Privacy, 1998.
ACM Conference on Computer and Communications Security, 1998.
USENIX Security Symposium, 1998.
USENIX Technical Conference, 1998.
Symposium on Operating Systems Design and Implementation, 1996.

Corporate Advisory Boards

Sun Microsystems, Java Security Advisory Council.
Cigital Inc.: Technical Advisory Board.
Cloakware Ltd.: Technical Advisory Board.
Propel.com: Technical Advisory Board.
Finjan Software: Technical Advisory Board.
Netcertainty: Technical Advisory Board.
FullComm LLC: Scientific Advisory Board.

University and Departmental Service

Faculty Advisory Committee on Policy, 2002-present.
Council of the Princeton University Community, 2002-present (Executive Committee)
Faculty Advisory Committee on Athletics, 1998-2000.
Computer Science Academic Advisor, B.S.E. program, class of 1998 (approx. 25 students)
Faculty-Student Committee on Discipline, 1996-98.
Faculty-Student Committee on Discipline, Subcommittee on Sexual Assault and Harrassment, 1996-98.

Students Advised

Ph.D. Advisees:

Minwen Ji (Ph.D. 2001). Dissertation: Data Distribution for Dynamic Web Content.
Researcher at Compaq Systems Research Center.
Dirk Balfanz (Ph.D. 2000). Dissertation: Access Control for Ad Hoc Collaboration.
Researcher at Xerox Palo Alto Research Center.
Dan S. Wallach (Ph.D. 1998). Dissertation: A New Approach to Mobile Code Security.
Assistant Professor of Computer Science, Rice University.
Robert A. Shillner (Ph.D. expected 2004). Tentative dissertation title: Improving Distributed File Systems using a Shared Logical Disk. Technical staff member at Google.
Michael Schneider (Ph.D. expected 2003). Dissertation topic: Network Defenses against Denial of Service Attacks.

Significant Advisory Role:

Drew Dean (Ph.D. 1998). Advisor: Andrew Appel. Researcher at SRI International.
Stefanos Damianakis, Ph.D. 1998. Advisor: Kai Li. President, Netrics, Inc.
Pei Cao, Ph.D. 1996. Advisor: Kai Li. Assistant Professor of Computer Sciences, University of Wisconsin. On leave at Cisco Systems.

Lujo Bauer, Ph.D. 2003. Advisor: Andrew Appel. Postdoctoral researcher at Carnegie-Mellon University.

Publications

Books and Book Chapters

- [1] Freedom to Tinker. Edward W. Felten. Publication expected, 2004.
- [2] Securing Java: Getting Down to Business with Mobile Code. Gary McGraw and Edward W. Felten. John Wiley and Sons, New York 1999.
- [3] Java Security: Web Browsers and Beyond. Drew Dean, Edward W. Felten, Dan S. Wallach, and Dirk Balfanz. In "Internet Besieged: Countering Cyberspace Scofflaws," Dorothy E. Denning and Peter J. Denning, eds. ACM Press, New York, 1997.
- [4] Java Security: Hostile Applets, Holes and Antidotes. Gary McGraw and Edward Felten. John Wiley and Sons, New York, 1996.
- [5] Dynamic Tree Searching. Steve W. Otto and Edward W. Felten. In "High Performance Computing", Gary W. Sabot, ed., Addison Wesley, 1995.

Journal Articles

- [6] Mechanisms for Secure Modular Programming in Java. Software – Practice and Experience, 33:461-480, 2003.
- [7] The Digital Millennium Copyright Act and its Legacy: A View from the Trenches. Illinois Journal of Law, Technology and Policy, Fall 2002.
- [8] DRM and Fair Use: A Skeptical View. Edward W. Felten. Communications of the ACM. April, 2003.
- [9] The Security Architecture Formerly Known as Stack Inspection: A Security Mechanism for Language-based Systems. Dan S. Wallach, Edward W. Felten, and Andrew W. Appel. ACM Transactions on Software Engineering and Methodology, 9:4, October 2000.
- [10] Statically Scanning Java Code: Finding Security Vulnerabilities. John Viega, Tom Mudosch, Gary McGraw, and Edward W. Felten. IEEE Software, 17(5), Sept./Oct. 2000.
- [11] Client-Server Computing on the SHRIMP Multicomputer. Stefanos N. Damianakis, Angelos Bilas, Cezary Dubnicki, and Edward W. Felten. IEEE Micro 17(1):8-18, February 1997.
- [12] Fast RPC on the SHRIMP Virtual Memory Mapped Network Interface. Angelos Bilas and Edward W. Felten. IEEE Transactions on Parallel and Distributed Computing, February 1997.

- [13] Implementation and Performance of Integrated Application-Controlled File Caching, Prefetching and Disk Scheduling. Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li. *ACM Transactions on Computer Systems*, Nov 1996.
- [14] Virtual Memory Mapped Network Interface Designs. Matthias A. Blumrich, Cezary Dubnicki, Edward W. Felten, Kai Li, and Malena Mesarina. *IEEE Micro*, 15(1):21-28, February 1995.

Symposium Articles

- [15] Receiver Anonymity via Incomparable Public Keys. Brent R. Waters and Edward W. Felten. *ACM Conference on Computer and Communications Security*. November 2003.
- [16] Attacking an Obfuscated Cipher by Injecting Faults. Matthias Jacob, Dan Boneh, and Edward W. Felten. *ACM Workshop on Digital Rights Management*, November 2002.
- [17] A General and Flexible Access-Control System for the Web. Lujo Bauer, Michael A. Schneider, and Edward W. Felten. 11th *USENIX Security Symposium*, August 2002.
- [18] Informed Consent in the Mozilla Browser: Implementing Value-Sensitive Design. Batya Friedman, Daniel C. Howe, and Edward W. Felten. *Hawaii International Conference on System Sciences*, January 2002. (Best Paper award, organizational systems track.)
- [19] Reading Between the Lines: Lessons from the SDMI Challenge. Scott A. Craver, John P. McGregor, Min Wu, Bede Liu, Adam Stubblefield, Ben Swartzlander, Dan S. Wallach, Drew Dean, and Edward W. Felten. *USENIX Security Symposium*, August 2001.
- [20] Cookies and Web Browser Design: Toward Realizing Informed Consent Online. Lynette I. Millett, Batya Friedman, and Edward W. Felten. *Proc. of CHI 2001 Conference on Human Factors in Computing Systems*, April 2001.
- [21] Timing Attacks on Web Privacy. Edward W. Felten and Michael A. Schneider. *Proc. of 7th ACM Conference on Computer and Communications Security*, Nov. 2000.
- [22] Archipelago: An Island-Based File System for Highly Available and Scalable Internet Services. *USENIX Windows Systems Symposium*, August 2000.
- [23] Proof-Carrying Authentication. Andrew W. Appel and Edward W. Felten. *Proc. of 6th ACM Conference on Computer and Communications Security*, Nov. 1999.
- [24] An Empirical Study of the SHRIMP System. Matthias A. Blumrich, Richard D. Alpert, Yuqun Chen, Douglas W. Clark, Stefanos, N. Damianakis, Cezary Dubnicki, Edward W. Felten, Liviu Iftode, Margaret Martonosi, Robert A. Shillner, and Kai Li. *Proc. of 25th International Symposium on Computer Architecture*, June 1998.

- [25] Performance Measurements for Multithreaded Programs. Minwen Ji, Edward W. Felten, and Kai Li. Proc. of 1998 SIGMETRICS Conference, June 1998.
- [26] Understanding Java Stack Inspection. Dan S. Wallach and Edward W. Felten. Proc. of 1998 IEEE Symposium on Security and Privacy, May 1998.
- [27] Extensible Security Architectures for Java. Dan S. Wallach, Dirk Balfanz, Drew Dean, and Edward W. Felten. Proc. of 16th ACM Symposium on Operating Systems Principles, Oct. 1997. Outstanding Paper Award.
- [28] Web Spoofing: An Internet Con Game. Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Proc. of 20th National Information Systems Security Conference, Oct. 1997.
- [29] Reducing Waiting Costs in User-Level Communication. Stefanos N. Damianakis, Yuqun Chen, and Edward W. Felten. Proc. of 11th Intl. Parallel Processing Symposium, April 1997.
- [30] Stream Sockets on SHRIMP. Stefanos N. Damianakis, Cezary Dubnicki, and Edward W. Felten. Proc. of 1st Intl. Workshop on Communication and Architectural Support for Network-Based Parallel Computing, February 1997. (Proceedings available as Lecture Notes in Computer Science #1199.)
- [31] Early Experience with Message-Passing on the SHRIMP Multicomputer. Richard D. Alpert, Angelos Bilas, Matthias A. Blumrich, Douglas W. Clark, Stefanos Damianakis, Cezary Dubnicki, Edward W. Felten, Liviu Iftode, and Kai Li. Proc. of 23rd Intl. Symposium on Computer Architecture, 1996.
- [32] A Trace-Driven Comparison of Algorithms for Parallel Prefetching and Caching. Tracy Kimbrel, Andrew Tomkins, R. Hugo Patterson, Brian N. Bershad, Pei Cao, Edward W. Felten, Garth A. Gibson, Anna R. Karlin, and Kai Li. Proc. of 1996 Symposium on Operating Systems Design and Implementation.
- [33] Java Security: From HotJava to Netscape and Beyond. Drew Dean, Edward W. Felten, and Dan S. Wallach. Proc. of 1996 IEEE Symposium on Security and Privacy.
- [34] Integrated Parallel Prefetching and Caching. Tracy Kimbrel, Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li. Proc. of 1996 SIGMETRICS Conference.
- [35] Software Support for Virtual Memory-Mapped Communication. Cezary Dubnicki, Liviu Iftode, Edward W. Felten, and Kai Li. Proc. of Intl. Parallel Processing Symposium, April 1996.
- [36] Protected, User-Level DMA for the SHRIMP Network Interface. Matthias A. Blumrich, Cezary Dubnicki, Edward W. Felten, and Kai Li. Proc. of 2nd Intl. Symposium on High-Performance Computer Architecture, Feb. 1996
- [37] Improving Release-Consistent Shared Virtual Memory using Automatic Update. Liviu Iftode, Cezary Dubnicki, Edward W. Felten, and Kai Li. Proc. of 2nd Intl. Symposium on High-Performance Computer Architecture, Feb. 1996

- [38] Synchronization for a Multi-Port Frame Buffer on a Mesh-Connected Multicomputer. Bin Wei, Gordon Stoll, Douglas W. Clark, Edward W. Felten, and Kai Li. Parallel Rendering Symposium, Oct. 1995.
- [39] A Study of Integrated Prefetching and Caching Strategies. Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li. Proc. of 1995 ACM SIGMETRICS Conference. Best Paper award.
- [40] Evaluating Multi-Port Frame Buffer Designs for a Mesh-Connected Multicomputer. Gordon Stoll, Bin Wei, Douglas W. Clark, Edward W. Felten, Kai Li, and Patrick Hanrahan. Proc. of 22nd Intl. Symposium on Computer Architecture.
- [41] Implementation and Performance of Application-Controlled File Caching. Pei Cao, Edward W. Felten, and Kai Li. Proc. of 1st Symposium on Operating Systems Design and Implementation, pages 165-178, November 1994.
- [42] Application-Controlled File Caching Policies. Pei Cao, Edward W. Felten, and Kai Li. Proc. of USENIX Summer 1994 Technical Conference, pages 171-182, 1994.
- [43] Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer. Matthias A. Blumrich, Kai Li, Richard D. Alpert, Cezary Dubnicki, Edward W. Felten, and Jonathan S. Sandberg. Proc. of Intl. Symposium on Computer Architecture, 1994.
- [44] Performance Issues in Non-Blocking Synchronization on Shared-Memory Multiprocessors. Juan Alemany and Edward W. Felten. Proceedings of Symposium on Principles of Distributed Computing, 1992.
- [45] Improving the Performance of Message-Passing Applications by Multithreading. Edward W. Felten and Dylan McNamee. Proceedings of Scalable High-Performance Computing Conference (SHPCC), 1992.
- [46] A Highly Parallel Chess Program. Edward W. Felten and Steve W. Otto. 1988 Conference on Fifth Generation Computer Systems.

Other Publications

- [47] Freedom to Tinker weblog, at <http://www.freedom-to-tinker.com>. Commentary on technology law and policy. Approximately 4000 readers per day.
- [48] Secure, Private Proofs of Location. Brent Waters and Edward W. Felten. Submitted for publication, January 2003.
- [49] An Efficient Heuristic for Defense Against Distributed Denial of Service Attacks using Route-Based Distributed Packet Filtering. Michael A. Schneider and Edward W. Felten. Submitted for publication, January 2003.
- [50] Written testimony to House Commerce Committee, Subcommittee on Courts, the Internet, and Intellectual Property, oversight hearing on "Piracy of Intellectual Property on Peer to Peer Networks." September 2002.

- [51] Written testimony to Senate Judiciary Committee hearings on "Competition, Innovation, and Public Policy in the Digital Age: Is the Marketplace Working to Protect Digital Creativity?" March 2002.
- [52] Informed Consent Online: A Conceptual Model and Design Principles. Batya Friedman, Edward W. Felten, and Lynette I. Millett. Technical Report 2000-12-2, Dept. of Computer Science and Engineering, University of Washington, Dec. 2000.
- [53] Mechanisms for Secure Modular Programming in Java. Lujo Bauer, Andrew W. Appel, and Edward W. Felten. Technical Report CS-TR-603-99, Department of Computer Science, Princeton University, July 1999.
- [54] A Java Filter. Dirk Balfanz and Edward W. Felten. Technical Report 567-97, Dept. of Computer Science, Princeton University, October 1997.
- [55] Inside RISKS: Webware Security. Edward W. Felten. Communications of the ACM, 40(4):130, 1997.
- [56] Simplifying Distributed File Systems Using a Shared Logical Disk. Robert A. Shillner and Edward W. Felten. Princeton University technical report TR-524-96.
- [57] Contention and Queueing in an Experimental Multicomputer: Analytical and Simulation-based Results. Wenjia Fang, Edward W. Felten, and Margaret Martonosi. Princeton University technical report TR-508-96.
- [58] Design and Implementation of NX Message Passing Using SHRIMP Virtual Memory Mapped Communication. Richard D. Alpert, Cezary Dubnicki, Edward W. Felten, and Kai Li. Princeton University technical report TR-507-96.
- [59] Protocol Compilation: High-Performance Communication for Parallel Programs. Edward W. Felten. Ph.D. dissertation, Dept. of Computer Science and Engineering, University of Washington, August 1993.
- [60] Building Counting Networks from Larger Balancers. Edward W. Felten, Anthony LaMarca, and Richard Ladner. Univ. of Washington technical report UW-CSE-93-04-09.
- [61] The Case for Application-Specific Communication Protocols. Edward W. Felten. Univ. of Washington technical report TR-92-03-11.
- [62] A Centralized Token-Based Algorithm for Distributed Mutual Exclusion. Edward W. Felten and Michael Rabinovich. Univ. of Washington technical report TR-92-02-02.
- [63] Issues in the Implementation of a Remote Memory Paging System. Edward W. Felten and John Zahorjan. Univ. of Washington technical report TR-91-03-09.

THE UNIVERSITY OF WASHINGTON

831 PH Ex. 11

15

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re reexamination application of:

DOYLE et al.

Application No.: 90/006,831

Filed: October 30, 2003

For: DISTRIBUTED HYPERMEDIA
METHOD FOR AUTOMATICALLY
INVOKING EXTERNAL
APPLICATION PROVIDING
INTERACTION AND DISPLAY OF
EMBEDDED OBJECTS WITHIN A
HYPERMEDIA DOCUMENT

Examiner: Caldwell, A. T.

Art Unit: 2151

Declaration Under 37 C.F.R. §132

DECLARATION OF CHARLES E. KRUEGER

I, Charles E. Krueger, declare as follows:

1. I am the attorney of record for the above-referenced reexamination proceeding.

2. In the action styled EOLAS TECHNOLOGIES, INC. and THE REGENTS OF THE UNIVERSITY OF CALIFORNIA, Plaintiffs, v. MICROSOFT CORPORATION, Defendant., Northern Dist. of Ill, No. 99 C 0626, Judge James B. Zagel presiding, Mr. David Raggett, the author of the Raggett I and Raggett II references cited by the examiner, testified as follows:

Question: Let me direct your attention to the fourth line from the bottom where it says, "Sophisticated browsers can link to external editors for creating or revising embedded data." Do you see that, sir?

Answer: (Mr. Raggett) Yes.

Question: What does that mean?

Answer: (Mr. Raggett) In the example of the mathematical equation, you might want to be able to pop up a kind of like an editor for mathematics which might have menus. So it's a simple thing. You might pop up a separate window with a pallet with different kinds of mathematical symbols.

Question: Could you use this technology with simple drawings?

Answer: (Mr. Raggett) Sure. Pallets of drawing tools.

3. In the same action Mr. Raggett further testified as follows:

Question: The Netscape plug-ins had the ability to interact with an embedded program object in a web page, right, sir?

Answer: (Mr. Raggett) That is correct.

MS. CONLIN: I have no further questions.

FURTHER RE-CROSS EXAMINATION BY MR. BAUMGARTNER:

Question: And you envisioned that, didn't you?

Answer: (Mr. Raggett) I can't say I did. I think Pei Wei's and other people's ideas contributed to these discussions about HTML Plus back in '93. And as you can see, we talked about dynamic linked libraries in that paragraph and other -- or shared objects or shared libraries for UNIX. So I think all the ideas were present before.

4. The entire transcript of Mr. Raggett's testimony in the above-mentioned action is attached as Exhibit A. Discussions between the attorneys and the Court have been deleted. The testimony quoted in paragraph 2 is located at page 1889-1890 of Exhibit A and the testimony quoted in paragraph 3 is located at page 1896 of Exhibit A.

5. In the 1996 paper entitled "Inserting objects into HTML", edited by David Raggett and including Tim Berners-Lee as an author, a new tag <OBJECT> is defined that allows an HTML author to specify the data to be inserted into HTML documents as well as the code that can be used to display/manipulate that data. It is stated that the <OBJECT> tag subsumes the role of the IMG tag.

This paper also states that developers have been experimenting with new ideas for dealing with new media. The EMBED tag proposed in Raggett I and Raggett II is not mentioned as one of the new ideas.

6. The 1996 paper entitled "Inserting objects into HTML", edited by David Raggett, is attached hereto as Exhibit B.

7. I declare that all statements made herein of my knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001, and that such willful false statements may jeopardize the validity of the patent

Dated: May 10, 2004


CHARLES E. KRUEGER

PH_001_0000785467

831 PH Ex. 12



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address COMMISSIONER FOR PATENTS—
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/006,831	10/30/2003	5838906		9718

30080 7590 08/16/2004
LAW OFFICE OF CHARLES E. KRUEGER
P.O. BOX 5607
WALNUT CREEK, CA 94596-1607

EXAMINER

Andrew Caldwell

ART UNIT PAPER NUMBER

2151

16

DATE MAILED: 08/16/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Vertical text on the left margin, possibly a barcode or reference number.



DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

AUG 16 2004

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. 90/006,831.

PATENT NO. 5838906.

PART UNIT 2151.

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(e)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(e)).

Office Action in Ex Parte Reexamination	Control No. 90/006,831	Patent Under Reexamination 5838906	
	Examiner Andrew Caldwell	Art Unit 2151	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

- a Responsive to the communication(s) filed on 11 May 2004. b This action is made FINAL.
c A statement under 37 CFR 1.530 has not been received from the patent owner.

A shortened statutory period for response to this action is set to expire 2 month(s) from the mailing date of this letter. Failure to respond within the period for response will result in termination of the proceeding and issuance of an *ex parte* reexamination certificate in accordance with this action. 37 CFR 1.550(d). **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).** If the period for response specified above is less than thirty (30) days, a response within the statutory minimum of thirty (30) days will be considered timely.

Part I THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:

1. Notice of References Cited by Examiner, PTO-892. 3. Interview Summary, PTO-474.
2. Information Disclosure Statement, PTO-1449. 4. _____.

Part II SUMMARY OF ACTION

- 1a. Claims 1-10 are subject to reexamination.
1b. Claims _____ are not subject to reexamination.
2. Claims _____ have been canceled in the present reexamination proceeding.
3. Claims _____ are patentable and/or confirmed.
4. Claims 1-10 are rejected.
5. Claims _____ are objected to.
6. The drawings, filed on _____ are acceptable.
7. The proposed drawing correction, filed on _____ has been (7a) approved (7b) disapproved.
8. Acknowledgment is made of the priority claim under 35 U.S.C. § 119(a)-(d) or (f).
a) All b) Some* c) None of the certified copies have
1 been received.
2 not been received.
3 been filed in Application No. _____.
4 been filed in reexamination Control No. _____.
5 been received by the International Bureau in PCT application No. _____.
* See the attached detailed Office action for a list of the certified copies not received.
9. Since the proceeding appears to be in condition for issuance of an *ex parte* reexamination certificate except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte* Quayle, 1935 C.D. 11, 453 O.G. 213.
10. Other: _____

cc: Requester (if third party requester)

Art Unit: 2137

1

2

Claim Rejections - 35 USC § 103

3

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

4

obviousness rejections set forth in this Office action:

5

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

6

7

8

9

10

11

This application currently names joint inventors. In considering patentability of

12

the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of

13

the various claims was commonly owned at the time any inventions covered therein

14

were made absent any evidence to the contrary.

15

16

The Prior Art as Applied to Claims 1-10:

17

18

Berners-Lee, T., et al., Hypertext Markup Language (HTML),
Internet Draft, IETF, pages 1-40, (June 1993).

19

20

21

Raggett, D., HTML+ (Hypertext Markup Language), (July 23, 1993).
Hereinafter referred to as "Raggett I."

22

23

24

25

Raggett, D., Posting of Dave Raggett, dsr@hplb.hpl.hp.com
towww-talk@nxocOl.cern.ch (WWW-TALK public mailing list),
(Posted June 14, 1993). Hereinafter referred to as "Raggett II."

26

27

28

29

Toye, G., et al., SHARE : A Methodology and Environment for
Collaborative Product Development, Proceedings, Second
Workshop on Enabling Technologies: Infrastructure for
Collaborative Enterprises, 1993, IEEE, pp. 33-47, April 22, 1993.

30

31

32

33

33

Art Unit: 2137

1 Claims 1-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over the
2 admitted prior art in the '906 patent and the teachings of Berners-Lee, Raggett I,
3 Raggett II, and Toye.

4

5 Regarding claim 1 of the '906 patent, the admitted prior art teaches a portion of
6 the claimed invention of claim 1 of the '906 patent, namely a method comprising:

7

8 "providing at least one client workstation" (See USP '906: Figure 2, element
9 130; Col. 4, Lines 32-40 which indicate that "small computer" 130 can be a
10 client) "and one network server" (See USP '906: Figure 2, element 132)

11

12 "coupled to a network environment" (See USP '906: Figure 2, element 100
13 Internet), "wherein the network environment is a distributed hypermedia
14 environment" (See USP '906: Col. 5 lines 24-25);

15

16 "executing, at the client workstation, a browser application" (See USP '906: Col.
17 3 lines 9-13), "that parses a first distributed hypermedia document to identify text
18 formats included in the distributed hypermedia document and for responding to
19 predetermined text formats to initiate processing specified by the text formats"
20 (See USP '906: Col. 1, lines 1-Col. 3, line 51, with particular emphasis on
21 Col. 2, line 63-Col. 3, line 25 showing a browser executing on client that
22 parses and then displays a hypermedia document; where the user clicks on
23 a link/image icon causing the browser to invoke a viewer application
24 displaying the image in a separate window); and

25

26 "utilizing the browser to display, on the client workstation, at least a portion of a
27 first hypermedia document received over the network from the server, wherein
28 the portion of the first hypermedia document is displayed within a first
29 browser-controlled window on the client workstation." (See USP '906: Figure 1,
30 element 10 as hypermedia document displayed on client; Col. 2 lines
31 28-36).

32

33 While the admitted prior art describes a method in which a hypermedia page
34 (See USP '906: Figure 1, element 10) is displayed in a browser (See USP '906: Col.
35 1, lines 1-Col. 3, line 51, particularly Col. 2, line 63-Col. 3, line 25), the admitted prior
36 art does not teach, as in claim 1 of the '906 patent, the particular steps used by the
37 browser in order to process and display the hypermedia page. To summarize, the
38 admitted prior art does not teach a method wherein the browser application parses a
39 first distributed hypermedia document to identify text formats included in the distributed
40 hypermedia document and for responding to predetermined text formats to initiate
processing specified by the text formats.

Art Unit: 2137

1
2 Nevertheless, Bemers-Lee teaches that HTML browsers parse HTML. (See
3 **Berners-Lee at p. 2 as printed - paragraph starting; "Implentations of ..."**) The
4 parsing is used to identify characters interpreted as markup elements, such as the
5 various tags (see **Berners-Lee at page 5**) in the structured text example, and to
6 associate text with various tags. These tags correspond to the claimed "text formats."
7 Bemers-Lee also teaches that the browser processes the HTML by rendering it into a
8 displayable form. (See **Berners-Lee at p. 3, definition of rendering**). Bemers-Lee
9 also discusses how specific markup elements are to be rendered. (See for example,
10 **Berners-Lee at p. 14, typical rendering of address tag; p.15 typical rendering of**
11 **block quote**). Bemers-Lee therefore teaches a method in which a browser application
12 parses a first distributed hypermedia document to identify text formats included in the
13 distributed hypermedia document and for responding to predetermined text formats to
14 initiate processing specified by the text formats.

15
16 It would have been obvious to a skilled artisan to combine (1) the teachings of
17 Bemers-Lee regarding the processing of HTML documents performed by a browser,
18 with (2) the HTML browser of the admitted prior art in light of the statement made by the
19 admitted prior art that its hypermedia system is designed to handle hypermedia
20 documents according the HTML markup standard. (See **USP '906: Col. 5, lines**
21 **28-31**).

22
23 Regarding the processing of the claimed "text formats," patentee acknowledges
24 that the prior art teaches a method wherein a browser invokes an external viewer
25 program to process various file formats not handled directly by the browser. (See **USP**
26 **'906: Col. 3, lines 13-20**). Specifically, the prior art describes an example wherein the
27 file format not handled by the browser is an image file in ".TIF" or ".GIF" format and the
28 browser invokes an image viewer program to display the full image in a separate
29 window. (See **USP '906: Col. 3 lines 13-20**). While the prior art teaches that certain
30 tags may cause the browser to invoke external applications to process particular file
31 formats, these applications do not display their data in the browser window. Therefore,
32 the admitted prior art does not teach the portion of the method of claim 1 of the '906
33 patent wherein:

34
35 " Said first distributed hypermedia document includes an embed text format,
36 located at a first location in said first distributed hypermedia document, that
37 specifies the location of at least a portion of an object external to the first
38 distributed hypermedia document;

39
40 Said object has type information associated with it utilized by said browser to
41 identify and locate an executable application external to the first distributed
42 hypermedia document, and
43

Art Unit: 2137

1 Said embed text format is parsed by said browser to automatically invoke said
2 executable application to execute on said client workstation in order to display
3 said object within a display area created at said first location within the portion of
4 said first distributed hypermedia document being displayed in said first
5 browser-controlled window."
6

7 However, Raggett I teaches various extensions to the HTML specification including
8 an EMBED tag that provides a simple form of object level embedding. (**See Raggett I:
9 p. 6 "Embedded data in an external format" and p. 26 embedded.**) For example,
10 Raggett I teaches an HTML document including an EMBED tag that identifies embedded
11 data in a foreign format. (**See Raggett I: p. 6 <embed ...> and <embed> tags.**) This
12 embedded data is an object that cannot be directly processed by the browser. The
13 foreign format data, or object, is embedded in the HTML document by placing it
14 between the <embed ...> and </embed> tags. (**See Raggett 1: p. 6 "2 pi int sin
15 (omega t)dt" as an example of embedded foreign data.**) Raggett I describes the
16 example of an embedded equation, where the browser calls a program for rendering an
17 equation by providing ascii character information to an external program and receives a
18 pixmap image of the equation from the external program that is then displayed in the
19 browser window. (**See Raggett I: p. 6, particularly the last ten lines.**) Raggett I
20 therefore teaches "a first distributed hypermedia document that includes an embed text:
21 format, located at a first location in said first distributed hypermedia document," that is
22 used to identify embedded foreign data. Raggett I also teaches that the embed tags
23 include a type attribute specifying a registered MIME content type that is used by the
24 browser to identify the appropriate external filter to use to render the embedded foreign
25 data. (**See Raggett I: p. 6 type="application/eqn".**) Raggett I thus teaches a method
26 wherein "the object has type information associated with it utilized by said browser to
27 identify and locate an executable application external to the first distributed hypermedia
28 document and wherein said embed text format is parsed by said browser to
29 automatically invoke said executable application to execute on said client workstation in
30 order to display said object."
31

32 It would have been obvious to a skilled artisan to combine (1) Raggett I's teachings
33 regarding extensions to the HTML standard (i.e., the proposed HTML+ Specification)
34 allowing the embedding of data in foreign formats within web pages with (2) the method
35 as taught by patentee's admitted prior art. This combination would have been obvious
36 based on Raggett I's acknowledgment that this particular extension to HTML is
37 advantageous and it represents a "substantial improvement." (**See Raggett I: p. 1 2nd
38 paragraph of abstract.**)
39

40 The combination of patentee's admitted prior art in view of Berners-Lee and Raggett
41 I does not explicitly teach a method wherein "the embed text format specifies the
42 location of at least a portion of an object external to the first distributed hypermedia
43 document." Raggett I describes a method in which the object itself is embedded in the
44 HTML document. (**See Raggett I: p. 6 embedded data in an external format - see**

Art Unit: 2137

1 **example on the last two lines of the page where the object, the text representation**
2 **of the equation, is within the embed tags).**

3
4 Raggett II, though, teaches putting the foreign data in a separate file and then
5 referencing that file by a URL in the HTML+ embed tag. **(See Raggett II: last line.)** It is
6 thus argued that Raggett II describes a system wherein "the embed text format specifies
7 the location of at least a portion of an object external to the first distributed hypermedia
8 document."

9
10 It would have been readily apparent to a skilled artisan to modify the method
11 discussed above, combining the teachings of the admitted prior art in view of
12 Berners-Lee and Raggett I, by further substituting a URL which references a separate
13 file containing foreign data for the embedded foreign data within the hypermedia
14 document of the combination. Such a further modification would have been apparent
15 based on Raggett II's explicit suggestion to make such a substitution. **(See Raggett II:**
16 **last line).**

17
18 The combination of patentee's admitted prior art in view of Berners-Lee, Raggett I,
19 and Raggett II does not explicitly teach a method that "enables interactive processing of
20 said object." The combination teaches a method that embeds static objects, as
21 opposed to dynamic objects, within distributed hypermedia documents.

22
23 Toye on the other hand discloses a distributed hypermedia system in which a
24 hypermedia browser allows a user to interactively process an object embedded within a
25 distributed hypermedia document **(See Toye: p. 40 description of NoteMail,**
26 **particularly p. 40, col. 2, first complete paragraph).**

27
28 It would have been readily apparent to a skilled artisan to modify the method
29 discussed above, combining the teachings of the admitted prior art in view of
30 Berners-Lee, Raggett I, and Raggett II, by further modifying the combination's static
31 embedded object to be a dynamic embedded object as taught by Toye. Such a further
32 modification would have been apparent based on Toye's teaching that its architecture
33 provides openness and flexibility **(See Toye: p. 40 col. 2 second complete**
34 **paragraph).**

35
36 Regarding claim 2 of the '906 patent, Toye teaches a method wherein "said
37 executable application is a controllable application" and the method further comprises
38 the step of "interactively controlling said controllable application on said client
39 workstation via interprocess communications between said browser and said
40 controllable application." **(See Toye: p. 40, col. 2 first complete paragraph**
41 **describing editing or updating data without leaving the notebook environment).**

42
43 Regarding claim 3 of the '906 patent, the combination of patentee's admitted prior
44 art in view of Berners-Lee, Raggett I, and Raggett II, and Toye teaches the invention

Art Unit: 2137

1 substantially as claimed. (See the rejection of claim 2, above.) However, the
2 combination of the patentee's prior art in view of Berners-Lee, Raggett I, Raggett II, and
3 Toye does not explicitly teach the additional limitation of claim 3. Nevertheless, Toye
4 teaches that selecting the displayed data within a page will restart the original
5 application so that data can be edited or updated without leaving the notebook
6 environment. **(See Toye: p. 40, col. 2 first complete paragraph)**. The term editing
7 suggests a continued and interactive process controlled by the browser user. Toye
8 teaches that this editing occurs without leaving the notebook environment. **(See Toye:**
9 **p. 40, col. 2 first complete paragraph)**. A skilled artisan would therefore reasonably
10 infer that the combination of the admitted prior art in view of Berners-Lee, Raggett I,
11 Raggett II, and Toye teaches a method wherein "communications to interactively control
12 said controllable application continue to be exchanged between the controllable
13 application" (i.e., Toye's "appropriate application") and the browser even after the
14 controllable application program has been launched.

15
16 Regarding claim 4, the combination of the admitted prior art in view of Berners-Lee,
17 Raggett I, Raggett II, and Toye teaches the invention substantially as claimed. (See the
18 rejection of claim 3, above.) The combination also describes a method wherein
19 additional instructions for controlling said controllable application reside on a network
20 server **(See Toye: p. 40 col. 2 first complete paragraph describing how the needed**
21 **application, if not locally resident, will be run remotely over the network; where**
22 **the computer remotely executing the needed application is the network server)**.
23 As to the remaining steps introduced in the claim, these steps all flow logically from the
24 movement of the controllable application from the client workstation to a network server.
25 The step of issuing, from the client workstation, one or more commands to the network
26 server flows logically from the fact that user editing commands entered at the browser
27 computer must be transmitted from the client workstation to the controllable application
28 executing on the remote machine. The step of executing, on the network server, one or
29 more instructions in response to the commands is taught by the controllable application
30 executing on the remote machine. The step of sending information from said network
31 server to said client workstation in response to said executed instructions is taught by
32 the controllable application returning a result of the editing process to the client
33 workstation. The step of processing said information at the client workstation to
34 interactively control said controllable application is taught by the client workstation
35 rendering the result of the edit in the browser window, thus allowing the user to see the
36 results of the editing operation so the user can decide what editing operation to perform
37 next.

38
39 Regarding claim 5, the combination of the admitted prior art in view of Berners-Lee,
40 Raggett I, Raggett II, and Toye teaches that the results returned by the controllable
41 application residing on the network server are displayed in the browser window. The
42 instructions performing this function are additional instructions for controlling said
43 controllable application reside on said client workstation.

44

Art Unit: 2137

1 Regarding claims 6-10 of the '906 patent, they are computer program product claims
2 corresponding to method claims 1-5, respectively. Since they do not teach or define
3 above the information in the corresponding method claims, the discussion and
4 application, supra, of the admitted prior art in combination with Berners-Lee, Raggett I,
5 Raggett II, and Toyé to method claims 1-5 is applied to claims 6-10, respectively.
6
7

8 **Response to Arguments**

9 As to the rejection of the claims under 35 U.S.C. 103(a), the patentee's
10 arguments filed on May 11, 2004 (paper no. 14) have been fully considered.

11 As to Part I of the traverse (pages 10-15 of the response), the patentee argues
12 that the specific examples of embedded objects in Raggett I and II are static and that
13 the external applications (e.g., TeX and eqn) that render those objects only return a
14 single static image to the browser (Response filed May 11, 2004, p. 12 first four
15 complete paragraphs after item b; Felten paragraphs 36-41). This argument as been
16 fully considered and is deemed persuasive. Therefore, the rejection has been
17 withdrawn. However, upon further consideration, a new ground(s) of rejection is made
18 in view of the admitted prior art in the '906 patent and the teachings of Berners-Lee,
19 Raggett I, Raggett II, and Toyé.

20 As to Part II of the traverse (pages 15-16 of the response), the patentee's
21 arguments have been considered but are not deemed persuasive. The patentee argues
22 that the Examiner has used impermissible hindsight by using the '906 patent as a
23 roadmap to modify the teachings of the references. In response to applicant's argument
24 that the examiner's conclusion of obviousness is based upon improper hindsight
25 reasoning, it must be recognized that any judgment on obviousness is in a sense
26 necessarily a reconstruction based upon hindsight reasoning. But so long as it takes

Art Unit: 2137

1 into account only knowledge which was within the level of ordinary skill at the time the
2 claimed invention was made, and does not include knowledge gleaned only from the
3 applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d
4 1392, 170 USPQ 209 (CCPA 1971). In this case, the rejection is based solely upon the
5 teachings of the references and the admitted prior art and, therefore, is not based on
6 improper hindsight.

7 As to Part III of the traverse (pages 16-17 of the response), the patentee's
8 arguments have been considered but are not deemed persuasive. The patentee argues
9 that secondary evidence supports the conclusion of nonobviousness and cites
10 evidence in the Doyle declaration showing professional approval, in the Felten
11 declaration showing the failure of others to follow Raggett I and II to implement the
12 claimed technology, and in the Krueger declaration.

13 As to the Doyle declaration, the patentee argues that the declaration shows
14 evidence of favorable reactions by experts that supports a conclusion of
15 nonobviousness. Although the Doyle declaration describes the reaction of various
16 audiences and experts as favorable, the declaration usually states these reactions were
17 favorable without explaining what the reactions were and the reason they were
18 favorable (Doyle items 3 and 6-10). There are many possible explanations for the
19 favorable reactions. For example, the favorable reactions may have been due to the
20 failure to conceive the possibility of interactive embedded objects displayed within a
21 browser window of hypermedia system. The favorable reactions may have been due to
22 the inability to figure out how to reduce to practice a preexisting conception of

Art Unit: 2137

1 interactive embedded objects displayed within a browser window of hypermedia system.
2 Or, the favorable reactions may have been due to the inventors' allocation of resources
3 to implement an obvious function that the WWW community had so far been unable to
4 devote resources to implementing. This latter interpretation of the favorable reactions is
5 consistent with Raggett's testimony that the group working on the HTML+ Specification
6 felt that there were higher priorities (Raggett – cross, p. 1884 lines 18-24). System
7 design is often incremental, and designers, having limited resources, must prioritize
8 which functions to implement first. In such a situation, just because an improvement is
9 innovative, in the sense of never having been implemented, does not mean that the
10 improvement is nonobvious. After considering the declaration's lack of specificity, the
11 declarant's obvious bias in favor of confirming the claims subject to reexamination, and
12 the other possible explanations for the favorable reactions, the Examiner concludes that
13 these facts have little probative value as to whether the technology of the '906-
14 enhanced Web browser was novel and nonobvious.

15 In items 4-5 of the Doyle declaration, the reaction of various unnamed Silicon
16 Graphics Corporation employees is characterized as "very enthusiasitic about the
17 *innovative* character" of the '906-enhanced Web browser technology (Doyle – p. 2 item
18 4 "SIGWEB") and is said to have resulted in an invitation to demonstrate the '906-
19 enhanced Web browser technology (Doyle – p. 2 item 5 "Silicon Graphics"). The
20 description of the reaction of the unnamed Silicon Graphics employees differs from the
21 reactions to the other demonstrations discussed above because a specific reaction is
22 described. However, the declaration fails to recite particular facts establishing how the

Art Unit: 2137

1 declarant established personal knowledge of these employees' states of mind. As to
2 this fact, the declaration is therefore given no weight because the declaration fails to
3 establish facts showing how the declarant's established personal knowledge of the
4 unnamed Silicon Graphics employees' states of mind. Furthermore, even if this portion
5 of the declaration should be given weight, the facts have little probative value. The
6 declaration fails to identify particular individuals who believed the '906-enhanced Web
7 browser technology to be innovative. When the declaration does name a particular
8 employee of Silicon Graphics, by stating that John Flynn invited the declarant to give an
9 on-site demonstration of the '906-enhanced Web browser technology, the declaration is
10 carefully worded to not include John Flynn in the group of unnamed Silicon Graphics
11 employees who believed the '906-enhanced Web browser technology to be innovative.
12 Given this lack of specificity and the declarant's obvious bias in favor of confirming the
13 claims subject to reexamination, the Examiner concludes that these facts have little
14 probative value as to whether the technology of the '906-enhanced Web browser was
15 novel and nonobvious.

16 As to the Doyle declaration, the declaration states in item 10 at page 3 that Dr.
17 Scott Baldwin spent several months trying to recruit Dr. Doyle to join the University of
18 Pennsylvania faculty as a result of a demonstration of the '906-enhanced Web browser.
19 The declaration is unclear as to whether this job offer was a result of the '906-enhanced
20 Web browser or the personal attributes of Dr. Doyle himself. Having met Dr. Doyle in
21 the interview on April 27, 2004, the Examiner concludes it is equally possible that Dr.

1 Baldwin would have attempted to recruit Dr. Doyle even if Dr. Baldwin thought the '906-
2 enhanced Web browser was not a patentable improvement over the prior art.

3 As to the Doyle declaration, the patentee argues that the invitation to write a
4 cover article for Dr. Dobbs Journal is evidence of a favorable reaction that supports a
5 conclusion of nonobviousness. The declaration describes how the inventors of the '906
6 patent were invited to submit an article about their "innovative 906-based browser
7 technology" (Doyle p. 3 item 11). It is unclear from the declaration who believed the
8 '906-based browser to be innovative. On the one hand, the editor of Dr. Dobbs Journal
9 may have considered the '906-based browser to be innovative and therefore extended
10 an invitation to submit an article. Or, the editor of Dr. Dobbs Journal may have invited
11 an article about the 906-based browser without expressing any opinion as to whether it
12 was innovative. There is nothing in the cited portions of Dr. Dobbs Journal indicating
13 the reason why the article was published. The declaration's description of the '906-
14 based browser as innovative may merely be the opinion of the declarant. After
15 considering the declaration's lack of specificity and the declarant's obvious bias in favor
16 of confirming the claims subject to reexamination, the Examiner concludes that this
17 portion of the declaration has little probative value as to whether the technology of the
18 '906-enhanced Web browser was novel and nonobvious.

19 As to the Felten declaration, the patentee argues that it shows the failure of
20 others to follow Raggett I and II to implement the combination used in the rejection and
21 is therefore objective evidence supporting the conclusion of nonobviousness (Felten,
22 paragraphs 45, 63). Assuming without conceding that Dr. Felten's knowledge of the

1 hypermedia art is complete, the failure of others to implement Raggett I and II's
2 proposed embed tag is not necessarily evidence that the combination is nonobvious. In
3 a standards-based technology, like the World Wide Web, there are economic and
4 practical reasons for conforming to standards. The standards process is driven by
5 consensus and the practical constraints on what is feasible to implement given limited
6 resources (Raggett – cross, p. 1884 lines 18-24 indicating that the group working on
7 the HTML+ Specification felt that there were higher priorities). The standards process
8 restricts future systems complying with the standard because new features must be
9 incorporated into the standard. Stotts, P., et al., Hyperdocuments as Automata: Trace-
10 based Browsing Property Verification, UNC CS Technical Report, TR92-038,
11 citeseer.ist.psu.edu/stotts92hyperdocument.html, p. 1, 1992. The failure of others to
12 implement Raggett I and II's proposed embed tag could therefore have been due to the
13 desire to implement systems conforming with the standard as opposed to any technical
14 limitation. The declaration ignores the fact that the World Wide Web is standards driven
15 and fails to provide any evidence that the failure to implement the combination is due to
16 technical reasons as opposed to the economic and practical reasons for conforming to
17 the standard. These portions of the declaration therefore have little probative value as
18 to whether the technology of the '906-enhanced Web browser was novel and
19 nonobvious.

20 The response also argues that secondary evidence of nonobviousness is the fact
21 that the author of Raggett I and II never contemplated the functionality described in
22 claims one and six and points to a particular portion of the trial testimony of Mr. Raggett

1 (response p. 17 after "FURTHER RE-CROSS EXAMINATION BY MR.
2 BAUMGARTNER"). This testimony must be considered in view of the totality of Mr.
3 Raggett's testimony that is presented in Appendix A to the Kreuger declaration. Mr.
4 Raggett's testimony indicates that not all ideas in the HTML+ Specification were his
5 (Raggett – direct, p. 1806 lines 12-13, p. 1809 lines 5-8, p. 1867 line 22 to p. 1868 line
6 11). Mr. Raggett's statement that *he* did not envision the functionality of Netscape
7 plugins developing the HTML+ Specification appears to be an attempt to not claim credit
8 for someone else's ideas. The fact that Mr. Raggett was unwilling to take credit for an
9 idea that he did not believe was his has no probative value to the question of whether
10 the claimed invention is obviousness.

11 The patentee also argues that it is secondary evidence of nonobviousness that
12 the embed tag proposed in Raggett I and II was abandoned for technical reasons
13 (response p. 17 – 2nd complete paragraph after Raggett's testimony). In reviewing
14 Raggett's testimony, the technical reason for not pursuing the functionality of the
15 HTML+ Specification's embed tag in version 2.0 of the HTML Specification was security
16 (Raggett direct – p. 1867 lines 5-15). The fact that the group working on the HTML+
17 Specification was uncertain as to how to securely implement embedded objects has
18 little probative value to the question of whether an insecure system, like the one
19 described in the '906 patent, is obvious.

20 The patentee also argues that it is secondary evidence of nonobviousness that
21 that the author of Raggett I and II never pointed to the HTML+ Specification as relevant
22 prior art when editing the W3C working draft "Inserting Objects into HTML," which was

1 published a few years after the HTML+ Specification. In essence, the patentee is
2 arguing that it is evidence of nonobviousness that Mr. Raggett failed to describe his own
3 work. When characterizing the working draft, the patentee argues that the working draft
4 says that "developers have been experimenting with *new* ideas for dealing with new
5 media." In examining the copy of the working draft provided by the patentee, the
6 Examiner fails to see where the ideas are described as new. The working draft only
7 says that "developers have been experimenting with ideas for dealing with new media."
8 The working draft is not, as suggested by the patentee's argument, an assertion by Mr.
9 Raggett that the idea of an HTML document containing active embedded objects is
10 new. Furthermore, Raggett's testimony indicates that the working draft was an attempt
11 to generate consensus between companies on a standard way of embedding objects in
12 web pages (Raggett redirect p. 1894 lines 7-21). Since the purpose of the working draft
13 was to reconcile the approach of the major players (i.e., Sun, Microsoft, and Netscape),
14 it has little probative value to the question of obviousness that the working draft only
15 discusses the solutions of Microsoft, Sun and Netscape and fails to exhaustively list all
16 possible solutions.

17 In view of the foregoing, when all of the evidence is considered, including the
18 evidence cited in the new grounds of rejection, the totality of the rebuttal evidence of
19 nonobviousness fails to outweigh the evidence of obviousness.

20 As to the patentee's arguments with respect to the dependent claims (pages 17-
21 19 of the response), they have been fully considered but are moot in view of the
22 new grounds of rejection.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

Conclusion

The patent owner is reminded of the continuing responsibility under 37 CFR 1.565(a), to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving Patent No. 5,838,906 throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.

In order to ensure full consideration of any amendments, affidavits or declarations, or other documents as evidence of patentability, such documents **must** be submitted in response to this Office action. Submissions after the next Office action, which is intended to be a final action, will be governed by the requirements of 37 CFR 1.116, which will be strictly enforced.

A shortened statutory period for response to this action is set to expire **two months** from the mailing date of this action.

Extensions of time under 37 CFR 1.136(a) do not apply in reexamination proceedings. The provisions of 37 CFR 1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Further, in 35 U.S.C. 305 and in 37 CFR 1.550(a), it is required that reexamination proceedings "will be conducted with special dispatch within the Office."

Extensions of time in reexamination proceedings are provided for in 37 CFR 1.550(c). A request for extension of time must be filed on or before the day on

Art Unit: 2137

1 which a response to this action is due. The mere filing of a request will not effect any
2 extension of time. An extension of time will be granted only for sufficient cause, and for
3 a reasonable time specified.

4 Any inquiry concerning this communication or earlier communications from the
5 examiner should be directed to Andrew Caldwell, whose telephone number is (703)
6 306-3036. The examiner can normally be reached on M-F from 9:00 a.m. to 5:30 p.m.
7 EST.

8
9 Any inquiry of a general nature or relating to the status of this application should
10 be directed to the Group receptionist at (703) 305-9600.

11
12
13
14 

15
16
17 Andrew Caldwell
18 703-306-3036
19 August 15, 2004

20
21
22
23
24
25
26