

# An Architecture for Ink Annotations on Web Documents

Sriram Ramachandran  
Dept of Electrical Engineering  
Rutgers University, Piscataway, NJ  
[sriram@caip.rutgers.edu](mailto:sriram@caip.rutgers.edu)

Ramanujan Kashi  
Avaya Labs Research  
Basking Ridge, NJ  
[ramanuja@research.avayalabs.com](mailto:ramanuja@research.avayalabs.com)

## Abstract

*There have been recent improvements in document technologies like the standardization of object interfaces to access and manipulate the properties of web documents. There has also been significant progress in pen based computing for recognition of digital ink in desktops, tablets and handheld devices. These have necessitated a need for further research on annotation architectures for digital documents, specifically pen-based annotation systems. This paper presents an attempt to leverage the new standards of DHTML and W3C DOM that are being gradually implemented by popular browsers, to build a prototype of an ink annotation system with common components across browsers. One of the primary goals in this study is to semantically link ink data with underlying document elements like text and images. The system has three components: a) Ink capture and rendering b) Ink Understanding, which recognizes and associates ink with the underlying document and c) Ink storage and retrieval.*

## 1. Introduction

The variety of annotation systems ranges from uses as simple as personalizing web pages with simple locally-stored annotations to complex collaboration systems involving multiple annotation servers and clients (discussion servers for instance). Existing annotation systems offer facilities of highlighting text within a web document, adding notes at certain points or creating annotated links to other resources [1,2,3,5]. None of them seem to offer a comprehensive ink annotation and manipulation framework. There have been a few annotation systems [10,11] offering basic pen functions like rendering static ink over the web page but they do not support ink associations with the underlying document. The role of pen or ink annotations in paper documents cannot be emphasized enough. Ink has served as the most normal and convenient medium to make ideas clearer in any kind of paper document, in such forms as sketches, handwritten text notes and annotations on images or pictures. These capabilities would also significantly enhance current digital documents, of which HTML-based web documents form a major part.

Since web documents can only be changed on the server side, annotations could either be stored on the client side [1] or in annotator proxies [12] as separate layers of accompanying data that are retrieved and rendered when the browser loads the document resource. There are some interesting challenges to be faced while designing an ink annotation framework. Rendering and retrieval of the ink along with the document resource when the browser window is resized or with a different browser, and when variations occur in font size and html content, pose challenging problems. Developing efficient, standardized storage for ink annotations with the corresponding associated document elements is an important issue. Inking with pen or mouse need not be used only for ink-annotating the document but can also be used as input gestures for navigating & manipulating the underlying document. Finally, semantically associating the ink with the underlying document structure is a fertile topic of ongoing research [14].

This effort is aimed at developing a preliminary prototype to capture, render and understand ink data on web documents. Part of the framework is building tools to retrieve object information using the W3C's Document Object Model (DOM) that is being implemented to some extent in most popularly used browsers. The DOM level 1 specifies elementary methods for interacting with the elements in XML or HTML documents with practically no support for pen annotations, because there are no event modules. The DOM 2 specifies events, views, traversal and range recommendations that are implemented to some degree in the IE6 and NS6 browsers. These provide the basic backbone for any pen annotation system [8].

Some of the tools that we have built include capturing ink coordinates (mouse positions for desktop and pen positions for handhelds), rendering them and associating the ink with the underlying objects on the web pages. These underlying objects are accessed using the DOM APIs. The ink captured has then been processed in two ways. In the annotation scenario, the ink is rendered along with the documents and is also saved along with the underlying documents. In the recognition scenario, the ink captured is treated as a pen gesture and is sent to a recognition module that decides on the actions to be taken.

## 2. Ink Capture & Rendering

The functionality provided by the browsers for DOM and Dynamic HTML (DHTML) is used for capturing mouse coordinates. The mouse event property of the DOM Window Object gives direct access to the instant ink coordinates. The ink coordinates are smoothed in real time using a hysteresis filter [16] to reduce the jitter introduced by the stylus or the mouse. Such a non-linear filter also helps in smoothing out the jaggedness associated with writing notes. These coordinates are screen coordinates relative to the top and left of the user area of the browser, which is our origin of reference. After adding offsets for scrolling, the absolute position of the mouse with respect to the reference origin is obtained. Keyboard modifiers – ctrl, alt, and shift keys – have been used to differentiate the ink to be used either for the annotation mode or the recognition mode. The implementation of the capture engine is slightly different for different browsers. Event handling functions handle mouse events like up, down, move, and drag and populate data structures with the coordinates for recording.

The rendering is done using browser-specific components (ActiveX for Internet Explorer) and the above event-handlers deal with the allocation and withdrawal of the drawing components like the pens, colors, the device context of the browser window and refreshing the browser. Rendering the pen or mouse movements is a browser specific task. A rendering engine has been developed for Internet Explorer 6 using helper objects and connecting to the webBrowser2 COM component [15].

## 3. Ink Understanding

We have classified ink understanding into two separate stages: ink gesture recognition, and ink-to-document association. Once the ink points are captured, smoothed and rendered, they are sent for computation to either a gesture module or to an ink-document association module in the current implementation. Another component that is relevant to ink understanding is the ink registration module. This comes into play when loading the annotation layer in the browser after the document is loaded and when there are changes in browser attributes. This is discussed in Section 6, Ink Retrieval.

### 3.1. Ink Gesture Recognition

The gesture module is currently a very basic gesture recognizer that can recognize short straight-line gestures like  $\rightarrow$ ,  $\leftarrow$ ,  $\uparrow$ ,  $\downarrow$ . The algorithm we currently have implemented determines the slope of the best-fit line

obtained by the ink coordinates captured during the gesture mode.

The ink-gesture is checked against the above-mentioned gestures and on a match the appropriate gesture handlers are invoked. Some examples of gesture handling routines that we have implemented modify the document structure (annotations like highlighting, bold, etc.) by using DOM APIs, or access the document history object for navigation, or create a partial HTML page with inline annotations. We are in the process of building a recognizer to understand more complex gestures. At the current time, it illustrates the utility of combining gestures with the DOM to create annotations. Table 1 tabulates the accuracy of our gesture recognition. These results were obtained with 1000 gesture instances with about 250 instances for each of the four gestures.

**Table 1. Accuracy of the gesture recognizer.**

Gesture	Left $\leftarrow$	Right $\rightarrow$	Up $\uparrow$	Down $\downarrow$
Accuracy	100 %	99%	98%	99%
Action	Creates a partial HTML page	Changes all current Text Ranges	Scrolls up	Moves back in History

### 3.2. Ink to document association

The ink-to-document association algorithm has been implemented in Javascript [17] using DOM and DHTML to associate underlying HTML elements with ink. The logical mapping from screen points in the physical coordinate system to HTML elements is achieved by modifying basic DOM methods. For instance, the Internet Explorer 6.0 (henceforth IE6) DOM gives access to text range objects at the word or character level given the physical coordinates in the browser user area. The type of HTML elements in proximity with the ink points are determined using DOM APIs to decide whether the ink association is to be mapped with just text elements or with table, image or object elements.

Figure 1 shows the association of ink with text elements. The bounding box of the ink is determined and the text ranges within the boundary region are queried using the DOM API's. The collection of the text ranges within the ink-bounding box is then stored with the annotated ink.

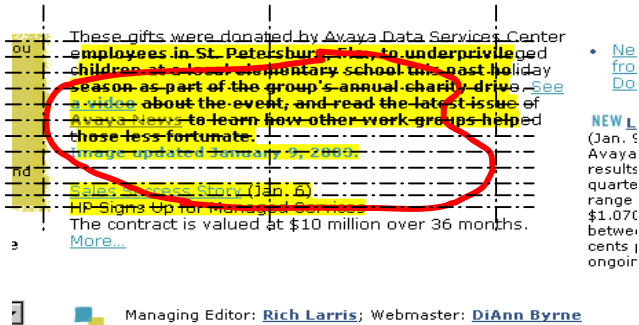


Figure 1 Associating text with an enclosed shape created by ink. The text ranges within the bounding rectangle of this shape is joined to obtain the text associated with the ink.

Figure 1 also shows the selected text ranges by change in the background color. This is also seen in Figure 2 for the text associated with pen annotations.

Selecting text ranges with ink is useful for selecting columns within tables. This is shown in Figure 2 where the first column of the stock table is selected using ink. We obtain an array of text ranges and the DOM can specify the corresponding elements (the column tag, here), so we can repopulate a table with that column. Other usefulness includes selecting a block of non-contiguous text that can be stored in the clipboard and for creating an annotated partial HTML page for offline collaboration and as an authoring tool to clip webpages for viewing on handheld devices. (Fig. 3)

DOM and change the background and font of all those ranges.

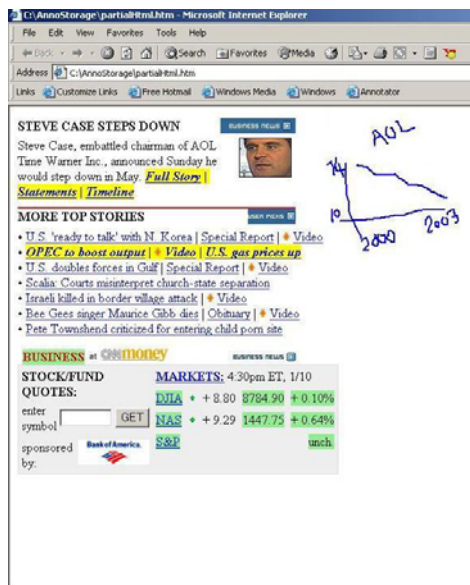


Figure 3 Partial HTML page with inline annotations from Figure 2. This partial page has been further annotated by the addition of a hand drawn plot.

The W3C DOM provides methods to get fragments of a HTML page. We can get fragments of a selection inside an HTML page and reconstruct the partial HTML. We use the Selection object provided in the DOM of popular browsers to obtain the ranges and create a complete new page from a fragment. In our implementation, a gesture handler implements this method to pop up a partial page that has a dynamic snapshot of the annotations in the main page, as shown in Figure 3. Figure 3 shows a partial HTML file from Figure 2 and has all the annotations transferred as well. Note that the ink seen in Figure 2 was used to select the various text ranges and subsequently annotated (bold, italics etc). Another important use of ink annotation is the ability to draw sketches or write a note as shown in Figure 3.

#### 4. Ink storage

In the current prototype implementation, the inking coordinates and all the attributes and properties needed to store ink annotations are stored in the local client machine as a separate annotation layer. Whenever the browser loads a URL the layer is dynamically overlaid on the rendered document.

The inked points, text ranges, relative reference positions and other annotation attributes like window

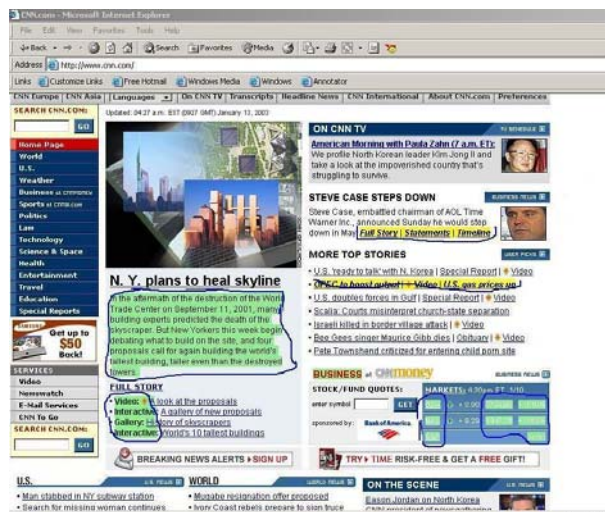


Figure 2 Web page with ink used to select text ranges.

The text ranges themselves are present in an annotation specific data structure such as a collection or an array. A subsequent call to a gesture recognizer can access the

sizes and time stamps are stored along with the URL of the annotated page in an annotation XML schema [6, 13], as shown in Figure 4. The DOM gives access to the bounding rectangles where the text ranges are rendered by the browser. The ink points are first converted into coordinates relative to the top left corner of the bounding box of one of the ranges. Most tags in the XML schema and values are self-explanatory. The different styles that the text can be modified to, and the different options for pens and brushes, can be added to the STYLES element as STYLE and PENSTYLE child elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<ANNODATA>
  <STYLES>
    <STYLE>
      <NAME>Default</NAME>
      <STYLESTRING>
        color:#000000;backgroundcolor:#ffff00
      </STYLESTRING>
    </STYLE>
    <STYLE> ..... </STYLE>
  </STYLES>
  <PENSTYLE>
    <NAME>REDMEDIUM</NAME>
    <STYLESTRING>
      pencolor #ff0000; penwidth 2
    </STYLESTRING>
  </PENSTYLE>
  <PENSTYLE> ..... </PENSTYLE>
</ANNODATA>
<ANNOTATIONS>
  <ANNOTATION TYPE:TEXT_INK POSITION:RELATIVE>
    <AUTHOR>Ramanujan Kashi</AUTHOR>
    <STYLENAME>Default</STYLENAME>

    <REFURL>http://www.avaya.com/</REFURL>
    <TIMING>30 Dec 2002, 17:50:23</TIMING>

    <WINDOW>
      <WIDTH>800</WIDTH>
      <HEIGHT>580</HEIGHT>
      <CLIENTWIDTH>780</CLIENTWIDTH>
      <CLIENTHEIGHT>2000</CLIENTHEIGHT>
    </WINDOW>
    <REFTEXT NUM:9>
      <RANGE ID:1 OC:1> employees </RANGE>
      <RANGE ID:2 OC:1> children at a local </RANGE>
      <RANGE ID:3 OC:1> season as part of the </RANGE>
      <RANGE ID:4 OC:1> video about the event </RANGE>
      .....
    </REFTEXT>
    <TITLE> Avaya Net Home Page </TITLE>
    <LINK> </LINK>
    <INKSTRING REF:1 SAMPLERTIME:1100 NUM:125>
      204,409;1,0;1,2; .....
    </INKSTRING>
    <REFPOS>
      <TOP>381</TOP>
      <LEFT>210</LEFT>
    </REFPOS>
    <ID> E34&5%FOL4$DR#(U </ID>
  </ANNOTATION>
</ANNODATA>
```

**Figure 4 – XML Schema for Annotation storage**

The annotations are stored within ANNOTATION child elements within the ANNOTATIONS element. The attributes of the ANNOTATION element define the type and position of the annotations. The type could be TEXT\_INK for text linked to ink, PLAIN\_INK for unassociated and unrecognizable ink (graphics and sketches), or CURSIVE\_INK for ink recognized to be cursive text. The position attribute just indicates if the annotation position is relative to some HTML element or if it is the absolute position when first rendered in a browser. There are other child elements within each annotation with obvious values. The WINDOW child of an ANNOTATION element gives an indication of the window size and client area of the document when the annotation occurred. The attributes of the INKSTRING element give the characteristics of ink, namely the time for sketching excluding pen-up time (SAMPLERTIME attribute) and the number of inking points (NUM attribute) to be parsed from the string representation.

The REFTEXT element of a TEXT\_INK annotation is populated with the RANGE children that just contain anchor text from the text-range array. The LINK child, if populated, gives an indication that the entire annotation is linked to point to another resource, which could be a URL or an ID of another annotation. On the basis of its attributes every annotation can be hashed to a unique ID that is stored as an ID child element in the annotation itself and which can be used to address the annotation. This could help in linking Ink Annotations among themselves and also to document URLs.

The CURSIVE\_INK annotations also could have the same child elements as TEXT\_INK annotations, as they can also be associated semantically to document elements. But the main distinction is the child element CURSIVETEXT that would contain recognized text. The PLAIN\_INK annotations are the ones that cannot be recognized as any shape or any text and also cannot be associated with any document text and elements. As such, the fields would be the same as TEXT\_INK annotations except for the REFTEXT child element. They have an absolute position attribute and can statically be positioned at the same point in a browser window.

## 5. Ink retrieval

Whenever a page is loaded into the browser, the corresponding event from the DOM invokes our retrieval handler. From the stored XML file, as shown by the schema in Figure 4, the URL presence is checked in all REFURL tags, and the available ink, style, reference position, and the text string attributes are read from each confirming annotation element. The strings are parsed

into values and populated in runtime annotation objects. The occurrence-counts (OC) or rank of the text strings within the XML annotation file (the OC attributes of RANGE elements) are also found. Now, the loaded document can be searched for the particular ranges using DOM element access and walk-through methods. From the boundary rectangles of the text ranges, the reference point for plotting ink is obtained. Using a simple shift of origin from the REFPOS reference element and the current reference point, ink can be plotted onto the document. This methodology is dynamic and hence it works in conditions such as a browser window resize.

We may also decide to show or hide the ink part of the annotation within the current document depending on absent text ranges due to modification of the document or if the bounding rectangles of the ranges do not match up with the area covered by the bounding rectangle of the Ink. The latter case occurs when text ranges wrap around during the rendering. The ink associated linked text ranges are normally rendered in some changed format from their normal rendering so as to show the association. Our implementation changes the background or the bold, italic attributes of the text as soon as the association is complete.

## 6. Summary

This effort is part of an ongoing project and our current implementation is a preliminary prototype of an ink annotation system for web documents. A number of useful tools have been developed for the ink capture, rendering, storage, retrieval and understanding. We are working on enhancing our prototype to support collaboration and developing additional methodologies to semantically associate the ink with the underlying documents.

## 7. References

- [1] Laurent Denoue and Laurence Vignollet, "An annotation tool for Web Browsers and its applications to information retrieval", Proceedings of RIAO 2000, Sixth Conference on Content Based Multimedia Information Access, Paris, France, April 2000.
- [2] A.J. Bernheim Brush, "Annotating Digital Documents for Asynchronous Collaboration", Technical Report, Dept. of Computer Science and Engineering, September 2002
- [3] Marja-Ritta Koivunen, Dan Brickley, Jose Kahan, Eric Prud'Hommeaux, and Ralph R. Swick, "The W3C Collaborative Web Annotation Project", World Wide Web Consortium, May 2000
- [4] Thomas A. Phelps and Robert Wilensky, "Robust Intra-document Locations", Proceedings of the Ninth World Wide Web conference, Amsterdam, May 2000.
- [5] Thomas A. Phelps and Robert Wilensky, "Multivalent Annotations", Proceedings of First European Conference on Research and Advanced Technology for Digital Libraries, Pisa, Italy, September 1997.
- [6] David Bergeron and Anoop Gupta, "A Common Annotation Framework", Microsoft Tech. Report, MSR-TR-2001-108, 2001
- [7] Venu Vasudevan and Mark Palmer, "On Web Annotations: Promises and Pitfalls of Current Web Infrastructure", Proceedings of the 32<sup>nd</sup> Hawaii International Conference on System Sciences, 1999
- [8] Ben Chang, Mike Champion, James Davidson, Angel Diaz, Andy Heninger, Joe Kesselman, Philippe Le Hégaret, Arnaud Le Hors, Tom Pixley, Jared Sorensen, Ray Whitmer, Lauren Wood, "Document Object Model (DOM) Requirements", W3C Working Draft, April 2001
- [9] Jose Kahan, Marja-Riita Koivunen, Eric Prud'Hommeaux and Ralph R. Swick, "Annotea: An Open RDF Infrastructure for Shared Web Annotations", Proceedings of the Tenth World Wide Web Conference, Hong Kong, May 2001, pp 623-632
- [10] Bill N. Schilit, Gene Golovchinsky and Morgan N. Price, "Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations", Proceedings of CHI 1998, Los Angeles, CA, April 1998, pp.249-256
- [11] iMarkup Solutions, <http://www.imarkup.com/>
- [12] T. Hiroshu, T. Takada, S. Aoyagi, K. Sato, and T. Sugawara, "Cmew/U – A Multimedia Web Annotation Sharing System", Proceedings of the IEEE Region 10 Conference (TENCON '99), Cheju, Korea, September 1999, pp. 356-359
- [13] Patrick Phillippot, Where Did I Read That?, PC Magazine (online), April 9, 2002  
[http://www.pcmag.com/print\\_article/0,348,a=24007.asp](http://www.pcmag.com/print_article/0,348,a=24007.asp)
- [14] W3C Semantic web  
<http://www.w3.org/2001/sw/>
- [15] Scott Roberts, "Programming Microsoft IE 5", Microsoft Press, 1999
- [16] Richard Duda & Peter Hart, "Pattern Classification and scene analysis", John Wiley & Sons, NY, 1973
- [17] David Flanagan, "JavaScript: The Definitive Guide, 4th Edition, O'Reilly, NY 2001.