WebRunner™: W*hat* & W*hy*

The Internet is a vast sea of data represented in many formats and stored on many hosts. Mosaic™ and NetScape™ are two of many browsers for that data. Browsers allow people to treat the data spread across the Internet as one cohesive whole. They integrate the function of fetching the data with figuring out what it is and displaying it. One of the most important file types they understand is the hypertext markup language (HTML). HTML allows text data objects to embed simple formatting information and references to other objects.

WebRunner is a web browser that makes the Internet "come alive." It builds on the network browsing techniques established by Mosaic and expands them by adding dynamic behavior that transforms static documents into dynamic applications. Current documents in Mosaic are limited to text, illustrations, low-quality sounds and videos.

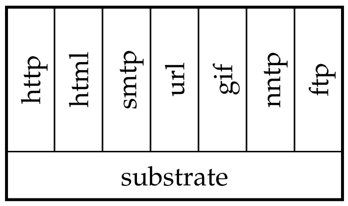
WebRunner shatters these limitations by adding the capability to add arbitrary behavior. Using WebRunner you can add applications that range from interactive science experiments in educational material, to games and specialized shopping applications. You can implement interactive advertising, and customized newspapers; the possibilities are nearly endless.

In addition, WebRunner provides a way for users to access these applications in a new way. Software transparently migrates across the network. There is no such thing as "installing" software. It just comes when you need it (after, perhaps, asking you to pay for it). "Content" developers for the World Wide Web don't have to worry about whether or not some special piece of software is installed in a user's system, it just gets there automatically. This transparent acquiring of application frees developers from the boundaries of the fixed media types like images and text and lets them do whatever they'd like.

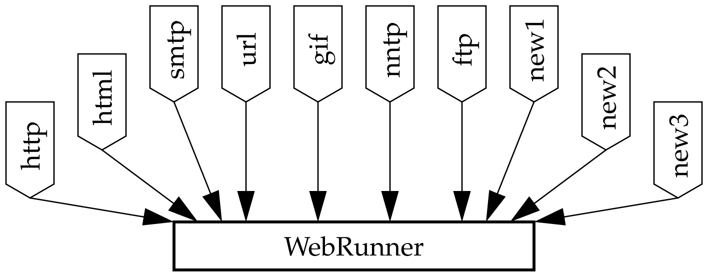
The Essential Difference

The central difference between WebRunner and other browsers is that while these other browsers each have a lot of detailed, hard-wired knowledge about the many different data types, protocols and behaviors necessary to navigate the Web, WebRunner understands essentially none of them. But what it does understand is to how find out about those things that it doesn't understand.

The result of this lack of understanding is great flexibility and the ability easily to add new capabilities.



A conventional browser: a monolithic chunk, all bound tightly together.



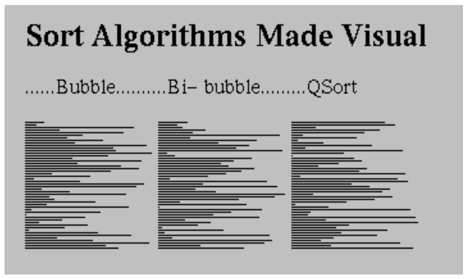
WebRunner: the coordinator of a federation of pieces, each with individual responsibility. New pieces can be added at any time. Pieces can be added from across the network, without needing to be concerned with what CPU architecture they were designed for and with reasonable confidence that they won't compromise the integrity of a user's system.

What makes this federation of pieces and dynamic addition of capabilites possible is Java, the underlying programming language and environment on which WebRunner is built. (See the companion document to this, *Java: an Overview*, for and explanation about how Java enables WebRunner.) Briefly, one can think of Java as a simplified, safe and portable version of C++. It has an architecture-neutral distribution format, meaning that after you've compiled a piece of Java code it will run on any CPU architecture.

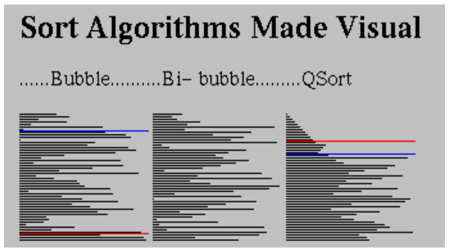
Dynamic content

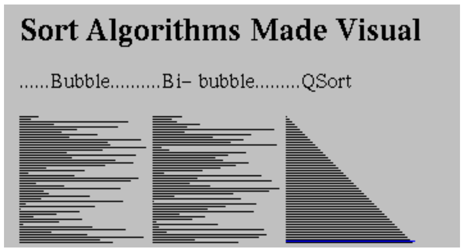
One of the most visible uses of WebRunner's ability to dynamically add to its capabilities is something we call *dynamic content*. For example, someone could write an Java program following the *WebRunner* API that implemented an interactive chemistry simulation. People browsing the net with the *WebRunner* browser could easily get this simulation and interact with it, rather than just having a static picture with some text. They can do this and be assured that the code that brings their chemistry experiment to life doesn't also contain malicious code that damages the system. Code that attempts to be malicious or which has bugs essentially can't breach the walls placed around it by the security and robustness features of Java

For example, the following is a snapshot of WebRunner in use. Each diagram in the document represents a sort algorithm. Each algorithm sorts an array of integers. Each horizontal line represents an integer: the length of the line corresponds to the value of the integer and the position of the line in the diagram corresponds to the position of the integer in the array.


Sort Algorithms Made Visual
......Bubble..........Bi- bubble.........QSort

In a book or HTML document, the author has to be content with these static illustrations. With WebRunner the author can enable the reader to click on the illustrations and see the algorithms animate:


Sort Algorithms Made Visual
......Bubble..........Bi- bubble.........QSort

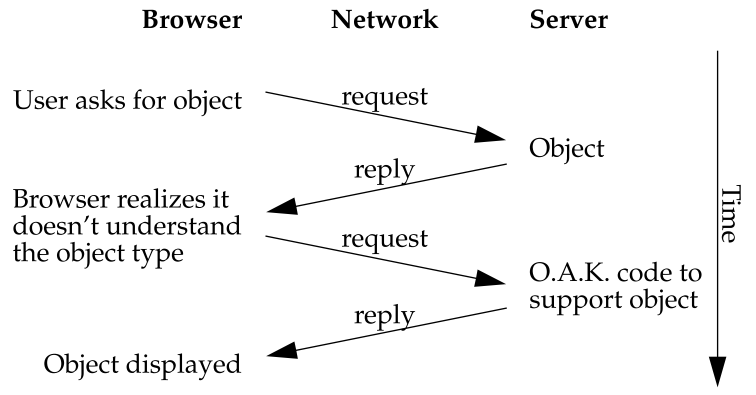

Sort Algorithms Made Visual
......Bubble..........Bi- bubble.........QSort

Using these dynamic facilities, content providers can define new types of data and behavior that meet the needs of their specific audiences, rather than being bound by a fixed set of objects.

Dynamic types

WebRunner's dynamic behavior is also used for understanding different types of objects. For example, most web browsers can understand a small set of image formats (typically GIF, X11 pixmap, and X11 bitmap). If they see some other type, they have no way to deal with it directly. WebRunner, on the other hand, can dynamically link the code from the host that has the image allowing it to display the new format. So, if someone invents a new compression algorithm, the inventor just has to make sure that a copy of the Java code is installed on the server that contains the images they want to publish; they don't have to upgrade all the browsers in the world. WebRunner will, in effect, upgrade itself on the fly when it sees this new type.

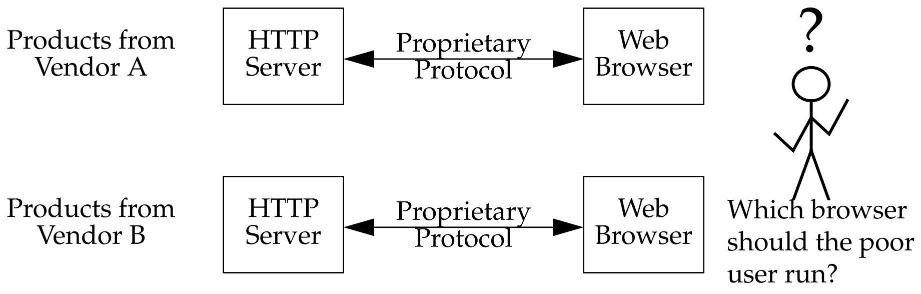
The following is an illustration of how WebRunner negotiates with a server when it encounters an object of an unknown type:



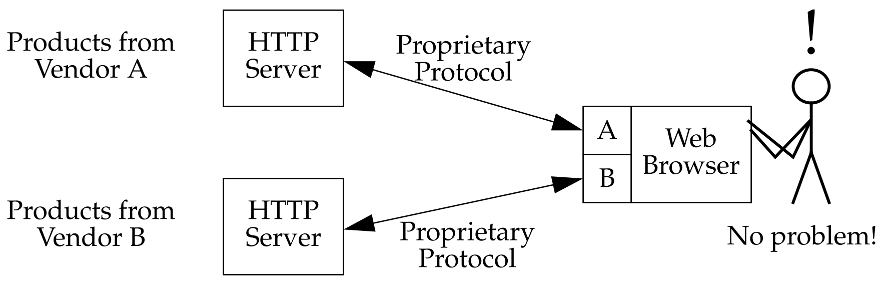
Dynamic protocols

The protcols that Internet hosts use to communicate among themselves are key components of the net. For the World Wide Web (WWW), HTTP is the most important of these communication protocols. In documents on the WWW a reference to a document is called a URL. The URL contains the name of the protocol, HTTP, that is used to find that document. Current web browsers have the knowledge of HTTP built-in. Rather than having built-in protocol handlers, WebRunner uses the protocol name to link in the appropriate handler. This allows new protocols to be incorporated dynamically.

The dynamic incorporation of protocols has special significance to how business is done on the Internet. Many vendors are providing new web browsers and servers with added capabilities, such as billing and security. These capabilities most often take the form of new protocols. So each vendor comes up with their unique style of security (for example) and sells a server and browser that speak this new protocol. If a user wants to access data on multiple servers on which each has proprietary new protocols, the user needs multiple browers. This is incredibly clumsy and defeats the synergistic cooperation that makes the WWW work.



With WebRunner as a base, vendors can produce and sell exactly the piece that is their added value, and integrate smoothly with other vendors, creating a final result that is seamless and very convenient for the end user.


Products from Vendor A
HTTP Server
Proprietary Protocol
Products from Vendor B
HTTP Server
Proprietary Protocol
A
B
Web Browser
!
No problem!