
From: chrisjo@exchange.microsoft.com
Sent: Thursday, August 10, 1995 11:26 AM
To: bens@MICROSOFT.com; victors@MICROSOFT.com; thomasre@MICROSOFT.com; johnlu@MICROSOFT.com
Subject: Internet Plans -- Take 3

Here's take three of this, also on \\chrisjo4\public\internet team plans.doc.

Kind of long, but I think it covers most of the points.

Chris

INTERN~2.DOC

Internet Team Plans

The purpose of this document is to provide an overview of the Internet Team plans in 95, 96, and 97. This is an initial proposal and vision statement - comments are welcome and should be sent to ChrisJo.

Summary

- The Internet enables a new class of applications, and the best platform for these applications will win. When you look at what authors want to bring to users, both today and tomorrow, there is a set of common problems they all need to solve.
- Windows and OLE do not support this class of applications and documents well today. Other companies have recognized this opportunity and are delivering middleware solutions, which threaten our core business and steal developer mind share. To fill this void, we need two focused efforts; Web browsing and runtime services.
- To address this need, we will staff a browser team, whose mission is to go head to head with NetScape, competing with them as a product and an offering, and agressively providing new, rich features for both users and publishers. This team will focus on beating NetScape at their game, in their space.
- We will also staff a runtime services team, whose focus is to deliver the programming model, coupled with supporting services, which enables developers solve this class of problems. It is this area, and the integration with all aspects of Windows, which will enable us to flank the middleware vendors, fill this hole, and cement Windows as the platform of choice for the Internet.

Goal and Mission

The goal of the Internet Team is to cement Windows position as the premium client operating system for content and applications. While there are many factors which determine an OS purchase, fundamentally consumers purchase the system that runs the cool applications first and best. Windows is by far the leader in productivity applications, and to date this has determined purchase.

However, the Internet raises a new set of challenges, where content can be viewed on multiple operating systems equally well, and in some cases better on competing systems. By providing these services, we are trying to ensure that titles are developed on Windows, distributed for Windows first, and look better on Windows when there are multiple platform implementations. This is how we will measure our success.

The Key – Delivering User Benefit

As mentioned above, our team is in the platform business, and as such our primary deliverables are to ISV's, tool vendors, and content providers, who in turn produce the content and applications that drives consumer purchase. At the end of the day, however, our success will be measured by improved user experience - one that makes the customer choose to use our platform above others because they can do more things, quickly and easily.

We've tried to understand this need by focusing on these users - we've gone and talked to different product teams about how they think they will improve the user experience using the Internet in their product, and why delivering these products are difficult for them today.

Internet Titles

Web authors are all looking for ways to differentiate their site from others, and the main push has been interactivity. From HTML extensions which do animation of bitmaps to the latest craze over

Java applets, sites are currently using solutions which do not benefit the Windows platform. There are many examples of how users and publishers would benefit from this, but I'll list two below.

HTML Hosting Controls -- ESPN

The user gets a piece of mail about the great new ESPN page, which lets you watch a baseball game in real time. To jump to the page, they click on the shortcut, and are immediately taken there and it runs. To compensate for the low bandwidth pipe and to reduce server side load, the ESPN authors have created a custom control which understands how to take textual information about the status of the game, and render it graphically in real time on the user's machine. This control looks like a part of the HTML page, and the user never realizes that code has been downloaded automatically to their machine - they only know that suddenly they are watching a ball game on their machine in real time.

Secure, Real Time Internet Applications -- WinQuote

This same user might have a stock ticker application running, just like WinQuote from Microsoft today. The difference is, they received this application as part of a service purchased from PC Quote, where for a fee PC Quote provides secure, guaranteed access, with up to the minute information. The control allows the user to take this stream of information and choose the presentation they want, either graph, stock ticker, or history over time, as well as the stocks they want to track, and make these changes on a per user basis. Of course, this all works through existing firewalls, and the downloaded code is guaranteed not to have any viruses.

Reference Publishing -- Encarta

All reference titles are trying to figure out how to use the Web to increase the value of their products and their users experience, and basically want to enable the same user scenarios. They see the Web as three things: a real time publishing mechanism, a provider of valuable information, and longer term as their distribution mechanism.

Real Time Information

The user starts Encarta, looking for information. They click the What's New button in Encarta, and view the latest breaking items, published weekly on the Internet by the Encarta team. This information looks and feels like a part of the Encarta title - in fact, it is automatically included in all searches, just like any other topic in Encarta. The user doesn't even realize they've connected to the Internet, except that the content is progressively rendered to accommodate the low bandwidth pipe, and is also cached for quick reuse.

Linking to the Web

They then decide to get some information about California, because they are planning a trip. Encarta provides them with a crisp summary and overview, with links to other relevant information. Some of these links refer to the Encarta title, while others refer to information published by the California Department of Tourism, located on the World Wide Web. When the user clicks on these links, the content is launched in a new window, the Web browser - they are given the signal that they are on the Internet.

Real Time Subscriptions

Finally, the user purchases an ISDN line, and decides that instead of upgrading Encarta every year with a new CD, they instead want to subscribe to the online version, which is constantly updated for a low monthly fee. Whenever they want to search for information, they simply click the URL to Encarta. They are automatically validated and registered, and can use Encarta just as if it were on a CD-ROM. Again, they benefit from features that compensate for a low bandwidth pipe.

On Line Banking -- Money and Quicken

Today, the Money team uses an X.25 phone connection to dial in to banking for on line banking

services. They want to move this infrastructure to the Web in order to deliver services more quickly to customers, and provide a compelling sell to their banks.

Publishing Bank Information and Statements

The bank runs a Web site which has all of the customer statements on it, plus information about the latest rates, changes and offers. Customers can browse this site with any HTML viewer, regardless of what financial application they use. For customers who want to view their statement, the bank provides a secure transaction service that restricts access to the customers information.

Accessing this Information from Money

If the user is a Money customer, they can click a button in Money which displays the Web browser with updated bank information. Just like the Encarta example, the user can click Back in the browser to return to Money.

Receiving Electronic Statements

When the user wants to balance their checking account, Money automatically fetches the statement information from the bank – this statement is delivered in .MNY format, so it can be quickly integrated with the existing Money transactions.

Making Transactions

Finally, the bank offers transfer services on their site using STT or SSL, so the user can transfer funds from savings to checking real time with the click of a button.

Catalog Ordering

A user goes to visit the LL Bean site, and browses their catalog for information. The presentation of this information is super important to LL Bean – they have chosen to write a custom catalog using Director (or some other tool), rather than using HTML.

Browsing the Catalog

As they move through the pages of the catalog, they click a button next to an item when they want to add something to their order. They can view their order status to now at any time by clicking the order button on their toolbar. When they have finished shopping, they choose to exit the catalog, which brings up their custom order form, pre-filled in with order information.

Ordering Items

Within the form, they change the number of shirts they want to buy, which automatically updates the total. They also type in their shipping address and credit card number – of course all of the fields do validation to make sure that the information entered is correct. With a click the order is delivered to LL Bean, securely of course, so nobody can read the credit card number.

Processing the Order

At LL Bean, the order is automatically entered into a database, and a receipt confirmation is delivered. As the order progresses through LL Bean, updated information is sent electronically, providing information about status, shipping, and availability. The customer can also go to the LL Bean page, and fill out a form with their order number or name, which returns real time status about the order.

The Corporate Web – Office

Many corporations want to be able to publish the documents they have today on internal webs, and then browse them in a web-like style, without converting all of them to HTML – in particular, spreadsheets, presentations, and other document types which lose behavior and value on conversion.

Getting to the Web Site

A Microsoft employee wants to find out more about the Internet Explorer team. They click the Search button in their browser, and search for Internet Explorer, which returns a shortcut to the Internet Explorer site. On this site, they can view specs, see the RAID database, or look at the development schedule.

Browsing for Specs

When they click on the specs shortcut, it jumps to a list of the specs, with comments and descriptions below each one. Because the specs are written using Word, clicking on a shortcut to any document displays that document in the viewer frame, just like an HTML page. The user can click on Back to return to the top level spec folder, or continue to follow hyperlinks in the Word document as they use hyperlinks in HTML.

Viewing a Presentation

The user might then click on a presentation for an overview of the O'Hare team plans. However, the presentation is in PowerPoint format, and they don't have the PowerPoint viewer. Not to worry, because the browser automatically detects this, asks them if they want to download the viewer, then fetches the viewer from the corporate server and launches it with the presentation. Because it is a presentation, it is launched in a new window in play mode, with big buttons for previous and next, and cool transitions and effects. Once again, if the user clicks back, they return to the Internet Explorer Site.

Looking at Schedules

Finally, the user can click on the development schedule, which is launched in place in the browser. In this schedule, they can use Excel AutoFilter to search for the particular feature they are interested in, and view the status and remaining buffer time – all within the context of the browser.

Workgroup Forms – Exchange, MSN, and Notes

The Exchange team wants to deliver workgroup forms that are competitive with Lotus Notes. They are also extremely interested in figuring out how their solution scales to the Internet. Other teams at Microsoft are interested in these problems as well, specifically Ren and MSN, as well as external groups or publishers creating Web discussion databases.

Designing the Form and Folder

The corporation sets up a folder of information discussion and sharing for internal use. Because they want to guarantee some consistency, they design a set of forms used to enter information; one form for entering a new topic, another for responding to a topic. Each of these forms has unique behavior associated with it – fields are validated automatically as the user types in information, and the reply form inherits some fields from the entry form.

Installing the Form

When an user wants to enter a new piece of information, they go to the public folder and choose New Entry, which launches the entry form. If the form is not on their machine, it is automatically downloaded from the server and cached on their hard disk for future use – if the form is cached, it is checked against the server to ensure that the latest version is in use.

Changing the Form

At this point, the user might decide to change the way the form works, perhaps add or edit a field. Because the form description has been downloaded to their machine, they can switch to a form editor, make some changes, and then save the form back to the server. And, because the data binding is consistent, information entered in an older version of the form displays great in the new version.

Entering Information

To enter new information, the user fills in the form. As they fill in the form and choose controls, different features are enabled or disabled depending on their choices. For example, the name control is only activated after the user checks the Add My Name checkbox. When finished, they choose Done to post the data to the server. At this point, any other user can double click on the entry, which displays the form with the fields filled in and locked for editing.

Publishing on the Web

The corporation then decides to publish some set of this information externally on the WWW. Because HTML is popular, they want to be able to quickly convert both the form and the data into HTML, so that any viewer can view it – ideally this would be their native format, or close to it.

What's Wrong Today?

Many of these services can be delivered today, and in fact many people are delivering solutions which solve some set of these problems. In addition, these solutions are all available on Windows. So what's wrong?

Our Solutions Do Not Focus on Runtime Documents

Today's programming models and services do not focus on runtime documents – instead they are targeted at "Office-style" run and edit applications. There is nothing wrong with this approach – it has been widely successful *in that class of documents*.

In fact, even our internal customers are not using these services to create runtime applications. For example, MediaView implements it's own control scheme called embedded windows, even though they would prefer to use OLE controls, because the overhead of supporting these controls (mainly size) is not worth the benefit of the standard. And many people who want to use parts of OLE write their own implementation, with it's own quirks, because the cost of using "full OLE" is simply too high.

We Have No MacIntosh Story

If you look at the offerings that have been successful on the Internet, they have all been cross platform; in particular, QuickTime, Acrobat, Director, GIF, AU, and NetScape. New entries like Java differentiate themselves from our solutions mainly because of their cross platform ability, coupled with security. None of our product offerings in this space (OCX, VB, Web browser, AVI) provide true cross platform support.

Competitors are Filling This Void

We are so dominant in all other aspects of the market that we can never be displaced by a full frontal assault. However, when we do leave a hole in our strategy, there are many companies eager to move in and try to leverage this hole to grow into our other businesses. And they have: you only have to browse the Web to realize that NetScape, Sun, Apple, Adobe, and MacroMedia are establishing a presence.

The real threat to our business is solutions like Java, which present a different programming model than Windows and take developer and content provider mind share. This platform offering is quickly evolving, with two key players moving forward with their offerings and evangelism. In addition to Java, NetScape has announced an interface for plugging in different document types, while in turn Apple is building a programmable browser using OpenDoc.

The Result – People Aren't Writing to Our Interfaces

The solutions people have implemented today do not benefit Windows uniquely – they work on all

platforms equally well. More importantly, these solutions are being driven by other companies rather than our own – specifically, NetScape and Sun. Without an alternative to this platform we will lose control of a critical segment of the developer (and customer) market.

Our Proposal

We propose that we implement a set of services targeted to support the scenarios above. These services are designed to support *interactive applications*, running over *low bandwidth connections*, which gracefully *embrace and extend existing Internet standards and paradigms*. In the short term, we will deliver these as additional services on top of Win95 and Win96. Longer term, these services be folded back into the base parts of Windows, namely shell, core, and multimedia.

Web Browser Team

The main job of the Web browser team is to beat NetScape at its game, providing the industry standard Web browser.

Short Term Plans

For the next 3 months, this team will focus it's efforts on gaining market share with our existing code base. This is divided into three efforts: adding competitive features, increased distribution, and evangelism.

Competitive Features (O'Hare 1.1)

The Web browser team, led by JCordell and ArthurBl is focused on delivering Internet Explorer 1.1. The primary goal of this release is to make sure people do not switch from our browser to a competing solution (NetScape). The team hopes to accomplish this goal in two ways:

1. *NetScape features*
   We will add the key NetScape features which cause people to switch, namely tables and SSL. The key here is to remove features that publishers require, and therefore remove the incentive to recommend NetScape.
2. *HTML Enhancements*
   We will also add support for new HTML enhancements that we think will be attractive to publishers, specifically in line AVI, background sounds, and marquees. We will submit these to standards bodies, and work with marketing and DRG to evangelize these extensions broadly.

For more detailed information, see the product specification, or contact ArthurBl.

Increased Distribution (Corporations and ISP's)

Currently, the Internet Explorer is available in the Win95 OEM product, the Plus! package, and electronically for no charge. It will also be made available as part of the retail refresh later this year. JodyG is working to increase the distribution of our browser in the two remaining channels, corporations and Internet service providers (ISP's).

1. *ISP's*
   We will try to put together a compelling, brandable offering that let's the ISP set up the Internet Explorer to sign in to their service, and we will price this aggressively to gain share.
2. *Corporations*
   We will update the Windows resource kit with information on how to install the Internet Explorer. This includes making sure our browser works with standard proxies, but is mainly instructions and sample REG files for setting up our browser in a corporation.

We will add critical features to 1.1 where necessary to support this effort, but we don't want to defocus the 1.1 team. Wherever possible to use contract development and third parties to develop any

additional features required.

Evangelism

Finally, we will work to aggressively evangelize our Internet standards and extensions, and gain share in the Internet channel. This portion is less well defined, but ideas include a PSD/BSD Windows Internet Developers Conference, an effort to "seed" critical sites with NT servers, and an Internet Explorer evangelism effort that might include sample pages and sites, cool additions for Web pages, and education on Internet Explorer's capabilities.

Long Term Goals

In the next 12 months, the browser team will focus on how to entrench our browser as the browser of choice. The top priorities for this team:

- Provide a Web browser frame that can contain and navigate through different data types.
- Provide a viewer for HTML that is feature competitive with other offerings.
- Provide a way to add behavior to same through controls and scripts.

Web Browser/
Web Standards

[redacted]

Web Browser Frame

Today, Web browsers are monolithic applications which are linked into HTML. We want to open up the browser to a variety of data types, and set the stage for having that navigation migrate into the Windows shell. The API for embedding a viewer in our browser will be DocObj, plus extensions, although we may support a "runtime only" version of this which scales up gracefully to full DocObj depending on footprint.

HTML/VRML Viewers

This team will also supply two standard viewers which can be plugged in to the Web browser frame. These will implement the latest versions of HTML and VRML, and the goal is to have them be the preferred viewers of these formats.

One possible way the HTML viewer might be built is using the standard runtime components, per the picture at right.

[redacted]

Code, Controls, and Scripts

As part of this work, the team will add support in the HTML viewer to embed OLE controls and attach VBA scripts. They will also work with the Web standards bodies to define tags in HTML that allow authors to include controls and scripts as part of a page.

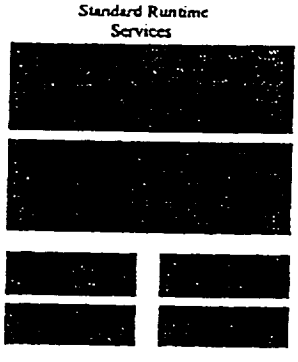
In doing so, this team will also understand Java and all of it's ramifications, and figure out the best way to embrace Java objects and code. Ideally, this would be through standard controls and/or a plug in viewer, but this may not be feasible.

Standard Runtime Services Team

Over the next 12-18 months, this team will deliver the set of runtime services needed to support these applications, focusing on two major areas:

- Providing Win32 services used by all classes of applications.
- Providing a runtime architecture designed for retrieve and render applications.

The sections below describe our initial thinking in this area, but we are in the process of defining exactly which of these we can reasonably deliver, and in what time frame.



Runtime Performance

Define the minimum subset of the control/container interfaces

We will work with these teams to define a base set of interfaces which is the *minimum* necessary to be a control or a container. We will also ensure that these gracefully scale up to "full blown" Office controls and containers.

Provide standard implementations of these interfaces

We will then take this set of interfaces and produce runtime interpreters for the base containers (2-D and 1-D streaming) and base controls (edit, radio button, picture, video, sound) which can be customized by tool vendors as their runtime environment.

Streaming for media types

We will make changes and enhancements to these services, where necessary, to support applications who need access to streaming media over the wire, like sound, graphics, and video. These may include quality of service guarantees.

Locate and Retrieve

Our initial thinking is that these fall into the following categories:

Protocols
Standard implementations of Web protocol services, including HTTP, FTP, and Gopher. This also includes secure versions of these services using SSL, and hooks to plug in other security mechanisms like STT. We will work with BSD to define a single API for these based on the Catapult API's.

Caching
Users of our protocols will also benefit from file level caching that is integrated with these. We will work with our clients to understand other caching needs they might have, including settings for expiring information in the cache, etc.

Link/URL/URN
A standard way to resolve a URL/URN and either launch the application for that link, or return a file handle to that data once it is fetched.

History, Favorites, Back, Next
System services which make it easy for applications to create and modify these entries, so that writers of different applications can participate in these mechanisms.

Scripting/Code

Code download
For those applications who want to fetch digitally signed code, we will provide a service which gets, extracts, and automatically installs those pieces when necessary, ideally by include a global registry (on the Web) where different viewers are registered.

Provide a standard VB runtime
We will work with the VB team as they define their 96 product plans, and ship a standard runtime interpreter for VB. We will provide this interpreter as part of our runtime, and support the execution of code behind controls and containers.

Plan for upgradability
The most important piece is that our components will be designed to "gracefully upgrade" to "standard" OLE controls and containers. If you drop a "full featured" OCX in our minimum container, we would like to swap in the code needed to handle the additional interfaces. We will work with the Forms', OLE, and VBA teams to design the right solution.

Benefits to Microsoft

The benefits to Microsoft here are that we fill the void in competing with NetScape, we move our platform forward and keep developers interested, we extend our existing OLE message to this new class of documents, and we reduce duplication of effort among internal groups.

Provide a Compelling Competitor to NetScape, Java, Apple

First and foremost, we'll deliver a compelling feature competitor to NetScape in 1996. They have proposed extensions to their client platform which allow for viewing of different data types in the frame, as well as adding behavior to HTML through Java. We will compete on three levels:

- At the browser level, with a team of people dedicated to providing a more compelling browsing solution, whose sole focus is to be the most popular browser.
- At the platform level, with a team of people thinking through how to enhance the basic services

delivered in Windows, to enable developers to deliver cooler content and a richer user experience quickly, easily, and on Windows first.

- And most importantly, at the OS level, as our services benefit *directly* from working together with the different feature teams to make Internet support an integrated part of all of our offerings, from the shell to kernel to user and GDI.

Keep Developer Mind Share

The second real benefit here is that we provide a path for developers which keeps them writing to Windows API's. There is a need in the marketplace for services which help to deliver this class of documents and applications – by investing in our platform we can define how Windows developers "go Internet" the Microsoft way.

Leverage Our Existing Evangelism and Investment

Visual Basic is the most popular language ever. OLE controls have been, and will continue to be, the big push for the Developer division moving forward. By delivering a true "runtime" version of these two pieces, we will attract a new class of developers to Windows centric solutions. In addition, because this solution scales from runtime to "Office-style" controls, we can leverage our existing tools and ISV education, and deliver a story which no other company can match.

Remove Internal Duplication of Effort

Last, but certainly not least, these services, implemented and designed correctly, will remove a tremendous amount of duplication of effort within our company. Many groups have come across this same set of problems, and because general services did not exist, they are writing and supporting their own. This has happened for control architectures (MediaView, SPAM), protocols (Word, VB), caching (BlackBird), forms (Exchange), and there are plans for HyperLink services (Office) and HTML controls (VB, MediaView). None of these groups want to be in the platform business, but they have to fill these holes to deliver the products their users are demanding. By solving a set of these problems in Windows, these groups can all spend time thinking about other things, and, almost more importantly, our products would look, feel, and work better together.

Schedule/Delivery

Our rough schedule is below – given buy off on the vision and goals, we feel we can execute on it. Today, there is a SweeperInfo alias set up with all interested clients on it – we've also scheduled biweekly client meetings to ensure that everyone is up to date and we resolve questions and issues quickly.

9.15	Design review
9.30.95	Design complete, first development schedule
10.1.95	Coding begins
Q495	M1 - Drop to Browser ITV, includes base container support, HTML hosting OCX's
Q296	M2 - Internet Developers Conference. Drop externally, coincide with CDK, early tool available.
Q396	Win96 - RTM final version.
Q197	Feature complete on architecture, includes safe scripting, TomFitr's tool available.

Why We Will Win

We will win for the same reasons Win95 will win – performance, ease of use, compatibility, and integration. While some companies may match some set of these features, nobody will come close to delivering what we can.

High Performance

First and foremost, we will win because we will have the best performance, period. One part of this is size and speed – you only have to look once at the hoops people are willing to jump through to achieve size and speed to recognize this. All of our services will be designed to use *the minimum* RAM necessary for any operation (our base container will be less than 50K). In those areas where we cannot improve the speed (like 14.4 modems) our resource management hooks will enable developers to get the information they need about what is going on in the system, and react accordingly.

Easy

A close second is ease of use, both for the end user and the developer. On the end user side, our browser will make the Internet easy, by automatically handles fetching the things the user needs to view Web pages, and seamlessly integrating different types of documents, and most importantly by leveraging the shell wherever possible so that the user has to learn on thing, one time only. On the developer side, people can choose their level – from simple applications who want to use the standard services, to complex applications which want to write a custom protocol, but still take advantage of our container and link services.

Compatible

Compatibility is a premium requirement, both for users and for developers. Our browser will be fully compatible with standard Web content – users will have no need to switch to view their favorite page. Our runtime services will be available on both Windows and the Mac, so that it is easy for developers to target both, of course developing for Windows first. Finally, for people writing controls or integrating with Forms[3] 96, we provide upward compatibility with these Office-style controls and objects.

Integrated

Finally, and perhaps most importantly, we will win because we are a part of the Windows team. When we need to provide a more integrated user experience, we will work together with the shell team to make sure that we just seem like part of the shell. When we need to have high performance in controls and containment, we will work with the user team to build in services designed to make us fast. When we need to have improved synchronization for multimedia, we will work with the Quartz team to figure out how to match our technologies. And long term, many of the deliverables we have for 96 will simply become part of the other feature teams, as Windows itself becomes the Internet platform.

Issues/Next Steps

Dependencies

Wherever possible, we are leveraging code and thinking from other teams, there are some deliveries which we are relying on to be successful.

VB	Size and speed improvements, Safe VB
Forms[3]	OCX 96, Forms[3] 96, specifically windowless controls
BSD	Catapult API's, WinSock, integrated security
Quartz	Media synchronization playback.

There are other teams where we have leverage working together, but if absolutely necessary we could ship without working together.

BlackBird	Potential code sharing
Office	Link services
CDK	Support for developing runtime and design time controls using a Wizard
Shell	Integration of browser and shell, code sharing for Shell OLE implementation

Safe code

Although mentioned as a dependency above, this point deserves a closer look. Currently Java is successful because it is cross platform and safe. Somehow, we have to embrace Java, by either pre-empting them with a compelling alternative (VB), by allowing users to view Java documents in our browser, or by hoping that they will go away.

Data fetching

Specifically, figuring out how to unify links, monikers, and URL's into one structure, and then delivering services which are factored so they can be used.

Code fetching

This is two problems – how do you find the code, and once you do, how do you know that it is safe?

Tools/editors that target our runtime

These services are useless if ISV's don't use them, and not just internal ISV's. If we are to be effective, we must have a set of tools which require our runtime, including a tool which edits HTML.

Related Reading

Internet Explorer 1.1 Spec – ArthurBl

Customer Requirements – ChrisJo