Abteilung für Wirtschaftsinformatik

An Introduction to Procedural and Object-oriented Programming (Object Rexx) 7

"Automating Windows and Windows Applications"

OLE-Automation/ActiveX-Automation,
the Object Rexx Proxy Class "OLEObject",
Examples, Microsoft Internet Explorer's DHTML

Prof. Dr. Rony G. Flatscher

OLE (ActiveX) Automation, 1

- COM
  - Component Object Model
    - RPC ("remote procedure call")
    - Interfaces (e.g. "IUnknown")
    - Developed further
      - DCOM, COM+
- OLE
  - Object Linking and Embedding
    - COM-based
    - Linking of documents (dynamic data exchange, DDE)
      - Cold link
      - Warm link
      - Hot link
    - Embedding of alien/foreign documents

OLE (ActiveX) Automation, 2

- VBX, OCX, ActiveX
  - Set of COM-Interfaces for defining Windows "components"
    - Windows programs, which can be combined as building stones
    - Defined interface for communicating with components
  - Acronyms
    - Visual Basic Extension
      - Mostly developed for GUI
    - Object Component Extension and ActiveX
      - Independent of Visual Basic, therefore usable for all Windows programs

OLE (ActiveX) Automation, 3

- OLE (ActiveX) Automation
  - Interface for remote controlling Windows applications/ components
  - Set of COM-based Interfaces
    - Standardized definition of programming interfaces for (scripting) languages
      - Invocation of functions ("methods") in a Windows program
      - Querying and setting of variable values ("attributes") of a Windows program
      - Intercepting of events, which are raised in a Windows program
    - Logging of user actions, which later can be repeated in form of a generated scripting program ("macros")

OLE (ActiveX) Automation, 4

- OLE, ActiveX
  - Information about Windows applications and components is usually stored in the Windows Registry
    - HKEY_CLASSES_ROOT
      - CLSID
        - GUID resp. UUID
        - Global resp. Universal Unique Identifier
      - ProgID
        - A string that can be easyly comprehended and memorized by humans
        - VersionindependentProgID
  - Addressing can be carried out via CLSID, PROGID or a "moniker" (a human readable string)

The Object Rexx Class ".OLEObject", 1

- "Proxy" class for addressing OLE resp. ActiveX Windows programs, which allows for:
  - Locating and addressing running OLE/ActiveX programs
  - Creating new instances of OLE/ActiveX programs
  - Querying the published programming interfaces, attributes, constants and events
  - Addressing (invoking) of published programming interfaces by sending the respective Object Rexx messages
    - Arguments are automatically converted by Object Rexx
    - Return values are automatically converted by Object Rexx

The Object Rexx Class ".OLEObject", 2

- Querying/setting of attribute values using the respective Object Rexx messages
  - Cf. Directives for attribute methods, which generate a getter and a setter method (possesses a trailing equal sign)
- Reacting upon events, by invoking the Object Rexx methods defined for the events on the Object Rexx side
- Automatically carries out the datatype conversion for Windows and Object Rexx, ie. between Object Rexx and the following (COM) Windows datatypes
  - VARIANT, VT_EMPTY, VT_NULL, VT_VOID, VT_I1, VT_I2, VT_I4, VT_I8, VT_UI1, VT_UI2, VT_UI4, VT_UI8, VT_R4, VT_R8, VT_CY, VT_DATE, VT_BSTR, **VT_DISPATCH**, **VT_VARIANT**, **VT_PTR**, VT_SAFEARRAY

The Object Rexx Class ".OLEObject", 3

- Methods
  - Init(ProgID | CLSID [, NOEVENTS|WITHEVENTS])
    - Creates a new instance of the OLE/ActiveX program ("COM class")
    - Returns an Object Rexx (proxy) object for it
  - GetObject(Moniker [, SubklasseVonOLEObject])
    - Class method, which searches an existing instance of a COM class
    - Returns an Object Rexx (proxy) object for it
  - GetConstant([ConstantName])
    - Returns the value, which got predefined under the name ConstantName
    - Returns *all* published constants with their defined values as a Rexx stem

The Object Rexx Class ".OLEObject", 4

- Methods
  - GetKnownEvents, GetKnownMethods
    - Returns a stem containing all published events and methods of the COM class
  - GetOutParameters
    - Returns an array object with the "out" parameters of the last invocation
  - Dispatch(MessageName [, Argument1, Argument2, ...])
    - Invokes a method on the Windows side which has the same name as "MessageName" and forwards any supplied arguments

The Object Rexx Class ".OLEObject", 5

- Methods
  - UNKNOWN(MessageName [, ArrayWithArguments])
    - This method processes all unknown messages and forwards them to the OLE/ActiveX program, which gets represented by the proxy object
      - Pay attention for Windows messages that carry the same name as the Object Rexx classes **OBJECT** resp. **OLEOBJECT** ! Such messages are processed on the Object Rexx side!
      - This is mostly a problem for Windows methods named "COPY" or "CLASS", for which methods also exist in the Object Rexx class OBJECT
      - Hence one needs to send directly the UNKNOWN message, supplying the Windows method name as the first argument, e.g.

```
proxy~UNKNOWN("COPY")
proxy~DISPATCH("COPY") /* since ooRexx 3.1 */
```

Examples, Hints

- The Windows version of Open Object Rexx (ooRexx) is delivered with numerous OLE/ActiveX examples, which can be found under the ooRexx installation directory:

```
?\ooRexx\samples\ole
```

- Attention!
  - The following foils only depict a subset of the supplied examples
  - Therefore please study and run all examples in all of the samples subdirectories
    - These examples do not change your computer settings permanently!

...\OLE\APPS\SAMP01.REX

```
/* create an object for IE */
myIE = .OLEObject~New("InternetExplorer.Application")

myIE~Width = 454
myIE~Height = 232

Say "Current dimensions of IE are:" myIE~Width "by" myIE~Height

/* set new dimensions and browse IBM homepage */
myIE~Width = 800
myIE~Height = 600
myIE~Visible = .True
myIE~Navigate("http://www.ibm.com")

/* wait for 10 seconds */
Call SysSleep 10

myIE~Navigate("http://www.ibm.com/news")

/* wait for 10 seconds */
Call SysSleep 10
myIE~quit
```

...\OLE\APPS\SAMP02.REX

```
WshShellObj = .OLEObject~New("WScript.Shell")

WshEnv = WshShellObj~Environment
Say "Operating system:" WshEnv["OS"]
Say "You have" WshEnv["NUMBER_OF_PROCESSORS"] "processor(s) of",
    WshEnv["PROCESSOR_ARCHITECTURE"] "architecture in your system."

Say "The following directories represent special folders on your system:"
Do Folder Over WshShellObj~SpecialFolders
  Say "   " Folder
End

Say "Creating a shortcut for NOTEPAD.EXE on your Desktop..."
Desktop = WshShellObj~SpecialFolders("Desktop")
ShortCut = WshShellObj~CreateShortcut(Desktop || "\Shortcut to Notepad.lnk")
ShortCut~TargetPath = "%WINDIR%\notepad.exe"
ShortCut~Save

WshShellObj~Popup("Processing of REXX script has finished!")
```

...\OLE\APPS\SAMP03.REX

```
WshNetObj = .OLEObject~New("WScript.Network")

Say "Computer Name:" WshNetObj~ComputerName
Say "User Domain:" WshNetObj~UserDomain
Say "User Name:" WshNetObj~UserName

Say "The following network drives are currently mapped:"
MappedDrives = WshNetObj~EnumNetworkDrives
Do i=0 To MappedDrives~Count/2 - 1
  Say "   Drive" MappedDrives[i*2] "is mapped to" MappedDrives[i*2 + 1]
End

Say "The following network printers are currently connected:"
Printers = WshNetObj~EnumPrinterConnections
Do i=0 To Printers~Count/2 - 1
  Say "   Port" Printers[i*2] "is connected to" Printers[i*2 + 1]
End
```

...\OLE\APPS\SAMP09.REX

```
excelObject = .OLEObject~new("Excel.Application")
Worksheet = excelObject~Workbooks~Add~Worksheets[1]
myTitles="ABCDEFGHI"

do j = 1 to 10
  do i = 1 to myTitles~length
    title = myTitles~substr(i,1)
    cell = Worksheet~Range(title||j) -- e.g. ~Range("A1")
    if j = 1 then do
      cell~value = "Type" title  -- header of first row
      cell~font~bold = .true
    end
    else if j = 10 then do -- final row? yes, build sums
      /* set formula, e.g. "=sum(B2:B9)" */
      cell~formula = "=sum(?2:?9)"~translate(title,"?")
      cell~Interior~ColorIndex = 24 -- light blue
    end
    else -- a row between 2 and 9: fill with random value
      cell~value = random()
  end
end

/* save sheet in default TEMP directory */
Worksheet~SaveAs( value("TEMP",,ENVIRONMENT)"\demo.xls")
excelObject~Quit
```

...\OLE\APPS\SAMP10.REX

```
fsObject = .OLEObject~new("Scripting.FileSystemObject")
allDrives = fsObject~drives
if allDrives = .NIL then do
  say "The object did not return information on your drives!"
  exit 1
end

do i over allDrives
  info = i~DriveLetter "-"
     /* show the DriveType in human-readable form */
  j = i~DriveType
  select
     when j=1 then do
            info = info "Removable"
          end
    when j=2 then do
            info = info "Fixed"
          end
    when j=3 then do
           info = info "Network"
          end
    when j=4 then do
            info = info "CD-ROM"
          end
    when j=5 then do
            info = info "RAM Disk"
          end
    otherwise
          info = info "Unknown"
  end

     /* append the ShareName for a network drive... */
  if j=3 then info = info i~ShareName
     /* ...and the VolumeName for the other ones */
  else if i~IsReady then info = info i~VolumeName
  say info
end
```

...\OLE\APPS\SAMP11.REX

```
/* Get stock price from IBM internet page with MS IE and OLE */

Explorer = .OLEObject~new("InternetExplorer.Application")
/* uncomment the next line if you want to see what is happening */
-- Explorer~Visible = .true
Explorer~Navigate("http://www.ibm.com/investor/")

/* wait for browser to load the page */
/* if the page is not loaded by then, an error will occur */
call SysSleep 5

/* obtain text representation of the page */
doc = Explorer~document           -- DOM document
body = doc~body                   -- get BODY
textrange = body~CreateTextRange  -- get TextRange
text = textrange~Text             -- get the contents

/* extract stock price information */
parse var text . "Current price:" stockprice "0d0a"x .
if stockprice = "" then stockprice = "<could not read stock price>"

/* end Explorer */
Explorer~quit

say "IBM stocks are at" stockprice"."
```

Excerpt from: ...\OLE\APPS\SAMP12.REX

```
/* instantiate an instance of the Internet Explorer */
myIE = .watchedIE~new("InternetExplorer.Application","WITHEVENTS")

myIE~visible = .true
myIE~navigate("http://www.ibm.com/")

/* wait for the OnQuit event of the browser to change */
/* the !active attribute of the REXX object to false  */
myIE~!active = .true
do while myIE~!active = .true
  call syssleep(2)
end
```

```
::CLASS watchedIE SUBCLASS OLEObject

/* ... Cut ... Lines deleted, please lookup the original file in your installation ! */

/* this is an event of the Internet Explorer */
::METHOD TitleChange
  use arg Text
  say "The title has changed to:" text

/* this is an event of the Internet Explorer */
::METHOD OnQuit
  self~!active = .false    -- terminates the waiting loop in main code

::METHOD !active ATTRIBUTE -- store the active attribute
```

...\OLE\APPS\SAMP14.REX

```
-- Initialize string to database path.
strDB = "c:\temp\newdb.mdb"

-- Create new instance of Microsoft Access.
appAccess = .OLEObject~new("Access.Application")

-- Open database in Microsoft Access window.
appAccess~NewCurrentDatabase(strDB)

-- Get Database object variable.
dbs = appAccess~CurrentDb

-- Create new table.
tdf = dbs~CreateTableDef("Contacts")

-- Create field in new table.

/* Please note how to access the constant.
   Microsoft documentation and the MS OLEViewer output
   these constants as dbText, dbBinary, etc. - the type library
   however prints them as DB_TEXT, DB_BINARY, etc.. Unless
   documentation is found why the names should be translated,
   the OLE code will *NOT* convert the names. */
fld = tdf~CreateField("CompanyName", appAccess~getConstant("db_Text"), 40)

-- Append Field and TableDef objects.
tdf~Fields~Append(fld)
dbs~TableDefs~Append(tdf)

appAccess~quit
```

...\OLE\ADSI\ADSI1.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")
myComputer = .OLEObject~GetObject("WinNT://"||ComputerName||",computer")

say "Standard properties of this computer:"
say left("Name:",10) myComputer~name

/* in this case, using myComputer~class would invoke the standard REXX */
/* method "Class", therefore the OLE objects' "class" method has to be */
/* called explicitly using the "Unknown" method (see documentation for */
/* details on this mechanism).                                         */
-- say left("Class:",10," ") myComputer~unknown("class",.nil)
-- since summer 2006 with ooRexx 3.1 the following is preferable:
say left("Class:",10) myComputer~dispatch("class")

say left("GUID:",10) myComputer~guid
say left("ADsPath:",10) myComputer~adspath
say left("Parent:",10) myComputer~parent
say left("Schema:",10) myComputer~schema
```

...\OLE\ADSI\ADSI2.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")      -- get ComputerName
UserID       = value("USERNAME",,"ENVIRONMENT" )         -- get UserName

userObject = .OLEObject~GetObject("WinNT://"||ComputerName||"/"||UserID||",user")

/* using the object property */
say "The full name for" UserID "is" userObject~FullName

/* using the standard get method for ADSI objects */
say "The full name for" UserID "is" userObject~Get("FullName")

say "Would you like to rename the full name (y/n)?"
pull answer

if answer = "Y" then do
  say "New full name:"
  parse pull answer

  /* set the property                                                  */
  /* as an alternative, the property can also be set with the standard put */
  /* method of ADSI objects:                                           */
  /* userObject~Put("FullName",answer)                                 */
  userObject~FullName=answer

  /* because properties are cached to avoid network calls, changing the    */
  /* properties of an object will only affect the cache at first.          */
  /* the object gets updated with the SetInfo method:                      */
  userObject~SetInfo

  say "updated the full name for" UserID
end
```

...\OLE\ADSI\ADSI3.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")

container = .OLEObject~GetObject("WinNT://"ComputerName||"/Administrators")

do member over container~members
  say member~dispatch("class") ":" member~name "[" member~description "]"
end
```

...\OLE\ADSI\ADSI4.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")

computerObject = .OLEObject~GetObject("WinNT://"||ComputerName)

computerObject~Filter = .array~of("Group","Service")

/* show only objects of type Group and Service: */
do item over computerObject
  /* avoid calling the CLASS method of the REXX object by using the */
  /* "unknown" mechanism (this calls the CLASS method of "item").   */
  say item~unknown("class",.nil) ":" item~name
  /* above is same as: say item~dispatch("class") ":" item~name */
end
```

...\OLE\ADSI\ADSI5.REX

```
myADS = .OLEObject~GetObject("ADs:")

do namespace over myADS
  say "Domains in" namespace~Name
  do domain over namespace
    if domain \= .nil then
      say "   " domain~name
    else
      say domain
  end
end
```

...\OLE\ADSI\ADSI6.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")

myDomain = .OLEObject~GetObject("WinNT://"||ComputerName)
mySchemaClass = .OLEObject~GetObject(myDomain~schema)

say "Properties for the" myDomain~name "object:"
say

if mySchemaClass~container = 1 then do
  say myDomain~name "may contain the following objects:"
  do i over mySchemaClass~Containment
    say "   " i
  end
end
else
  say myDomain~name "is not a container."

say
say "Mandatory properties:"
do i over mySchemaClass~MandatoryProperties
  say "   " i
end

say
say "Optional properties:"
do i over mySchemaClass~OptionalProperties
  say "   " i
End
```

...\OLE\ADSI\ADSI7.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")  -- get ComputerName
computer = .OLEObject~GetObject("WinNT://"||ComputerName)
/* create a new group */
newGroup = computer~Create("group", "REXX-TestGroup")
newGroup~Description = "A test group created with REXX"
newGroup~SetInfo

/* make sure the information in the object cache is up-to-date */
newGroup~GetInfo
say "Created new group" newGroup~Name
say "Description:" newGroup~Description; say
say "Creating 15 users in this group:"
say "User01..User15 with passwords demo01..demo15"
/* create several new users */
do i = 1 to 15
  /* create name and other information */
  userName = "User"right(i,2,'0')
  userFullName = "Demo User Number" i
  userDescription = "A demo user that was created with REXX"
  userPassword = "demo"right(i,2,'0')

  newUser = computer~Create("user", userName)
  newUser~FullName = userFullName
  newUser~Description = userDescription
  newUser~SetPassword(userPassword)
  newUser~SetInfo
  newGroup~Add(newUser~ADsPath)
end
```

...\OLE\ADSI\ADSI8.REX

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")  -- get ComputerName
computer = .OLEObject~GetObject("WinNT://"||ComputerName)

say "Removing the fifteen users..."
do i = 1 to 15
  computer~Delete("user", "User"||right(i,2,'0'))
end

say "Removing the test group..."

computer~Delete("group", "REXX-TestGroup")

say "done"
```

...\OLE\WMI\ACCOUNTS.REX

```
WMIObject = .OLEObject~GetObject("WinMgmts:{impersonationLevel=impersonate}")
userAccounts = WMIObject~InstancesOf("Win32_Account")

do instance over userAccounts
  say
  say "="~copies(16) instance~name "="~copies(16)
  do i over instance~properties_
    say left(i~name":",20,' ') i~value
  end
end
```

...\OLE\WMI\PROCESS.REX

```
WMIObject = .OLEObject~GetObject("winmgmts:{impersonationLevel=impersonate}")

say "Here is a list of currently running processes"

do process over WMIObject~InstancesOf("win32_process")
  say process~processid process~name
end
```

Further Links, 1

- Rexx Language Association (RexxLA)

  http://www.RexxLA.org/

  - Numerous additional links
- Object Rexx related OLE/ActiveX URL

  http://pragmaticlee.safedataisp.net/

  - Lee Peedin's ActiveX for Object Rexx page, a **must** to visit !
- OLE-/Active-X Query Tool written in (Open) Object Rexx
  - The following file contains all necessary explanations and URLs which you need:

    http://wi.wu-wien.ac.at/rgf/rexx/orx15/readme.html

    - The tool runs using Microsoft's Internet Explorer (Version 6.0 or later), just open the file "**`rgf_oleinfo.hta`**" (a DHTML application written in Open Object Rexx)

Further Links, 2

- A collection of Microsoft (!) scripts for (Open) Object Rexx

  http://www.microsoft.com/technet/scriptcenter/scripts/rexx/default.mspx

- A collection of Microsoft of administrative scripts for Windows
  - A self unarchiving Windows help file

    http://download.microsoft.com/download/.NetEnterpriseServer/Utility/1.0/NT5XP/EN-US/netscrpt.exe

    - Collection of Visual Basic scripts for administrative tasks
    - Simply transcribable to Object Rexx
      - Replace the dot in Visual Script programs with the (Open) Object Rexx message operator (the tilde): ~

- Skripting book for administrative Windows tasks

  http://www.microsoft.com/mspress/books/6417.asp

  - Book with content that can be applied to Windows XP as well

DHTML: Overview

- Terms
- DOM/DHTML
- Examples

Terms, 1 (Markup Languages)

- Tag
  - Enables one to use tags to embrace regular text
    - Opening tag (a.k.a. start tag)

      <some_tag_name>

    - Closing tag (a.k.a. end tag)

      </some_tag_name>
  - Allows for analyzing text, by noticing which parts of a text are surrounded ("embraced") by which tags
  - "Element"
    - The sequence "opening tag", text, "closing tag"

Terms, 2 (Markup Languages)

- Document Type Definition (DTD)
  - Defines the tags and their attributes, if any
    - Name (identifier) of the tag
    - Attributes for tags
    - "Content model"
      - Nesting of tags and the allowed sequence of tags
        - **Hierarchical structure !**
      - Allows to determine how many times an element may occur
  - "Instance" of a DTD
    - A document with text that got marked-up according to the rules defined in a DTD
    - A document that has been checked whether the DTD rules were applied correctly is named a "valid" document

Terms, 3 (Markup Languages)

- HTML
  - A markup language for the WWW
    - HTML-Browser
      - Parses a document marked up according to HTML
      - Formats the text, depending on the used tags
  - DTD
    - Version 4.01: three variants defined
    - SGML-based, hence it is possible to
      - Use any case for the tags and attribute names
      - Some closing tags can be omitted, if the end tags could be determined by the rules set forth in the DTD
      - It is possible to define exclusions

Terms, 4 (Markup Languages)

- XML
  - A slightly simplified version of SGML
    - Allows the definition of DTDs for markup languages
      - Since 2002 an alternative got introduced in the form of "XML Schema": `http://www.w3c.org`
    - Tag and attribute names must be written in exact case
    - End tags must be always given
    - Attribute values can now be enclosed within apostrophes/single quotes (') in addition to double quotes (")
    - It is possible to explicitly denote empty elements

```
<some_tag_name/>
```

Terms, 5 (Markup Languages)

- XML DTDs can be omitted
  - A matching DTD can be always inferred, if the document is "well formed":
    - All tags must be nested
    - Tags must not overlap
    - Start tags must have matching end tags
- Structure is always indepent of the formatting!
  - Cascading Style Sheets (CSS)
    - Allows to define formatting (layout) rules for elements
    - It is possible to define specific formatting (layout) rules for elements with attributes that have specific values or depending on the sequence of the elements

Terms (HTML)

- Text, marked up in HTML

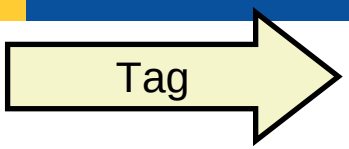
```
<html>
  <head>
     <title>This is my HTML file</title>
  </head>
  <body>
     <h1>Important Heading</h1>
     <p>This <span class="verb">is</span> the
        first paragraph.
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.
     <p id="9876">This <span class="verb">is</span> it.
  </body>
</html>
```

Example: Linking a Cascading Style Sheet (CSS)

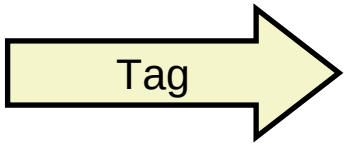
- Text, marked up in HTML

```
<html>
  <head>
     <title>This is my HTML file</title>
     <link rel="stylesheet" type="text/css" href="example2.css">
  </head>
  <body>
     <h1>Important Heading</h1>
     <p>This <span class="verb">is</span> the
        first paragraph.
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.
     <p id="9876">This <span class="verb">is</span> it.
  </body>
</html>
```

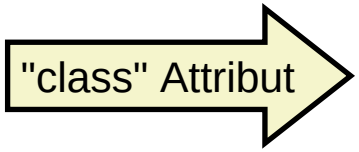
Example of a Cascading Style Sheets (CSS)



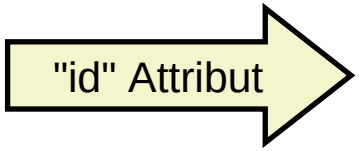

```
H1      { color: blue;
          text-align: center;
          font-family: Arial,sans-serif;
          font-size: 200%; }
```



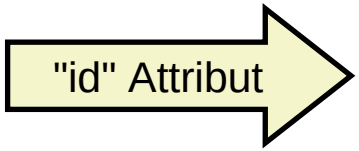

```
body    { background-color: yellow;
          font-family: Times, Avantgarde;
          font-size: small; }
```




```
.verb   { background-color: white;
          color: red;
          font-weight: 900; }
```




```
#xyz1   { font-variant: small-caps;
          text-align: right; }
```



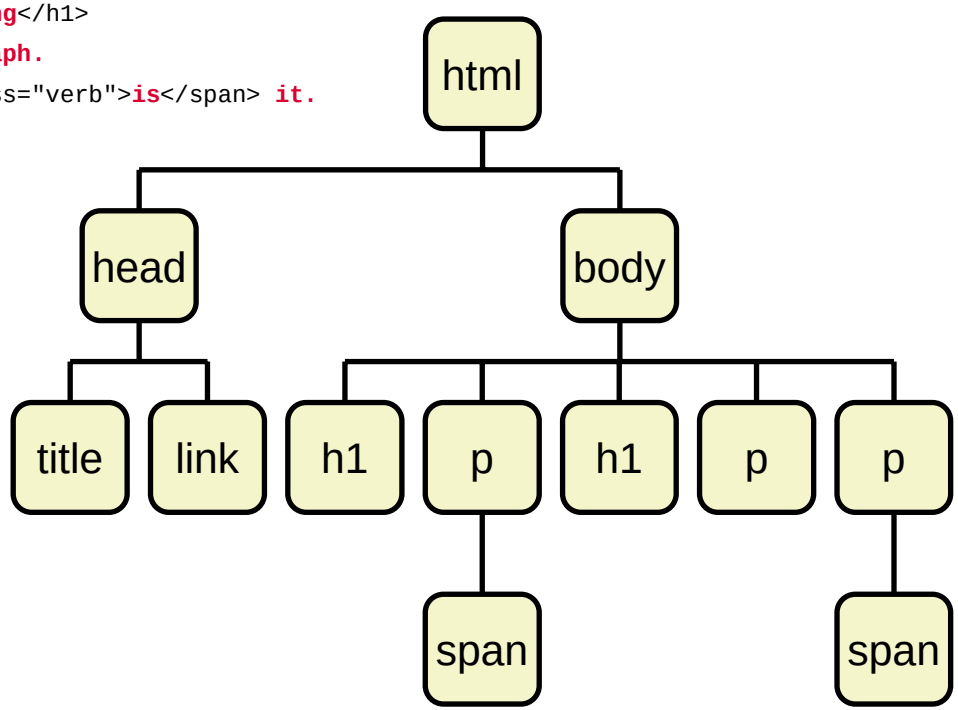

```
#9876   { font-size: large; }
```

Document Object Model (DOM)

- Parsing of a HTML/XML file
  - Creates a parse tree in which the elements are nodes
    - Each HTML browser needs to do this with each HTML file!
- Application Programming Interfaces (API) for
  - Creating, querying, changing and deleting nodes from the tree
    - This includes the manipulation of attributes as well!
    - Intercepting events, while the user is working with the parse tree
    - User generated events as a result of using the mouse or the keyboard
    - Application generated events like "document loaded"

Document Object Model (DOM) Example

```
<html>
  <head>
     <title>This is my HTML file</title>
     <link rel="stylesheet" type="text/css" href="example2.css">
  </head>
  <body>
     <h1>Important Heading</h1>
     <p>This <span class="verb">is</span> the
        first paragraph.
     <h1>Another Important Heading</h1>
     <p id="xyz1">Another paragraph.
     <p id="9876">This <span class="verb">is</span> it.
  </body>
</html>
```



Adding Programming Instructions (Scripts) to HTML/XML, 1

- <script>-tag

```
<script language="Object Rexx"> -- <![CDATA[
      ... Rexx code ...
    -- ]]>
</script>
```

  - orr:

```
<script language="Object Rexx" src="file.rex"></script>
```

- "on..."-attributes ("event attributes")

```
<some_tag onclick="call beep 90,90" language="Object Rexx">
```

  - One can directly call any previously defined public routine (keyword PUBLIC)
    - "**this**"
      - MS IE Object, which represents the element in which the event got raised
      - Can be used as an argument for the invoked routine

Adding Programming Instructions (Scripts) to HTML/XML, 2

- Microsoft Internet Explorer (MSIE)
  - Windows Script Host
  - Supplies the "window" object upfront
    - A COM object implementing DOM
      - "DHTML": dynamic HTML
    - Also all attributes of "window" are registered MSIE upfront, e.g. "document"
      - Represents a HTML/XML document
      - Possesses functions/methods to get e.g. all elements (nodes) in document order **all()** or all **tables()**
      - Allows the insertion, updating and deleting of elements (nodes)
  - Controls the execution of embedded script programs
    - Extracting, invoking the respective program statements

Adding Programming Instructions (Scripts) to HTML/XML, 3

- Programming statements can be interspersed anywhere in the document
  - Execution occurs in the order, in which the statements are found in the document ("document order")
  - Public ooRexx routines can thereafter be invoked from everywhere, even from scripts in other programming languages!
  - Program statements that are given in event attributes are executed, whenever the event occurs ("fires")

Example: HTML-File with Embedded ooRexx Code, 1

```
<html>
  <head>
     <script language="Object Rexx">
        document~writeln("<title>Produced by Rexx #1</title>")
     </script>
  </head>
  <body>
     <script language="Object Rexx">
        document~writeln("It is:<em>" date("s") time()"</em>, isn't it?")
     </script>
  </body>
</html>
```

Example: HTML-File with Embedded ooRexx Code, 2

```
<html>
  <head>
     <script language="Object Rexx">
        document~writeln("<title>Produced by Rexx # 2</title>")
        ::routine info public -- to be called later on
           use arg o
           window~alert("this=["o~tagName"] innerText=["o~innerText"]")
           window~alert("this=["o~tagName"] innerHtml=["o~innerHtml"]")
           window~alert("this=["o~tagName"] outerHtml=["o~outerHtml"]")
           tmpStr=""
           do item over document~all  -- iterate over all elements
              tmpStr=tmpStr item~tagName
           end
           call alert "elements:" tmpStr
     </script>
  </head>
  <body onclick="call info this" language="Object Rexx">
     <script language="Object Rexx">
        document~writeln("It is:<em>" date("s") time()"</em>, isn't it?")
     </script>
  </body>
</html>
```

Security Considerations

- MSIE
  - "Sandbox"
  - (Open) Object Rexx uses its own "Security Manager", to e.g.
    - Inhibit the access of the environment
    - Scripts can effectively be restricted to the login directory, etc.
- Secure, local execution
  - Rename the file type from "htm" or "html" to "hta"
    - "HTML Application"

DHTML: Wrap-Up

- HTML/XML Files
  - Hierarchical (parse tree)
  - Querying, inserting, changing and deleting of elements in the parse tree
  - Combine programming statements with events
- DOM
  - Document Object Model, W3C
  - DHTML
    - Microsoft's implementation of DOM
    - Incomplete, proprietary extensions
- Can be employed for the creation of GUIs and for printing!

Concluding DHTML-Example (Input and Output)

```
<head>
    <title>Object Rexx: Process User Input</title>

    <script language="Object Rexx">
        -- queries input value and outputs that string in reversed form
        ::routine work public
          output~innerhtml=reverse(input~value)
    </script>

</head>

<body>
    Please enter some text:<p>

    <input id="input" type="textarea" size="100"> <BR>

    <input type="BUTTON" onclick="call work" language="Object Rexx"
           value="Please Press Me!"><p>

    <p id="output">
</body>
```

Further Information

- World Wide Web Consortium

  http://www.w3c.org

  http://www.w3c.org/Style/CSS/

  http://www.w3c.org/DOM/

  http://www.w3c.org/MarkUp/ (HTML)

- Microsoft

  http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/dhtml.asp

  http://people.freenet.de/JavaScript/javap00.htm (Javascript-basiert)

- SelfHTML
  - Excellent, easy to understand resources about HTML, CSS, XML, DOM, …!
    - Unfortunately not in English, yet the examples are understandable
  - Tutorial
  - URL

    http://www.selfhtml.org/

Problem: Windows XP Service Pack 2 and MSIE

- Windows XP Professional, Service Pack 2, 2004-08-03
  - For "security reasons" MSIE is not able anymore to be automatically scripted with WSH scripting languages
    - Controlled via a Windows registry entry
    - (Open) Object Rexx examples do not run anymore
  - (Open) Object Rexx MSIE scripts are executed using the ooRexx "Security Manager"
    - Secure execution is therefore guaranteed
- Possible solutions
  - Change the file type from ".htm" resp. ".html" to ".hta"
    - "HTML"-Applications: have the same rights to the Windows system as any other application the user executes
  - Changing of the respective reigstry entry
    - Allow (again) Active Scripting under MSIE

Problem Solution: Change a MSIE-Registry Entry

```
/* ---rgf, 2004-09-15
   purpose: enable/disable Active Scripting in MSIE

Object Rexx version of:
   From: "mayayana" <mayaXXyana1a@mindYYspring.com>
   Newsgroups: microsoft.public.scripting.wsh
   Subject: Re: Please confirm: No future for Scripting of IE?
   Date: Tue, 17 Aug 2004 15:39:27 GMT
   NNTP-Posting-Date: Tue, 17 Aug 2004 08:39:27 PDT
*/

parse upper arg switch .
bEnable= (switch = "") | (switch = 1) | (switch~left(3) = "YES") | (switch~left(2) = "ON")
if bEnable then iVal=0
           else iVal=1

SH = .oleObject~new("WScript.Shell") -- allows to use registry Windows-style

   -- define local part of the key in the registry pointing to IE
sReg = "Software\Microsoft\Internet Explorer\Main\" || -
       "FeatureControl\FEATURE_LocalMachine_Lockdown\IExplore.exe"

SH~RegWrite("HKLM\"sReg, iVal, "REG_DWORD")
SH~RegWrite("HKCU\"sReg, iVal, "REG_DWORD")

if iVal=0 then say "MSIE enabled for Active Scripting."
          else say "MSIE barred from Active Scripting."
```