AWeb

AWeb II-2.0, July 1996

The World Wide Web Browser for the Amiga computer

distribution by AmiTrix Development

Contents

Chapter 1

Introduction

1.1 Introduction

AWeb is a World Wide Web browser for the Amiga computer. It offers the following features:

- AWeb doesn't use Magical User Interface (MUI) (see section 1.1.1). It uses BOOPSI classes instead, which results in less memory usage and increased speed. The BOOPSI classes are partially custom designed for AWeb, and partially from the ClassAct kit (see section 1.1.2).
- AWeb can use a wide range of TCP-stacks: AmiTCP/IP, I-Net225, AS-225 or compatible. Without TCP stack running, you can still view local files.
- AWeb uses extensive internal multitasking, which leads to total asynchroneous and parallel network access. Images are starting to load while the document is still loading. It is possible to follow a link while the previous document is still loading. A separate network status window shows all pending network and local file accesses. All network accesses can be interrupted *immediately*.
- The HTML-2 standard is fully supported, including forms. Some of the most important so-called HTML-3 extensions are also supported. For buggy pages not conforming to the standard, AWeb offers a compatibile mode.
- AWeb supports the use of proxies for HTTP, FTP, Gopher and Telnet. Because some proxies can't handle forms or other special cases, the use of proxies can be temporarily disabled from the menu.
- The HTTP and Gopher protocols are supported internally. By using external programs, FTP and Mailto can be used too. HTTP user authorization is supported.

- AWeb lets you load all images, delay all image loading, or load only clickable maps and delay other images. For delayed images, the ALT text is taken into account. Transparent GIFs are supported. The 24-bit picture datatype (picturedtV43) is supported.
- To improve network speed, host names and network addresses are cached so addresses are looked up only once during a session.
- AWeb can open it's windows on the default public screen, on a named public screen or open its own screen.
- AWeb has a hierarchical hotlist, with options to group entries. In extension to its own hotlist, AWeb can read other hotlists, like those of AMosaic.
- An advanced settings requester is integrated in the program. You can control many aspects of the program in this requester.
- AWeb has an ARexx interface, and a unique and powerful shell command interface.

1.1.1 Why not use MUI ?

I must admit that MUI is a very clever piece of programming. And basically, I like the idea of a fully user-configureable GUI very much. But in my opinion, visual feedback and intuitivity are more important features of a GUI than configurability. And it is in these areas that MUI scores very badly.

Visual feedback

Intuition and BOOPSI gadgets give immediate visual feedback when the user plays with them. This is due to the fact that the gadget imagery update is done by the input task, not by the application. The only circumstance in which there is no immediate visual feedback, is when some program has switched off multitasking, but that's not a normal condition. With MUI, all feedback is done in the application's context. This effectively means cooperative multitasking instead of preemptive multitasking. And this cooperative "multitasking" is exactly why most Amiga owners dislike MS-Windows. For the very same reason many Amiga owners (me included) don't like MUI.

To understand the consequences, imagine an application that is busy performing a task that takes a considerable amount of time. Meanwhile, it offers the user an interface to affect the current task (a common example is a cancel button). It checks the user input every second. Or imagine an application busy performing a short task, taking less than a second. The GUI isn't disabled, allowing the user to select another GUI element during this short time. This user action is than queued for a fraction of a second until the application is ready to process it. This is of course the way most applications work.

Both are examples of totally acceptable application response, providing that there is immediate visual feedback for the user action. There is a difference between the visual feedback of a user action, and the application response to that action. The former must be immediate, while the latter in general may take a short time, up to a second or so, without confusing the user. (Depending of the kind of application, of course.) MUI applications treat visual feedback as being an application response, thereby failing miserably.

It should be obvious that blocking user input entirely and showing a busy pointer, like some people suggest, is not an option at all in the above examples.

I realize some gadtools gadgets have the same behaveour as MUI gadgets, and that's one of the reasons why I don't like gadtools very much, either. But at least gadtools *buttons* give immediate feedback.

Intuitivity

Gadtools offers a cycle gadget. Click anywhere in the gadget, and the next option is selected. You may like or may not like this kind of gadget, but the behaveour exists and is part of the Amiga look and feel.

MUI offers a gadget that looks exactly the same, but behaves differently. Click in the text part of the gadget, and nothing happens. Just a quick flash of a pop-up menu. I have clicked many times on gadgets of this kind and wondered why nothing changes before I realized it was a MUI application so this kind of gadget behaves differently.

The most important rule in GUI design is: be consistent. Gadgets that look the same must behave the same. Gadgets that behave differently must look different. MUI fails to be consistent with the Amiga OS look and feel.

I am aware that there are commodities that change the behaveour of the Gadtools cycle gadget into a pop-up menu. Basically these commodities suffer from the same fault: it changes the behaveour of the gadget but not its appearance.

More intuitivity

Fortunately, MUI 3.x has left the silly behaveour of configuring all your MUI applications from one preference program. But still, configuring your MUI application can be very difficult and not obvious for the casual MUI user. The best example of this is the ever returning question in many Amiga newsgroups: HOW DO I GET AMOSAIC TO RUN ON ITS OWN SCREEN? I believe that the 18 (eighteen!) steps to perform to accomplish this, are far too many.

And even if the way configuring MUI aspects of an application is improved, there are still many MUI applications offering *two* settings requesters - one for the MUI aspects, and one for the application's own parameters. This is also confusing. For the user, a system like MUI should be transparent. The user should not need to know which aspect is controlled by MUI, and which by the application itself.

Minor objections

- Size and speed
  Many people complain that MUI is big and slow. Many other people say you can't expect a 1 MB, 8 MHz 68000 machine to run modern applications, and the things MUI offers is well worth the extra resource cost.

  Fact is, MUI takes up memory and CPU time, which makes it bigger and slower than no MUI at all. If you don't give much about fancy looking gadgets, MUI rapidly becomes *too* big and *too* slow.

- Shareware
  MUI is shareware. And MUI applications *need* MUI, it is not an option. I want the user to have a choice which extras he or she wants to buy.

Flames?

Don't flame me about this. I won't read it. If you are a MUI lover, well, there are at least three MUI based browsers available. Use one of them instead of AWeb.

1.1.2 Classact

AWeb uses the ClassAct GUI toolkit. ClassAct is a set of BOOPSI gadget and image classes, designed to speed up and simplify the development of GUI applications. Free to users of the applications, ClassAct is available as shareware and commercial licences for Amiga developers. The licence includes the development kit, support, and distribution rights to the shared ClassAct class libraries. For more information, refer to *http://www.warped.com/~timmer/classact.html* or *http://www.nai.net/~caldi/* or send mail to caldi@nai.net.

1.2 Legal issues

1.2.1 Copyright

1.2.2 Disclaimer

This software is provided "as is". No warranties are made, either expressed or implied, with respect to reliability, quality, performance, or operation of this software. The use of this program is at your own risk. Yvon Rozijn and AmiTrix Development assume no responsibility or liability for any damage or losses resulting from the use of this software, even if advised of the possibility of such damage or loss.

1.2.3 Licence

This licence does not apply to parts of the distribution not covered by the AWeb copyright. For the licence of these parts, refer to the respective documentation.

1.2.4 Distribution

The AWeb-II package, containing the AWeb browser, is distributed by AmiTrix Development.

Distribution otherwise than via AmiTrix Development is not allowed without prior written permission from both the author and AmiTrix Development.

Chapter 2

Before you begin

2.1 System requirements

AWeb needs the following to run:

- OS 3.0 or better
- At least 2 MB of memory, preferably more
- Either AmiTCP/IP, I-Net225 or AS-225 to access the World Wide Web.
- Appropriate datatypes to view inlined images. At least a GIF and a JPEG datatype are needed to view most of the images on the World Wide Web.
- Some classes from the ClassAct kit (included).

2.2 Installation

2.2.1 Installing AWeb

Installing AWeb is straightforward. Just double-click the Install icon.

The archive contains three different Workbench icons for AWeb: for the standard Workbench, for MagicWB, and for NewIcons. The installation process will ask you which you want to install. The other icons are saved in a separate drawer so you can switch icons later.

AWeb needs some ClassAct classes. These are included in the archive, and the installation process will ask you if you want to install them. You are strongly encouraged to do so. If you do not install ClassAct, you have to make sure that AWeb can find its classes, for instance by executing the MakeAssign script before you start AWeb. If you *do* install ClassAct, you don't have to worry about this.

If you haven't saved your settings from within AWeb, the installation procedure will also ask you some questions to be able to install one of the predefined configurations. Note that you can always change the configuration afterwards, using the settings requester.

- First, it asks if you have 2 MB of memory in your Amiga, or more. If you have only 2 MB of memory, then a different setup is needed to get the most out of AWeb. Have look at these tips to see how you should set up AWeb for a 2 MB Amiga.
- Also, you will be asked if you prefer small fonts or large fonts. The large fonts option will make AWeb look as closely as possible to the standard Netscape setup, using only standard Workbench fonts. Although the HTML standard doesn't impose specific fonts in any way, many pages on the World Wide Web are designed to look best using these fonts. If you plan to use AWeb on a small screen (less than 640 x 400), you might want to use the small fonts, or else the window will contain very little text. **Note:** The *large fonts* settings makes use of the *CGTimes* scalable font found on the Workbench Fonts diskette. Make sure you have this font properly installed if you choose *large fonts*.

2.2.2 Configuring the JFIF datatype

If you use the JFIF datatype (by Christoph Feck, TowerSystems), then please read this:

The JFIF datatype doesn't seem to handle shareable pens on public screens correctly under all circumstances. You have to install AWeb in the datatype or else AWeb will not be able to show inline JPEG images.

In the JFIF preference editor, you must add an application named **AWebIP** (AWeb Image Processing), and then select *Single-Pass Quantization* (in the GadTools version of the preference editor) or *One-pass* (in the MUI version).

2.3 Tips for 2MB Amiga users

You can use AWeb on a 2MB Amiga, if you configure it properly. Here are some suggestions:

- Use AWeb on the default public screen.
- Set image loading *Off*, and only load images you really want to see by clicking the icons.
- Set maximum number of network connections to 1. Every connection takes up precious memory.
- Set your temporary path to a directory on your hard disk, not in RAM.
- Set your cache as follows:
  - document memory: about 40% of the amount of free memory when AWeb and your TCP stack are running
  - document disk: as large as you like and have room on your hard disk
  - image memory: the same as *document memory*
  - image disk: as large as you like and have room on your hard disk

If you follow these steps, websurfing shouldn't be a problem with only 2MB. Of course, it will be more comfortable with more memory. Then you can run AWeb on its own screen, using more colours, using a bigger cache, etc.

Chapter 3

Working with AWeb

3.1 Starting AWeb

3.1.1 From the Workbench

You can start AWeb by a double-click on its icon.

AWeb can be specified as *default tool* in a project icon. You can use extended selection (shift-click) to select one or more project icons. AWeb will load the projects selected as local documents.

In addition, AWeb supports the following *tool types*:

URL=*url_or_filename*
: Specify this tool type one or more times to open these documents when AWeb is started. If you specify a local filename, use the LOCAL tool type too.

LOCAL
: If this tool type is present, the names in the URL tool types will be interpreted as local file names, rather than network URLs. This tool type is not needed if you select documents by their project icon, as mentioned above.

CONFIG=*settings_filename*
: Use this file as settings file instead of the default, AWeb.prefs. If the file doesn't exist, some default settings are used. Saving the settings will save to this file name.

HOTLIST=*hotlist_filename*
: Use this file as AWeb's hotlist instead of the default, AWeb.hotlist. If the file doesn't exist, it will be created the first time you add an entry to the hotlist.

3.1.2 From the Shell

You can start AWeb from the Shell (or the CLI).

Format: **AWeb** [*url_or_filename*]**...** [**LOCAL**] [**CONFIG** *settings_filename*] [**HOTLIST** *hotlist_filename*]

Template: **URL/M,LOCAL/S,CONFIG/K,HOTLIST/K**

The documents in the URL argument are loaded when AWeb starts. If the LOCAL argument is present, the names will be interpreted as local file names, rather than network URLs.

The file in the CONFIG argument will be used as settings file instead of the default, AWeb.prefs. If the file doesn't exist, some default settings are used. Saving the settings will save to this file name.

The file in the HOTLIST argument will be used as AWeb's hotlist instead of the default, AWeb.hotlist. If the file doesn't exist, it will be created the first time you add an entry to the hotlist.

3.2 The Graphical User Interface

3.2.1 Overview

In figure 3.1 an overview of the graphical user interface is presented. The items depicted in this figure are explained in sections 3.2.2 through 3.2.13.

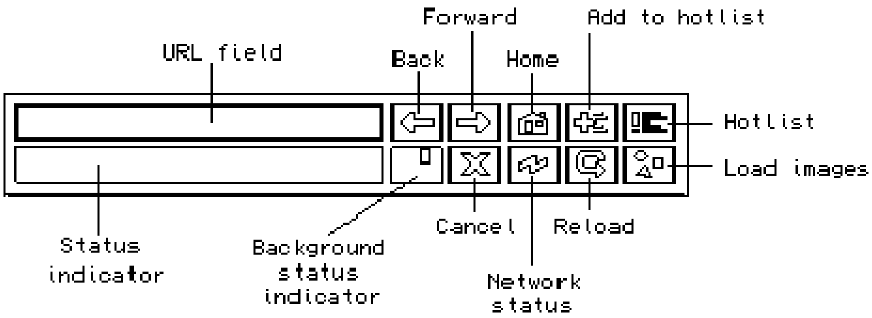

Figure 3.1: An overview of the graphical user interface.

3.2.2 URL field

This field shows the URL (network address) of the currently shown page. You can enter a new URL here, then ENTER will retrieve the page for that URL.

The `Project / Open URL` menu function or its shortcut, **AU**[1], will clear this field and activate it, so you can type the new URL right away.

[1] **A** is used throughout this document to indicate the right amiga key.

The `Project / Open WWW` menu function or its shortcut, **A**`W`, will activate this field and preload it with "http://www." for even more convenience.

3.2.3 Status indicator

This field serves two purposes.

First, when browsing through a page, it shows the URL "behind" the link currently pointed to with the mouse. That is, when you click this URL will be retrieved.

Second, when a page is being loaded for this window, is shows the current state. If actual data is retrieved, a progress bar will appear showing how far the load process is. The progress bar will not appear if the final size of the document is not known on forehand.

3.2.4 Background status indicator

When one or more load operations are in progress, this indicator will show a little square. On every connection made, and on every block retrieved, the square will advance one step.

This indicator lets you know if there are still things loading in the background, and how rapidly they progress. If you want more detail, the network status window will tell you everything.

3.2.5 Back button

This button lets you walk back through the window history. The window history contains all pages viewed before *in this window*.

The `Navigate / Back` menu function, or its shortcut, **A**`B`, or the **Alt + cursor left** key combination, will do the same.

3.2.6 Forward button

This button lets you walk forward through the window history. The window history contains all pages viewed before *in this window*. Note that the window history is currently linear, without branches. That means that if you walk back a few steps, then retrieve another page, the window history "after" that page will be forgotten.

The `Navigate / Forward` menu function, or its shortcut, **AF**, or the **Alt + cursor right** key combination, will do the same.

3.2.7 Home button

This button retrieves the URL that is configured as your home page.

The `Navigate / Home document` menu function, or its shortcut, **AD**, will do the same.

3.2.8 Add to hotlist button

This button adds the current document to your hotlist.

The `Hotlist / Add to hotlist` menu function, or its shortcut, **AA**, will do the same.

3.2.9 Hotlist button

This button shows your hotlist.

The `Hotlist / Show hotlist` menu function, or its shortcut, **AH**, will do the same.

You can configure AWeb so that this button opens the hotlist maintenance window instead.

3.2.10 Cancel button

This button will interrupt (cancel) the load of a page in this window. Background loads cannot be cancelled by this button, use the cancel button in the network status window instead.

The `Control / Cancel load` menu function, or its shortcut, **AX**, and the **Esc** key, will do the same.

3.2.11 Network status button

This button will open the network status window, or bring it to front if it is already open.

The `Control / Network status` menu function, or its shortcut, **A?**, will do the same.

3.2.12 Reload button

This button will reload the current document. The page is deleted from the cache, and retrieved again.

The `Control / Reload current` menu function will do the same.

3.2.13 Load images button

This button will load all unloaded images in the current page.

The `Control / Load images now / All images` menu function, or its shortcut, **AI**, will do the same.

3.3 The menus

3.3.1 Project menu

The *Project menu* offers functions to open or close windows, fetch or save documents, or quit AWeb.

New window
Open a new window. Only available in the registered version.

Close window
Close the current window. Only available in the registered version.

Open URL
Clear the URL field () and activate it so you can type a new URL.

Open WWW
Preset the URL field () with "http://www." and activate it for maximum convenience if you want to load a WWW page.

Open local...
Opens a standard file requester. After you select a HTML file, the file will be loaded in the current window.

Search engines
Opens the local page extras/search.html which contains a quick interface to several search engines on the Internet.

View source...
Show the HTML source of the current page, using the viewer program that was installed as HTML source viewer.

Save source...
Opens a standard file requester. After you type a file name or select a file, the HTML source of the current page is saved. If the selected file already exists, you have the choice to:

- overwrite the old file,
- append the source to the old file,
- select another name, or
- cancel the save alltogether.

About...
Opens a window with version information. If the current window has an ARexx port, the name is shown here.

Quit
Quit AWeb after confirmation. If you confirm, all pending network operations are cancelled.

3.3.2 Control menu

The *Control menu* offers functions to control the operation of AWeb, other than cache functions.

Load images now
Initiates the load of images in the current document. This menu item has 2 sub-items:

All images
Initiates the load of all images in the current document that aren't loaded yet. Pressing the Load Images button will do the same.

Maps only
Initiates the load of all clickable maps in the current document that aren't loaded already.

Network status...
Open the network status window, or bring it up to front if it is already open. Pressing the Network Status button will do the same.

Reload current
Reload the current document. The page is deleted from the cache, and retrieved again. Pressing the Reload button will do the same.

Cancel load
Interrupt (cancel) the load of a page in this window. Background loads cannot be cancelled by this menu function, use the cancel button in the network status window instead. Pressing the Cancel button, or pressing the **Esc** key will do the same.

Next window
Activates and brings up to front the next window. Only available in the registered version.

Previous window
Activates and brings up to front the previous window. Only available in the registered version.

HTML mode
Select the way HTML should be interpreted in this window. This menu item has 2 sub-items:

Strict
Let AWeb follow the HTML standard.

Compatible
Let AWeb be compatible with other browsers that are buggy. Useful for viewing a page that was composed with such a buggy browser.

Disable proxy?
If you have proxies configured, you can select this menu item to toggle the use of proxies on and off. This feature is included because some proxies tend to handle forms and authorized requests incorrectly.

3.3.3 Cache menu

The *Cache menu* offers functions to flush the cache, and to save or flush the cached authorizations.

Flush images in current
Inlined images that appear in the current document are deleted from both the memory and disk caches. Note that if the image also appears in another document, it is undisplayed in that document, too.

Flush old images
Inlined images that do *not* appear in any currently displayed document are deleted from both the memory and disk caches.

Flush all images
The image cache is cleared completely. After this function, no images are left in the cache.

Flush documents
All documents that are not currently displayed are deleted from the cache.

Save authorizations
Save the current authorization details.

Flush authorizations
Forget all the current authorization details. Note that this flushes the internal authorizations cache only. To flush the disk cache too, you have to select `Cache / Save authorizations` afterwards.

3.3.4 Navigate menu

The *Navigate menu* offers functions to navigate in the document history.

Back
Walk back one document through the window history. The window history contains all pages viewed before *in this window*. Pressing the Back button, or using the **Alt + cursor left** key combination, will do the same.

Forward
Walk forward one document through the window history. The window history contains all pages viewed before *in this window*. Note that the window history is currently linear, without branches. That means that if you walk back a few steps, then retrieve another page, the window history "after" that page will be forgotten. Pressing the Forward button, or using the **Alt + cursor right** key combination, will do the same.

Home document
Retrieve the URL that is configured as your home page. Pressing the Home button will do the same.

Window history
Show the window history requester (see section 3.8).

3.3.5 Hotlist menu

The *Hotlist menu* offers functions to use the hotlist, or to read foreign hotlists.

Add to hotlist
Add the current document to the end of your hotlist. Pressing the Add to hotlist button will do the same.

Show hotlist
Show the hotlist in the browser window. Pressing the Hotlist button will do the same.

Maintenance...
Opens the hotlist maintenance window. You can configure AWeb so that the Hotlist button will do the same.

Save hotlist
Save any changes made to the hotlist now. Changes will be saved when you quit AWeb automatically, but sometimes you might want to save changes earlier.

Restore last saved
Reverts the hotlist to the last saved version.

AMosaic (ARexx)
Show the ARexx-based hotlist of AMosaic 1.2. AWeb expects it to reside under the name ENV:mosaic/hotlist.html.

AMosaic (2.0)
Show the hierarchical hotlist of AMosaic 2.0 (pre-release). AWeb expects it to reside under the name ENV:mosaic/.mosaic-hotlist-default.

Other...
Opens a standard file requester, from which you can select other hierarchical hotlists. These include hotlists saved by older versions of IBrowse, and other hotlists created by AWeb.

3.3.6 Settings menu

The *Settings menu* offers functions to configure AWeb.

Image loading
This menu item provides a quick way to set the image loading. It has 3 sub-items:

- All images
- Maps only
- Off

with the same meaning as the choices in the image loading chooser.

Change settings...
Brings up the settings requester (see section 3.9).

Save settings
Save the current settings. This function will take a snapshot of your window positions first.

3.3.7 Help menu

The *Help menu* offers you more information.

Documentation
Shows the AWeb manual in this window.

AWeb home page
Retrieves the AWeb Home page. A TCP stack must be running, and you must be connected to the Internet for this function to work.

AWeb FAQ
Retrieves the AWeb FAQ. A TCP stack must be running, and you must be connected to the Internet for this function to work.

3.3.8 ARexx menu

The *ARexx menu* offers functions to start ARexx macros.

Start ARexx macro...
Opens a standard file requester. After you select an ARexx macro, that macro will be executed with the original window as default port.

User-configurable items
The remainder of this menu contains a list of ARexx macros you can configure yourself. Use the ARexx macro menu settings page for this.

3.4 The browser window

The browser window is the most important window of AWeb. It is the place where World Wide Web pages are displayed. On top of the window there are some gadgets, already explained in section 3.2.

3.4.1 Scrolling the page

You can scroll the window contents horizontally and vertically, provided the size of the document is larger than the window.

Of course you can use the scroll bars and arrow buttons to scroll, but AWeb also understands the following keys:

- **cursor up/down** (both cursor keypad and numeric keypad) scroll up and down over a short distance.
- **shift + cursor up/down** (cursor keypad) scroll up or down a page.
- **PgUp, PgDn** (numeric keypad) scroll up or down a page.
- **Space, Backspace** scroll up or down a page.
- **Home, End** (numeric keypad) jump to the beginning or the end of the document.
- **cursor left/right** (both cursor keypad and numeric keypad) scroll left and right. This is only possible if the document contains an image or preformatted text wider than the window, because otherwise AWeb will keep the text formatted to fit within the window width.
- **shift + cursor left/right** (cursor keypad) scroll left or right a full page width.

3.4.2 Following links

One of the most important features of the World Wide Web is the ability to include *hyperlinks* in documents. A hyperlink is displayed in another colour, and by default underlined. You can change the colour and the underlining in the Browser 2 page in the settings requester (see section 3.9).

Images that are also links have a frame drawn around them. If you have deselected the link underlining in the settings requester, then this frame isn't drawn.

Click the text or image to follow the link, i.e. retrieve and display the document "behind" the link. The URL (network address) of the document that is linked to, is shown in the status indicator when the mouse pointer is over the hyperlink.

3.4.3 Inlined images

A document can contain inlined images interspersed with the text. If an inlined image is not (yet) loaded, AWeb displays an icon for that image. You can select if you want images to be loaded immediately or not using the image loading chooser in the settings requester.

AWeb displays different icons under different circumstances. You can tell AWeb to use your own images here. Please refer to the on-disk documentation for the images currently in use and the explanation of these various icons. For instructions on how to use your own default images please refer to section 4.4.

3.4.4 Downloading

Instead of following a link and display the new document, or loading and displaying an inlined image, you can *download* a document or an inlined image. To do so, hold the **Shift** key while clicking the link or image icon. The document or image is retrieved, and a standard save requester will pop up to let you specify a file name.

If the document or image is already in cache, it will only be saved, not retrieved again over the network.

Note you can also save a displayed image in this way. Just press the **Shift** key and click the image. This will work even for background images (only if background images are displayed): just shift-click somewhere in the background and you will be asked for a filename to save the background image.

If the image is also a link, shift-clicking the image could be abmiguous; therefore AWeb will save the image in this case. If you want to download the document "behind" the link, you can either select `Cache / Flush images in current` from the menu, and shift-click the upper left half of the icon, or click the image to load and display the document, and then select `Project / Save as HTML` from the menu to save the document.

3.5 Starting your TCP connection

Before you can access documents from the World Wide Web, you must start your *TCP connection* with the Internet. The connection is maintained by the TCP package you are using, also known as *TCP stack*.

You can start your TCP stack yourself before you start AWeb, or while AWeb is running. To make life easier, you can even let AWeb start your TCP stack when it is needed for the first time. For example, if you are browsing through local documents, and then follow a hyperlink to some document out there on the World Wide Web, then AWeb could start the TCP stack automatically.

To use this feature, you have to configure (see section 3.9) the start script (or program) for your TCP package.

If you have configured the stop script (or program) for your TCP stack, AWeb can terminate the TCP stack automatically after you quit AWeb. Note that it does so only if AWeb has started the TCP stack before, and only after asking for your permission to do so. If you started the TCP stack yourself, AWeb will not terminate it.

AWeb will try to start the TCP stack only *once*. If AWeb has started the TCP stack, and it appears to have failed for some reason, then AWeb will never try to start it again. You have to start it manually in that case.

3.6 The network status window

3.6.1 Purpose

The Network Status Window shows all pending network and local file accesses. There are possibilities to cancel selected transfers, or all transfers.

3.6.2 Opening the window

You can open the window in three ways:

- By clicking the Network Status button.
- By selecting the `Control / Network Status...` menu item
- By using the hotkey *A* ?

3.6.3 Contents of the list

The listview contains a line for each pending or queued transfer. The line consists of the filename retrieved (without hostname and path), and the current status. Current status can be one of the following:

- **Queued** : image will be loaded when network slots become available.
- **Started** : transfer process has started.
- **Looking up** : looking up host name.
- **Connecting** : making connection.
- **Waiting** : request sent; waiting for response.
- **99999/99999** : Reading.
- **Processing** : Processing image (remapping it to the screen colors).

3.6.4 Cancel transfers

You can cancel one specific transfer by first selecting the entry in the listview, then click the button marked with one X.

You can cancel all transfers by clicking the button marked with three times X.

3.7 The hotlist

3.7.1 Adding a page

The AWeb hotlist can be used to remember the addresses of interesting pages. Whenever you find an interesting page that you think you would like to visit again in the future, you should add the page to the hotlist. Do this by pressing the Add to Hotlist button, or using the `Hotlist / Add to hotlist` menu item.

3.7.2 Using the list

Press the Hotlist button, or use the `Hotlist / Show hotlist` menu item to load a page with all links to all remembered pages.

With the *Hotlist button gives requester* setting, you can make the button pop up a requester. This is the same requester as for the `Hotlist / Maintenance...` menu item. The functions in this requester are described in the next section.

3.7.3 Maintenance

When you choose the `Hotlist / Maintenance...` menu item, a requester will open. In this requester you can change entries in the hotlist, move them around, group entries or delete them.

Use the window close gadget, or the **Esc** key to close the requester.

Change an entry

Select the entry you want to change, either with the mouse or with the cursor up and down keys. Now you can change the name of the entry and the URL that it points to.

Move an entry

Select the entry you want to move, either with the mouse or with the cursor up and down keys. Now you can move the entry up and down in the list with either the arrow up and arrow down buttons, or with the **Ctrl** + cursor up/down keys. The entry will step into open groups, but will skip closed groups.

If you move a group title, the entire group will move.

Add an entry

Use the *Add a link* button in case you want to add an entry to the list and type the name and URL yourself.

Group entries

Use the *Add a group* button to create a group, then change its name. Now you can add entries to the group, or move existing entries into it. Entries within a group will be displayed indented in the list. You can include other groups within a group, to any level.

Click on the little arrow next to the group title in the list to open or close the group. If the group is closed, its members are not visible, making the list clearer. Also, you can use the **Enter** key to open and close a group if the title is selected.

Remove an entry

Select the entry you want to remove, either with the mouse or with the cursor up and down keys. Now you can remove the entry with the *Remove* button. If you remove a group, all members will be removed too.

Follow a link

Select the entry for the document you want to load, either with the mouse or with the cursor up and down keys. Now you can load the document with either the *Follow* button or with the **Enter** key. Another way to follow a link is to double-click on it.

If you have selected the *Hotlist requester auto close* setting, the requester will close if you follow a link. Otherwise the requester stays open.

3.7.4 Save and restore

If the hotlist was changed, it will be saved automatically when you leave AWeb. You can save it earlier with the `Hotlist / Save hotlist` menu item.

Use the `Hotlist / Restore last saved` menu item to revert the hotlist to the state it was in when it was last saved.

3.7.5 Using other hotlists

You can specify a different hotlist to use with the HOTLIST tooltype or argument.

You can load another AWeb hotlist using the `Hotlist / Other...` menu item. The hotlist will be shown as document, not in the hotlist maintenance requester.

3.8 The history window

3.8.1 Purpose

In the history window, you can check all the pages you have visited in the current AWeb session. You can select old pages to display again. The history window offers filtering and ordering options.

3.8.2 Opening the window

Select `Navigate / Window history` from the menu to open the history window. The display will be filtered automatically for the window in which you selected the menu item.

If the history window is already open, it will pop up to front, and will be activated. Also, the selected window will change to the window for which you opened the history.

3.8.3 The history list

In the list, all visited pages are shown.

The **first column** contains the *window number*. This is the same number as appears in the title bar of the window.

The **second column** contains the *mainline indicator*. This is a little arrow. Suppose you retrieve pages **A** - **B** - **C** - **D**, then go back to **C**, back to **B**, and then retrieve page **E**. In this case the mainline is **A** - **B** - **E**. These are the pages that the back and forward buttons will walk along.

The **third column** contains the *title* of the page, or the URL if no title is known.

The **fourth column** contains the *cache indicator*. If the 'in-cache' symbol is present, the page is still in AWeb's cache. Going back to this page won't need any network access.

If you selected *natural* or *mainline* order, the current document will be highlighted. The two fields below the list show the title and URL for the highlighted page.

3.8.4 Redisplay a page

Select the page you want to see again from the list. You can click on the entry, or use the arrow keys to move the highlight. Then click the *display* button, or hit the **Enter** key.

Another way to redisplay a page is to doubleclick on the entry.

If you have the History window auto close setting selected, the window will close automatically.

3.8.5 Filtering

If the *Filter* checkbox is selected, the list will only show entries for the window number selected in the *window* field.

If you deselect the *Filter* checkbox, the list will show entries for all windows.

3.8.6 Ordering

With the *Order* chooser, you can select how the entries in the list should be ordered.

Natural
: Natural ordering shows all displayed pages in the order you have displayed

them. If you go back a few pages, then go to another page, the latest page is added at the end of the list. If you redisplay a page that was displayed before, it is also added again to the end of the list.

Mainline
The list shows only the pages on the mainline. That is, entries for which the *mainline indicator* is off will not be shown.

Retrieved
The list contains the pages in the order as they were retrieved for the first time. The list will show every page only once.

Title
The entries in the list are sorted by their title.

URL
The entries in the list are sorted by their URL (network address). The list will show URLs instead of titles if you select this ordering.

3.9 Settings

3.9.1 Purpose

In the settings requester, you can change many aspects of AWeb. The changed settings can be saved, only used for this session, or cancelled.

3.9.2 Opening the Settings Requester

The settings requester is opened by selecting the `Settings / Change Settings...` menu item.

3.9.3 Controlling the settings requester

Because there are far too many parameters to change to fit in one window, the settings requester is organized in pages. You can select a page from the *chooser* gadget in the upper part of the window.

A click in the wide part of the chooser gadget pops up a menu from which you can choose a page. A click in the smallest (right) part switches to the next page; holding the SHIFT key while you click switches to the previous page. The pages the requester offers are discussed in section 3.9.6 to 3.9.18.

3.9.4 Command and arguments

On several locations within the settings requester, you can enter some sort of external command. For convenience, there are separate string gadgets for the command itself and its arguments.

In the *command* string gadget, you can type in a command. It is advised that you specify the full path, or else AWeb might not be able to execute the command. You can click on the button to pop up a file requester, so you can easily pick the command.

In the *arguments* string you specify the arguments for the command. Any text you type here will be passed to the external program, but the first two or three **%s** instances will be replaced by parameters from AWeb. These parameters differ for each command, but you can click the 'arrow' gadget to pop up a short descriptive list.

Instead of a program, you can specify a DOS script. If you do, make sure that the *script* bit for this file is set. You can set this bit either by the DOS command
protect *script_name* +s
or by the Workbench Information program (select the *Script* checkbox).

Note that an external program setting will *only* be recognized if *both* fields, command *and* arguments, are supplied.

3.9.5 Closing the requester

The bottom region of the requester contains three buttons:

Save
Apply all changes and save the settings. The next time you start AWeb the same settings will be used.

Use
Apply all changes for this session only. Unless you save the settings later using the `Settings / Save Settings` menu item, they will be forgotten the next time you start AWeb.

Cancel
Don't apply changes.

3.9.6 Browser 1: Fonts

On this settings page, you can change the font and style AWeb should use for different types of text.

The first column in the list contains the HTML tag for the type of text. The second column contains the current font and style selected. Note that *Normal* is not a HTML tag, but merely denotes normal text that is not subject to text style tags.

A brief description of the meaning of the HTML tag in the selected row is displayed below the list.

The Ff button pops up a standard font requester, where you can change the font, the size and the style.

3.9.7 Browser 2: Colours

On this settings page, you can change the default colours to use.

Change the default colours

To change the colour of a link, select the appropriate row in the list. There are different colours for:

New link
This colour is used for links to pages that you haven't visited yet.

Visited link
Links to pages you have visited before are shown in this colour.

Selected link
This is the highlight colour a link will have when you click it.

Background
The background colour of a page without a background colour or background image.

Text
The default colour of normal text on a page.

Then press the **Change colour** button, this will pop up a colour requester. Note that this colour requester,as opposed to the Palette colour requester (section 3.9.10), does *not* change the screen's palette. Instead, it picks the colour from the available palette that fits best to the selected colour values. Internally, the 24 bit colour values are stored, and for every screen AWeb opens on the best fitting colours are determined.

These settings are ignored if a page defines its own colours.

Use screen text and background colours

Suppose you don't want any special colour for your background and text, but just the normal screen colours. Instead of trying to get the palette exactly right to match the screen colours, you can simply select this checkbox. Then the palette settings for browser background and text are ignored, and those of the screen are used (unless the page defines its own colours).

3.9.8 Browser 3: Options

On this settings page, you can change several browser options.

HTML mode

Use this chooser to select the HTML mode. It is recommended that you leave this set to *tolerant* unless you encounter problems with a page. More information on the HTML mode can be found in section 4.1.

Change the underlining

The checkbox determines whether links should be displayed underlined or not. If underlined is selected, *new links* are underlined with a solid line, and *visited links* will have a dashed line.

If this checkbox is selected, images that are links have a border in the appropriate colour.

Change the cycle field

Many people use a commodity that turns cycle gadgets into popup menus. In section 1.1.1 you can read why this is not a good idea. Because a cycle gadget certainly has its drawbacks, AWeb offers the possibility to display a cycle field in a form as a list. Note that selection fields with more than 5 selections, or with multiple selections, are always turned into a list.

If this checkbox is selected, all selection fields are displayed as lists, even those with less than 5 selections.

Use fancy backgrounds

Many pages contain background images or background colours. AWeb will display these only if this checkbox is checked. If it is unchecked, AWeb will use only the colours defined in the Browser 2: Colours settings page.

3.9.9 Screen 1: Screen

On this settings page, you can specify on which screen AWeb should open its windows.

Using the chooser, you can make AWeb to open on the default public screen, a named public screen, or let AWeb open its own public screen.

Default public screen

If this is selected, AWeb will open its windows on the default public screen. Usually this is the Workbench screen.

Named public screen

AWeb will open its windows on a public screen that is not necessarily the default. You can enter the screen name you want AWeb to open on.

If the screen doesn't exist when AWeb starts, the default public screen is used instead.

Own public screen

Using this selection, AWeb will open its own public screen. The name of this screen is **AWeb**.

Clicking the 'monitor' button will pop up a standard screen mode requester. Here you can select a screen mode, width, height and number of colours.

If the **Load spread palette** checkbox is selected, AWeb will load a palette for the screen. This palette contains an even spread of colours, that will be used by all images. Using this feature avoids distortion of the palette that happens if one image takes up all colours, leaving no reasonable colours for next images.

AWeb will never load such a palette for screens with more than 256 colours. Those screens will be on a graphics card that is assumed to accomodate "enough" colours for all possible images.

Please be aware that 256 colours (or less) is not many. Although AWeb tries to calculate an evenly spread palette, still many images will look suboptimal. If you don't load the palette, the first few images look great, but later images may look even worse.

3.9.10 Screen 2: Palette

On this settings page, you can change the colour settings for the AWeb own public screen, pretty much like the Workbench Palette preferences program does for the Workbench screen.

The gadgets on this settings page are only enabled if AWeb runs on its own screen. When using another screen, you should use the colour settings method provided by the owner of that screen.

The palette settings are only effective when AWeb opens its own screen. When using another screen, it respects the settings for that screen.

This settings page has two major parts.

Pen settings

The list, and the upper palette button row, specify the screen pen settings, comparable to the right-hand part of the Workbench Palette program. To change a pen, first select the pen from the list, then select the color from the palette.

Change palette

The lower palette button row allows you to change the actual color of the pens. This is comparable to the left-hand part of the Workbench Palette program. Press the **Change colour** button to pop up a colour requester for the selected colour from the palette.

3.9.11 Network 1: General

On this settings page, you can change some general settings regarding the network access.

Image loading

This chooser lets you select if you want AWeb to load images:

All images
: AWeb will start loading every image as soon as it is encountered in a page. This could consume a lot of bandwidth, especially if you are using a slow connection.

Maps only
: This option doesn't load all images, but only clickable maps. This can save a lot of traffic, and still lets you use clickable maps, as these are often essential navigation tools.

Off
: AWeb will never load images automatically. If you want to see an image, you have to click on its icon.

Max. network connections

AWeb is capable of handling an unlimited number of parallel network connections. You might want to limit this number to a reasonable maximum, to avoid overloading your network line.

Note that if the maximum is reached, *image* loading will be queued, but not *document*. A document will always be loaded immediately, so the actual number of connections could be somewhat higher if you are retrieving a page.

The status of all connections can be viewed in the network status window.

Home page

This is the URL of your *home page* as far as AWeb is concerned. It doesn't need to be the same as your home page on the WWW. It is merely the page that will be shown if you select `Navigate / Home document` from the menu, or click the Home button.

You can type in the address, or click the = button to copy the current URL from your first (or only) browser window.

Local index

If the name of a local file requested ends in a slash (/), this name is appended to the file name. This allows setting up your own WWW pages locally without having to change the names.

Start with homepage

If this checkbox is checked, AWeb will retrieve the page defined as your home page immediately when you start the program. If this option is not checked, you will initially get an empty window.

Browse anonymously

Normally, if you follow a hyperlink, AWeb sends the address of the page that link appeared on to the server. This allows servers to generate lists of back-links for interest, logging, optimized caching, tracing obsolete or mistyped links, etc. Some servers, like chat servers, actually need this information to let you get the page.

Because the source of a link may be private information, or may reveal an otherwise private information source, AWeb allows you to disable this feature. If the checkbox is selected, AWeb doesn't send this address, and you are browsing anonymously.

Ignore server MIME type

As explained in the MIME types and external viewers section, most servers send the MIME type along with the data. Unfortunately, some servers send the wrong MIME type if they fail to recognize the true type of the data. This may lead to a file being saved instead of passed to an external viewer as expected.

In those cases, it is best to ignore the MIME type as reported by the server, and let AWeb identify the MIME type by the file name extension.

If this checkbox is selected, the MIME type as reported by the server is ignored.

3.9.12 Network 2: Proxy

On this settings page, you can configure proxy servers.

Proxy servers

A proxy server is a special server, that acts as a gateway between your computer and the Internet. Instead of having to establish a connection to a server possibly on the other end of the world, a browser only connects to the proxy. The proxy server has a cache of the most popular pages, so there is a chance the page you requested is already there. If not, the proxy server retrieves the document for you. This will decrease the traffic on the network, thus speeding up websurfing.

Sometimes, the proxy server is the *only* way to connect to your provider, thereby acting as a firewall.

Configuring a proxy

First, choose the protocol you want to define the proxy for. Use the chooser for this, it has 4 options:

- HTTP
- FTP
- Gopher
- Telnet

Then type in the address of the proxy in the *proxy* field. Make sure it is in one of these two forms: **http://proxy.foo.bar** or **http://proxy.foo.bar:8080**, where the name and port number may differ, of course.

If the address doesn't start with "http://", AWeb will prepend this to the address.

Limited proxy usage

Some proxies can't handle submitting forms with METHOD=POST, or authorized pages correctly. If you have problems with such pages, try selecting this checkbox. It makes AWeb handle these pages directly, without going through the proxy.

3.9.13 Network 3: External programs

On this settings page, you can configure some external programs for network accesses that AWeb doesn't support directly yet. Also, you can configure the commands that AWeb should use to start and stop your *TCP stack*.

Protocol "plug-ins"

Of all possible internet protocols, AWeb understands only HTTP: and Gopher: by itself. There are browsers available (mostly for the PC, but some for the Amiga) that handle other protocols internally, like FTP:, mailto: and news:. However, handling these protocols internally would make the executable bigger, and will never offer the same ease of use as dedicated software. Therefore AWeb offers the possibility to configure external programs that should be started when you follow a hyperlink using one of these protocols.

Use the chooser to select from one of the protocols listed below. Then use the Command and Arguments fields to specify your plug-in command and arguments.

You can configure plug-ins for these protocols:

mailto:
A *mailto:* address is for sending e-mail. If your mail reader supports

command line arguments to send a mail, you can use this feature. An example is the *Voodoo* mail reader (by Osma Ahvenlampi), that can be used like this: **Voodoo MAIL TO someone@foo.bar**

If you don't use such a mailer, you can use a script that calls your editor, and then a mail post program. An example script is included in the on-disk documentation.

Argument parameters are:
first %s = e-mail address to send the mail to.
second %s = screen name that AWeb is running on, in case your mail program supports opening on a public screen.

ftp:

An *ftp:* address is for retrieving files via the FTP protocol. If you own a FTP client that supports command line arguments, you can use this feature. A simple example is the *ncftp* program that comes with the AmiTCP/IP package.

Argument parameters are:
first %s = host name to connect to.
second %s = full path and file name to fetch.
third %s = screen name that AWeb is running on, in case your FTP program supports opening on a public screen.

telnet:

A *telnet:* address is used to start a telnet session. If you own a telnet client that supports command line arguments. you can use this feature. Note that a telnet address can contain: username:password@hostname:port If your telnet client doesn't support the full form, you can use a script to break it down, or accept that some telnet links won't work properly.

Argument parameters are:
first %s = host name or the extended form: user, password, host and port.
second %s = screen name that AWeb is running on, in case your telnet program supports opening on a public screen.

news:

A *news:* address points to a newsgroup or an article in a newsgroup. If you own a news reader that supports command line arguments or ARexx commands, you can use this feature.

Argument parameters are:
first %s = newsgroup name or article identification.
second %s = screen name that AWeb is running on, in case your news program supports opening on a public screen.

Start and stop your TCP stack

AWeb is able to start your TCP connection automatically when it is needed.

Use the chooser to select one of the entries below. Then use the Command and Arguments fields to specify the necessary command and arguments.

Start TCP
This command is used to start your TCP stack automatically.

For example, if you use the AmiTCP/IP package, you would set *command* to: *AmiTCP:bin/startnet* and leave the *arguments* field blank.

Argument parameters are:
first %s = screen name that AWeb is running on, in case your TCP script or program supports opening windows on a public screen.

End TCP
This command is used to stop your TCP connection after your confirmation. There are no argument parameter substitutions.

3.9.14 Program 1: General

On this settings page, you can change some general settings.

Save path

AWeb will always ask where to save a downloaded file, or the HTML source when using the `Project / Save As` menu function. The default save path will be the initial drawer used in the save file requester.

Scroll overlap

If you scroll up or down by a page, there is some overlap. Bacause you might want to have a larger overlapping area when you are using a larger font, you can change the overlap size.

Set this gadget to the desired overlap size, measured in pixels.

Allow Shell commands in links

AWeb offers a powerful facility to excute Shell commands just by clicking a hyperlink or submitting a form.

Although this feature can be very useful, it could also cause severe damage if an undesired command like FORMAT would be executed. Therefore this feature is disabled by default. Select this checkbox to enable it.

Hotlist requester auto close

If this checkbox is selected, the hotlist maintenance window will close automatically if you follow a link in that window. If this checkbox is not selected, the window remains open until you close it yourself.

Hotlist button gives requester

If this checkbox is selected, the button will open the hotlist maintenance window. Otherwise, the hotlist is displayed in the browser window.

History window auto close

If this checkbox is selected, the history window will close automatically if you pick a document from the window to display. If this checkbox is not selected, the window remains open until you close it yourself.

3.9.15 Program 2: External programs

On this settings page, you can configure some external programs to be used by AWeb.

Editor

This is the editor command invoked when you click the gadget in text area form fields.

Use the Command and Arguments fields to specify your editor command. Argument parameters are:
first %s = file name to edit.
second %s = screen name that AWeb is running on, in case your editor supports opening on a public screen.

Make sure the command will **not** return until you leave the editor. For some editors, this will need a STICKY or KEEPIO argument.

HTML source viewer

Currently, AWeb relies on an external viewer for the `Project / View source` menu function.

Use the Command and Arguments fields to specify your source viewer command. Argument parameters are:
first %s = file name to edit.
second %s = screen name that AWeb is running on, in case your viewer supports opening on a public screen.

Note that the default setting, *MultiView*, will not produce the expected results if you happen to have a HTML datatype installed on your system. In that case, the datatype will show the source as HTML again. If this happens, you should configure another viewer.

3.9.16 Program 3: Cache

On this settings page, you can change settings related to AWeb's cache.

Temp path

This is the directory where AWeb will create its temporary files. Temporary files are needed for several reasons: retrieved files for an external viewer, downloaded files, source files for inlined images, and swapped documents.

Type the full path name, or click the button located to the right of the gagdet to pop up a standard drawer requester.

Caches

Use these four gadgets to fine-tune the amount of cache AWeb should use.

All sizes should be given in kB.

Minimum free memory

These two gadgets determine the minimum amount of chip and fast memory that AWeb should leave free. Read the low memory description in section 4.6 to understand what these settings mean.

If you specify more memory than can possibly be freed (e.g. a non-zero fast memory limit on a machine with no fast memory), AWeb will continuously try to swap out all documents and images. So be careful what you enter here.

3.9.17 MIME types and external viewers

On this settings page, you can configure the MIME types AWeb should recognize, and the external viewers to use for each MIME type.

About MIME types

MIME (Multipurpose Internet Mail Extensions) is a mechanism for specifying and describing the format of Internet message bodies. It was primarily designed for e-mail, but MIME types are used also to identify the type of data in the HTTP protocol, the most widely used protocol on the World Wide Web.

For a browser like AWeb, the MIME type of a document determines whether the file should be displayed in the browser window, or be processed by some other program.

A MIME type consists of a *type* and a *subtype*. The *type* describes the major class of data, like text or image. The *subtype* is used for a subdivision of the major type into different formats, like GIF or JPEG images.

According to RFC 1521, the following official MIME types are defined:

TEXT/HTML
This is a document in the HTML hypertext format. Virtually all pages on the Web are in this format.

TEXT/PLAIN
This type is used for plain text documents (normally in ASCII).

APPLICATION/OCTET-STREAM
This describes a binary file. The file could be processed by some application. An example of this would be an LHA archive.

APPLICATION/POSTSCRIPT
The document is in PostScript format.

IMAGE/GIF and **IMAGE/JPEG**
These are images, in GIF and JPEG format.

AUDIO/BASIC
This type is used for audio data encoded using 8-bit ISDN mu-law [PCM].

VIDEO/MPEG
This is an animation in MPEG format.

In addition to these official types and subtypes, it is allowed to define extension MIME types and subtypes. These should start with **X-** to avoid collisions with future official MIME types.

Changing MIME types

Select the MIME type you want to modify from the listview. Use the **Add** button to add a new blank row. Use the **Del** button to remove the selected row. Note that the TEXT/HTML and TEXT/PLAIN types cannot be removed.

MIME type and subtype

In these string gadgets, you specify the MIME type and subtype. You can use an asterisk to specify a wildcard subtype. AWeb will use the external viewer defined in this row for files with the same type but a subtype for which no external viewer is defined. See the example.

Extensions

Most servers send the MIME type together with the data. AWeb will then use this MIME type, unless Ignore server MIME type is selected. If the server doesn't specify the MIME type (or if it is ignored), AWeb tries to determine the MIME type from the file name extension. If that fails, AWeb looks at the data to see if it is HTML text or plain text.

The extensions are especially important when looking at local files. As there is no server for local files, there is only the extension that tells AWeb about the type of the file.

In this string gadget, you type the extensions that could identify this MIME type. Separate multiple extensions by spaces or commas. The extensions are not case sensitive.

Processing

Files of types TEXT/HTML and TEXT/PLAIN will be shown in the browser window. Files of other types are processed by an external viewer. In spite of the name *viewer*, this is not limited to graphical files. The external "viewer" for an audio file, for example, will play the audio file.

Use the Command and Arguments fields to specify the viewer command to execute for this MIME type. Argument parameters are:**first %s** = file name to "view"**second %s** = screen name that AWeb is running on, in case your external viewer supports opening on a public screen. Use this only if you want it to open on the same screen as AWeb.

If AWeb can't determine the MIME type, or if the MIME type is known but not in the list, or if the MIME type is in the list but there is no external viewer defined, AWeb will pop up a save requester. You can than save the file, and try to process it later.

Example

Suppose you want to see JPEG images using the VT program, and other images using the MultiView program on its own screen. You know that JPEG files can have extensions **jpeg**, **jpg**, or **jfif**, and that GIF files have an extension **gif**. IFF images can be recognized by **iff**, **ilbm**, **ham** or **ham8**. You want AWeb to recognize other image formats you don't know of.

Then you would configure the following MIME types:

IMAGE/GIF gif
: This row specifies that GIF files can be recognized from their .gif extension. You specify no viewer because you want to use the default image viewer, defined in the IMAGE/* row.

IMAGE/JPEG jpeg jpg jfif SYS:Utilities/VT %s
: This row defines the possible extensions .jpeg, .jpg and .jfif for JPEG images. It also specifies that JPEG images should be displayed using the VT program.

IMAGE/X-IFF iff ilbm ham ham8
: This row defines an extension MIME type for IFF images. Note that the subtype starts with **X-** because it is not an official MIME type. This line is important when looking at IFF files on your local computer, as AWeb has no way to identify them as IFF images other than the extensions given here.

IMAGE/* SYS:Utilities/MultiView %s screen
: This row defines what viewer (MultiView) to use for all other images but JPEG. Even files with different subtypes than GIF or JPEG (but main type IMAGE) will be shown using this viewer. There are no extensions defined here, because all extensions are given in the different subtype rows. As an alternative, you could remove the IMAGE/GIF and IMAGE/X-IFF rows, and specify all extensions (gif iff ilbm ham ham8) here.

3.9.18 ARexx macro menu

On this settings page, you can configure what ARexx macros you can start from the ARexx menu.

Changing menu entries

Select the entry you want to modify from the listview. Use the **Add** button to add a new blank entry. Use the **Del** button to remove the selected entry.

Title

Type the title as you like it to appear in the ARexx menu.

Shortcut

You can select a shortcut from the chooser, or leave it to *none* if you don't want this entry to have a shortcut. Available shortcuts are **A**1 through **A**0.

You can assign different entries the same shortcut, but the shortcut will actually work only for the topmost menu item.

Macro

In this gadget, you can type in the name of the ARexx macro to execute. You can click on the button to pop up a file requester, so you can easily select the macro.

Rearranging menu entries

You can move an entry in the menu. Select the entry, either with the mouse or with the cursor up and down keys. Now you can move the entry up and down in the list with either the arrow up and arrow down buttons, or with the **Ctrl** + cursor up/down keys.

The **Sort** button will sort the entries. Entries with shortcut are moved to the top, in sequence of their shortcut. Entries without shortcut are moved to the bottom and are sorted alphabetically by their title.

Chapter 4

Advanced topics

4.1 HTML modes

4.1.1 About HTML

Most of the documents ("pages") found on the World Wide Web are written in *HTML* (HyperText Markup Language). HTML was originally designed as a standard hard- and software independent way of formatting documents. It is an application of *SGML* (Standard Generalized Markup Language).

The only official standard is HTML 2.0, which contains very limited possibilities. The W3 consortium was developing a new, extended standard, HTML 3.0.

Meanwhile, in the recent boom of Internet and the World Wide Web, some browser manufacturers have introduced several ad-hoc extensions to HTML, of which many didn't fit in the new HTML 3.0 standard. Probably because this would make HTML 3.0 an academic standard without any practical use, the development of HTML 3.0 was abandoned. AWeb supports some of the extensions found in HTML 3.0.

Recently the W3 group proposed a new HTML 3.2 standard, which contains many of the widely used NetScape and Microsoft Internet Explorer specific extensions. AWeb will fully support HTML 3.2 in the near future.

The large browser manufacturers have introduced other tags, that aren't included in the HTML 3.2 standard. AWeb will try to support most of these non-standard extensions.

To make things even more inconvenient, some earlier versions of popular PC browsers didn't stick to the SGML rules. And even recent versions of those browsers still have problems with SGML comments. Because many people design their pages using these browsers, there are many documents on the web that just are bad HTML.

4.1.2 HTML modes

AWeb does its best to display all pages correctly, but sometimes you have to set the way HTML should be interpreted to get the best results.

You can choose out of three *HTML modes*:

Strict
: In *strict* HTML mode, AWeb understands only the proposed HTML 3.2 standard.

Tolerant
: In *tolerant* HTML mode, AWeb understands also other extensions. These are both extensions specific to other browsers, and HTML 3.0 extensions that can't be found in HTML 3.2.Below is a list of all extensions that AWeb supports.

Compatible
: In *compatible* HTML mode (a nice way of formulating "buggy"), AWeb tries to interpret buggy HTML. Below is a description of the deviations of the standard when compatibility mode is used.

Set one of these in the Browser 3: Options settings page.

4.1.3 Extensions to HTML 2.0

AWeb currently supports the following proposed and/or other extensions to the HTML 2.0 standard. With each extension, the HTML modes that recognize the extension are mentioned within parentheses.

- **<BODY BACKGROUND=***url* **BGCOLOR=#***rrggbb* **TEXT=#***rrggbb* **LINK=#***rrggbb* **VLINK=#***rrggbb* **ALINK=#***rrggbb***>**
  Background images and document colours. If no BACKGROUND or BGCOLOR is given, then TEXT, LINK, VLINK and ALINK are ignored. (strict, tolerant, compatible)

- **<DIV ALIGN=***align***>**
  <H*n* **ALIGN=***align***>**
  <P ALIGN=*align***>**
  <CENTER>
  Alignment of divisions, headings and paragraphs. *align* = LEFT, CENTER and RIGHT are supported. (strict, tolerant, compatible)

- **<DFN>**
  <STRIKE>
  Phrase markup. (strict, tolerant, compatible)

- **<IMAGE BORDER=***n* **WIDTH=***n* **HEIGHT=***n***>**
  Image border size and preset dimensions. (strict, tolerant, compatible)

- **<HR ALIGN=***align* **NOSHADE SIZE=***n* **WIDTH=***n%—n***>**
  Enhanced horizontal ruler. (strict, tolerant, compatible)

- **<OL START=***n* **TYPE=***otype***>**
  <UL TYPE=*utype***>**
  <LI TYPE=*otype—utype* **VALUE=***n***>**
  Enhanced lists. *otype* can be **A**, **a**, **I**, **i** or **1**. *utype* can be **DISC**, **CIRCLE** or **SQUARE**. (strict, tolerant, compatible)

- **<OL CONTINUE SEQNUM=***n***>**
  <UL PLAIN SRC=*url* **DINGBAT=***name***>**
  <LI SKIP=*n* **SRC=***url* **DINGBAT=***name***>**
  List extensions from the HTML 3.0 specification. (tolerant, compatible)

- &*icon.name*
  All proposed WWW icon entities (dingbats). Look at the on-disk overview for all supported icons. (tolerant, compatible)

- **<FRAME SRC=***url* **NAME=***name***>**
  Very limited frame support. AWeb does not yet support frames, but it recognizes the <FRAME> tag and shows a hyperlink for each document. You will still see those annoying "your browser doesn't support frames, download NetScape here" messages, but at least you can access the information. (tolerant, compatible)

4.1.4 Compatible mode

As mentioned above, some pages contain bad HTML. When you view such a page, it can look distorted. You can expect large parts of the page missing, links to URLs that seem to contain HTML tags, and other strange things.

If you encounter such problems, try using the *compatible* HTML mode of AWeb. **Warning:** Using compatible HTML mode, documents containing *valid* HTML might look distorted in turn.

In compatible mode, AWeb exposes the following deviations from the SGML standard:

- quoted attribute values are terminated by any occurrence of ">"
- quoted attributes that contain URLs, like HREF, SRC and ACTION, are terminated by whitespace
- comments are terminated by any occurrence of "–>"

4.2 ARexx interface

4.2.1 ARexx port names

Every AWeb window has its own ARexx port. This port is named **AWeb.#**, where # is a unique number.

The About requester shows the actual name of the ARexx port for the window from which you selected the About menu item.

Due to internal limitations, it is not possible to have more than 14 ARexx ports open simultaneously. When you open more than 14 windows, the 15th and later windows will not have an ARexx port.

Note that the first window can be closed while later windows are still open, so it is not guaranteed that the port AWEB.1 exists. Use the following code fragment to determine a valid AWeb port:

```
ports = SHOW('P')
PARSE VAR ports dummy 'AWEB.' portnr . /* note the trailing period! */
ADDRESS VALUE 'AWEB.' || portnr
```

Once you have got a port, you can use the GET ACTIVEPORT command to get to the active window:

```
OPTIONS RESULTS
'GET ACTIVEPORT'
ADDRESS VALUE RESULT
```

4.2.2 ARexx commands

Currently a very rudimentary command set is implemented. More commands will be added in the future.

Available commands are:

OPEN URL/A,RELOAD/S
Retrieve and show the document for this URL. The RELOAD switch will reload the document even it is still in the document cache.

RELOAD
Reload the current document.

ALLOWCMD
Temporarily allow shell commands and ARexx macros to be started from hyperlinks regardless of the Allow shell commands setting. This allows easier usage of ARexx plug-ins without need for the user to explicitly change the settings.

Commands are allowed only in the current document, or in the document that is being loaded. After a new document is loaded in the window, the normal settings apply.

GET ITEM/A,VAR/K
Get information from the document in this window. The *ITEM* argument determines the information to return:

URL - Retrieve the URL of the document.

SOURCE - Retrieve the HTML source of the document. Note that due to an ARexx limitation only the first 65535 bytes are returned.

TITLE - Retrieve the title of the document. If no title was defined in the document, the document's URL is returned.

SCREEN - Retrieve the name of the screen that AWeb uses to open its windows on.

ACTIVEPORT - Retrieve the name of the ARexx port associated with the most recently activated AWeb window. Useful in ARexx macros started via a shell script or shell command to find the active window.

The information is returned in the reserved variable RESULT, unless the *VAR* argument is used to specify the variable name.

CACHE ITEM/A,FROM/A

Get information about AWeb's cache. The *ITEM* argument determines the information to return, and the *FROM* argument determines the object to get information for. *ITEM* can be one of the following:

URL - Get the URL for a file in cache. *FROM* must be the name of a cache temporary file, with or without path.

NAME - Get the file name for a file in cache. *FROM* must be the name of a cache temporary file, with or without path. The file name returned is the last part of the URL, after the last slash.

TEMP - Get the temporary file name for a URL. *FROM* must be a URL of a document or image in cache. Note that no URL parsing is done, so the URLs must match exactly. Also note that many documents won't have a corresponding cache file.

The information is returned in the reserved variable RESULT.

SAVEAS NAME,APPEND/S

Save the HTML source of the document.

If *NAME* is given, the source is saved under this name. If the *APPEND* switch is set, the source is appended to the file, otherwise the file will be overwritten.

If no *NAME* is given, a save requester will pop up.

ACTIVATEWINDOW

Make this window the active window.

WINDOWTOFRONT

Move this window in front of all other windows on the screen.

WINDOWTOBACK

Move this window to the back of all other windows on the screen.

SCREENTOFRONT

Move the screen that AWeb is using in front of all other screens.

SCREENTOBACK

Move the screen that AWeb is using to the back of all other screens.

CLOSE FORCE/S

Close this window. The FORCE switch suppresses the "Are you sure" requester if this was the last window.

QUIT FORCE/S

Quit AWeb. The FORCE switch suppresses the "Are you sure" requester.

4.2.3 Return values from commands

Every ARexx command returns a completion code in the reserved ARexx variable RC.

ARexx commands return the following codes:

0 Command executed succesfully.

1 Command executed succesfully, but there is some condition that might be of interest.

5 The command was syntactically valid, but could not be completed for some reason.

10 You supplied invalid arguments for this command.

11 You submitted an unknown command.

20 There was an internal error. The command is not executed.

4.3 Shell command and ARexx macro interface

AWeb offers a unique and powerful facility to execute Amiga DOS Shell commands and ARexx macros from a page, just by clicking on a hyperlink or by submitting a form. With some effort, you can create complex applications using AWeb as the user interface, starting scripts that dynamically compose new documents that are loaded into AWeb via ARexx, etcetera.

Although this feature can be very useful, it could also be very dangerous. Therefore this feature works only from local pages (with a URL starting with file://localhost/), and only if the Allow Shell commands setting is selected.

4.3.1 Simple shell commands

To include a simple command, just add a normal hyperlink in your document that points to a URL of the form x-aweb:command/*your_DOS_command*. If the user clicks on the hyperlink, *your_DOS_command* is executed. The output of the command is directed to an auto opening console window, unless you specify another output redirection in your command.

Because compatible HTML mode stops the URL at a space, make sure you have escaped all spaces in the command by "** **" or else the command won't work if the user has selected compatible HTML mode.

Example: get dir would allow the user to execute the *dir sys all* command by a click on the words "get dir".

Note: The DOS command is executed in a separate shell, with a current directory set equal to the current directory of AWeb. You are advised to use only absolute path names in the DOS command, or else the result will depend on which directory happened to be the current directory when you started AWeb.

4.3.2 ARexx macros

Starting ARexx macros from your page works in a similar way. Just add a normal hyperlink that points to a URL of the form x-aweb:rexx/*your_ARexx_macro*. If the user clicks on the hyperlink, *your_ARexx_macro* is started with the ARexx port for this window as the default command port.

4.3.3 Parameters

You can use a HTML *form* or a *clickable map* to pass parameters to your DOS command or ARexx macro.

Forms

Supply a ACTION="x-aweb:command/*your_command*" attribute in your <FORM> tag to execute the command if the user submits the form. Similarly, you can include a ACTION="x-aweb:rexx/*your_macro*" attribute to start the ARexx macro.

Form parameters are converted to Amiga DOS style parameters: the field name will be used as the argument name, and the field value will be used as argment value. The value will be quoted, with the *escape*, *newline* and *quote* characters in the value escaped as required by Amiga DOS.

Note: *switch arguments* (/S) cannot be passed in this way. You could use a script instead, like the example below.

Clickable maps

When using a clickable map, the x and y coordinates of the mouse pointer within the image are passed to the command as parameters without keyword.

ARexx arguments

Parameters for ARexx macros are passed in the same format as for DOS scripts. The argument string will contain the name, an equal sign, and a quoted value for each form parameter. Have a look at the second example below for one possible way of parsing this.

Load the result back into AWeb

If your script or macro has created a HTML document (or just a plain text file), you can automatically load this file back into AWeb. Use the ARexxOPEN command for this purpose. If you re-use the name of your file for different responses, be sure to add the RELOAD switch to prevent AWeb from showing the previous (cached) document again.

Of course, this will work better from within an ARexx macro than from within a DOS script. In a DOS script, you have no way of determining to which ARexx port you should address the OPEN command.

4.3.4 Examples

The on-disk documentation contains two examples of this feature. Have a look at the source of the examples.

4.4 Using your own default images

There are moments where just displaying text is not sufficient, but some imagery is needed. For some of these imagery, AWeb uses external images that you can replace with your own.

AWeb uses these external images in the following places:

- As default icons for unloaded images.
- For displaying WWW icon entities like &document.

All external images are located in the Images drawer. These are image files, that can be in any format for which a datatype is available. Transparent GIF's are also supported.

For unloaded images, AWeb uses the following files:

- **def_image** for an unloaded normal image.
- **def_imagemap** for an unloaded image map.
- **def_errimage** for an image that couldn't be loaded.

Please refer to the on-disk documentation for the actual images currently used.

For all supported icon entities, there is a file with the same name. So for `&document` there is a file named document in the Images drawer.

An overview for all supported entities is presented in the on-disk documentation.

To install your own image, just copy your image over the old one. Be sure to use the same name, and don't use a suffix like .iff or .gif.

The Storage drawer contains some alternatives for the icon entities, but you can use any image of course.

4.5 Extension URLs

Internally, AWeb uses an extension to the URL scheme for some tasks. You can take advantage of this, by using the same URLs. These extension URLs always start with "**x-aweb:**". Note that you should only include such extension URLs in pages that will only be viewed by AWeb, because other browsers will not recognize these URLs.

A good place for extension URLs would be your hotlist. You can, for instance, access the AMosaic or IBrowse hotlist from within your hotlist, or even configure one of these as your home page within AWeb.

4.5.1 Hotlists

AWeb uses extension URLs to identify its hotlist, and other browser's hotlists.

x-aweb:hotlist
Identifies AWeb's own hotlist.

x-aweb:amhotlist.rexx
Identifies the ARexx based hotlist of AMosaic version 1.2. It uses the file ENV:mosaic/hotlist.html.

x-aweb:amhotlist.20
Identifies the hierarchical horlist of AMosaic 2.0 prerelease. It uses the file ENV:mosaic/.mosaic-hotlist-default.

x-aweb:ibhotlist/*path*
Identifies the hotlist of IBrowse. Because this hotlist doesn't have a fixed location, you must specify the full path and file name in the URL.

4.5.2 Shell commands and ARexx macros

Two extension URLs exist to start shell commands or ARexx macros.

x-aweb:command/*shell_command*
Forms the interface to start Shell commands.

x-aweb:rexx/*ARexx_macro*
Forms the interface to start ARexx macros.

4.6 How the cache works

AWeb uses a special caching system, partially in memory and partially on disk, to reduce the number of network accesses.

Note that the AWeb cache is a *cache*, not a *proxy* system. This means that the cache is cleared when AWeb finishes. Also, the cache doesn't take the HTTP expiration date into account. This might change in later versions of AWeb.

4.6.1 Documents

Documents that can be displayed by AWeb are initially loaded in memory. When the document cache is full, the least recently displayed document is swapped out to disk, thereby freeing memory. If a swapped-out page must be displayed again, it is swapped back into memory, thereby probably swapping out other pages.

The swapped documents are stored in AWeb's temporary directory. It should be clear that this would be only meaningful if the temporary directory isn't in RAM. So, if you are using a temporary directory in RAM, be sure to set the *document cache memory size* large enough.

When the total size of swapped out documents is larger than the *document cache disk size*, the least recently displayed documents are deleted.

4.6.2 Images

Inlined images are treated completely differently. Images are stored in the temporary directory when they are loaded. If the image is received completely, the datatype will *process* the file, resulting in a displayable image in *chip* memory (the image cache). When the image cache is full, undisplayed images in memory are removed, but the disk file remains. If the image must be displayed later, the disk file is processed again, but no network access is needed.

Because the datatype locks the file, the disk file must remain while the image is in memory.

When the total size of image files is larger than the *image cache disk size*, the least recently displayed images are deleted.

4.6.3 Low memory

To avoid crashes when running out of memory (caused by the datatypes or other external programs), AWeb tries to leave at least some amount of memory free. Whenever there is less memory free than a configurable minimum, AWeb will start to swap out documents, even if the document cache isn't full yet. Displayed documents will never be swapped out. If this still haven't freed up enough memory, AWeb will start flushing images from memory, even if they are being displayed. As long as the image disk cache isn't full, they remain on disk.

4.6.4 Setting the cache sizes

You can set the cache sizes from the Program 3: Cache page in the settings requester.

If you are using a 2MB Amiga, have a look at the 2MB tips in section 2.3. Otherwise, you will probably want to use a

- temporary directory in RAM;
- a *document memory cache* large enough,
- a *document disk cache* of size 0;
- a reasonably sized *image memory cache* (this will take up chip memory),

- a large *image disk cache*.

Of course, you can use other settings if you like. You can fine-tune the memory and disk usage by changing these settings.

If you use AWeb on a native screen, you will want to keep some chip memory free as a work space for the datatypes. On CyberGraphics systems images will go in fast memory, so you will want to leave some fast memory free on those systems. The exact amount depends on the number of colours on the CyberGfx screen, and of course on the total amount of memory.

Chapter 5

Miscellaneous

5.1 Common problems

Listed below are some common problems you might encounter, and their solution.

- *AWeb crashes when loading a GIF image*
  The commonly used ZGif datatype version 39.16 had a bug. Be sure to use version 39.18 or better of the ZGif datatype, or another GIF datatype.
- *AWeb doesn't display all of the page. Other browsers display the page correctly*
  Probably the page contains some erroneous HTML. One of the common errors is the use of something like <!——————————-> as a dividor line in the HTML source. This can lead to large parts of the page commented out if the number of dashes is not exactly right. Apart from notifying the author of the page that his page contains erroneous HTML, you can try to use HTML mode compatible.
- *AWeb becomes terribly slow if I run it on its own screen.*
  This has something to do with chip memory usage and DMA bandwidth. Apparently, the problem becomes worse when you have an accelerated machine. The solution is to use either a lower or a higher speed ("baudrate") to communicate with your modem. Start at the modem speed (so if you own a 14k4 modem, start at 14k4 communication speed). Then increase the speed until you find a dramatic slowdown. Then go one step back. Changing the communication speed with your modem might involve adapting the settings for your dialler, TCP stack and/or your SANA driver. Refer to the respective documentations.
- *Backgrounds in AWeb are terribly slow, much slower than competitive Amiga browsers.*
  This might be caused by an unsuited picture datatype. Use the 24-bit picture datatype **only** on CyberGraphics screens; on original Amiga screens this datatype becomes very slow. Use the original picture.datatype from your Workbench diskette.

- *The colour requester from the Screen 2: palette settings page messes things up on a CyberGraphics screen.*
  This is caused by the palette-orientated design of the Amiga OS. This works fine on Amiga chipset screens, but isn't really suited for graphics cards.
- *Closing the AWeb public screen while there is a visitor window open crashes if the screen is a CyberGraphics screen.*
  This is a bug in CyberGraphics V 2.15 and lower. The same thing happens on other CyberGraphics screens.
- *If the window is not active, a click in the scroller container (outside the knob) moves the scroller but doesn't scroll the window contents.*
  This is a bug in Intuition. The same thing happens sometimes with MultiView.
- *I use MagicMenu, and every now and then my system hangs if I open a menu.*
  This is a known problem in MagicMenu, caused by the fact that MagicMenu is not 100% compatible with the way Intuition handles menus.
- *I don't use MagicMenu, but my system hangs if I click on the wide part of the chooser gadget in the settings window.* Be sure to use the version of chooser.gadget included in this archive. Also, some animated mouse pointer commodities are known to cause this hang.

5.2 Known bugs

5.2.1 AWeb bugs

Although AWeb is thoroughly tested, it is very likely that there are some bugs left. At the time of release there were no known bugs. For the latest information, check the AWeb Support Page at
http://www.networkx.com/amitrix/index.html

5.3 Things to do

This page lists all suggested enhancements at the time of release. The online to-do list lists additional suggestions, received after the release. Please check both this page and the online page before sending me any suggestions.

5.3.1 Future releases

AWeb is still being developed. Future releases of AWeb will support:

- More HTML-3.2 commands like tables and image alignment
- Client pull

- Server push
- Full ARexx command set
- Persistent document and image cache on disk.

5.3.2 Far future

In the far future, AWeb *may* support:

- Save page and images as PostScript
- Separate user-maintained disk cache to store documents and images so you can view them later off-line.
- Java
- Style sheets
- User configureable keyboard
- Select text and copy to clipboard

5.3.3 Other enhancements

These enhancements were suggested to me. I don't know when I'll include these things in AWeb, if at all, but probably many of them will be in the next major release.

Enhancements in functionality

Network

- Support FTP internally.
- Add exclusion list for proxy's with domains not to use the proxy for.
- Retry connection past a proxy if connection to proxy fails.
- Option: cancel all pending inline image loads for a document when following a link.
- "Pause network activity" button.
- Support telnet: with external program.

Browsing

- Implement global history. Also: possibility to choose an URL from the global history.
- Global window history, editing, rearrange, copy to hotlist.
- Configurable 'direct buttons'.
- Document headers feature. Either: 1) append <hr> and <dl> with headers to every document; 2) open window on user request with headers in it; 3) create button that links to x-aweb doc that has them, and maybe other interesting things as well.
- Let ctrl-click load the page in a new window.
- Use external viewer with pipe
- Option: View inline image with external viewer
- Direct support for .z and .gz files: uncompress before processing or starting external viewer.
- Different font settings for <address>, <blockquote> and <cite>
- Let dragging with the mouse scroll the page.
- Option: render images vertically compressed on 1:2 pixel screens (like 640x200).

Program

- Do 'Save as formatted text'
- Include the Print function
- Include the Find in current function
- View HTML using textview.gadget (editable?)
- Iconify option (transfers continue while iconified).
- Load settings, Save settings as...
- Check if saved/downloaded file fits on disk
- Screen preferences: alternative screen name if first one doesn't exist.
- Screen preferences: wildcards in screen name like DOPUS.*
- Ask save filename before download, instead of saving to temp file first.
- Support for HAM Workbench
- Support environment variables EDITOR and HTTP_PROXY etc.

ARexx

- ARexx DOWNLOAD command with option to return document in RESULT
- ARexx command to get all URLs in a document

Miscellaneous

- Character entities should stop if unambiguous name is found (e.g. &eumln should recognize the '&euml') Only in compatible HTML mode.
- Use ENCTYPE for forms sent to x-aweb:command, allowing running cgi applications.

Cosmetic enhancements

GUI

- Make gadget font configurable.
- Preference: buttons left or right, 1 or 2 button rows
- Block 2nd 'open local' requester from same window if one is still open. Same for 'save as' (any more?)
- Add up/down arrows to URL gadget to scroll trough window history.
- Text in navigation buttons with single hotkey
- Enhanced graphics in navigation buttons

Browsing

- No redisplay if loaded image is completely above the visible window.
- Add extra pixel row between lines on hi res screens
- Display <p>item</p> without LF between bullet and text
- Option: display paragraphs indented without blank line
- Form field: <select size=2> en #options=2: make listview size 2 instead of oversized listview
- Put text 'loading images' in progress gadget while loading images for displayed document.
- Instead of refreshing after each 'image processing ready' message received, collect all messages from the port and do one refresh. Done this in 0.10, but doesn't give the desired results. Instead, after receiving an 'image ready' message, it should wait a while (user configureable) for more images to be completed before refreshing the window.

- Option: new/visited links underline type. Or a user selectable image for underlining.
- Flag: proxy status on/off in window titlebar.
- Special colour for links to pages in cache.
- Enhanced list item bullets (3d)

Program

- Revert to previous screen mode if selected screen mode doesn't open, instead of falling back to Workbench.
- Save positions for 2nd and later windows.
- Take display clip into account when opening full-sized window.
- If settings changed but not saved, show warning requester before quit.
- If named screen does not exist, show error requester (and open on default public screen).
- On a snapshot, remember if the network window was open and re-open it after a restart.
- Save the zoomed window dimensions too.
- Preload grayscale palette for 16 or less colours.
- Screen preferences: requesters with present public screens to choose from.
- Open information window if external editor starts.
- Window with cache or memory indicator.

Miscellaneous

- Don't refresh border of listview and textarea form fields with every character typed
- Do proper URL encoding; decode on file://localhost
- Proxy prefs: separate string gadget for port number.

5.4 How to contact the people responsible

You can contact AmiTrix Development

- by e-mail: support@amitrix.com
- by snail-mail:
  AmiTrix Development
  5312 - 47 Street
  Beaumont, Alberta
  T4X 1H9
  CANADA

  Phone or fax: 1+ 403-929-8459
- through the Web : http://www.networkx.com/amitrix/index.html

For more information on ordering AWeb-II or other AmiTrix products, contact us at the above address, or by e-mail: sales@amitrix.com

Dealer inquiries are welcomed.

5.5 Acknowledgements

I especially wish to thank:

- **Osma Ahvenlampi** ("Tau") for the TCP stack-adaptive design
- **Josef Faulkner** for writing many useful ARexx add-ons
- **Jeroen Oudejans** for creating and maintaining the AWeb FAQ and for creating the postscript printable version of the docs
- **Thomas Tavoly** ("aTmosh") for enhancing the logo and creating the Workbench icons

I thank for their translations:

- Anders Bakkevold (Norwegian)
- Jean-Michel Bezeau (French)
- Gabriele Favrin (Italian)
- José Roberto González Rocha (Spanish)
- Ulf Holm (Swedish)
- Pantelis Kopelias (Greek)
- Richard Marti (German)

I also wish to thank my beta testers, without them AWeb would never have become the great program it is.

- Osma Ahvenlampi ("Tau")
- Jeroen Oudejans
- Thomas Tavoly ("aTmosh")
- Vincent Groenewold ("supernov")
- Donovan Janus
- Mike Meyer
- Paul Kolenbrander
- Donald Voogd
- Kristian Phillips
- Dale Currie
- Christopher Aldi
- Josef Faulkner

AWeb was written by Yvon Rozijn.