Classical Hypermedia Virtues on the Web with Webstrates

Niels Olof Bouvin
Department of Computer Science
Aarhus University, Denmark
bouvin@cs.au.dk

Clemens Nylandsted Klokmose
Center for Advanced Visualization & Interaction
Aarhus University, Denmark
clemens@cavi.au.dk

ABSTRACT

We show and analyze herein how Webstrates can augment the Web from a classical hypermedia perspective. Webstrates turns the DOM of Web pages into persistent and collaborative objects. We demonstrate how this can be applied to realize bidirectional links, shared collaborative annotations, and in-browser authorship and development.

Categories and Subject Descriptors

H.4.5 [**Hypertext/Hypermedia**]: Architectures

Keywords

Web, hypermedia, collaboration, dynamic documents

Introduction

The vision of hypermedia was to create a richly intertwingled world of words, ideas, and concepts; an all-encompassing collection of documents and works readily available for the information worker to peruse, structure, correlate, add to, and amend using powerful tools and abstractions [15, 23, 36].

What we have ended up with is both less and far more. The modern Web *is* all-encompassing in scope, and available to a high and growing percentage of the world's population. On the Web, we find incredibly rich collections of human knowledge, vast social networks connecting billions, commercial as well as public ventures that have upended business as usual, and communities, small and large, around any topic you might care to mention. The Web has shaped the way software is being built, deployed, and used. A modern developer is faced with a bewilderingly rich range of choices in technologies, platforms, frameworks, and concepts.

Yet all is not as well as it might be. The Web still does not offer the level of hypermedia structuring mechanisms once considered standard for a hypermedia system (e.g., bidirectional links). The original vision of the Web [10] included the notion of a collaborative inter-creative space where authorship and consumption were equal. While that can be found in the form of, e.g., wikis and Web-based productivity suites such as Google Docs, these are usually either limited in functionality, or closed proprietary platforms.

HT '16, July 10-13, 2016, Halifax, NS, Canada
© 2016 ACM. ISBN 978-1-4503-4247-6/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2914586.2914622

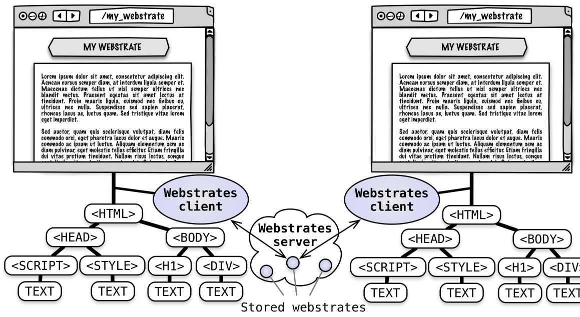


Figure 1: Webstrates persist and synchronize through operational transformations any changes to the Document Object Model of pages, called Webstrates, served from the Webstrates server to all clients of the same webstrate (from [32]).

Earlier attempts have been made to rectify some of the shortcomings of the Web, such as augmenting the hypermedia functionality available [13]. Invariantly, these attempts have not been successful in gaining widespread use.

The system Webstrates [32] demonstrate how a relatively simple change to the workings of the Web can provide users with a collaborative and, crucially, user extensible platform for the creation, editing, and structuring of documents *and* applications, limited only by the modern Web browser.

Webstrates makes the Document Object Model (DOM), which is the runtime in-memory representation of Web pages, a persisted and collaborative object. Client-side changes to the DOM are persisted to the server and synchronized to all other clients of the same page using operational transform [22] (Figure 1). Webstrates is presented as a prototype realization of a software vision, *shareable dynamic media*, that builds upon Alan Kay's seminal software vision but where collaboration among users and distribution across heterogeneous devices are emphasized. The authors demonstrate in [32] from a UI systems perspective how Webstrates can enable software malleability and personalization, collaboration with personalized interfaces, remote user interface extension at run-time, and orchestration of complex distributed and collaborative user interactions.

We review herein historical hypermedia, and based on this, analyze the modern Web, and present examples of how the extension to the Web brought forth by Webstrates can realize principles from classical hypermedia on the Web.

Classical Hypermedia Systems

Hypermedia as a discipline is quite old. We describe in this section select aspects of the systems that predate the Web, which has become largely lost, and some of which we would like to see reintroduced into common usage.

The original vision for hypermedia [15] had the knowledge worker working alone, reading, annotating, and linking documents, and occasionally sharing documents and trails with others, or purchasing trails pre-authored by *trail blazers*. As hypermedia moved from theoretical construct to actuality, Engelbart and his lab [24] led the way in demonstrating NLS/Augment, that, among many other groundbreaking achievements, featured collaborative authoring and linking of structured documents across multiple remote computers.

Much later, systems such as Intermedia [34] would demonstrate the possiblities of a collaborative hypermedia system that supported a rich set of media types and associated editors, as well as disjunct "webs" of hypermedia structures over a corpus of documents. Separating documents and hypermedia structures became a cornerstone of the Dexter model [30], and the foundations of open hypermedia, inspired in part by the observation of Meyrowitz [35] that lack of widespread hypermedia adoption could be attributed to the closed nature of the systems—all required their own environments and editors, rather than those applications commonly used. The open hypermedia community addressed this through a class of hypermedia systems [9, 18, 25, 27] that could almost seamlessly integrate with existing tools, and by doing so, allowed their users to link documents across third-party applications.

While the various open hypermedia initiatives showed that hypermedia integration was viable, the approach was both labour-intensive (making third-party applications work with a hypermedia service is hard) and fragile (a new version of an applications could break functionality; links, etc., required the hypermedia service to be present). One way to ensure robustness is to have control over editors as well as hypermedia functionality, and while this, by definition, has been the case for all monolithic hypermedia systems, one of the more remarkable was NoteCards [31]. NoteCards is a significant hypermedia system, not least because it was built in the, for the time highly advanced, InterLisp software development environment, which enabled its developers at Xerox PARC to easily extend it. Thus, NoteCards became a testbed for many different structuring mechanisms, such as graphical structure browsers, guided tours, and tabletops [40]. Having a fertile and highly interactive development platform, where developers and researchers could easily extend functionality, made this possible. A contemporaneous system was HyperCard [26], and if NoteCards represents one of the cutting edge research systems of its day, then HyperCard was the everyman hypermedia authoring tool available to all owners of Mac computers. While neither its hypermedia functionality nor its flexibility was as rich as NoteCards, it provided its users with an easily accessible development model, which made the creation of a plethora of HyperCard stacks possible. HyperCard demonstrated that if tools are available and easily used, then users can and will become creators and developers, not only of hypertexts, but of interactive programs.

The Web as Hypermedia

Whether the Web qualifies as a hypermedia system may once have been a matter of debate [38], but it is today the dominant paradigm for information exchange. While Berners-Lee's original graphical browser [10], was a browser and an editor both, the dominant paradigm became the Mosaic browser and all its heirs, which focused solely on browsing. Rather than a vision of global editability [20], it became mainly a vehicle for consumption. Web authoring became largely the domain of specialists. If readers were permitted to contribute, it was in closely delineated spaces, filling and submitting forms. With enough freedom and community dedication, very rich sites, e.g., Wikipedia, could be built.

The modern Web browser remains largely a tool for browsing, but it has gained notable functionality over the years, e.g., access to developer tools for inspection, debugging, and manipulation of Web pages. Even so, most Web development still takes place outside the browser, working with editors and servers. As shall be seen below, we believe that this need no longer be the case, and by focusing on developing *for* the browser *in* the browser (and by the people), it is possible to once again shift the way we have come to use computers.

The original hypertext pioneers had bold visions of what hypertext might entail in collecting and correlating all human knowledge, and in doing so augmenting the human mind. What they, and perhaps none, could have foreseen were the uses a system as flexible and global as the Web could be put to. Today, the Web has gone beyond "simply" hypermedia to encompass not only most of human knowledge, but also a large part of humanity's economic transactions, and a not inconsiderable part of its social and political life. Its very ubiquity, across all modern mobile and desktop platforms, makes it an ideal platform to build new types of software.

The Web supports basic inline unidirectional linking between documents, where earlier hypermedia systems often supported far richer structures, such as external bidirectional n-ary links, first-class composites, external annotations, or hierarchical structures, sometimes even collaboratively created. The lacklustre hypermedia functionality of the Web prompted the development of a special class of Web augmenting hypermedia systems [12, 14, 16, 17, 21, 29], where users could add links and other external structures to existing Web pages. Sharing the weaknesses of other open hypermedia systems, none of these systems gained any widespread use. The Web community, too, attempted generating meta-layers of meaning through XLink [17, 19] and the many initiatives collectively known as the Semantic Web [11, 33]. Both relied on either browser functionality that never materialised, or communities of use that, outside of specialist fields, never grew to a sizeable part of the Web.

However, with the advent of the modern Web browser, there is no longer any need for either integration with outside applications (as with open hypermedia) or waiting for the general adoption of new Web standards (as with the Semantic Web). Everything needed to create a high functioning hypermedia system is present in the average desktop or mobile device.

The Dexter model [30] formally recognized hypermedia structures as first class objects, and not just attributes of selections. By separating documents, anchors, and the structures referencing the anchors, one can create an extensible and collaborative environment for editing, sharing, annotating, and structuring media in its many forms. As we demonstrate below, the modern Web browser has the infrastructure to handle all this, enabling a far richer hypermedia experience than the conventional Web support.

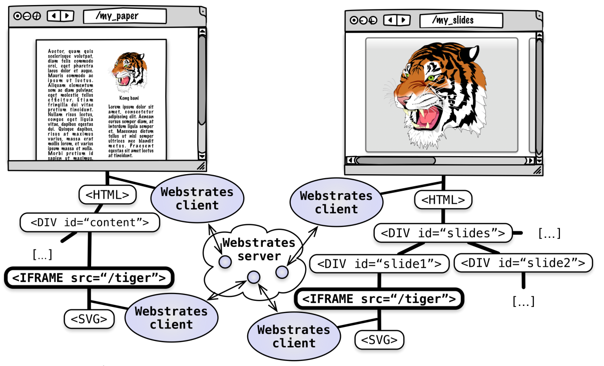


Figure 2: A figure webstrate is transcluded using the `iframe` node in both a paper webstrate and a slideshow webstrate (reprinted from [32] with permission).

Webstrates

Webstrates is a DOM-centric Web architecture that makes the DOM of any page, called a webstrate, served from the Webstrates server a collaborative and persistent object. Any changes to the DOM of a webstrate are persisted on the server and synchronized to all other clients of the same webstrate, including changes to any inlined JavaScript or CSS. Consistency between clients is ensured through operational transformations [22]. Webstrates does not introduce a new framework or API, but instead relies on the standardized API of and interaction with the DOM, but with with a new set of rules of engagement.

Klokmose et al. [32] demonstrate a number of uses for Webstrates focusing on enabling software malleability and personalization, collaboration through personalized interfaces, remote user interface extension at run-time, and orchestration of complex distributed and collaborative user interactions. These examples leverage *transclusions* [37] of one webstrate in another. The Web supports transclusion through the **`iframe`** tag, where one Web page can be embedded in another by reference. When transcluder and transcludee are served from the same domain, the JavaScript runtime of the two pages can interact. Thus, JavaScript from the outer page can manipulate the DOM of the inner and vice versa.

Transclusion can allow a document webstrate to be transcluded into an application-like webstrate that provides tools for editing the document. This mechanism can allow two or more users in real-time to collaborate on authoring a paper through personal text-editors that are functionally as well as visually different, and the authors can, at run-time, share tools and extend their respective user interfaces. A text-editor webstrate can be transcluded into a developer webstrate and thereby allow live modification of a running user interface (of another user). Transclusion in Webstrates is illustrated in Figure 2 where a figure webstrate is both transcluded in a paper webstrate and a slideshow webstrate. Changes to the figure webstrate will be reflected in both the paper and slideshow.

Development and authoring in webstrates happen *inside* the browser; either directly through the developer tools of the browser, or through webstrates built for development as, e.g., the code editor mentioned above. Thus, the distinction between development and use or browsing and authorship becomes a phenomenon of use.

Reference implementation

The reference implementation of Webstrates [8] consists of a NodeJS [3] server and a transparent JavaScript client. Synchronization and consistent concurrent editing of the DOM are implemented using operational transformations [22] through the ShareJS [6] library, and webstrates are persisted in a MongoDB [2] database as JsonML [1] documents. When a Web page is loaded from the Webstrates server, the Webstrates JavaScript client is statically served to the client browser. The client asynchronously loads the data of the given webstrate in JsonML from the server based on the resource name in the URL, and populates the DOM of the page. The client observes local changes to the DOM using the MutationObserver DOM API [7] and subscribes to changes from other clients of the same page through a web socket connection to the server (see Figure 1). Observed mutations to the DOM are translated to JSON operations on the JsonML representation of the DOM. These operations consist of an absolute path into the JSON document, an action (e.g., insertion in a list or deletion in a string) and a value (e.g., the element to be inserted or the substring to be deleted). As the browser guarantees that the DOM can be serialized to well-formed HTML, our JsonML representation will always be well-formed as well.

The reference implementation includes a basic authentication mechanism using external providers (e.g., GitHub), and webstrates can be annotated with access rights (read-write, read, no access) as an attribute on the root `HTML` node of the DOM. A new webstrate is created by requesting a new resource name. Due to the reliance of transclusion through iframes, the current implementation of Webstrates is centralized, as modern browsers restrict crossing iframe boundaries when pages are served from different domains. This is something that can be overcome technically, but which also opens up for a number of security challenges.

The reference implementation provides a simple HTTP API for creating new webstrates using other (versions) of webstrates as prototypes. Duplicating a webstrate can also be achieved by copying the HTML serialized DOM of a webstrate into another using the developer tools of the browser, as all state and behavior are stored in the DOM.

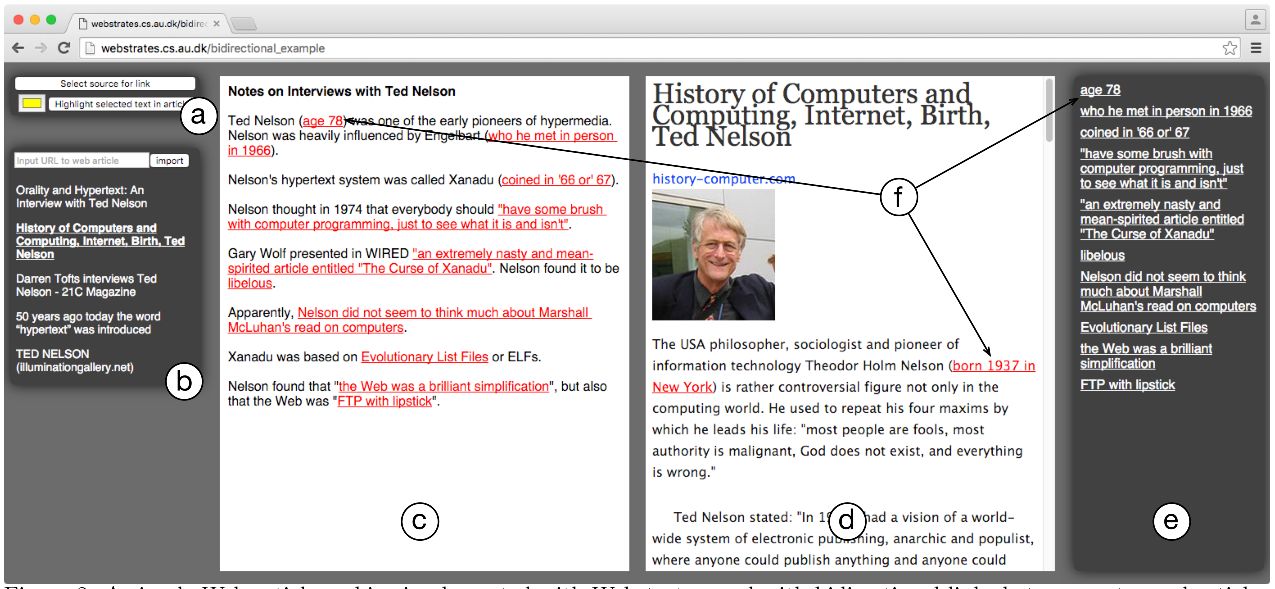
Demonstrators

We present below two demonstrators that illustrate how rich hypermedia functionality can be realized in Webstrates. Both demonstrators are simple yet usable pieces of software that each required little more than a work day to implement. While there are many other kinds of hypermedia structures that we could have chosen as our test cases (e.g., guided tours [40] or fluid annotations [41]), linking and annotating form the basis on which many other mechanisms can be built.

Web article archive with bidirectional links

The first demonstrator is a Web article archive, where a user can collect articles, take notes, and create bidirectional links between the articles and the notes. Figure 3 shows a user researching interviews with Ted Nelson.

The Web archive consists of three types of webstrates: an archive webstrate (the top level webstrate), a set of article webstrates, and a webstrate for storing an external bidirectional link collection. The user can paste in a URL to a Web page (Figure 3b top), upon which it will


webstrates.cs.au.dk/bidirec
webstrates.cs.au.dk/bidirectional_example

Select source for link
Highlight selected text in artic

Input URL to web article
import

Orality and Hypertext: An Interview with Ted Nelson

History of Computers and Computing, Internet, Birth, Ted Nelson

Darren Tofts interviews Ted Nelson - 21C Magazine

50 years ago today the word "hypertext" was introduced

TED NELSON (illuminationgallery.net)

Notes on Interviews with Ted Nelson

Ted Nelson (age 78) was one of the early pioneers of hypermedia. Nelson was heavily influenced by Engelbart (who he met in person in 1966).

Nelson's hypertext system was called Xanadu (coined in '66 or' 67).

Nelson thought in 1974 that everybody should "have some brush with computer programming, just to see what it is and isn't".

Gary Wolf presented in WIRED "an extremely nasty and mean-spirited article entitled "The Curse of Xanadu". Nelson found it to be libelous.

Apparently, Nelson did not seem to think much about Marshall McLuhan's read on computers.

Xanadu was based on Evolutionary List Files or ELFs.

Nelson found that "the Web was a brilliant simplification", but also that the Web was "FTP with lipstick".

History of Computers and Computing, Internet, Birth, Ted Nelson

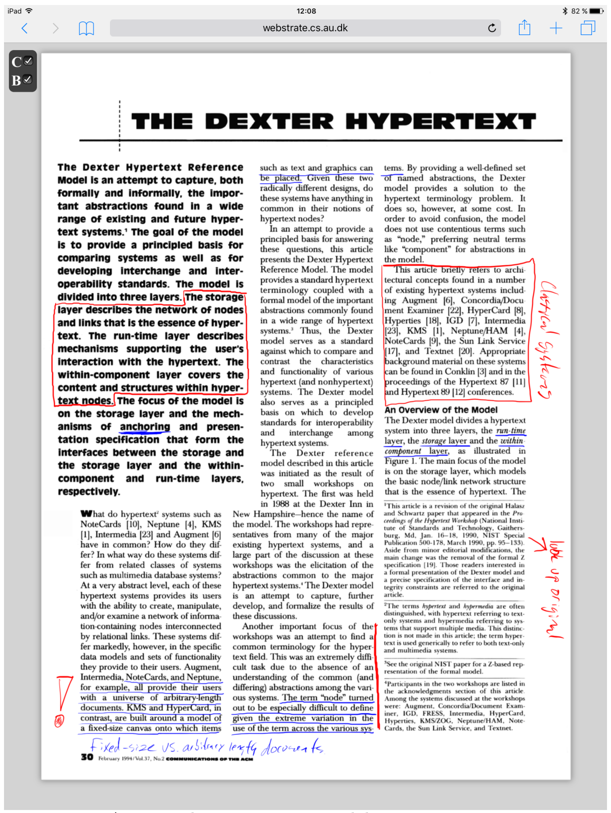
history-computer.com

The USA philosopher, sociologist and pioneer of information technology Theodor Holm Nelson (born 1937 in New York) is rather controversial figure not only in the computing world. He used to repeat his four maxims by which he leads his life: "most people are fools, most authority is malignant, God does not exist, and everything is wrong."

Ted Nelson stated: "In 1[illegible] had a vision of a world-wide system of electronic publishing, anarchic and populist, where anyone could publish anything and anyone could

age 78
who he met in person in 1966
coined in '66 or' 67
"have some brush with computer programming, just to see what it is and isn't"
"an extremely nasty and mean-spirited article entitled "The Curse of Xanadu"
libelous
Nelson did not seem to think much about Marshall McLuhan's read on computers
Evolutionary List Files
the Web was a brilliant simplification
FTP with lipstick

Figure 3: A simple Web article archive implemented with Webstrates, and with bidirectional links between notes and articles. a) Toolbar for creating links and highlighting text. b) Archive of Web articles. New articles are added using the form in the top. c) Notes. d) Article view. e) Collection of bidirectional links stored in a transcluded webstrate. f) The two anchors of a bidirectional link and the link in the external link structure.



THE DEXTER HYPERTEXT

The Dexter Hypertext Reference Model is an attempt to capture, both formally and informally, the important abstractions found in a wide range of existing and future hypertext systems.[1] The goal of the model is to provide a principled basis for comparing systems as well as for developing interchange and interoperability standards. The model is divided into three layers. The storage layer describes the network of nodes and links that is the essence of hypertext. The run-time layer describes mechanisms supporting the user's interaction with the hypertext. The within-component layer covers the content and structures within hypertext nodes. The focus of the model is on the storage layer and the mechanisms of anchoring and presentation specification that form the interfaces between the storage and the storage layer and the within-component and run-time layers, respectively.

What do hypertext[2] systems such as NoteCards [10], Neptune [4], KMS [1], Intermedia [23] and Augment [6] have in common? How do they differ? In what way do these systems differ from related classes of systems such as multimedia database systems? At a very abstract level, each of these hypertext systems provides its users with the ability to create, manipulate, and/or examine a network of information-containing nodes interconnected by relational links. These systems differ markedly, however, in the specific data models and sets of functionality they provide to their users. Augment, Intermedia, NoteCards, and Neptune, for example, all provide their users with a universe of arbitrary-length documents. KMS and HyperCard, in contrast, are built around a model of a fixed-size canvas onto which items such as text and graphics can be placed. Given these two radically different designs, do these systems have anything in common in their notions of hypertext nodes?

In an attempt to provide a principled basis for answering these questions, this article presents the Dexter Hypertext Reference Model. The model provides a standard hypertext terminology coupled with a formal model of the important abstractions commonly found in a wide range of hypertext systems.[3] Thus, the Dexter model serves as a standard against which to compare and contrast the characteristics and functionality of various hypertext (and nonhypertext) systems. The Dexter model also serves as a principled basis on which to develop standards for interoperability and interchange among hypertext systems.

The Dexter reference model described in this article was initiated as the result of two small workshops on hypertext. The first was held in 1988 at the Dexter Inn in New Hampshire—hence the name of the model. The workshops had representatives from many of the major existing hypertext systems, and a large part of the discussion at these workshops was the elicitation of the abstractions common to the major hypertext systems.[4] The Dexter model is an attempt to capture, further develop, and formalize the results of these discussions.

Another important focus of the workshops was an attempt to find a common terminology for the hypertext field. This was an extremely difficult task due to the absence of an understanding of the common (and differing) abstractions among the various systems. The term "node" turned out to be especially difficult to define given the extreme variation in the use of the term across the various systems. By providing a well-defined set of named abstractions, the Dexter model provides a solution to the hypertext terminology problem. It does so, however, at some cost. In order to avoid confusion, the model does not use contentious terms such as "node," preferring neutral terms like "component" for abstractions in the model.

This article briefly refers to architectural concepts found in a number of existing hypertext systems including Augment [6], Concordia/Document Examiner [22], HyperCard [8], Hyperties [18], IGD [7], Intermedia [23], KMS [1], Neptune/HAM [4], NoteCards [9], the Sun Link Service [17], and Textnet [20]. Appropriate background material on these systems can be found in Conklin [3] and in the proceedings of the Hypertext 87 [11] and Hypertext 89 [12] conferences.

An Overview of the Model

The Dexter model divides a hypertext system into three layers, the *run-time* layer, the *storage* layer and the *within-component* layer, as illustrated in Figure 1. The main focus of the model is on the storage layer, which models the basic node/link network structure that is the essence of hypertext. The

[1]This article is a revision of the original Halasz and Schwartz paper that appeared in the *Proceedings of the Hypertext Workshop* (National Institute of Standards and Technology, Gaithersburg, Md, Jan. 16–18, 1990, NIST Special Publication 500-178, March 1990, pp. 95–133). Aside from minor editorial modifications, the main change was the removal of the formal Z specification [19]. Those readers interested in a formal presentation of the Dexter model and a precise specification of the interface and integrity constraints are referred to the original article.

[2]The terms *hypertext* and *hypermedia* are often distinguished, with hypertext referring to text-only systems and hypermedia referring to systems that support multiple media. This distinction is not made in this article; the term hypertext is used generically to refer to both text-only and multimedia systems.

[3]See the original NIST paper for a Z-based representation of the formal model.

[4]Participants in the two workshops are listed in the acknowledgments section of this article. Among the systems discussed at the workshops were: Augment, Concordia/Document Examiner, IGD, FRESS, Intermedia, HyperCard, Hyperties, KMS/ZOG, Neptune/HAM, NoteCards, the Sun Link Service, and Textnet.

30 February 1994/Vol.37, No.2 COMMUNICATIONS OF THE ACM

Figure 4: A PDF being annotated by two users, one on an iPad. Their individual annotation layers can be toggled using the controls in the top left corner.

be processed into clean easily readable HTML (using the Readability [5] API) and stored in a new webstrate. A link to the article webstrate is added to the DOM of the archive webstrate (Figure 3b). Clicking on an article link will translude it in an `iframe` in the archive webstrate (Figure 3d). The notes (Figure 3c) are a `div` element in the archive webstrate that is editable through the `contentEditable` attribute. To create a link, the user makes a selection in the notes, presses the "select source for link" button (Figure 3a), makes a selection in an article, and presses the same button now labelled "select target for link". An anchor element with a unique ID is added to both the notes and the article (of the form `<a id='anchorId'>text</a>`), and a bidirectional link node is added to an external link webstrate that is trancluded in an `iframe` (Figure 3e-f). The bidirectional link has the form `<bilink anchor1='url:anchorId' anchor2='url:anchorId'>text</bilink>`. An anchor in the notes or articles is only visualized as links (red and underlined) if the anchor is part of a bidirectional link in the link collection. Clicking a link in the notes will open the linked article and highlight the anchor. Clicking a link in an article will scroll the notes to the given anchor and highlight it. Clicking a link in the list of bidirectional links will highlight the link in both notes and article.

This demonstrator shows that it is straightforward to implement bidirectional links with an external link structure on top of the Webstrates platform. We leverage that changes to the DOM are persisted, so storing a link, the HTML for an article, or a note is just a matter of manipulating the DOM, and does not require any explicit interaction with, or development of, a backend server. Here, we use inlined anchors. This works well for a single user application, but if articles and notes were shared between users, each applying their own external link structures, documents could become polluted with anchors, leading to conflicts with overlapping anchors (forbidden in HTML), and requiring the users to have write permission to the linked webstrates. Alternatively, external, computable location specifications [28] could be used. This could be implemented by storing a DOM query expressed as a JavaScript function in the bidirectional link instead of an explicit anchor reference, not unlike [39]. A third more robust option would be to extend Webstrates with unique DOM node IDs, and link directly to nodes or sets of nodes.

This demonstrator is designed for a single user, but could easily be extended to support external link collections shared between users similar to how it is done with annotations in the next demonstrator. The implementation of this demonstrator contains ∼240 lines of JavaScript (excluding libraries).

Shared PDF annotations

The second demonstrator is a collaborative PDF annotation tool. Figure 4 features the first page of [30] displaying annotations of two users, C (blue) and B (red). This demonstrator consists of three types of webstrates: a PDF data webstrate, two PDF viewer webstrates, and two annotation webstrates. The data webstrate contains the raw data of a PDF as a base64 encoded string directly in the DOM. The PDF viewer webstrates transclude the data webstrate and render the PDF on a `canvas` element using PDF.js [4]. The viewer webstrate is personal and identifies the given user and stores their preferences (e.g., annotation color). The PDF is annotated using a pen (here an Apple Pencil on an iPad Pro) or a mouse. Annotations are stored in separate webstrates transcluded into the viewer and rendered on top of the PDF. Thus, annotations can be shared between users. Figure 4 shows user C's PDF viewer, where user B's annotations has been transcluded in as well and can be toggled on and off using the top left controls. The PDF viewer observes changes to annotations (using the MutationObserver DOM API), so user C will see any new annotations B may make in real-time. Changing page is done by updating a page number attribute in the DOM of the view. This is done through JavaScript using the keyboard on a desktop computer and with gestures on a tablet. Thus, page browsing will be synchronized between all devices with the same viewer open.

This demonstrator shows how external annotation layers can be created using Webstrates and transclusion, and how Webstrates can facilitate collaboration. It also shows how non Web-native media like a PDF can be integrated using modern Web standards. The implementation of this demonstrator contains ∼400 lines of JavaScript code (excluding libraries).

Discussion

We believe Webstrates has the potential to be a vehicle to bring the roots of hypermedia (back) into the Web. With the two demonstrators above we show how external link structures (in the vein of the Dexter Reference Model) and annotation layers can be realized as hypermedia documents themselves; documents that can be shared and collaboratively edited. Webstrates breaks with the tradition of Web development happening offline on static files. Webstrates is *document centric* and makes the DOM a first class citizen of a hypermedia system, just as earlier hypermedia systems elevated the link and the anchor. This is in contrast to the current trend in Web frameworks where the DOM is used primarily as an ephemeral view in a traditional Model-View-Controller application architecture. Development of Webstrates currently requires experience in Web development and a certain proficiency in using the developer tools of the browser, but we can easily imagine webstrates developed for end-user authorship akin to HyperCard and beyond. Webstrates is yet an immature platform, but it has demonstrated significant potential; not necessarily to support the Web as a heavy-weight application platform to create software for millions, but instead as a platform for user expression and *personal* media. Limitations and challenges of implementing Webstrates include (but are not limited to) dealing with large amounts of data exceeding what can be stored in the DOM (and held in memory of the browser), overcoming the restrictions of transclusion using `iframes` while at the same time keeping user data safe, some performance and synchronization issues (as raised in [32]), and providing a distribution platform for users to share and collaborate on webstrates.

Conclusion

The Web grew to its current popularity partly because of its simplicity, but lost in the process much that had been commonplace for hypermedia systems. Now the Web and its tools have evolved to a stage, where the common browser can become the platform for changing the way we work with information, alone or together. We have demonstrated that principles of classical hypermedia systems can be reinvigorated on the Web by rethinking the serving foundation of the Web, and in doing so we have made these principles available across both desktop and mobile devices.

We invite the interested to experiment with the reference implementation of Webstrates [8], and to explore the possibilities of what can be achieved once you realize that a synchronised DOM forms a strong foundation for collaborative hypermedia on the Web.

Acknowledgement

This work was funded by ParticipatoryIT, AU, Denmark.

References

[1] JsonML. http://jsonml.org/.

[2] MongoDB. http://mongodb.org/.

[3] Node.js. http://nodejs.org/.

[4] PDF.js. http://mozilla.github.io/pdf.js/.

[5] Readability. https://readability.com.

[6] ShareJS. http://github.com/share/ShareJS/.

[7] W3C DOM4. http://www.w3.org/TR/domcore/.

[8] Webstrates. http://github.com/cklokmose/Webstrates.

[9] K. M. Anderson, R. N. Taylor, and E. J. Whitehead Jr. Chimera: Hypertext for heterogeneous software environments. In *Proc. 1994 Euro. Hypertext Conf.*, pages 97–107. ACM, Sept. 1994.

[10] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollerman. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 1(2), 1992.

[11] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):28+, 2001.

[12] N. O. Bouvin. Unifying strategies for Web augmentation. In *Proc. 10th Hypertext Conf.*, pages 91–100, Darmstadt, Germany, Feb. 1999. ACM Press.

[13] N. O. Bouvin. Augmenting the Web through open hypermedia. *The New Review of Hypermedia and Multimedia*, 8:3–26, 2002.

[14] N. O. Bouvin, P. T. Zellweger, K. Grønbæk, and J. D. Mackinlay. Fluid annotations through open hypermedia: Using and extending emerging Web standards. In *Proc. WWW2002 Conf.*, pages 160–171, Honolulu, USA, May 2002. W3C. .

[15] V. Bush. As we may think. *The Atlantic Monthly*, 176 (1):101–108, July 1945.

[16] L. A. Carr, W. Hall, and S. Hitchcock. Link services or link agents? In *Proc. 9th Hypertext Conf.*, pages 113–122, Pittsburgh, PA, USA, June 1998. ACM Press.

[17] B. G. Christensen, F. A. Hansen, and N. O. Bouvin. Xspect: bridging open hypermedia and XLink. In *Proceedings of the 12th International World Wide Web Conference*, pages 490–499, Budapest, Hungary, May 2003. W3C, ACM Press. .

[18] H. C. Davis, D. E. Millard, S. Reich, N. O. Bouvin, K. Grønbæk, K. M. Anderson, U. K. Wiil, P. J. Nürnberg, and L. Sloth. Interoperability between hypermedia systems: The standardisation work of the OHSWG. In *Proc. 10th Hypertext Conf.*, pages 201–202, Darmstadt, Germany, Feb. 1999. ACM Press.

[19] S. DeRose, E. Maler, D. Orchard, and B. Trafford (editors). XML Linking Language (XLink). W3C Recommendation 27 June 2001, W3C, June 2001. http://www.w3.org/TR/xlink/.

[20] A. Di Iorio and F. Vitali. From the writable web to global editability. In *Proc. 16th Hypertext Conf.*, pages 35–45, New York, NY, USA, 2005. ACM.

[21] O. Díaz, C. Arellano, and M. Azanza. A language for end-user web augmentation: Caring for producers and consumers alike. *ACM Trans. Web*, 7(2):9:1–9:51, May 2013.

[22] C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In *ACM Sigmod Record*, volume 18, pages 399–407. ACM, 1989.

[23] D. Engelbart. A conceptual framework for the augmentation of man's intellect. In P. Howerton, editor, *Vistas in Information Handling*, volume 1, pages 1–29. Spartan Books, Washington DC, USA, 1963.

[24] D. C. Engelbart and W. K. English. A research center for augmenting human intellect. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pages 395–410, New York, NY, USA, 1968. ACM.

[25] A. M. Fountain, W. Hall, I. Heath, and H. C. Davis. Microcosm: An open model for hypermedia with dynamic linking. In *Proc. Euro. Hypertext Conf.*, 1990.

[26] D. Goodman. *Complete HyperCard 2.0 Handbook*. Random House Inc., 1990.

[27] K. Grønbæk and R. H. Trigg. Design issues for a Dexter based hypermedia system. *CACM*, 37(2):40–49, Feb. 1994.

[28] K. Grønbæk and R. H. Trigg. Toward a Dexter based model for open hypermedia: Unifying embedded references and link objects. In *Proc. 7th Hypertext Conf.*, pages 149–160, Bethesda, MD, USA, Mar. 1996.

[29] K. Grønbæk, N. O. Bouvin, and L. Sloth. Designing Dexter based hypermedia services for the World Wide Web. In *Proc. 8th Hypertext Conf.*, pages 146–156, Southampton, UK, Apr. 1997. ACM Press.

[30] F. G. Halasz and M. D. Schwartz. The Dexter hypertext reference model. *CACM*, 37(2):30–39, Feb. 1994.

[31] F. G. Halasz, T. P. Moran, and R. H. Trigg. NoteCards in a nutshell. In *Proceedings of ACM Conference on Human Factors in Computing Systems and Graphics Interface*, pages 45–52, Toronto, Canada, Apr. 1987.

[32] C. N. Klokmose, J. R. Eagan, S. Baader, W. Mackay, and M. Beaudouin-Lafon. Webstrates: Shareable dynamic media. In *Proc. UIST 2015*, pages 280–290, New York, NY, USA, 2015. ACM. .

[33] C. C. Marshall and F. M. Shipman. Which semantic web? In *Proc. 14th Hypertext Conf.*, pages 57–66, Nottingham, UK, Aug. 2003. ACM Press. .

[34] N. K. Meyrowitz. Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. In *OOPSLA 1986 Proc.*, 1986.

[35] N. K. Meyrowitz. The missing link: Why we're all doing hypertext wrong. In E. Barrett, editor, *The Society of Text: Hypertext, Hypermedia and the Social Construction of Information*, pages 107–114. MIT Press, Cambridge, USA, 1989.

[36] T. H. Nelson. *Computer Lib/Dream Machines*. Mindful Press, 1974.

[37] T. H. Nelson. The heart of connection: hypermedia unified by transclusion. *CACM*, 38(8):31–34, 1995.

[38] P. J. Nürnberg and H. Ashman. What was the question? reconciling open hypermedia and world wide web research. In *Proc. 10th Hypertext Conf.*, pages 83–90, Darmstadt, Germany, Feb. 1999. ACM Press.

[39] T. A. Phelps and R. Wilensky. Robust intra-document locations. In *Proc. WWW2000 Conf.*, pages 105–118, Amsterdam, Holland, May 2000. W3C.

[40] R. H. Trigg. Guided tours and tabletops: tools for communicating in a hypertext environment. In *Proc. CSCW 1998 Conf.*, pages 216–226, 1988. .

[41] P. T. Zellweger, N. O. Bouvin, H. Jehøj, and J. D. Mackinlay. Fluid annotations in an open world. In *Proc. 12th Hypertext Conf.*, pages 9–18, Århus, Denmark, Aug. 2001.