Browsers - from Static to Dynamic

Jan Newmarch `jan@ise.canberra.edu.au`
http://pandonia.canberra.edu.au

Last modified: 14 May, 1997. This tutorial looks at the ways in which browsers can be extended to allow interactive elements

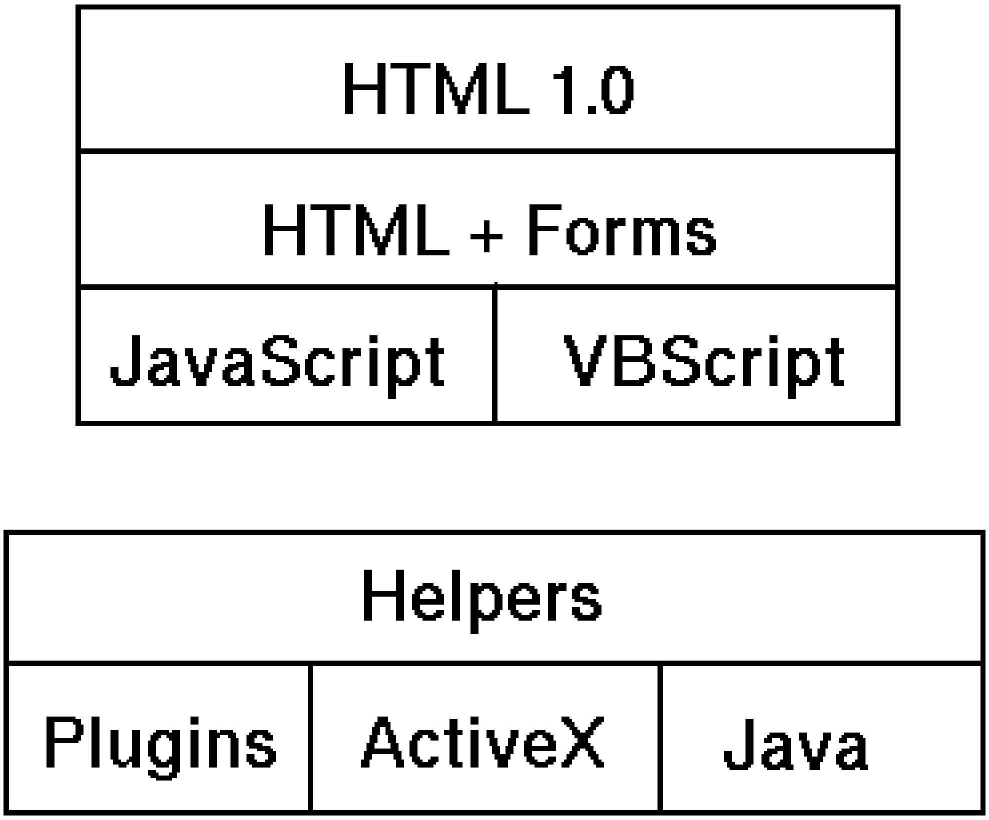
1. What this Session is About

- Browsers - from Static to Dynamic
- Forms
- JavaScript
- VBScript
- Helpers
- Plugins
- ActiveX
- Java

2. Browsers - from Static to Dynamic

- The Web is becoming the standard medium for both Internet and Intranet applications
- The Web browser is becoming the standard interface to both local and remote data sources
- The browser must deliver all the functionality of all applications
- The browser needs to move from passive display of static information to dynamic interaction

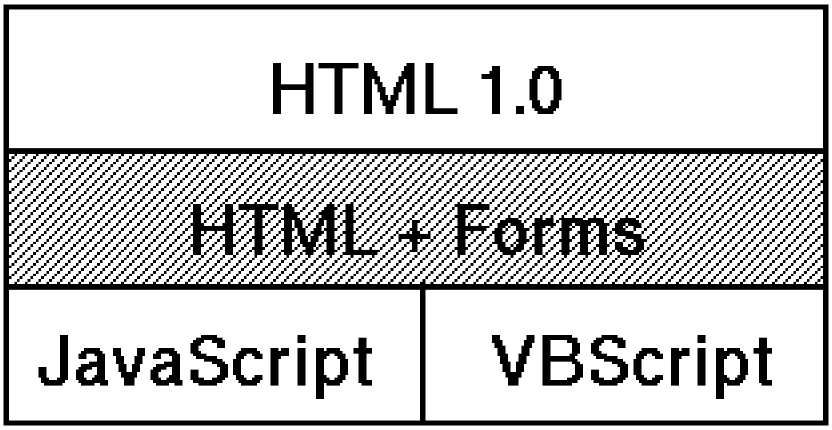
3. Moving to Interaction

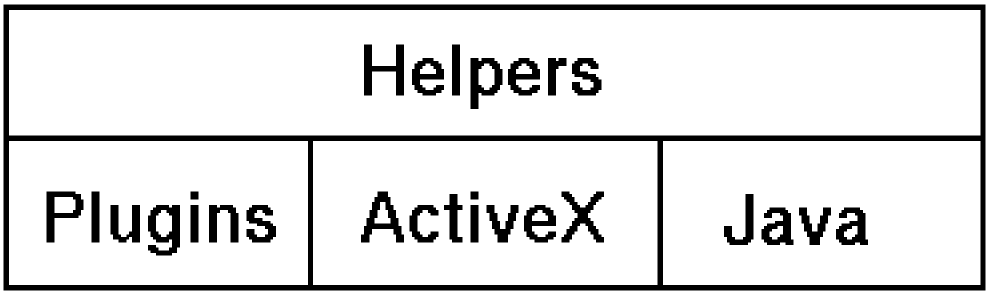


4. HTML 1.0

- Displays text and images, with limited formatting
- User navigates by hypertext links to other HTML documents
- No interactive elements
- This slide is in HTML 1.0

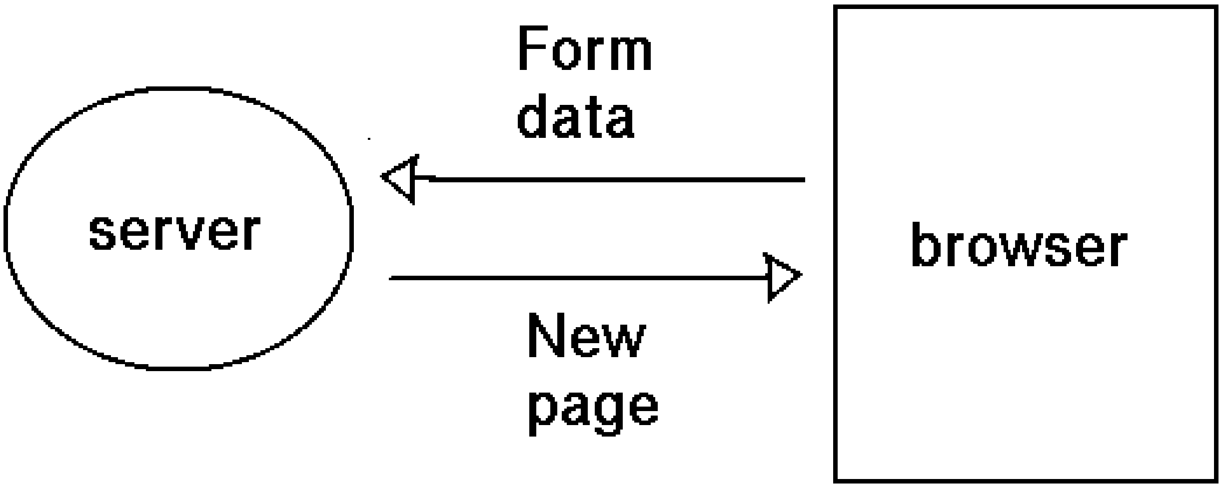
5. Forms





6. Forms

- Introduced in HTML 2, now standard in all browsers
- Designed for interaction between user and server



- Completed form is submitted to server for processing by e.g. CGI script

7. Form Elements

- HTML elements such as static text, images, static lists, etc may be part of a Form
- Interactive elements such as TextField, TextArea, List, Checkbox, etc allow user interaction
- ‘‘Submit’’ buttons cause transmission of information to server

8. Form submission methods

A Form can use

- POST - has side effects
- GET - idempotent

9. Information mangling

- Form information is transferred with the syntax

```
name=value&name=value...
```

- Spaces in values are translated into ‘+’
- Any funny chars are translated into ‘%XX’ where XX is the Ascii hexadecimal value of the character
- URL called has the information added after ‘?’

10. Form Limitations

- Small number of interaction elements
- The interaction elements have only a limited amount of customisation
- The interaction elements cannot be extended
- There is no client-side processing e.g. field validation
- All processing is performed on the server side

11. CGI Scripts

- Form submission usually processed by CGI scripts running on the server
- CGI scripts may be called directly by their URL, without needing a Form
- CGI scripts may be in any language
- CGI scripts are invoked by the server based on path name patterns - usually /cgi-bin/script

12. CGI information passing

- A script is invoked from its URL, but additional information may have been added to the URL
- The additional information is either set in the environment variable QUERY_STRING or (if unset) comes on standard input
- The information is in name/value form: name=value&name=value...
- The values are mangled

13. Extracting information

- The information string needs to be parsed and unmangled
- Libraries in C, tcl, Perl, etc, exist to do this - don't repeat work

14. Environment variables

- SERVER_SOFTWARE
- SERVER_NAME
- GATEWAY_INTERFACE
- SERVER_PROTOCOL
- SERVER_PORT
- REQUEST_METHOD
- HTTP_ACCEPT
- PATH_INFO
- PATH_TRANSLATED
- SCRIPT_NAME
- QUERY_STRING
- REMOTE_HOST
- REMOTE_ADDR
- REMOTE_USER
- REMOTE_IDENT
- AUTH_TYPE
- CONTENT_TYPE
- CONTENT_LENGTH

15. CGI Security - Server

Severe security issues on server side

- Form processing is performed with *some* access to server system
- Typical languages used are Perl, Shell, Tcl - weakly typed scripting languages with `eval` capabilities
- The CGI scripts can be attacked via the Forms, or directly

16. Form Security - Client

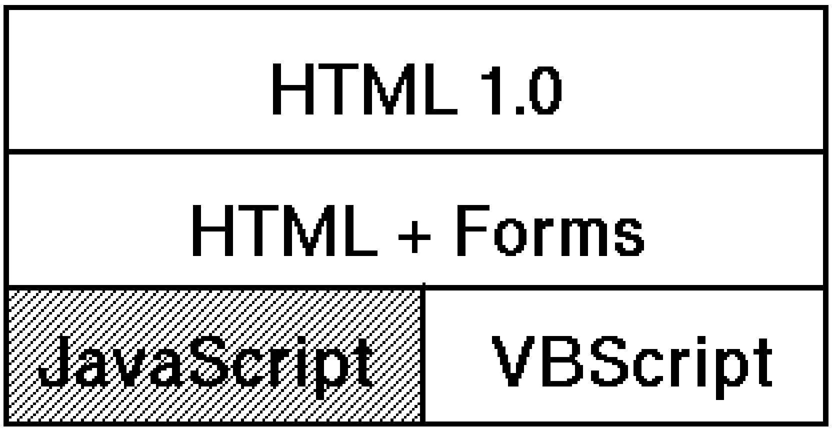
- No security issues on client side
- Very limited information transmitted to server side
  - Client host id known
  - Client user id unknown

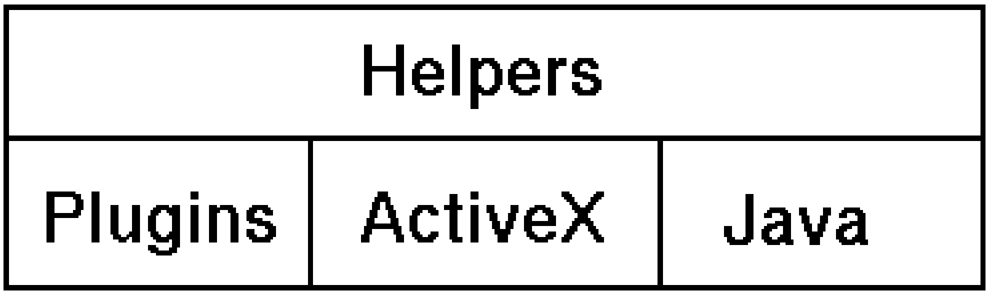
- If the client passes sensitive information to the server, security of the server is an issue to the client
- Users may be asked for *e.g.* PIN numbers - and may supply them

17. Form Availability

- Standard part of HTML 2.0 and later
- Available on most browsers
- Available on most platforms
- Knowledge in writing CGI scripts becoming more common
- CGI scripts being used to deliver dynamically created Web pages without explicit Form
- Refinements to CGI model being developed, to allow session control

18. JavaScript





19. JavaScript

- Scripting language for HTML elements
- Unrelated to Java (developed under the name of LiveScript)
- Object-based, but not O/O language - no inheritance
- Weakly typed language
- Uses an event processing model
- Proprietary to Netscape
- Supported by browsers using the `SCRIPT` tag

20. JavaScript HTML Elements

- Every HTML document has objects `window, location, history, document`
- Each HTML element in a page defines an object named by its NAME attribute, forming a hierarchy from `document`:

```
<FORM NAME="myform">
<INPUT TYPE="button" NAME="Button1">
</FORM>
```

defines objects including

- `document.myform`
- `document.myform.Button1`

21. JavaScript Event Model

- The following event types are generated
  - Focus, Blur, Change events: text fields, textareas, selections
  - Click events: buttons, radio buttons, checkboxes, submit buttons, reset buttons, links
  - Select events: text fields, textareas
  - MouseOver events: links
- If an event is generated for an object, then a method of that object can be defined to handle it

```
<INPUT TYPE="button" onClick="...">
```

22. JavaScript Location

- JavaScript function definitions may be given in a document's HEAD or BODY, using the SCRIPT tag

```
<SCRIPT language="JavaScript">
function bar() {
   document.write("Hello World");
}
</SCRIPT>
```

- Functions need to be declared before use, so place them in the HEAD
- Hiding code in comments is a kludge

- Event handlers are placed in the HTML tag for the object

```
<INPUT TYPE="button" onClick="bar()">
```

23. JavaScript non-HTML Elements

- The `window` object can open or close a browser window
- The `window` object can popup an `alert` dialog box
- The `window` object can popup a `confirm` dialog box
- Objects and methods can be defined
- String, Math, Date objects predefined

24. JavaScript Example

This example validates a TextField as containing numeric characters only

```
<script>
function validate(obj) {
  value = obj.value;
  for (n = 0; n < value.length; n++)
    if ( ! isDigit(value.charAt(n))) {
      alert("Field must be numeric");
      return;
    }
}
</script>
<input type="text" onChange="validate(this)">
```

25. JavaScript Security

- No formal security model
- Access to disk, network limited by non-availability of such JavaScript functions
- Security is dependant on correct implementation of browsers
- Previous bugs included
  - Read user’s history
  - Read user’s URL cache
  - Forge email
  - Recursively list local disks

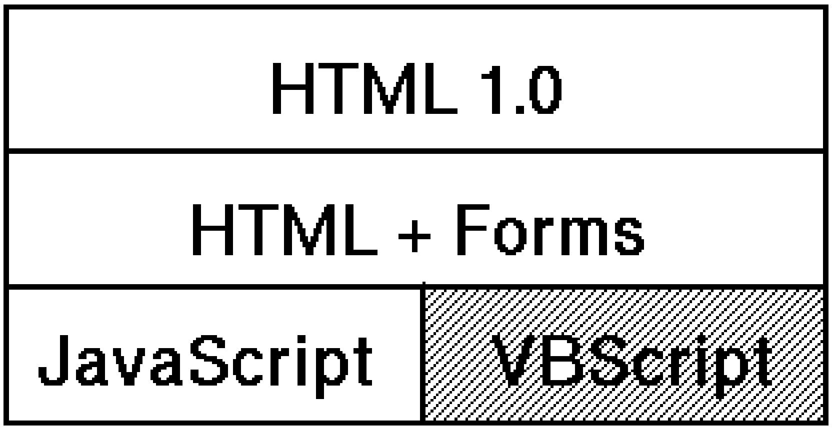
26. JavaScript Availability

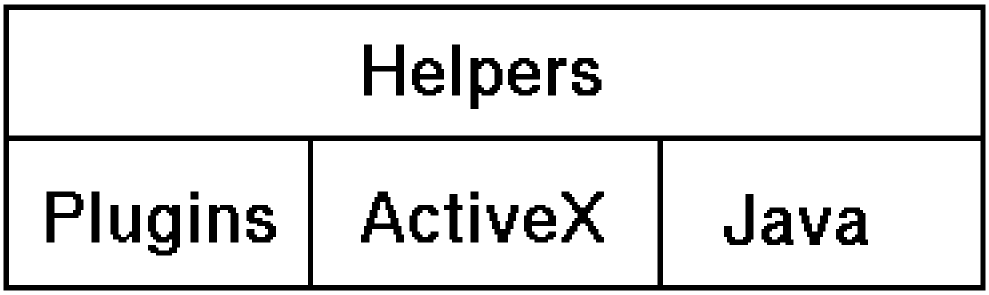
- Netscape, InterNet Explorer
- All platforms for these browsers
- Also available to some servers, but without HTML elements

27. JavaScript Communication

- JavaScript can call Java methods
- Java methods can call JavaScript

28. VBScript





29. VB Script

- Scripting language for HTML elements
- Same “space” as JavaScript
- Object-based, but not O/O language - no inheritance
- Weakly typed language

- Uses an event processing model
- Proprietary to Microsoft, but may change
- Supported by browsers using the `SCRIPT` tag
- Smallest member of VB family

30. VB Script event Model

- HTML elements are accessible to VB Script via their names
- Event procedures are labelled by event type appended to object name

```
<SCRIPT LANGUAGE="VBScript">
Sub Button1_OnClick
End Sub
</SCRIPT>

<FORM>
<INPUT TYPE="button" NAME=Button1">
</FORM>
```

31. VB Script non-HTML Elements

- InputBox
- MsgBox
- String, Math, Date handling

32. VBScript Example

```
<script language="VBScript">
Sub Text_LoseFocus
  If not IsNumeric(Document.F.T.Value)
  Then
    MsgBox "Field must be numeric"
  End If
End Sub

<FORM name="F">
<INPUT name="T" TYPE="text">
</FORM>
```

33. VBScript Availability

- Internet Explorer
- All Internet Explorer platforms

34. VBScript Security

- No formal security model
- Access to disk, network limited by non-availability of such VBScript functions
- Security is dependant on correct implementation of browsers

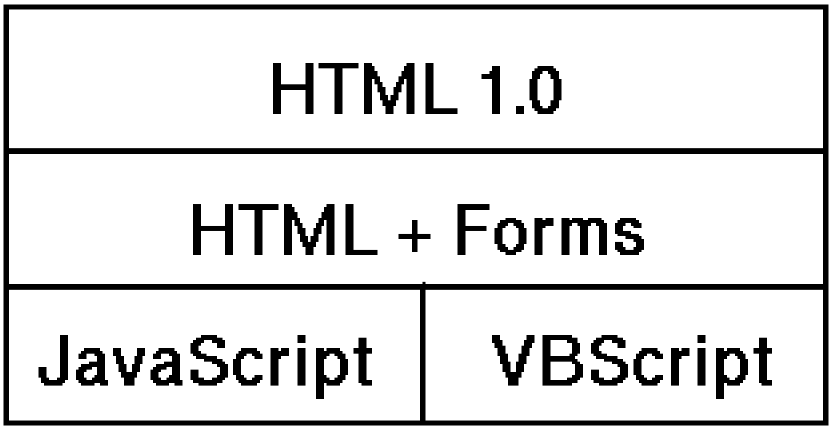
35. VBScript Communication

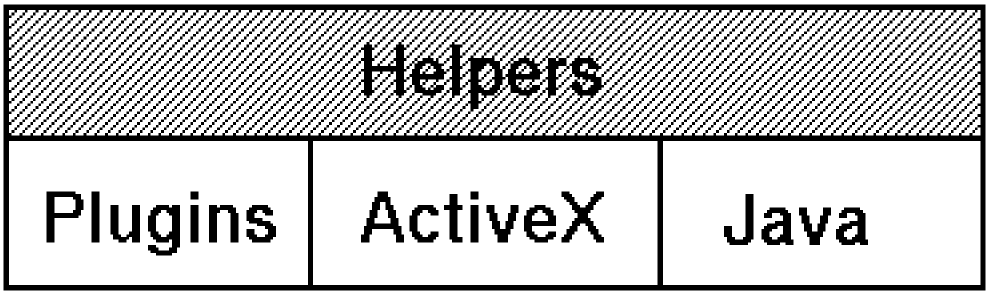
- VBScript can talk to
- ActiveX controls (formerly OLE controls)
- Java objects

36. Helpers and Plugins

- Documents are characterised by their MIME type
- This type is often obtainable through the filename extension
- Only some MIME types are handled by browsers
- Helpers and plugins extend this set in different ways

37. Helpers





38. Helpers

- Complete applications running on the client
  - Separate window space
  - Separate address space
- All of the document handed to the helper (causes delays)
- Limited interaction between browser and helper
  - Signals
  - X Window properties
  - Common Client Interface

39. Example Helpers

- VRML viewers
- MPEG viewers such as `mpeg_play`
- Postscript viewers such as `ghostscript`

40. Helper Setup

- Entries must be made in the MIME types files
- Needs knowledge of MIME types used and the helper to handle them
- More likely to be done *for* users than *by* them

41. Helpers and HTML

- Data files are referenced by normal HTML links
- The file extension gives the MIME type
- The MIME type gives the handler on the browser side

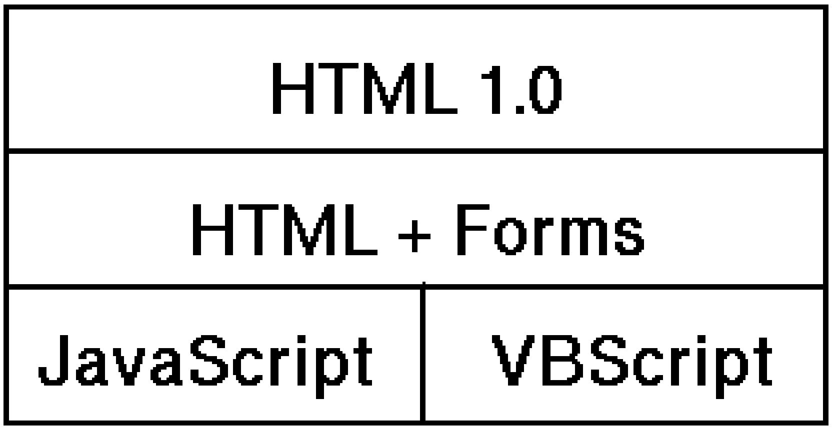
42. Helper Security

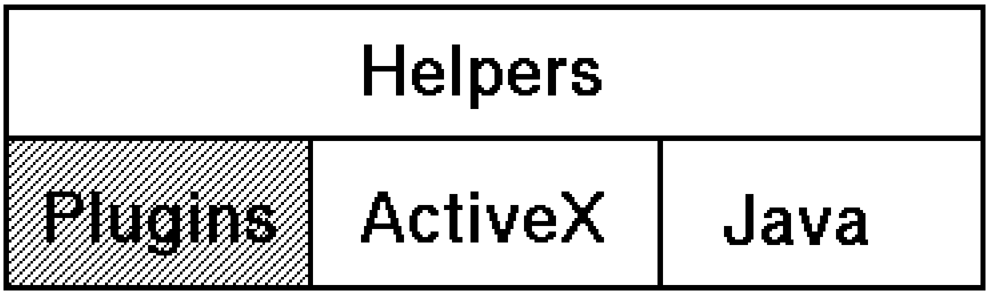
- The helper is a native code application
- It has access to all the client resources
- It has access to all network resources that the client has access to
- Installation and setup are under client control
- Some helpers are inherently dangerous
  - Postscript is a full programming language
  - It includes file access commands
  - A Postscript viewer can access your file system
  - There is no ‘‘safe’’ subset of Postscript run by viewers
- Similarly for Microsoft Word, and many other applications that accept scripting languages
- Users can setup their own handlers

43. Helper Availability

- A MIME type doesn’t require a specific program, just a handler for its type
- Platform-dependant handlers may or may not exist
- Handlers are written in any language
- Handlers are independant of any browser

44. Plugins





45. Plugins

- Plugins run as dynamic link libraries
  - Run within browser window space
  - Run within browser address space

- Tighter communication possible between browser and plugin
- Plugins are requested by a browser using the `EMBED` tag

46. Example Plugins

- Live3D - Netscape VRML browser
- CoolTalk - Netscape telephone
- Shockwave by Macromedia (Mac, Windows, Win95)
- Acrobat reader by Adobe (many platforms)
- ActiveX by NCompass (Win95 and Win NT)
- PowerBuilder by PowerSoft (Windows, Win95, Win NT)
- Tcl/Tk by Sun (Win, Mac, Sun OS, Irix, Linux)

Most plugins available for various Windows platforms, some for Mac, some for common Unix’s

47. Installation of Plugins

- Plugins can be installed manually by copying the plugin to the correct directory
- Plugins may have an automatic installation script
- Plugins may be downloaded across the Net using browser assistance

48. Plugin Availability

- Plugins are implemented by dll's
- dll's have a plugin interface
- Plugins must be specially contructed to meet this interface
- Platform dependant
  - native code
  - need O/S dll support

- Browser must support plugins *e.g.* Netscape and Internet Explorer

49. Plugin Communication

- Communication between plugin and browser is available via the dll interface
- dll interface is currently limited to
  - Registering MIME types
  - Drawing into a part of a browser window
  - Receiving keyboard and mouse events
  - Obtaining data via URLs
  - Posting data to URLs
  - Adding hyperlinks to new URLs

- Plugins can talk to Java

50. Plugin Security

- Plugins can use any browser capabilities
- Plugins can use any resource of the client
- Plugins can access any network resource that the client can access
- Plugins can be installed by any user

51. Plugins and HTML

Plugins are loaded when the browser comes across an <EMBED> ... </EMBED> tag with attributes

- HEIGHT, WIDTH, UNITS

- SRC - the URL of the plugins data file
- MIME - the MIME type of the tag, to choose the plugin
- PLUGINSPAGE - the URL of instructions for installing the plugin
- Other attributes are passed to the plugin

52. Browser to plugin methods

The following are some of the C functions that must be implemented in the plugin, as they may be called by the browser:

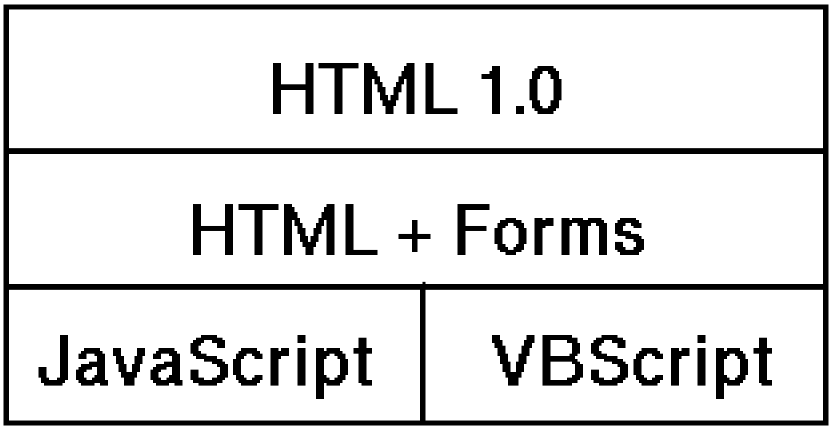
- NPP_Initialize() - plugin loaded
- NPP_New() - new isntance of plugin
- NPP_HandleEvent() - (mac only) handle a window event
- NPP_SetWindow() - the plugin is told of the window it is using
- NOO_Shutdown() - instance closed
- NPP_Destroy() - plugin exited
- NPP_GetJavaClass() - find the class of this instance

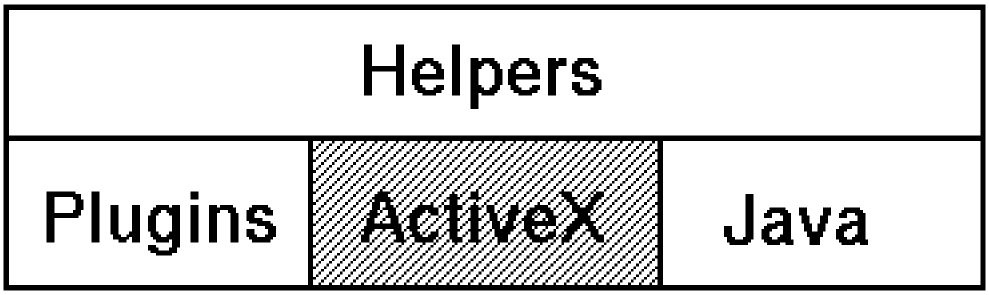
53. Plugin to browser methods

The plugin can call these methods in the browser:

- NPN_GetURL() - load a URL into Netscape
- NPN_PostURL() - post to a URL (CGI script)
- NPN_MemAlloc() - ask for memory
- NPN_MemFree()
- NPN_Write() - write to a Netscape stream
- NPN_RequestRead() - read from a Netscape stream
- NPN_GetJavaEnv() - Java env to make calls to Java using native interface
- NPN_GetJavaPeer() - the peer object for this Java instance

54. ActiveX





55. ActiveX

- Formerly known as OLE Controls or OCXes
- Embedded into browser using Microsoft's COM
- Uses the `OBJECT` tag within HTML

```
<OBJECT
  id=HelloWorld
  CODEBASE="http://www.microsoft.com/iemenu.ocx"
  classid="clsid:7823A620-9DD9-11CF-A662-00AA00C066D2">
</OBJECT>
```

56. Example ActiveX Controls

There are hundreds of controls, including

- Microsoft Forms controls include Label, Textbox, Combo Box, Toggle Button *etc*
- Chart
- Menu

- Timer
- Stock Ticker

57. ActiveX Availability

- ActiveX controls are available to *any* application supporting COM
- In particular, available to Internet Explorer
- Only on COM platforms (Windows NT, etc)
- Controls are written in any language
- Controls run as native code (or interpreted code, maybe)

58. ActiveX Security

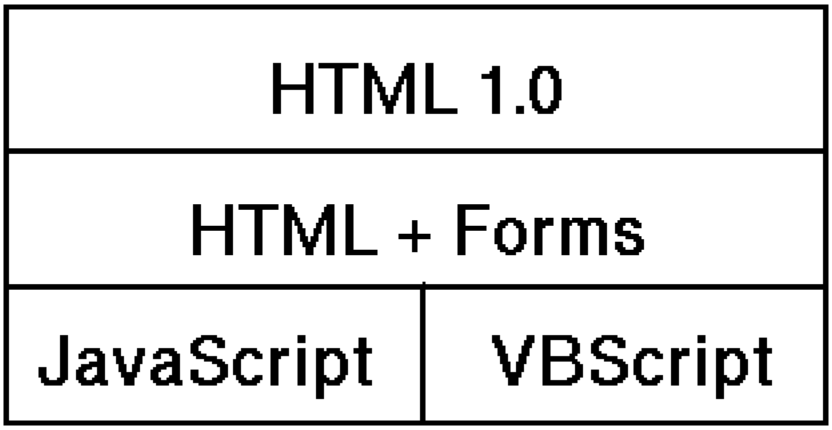
- Control is loaded from local disk if it is there
- If not found locally, then downloaded automatically
- Control is checked by
  - Publisher receives a certificate from a trusted Certifyinh Authority
  - Publisher encrypts certificate into document using their private key
  - Client uses publisher's public key to decode and check document

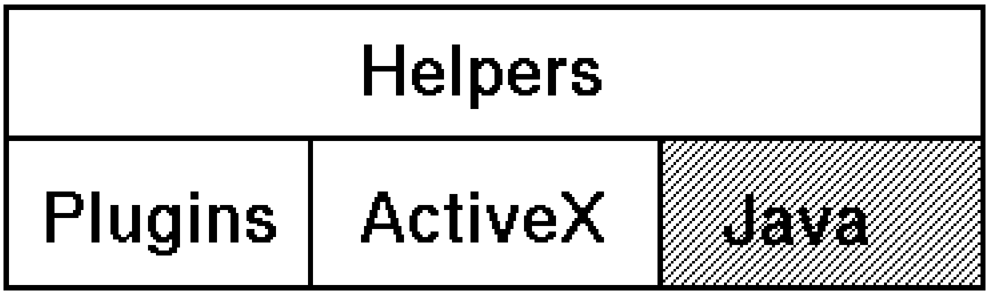
- "Exploder" (`http://www.halcyon.com/mclain/ActiveX`) has been through this process. It shuts down Windows95 cleanly, but could do worse

59. ActiveX Communication

- VB Script can invoke ActiveX controls, using its `id`
- Java is planned to be able to invoke ActiveX controls
- A control can fire events which can be caught by VBScript

60. Java





61. Java

- General purpose O/O language
- Platform independant - uses a virtual machine
- Class libraries are a mixture of Java code and native code
- Class libraries adopt a “common subset” model of native capabilities
- Java programs can be embedded in browsers using the `APPLET` tag

62. Java Language Features

- No pointer dangers
- No funny parameter mechanisms
- Automatic garbage collection
- Threads
- Weak multiple inheritance (specification, not implementation)
- No generics
- No low-level access
- Few low-level data types
- Intrusive exception handling

63. Java Libraries

Java standard class libraries support

- Graphics
- Animations
- Graphical User Interfaces
- Networking

64. Java graphics

- Restricted to `drawLine()`, `drawRect()`, etc
- No optimisation
- No tricky stuff
- No 3D stuff
- Is really "moving gif"
- No MPEG support
- No animation engine

65. Java GUI

- Java supports event-driven programming
- Java has a large set of GUI elements
- Uses subset model of all GUIs
- Many common GUI elements missing
- No drag and drop control
- No Object Link Embedding
- No Multiple Document Interface

66. Java Networking

- Applets can be embedded in Web documents
- Code can be downloaded on demand
- Security model restricts local capabilities
- Security model restricts distributed capabilities
- Security bugs have caused problems
- Charging models not clear
- Lacking transportability of Telescript

67. Java Future Libraries

Future libraries support

- Security
- Electronic commerce
- ODBC
- Development environments
- Internationalisation
- remote Method Invocation
- Persistence

68. Java Security

- Compiled VM code is checked by a byte-code verifier
- Access to many resources is controlled by a Security Manager
- Default Security Manager allows no local disk access or local process creation
- The native code libraries must be secure

69. Java Example

```
class MyText extends TextField {
  public boolean action(Event evt) {
    String str = getText();
    for (int n = 0; n < str.length; n++)
      if ( ! str.charAt(n).isDigit()) {
        new ErrorDialog("Must be numeric").show();
        break;
      }
    return true;
  }
}
```

70. Java Availability

- Most modern browsers
- All platforms supporting
  - Java VM (requires threads)
  - Standard java libraries (GUI, Networking)

- Most Unix's (including Linux), Windows NT, Windows95, Mac
- Java programs can also run standalone

71. Java Communication

- Applets within the browser can locate each other and communicate
- Access to the originating host is allowed
- Java can communicate both ways with JavaScript
- VBScript can call Java methods

72. Open?

- JavaScript, VBScript, ActiveX, Java are all proprietary products
- All have market weight to be called ‘‘standards’’
- Some vendors are more committed to developing standards than others
- Some vendors are more committed to following standards than others
- Many ‘‘standards’’ are still Internet Engineering Task Force ‘‘Requests For Comments’’
- Some ‘‘standards’’ may be accepted by standards bodies such as IEEE, but slowly

73. Further Information

- Java information is available from `http://java.sun.com` and `http://www.gamelan.com`
- JavaScript information is available from `http://home.netscape.com`
- VBScript and ActiveX information is available from `http://www.microsoft.com/intdev/sdk/actlogo-f.htm`

74. Conclusion

- Plugins and ActiveX controls are really only available for Microsoft platforms
- JavaScript is available for at least two major browsers on many platforms
- Java is available for many browsers on many platforms
- Large number of developers already know VB and ActiveX
- Large number of developers learning about Java
- Java security is tunable, but flexible support needed by browsers
- Network computer has to reach market and gain acceptance