W3A – WWW Applets API

May 30, 1995

Draft with annotations (28 Feb 1995) for W3A version 1.1, by **Bert Bos**
See also:

1. (27 Jun '94) **Questions and answers**, a regularly updated summary of discussion.
2. Somewhat **related issues.**
3. History of **changes.**

An API for WWW Applets

This proposal describes a way in which capabilities (in the form of software modules) can be added to World-Wide Web browsers. It is an interface specification that lists the functions that browsers and external modules must export.

It assumes that the browser supports dynamic linking or explicit loading (that is, it must be capable of linking object files to itself while it is running). However, the modules can also be linked statically when the browser is compiled, e.g., on platforms that do not support dynamic linking or for debugging purposes.

Contents

11. **Appendix: a GIF viewer**
12. **Appendix: sample `w3a.h` file**

And soon:

1. Appendix: a Gopher agent
2. Appendix: compilation instructions for HP-UX

Background

There are several reasons why it is useful to define a standard API for dynamic linking of *applets* (loosely defined here as a piece of software that needs a *host* program to attach itself to, like a cooperative parasite in nature).

- Widely dispersed programmers can write to the specification and be sure that their code works when combined.
- Changes in protocols, formats, etc. can be accomodated by writing a new applet, without the need to change the browser.
- People can experiment with new functions, protocols, etc. without having to write a completely new browser.
- (Individual) people with limited time/resources can still contribute significant functionality to the Web.

Five areas can be distinguished in a browser that lend themselves to attaching applets. A browser can choose which of these it will support:

W3A/A Making connections & retrieving data can be done with applets that I'll call **agents.** The part of the standard that defines the API (Application Program Interface) for agents is called 'W3A-A.'

W3A/F A **filter** is a program that converts a document from one format to another. It can be an alternative to writing a new viewer for that format.

W3A/V A **viewer** is used to display a document in a particular format and to manage the user's interaction with it.

W3A/U A **user function** or **accessory** adds a function to the user interface. It can be as simple as a pop-up calculator or as complex as a WYSIWYG HTML editor. A remote-control utility is another possibility.

W3A/P A **printer driver** is similar to a viewer, except that it renders a document of a certain type on a printer (hardcopy device), instead of on the screen.

common functions In addition, there is a common set of functions that every browser must provide.

The model for the W3A API is based on dynamic linking and a naming convention for functions. When data is to be exchanged between applet and host a stream metaphor is used (open, read/write, close).

A browser will usually have a configuration file that is read when the program starts. To add an applet to a browser, the applet's signature is added to the configuration.

All applets must be fully reentrant (although user functions may want to return immediately if they are called twice). Consider for example the following situation: the user clicks on a hyperlink in a viewer, the viewer calls `W3Aprocess()`, which causes the browser to retrieve a document and start a new viewer, which happens to be the same viewer (but with a new window).

Applets can be viewed as objects (better: object classes) that are instantiated by the call to `open`.... The functions that applets export are their 'methods.'

> **Question:** *A viewer may want to execute the call to* `W3Aprocess` *in a subprocess. Is this allowed? Or should the browser be the only one allowed to call* `fork()`*? See the* **description and the note** *of* `W3Aprocess()`.

This definition assumes C bindings. Suitable translations will have to be defined when other languages are used.

> **Example:** *It shouldn't be too hard to write a small (< 50 lines) applet in C that executes arbitrary Python programs.*
