inlineLink: Inline Expansion Link Methods in Hypertext Browsing

Motoki Miura Buntarou Shizuki Jiro Tanaka
Institute of Information Sciences and Electronics
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan

Abstract

Conventional web browsers are designed to display a single web page for each window. A reader who wants to follow a link to a new web page while continuing to display the previous web page must do so by utilizing the "open link in new window" menu feature. Having done so, the reader may then control the positions and dimensions of both the old and the new windows, in order to display both web pages at the same time.

To reduce the number of steps that must be taken to simultaneously display two or more web pages, we are proposing a new browsing technique, which we have called "inlineLink." *This technique automatically shows both the linked web document and the original web page in a convenient format, thus allowing a reader to follow links more efficiently, without the need to manually manage window size and position, or constant use of the "Back" and "Forward" buttons. This makes it easier for readers to maintain the context of the current web pages, allowing them to concentrate on reading the content while using regular scrolling methods.*

Keywords: hypertext, browsing technique, WWW, usability, JavaScript, DOM

1 Introduction

Following links to new web pages is a significant and widely used method when browsing hypertexts. By following these links, further information related to the current page can easily be acquired. The advent of HTML and graphical web browsers such as Nexus and NCSA Mosaic made browsing a very attractive feature. However, the graphical browsing methods that are provided by conventional web browsers have not been changed since these browsers were first introduced.

1.1 Link following problems with conventional web browsers

The concept of a link is a very simple yet powerful feature. However, it is difficult and sometimes cumbersome to read hypertext documents when these documents have several pages that must be read using conventional web browsers. A typical way of following a link in a conventional web browser would be:

Operation 1 Normal clicking on an anchor.

Operation 2 Open a menu on an anchor and choose "Open link in new window."

Operation 3 Drag an anchor and drop it into another window.

Operation 1 is by far the most popular way to open a linked document. Since this operation is simple, intuitive and convenient, almost all web pages are accessed using this method, which works well unless the reader wants return to the first of the two pages frequently. The "Back" button allows the reader to look at the previous page, by replacing the currently displayed page with the previous one. However, pressing the "Back" button, requires looking away from the document, thus interrupting reading, breaking concentration and therefore making it difficult to maintain the context of the documents.

Operation 2 is used when the reader needs to display both the linked document and the previous one at the same time. This operation duplicates the current window, and then replaces the current document with the linked document in the duplicated window. However, this presents a problem,

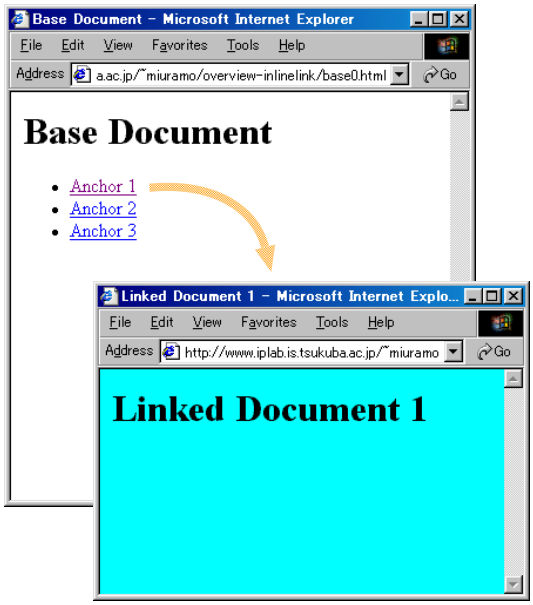

Figure 1: Following a link with a conventional technique

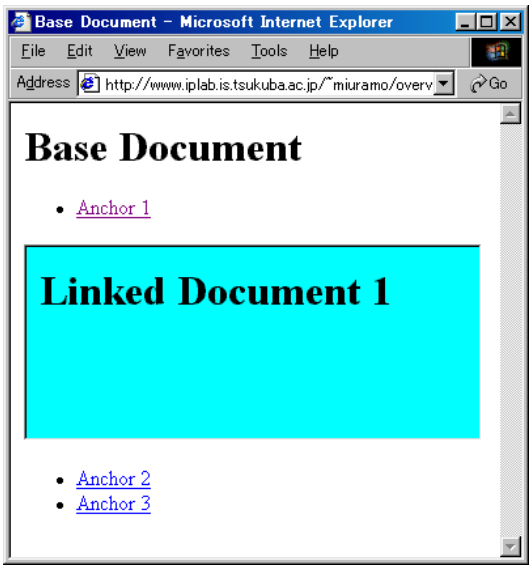

Figure 2: Following a link with inlineLink

as the duplicated window usually overlaps the current window. The reader must then change the size and location of the new, and possibly also the old, window, to be able to display both documents. These window management operations are a severe distraction to the reader's desire to read the documents efficiently.

Operation 3 may be useful if two or more non-overlapping windows have already been opened and arranged on the screen. Thereafter, the reader can choose to display an interesting link in any window that is available. Moreover, further window arrangement tasks, such as resizing and changing position, are not necessary. However, the reader is still forced to remember the relationships between the anchors and the windows in which the relevant links are displayed, while reading the documents.

1.2 The requirements for effective document reading

The problems noted above are largely due to unnecessary eye movements. The reader has to move the mouse cursor, which requires looking away from the document. This has a detrimental effect on reading performance and comprehension.

Since the procedure for following a link in conventional web browsers was not designed to reduce unnecessary eye movement, the reader cannot dedicate his full concentration to reading the document. As noted previously, following links is a very popular method that now constitutes a considerable part of a reader's activity in hypertext browsing. Therefore, it is essential that a technique be found that enables the reader to reduce the amount of eye movement necessary when following a link.

2 The “inlineLink” concept

We have designed a new link-following technique called “*inlineLink*,” which reduces eye movement. Using *inlineLink*, the linked document is displayed by inserting it near its anchor. The text following the anchor in the original document is placed after the end of the inserted document. Figure 1 shows an ordinary browser's appearance after operation 2, as described above, is performed on “Anchor 1.” Figure 2 shows the behavior opening the same link using *inlineLink*. By using *inlineLink*, several pages can be displayed simultaneously in a single window.

Even though *inlineLink* is quite simple, it is

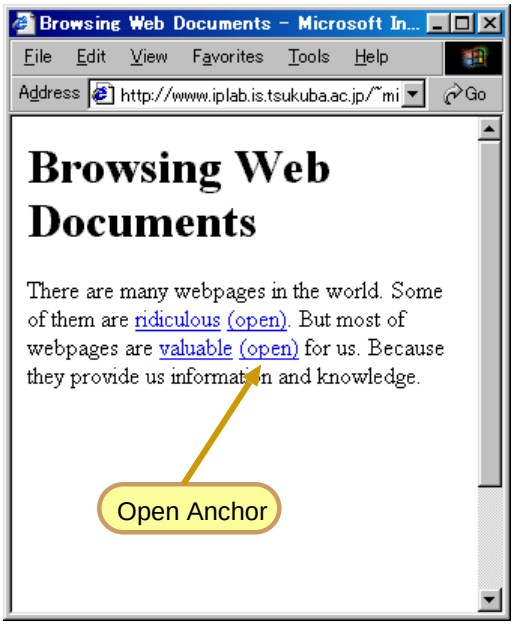

Figure 3: Open anchor (Separated from original anchor)

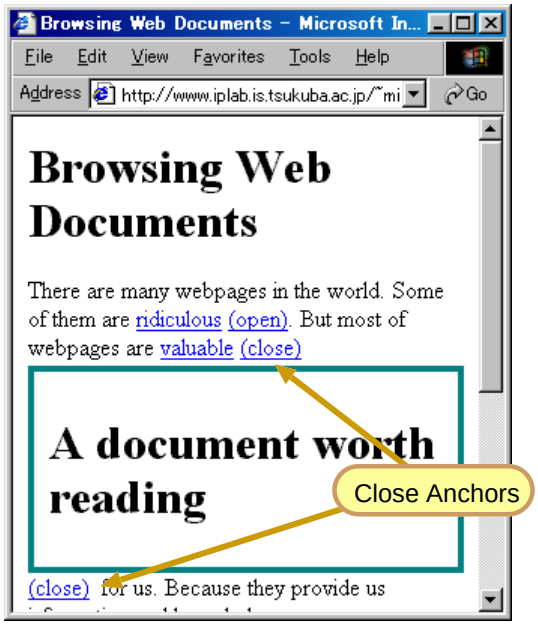

Figure 4: Embedded document and close anchors

able to significantly reduce the amount of reader distraction incurred when following a link. A link-following operation using *inlineLink* gives the reader the impression that he is "zooming in" on the link.

2.1 The open and close function anchors

inlineLink allows the reader to control the insertion of a linked page dynamically, using "open" and "close" function anchors. These function anchors enable the reader to change the state (opened or closed), style, and appearance of the linked page. An "open" anchor loads the linked document and displays it using *inlineLink*. A "close" anchor removes the inserted page from the display.

In Figure 3, the "open" anchors for the linked document are displayed separately from the original anchors. This was considered, as it allows the reader to select the linked document using the familiar link-following method if preferred. If the reader clicks the original anchor, the linked web page is displayed in the current window in place of the previous page. However, if the reader chooses the "open" anchor, *inlineLink* turns the selected "open" anchor into a "close" anchor, and then it inserts both the linked document and another "close" anchor as shown in Figure 4. The functions of the two "close" anchors are the same. We chose to add the second "close" anchor, to make it more convenient for a reader than having to return to the beginning of the inserted document to close it.

Although the technique using two "open" anchors worked effectively, the presence of the second "open" anchor proved to be distracting to the reader. Therefore, in our final design, we eliminated the second "open" anchor by integrating its functionality into the original anchor. An example of an integrated open anchor is shown in Figure 5. Using this integrated anchor is a straightforward and intuitive technique. Once the reader has selected the integrated anchor, the original anchor is displayed along with an "open" anchor to its right. The reader can then conveniently use either *inlineLink* or the conventional method to look at the linked document. While this requires an additional mouse click, it was felt worthwhile, to avoid the distraction of the additional anchor in the original document. Figure 5 also shows an example of a recursive (hierarchical) insertion with *inlineLink*; there is no limit on the number of insertion levels using *inlineLink*.

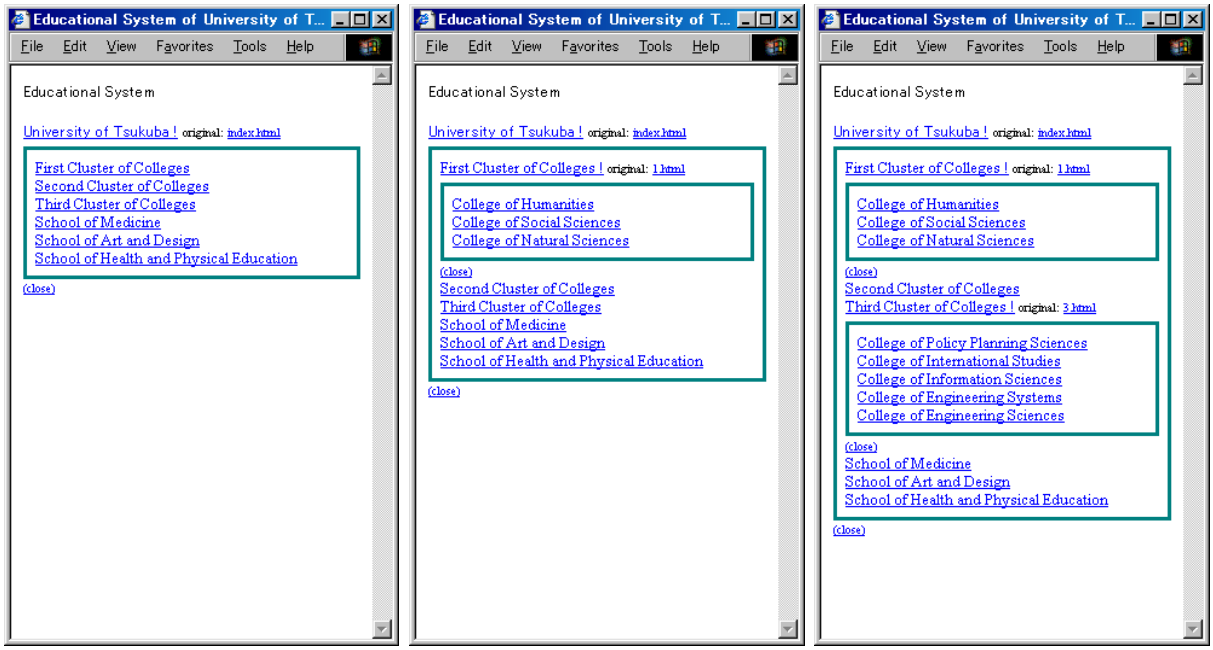

Figure 5: Hierarchical insertion (integrated open anchor)

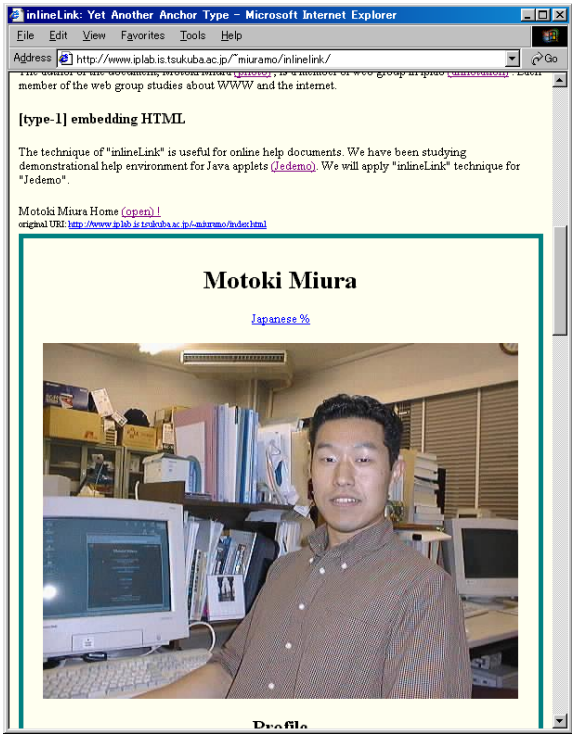

Figure 6: Whole insertion

Figure 7: Partial insertion

2.2 Two insertion styles: wholly vs. partially

In addition to its function anchors, *inlineLink* provides two document insertion styles. In the whole insertion style, all of the linked document is inserted into the base page (Figure 6). The partial insertion style displays only a portion of the linked document (Figure 7). In most cases, the whole insertion style is more effective than the partial one, because the reader is able to continue reading the linked document using normal scrolling operations. On the other hand, the partial method is convenient if the linked page is too long to display easily. The viewport of the embedded windows can be adjusted by scrolling. During browsing, the reader can manually select one of these styles for each linked document.

3 Implementation on conventional web browsers

3.1 Design

In order to add inline expansion link functions into common web browsers, we decided to utilize "Dynamic HTML." The core of the Dynamic HTML concept is a language that controls the document objects. The document objects are specified based on the Document Object Model (DOM) standard from the World Wide Web Consortium (W3C). A document written in HTML is parsed and separated into objects. These objects form a hierarchical structure that can be manipulated with instructions written in control languages such as JavaScript. This approach is convenient for both developers and readers. Once the *inlineLink* functions have been embedded into the linked document, the reader can enjoy the benefits of *inlineLink* with any of his favorite browsers.

3.2 Inserting and deleting linked documents

inlineLink alters the contents of a linked document during every insertion or deletion. Insertion consists of two steps: (1) acquire the linked document, and (2) display the linked document.

[insertion] step 1: acquire the linked document We use an "inline frame" element that can be embedded in a document. An inline frame (`iframe` element) is defined by HTML 4.0 Transitional DTD. An HTML source can be acquired and stored in an inline frame by providing its description such as `<iframe src=URI></iframe>`. By default, an inline frame is displayed with the specified dimensions. A partial insertion is realized by this framework. For a whole insertion, we use an invisible inline frame in order to load the contents of the linked document.

[insertion] step 2: display the linked document To perform the anchor rewriting and redefinition of the open/close actions, we use JavaScript. `span` elements are put into the HTML source in advance. Every `span` element can be set an identity label (ID) as an attribute. Initially, the content of a `span` element is an open anchor. When the open anchor is selected, the action defined by the anchor invokes a JavaScript function `insert_page()` with a target span ID as an argument. It also inserts the appropriate HTML sources after the close anchor. The `insert_page()` is an essential function for *inlineLink* to be able to convert an open anchor into a close anchor.

Deletion of inserted documents is easier than insertion. Since the close anchor remembers the previous contents of the open anchor, it returns to its previous state.

3.3 Implementation by JavaScript

inlineLink can be used with conventional web browsers such as Internet Explorer if an HTML document fulfills these conditions:

1. The HTML source includes a definition of *inlineLink* functions, which are necessary for an insertion and a deletion, and

2. "open anchors" are embedded in the HTML source.

A simple rewriting filter can convert an ordinary HTML page into an *inlineLink* version. We have prepared three conversion methods:

By web publisher in advance We have implemented an *inlineLink* filter written in Perl. Once the HTML source of a document has been converted using this program, the reader can follow links using *inlineLink*.

By server proxy The reader can specify a proxy server, which rewrites the source of the current document.

By client proxy: rewriting by JavaScript While the two methods described above are useful, they are not convenient. Either the publisher of the document or the reader must take the actions necessary to rewrite the document for *inlineLink*. As an alternative, we implemented a JavaScript function that automatically rewrites the loaded page before insertion. The publisher does not have to rewrite all of the contents. Only an anchor linked to the document root has to be rewritten.

Each of these methods inserts a definition of the *inlineLink* functions into an HTML source and alters the original anchors into "open" anchors (as shown in Figure 8 [original anchor] and [open anchor]). A partial source listing of an inlined page is also illustrated by the [close anchors and inlined document] section in Figure 8.

[original anchor]

```
<a href="http://www.google.com/">
google </a>
```

[open anchor]

```
<a href="http://www.google.com/">
google </a>
<span id="_google">
<a href="javascript:insert_page('_google',
'http://www.google.com/');">
(open)
</a></span>
```

[close anchors and inlined document]

```
<a href="http://www.google.com/">
google </a>
<span id="_google">
<a href="javascript:remove_page('_google',
'http://www.google.com/');">
(close) </a>
<hr>
<iframe src="http://www.google.com/" width=90% height=300>
</iframe>
<hr>
<a href="javascript:remove_page('_google',
'http://www.google.com/');">
(close) </a>
</span>
```

Figure 8: The contents in each state

4 Related works

Several approaches have been proposed to reduce unnecessary backtracking while browsing hypertexts. These approaches can be classified into two categories:

1. Showing additional information about linked pages near the anchor, to help readers to decide whether the link should be followed, and
2. Providing window management operations which are more suited to hypertext browsing.

4.1 Showing additional information

LinkPreview[1] pops up a balloon showing a thumbnail of the linked document near the anchor when the pointer is over the anchor. HyperScout Linktool[2] takes a similar approach. It pops up a balloon that contains information about the linked page, such as its title, author, language, last visited time, and server status. Neither of these approaches supports reading of hierarchically organized hypertexts. In contrast, our *inlineLink* technique allows readers to read multiple pages in a hypertext hierarchy by clipping two or more linked pages into the current one.

Fluid Links[3] proposed several kinds of display representations to render additional information about a linked page. *inlineLink* uses a portable implementation technique that realizes an effect similar to Fluid Link's *inlining* without any modification to the web browser.

4.2 Opening multiple windows

The Elastic Windows[4] method allows readers to open/close/replace pages as required in order to support typical browsing tasks. For example, a single operation can open multiple windows, each corresponding to a link on the current page. All of the windows are displayed simultaneously, and the placement and size of the windows is automatically set as part of the operation. This operation allows

the reader to browse effectively, while at the same time eliminating a considerable number of step-by-step operations such as “following a link” and selecting the “Back” or “Forward” buttons. Thereby, Elastic Windows successfully removes a significant number of tedious window operations. However, readers have to remember the relationship between the pages that are shown in the multiple windows. Moreover, the reader can only select the one operation that is most appropriate for him. In our approach, relationships are explicit since every linked document is inserted directly into the current page right after its anchor. Also, since the representation is so simple, the reader only has to choose between two operations, namely opening or closing.

5 Discussions with an informal user study

Due to some restrictions in the browser and our implementation of the current filter program, formal evaluation is not yet completed. However, an informal user study of *inlineLink* employing colleagues has been carried out.

We wrote a structured HTML document that included seven sections, one of which had six subsections (in Japanese). All of the sections and subsections are written as separate pages. The headings of each section and subsection are anchors that link to the appropriate contents. This document was distributed to the subjects for their comments.

The most popular and favorable comment from the users was “we don't need the `Back' button in *inlineLink*.” Usually, the “Back” button is used frequently while browsing. According to [5], pressing the “Back” button is the second major navigation action, constituting up to thirty percent of all navigation events. The notable reduction in use of the “Back” button represents a significant increase in browsing efficiency.

The most common complaint that we received was about scrolling. Users were reluctant to open all of the anchors, because the display length of the document becomes far too long and takes too much time to navigate. This is especially true when insertion of a lengthy page or pages causes the reader to begin to lose the context of the page, since the index of the structured document disappears from the window.

6 Conclusion

We have presented a new browsing technique called *inlineLink*, which displays linked documents by inserting them into the current page. By using *inlineLink*, use of the “Back” button while browsing hypertexts is significantly reduced. The number and complexity of other operations is also reduced, allowing a reader to concentrate on the context of the current and linked web pages. This is more efficient, and allows a better and more thorough understanding of the document.

References

[1] Theodorich Kopetzky and Max Mühlhäuser. Visual Preview for Link Traversal on the WWW. In *Proceedings of the WWW8*, 1999. `http://www8.org/w8-papers/4b-links/visual/visual.html`.

[2] Harald W. R. Weinreich and Winfried Lamersdorf. Concepts for Improved Visualization of Web Link Attributes. In *Proceedings of the WWW9*, 2000. `http://vsys-www.informatik.uni-hamburg.de/projects/hyperscout/`.

[3] Polle T. Zellweger, Susan Harkness Regli, Jock D. Mackinlay, and Bay-Wei Chang. The Impact of Fluid Documents on Reading and Browsing: An Observational Study. In *Proceedings of CHI'00*, pages 249–256, 2000.

[4] Eser Kandogan and Ben Shneiderman. Elastic Windows: A Hierarchical Multi-Window World-Wide Web Browser. In *Proceedings of UIST'97*, pages 169–177, 1997.

[5] Linda Tauscher and Saul Greenberg. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human-Computer Studies*, 47(1):97–138, 1997.