PDA Web Browsers: Implementation Issues*

Ian Cooper†& Royston Shufflebotham‡
Computing Laboratory
The University of Kent at Canterbury
Kent CT2 7NF
United Kingdom
email: (ihc,rws)@ukc.ac.uk

November 9, 1995

Abstract

Personal Digital Assistants (PDAs) are not currently used as Web browsers. However, their portability and personal nature offer significant advantages over other environments. By directly addressing the perceived disadvantages of PDAs, we demonstrate that effective Web browsers can be developed using current technology.

Keywords: PDA; WWW; Web; personal environment; overview; folding; StretchText; byte-range; HTML

1 Introduction

Personal Digital Assistants (PDAs) such as the Apple Newton are becoming increasingly popular for mobile computing. At the same time, the popularity of the World-Wide Web [BLCL+94]is also growing rapidly. At present, most PDAs are isolated from the rest of the world. However, as increasing numbers of people want mobile Internet access, natural progression suggests that World Wide Web clients will be developed for PDAs.

Apart from their obvious advantage of being portable, pen-based (or pen-supplemented) interfaces offer considerable advantages over any other type of interface for Web browsing:

- The primary Web interface action is the selection of links. This is intuitive (simply a case of tapping on the link) and direct (the reader operates with the link directly). The interface doesn't get in the way of the task [GK94].
- Form-filling is simply a case of writing within the appropriate fields, or choosing between a set of options. (It would, however, be useful to extend input types to help PDAs determine the acceptable data in some places, as discussed below.)

* HTML version at <URL:http://alethea.ukc.ac.uk/Dept/Computing/Research/STEP/Papers/WWWPDA/>
† <URL:http://alethea.ukc.ac.uk/wp?94ihc>
‡ <URL:http://alethea.ukc.ac.uk/wp?94rws>

The fact that a user is mobile no longer means that they cannot connect to other computing resources. Indeed, the Motorola Marco and Envoy (repackaged versions of the Apple Newton and Sony MagicLink) are already supplied with built-in radio modems, allowing them to communicate with Internet Service Providers. The Newton MessagePad 120 incorporates support for using the GSM (Global System for Mobile communications) phone network. This allows mobile telecommunications within, and between, countries which are part of the GSM network. People are increasingly using GSM to give world-wide connectivity to the World-Wide Web.

1.1 Personal information

The function of a PDA is to store personal information. This information can be readily extended to Web hotlists and passwords, which *enrich* the personal environment. The PDA may then *automatically* perform actions such as Web authentication on behalf its user, without having to ask.

There is naturally a higher probability that personal information, such as passwords, will be stored on a PDA rather than a multi-user system. The PDA is *personal* to its owner, and is additionally protected by its physical security, since it is seldom left unattended by its user.

Other computing environments allow personal information to be stored. However, they do not have the same level of integration potentially available on PDAs. Applications normally store information in their own configuration files, as there is no central location where they are guaranteed to find it. For example, Unix Netscape stores the users name in a file `.netscape-preferences`. This information may already be stored in configuration files for other environments. Further, since personal information is potentially available by other users, it is unlikely that people will want to store it in these environments.

1.2 PDAs as our only Web browser?

PDAs have more than enough power for simple Web browsing. The majority of browsing time is spent with the user reading information on the screen (when the machine is relatively idle). The communications bandwidth is currently the limiting factor of mobile Internet devices, although this is the case with *any* dial-up account.

We do not suggest that PDAs become our only platform Web browsing. Instead, the PDA will supplement other environments. We will be able to interact with the Web while away from our usual workplace, to find resources that can be studied in more detail when there is superior equipment available, to read a personalised newspaper, or to *surf* for pleasure when there is nothing else to do. No longer must we wait until we are in front of a desktop computer to access the World of information in the Web - we carry the information in our hands.

When working in their office, users are likely to have more powerful equipment available. Docking a portable machine (as we currently dock laptop computers) into higher performance equipment allows the advantages of the portable's personal environment to be used, while providing higher specification peripherals.

The *concept* of docking a PDA is already widespread. The portable machine will often be connected to a "base-station" in order to transfer information (in some situations it is possible for data to be shared directly - one machine directly reading the data from another). Exchangeable memory cards (such as the PCMCIA card) are also used. Interestingly, the PCMCIA card already allows a user to carry their personal environment with them. They can simply plug the card into another machine and carry on working.

However, this is not a substitute for a PDA: the environment is not continually available.

2 Related work

Two PDA based Web browsers have already been implemented. Both use the Apple Newton as the development platform:

1. Gessler & Kotulla developed a browser that performs (virtually) all client functionality on the Newton [GK94]. HTML documents are downloaded via a serial link connected to a host computer, which provides access to the Internet. This dependence on a physical link was due only to the limitations of wireless communications at the time of development.
2. Bartlett's model was to use the PDA as a "video-text client" [Bar95]. Web pages were transferred to, and rendered by, a host computer. The graphical representation of the text, and link information, was then sent to the Newton via a modem connected to a portable telephone system.

These systems demonstrate that it is possible to develop a PDA-based Web browser. However, we feel that the current systems are just direct ports of desktop applications, and have neither exploited the advantages of the PDA nor addressed their limitations.

Bartlett implies that there is no real problem with the screen size, stating that users were more aware of the lack of functionality (inline images and forms had not been implemented). Gessler, on the other hand, suggests that there *is* a problem: the reduction in the amount of visible text hinders the ability of the reader to keep an overview of where they are in the document.

2.1 'Problems' with PDAs as Web browsers

We outline the 'problems' of using a PDA as a Web browser below. As we hope to demonstrate, it is only the limited screen size that presents a genuine problem.

1. **Screen size**
   The screen is usually the largest part of a PDA, and therefore dictates the overall size of the machine. The physical size of the screen must therefore remain small in order for a PDA to remain highly portable.

   One means of enlarging the screen image would be to project it onto a larger surface. Given sufficient resolution, the user could then produce as large a screen as required. (There are obvious issues of privacy and power supply which we do not intend to address here.)

   A small screen limits the amount of text that can be displayed on it, and limits the displayable width of any nested lists. In the authors' experience, Web pages that fit in their entirety on a PDA screen would only be used as a means of locating further information. Pages containing usable content often run to more than a single screen on a static terminal, and would therefore be much longer than a single PDA screen.
2. **Screen resolution**
   Screen resolution is currently low. A Newton screen has a resolution of 320×260 pixels, while even low specification PCs tend to have a minimum resolution of 640×480 pixels (four times the area). This reduction in resolution affects the way that images are displayed (an image takes up a larger

proportion of the screen on a low resolution device than it would on a high resolution device) and the quality of text on the display.

PDA screen resolutions will undoubtedly increase, just as resolutions on other systems have increased. This will provide better graphics presentation. However, experience indicates that the content of a very high proportion of Web pages is not affected if graphics are removed, so long as the pages have been well designed. It is therefore not currently necessary for a PDA to be able to handle detailed graphics.

3. **Lack of colour**
A lack of colour affects the way in which hypertext links can be presented to the reader. On colour screens it is sometimes possible to turn off the "traditional" underlining of links, relying instead upon a different colour to convey the information, reducing screen clutter. Figure 1 shows links using colour (blue) only. Note that the links would not be visible on a monochrome screen.

Figure 1: Coloured links only

On a monochrome screen, it is necessary to convey the information by some other means. In the Web environment, this has traditionally been accomplished by underlining the links. Figure 2 shows links using underlining. Note that the links clutter the text, making it harder to read.

Figure 2: Underlined links

The introduction of colour screens to new PDAs will undoubtedly happen in the near future: laptop computers have already had colour displays for a number of years, and it is a natural progression that PDAs will have them also. Many Web browsers on desktop systems rely upon underlined text to denote hypertext links, even though colour is available. The short term lack of colour on PDAs is consequently not a key issue.

4. **Lack of font sizes**
PDAs tend to have a limited number of font sizes. This is a function of the screen size: large fonts reduce the amount of information that can be displayed on the screen.

This lack of font sizes affects the way that HTML headings would be rendered on a PDA. On other 'graphical' platforms (platforms that use graphical fonts rather than a text mode) a range of font sizes is used to show headings from level 1 to level 6. Text mode browsers have already addressed this issue. For example, the Lynx browser [MGR] displays headings with a range of indentations and capitalisation.

The authors feel that it is not necessary to have different sized fonts for headings beyond level 3 (<H3>). Headings 4 to 6 are rarely genuinely required (in the authors' experience they are only used to obtain "small" text), as a level 3 heading approximates to the \subsubsection part of a LaTeX document.

A lack of many font sizes is therefore less important than might initially be anticipated.

2.2 Summary

We have demonstrated that some of the traditional "problems" of PDAs do not restrict Web browsing, but that screen size *is* a limiting factor. In the next section we outline a range of interface functionality that may be used to reduce the effect of 'information overload' on a small screen.

3 New ways of presenting information

In this section we outline a number of ways in which required screen resources for Web pages can be reduced.

3.1 *StretchText* approach

StretchText [Nel87] is suggested as a way of providing different levels of explanation of a problem within a single document.

The user chooses the level of display by operating a "throttle" (a slider or some such device) to control the level of detail shown on the screen. As the slider is moved, the amount of text grows or shrinks, giving more or less detail, moving the reader between different "altitudes" of the text.

The Guide hypertext system [Bro86] is based on a StretchText model. Users select buttons within the text to expand the level of detail shown. This allows the reader to explore the areas they are interested in, while ignoring the others.

This model is very useful for small screen displays. Users can limit the amount of information shown, to present just that in which they are particularly interested.

The concept of StretchText does not carry well to span-based environments (where sections of the text are enclosed within spans of mark-up, such as emphasis). The authors do not believe that it would be possible to design a *span*-based StretchText model within HTML (where a span is a range of text). In order to present valid HTML documents, various parts of the HTML mark-up must be correctly nested. It is not possible for a StretchText span to overlap other mark-up, since this would interfere with nesting.

For example:

```
<em>wibble wibble<st level="4"> wobble</em>woggle</st>
```

(where <st> represents a fictional StretchText tag.)

More important considerations are, perhaps, authoring effort and document size. The authoring effort required to develop a StretchText document would be very high - even with appropriate tools - if all possible documents (visible at different "altitudes") are to remain readable. The mark-up required to generate the StretchText also adds significantly to its size if large amounts of stretchable text is added.

A coarser approach would be to allow individual sentences and paragraphs to be marked with a display level (altitude). This would allow entire sentences or paragraphs to appear and disappear within the text. HTML need only be modified to add support for sentences, and to allow additional StretchText level attributes within both the new sentence and existing paragraph structures. Authoring effort is reduced, and the size of the resulting HTML file is not greatly enlarged.

Paragraphs and sentences are marked with their altitude, defaulting to level 0 if not supplied. Level 1 is more abstract, level 2 more abstract still. (The browser software would determine the highest altitude based on the content of the current document.)

For example:

```
<p><s level="2">More important considerations are, perhaps, authoring
effort and document size.</s> <s level="1">The authoring effort required
to develop a StretchText document would be very high - even with
appropriate tools - if all possible documents (visible at different
``altitudes'') are to remain readable.</s> <s>The mark-up required to
generate the StretchText also adds significantly to its size if large
amounts of stretchable text is added.</s></p>
```

(where <s> represents a new sentence construct.)

An associated concept is that of "folding", as discussed in the next section.

3.2 Folding

Folding is a process whereby text is hidden behind a descriptive title. This is different to the StretchText approach, where the user would not see some details at particular levels. With folding, the heading remains visible. Folding is often used as a convenient way of hiding information of little relevance, while the user concentrates on something else. The folded heading is retained as a means of accessing the hidden information.

Certain constructs within HTML documents enable them to be folded with relative ease:

1. Headings, if used in their correct order, act as containers of the text between them. Thus, all text (and headings) between two headings at level 1 (<H1>) belong to the first heading (this is similar to the way that paragraphs and subsections belong to a \section or \chapter in LaTeX [Lam94]).

   Divisions (as suggested in HTML 3.0 [Rag]) allow a way of grouping information into chapters, sections, etc. This is more powerful that the "good authoring practice" mentioned above.

   Headings (or divisions) can be folded to provide different overview levels of the document. Selective unfolding displays detail on a specific section of a document (if one heading is unfolded), or the entire document (if all headings are unfolded).

2. Ordered, unordered, and descriptive lists may all be collapsed. There are some common, and list-specific ways in which this folding can be done:

   - It is intuitive that lists (and sublists) will have introductory text. Lists and sublists may therefore be folded under their introduction.
   - Descriptive list information (<DD>) may be folded under the corresponding title (<DT>). By definition, the description (which may include further subtitles and descriptions) belong to the title.

[GK94] identifies a significant problem with the presentation of nested lists. Traditionally, when displaying a list, items are indented from the left margin by a substantial amount. Subsequent sub-lists are indented further.

With the limited screen resources of PDAs, it is difficult to indent in this manner. The level of difficultly is related to the screen orientation (a portrait oriented screen will naturally suffer more). In Gessler's software, nested lists were collapsed to the first level. This removes the structural information that the author wished to convey. We need to provide information about the nested lists in a visual format not dependent on large indentations.

When colour screen PDAs become available, it may be possible to use colour to represent the relative depth of nested lists. A current approximation is the use of coloured ball images in unordered lists. There are obvious problems related to the presentation of coloured link buttons within coloured text, and the problems of certain colour combinations.

A second method is to use vertical bars identifying the level of nesting beside the list. An example with a simple list is shown in Figure 3.

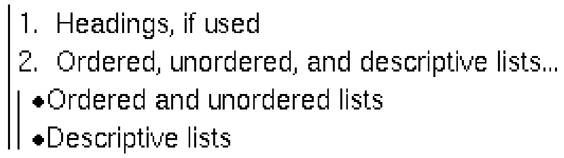

Figure 3: Use of vertical bars to show nesting

Again, there are problems with this model, particularly when nesting beyond five levels is reached.

The problem of screen orientation can be partially solved by rotating the PDA (and therefore its screen) by 90 degrees - something we are not likely to be able to do with a standard terminal.

3.3 Document overviews & byte ranges

It would be beneficial to provide a "table of contents" for every document, within the HTTP response header [BLFFN95]. This would be used to provide an initial overview that could be unfolded. This could be further extended in order to reduce network traffic, by providing a byte-range of the extent of each heading, so that only those sections that the reader explicitly unfolds are downloaded.

This byte range information is extremely fragile, changing between different versions of the same document, and should only be generated by the computer. There are two ways of generating such an overview:

1. The overview is extracted by the server at the time of the HTTP request. This method would place a significant load on the server as it would have to generate the overview information for each request. (This impact could, of course, be reduced by caching the results of previous requests.)
2. The overview is automatically extracted by a document management system (part of the authoring process) and is placed into a special meta-information area of the server. (The CERN HTTPD server [LFBL] can send this information within an HTTP response header.) The system only needs to extract the information once for each revision of the document.

> Document management will be used increasingly as more documents are created. As the authoring of HTML moves towards true authoring environments, and away from basic text editing, these facilities will become part of the authoring environment.
>
> The *Alethea*[1] Service authoring environment is designed to separate documents in preparation from documents to be served. During the transfer of a document from the development area to the live area, a utility responsible for the transfer automatically extracts meta-information from the file. This meta-information is stored in a special area on the server, and is sent to the client in the HTTP response header.

Upon receiving the meta-information from the server (via an HTTP HEAD), the client software can produce an overview of the document. Individual sections of the document may be unfolded by the reader, causing byte-range HTTP GET requests to be sent to the server. The server responds with the appropriate section of the document, thus reducing network traffic for larger documents (provided that the reader does not read the entire document!). (Smaller documents should be downloaded in full by the client.)

As briefly stated above, byte range information is fragile. In the time between an overview being supplied and a request for a byte range being made, the position of a piece of text as identified by its byte range can move. All transactions working on byte ranges must therefore be time-stamped.

When a document overview is obtained the server time-stamps it. When a request is made for a byte range, it incorporates the associated time-stamp from the overview. The server then compares this time-stamp with the last-modified time of the file. If the file has been modified then the request must fail. However, the new byte range information would be returned in the failure response header enabling a possible re-request to be sent. There are obviously some pages (particularly those generated by server-side scripts) that are not appropriate for such byte range requests.

We are *not* discussing the possibility of providing support for byte range requests from standard HTML links.

3.4 Mini-overviews

Displaying a shrunken image of a document enables us to present more structural detail than is possible in a standard, *clipped*, view. This concept is used in print preview facilities in many word processors.

Mini-overviews are provided in a range of mobile computers to compensate for their reduced screen size. The Cambridge Z88 provided a permanent on-screen active overview of a final A4 page, which was modified as the user typed. The Psion Series 3a has print-preview options within its standard applications. The NewtonBook facility on the Apple Newton provides graphical representations of pages to act as bookmarks. Within these overviews, basic structures of the document can be identified.

Figure 4 shows a screen shot of the Psion Series 3a print preview on an early draft of this paper. Lists can clearly be identified in all of the above panes.

Figure 5 shows an enlarged version of the first pane in Figure 4. The nesting of lists and sub-lists is clearly visible.

The screen coordinate system is effectively enlarged when providing mini-overviews. We are therefore able to show more of the document on one screen, and are able to exaggerate horizontal indentation to make lists more obvious.

[1] <URL:http://alethea.ukc.ac.uk/>

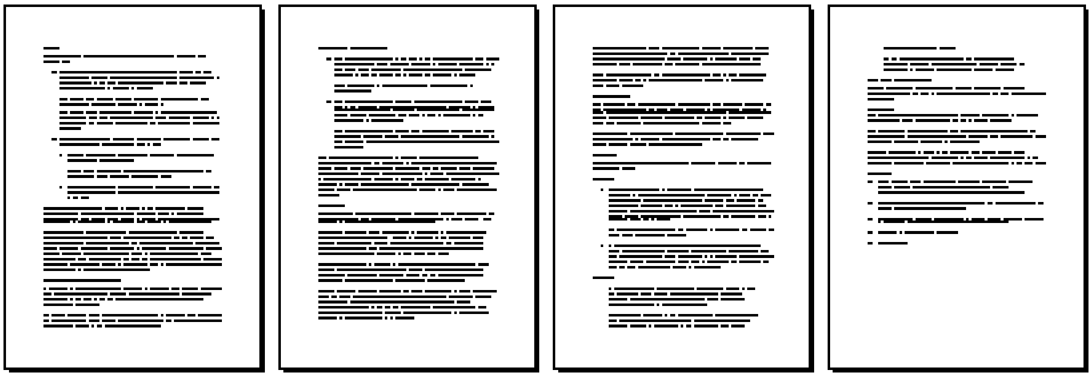

Figure 4: Print preview facility from Psion Series 3a

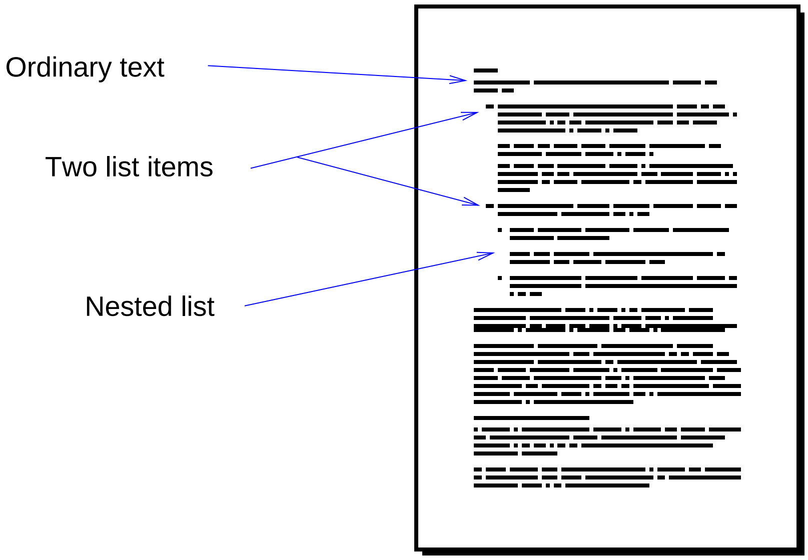

Figure 5: Enlarged view showing sub-lists

Readers are also able to identify specific parts of documents within these views. A facility to jump directly to a specified point within the document could then be offered, saving the user tedious scrolling.

4 The personal environment

There are two principle ways in which the personal environment on a PDA can be used in the Web:

1. Handling Web authentication dialogue boxes on pen-based PDAs is problematic - there is no way that the password field can be effectively masked. However, the increased security of the PDA environment allows the user to store Web authentication information within it. The PDA could then supply the information automatically.

2. Having gone to the effort of telling the PDA about oneself, it would be useful if the software were to supply this information whenever possible.

   PDAs can switch from a passive browsing mode to an active form-filling mode on receipt of an HTML form. The PDA would then attempt to complete the form using personal information extracted from its environment.

This active mode would benefit from additional attributes within HTML `input` tags.

For example:

```
<input type="text" name="lastname" value="" pinfo="surname">
```

The PDA then actively seek `pinfo` attributes, supplying the appropriate information from its environment.

HTML forms contain constructs which, if not treated with care, could obtain personal information without the user's knowledge. Both hidden and password fields effectively mask the information being supplied by the user. If such fields were filled by the PDA directly, the user would be unaware of the information being sent to the remote server. The PDA should therefore warn its user of any information being supplied in these fields.

5 Summary of modifications to HTML

We have already suggested that we introduce the concept of the sentence to HTML, and that both this and the existing paragraph tag should have a *level* attribute to identify the StretchText level.

As suggested in the previous section, a `pinfo` attribute needs to be added to the `input` element within HTML forms. Appropriate values for this attribute should be determined by the W3 Consortium.

Handwriting recognition on pen-based systems can be aided by supplying information about the acceptable characters within an input field. This could be achieved, in the first instance, by the addition of a `number` input type to complement the existing `text` type.

Discussion may well highlight other modifications to HTML forms that would be useful within the PDA environment.

6 Other issues that need to be addressed

6.1 Data sharing

One of the strengths of PDAs is their ability to communicate with other local PDAs by 'beaming'. This action enables machines in close proximity to share information, something that cannot easily be done in other environments.

It would be useful for machines within a limited range to share pages with each other (acting as local caches), rather than each having to communicate with its service provider. However, this type of sharing is fraught with problems: some pages will have to be paid for, others contain copy-written information. These are problems with caching systems in general, and will not be discussed further here.

Local communications *are* still useful, however. Beaming is an excellent means of transferring URLs (which are complex to transmit in other media) between machines.

6.2 Value added service providers

Service providers dealing with mobile computers will provide additional services to entice customers to use their facilities. They will store caches of common documents to save general download time [Smi94]. These caches have the effect of reducing the overall network traffic, which is beneficial for *all* network users - not only those using PDAs. In addition, the service providers may subsidise mobile communications by charging local land-line customers for the use of the cache.

The service provider is also in an ideal position to generate overview meta-information for documents where it has not been supplied by the primary information provider.

7 Conclusions

In this paper we have demonstrated that many of the 'problems' associated with PDAs do not affect the use of these machines as Web browsers. We have identified screen size as a limiting factor, and have proposed a number of ways to present information in spite of this:

- StretchText enhancements to HTML paragraphs and the introduction of sentences enable authors to provide different levels of abstraction.
- Folding of HTML constructs, especially headings and lists, enables readers to concentrate on specific parts of the document (although further research into the display of nested lists is required).
- Presenting mini-overviews gives extended views of the document. These views are not as limited by the screen dimensions, and increase the effective screen size.

We have proposed the generation and use of document overviews based on meta-information extracted by document management systems. Byte ranges will allow specific parts of documents to be expanded from these overviews.

The use of personal information held within PDAs was highlighted, both for supplying Web authentication and for actively supplying information in HTML forms. Extensions to the HTML `input` element were proposed.

In addition to providing a useful PDA Web interface, the ideas presented in this paper are equally applicable to desktop software, and would enhance the usability of existing applications.

8 Acknowledgements

Our many thanks to Peter Brown[2] and Duncan Langford[3] for their comments and suggestions in the preparation of this paper.

References

[Bar95] Joel F. Bartlett. Experience with a wireless world-wide web client. Technical report, Digital Western Research Laboratory, 1995. Technical Note TN-46 http://www.research.digital.com/wrl/techreports/abstracts/TN-46.html.

[BLCL+94] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, and A. Secret. The world-wide web. *Communications of the ACM*, 37(8):76–82, 1994.

[BLFFN95] T. Berners-Lee, R. T. Fielding, and H Frystyk Nielsen. Hypertext transfer protocol – HTTP/1.0. Internet Draft, March 1995. http://www.w3.org/hypertext/WWW/Protocols/HTTP1.0/draft-ietf-http-spec.html.

[Bro86] P.J. Brown. Dynamic documentation. *Software-Practice and Experience*, 16(3):291–299, March 1986.

[GK94] Stefan Gessler and Andreas Kotulla. PDAs as mobile WWW browsers. In *The Second International WWW Conference '94: Mosaic and the Web*, Chicago, USA, October 1994. http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/gessler/www_pda.html.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 2 edition, 1994.

[LFBL] Ari Luotonen, Henrik Frystyk, and Tim Berners-Lee. *CERN httpd*. CERN, c/o W3 Consortium. http://www.w3.org/hypertext/WWW/Daemon/Status.html.

[MGR] Lou Montulli, Michael Grobe, and Charles Rezac. About lynx. HTML document. http://kufacts.cc.ukans.edu/about_lynx/about_lynx.html.

[Nel87] Ted Nelson. *Computer Lib/Dream Machines*. Microsoft Press, Washington, 16011 NE 36th Way, Box 97017 Redmon, Washington 98073-9717, 1987.

[Rag] Dave Raggett. Hypertext markup language specification version 3.0. Intenet Draft. http://www.w3.org/hypertext/WWW/MarkUp/html3/CoverPage.html.

[Smi94] Neil Smith. What can archives offer the world wide web. In R. Cailliau, O. Nierstrasz, and M. Ruggier, editors, *First International World-Wide Web Conference, Advance Proceedings*, pages 101–112, Geneva, Switzerland, May 1994. http://www.hensa.ac.uk/ftp/pub/misc/www94/www.ps.

[2] <URL:http://www.ukc.ac.uk/computer_science/Html/pbrown.html>
[3] <URL:http://www.ukc.ac.uk/computer_science/Html/Langford/dl1.html>