Novell exteNd Composer

5.2

www.novell.com

ECMASCRIPT API GUIDE

Novell®

Legal Notices

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.

www.novell.com

exteNd Composer *ECMAScript API Guide*
June 2004

Online Documentation: To access the online documemntation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

ConsoleOne is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Composer is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc.
Novell is a registered trademark of Novell, Inc.
Novell eGuide is a trademark of Novell, Inc.

SilverStream Trademarks

SilverStream is a registered trademark of SilverStream Software, LLC.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Third-Party Software Legal Notices

The Apache Software License, Version 1.1

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JDOM.JAR

Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution. 3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org. 4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)." Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Sun

Sun Microsystems, Inc. Sun, Sun Microsystems, the Sun Logo Sun, the Sun logo, Sun Microsystems, JavaBeans, Enterprise JavaBeans, JavaServer

Pages, Java Naming and Directory Interface, JDK, JDBC, Java, HotJava, HotJava Views, Visual Java, Solaris, NEO, Joe, Netra, NFS, ONC, ONC+, OpenWindows, PC-NFS, SNM, SunNet Manager, Solaris sunburst design, Solstice, SunCore, SolarNet, SunWeb, Sun Workstation, The Network Is The Computer, ToolTalk, Ultra, Ultracomputing, Ultraserver, Where The Network Is Going, SunWorkShop, XView, Java WorkShop, the Java Coffee Cup logo, Visual Java, and NetBeans are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Indiana University Extreme! Lab Software License

Version 1.1.1

Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact http://www.extreme.indiana.edu/. 5. Products derived from this software may not use "Indiana University" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Phaos

This Software is derived in part from the SSLavaTM Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved. Customer is prohibited from accessing the functionality of the Phaos software.

W3C

W3C® SOFTWARE NOTICE AND LICENSE

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications: 1.The full text of this NOTICE in a location viewable to users of the redistributed or derivative work. 2.Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the W3C Software Short Notice should be included (hypertext is preferred, text is permitted) within the body of any redistributed or derivative code. 3. Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

Contents

1 Composer Utility Methods

Top-Level

The following utility methods and properties are top-level calls, available in any Composer component, wherever ECMAScript can be used. The methods that are described as Composer component methods must be called on `theComponent` (see description of `theComponent` further below).

ERROR

Composer extension object. This object contains the error message and `_SystemFault` part thrown by Composer whenever an exception occurs. Or it contains the custom Error Message from a Throw Fault action. Has methods `getFaultName()`

and `getFaultPart()`. Use it in the Source Expression of a Map action or the Log Expression of a Log action.

base64Decode(encodedString)

byte[] base64Decode(encodedString) Composer extension method. This method decodes base64 encoded contents. Returns byte array of decoded binary content. encodedString parameter is of type String (base64 encoded data).

base64Encode(bytes)

String base64Encode(byte [] byteArray) Composer extension method. This method base64 encodes input parameter. Returns base64 encoded string. bytes parameter is of type byte[].

exportObject(key, value)

exportObject (String key, Object value) Interpreter extension—Composer components. Publishes an object to other called components that use the getExportValue(key) method. key is a string and value is an ECMA or Java object. Example:

```
theComponent.exportObject("pubVar", localVar);
```

getExportValue(key)

Object getExportValue(key) Interpreter extension—Composer components. Retrieves an object published from another component that used the exportObject(key,value) method. key is a string value. Example:

```
myVar = theComponent.getExportValue("pubVar");
```

getFaultName()

String getFaultName() Composer extension method. Returns the name of the fault contained in the ERROR object.

getFaultPart()

Document getFaultPart() Composer extension method. Returns the fault part contained in the ERROR object.

getLDAPAttr(connection, DN, attrName)

String getLDAPAttr(String connection, String DN, String attrName) This function returns the value of an attribute from a directory object. Parameters: string connection – name of a LDAP Connection Resource string DN – Distinguished Name of a directory object string attr – name of attribute (value to be returned) Example:

```
addr = getLDAPAttr('myCNX','cn=AClark,o=NOVELL','eMailAddress')
```

getName()

String getName() Interpreter extension—Composer components. Returns the name of the currently executing component. Must be called on `theComponent`.

```
theComponent.getName()
```

getSessionValue(key)

String getSessionValue (key) Interpreter extension—Composer components. Obtains a given value previously exported via the putSessionValue() call (see below). Values are available to all components that are called in the same Servlet Session, which may span many html hits. The session life is dependent upon the HTTP Server Session timouts. Please note that this call raises when error when Composer is deployed as an EJB or as a JMS message listener. The key is a String.

getValidationViaSchemaLocation()

getValidationViaSchemaLocation() Interpreter extension—Composer components. XML documents which have xsi:schemaLocation or xsi:nonamespaceschemaLocation can use this flag to check whether validation of such documents is done.

putSessionValue(key, value)

putSessionValue (String key, String value) Interpreter extension—Composer components. Exports a given value available to all components that are called in the same Servlet Session, which may span many html hits. The session life is dependent upon the HTTP Server Session timouts. These objects are referenced by using the key in an expression (see getSessionValue() below). Please note that this call raises an error when Composer is when deployed as an EJB or as a JMS message listener. The key is a String and the value is a Java object.

removeSessionValue(key)

removeSessionValue (key) Interpreter extension—Composer components. Removes a given value previously exported via the putSessionValue() call (see above) Please note that this call raises when error when Composer is deployed as an EJB or as a JMS message listener. The key is a String.

setValidationViaSchemaLocation(abFlag)

setValidationViaSchemaLocation(abFlag) Interpreter extension—Composer components. XML documents which have xsi:schemaLocation or xsi:nonamespaceschemaLocation can use this flag to validate the documents. Please note, once set to true it will remain true till the component is closed/end of execution/explicitly set to false again.

theComponent

This top-level property represents a handle to the xObject in which the script is running. Usually, you will be running scripts from within a Component (e.g., XML Map Component), in which case the property is a Component object reference. Most of the above methods can be called on `theComponent`.

I/O Extensions

The following methods are not part of core ECMAScript (and not natively supported in Mozilla Rhino). They are custom extensions implemented by Composer.

alert(message)

alert(message) Interpreter extension for Basic I/O. Write a message on the error stream. The message parameter is of type String. In Composer a msgBox is displayed. e.g. alert("The element=" + Input.XPath("root/child"))

canRead()

boolean canRead() Interpreter extension—File object method. Returns true if the file exists and is readable.

canWrite()

boolean canWrite() Interpreter extension—File object method. Returns true if the file exists and is can be written to.

clearError()

clearError() Interpreter extension—File object method. Clears the error indicator.

close()

boolean close() Interpreter extension—File object method. Closes a file and returns true if successful.

confirm(message)

confirm(message) Interpreter extension for Basic I/O. Ask the user for a confirmation. Returns true or false. The message parameter is of type String.

eof()

boolean eof() Interpreter extension—File object method. Return true if the file is at eof or if it is not open for reading. An error is set if the file was not opened for reading.

error()

String error() Interpreter extension—File object method. Returns the string of the last error on this file object since it was created or since clearError was called. Returns the empty string (which is equivalent to false) if no error was detected.

exists()

boolean exists() Interpreter extension—File object method. Check if the file exists. Return true if the file exists (whether opened or not).

File(FileName)

File(String fileName) File constructor. A File object is created by File("pathname"), File("directory","filename"), or simply File( ). The directory can be a File object. Doing toString() on the File object returns `fileName`.

flush()

boolean flush() Interpreter extension—File object method. Flush the output buffer on a file opened for writing. Returns true in case of success.

getAbsolutePath()

String getAbsolutePath() Interpreter extension—File object method. Returns the complete path of the file.

getLength()

number getLength() Interpreter extension—File object method. Returns the length of the file, 0 if it does not exist.

getName()

String getName() Interpreter extension—File object method. Returns the file component part of the path.

getParent()

String getParent() Interpreter extension—File object method. Returns the directory part of the file or the parent directory of a directory. Returns a null string if applied to the root directory (of any drive in Windows).

getPath()

String getPath() Interpreter extension—File object method. Returns the path given at construction time. Same as toString().

isAbsolute()

boolean isAbsolute() Interpreter extension—File object method. Returns true if the path is absolute, false otherwise.

isDirectory()

boolean isDirectory() Interpreter extension—File object method. Checks for the existence of a directory. Returns true if the file exists and is a directory.

isFile()

boolean isFile() Interpreter extension—File object method. Return true if the file exists and is a regular file (not a directory).

isopened()

boolean isopened() Interpreter extension—File object method. Checks if file is already open. Return true if the file is opened (even if at EOF), otherwise return false.

lastModified()

String lastModified() Interpreter extension—File object method. Returns the date and time when the file was last modified, or “Jan 1 1970” if it does not exist.

list()

[Stringarray] list() Interpreter extension—File object method. Returns list of files or directories in a directory, as an array of strings. Returns null if the File object does not represent a directory.

load(moduleName)

load(moduleName) Interpreter extension for Basic I/O. Locate the module in the directories and jar/zip files of the property Interpreter.path or (if the property is not defined) in the classpath (possibly adding the extension ".es", ".esw' or ".js"). Load the module and evaluate its content at run-time.

mkdir()

boolean mkdir() Interpreter extension—File object method. Creates the directory or directories specified by the path of the File object if they do not exist. Returns true if the directories were successfully created, false otherwise.

open()

boolean open() Interpreter extension—File object method. Open the file for reading if it exists, for writing if it does not exist. Return true if successful. To create a new file, just delete it (using remove()) before calling open.

prompt(Msg,DefaultText)

prompt(Msg,DefaultText) Interpreter extension for Basic I/O. Prompt the user for information, proposing the default string. The Msg and DefaultText parameters are of type String.

readAll()

String readAll() Interpreter extension—File object method. Open the file, read all text, close the file. Returns the read text as a single string. Returns null in case of error. The file must not be opened when readAll is called!.

readln()

readln() Interpreter extension—File object method. Reads a line from an input file. Returns null if at eof or in case of error. It may be more convenient to use the error() routine to check for end of file or error conditions.

remove()

boolean remove() Interpreter extension—File object method. Deletes a file. Returns true in case of success. The name remove is used as delete and is an ECMAScript keyword.

renameTo(newname)

boolean renameTo(newname) Interpreter extension—File object method. Rename the file to the target name. Target name can be a File object or a string. Returns true if successful.

separator

separator Interpreter extension—File object property. This property returns the path separator, i.e. the backslash for Windows.

stderr()

stderr() Interpreter extension—File object property. This property returns the standard error.

stdin

stdin Interpreter extension—File object property. This property returns the standard input.

stdout

stdout Interpreter extension—File object property. This property returns the standard output.

toString()

String toString() Interpreter extension—File object method. Returns the path given at construction time. Same as getPath().

write(...)

boolean write(...) Interpreter extension—File object method. Write the parameters as a string to the file. Returns true if successful.

write(...)

write(...) Interpreter extension for Basic I/O. Write the parameters to the current output stream. The Text parameter is of type String.

writeln(...)

writeln(...) Interpreter extension—File object method. Write the parameters as a string to the file, followed by a new line. Returns true if successful.

writeln(...)

writeln(...) Interpreter extension for Basic I/O. Write parameters to the current output stream, followed by a new line.

Database Extensions

These are Composer-defined extensions (not defined in core ECMAScript). For examples of their usage, consult the Custom Scripts resource in the ActionExamples project that ships in Composer's Samples folder.

connect(URL)

boolean connect(URL) Interpreter extension—Database object method. Opens a new connection for the specified URL on the DbAccess object. Generates an error if the driver was not correctly initialized. Returns true if connection is successful, false otherwise. Use getLastError to get details on the error.

connect(URL, userName, password)

boolean connect(URL, userName, password) Interpreter extension—Database object method. Opens a new connection for the specified URL, with specified user name and password, on the DbAccess object. Generates an error if the driver was not correctly initialized, returns true if connection is successful, false otherwise. Use getLastError to get details on the error.

Database(driver)

globaldbobject Database(driver) Interpreter extension Database constructor. Creates an object used to access databases via JDBC. Returns a global database object.

disconnect()

boolean disconnect() Interpreter extension—Database object method. Close a connection (do nothing if not connected). Generates an error if the driver was not correctly initialized. Returns true if successful or already disconnected, false in case of error. Use getLastError to get details in case of error.

executeCommand(sqlString)

arg executeCommand(sqlString) Interpreter extension—Database object method. Used to implement INSERT, UPDATE, DDL statements and other non-value returning statements. Returns the number of rows impacted if the request is successful, otherwise returns false. If false was returned, getLastError may be called on the database object to get the error information. Note that an error is generated (rather than returning a status) if the connection was not successful.

executeRetrieval(sqlString)

RowSet executeRetrieval(sqlString) Interpreter extension—Database object method. Used to implement SELECT and other value returning statements. Return a RowSet object implementing the RowsetAccess protocol if the request is successful. Returns false otherwise. Additional arguments are ignored, and data is always returned as a RowSetAccess. The cursor is positioned before the fist row. If false was returned, getLastError may be called on the database object to get the error information. Note that an error is generated (rather than returning a status) if the connection was not successful.

getColumnCount()

number getColumnCount() Interpreter extension—Database RowSet object method. Get the number of columns of this result, identical to the length attribute. Its possible to call this routine before the first record is fetched.

getColumnDatatypeName()

String getColumnDatatypeName() Interpreter extension—Database RowSet object method. Get the name of the datatype associated with the column. See the JDBC documentation for details. Some database do not return a valid name, in that case undefined is returned. It is possible to call this routine before the first record is fetched.

getColumnDatatypeNumber()

number getColumnDatatypeNumber() Interpreter extension—Database RowSet object method. Get the number of the datatype associated with the column. See the JDBC documentation for details. It is possible to call this routine before the first record is fetched.

getColumnItem(name)

value getColumnItem(name) Interpreter extension—Database RowSet object method. Get the value of a column by its name (the value can be accessed by number simply indexing them—this is not faster than by name for Interpreter). The proper value is returned even if the name is used for a property of the RowSetAccess protocol, as next or length. A record must be available (that is next must have been called at least once).

getColumnName()

String getColumnName() Interpreter extension—Database RowSet object method. Get the name of a column, in a way which is always working. The names can be accessed as properties, but they are shadowed by the functions and properties of the RowSetAccess prototype object. It is possible to call this routine before the first record is fetched.

getLastError()

error getLastError() Interpreter extension—Database object method. Return the last error which occurred when connecting or executing a statement, undefined if none.

getLastError()

error getLastError() Interpreter extension—Database RowSet object method. Return the last error which occurred when connecting or executing a statement, null (which test as false) if none.

getMetaData()

metadata getMetaData() Interpreter extension—Database object method. Returns the meta data attached to the connection.

getMetaData()

metadata getMetaData() Interpreter extension—Database RowSet object method. Return the meta data attached to the row set.

hasMoreRows()

boolean hasMoreRows() Interpreter extension—Database RowSet object method. Optimistic view of the possibility that more rows are present. Currently only returns false if next returned false. It is possible to call this routine at any time.

next()

String next() Interpreter extension—Database RowSet object method. Get the next row of results. Returns true if there is a next row, false otherwise. Note that next must be called before the first row can be accesses.

release()

release() Interpreter extension—Database object method. Equivalent to disconnect, but does not return a status and always succeed. Usually this will release any associated RowSet.

release()

release() Interpreter extension—Database RowSet object method. Release the resources attached to this RowSet object. The object should not be accessed after it has been released. The use of release is not mandatory, but is recommended as otherwise the resources (which may include large cache and a database connection) is not reclaimed before the next garbage collection, in this only if it is not reachable anymore.

2 Connect-Specific Methods

3270

USERID

USERID Property

PASSWORD

PASSWORD Property

RESOURCENAME

RESOURCENAME Property

getText(aOffset, aLength)

String getText(Object aOffset, Object aLength) Composer extension method. Returns text string which is part of the screen text buffer starting from position at offset aOffset and having length aLength characters. Field attributes are substituted with spaces

getTextAt(aRow, aColumn, aLength)

String getTextAt(Object aRow, Object aColumn, Object aLength) Composer extension method. Returns text string which is part of the screen text buffer starting from position at row aRow, column aColumn and having length aLength characters. Field attributes are substituted with spaces

getTextFromRectangle(aStartRow, aStartColumn, aEndRow, aEndColumn)

String getTextFromRectangle(Object aStartRow, Object aStartColumn, Object aEndRow, Object aEndColumn)

Composer extension method. Returns text string concatenated from substrings which compose rectangular area of the screen text buffer determined by coordinates: start position—aStartRow, aStartColumn end position—aEndRow, aEndColumn. Field attributes are substituted with spaces

setText(aOffset, aText)

String setText(Object aOffset, Object aText) Composer extension method. Sets text—aText string at particular offset—aOffset of the screen text buffer. If text is longer then field text is inserted up to current fields length.

setTextAt(aRow, aColumn, aText)

String setTextAt(Object aRow, Object aColumn, Object aText) Composer extension method. Sets text—aText string starting at Row—aRow and Column—aColumn of the screen text buffer. If text is longer then field text is inserted up to current fields length.

5250 (see also “Telnet”)

USERID

USERID Property

PASSWORD

PASSWORD Property

CICS RPC

getField(String cobolDataDesc)

CopybookField getField(String cobolDataDesc) Composer extension method. Returns a CopybookField object. To resolve duplicate named cobolDataDescs in a Copybook, reference the parent cobolDataDesc as follows: getField("PARTID IN INDATA") given the following copybook: 01 COMMAREA 05 INDATA 10 PARTID 05 OUTDATA 10 PARTID The returned CopybookField object has two methods: toString() and setValue().

setValue(Object aValue)

void setValue(Object aValue) Composer extension method. This method sets the value for a CopybookField object.

toString()

String toString() Composer extension method. Returns the value set for the CopybookField object.

Data General, HP3000, T27

HOSTID

HOSTID As Defined in Connection

PASSWORD

PASSWORD As Defined in Connection

USERID

USERID As Defined in Connection

getAttribute(aRow, aColumn)

int getAttribute(Object aRow, Object aColumn) Composer extension method. Returns attribute of a character at a T27 screen position at row aRow and column aColumn.

getCols()

int getCols() Composer extension method. Returns number of columns from the T27 screen.

getCursorCol()

int getCursorCol() Composer extension method. Returns current T27 screen cursor column position.

getCursorRow()

int getCursorRow() Composer extension method. Returns current T27 screen cursor row position.

getNextMessage()

getNextMessage() Composer extension method. Returns the next captured message.

getPrompt()

String getPrompt() Composer extension method. Returns text from cursor back to the left margin of the T27 screen.

getRows()

int getRows() Composer extension method. Returns number of rows from the T27 screen.

getStatusLine()

String getStatusLine() Composer extension method. Returns text of the screen status line.

getText(aRow, aColumn, aLength)

String getText(Object aRow, Object aColumn, Object aLength) Composer extension method. Returns text string found at position aRow and aColumn for aLength characters.

getTextFromRectangle(aStartRow, aStartColumn, aEndRow, aEndColumn)

String getTextFromRectangle(Object aStartRow, Object aStartColumn, Object aEndRow, Object aEndColumn)

Composer extension method. Returns block of text.

hasMoreMessages()

boolean hasMoreMessages() Composer extension method. Returns true if more captured messages are available to obtain via the getNextMessage() method.

putString(aRow, aColumn, asText)

putString(Object aRow, Object aColumn, Object asText) Composer extension method. Sets text asText in the screen at position aRow and aColumn.

putStringInField(aField, asText)

putStringInField(Object aField, Object asText) Composer extension method. Sets text asText in the screen at position aField.

setMessageCaptureOff()

setMessageCaptureOff() Composer extension method. Turns off message capture.

setMessageCaptureOn()

setMessageCaptureOn() Composer extension method. Sets up the capture of a message from the host immediately following a Transmit. Use getNextMessage() to retrieve data, then use setMessageCaptureOff() to turn off.

typeKeys(asKeyText)

typeKeys(Object asKeyText) Composer extension method. Keystroke represented by asKeyText is emulated on the screen.

KEY EQUIVALENTS

Clear Home

Keyboard equivalent: Ctrl+Home

Forms Mode, Toggle

Keyboard equivalent: Esc, Alt+S, O

Local

Keyboard equivalent: F10 or F6

Next Page

Keyboard equivalent: PageDown

Previous Page

Keyboard equivalent: PageUp

Receive

Keyboard equivalent: F11 or F7

Specify

Keyboard equivalent: F9 or F5

Transmit

Keyboard equivalent: F12 or F8

EDI

TRANSMISSION OBJECT

hasMoreInterchanges()

boolean hasMoreInterchanges() Returns true if the Transmission contains more Interchanges for processing. Returns false if no additional Interchanges are available.

getValue()

String getValue() Returns the contents of an Transmission as a raw string in native EDI text format.

INTERCHANGE OBJECT

getInterchangeInfo()

Element getInterchangeInfo() Returns a DOM Element Node containing the XML representation of an EDI Interchange header and footer.

getSenderID()

String getSenderID() Returns the SenderID field data from an Interchange.

Example:

```
myInterchangeName.getSenderID()
```

getSenderIDQualifier()

String getSenderIDQualifier() Returns a table code value used to resolve the SenderID.

getStandard()

String getStandard() Returns a string indicating the EDI encoding standard of the Interchange (e.g. ANSIX12 or EDIFACT).

getUsageIndicator()

String getUsageIndicator() An X12 document returns a value of: P,T, or U. P = a production transaction, T = a test transaction, and U = ‘unknown’. An EDIFACT document returns a value of: 0 or 1 where 1 = a test transaction, and 0 = a production transaction.

getValue()

String getValue() Returns the contents of an Interchange in native EDI text format.

hasMoreDocuments()

boolean hasMoreDocuments() Returns true if the Interchange contains more Documents for processing.

DOCUMENT METHODS

getControlID()

String getControlID() Returns the Control ID assigned by the sender of a document used to uniquely identify each document.
Example:

```
myDocumentName.getControlID()
```

getDocType()

String getDocType() Returns the particular type of document within the EDI standard. For instance, 850 might be returned for an ANSI X12 Purchase Order. Example:

```
myDocumentName.getDocType()
```

getSenderID()

String getSenderID() Returns the SenderID field data in an Interchange. Example:

```
myDocumentName.getSenderID()
```

getSenderIDQualifier()

String getSenderIDQualifier() Returns a table code value used to resolve the SenderID. Example:

```
myDocumentName.getSenderIDQualifier()
```

getStandard()

String getStandard() Returns a string indicating the EDI encoding standard of the Interchange (e.g. ANSIX12 or EDIFACT). Example:

```
myDocumentName.getStandard()
```

getValue()

String getValue() Returns the contents of an EDI Document in native EDI text format. Example:

```
myDocumentName.getValue()
```

getVersion()

String getVersion() Returns the version number of the EDI standard used to encode the EDI transmission. Example:

```
myDocumentName.getVersion()
```

HTML

NOTE: In the HTML component editor environment, you can utilize standard HTML DOM-2 ECMAScript extensions as defined by W3C. For the complete API breakout, consult http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/ecma-script-binding.html.

HTMLRespMsg

HTTP Response Message

HTMLStatusCode

HTTP Status Code

JDBC

SQLCODE

A numeric code indicating whether the SQL statement executed successfully. In General 0 = success, 100 = no data found, a number > 0 = success with a warning, and a number < 0 = execution not successful. The meaning of SQLCODEs other than 0 and 100 varies with the particular product implementing SQL.

SQLSTATE

An SQL implementation specific numeric code indicating the success of an SQL command. See your database engine's documentation for details.

UPDATECOUNT

A number indicating how many records were updated by an SQL UPDATE command.

LASTSQL

Contains a string of the last SQL statement issued.

JMS

JMSMESSAGE

Root reference name for JMS message methods.

getField(String cobolDataDesc)

CopybookField getField(String cobolDataDesc) Composer extension method. Returns a CopybookField object. To resolve duplicate named cobolDataDescs in a Copybook, reference the parent cobolDataDesc as follows:

```
getField("PARTID IN INDATA")
```

given the following copybook:

```
01 COMMAREA
05 INDATA
10 PARTID
05 OUTDATA
10 PARTID
```

The returned CopybookField object has two methods: toString() and setValue().

getJMSBytesBody()

String getJMSBytesBody() Composer extension method. This method gets the value for a JMS BytesMessage objects body and repositions the stream to the beginning. Returns a Base64 encoded String.

getJMSBytesBody(int aiBufSize)

String getJMSBytesBody(int aiBufSize) Composer extension method. This method gets the value for a JMS BytesMessage objects body The aiBufSize parameter is the size of the body in bytes. Returns a Base64 encoded String.

getJMSBytesBodyAsBytes()

Object getJMSBytesBodyAsBytes() Composer extension method. This method gets the value for a JMS BytesMessage objects body and repositions the stream to the beginning. Returns a Java byte[] Object.

getJMSBytesBodyAsBytes(int aiBufSize)

Object getJMSBytesBodyAsBytes(int aiBufSize) Composer extension method. This method gets the value for a JMS BytesMessage objects body The aiBufSize parameter is the size of the body in bytes. Returns a Java byte[] object (i.e., byte array).

getJMSMapBody()

String getJMSMapBody() Composer extension method. This method gets a string representation of the JMS MapMessage body. Returns a String. If a field is type BYTES it is represented as a Base64 encoded String.

getJMSMapField(String asName, String asType)

String getJMSMapField(String asName, String asType) Composer extension method. This method gets the value for a JMS MapMessage object body field. Returns a String. If asType is BYTES returns a Base64 encoded String.

getJMSMessage()

Message getJMSMessage() Composer extension method. Returns a JMS Message object. To work with a specific type of JMS Message body, cast the returned message. For Example: TextMessage lMsg = (TextMessage)getJMSMessage().

getJMSMsgBody()

String getJMSMsgBody() Composer extension method. Returns a JMS Message objects body as a String.

getJMSMsgType()

String getJMSMsgType() Composer extension method. Returns a JMS Message objects type.

getJMSObjectBody()

Serializable getJMSObjectBody() Composer extension method. This method gets the value for a JMS ObjectMessage object body. Returns a Serializable.

getJMSStreamBody()

String getJMSStreamBody() Composer extension method. This method gets a string representation of the JMS StreamMessage body. Returns a String. If a field is type BYTES it is represented as a Base64 encoded String. The stream is reset.

getJMSStreamField(String asName, String asType)

String getJMSStreamField(String asName, String asType) Composer extension method. This method gets the value for a JMS StreamMessage object body field. Returns a String. If asType is BYTES returns a Base64 encoded String.

getJMSTextBody()

String getJMSTextBody() Composer extension method. This method gets the value for a JMS TextMessage object body. Returns a String.

hasMessages()

String hasMessages() Composer extension method. This method returns true if there are more messages to browse. Returns a boolean.

resetPosition()

resetPosition() Composer extension method. This method puts the bytes message body in read-only mode, and repositions the stream to the beginning.

setJMSBytesBody()

setJMSBytesBody() Composer extension method. This method sets the body for a JMS BytesMessage objects. Source data type is a Base64 String or a java byte[].

setJMSMapField(String asName, String asType)

setJMSMapField(String asName, String asType) Composer extension method. This method sets the name and type of a JMS MapMessage object body field. If asType is BYTES the source data type is a Base64 String or a java byte[].

setJMSMsgProperty(String asName, String asType, String asValue)

void setJMSMsgProperty(String asName, String asType, String asValue) Composer extension method. This method sets a JMS header property of a passed in name and type to a passed in value.

setJMSObjectBody(Serializable aObject)

setJMSObjectBody(Serializable aObject) Composer extension method. This method sets the body of a JMS ObjectMessage object.

setJMSStreamField(String asName, String asType)

setJMSStreamField(String asName, String asType) Composer extension method. This method sets the value of a JMS StreamMessage object body field. If asType is BYTES the source data type is a Base64 String or a java byte[].

setMessageSize(int aiSize)

void setMessageSize(int aiSize) xComposer extension method. This method allows the user to override the default copybook message size. By default, the maximum size defined by the copybook will be transmitted. The user's responsibility is to assign the message size such a way that it provides preserving all necessary data.

setValue(Object aValue)

void setValue(Object aValue) Composer extension method. This method sets the value for a CopybookField object.

toString()

String toString() Composer extension method. Returns the value set for the CopybookField object.

LDAP

createAttrACL(compare, read, write, self, supervisor, inherit, trusteeDN, attrName)

String createAttrACL(boole Compare, boolean read, boolean write, boolean self, boolean supervisor, boolean inherit, String trusteeDN, String attrName) Creates an eDirectory ACL in the form: "31#entry#cn=test,o=corp#attrName" Attribute rights give a trustee rights to the specifed attribute of an LDAP object. Rights may be applied to all attributes by specifying null as the "attrName".

createEntryACL(browse, add, delete, rename, supervisor, inherit, trusteeDN)

String createEntryACL(boolean browse, boolean add, boolean delete, boolean rename, boolean supervisor, boolean inherit, String trusteeDN) Creates an eDirectory ACL in the form: "31#entry#cn=test,o=corp#[Entry Rights]" Entry rights give a trustee rights to an LDAP object. Specify true for "inherit" to allow the whole subtree to inherit these rights.

getLDAPAttr(connection, DN, attrName)

String getLDAPAttr(String connection, String DN, String attrName) This function returns the value of an attribute from a directory object. Parameters: string connection – name of a LDAP Connection Resource string DN – Distinguished Name of a directory object string attr – name of attribute (value to be returned) e.g. addr=getLDAPAttr(\“myCNX\”,\” cn=AClark,o=NOVELL\”, \“eMailAddress\”)

INHERITANCE_MASK

INHERITANCE_MASK is a value of the trusteeDN parameter to both the createEntryACL() and createAttrACL() methods. Inheritance masks restrict which rights may be inherited by an LDAP object. To create an inheritance mask, specify INHERITANCE_MASK as the "trusteeDN" when creating an ACL.

PUBLIC

PUBLIC is a value of the "trusteeDN" parameter to both the createEntryACL()

and createAttrACL() methods. To create public rights, specify PUBLIC as the "trusteeDN" when creating an ACL.

Process Manager

DragSource

Composer extension property. It represents the DOM element "DragSource".

DropTarget

Composer extension property. It represents the DOM element "DropTarget".

PASSWORD

PASSWORD Property

USERID

USERID Property

String getValue()

String getValue() Composer extension method. This method gets the value for this Field object.

void setValue(Object aValue)

void setValue(Object aValue) Composer extension method. This method sets the value for this Field object.

SAP: Import/Export Methods

getExportField(String asName)

Object getExportField(String asName) Composer extension method. This method gets the value of an SAP Export field. Returns an Object.

getExportFieldInfo(String asName, String asType)

String getExportFieldInfo(String asName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Export field. Returns a String.

getImportField(String asName)

Object getImportField(String asName) Composer extension method. This method gets the value of an SAP Import field. Returns an Object.

getImportFieldInfo(String asName, String asType)

String getImportFieldInfo(String asName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Import field. Returns an String.

setImportField(String asName, String asType)

setImportField(String asName, String asType) Composer extension method. This method sets the value of an SAP Import field.

SAP: Structure Methods

getExportStructAsXML(String asStructName)

Node getExportStructAsXML(String asStructName) Composer extension method. This method gets the value of an SAP Export Structure as an XML Node. Returns a Node.

getExportStructField(String asStructName, String asFieldName)

Object getExportStructField(String asStructName, String asFieldName) Composer extension method. This method gets the value of an SAP Export Structure field. Returns an Object.

getExportStructFieldInfo(String asStructName, String asFieldName, String asType)

String getExportStructFieldInfo(String asStructName, String asFieldName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Export Structure field. Returns an String.

getImportStructAsXML(String asStructName)

Node getImportStructAsXML(String asStructName) Composer extension method. This method gets the value of an SAP Import Structure as an XML Node. Returns a Node.

getImportStructField(String asStructName, String asFieldName)

Object getImportStructField(String asStructName, String asFieldName) Composer extension method. This method gets the value of an SAP Import Structure field. Returns an Object.

getImportStructFieldInfo(String asStructName, String asFieldName, String asType)

String getImportStructFieldInfo(String asStructName, String asFieldName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Import Structure field. Returns an String.

setImportStructField(String asStructName, String asFieldName, String asType)

setImportStructField(String asStructName, String asFieldName, String asType) Composer extension method. This method sets the value of an SAP Import Structure field.

setImportStructWithXML(String asStructName)

setImportStructWithXML(String asStructName) Composer extension method. This method sets the value of an SAP Import Structure with an XML Node.

SAP: Table Methods

appendTableRow(String asTableName)

appendTableRow(String asTableName) Composer extension method. This method apppends a new row into the specified SAP Table.

getTableAsXML(String asTableName)

Node getTableAsXML(String asTableName) Composer extension method. This method gets the value of an SAP Table an XML Node. Returns a Node.

getTableField(String asTableName, String asFieldName)

Object getTableField(String asTableName, String asFieldName) Composer extension method. This method gets the value of an SAP Table field. Returns an Object.

getTableFieldInfo(String asTableName, String asFieldName, String asType)

String getTableFieldInfo(String asTableName, String asFieldName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Table field. Returns an String.

getTableRowCount(String asTableName)

int getTableRowCount(String asTableName) Composer extension method. This method gets the number of rows in an SAP Table. Returns a int.

nextTableRow(String asTableName)

boolean nextTableRow(String asTableName) Composer extension method. This method moves the cursor to the next row in the specified SAP Table.

setTableField(String asTablrName, String asFieldName, String asType)

setTableField(String asTableName, String asFieldName, String asType) Composer extension method. This method sets the value of an SAP Table field.

setTableRow(String asTableName, int aiRow)

settTableRow(String asTableName, int aiRow) Composer extension method. This method moves the cursor to the specified row in the specified SAP Table.

setTableWithXML(String asTableName)

setTableWithXML(String asTableName) Composer extension method. This method sets the value of an SAP Table with an XML Node.

SAP: Function Methods

dumpFunctionToLog()

dumpFunctionToLog() Composer extension method. This method dumps the value of the SAP Function as a formatted XML String to the system log.

getFunctionAsXML()

Node getFunctionAsXML() Composer extension method. This method gets the value of the SAP Function as an XML Node. Returns a Node.

getFunctionName()

String getFunctionName() Composer extension method. This method gets the current actions Function name. Returns a String.

getRFCObject()

Object getRFCObject() Composer extension method. This method gets the current actions RFC Object. Returns an Object.

SAP: Client Methods

getClient()

Object getClient() Composer extension method. This method gets the current Client Object. Returns an Object.

getLanguage()

String getLanguage() Composer extension method. This method gets the SAP logon language for the current Client. Returns a String.

getStats()

String getStats() Composer extension method. This method gets the SAP performance statistics for the current Client. Requires JCOPERFORM set to true in xConfig. Returns a String.

printSAPClientInfo()

printSAPClientInfo() Composer extension method. This method prints SAP JCO.Client information to System.out.

resetClient()

resetClient() Composer extension method. This method resets the JCO.Client.

resetStats()

resetStats() Composer extension method. This method resets the SAP performance statistics for the current Client. Requires JCOPERFORM set to true in xConfig.

SAP: Middleware Methods

getMiddlewareLayer()

String getMiddlewareLayer() Composer extension method. This method gets the current middleware layer. Returns a String.

getMiddlewareProperty(String asKey)

String getMiddlewareProperty(String asKey) Composer extension method. This method gets the current value of a middleware property. Returns a String.

setMiddlewareLayer(String asLayerName)

setMiddlewareLayer(String asLayerName) Composer extension method. This method sets the middleware layer.

setMiddlewareProperty(String asKey, String asValue)

setMiddlewareProperty(String asKey, String asValue) Composer extension method. This method sets a middleware property value.

SAP: Utility Methods

getLastError()

String getLastError() Composer extension method. This method gets the last error thrown while processing an SAP Function. Returns a String.

printSAPInfo()

printSAPInfo() Composer extension method. This method prints SAP JCo information to System.out.

setFieldGetAsString(boolean abFlag)

setFieldGetAsString(boolean abFlag) Composer extension method. This method sets how field data is retrieved from a function. When true all output is retrieved as a String. When false all output is retrieved as an appropriate Object type for the field being retrieved.

setFieldSetAsString(boolean abFlag)

setFieldSetAsString(boolean abFlag) Composer extension method. This method sets how field data is set into a function. When true all input is converted to a String before setting into a function. When false all input is converted, if necessary, to an appropriate Object type for the field before setting.SAPSERVICE: Simple Field Methods

getExportField(String asName)

Object getExportField(String asName) Composer extension method. This method gets the value of an SAP Export field. Returns an Object.

getExportFieldInfo(String asName, String asType)

String getExportFieldInfo(String asName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Export field. Returns a String.

getImportField(String asName)

Object getImportField(String asName) Composer extension method. This method gets the value of an SAP Import field. Returns an Object.

getImportFieldInfo(String asName, String asType)

String getImportFieldInfo(String asName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Import field. Returns an String.

setExportField(String asName, String asType)

setExportField(String asName, String asType) Composer extension method. This method sets the value of an SAP Export field.

setImportField(String asName, String asType)

setImportField(String asName, String asType) Composer extension method. This method sets the value of an SAP Import field.

SAPSERVICE: Structure Methods

getExportStructAsXML(String asStructName)

Node getExportStructAsXML(String asStructName) Composer extension method. This method gets the value of an SAP Export Structure as an XML Node. Returns a Node.

getExportStructField(String asStructName, String asFieldName)

Object getExportStructField(String asStructName, String asFieldName) Composer extension method. This method gets the value of an SAP Export Structure field. Returns an Object.

getExportStructFieldInfo(String asStructName, String asFieldName, String asType)

String getExportStructFieldInfo(String asStructName, String asFieldName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Export Structure field. Returns an String.

getImportStructAsXML(String asStructName)

Node getImportStructAsXML(String asStructName) Composer extension method. This method gets the value of an SAP Import Structure as an XML Node. Returns a Node.

getImportStructField(String asStructName, String asFieldName)

Object getImportStructField(String asStructName, String asFieldName) Composer extension method. This method gets the value of an SAP Import Structure field. Returns an Object.

getImportStructFieldInfo(String asStructName, String asFieldName, String asType)

String getImportStructFieldInfo(String asStructName, String asFieldName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Import Structure field. Returns an String.

setExportStructField(String asStructName, String asFieldName, String asType)

setExportStructField(String asStructName, String asFieldName, String asType) Composer extension method. This method sets the value of an SAP Export Structure field.

setExportStructWithXML(String asStructName)

setExportStructWithXML(String asStructName) Composer extension method. This method sets the value of an SAP Export Structure with an XML Node.

setImportStructField(String asStructName, String asFieldName, String asType)

setImportStructField(String asStructName, String asFieldName, String asType) Composer extension method. This method sets the value of an SAP Import Structure field.

setImportStructWithXML(String asStructName)

setImportStructWithXML(String asStructName) Composer extension method. This method sets the value of an SAP Import Structure with an XML Node.

SAPSERVICE: Table Methods

appendTableRow(String asTableName)

appendTableRow(String asTableName) Composer extension method. This method apppends a new row into the specified SAP Table.

getTableAsXML(String asTableName)

Node getTableAsXML(String asTableName) Composer extension method. This method gets the value of an SAP Table an XML Node. Returns a Node.

getTableField(String asTableName, String asFieldName)

Object getTableField(String asTableName, String asFieldName) Composer extension method. This method gets the value of an SAP Table field. Returns an Object.

getTableFieldInfo(String asTableName, String asFieldName, String asType)

String getTableFieldInfo(String asTableName, String asFieldName, String asType) Composer extension method. This method gets a specified type of metadata for an SAP Table field. Returns an String.

getTableRowCount(String asTableName)

int getTableRowCount(String asTableName) Composer extension method. This method gets the number of rows in an SAP Table. Returns a int.

nextTableRow(String asTableName)

boolean nextTableRow(String asTableName) Composer extension method. This method moves the cursor to the next row in the specified SAP Table.

setTableField(String asTablrName, String asFieldName, String asType)

setTableField(String asTableName, String asFieldName, String asType) Composer extension method. This method sets the value of an SAP Table field.

setTableRow(String asTableName, int aiRow)

settTableRow(String asTableName, int aiRow) Composer extension method. This method moves the cursor to the specified row in the specified SAP Table.

setTableWithXML(String asTableName)

setTableWithXML(String asTableName) Composer extension method. This method sets the value of an SAP Table with an XML Node.

SAPSERVICE: Function Methods

dumpFunctionToLog()

dumpFunctionToLog() Composer extension method. This method dumps the value of the SAP Function as a formatted XML String to the system log.

getFunctionAsXML()

Node getFunctionAsXML() Composer extension method. This method gets the value of the SAP Function as an XML Node. Returns a Node.

getFunctionName()

String getFunctionName() Composer extension method. This method gets the current actions Function name. Returns a String.

getRFCObject()

Object getRFCObject() Composer extension method. This method gets the current actions RFC Object. Returns an Object.

SAPSERVICE: Client Methods

getClient()

Object getClient() Composer extension method. This method gets the current Client Object. Returns an Object.

getLanguage()

String getLanguage() Composer extension method. This method gets the SAP logon language for the current Client. Returns a String.

getStats()

String getStats() Composer extension method. This method gets the SAP performance statistics for the current Client. Requires JCOPERFORM set to true in xConfig. Returns a String.

printSAPClientInfo()

printSAPClientInfo() Composer extension method. This method prints SAP JCO.Client information to System.out.

resetClient()

resetClient() Composer extension method. This method resets the JCO.Client.

resetStats()

resetStats() Composer extension method. This method resets the SAP performance statistics for the current Client. Requires JCOPERFORM set to true in xConfig.

SAPSERVICE: Middleware Methods

getMiddlewareLayer()

String getMiddlewareLayer() Composer extension method. This method gets the current middleware layer. Returns a String.

getMiddlewareProperty(String asKey)

String getMiddlewareProperty(String asKey) Composer extension method. This method gets the current value of a middleware property. Returns a String.

setMiddlewareLayer(String asLayerName)

setMiddlewareLayer(String asLayerName) Composer extension method. This method sets the middleware layer.

setMiddlewareProperty(String asKey, String asValue)

setMiddlewareProperty(String asKey, String asValue) Composer extension method. This method sets a middleware property value.

SAPSERVICE: Utility Methods

clearABAPException()

clearABAPException() Composer extension method. This method clears the JCO.ABAPException.

getLastError()

String getLastError() Composer extension method. This method gets the last error thrown while processing an SAP Function. Returns a String.

printSAPInfo()

printSAPInfo() Composer extension method. This method prints SAP JCo information to System.out.

raiseABAPException(String asKey, String asMessage)

raiseABAPException(String asKey, String asMessage) Composer extension method. This method stores a JCO.ABAPException to be thrown from the SAP Services request handler.

setFieldGetAsString(boolean abFlag)

setFieldGetAsString(boolean abFlag) Composer extension method. This method sets how field data is retrieved from a function. When true all output is retrieved as a String. When false all output is retrieved as an appropriate Object type for the field being retrieved.

setFieldSetAsString(boolean abFlag)

setFieldSetAsString(boolean abFlag) Composer extension method. This method sets how field data is set into a function. When true all input is converted to a String before setting into a function. When false all input is converted, if necessary, to an appropriate Object type for the field before setting.

Telnet

USERID

USERID As Defined in Connection

PASSWORD

PASSWORD As Defined in Connection

getAttribute(aRow, aColumn)

int getAttribute(Object aRow, Object aColumn) Composer extension method. Returns attribute of a character at a Telnet screen position at row aRow and column aColumn.

getColumnCount()

int getColumnCount() Composer extension method. Returns number of columns from the Telnet screen.

getCursorColumn()

int getCursorColumn() Composer extension method. Returns current Telnet screen cursor column position.

getCursorRow()

int getCursorRow() Composer extension method. Returns current Telnet screen cursor row position.

getPrompt()

String getPrompt() Composer extension method. Returns text from cursor back to the left margin of the Telnet screen.

getRowCount()

int getRowCount() Composer extension method. Returns number of rows from the Telnet screen.

getText(aOffset, aLength)

String getText(Object aOffset, Object aLength) Composer extension method. Returns text string which is part of the screen text buffer starting from position at offset aOffset and having length aLength characters.

getTextAt(aRow, aColumn, aLength)

String getTextAt(Object aRow, Object aColumn, Object aLength) Composer extension method. Returns text string which is part of the screen text buffer starting from position at row aRow, column aColumn and having length aLength characters.

getTextFromRectangle(aStartRow, aStartColumn, aEndRow, aEndColumn)

String getTextFromRectangle(Object aStartRow, Object aStartColumn, Object aEndRow, Object aEndColumn)

Composer extension method. Returns text string concatenated from substrings which compose rectangular area of the screen text buffer determined by coordinates: start position—aStartRow, aStartColumn end position—aEndRow, aEndColumn.

setText(asText)

String setText(Object asText) Composer extension method. Sets text asText in the screen text buffer starting from current offset.

UTS

HOSTID

HOSTID As Defined in Connection

PASSWORD

PASSWORD As Defined in Connection

USERID

USERID As Defined in Connection

getAttribute(aRow, aColumn)

int getAttribute(Object aRow, Object aColumn) Composer extension method. Returns attribute of a character at a UTS screen position at row aRow and column aColumn.

getCols()

int getCols() Composer extension method. Returns number of columns from the UTS screen.

getCursorCol()

int getCursorCol() Composer extension method. Returns current UTS screen cursor column position.

getCursorRow()

int getCursorRow() Composer extension method. Returns current UTS screen cursor row position.

getNextMessage()

getNextMessage() Composer extension method. Returns the next captured screen data message.

getPrompt()

String getPrompt() Composer extension method. Returns text from cursor back to the left margin of the UTS screen.

getRows()

int getRows() Composer extension method. Returns number of rows from the UTS screen.

getText(aRow, aColumn, aLength)

String getText(Object aRow, Object aColumn, Object aLength) Composer extension method. Returns text string found at position aRow and aColumn for aLength characters.

getTextFromRectangle(aStartRow, aStartColumn, aEndRow, aEndColumn)

String getTextFromRectangle(Object aStartRow, Object aStartColumn, Object aEndRow, Object aEndColumn)

Composer extension method. Returns block of text.

hasMoreMessages()

boolean hasMoreMessages() Composer extension method. Returns true if more captured screen data messages are available to obtain via the getNextMessage() method.

putString(aRow, aColumn, asText)

putString(Object aRow, Object aColumn, Object asText) Composer extension method. Sets text asText in the screen at position aRow and aColumn.

putStringInField(aField, asText)

putStringInField(Object aField, Object asText) Composer extension method. Sets text asText in the screen at position aField.

setMessageCaptureOff()

setMessageCaptureOff() Composer extension method. Turns off screen data message capture.

setMessageCaptureOn()

setMessageCaptureOn() Composer extension method. Sets up the capture of a screen data message from the host immediately following a Transmit. Use getNextMessage() to retrieve data, then use setMessageCaptureOff() to turn off.

typeKeys(asKeyText)

typeKeys(Object asKeyText) Composer extension method. Keystroke represented by asKeyText is emulated on the screen.

KEYBOARD EQUIVALENTS

MsgWait	Keyboard equivalent: Ctrl + W
SOE	Keyboard equivalent: Ctrl + S
Transmit	Keyboard equivalent: Enter
UnlckKbd	Keyboard equivalent: Esc
F1	Keyboard equivalent: F1
F2	Keyboard equivalent: F2
F3	Keyboard equivalent: F3
F4	Keyboard equivalent: F4
F5	Keyboard equivalent: F5
F6	Keyboard equivalent: F6
F7	Keyboard equivalent: F7
F8	Keyboard equivalent: F8
F9	Keyboard equivalent: F9
F10	Keyboard equivalent: F10
F11	Keyboard equivalent: F11
F12	Keyboard equivalent: F12
F13	Keyboard equivalent: Shift + F1
F14	Keyboard equivalent: Shift + F2
F15	Keyboard equivalent: Shift + F3
F16	Keyboard equivalent: Shift + F4
F17	Keyboard equivalent: Shift + F5
F18	Keyboard equivalent: Shift + F6
F19	Keyboard equivalent: Shift + F7
F20	Keyboard equivalent: Shift + F8
F21	Keyboard equivalent: Shift + F9
F22	Keyboard equivalent: Shift + F10
F23	Keyboard equivalent: Shift + F11
F24	Keyboard equivalent: Shift + F12

ECMAScript Core

NUMBER OBJECT

MAX_VALUE

Number.MAX_VALUE The largest positive finite value of the number type, which is approximately 1.7976931348623157e308

MIN_VALUE

Number.MIN_VALUE The smallest positive nonzero value of the number type, which is approximately 5e-324

NaN

Number.NaN The primitive value NaN represents the set of IEEE Standard "Not-a-Number" values.

NEGATIVE_INFINITY

Number.NEGATIVE_INFINITY The value of negative infinity.

Number()

Number() Constructor of Number has two forms: Number(value) and Number().

POSITIVE_INFINITY

Number.POSITIVE_INFINITY The value of positive infinity.

toString(radix)

toString() If the radix is the number 10 or not supplied, then this number value is given as an argument to the ToString operator; the resulting string value is returned. If the radix is supplied and is an integer from 2 to 36, but not 10, the result is a string, the choice of which is implementation-dependent. The toString function is not generic; it generates a runtime error if its this value is not a Number object. Therefore, it cannot be transferred to other kinds of objects for use as a method.

valueOf()

valueOf() Returns this number value. The valueOf function is not generic; it generates a runtime error if its this value is not a Number object. Therefore, it cannot be transferred to other kinds of objects for use as a method.

STRING OBJECT

String(x)

String(x) constructor of the string.

charAt(pos)

charAt(pos) Method returns a string containing the character at position pos in the string resulting from converting this object to a string. If there is no character at that position, the result is the empty string. The result is a string value, not a string object.

charCodeAt(pos)

charCodeAt(pos) Method returns a number (a nonnegative integer less than 2^16) representing the Unicode code point encoding of the character at position pos in the string resulting from converting this object to a string. If there is no character at that position, the result is NaN.

fromCharCode(char0, char1, . . .)

fromCharCode(char0, char1, . . .) returns a string value containing as many characters as the number of arguments. Each argument specifies one character of the resulting string, with the first argument specifying the first character, and so on, from left to right. An argument is converted to a character by applying the operation ToUint16 and regarding the resulting 16-bit integer as the Unicode code point encoding of a character. If no arguments are supplied, the result is the empty string.

indexOf(searchString, pos)

indexOf(searchString, pos) If the given searchString appears as a substring of the result of converting this object to a string, at one or more positions that are at or to the right of the specified position, then the index of the leftmost such position is returned; otherwise, -1 is returned. If position is undefined or not supplied, 0 is assumed, so as to search all of the string.

lastIndexOf(searchString, pos)

lastIndexOf(searchString, pos) If the given searchString appears as a substring of the result of converting this object to a string, at one or more positions that are at or to the left of the specified position, then the index of the rightmost such position is returned; otherwise, -1 is returned. If position is undefined or not supplied, the length of the string value is assumed, so as to search all of the string.

length

length property equals to the number of characters in the String value represented by this string object.

match(RegExp)

String match(RegExp) Takes a regular expression object as argument. It returns an Array of matches, else null.

replace(RegExp, String)

String replace(RegExp, String) Takes a regular expression and a replacement string. Returns original string with replacements accomplished.

search(RegExp)

String search(RegExp) Takes a regular expression as the sole arg and returns the offset of the first substring that matches, or -1 on no match.

split(separator)

split(separator) Method returns an Array object, into which substrings of the result of converting this object to a string have been stored. The substrings are determined by searching from left to right for occurrences of the given separator; these occurrences are not part of any substring in the returned array, but serve to divide up the string value. The separator may be a string of any length.

substring(start, end)

substring(start, end) Method returns a substring of the result of converting this object to a string, starting from character position start and running to the position end of the string. If second parameter is not present end position is considered end of the string. The result is a string value, not a string object.

toLowerCase()

toLowerCase() Method returns a string equal in length to the length of the result of converting this object to a string. The result is a string value, not a string object. Every character of the result is equal to the corresponding character of the string, unless that character has a Unicode 2.0 lowercase equivalent, in which case the lowercase equivalent is used instead. (The canonical Unicode 2.0 case mapping shall be used, which does not depend on implementation or locale.

toString()

toString() Method returns this string value. When concerned with the placement and use of whitespace line terminators, and semicolons within the representation string is implementation-dependent.

toUpperCase()

toUpperCase() Method returns a string equal in length to the length of the result of converting this object to a string. The result is a string value, not a string object. Every character of the result is equal to the corresponding character of the string, unless that character has a Unicode 2.0 uppercase equivalent, in which case the uppercase equivalent is used instead. The canonical Unicode 2.0 case mapping shall be used, which does not depend on implementation or locale.

valueOf()

valueOf() Method returns this string value. The valueOf() function is not generic. Therefore it will generate a runtime error if the object is not a String object.

DATE OBJECT

Date()

Date() constructor of the Date may have various signatures. The date constructor format can accept up to 7 parameters. Here is the format: new Date(year,month,date,hrs,mins,secs,ms)

getDate()

getDate() Method returns DateFromTime(LocalTime(t)).

getDay()

getDay() Method returns WeekDay(LocalTime(t)). The days of week are numbered from 0 -6. The number 0 represents Sunday and 6 represents Saturday.

getFullYear()

getFullYear() Method returns YearFromTime(LocalTime(t)).

getHours()

getHours() Method returns HourFromTime(LocalTime(t)).

getMilliseconds()

getMilliseconds() Method returns msFromTime(LocalTime(t)).

getMinutes()

getMinutes() Method returns MinFromTime(LocalTime(t)).

getMonth()

getMonth() Method returns MonthFromTime(LocalTime(t)). The months are returned as an integer value from 0-11. The number 0 represents January and 11 represents December.

getSeconds()

getSeconds() Method returns SecFromTime(LocalTime(t)).

getTime()

getTime() Method returns a number, which is this time value. The number value is a millisecond representation of the specified Date object.

getTimezoneOffset()

getTimezoneOffset() Method returns (t * LocalTime(t)) / msPerMinute. The difference is in minutes between (GMT) and local time.

getUTCDate()

getUTCDate() Method returns DateFromTime(t).

getUTCDay()

getUTCDay() Method returns WeekDay(t). The days of week are numbered from 0 -6. The number 0 represents Sunday and 6 represents Saturday.

getUTCFullYear()

getUTCFullYear() Method returns YearFromTime(t). There does not exist a getYearUTC method, therefore this method must be used to obtain a year from an UTC Date object.

getUTCHours()

getUTCHours() Method returns HourFromTime(t).

getUTCMilliseconds()

getUTCMilliseconds() Method returns msFromTime(t).

getUTCMinutes()

getUTCMinutes() Method returns MinFromTime(t).

getUTCSeconds()

getUTCSeconds() Method returns SecFromTime(t).

getYear()

getYear() Method returns YearFromTime(LocalTime(t))—1900. \The function getFullYear() is much to be preferred for nearly all purposes, because it avoids the year 2000 problem.

parse(string)

parse(string) Method applies the ToString operator to its argument and interprets the resulting string as a date; it returns a number, the UTC time value corresponding to the date. The string may be interpreted as a local time, a UTC time, or a time in some other time zone, depending on the contents of the string.

setDate(date)

setDate(date) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 30 or 31, the [Value] of this will then be added to the existing date value, not set.

setFullYear(year[,mon[,date]])

setFullYear(year[,mon[,date]]) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value.

setHours(hour[,min[,sec[,ms]]])

setHours(hour[,min[,sec[,ms]]]) Method sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of the this value. When entering a value for hours, an hour value greater than 23 will be added on to the exisitng hour value, not set.

setMilliseconds(ms)

setMilliseconds(ms) Method computes UTC from argument and sets the [Value] property of this value to TimeClip(calculatedUTCtime). Returns the value of the [Value] property of the this value.

setMinutes(min[,sec[,ms]])

setMinutes(min[,sec[,ms]]) Method sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of the this value.

setMonth(mon[,date])

setMonth(mon[,date]) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 11, the [Value] property for this will be added to the existing month, not set

setSeconds(sec [, ms])

setSeconds(sec [, ms]) Method sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of the this value.

setTime(time)

setTime(time) Method sets the [Value] property of the this to TimeClip(time). Returns the value of the [Value] property of the this value. The [Value] property of this is a millisecond value that is converted by the TimeClip(time) method.

setUTCDate(date)

setUTCDate(date) Method sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 30 or 31, the [Value] of this will then be added to the existing date value, not set.

setUTCFullYear(year[,mon[,date]])

setUTCFullYear(year[,mon[,date]]) Method sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of the this value.

setUTCFullYear(year[,mon[,date]])

toGMTString() Returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in UTC. NOTE: This function is for backwards compatibility only. Its use is not recomended. Use toUTCString() instead.

setUTCHours(min[,sec[,ms]])

setUTCHours(min[,sec[,ms]]) Method sets the [Value] property of this value to time. Returns the value of the [Value] property of the this value. When entering a value for hours, an hour value greater than 23 will be added on to the exisitng hour value, not set.

setUTCMilliseconds(ms)

setUTCMilliseconds(ms) Method sets the [Value] property of this value to time and returns the value of the [Value] property of the this value.

setUTCMinutes(min[,sec[,ms]])

setUTCMinutes(min[,sec[,ms]]) Method sets the [Value] property of this value to time. Returns the value of the [Value] property of the this value.

setUTCMonth(mon[,date])

setUTCMonth(mon[,date]) Method sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 11, the [Value] property for this will be added to the existing month, not set

setUTCSeconds(sec [, ms])

setUTCSeconds(sec [, ms]) Method sets the [Value] property of this value to time. Returns the value of the [Value] property of the this value.

setYear(year)

setYear(year) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value. Use of this function is not recomended. The function setFullYear is much to be preferred for nearly all purposes, because it avoids the year 2000 problem.

toLocaleString()

toLocaleString() Method returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form appropriate to the geographic or cultural locale.

toString()

toString() Method returns this string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in the current time zone.

toUTCString()

toUTCString() Method returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in UTC.

UTC()

UTC() Method, which may accept different number of arguments. UTC function differs from the Date constructor in two ways: it returns a time value as a number, rather than creating a Date object, and it interprets the arguments in UTC rather than as local time.

valueOf()

valueOf() Method returns a number, which is this time value. The valueOf() function is not generic. Therefore it will generate a runtime error if the object is not a Date object.

ARRAY OBJECT

Array(item0, item1, . . .)

Array() Constructor

join(separator)

Array join(separator) The elements of the array are converted to strings, and these strings are then concatenated, separated by occurrences of the separator. If no separator is provided, a single comma is used as the separator.

length

Array length The length property of this Array object

reverse()

Array reverse() The elements of the array are rearranged so as to reverse their order. The object is returned as the result of the call.

sort(comparefn)

Array sort() The elements of this array are sorted. The sort is not necessarily stable. If comparefn is supplied, it should be a function that accepts two arguments x and y and returns a negative value if x < y, zero if x = y, or a positive value if x > y.

toString()

Array toString() The elements of this object are converted to strings, and these strings are then concatenated, separated by comma characters. The result is the same as if the built-in join method were invoked for this object with no argument.

BOOLEAN OBJECT

Boolean()

Boolean() Constructor

toString()

Boolean toString() If this boolean value is true, then the string "true" is returned. Otherwise, this boolean value must be false, and the string "false" is returned.

valueOf()

Boolean valueOf() Returns this boolean value.

FUNCTION OBJECT

Function(p1, p2, . . . , pn, body)

Function Constructor. The last argument specifies the body (executable code)

of a function; any preceding arguments specify formal parameters.

length

The value of the length property is usually an integer that indicates the "typical" number of arguments expected by the function. However, the language permits the function to be invoked with some other number of arguments. The behaviour of a function when invoked on a number of arguments other than the number specified by its length property depends on the function.

toString()

String toString() An implementation-dependent representation of the function is returned. This representation has the syntax of a FunctionDeclaration. Note in particular that the use and placement of whitespace, line terminators, and semicolons within the representation string is implementation-dependent.

TOP-LEVEL METHODS AND PROPERTIES

ECMAScript provides certain “top-level” methods and properties, so-called because they are not parented off any particular object.

escape(string)

String escape() The escape function computes a new, URL-legal version of a string in which certain URL-illegal characters have been replaced by hexadecimal escape sequences.

eval(x)

eval() When the eval function is called with one argument x, the following steps are taken: 1. If x is not a string value, return x. 2. Parse x as an ECMAScript Program. If the parse fails, generate a runtime error. 3. Evaluate the program from step 2. 4. If Result(3) is "normal" completion after value "V", return the value V. 5. Return undefined.

Infinity

A special primitive value representing positive infinity.

isFinite(number)

isFinite() Applies Number( ) to its argument, then returns false if the result is NaN, +*, or **, and otherwise returns true.

isNaN(number)

isNan() Returns true if the argument evaluates to NaN (“not a number”), otherwise returns false.

NOTE: Any form of logical comparison of NaN against anything else, *including itself*, returns false. Use isNaN( ) to determine whether a variable (or a return value, etc.) is equal to NaN.

NaN

The primitive value NaN represents the set of IEEE standard "Not-a-Number" values.

parseFloat(string)

number parseFloat() Produces a floating-point number by interpretation of the contents of the string argument. If the string cannot be converted to a number, the special value NaN (see above) is returned.

parseInt(string, radix)

number parseInt() Produces an integer value dictated by interpretation of the contents of the string argument, according to the specified radix.

unescape(string)

String unescape() The unescape function computes a new version of a string value in which each escape sequences of the sort that might be introduced by the escape function is replaced with the character that it represents.

MATH OBJECT

All of the Math object’s properties and methods are static, which means you should prepend “Math” to the property or method name in your code. For example, use “Math.PI,” not simply “PI.”

E

The number value for *e*, the base of the natural logarithms, which is approximately 2.7182818284590452354.

LN10

The number value for the natural logarithm of 10, which is approximately 2.302585092994046.

LN2

The number value for the natural logarithm of 2, which is approximately 0.6931471805599453.

LOG2E

The number value for the base-2 logarithm of e, the base of the natural logarithms; this value is approximately 1.4426950408889634. (Note that the value of Math.LOG2E is approximately the reciprocal of the value of Math.LN2.)

LOG10E

The number value for the base-10 logarithm of e, the base of the natural logarithms; this value is approximately 0.4342944819032518. (Note that the value of Math.LOG10E is approximately the reciprocal of the value of Math.LN10.)

PI

The number value for *, the ratio of the circumference of a circle to its diameter, which is approximately 3.14159265358979323846.

SQRT1_2

The number value for the square root of 1/2, which is approximately 0.7071067811865476. (Note that the value of Math.SQRT1_2 is approximately the reciprocal of the value of Math.SQRT2.)

SQRT2

The number value for the square root of 2, which is approximately 1.4142135623730951.

abs(x)

Number abs(x) ECMAScript function of the Math Object This function returns the absolute value of the argument x; in general, the result has the same magnitude as the argument but has positive sign. The input value x can be any number value. Example: Math.abs(-123.23940) = 123.23940

acos(x)

Number acos(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the arc cosine of the argument. The result is expressed in radians and ranges from +0 to +PI(3.14159...)radians. The input value x must be a number between -1.0 and 1.0. Example: PI/4 = 0.785 Math.acos(0.785) = 0.6681001997570769.

asin(x)

Number asin(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the arc sine of the argument. The result is expressed in radians and ranges from -PI/2 to +PI/2. The input value x must be a number between -1.0 and 1.0. Example: PI/4 = 0.785 Math.asin(0.785) = 0.9026961270378197.

atan(x)

Number atan(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the arc tangent of the argument. The result is expressed in radians and ranges from -PI/2 to +PI/2. The input value x can be any number. Example: 3PI/4 = 2.355 Math.atan(2.355) = 1.1692404275454853.

atan2(x,y)

Number atan2(x,y) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the arc tangent of the quotient y/x of the arguments y and x, where the signs of the arguments are used to determine the quadrant of the result. Note that it is intentional and traditional for the two-argument arc tangent function that the argument named y be first and the argument named x be second. The result is expressed in radians and ranges from -PI to +PI. The input value x is the x-coordinate of the point. The input value y is the y-coordinate of the point. Example: PI/2 = 1.57 Math.atan2(1.57,-1.57) = 2.356194490192345.

ceil(x)

Number ceil(x) ECMAScript function of the Math Object This function returns the smallest (closest to -infinity) number value that is not less than the argument and is equal to a mathematical integer. If the argument is already an integer, the result is the argument itself. The input value x can be any numeric value or expression. The Math.ceil(x) function property is the same as -Math.floor(-x). Example:

```
Math.ceil(123.78457) = 123.
```

cos(x)

Number cos(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the cosine of the argument. The argument must be expressed in radians.

exp(x)

Number exp(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the exponential function of the argument (e raised to the power of the argument, where e is the base of the natural logarithms). The input value x can be any numeric value or expression greater than 0. Example:

```
Math.exp(10) = 22026.465794806718.
```

floor(x)

Number floor(x) ECMAScript function of the Math Object This function returns the greatest (closest to +infinity)number value that is not greater than the argument and is equal to a mathematical integer. If the argument is already an integer, the result is the argument itself. The input value x can be any numeric value or expression.

Example:

```
Math.floor(654.895869)=654.
```

log(x)

Number log(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to natural logarithm of the argument. The input value x can be any numeric value or expression greater than 0. Example:

```
Math.log(2) = 0.6931471805599453.
```

max(x,y)

Number max(x,y) ECMAScript function of the Math Object This function returns the larger of the two arguments. The input values x and y can be any numeric values or expressions. Example:

```
Math.max(12.345,12.3456)= 12.3456.
```

min(x,y)

Number min(x,y) ECMAScript function of the Math Object This function returns the smaller of the two arguments. The input values x and y can be any numeric values or expressions. Example:

```
Math.min(-12.457,-12.567)= -12.567.
```

pow(x,y)

Number pow(x,y) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the result of raising x to the power of y. The input value x must be the number raised to a power. The input value y must be the power that x is to be raised to. Example:

```
Math.pow(2,4) = 16.
```

random()

Number random() ECMAScript function of the Math Object This function returns a number value with a positive sign, greater than or equal to 0 but less than 1. The number value is chosen randomly or pseudo randomly with approximately uniform distribution over that range, using an implementation-dependent algorithm or strategy. This function takes no arguments. Example: Math.random()=0.9545176397178535.

round(x)

Number round(x) \ECMAScript function of the Math Object This function returns the number value that is closest to the argument and is equal to a mathematical integer. If two integer number values are equally close to the argument, then the result is the number value that is closer to +infinity. If the argument is already an integer, the result is the argument itself. The input value x can be any number. Example: Math.round(13.53) = 14.

sin(x)

Number sin(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the sine of the argument. The argument is expressed in radians. The input value x must be an angle measured in radians. To convert to radians multiply the input value by 0.017453293(2PI/360). Example: PI/2 = 1.57 Math.sin(1.57) = 0.9999996829318346.

sqrt(x)

Number sqrt(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the square root of the argument. The input value x must be any numeric value or expression greater than or equal to 0. If the input value x is less than zero, the string "NaN" is returned. The string "NaN" stands for "Not a Number".Example: Math.sqrt(25) = 5.

tan(x)

Number tan(x) ECMAScript function of the Math Object This function returns an implementation-dependent approximation to the tangent of the argument. The argument is expressed in radians. The input value x must be an angle measured in radians. To convert to radians multiply the input value by 0.017453293(2PI/360). Example: PI/4 = 0.785 Math.tan(0.785) = 0.9992039901050427

OBJECT

Object( )

Constructor for Object.

toString()

Object toString() When the toString method is called on an arbitrary object, the following steps are taken: 1. Get the [[Class]] property of this object. 2. Compute a string value by concatenating the three strings "[object ", Result(1), and "]". 3. Return Result(2).

valueOf()

Object valueOf() As a rule, the valueOf method for an object simply returns the object; but if the object is a "wrapper" for a host object, as may perhaps be created by the Object constructor, then the contained host object should be returned.

3 DOM Methods

This section lists all DOM-related methods and properties supported by Novell Composer, including not only DOM-1 and DOM-2 extensions (defined by the relevant W3C standards), but also Composer's own ECMAScript extensions. Composer extension methods are specifically noted as such in the text.

Node

attributes

W3C DOM Level 1 Node property. This property returns a NamedNodeMap object of the attributes for the Node.

childNodes

W3C DOM Level 1 Node property. This property returns a NodeList object consisting of the immediate children of the Node.

firstChild

W3C DOM Level 1 Node property. This property returns the first child node of a Node object.

lastChild

W3C DOM Level 1 Node property. This property returns the last child node of a Node object.

nextSibling

W3C DOM Level 1 Node property. This property returns the next sibling node for a Node object.

nodeName

W3C DOM Level 1 Node property. This property returns the node name as a String object.

nodeType

W3C DOM Level 1 Node property. This property returns the node type as a short where 1=Element, 2=Attribute, 3=Text, 4=CDATASection 5=EntityReference, 6=Entity, 7=ProcessingInstruction 8=Comment, 9=Document, 10=DocumentType, 11=DocumentFragment, and 12=Notation.

nodeValue

W3C DOM Level 1 Node property. This property returns the node text data as a String.

ownerDocument

W3C DOM Level 1 Node property. This property returns a Document object.

parentNode

W3C DOM Level 1 Node property. This property returns the parent node object for a Node object.

previousSibling

W3C DOM Level 1 Node property. This property returns the previous sibling node for a Node object.

XML

XML Composer extension property. This property returns a string representing the DOM. Useful in Log actions for debugging components. e.g. Input.XML

appendChild(newChild)

Node appendChild(newChild) W3C DOM Level 1 Node method. This method appends a node as the last child for a Node. The newChild parameter is of type Node.

cloneNode(deep)

Node cloneNode(deep) W3C DOM Level 1 Node method. This method creates an unattached Node object. The deep parameter is of type boolean.

createXPath(XPathType asPattern)

Object createXPath(XPathType asPattern) Composer extension method.Creates the XPath pattern.The XPath Type asPattern only supports abbreviated XPath notation and explicit ordinals.XPath functions are not supported.

hasChildNodes()

boolean hasChildNodes() W3C DOM Level 1 Node method. This method returns a boolean indicating whether the node has children.

insertBefore(newChild, refChild)

Node insertBefore(newChild, refChild) W3C DOM Level 1 Node method. This method inserts a node object into the parent node before the refChild node. The newChild parameter is of type Node. The refChild parameter is of type Node.

removeChild(oldChild)

Node removeChild(oldChild) W3C DOM Level 1 Node method. This method removes a node from a parent and returns an unattached Node. The oldChild parameter is of type Node.

replaceChild(newChild, oldChild)

Node replaceChild(newChild, oldChild) W3C DOM Level 1 Node method. This method replaces one node with another node. The newChild parameter is of type Node. The oldChild parameter is of type Node.

getXML()

String getXML() Composer extension method This property returns a string representing the DOM. Useful in Log actions for debugging components. Input.XPath("root/child").getXML()

ownerDocument

W3C DOM Level 2 modified Node property. Returns the Document object associated with this node. This is also the Document object used to create new nodes. Example: someNodeObject.ownerDocument

namespaceURI

W3C DOM Level 2 Node property. Returns the namespace URI of this node, or null if it is unspecified. Example: someNodeObject.namespaceURI

prefix

W3C DOM Level 2 Node property. Returns the namespace prefix of this node, or null if it is unspecified. Example: someNodeObject.prefix

localName

W3C DOM Level 2 Node property. Returns the local part of the qualified name of this node. Example: someNodeObject.localName

normalize()

void normalize() W3C DOM Level 2 modified Node method. Puts all Text nodes in the full depth of the sub-tree underneath this Node, including attribute nodes, into a "normal" form where only structure (e.g. elements, comments, processing instructions, CDATA sections, and entity references) separates Text nodes, i.e. there are neither adjacent Text nodes nor empty Text nodes.

hasAttributes()

boolean hasAttributes() W3C DOM Level 2 Node method. Returns true if the node has any attributes, false otherwise. Example: Temp.XPath("A/B/C").item(0).hasAttributes()

isSupported(feature, version)

boolean isSupported(feature, version) W3C DOM Level 2 Node method. Returns true if the specified feature is supported on this node, false otherwise. Parameters: feature—valid features include: Core, XML, HTML, Views, Stylesheets, CSS, CSS2, Events, UIEvents, MouseEvents, MutationEvents, HTMLEvents, Range and Traversal. version—This is the version number of the feature to test. In Level 2, version 1, this is the string "2.0". If the version is not specified, supporting any version of the feature will cause the method to return true. Example: aNodeObject.isSupported("Core","2.0")

Document

doctype

W3C DOM Level 1 Document property. This property returns a DocumentType object reflecting the dtd for the document. A Document also has all the properties and methods of Node.

documentElement

W3C DOM Level 1 Document property. This property returns an Element object (the root element)

. A Document also has all the properties and methods of Node.

implementation

W3C DOM Level 1 Document property. This property returns a DOMImplementation object. A Document also has all the properties and methods of Node.

text

Composer extension property. This property returns a concatenated string of all the text nodes (content)

under it.

createAttribute(name)

Attr createAttribute(name) W3C DOM Level 1 Document method. This method returns an unattached Attr object. The name parameter is of type String. A Document also has all the properties and methods of Node.

createCDATASection(data)

CDATASection createCDATASection(data) W3C DOM Level 1 Document method. This method returns an unattached CDATASection object. The data parameter is of type String. A Document also has all the properties and methods of Node.

createComment(data)

Comment createComment(data) W3C DOM Level 1 Document method. This method returns an unattached Comment object. The data parameter is of type String. A Document also has all the properties and methods of Node.

createDocumentFragment()

DocumentFragment createDocumentFragment() W3C DOM Level 1 Document method. This method returns an unattached DocumentFragment. A Document also has all the properties and methods of Node.

createElement(tagName)

Element createElement(tagName) W3C DOM Level 1 Document method. This method creates an unattached Element. The tagName parameter is of type String. A Document also has all the properties and methods of Node.

createEntityReference(name)

EntityReference createEntityReference(name) W3C DOM Level 1 Document method. Creates an unattached EntityReference. The name parameter is of type String. A Document also has all the properties and methods of Node.

createProcessingInstruction(target,data)

ProcessingInstruction createProcessingInstruction(target,data) W3C DOM Level 1 Document method. This method returns an unattached ProcessingInstruction object. The target and data parameters are of type String. A Document also has all the properties and methods of Node.

createTextNode(data)

Text createTextNode(data) W3C DOM Level 1 Document method. This method creates an unattached Text object. The data parameter is of type String. A Document also has all the properties and methods of Node.

getElementsByTagName(tagName)

NodeList getElementsByTagName(tagName) W3C DOM Level 1 Document method. This method returns a NodeList object consisting of the tagname element nodes. The tagName parameter is of type String. A Document also has all the properties and methods of Node.

reset()

void reset() W3C DOM Level 1 Document method. Clears the document.

setDTD(Node RootElementName, Object PublicName, Object URL)

setDTD(Node RootElementName, Object PublicName, Object URL) Composer extension method. Sets DTD file for the document.

setValue(Object aValue)

setValue(Object aValue) Composer extension method. Set the Value of a Document from the passed object, if it is Another document then this method copies child nodes (elements and attributes). If passed object is text, it is parsed to create a DOM.

toString()

String toString() Composer extension method. Converts a DOM document to an XML formatted string. e.g. Input.XPath("root/child").item(0).toString()

transformNodeViaDOM(XSLDOM)

String transformNodeViaDOM(XSLDOM) Composer XSL extension method. Transforms the document according to the XSLDOM and returns a string. The parameter XSLDOM is an XSL stylesheet, that may have been read into the component by an XML Interchange Action. This method could be used in the Source of a Map Action, or call it in a Servlet using the Server Framework class IGXSXSLProcessor.

transformNodeToObject(XSLDOM,OutputDOM)

void transformNodeToObject(XSLDOM,OutputDOM) Composer XSL extension method. Transforms the document according to the XSLDOM and returns results to OutputDOM. The parameter XSLDOM is an XSL stylesheet, that may have been read into the component by an XML Interchange Action. The parameter OutputDOM is the target DOM for the results. From a component this method could be in a Function Action, or from a Custom Script use it once you have all three DOMs, or call it in a Servlet using the Server Framework class IGXSXSLProcessor.

transformNodeViaXSLURL(XSLURLLocation)

String transformNodeViaXSLURL(XSLURLLocation) Composer XSL extension method. Transforms the document according to the XSLURLLocation and returns a string. The parameter XSLURLLocation is an XSL stylesheet. This method could be used in the Source of a Map Action, or from a Custom Script use it once you have a DOM, or call it in a Servlet using the Server Framework class IGXSXSLProcessor.

validate()

boolean validate() Composer extension method. Validates an XML file parsing it against its DTD or Schema.

XPath(String asPattern)

NodeList XPath(XPathType asPattern) Composer extension method. XPathTypes can be of type NodeList, String, Number, or Boolean. Usually used to return a Nodelist matching the XPath pattern. Use brackets to select a particular node from the list. e.g.Input.XPath("INVOICE/LINEITEM[1]") or Input.XPath("INVOICE/LINEITEM[last()]") Use the @ to select a node by attribute. e.g. Input.XPath("INVOICE/LINEITEM[@myattr]") To select by attribute value... Input.XPath("INVOICE/LINEITEM[@myattr='abc']")

importNode(sourceNode, deep)

Node importNode(sourceNode, deep) W3C DOM Level 2 Document method. Imports a node from a document to this document. This method creates a new copy of the sourceNode. The sourceNode is not altered. Parameters: sourceNode—the node to import. deep—a boolean. If true, recursively import the subtree under the specified node; if false, import only the node itself. Example: Temp.importNode(Input.XPath("A/B[2]"), false) A Document also has all the properties and methods of Node.

createElementNS(namespaceURI, qualifiedName)

Element createElementNS(namespaceURI, qualifiedName) W3C DOM Level 2 Document method. Creates an Element of the given qualifiedName and namespaceURI. Parameters: namespaceURI—string of the Element's namespace URI to create. qualifiedName—string of the Element's name to create. Example: Temp.createElementNS("someURI","nsprefix:PRICE") A Document also has all the properties and methods of Node. Note: qualifiedName = namespaceprefix + : + localName

createAttributeNS(namespaceURI, qualifiedName)

Attr createAttributeNS(namespaceURI, qualifiedName) W3C DOM Level 2 Document method. Creates an Attribute of the given qualifiedName and namespaceURI. Parameters: namespaceURI—string of the Attribute's namespace URI to create. qualifiedName—string of the Attribute's name to create. Example: Temp.createAttributeNS("someURI","nsprefix:PRICE") Note: qualifiedName = namespaceprefix + : + localName A Document also has all the properties and methods of Node.

getElementsByTagNameNS(namespaceURI, localName)

NodeList getElementsByTagNameNS(namespaceURI, localName) W3C DOM Level 2 Document method. Returns a NodeList of all the Elements with a given localName and namespace URI in the order in which they are encountered in a preorder traversal of the Document tree. Parameters: namespace URI is a string of the elements to match on. The special value "*" matches all namespaces. localName is a string of the elements to match on. The special value "*" matches all local names. Example: Temp.getElementsByTagNameNS("someURI", "someName") A Document also has all the properties and methods of Node.

getElementById(elementId)

Element getElementById(elementId) W3C DOM Level 2 Document method. Returns the Element whose ID is given by elementId. If no such element exists, returns null. Behavior is not defined if more than one element has this ID. Example: Temp.getElementById("someId") A Document also has all the properties and methods of Node.

setSkipNameSpaces(abFlag)

void setSkipNameSpaces(boolean flag) This method can be used to turn off usage of namespaces and match nodes without any prefixes, behaving like a wildcard match.

setEncoding(encoding)

void setEncoding(String encoding) This method sets character set encoding for the document.

setXPathProcessor(asProcessor)

setXPathProcessor(String asXPathProcessor) Composer extension method. Sets the XPath processor to either "xalan" or "jaxen". Default settings: jaxen, but xalan for HTML Connect Components created in v3.5 and below (HTMLScreenDoc only) Compatability issues: jaxen is faster than xalan. jaxen returns Nodelists via breadth first search while xalan uses depth first.

Element

tagName

W3C DOM Level 1 Element property. This property returns a String object containing the element name. An Element also has all the properties and methods of Node.

text

Composer extension property. This property returns the concatenated text of all the text nodes under it.

booleanValue()

boolean booleanValue() Composer extension method. Returns the boolean value (true | false) of this object if possible.

countOfElement(String propertyName)

Number countOfElement(String propertyName) Composer extension method. Returns a count of the named child.

doubleValue()

double doubleValue() Composer extension method. Returns a double value for this object if possible.

exists(String propertyName)

Boolean exists(String propertyName) Composer extension method. Check for the existence of the named child.

getAttribute(name)

String getAttribute(name) W3C DOM Level 1 Element method. This method returns a String consisting of the attribute value. The name parameter is of type String. An Element also has all the properties and methods of Node.

getAttributeNode(name)

Attr getAttributeNode(name) W3C DOM Level 1 Element method. This method returns an Attr. The name parameter is of type String. An Element also has all the properties and methods of Node.

getElementsByTagName(name)

NodeList getElementsByTagName(name) W3C DOM Level 1 Element method. This method returns a NodeList consisting of the tagname element nodes. The name parameter is of type String. An Element also has all the properties and methods of Node.

getIndex()

int getIndex() Composer extension method. Returns back the current index.

getParent()

Node getParent() Composer extension method. Returns the parent element.

normalize()

void normalize() W3C DOM Level 1 Element method. This method returns a void. An Element also has all the properties and methods of Node.

removeAttribute(name)

void removeAttribute(name) W3C DOM Level 1 Element method. This method removes an attribute from an element. The name parameter is of type String. An Element also has all the properties and methods of Node.

removeAttributeNode(oldAttr)

Attr removeAttributeNode(oldAttr) W3C DOM Level 1 Element method. This method removes an attribute from an element and returns an unattached Attr. The oldAttr parameter is of type Attr. An Element also has all the properties and methods of Node.

setAttribute(name,value)

void setAttribute(name,value) W3C DOM Level 1 Element method. This method sets the value of an attribute node for an element. The name parameter is of type String. The value parameter is of type String. An Element also has all the properties and methods of Node.

setAttributeNode(newAttr)

Attr setAttributeNode(newAttr) W3C DOM Level 1 Element method. This method attaches an attribute node to an element. The newAttr parameter is of type Attr. An Element also has all the properties and methods of Node.

setIndex(int aiIndex)

setIndex(int aiIndex) Composer extension method. Set the iterator index value for this element.

setText(String asText)

setText(String asText) Composer extension method. Set the text node associated with this element.

setValue(Object aValue)

setValue(Object aValue) Composer extension method. Set the Value of an Element from the passed object, if it is another element then this method copies child nodes also (elements and attributes).

toNumber()

Number toNumber() Composer extension method. Get the text node and convert it to a number.

toString()

String toString() Composer extension method. Get the text node associated with this element.

XPath(XPathType asPattern)

NodeList XPath(XPathType asPattern) Composer extension method. XPathTypes can be of type NodeList, String, Number, or Boolean. Usually used to return a Nodelist matching the XPath pattern. Use brackets to select a particular node from the list. e.g.Input.XPath("INVOICE/LINEITEM[1]") or Input.XPath("INVOICE/LINEITEM[last()]") Use the @ to select a node by attribute. e.g. Input.XPath("INVOICE/LINEITEM[@myattr]") To select by attribute value... Input.XPath("INVOICE/LINEITEM[@myattr='abc']")

getAttributeNS(namespaceURI, localName)

string getAttributeNS(namespaceURI, localName) W3C DOM Level 2 Element method. Returns the Attr value as a string. Parameters: namespaceURI—a string of the namespaceURI of the target Attr. localName—is a string of the localName of the target Attr. Example: Temp.XPath("A/B[0]").getAttributeNS("someURI", "someAttr") An Element also has all the properties and methods of Node.

setAttributeNS(namespaceURI, qualifiedName, value)

void setAttributeNS(namespaceURI, qualifiedName, value) W3C DOM Level 2 Element method. Adds a new attribute. If an attribute with the same local name and namespace URI is already present on the element, its prefix is changed to be the prefix part of the qualifiedName, and its value is changed to be the value parameter. Parameters: namespaceURI—a string of the namespace URI of the attribute to create or alter. qualifiedName—is a string of the qualified name of the Attr to create or alter. value—the value to set in string form. Example: Temp.XPath("A/B[0]").setAttributeNS("someURI", "someAttrName", "someAttrvalue") Note: qualifiedName = namespaceprefix + : + localName An Element also has all the properties and methods of Node.

removeAttributeNS(namespaceURI, localName)

void removeAttributeNS(namespaceURI, localName) W3C DOM Level 2 Element method. Removes an attribute by local name and namespace URI. If the removed attribute has a default value it is immediately replaced. The replacing attribute has the same namespace URI and local name, as well as the original prefix. Parameters: namespaceURI—a string of the namespace URI of the attribute to remove. localName—is a string of the name of the Attr to remove. Example: Temp.XPath("A/B[0]").removeAttributeNS("someURI", "someAttrName") An Element also has all the properties and methods of Node.

getAttributeNodeNS(namespaceURI, localName)

Attr getAttributeNodeNS(namespaceURI, localName) W3C DOM Level 2 Element method. Retrieves an Attr node by local name and namespace URI. Parameters: namespaceURI—a string of the namespaceURI of the Attr to retrieve. localName—is a string of the localName of the Attr to retrieve. Example: Temp.XPath("A/B[0]").getAttributeNodeNS("someURI", "someAttr") An Element also has all the properties and methods of Node.

setAttributeNodeNS(newAttr)

Attr setAttributeNodeNS(newAttr) W3C DOM Level 2 Element method. Adds a new attribute. If an attribute with that local name and that namespace URI is already present in the element, it is replaced by the new one. If the newAttr attribute replaces an existing attribute with the same local name and namespace URI, the replaced Attr node is returned, otherwise null is returned. Parameters: newAttr—a new Attr object. Example: Temp.XPath("A/B[0]").setAttributeNodeNS(newAttr) An Element also has all the properties and methods of Node.

getElementsByTagNameNS(namespaceURI, localName)

NodeList getElementsByTagNameNS(namespaceURI, localName) W3C DOM Level 2 Element method. Returns a NodeList of all the descendant Elements with a given local name and namespace URI in the order in which they are encountered in a preorder traversal of this Element tree. Parameters: namespaceURI—a string of the namespace URI of the elements to match on. The special value "*" matches all namespaces. localName—a string of the local name of the elements to match on. The special value "*" matches all local names. Example: Temp.XPath("A/B[0]").getElementsByTagNameNS("someURI", "someName") An Element also has all the properties and methods of Node.

hasAttribute(name)

boolean hasAttribute() W3C DOM Level 2 Element method. Returns true when an attribute with a given name is specified on this element or has a default value, false otherwise. Parameters: name—a string of the attribute name look for. Example: Temp.XPath("A/B[0]").hasAttribute("someName") An Element also has all the properties and methods of Node.

hasAttributeNS(namespaceURI, localName)

boolean hasAttributeNS(namespaceURI, localName) W3C DOM Level 2 Element method. Returns true when an attribute with a given local name and namespace URI is specified on this element or has a default value, false otherwise. Parameters: namespaceURI—a string of the namespaceURI of the attribute to look for. localName—a string of the local name of the attribute to look for. Example: Temp.XPath("A/B[0]").hasAttributeNS("someURI", "someName") An Element also has all the properties and methods of Node.

Attribute

name

W3C DOM Level 1 Attr property. This property returns a String object indicating the attributes tag name. An Attr also has all the properties and methods of Node.

specified

W3C DOM Level 1 Attr property. This property returns a boolean. An Attr also has all the properties and methods of Node.

text

Composer extension property. This property returns the text value of the attribute.

value

W3C DOM Level 1 Attr property. This property returns a String object representing the text value of the attribute. An Attr also has all the properties and methods of Node.

setValue(Object aValue)

setValue(Object aValue) Composer extension method. Set the Value of an Attribute from the passed object.

toString()

String toString() Composer extension method. Get the text node associated with this attribute.

ownerElement

ownerElement W3C DOM Level 2 Attr property. Returns the Element node this attribute is attached to or null if this attribute is not in use. Example: attributeObject.ownerElement An Attr also has all the properties and methods of Node.

CharacterData

data

data W3C DOM Level 1 CharacterData property. This property is of type String and represents the contents of the CharacterData object. A CharacterData also has all the properties and methods of Node.

length

W3C DOM Level 1 CharacterData property. This property represents the length of the CharacterData object. A CharacterData also has all the properties and methods of Node.

appendData(arg)

void appendData(arg) W3C DOM Level 1 CharacterData method. This method appends text to the CharacterData object. The arg parameter is of type String. A CharacterData also has all the properties and methods of Node.

insertData(offset, arg)

void insertData(offset, arg) W3C DOM Level 1 CharacterData method. This method inserts text in the CharacterData object. The offset parameter is of type unsigned long. The arg parameter is of type String. A CharacterData also has all the properties and methods of Node.

deleteData(offset, count)

void deleteData(offset, count) W3C DOM Level 1 CharacterData method. This method deletes text in the CharacterData object. The offset parameter is of type unsigned long. The count parameter is of type unsigned long. A CharacterData also has all the properties and methods of Node.

replaceData(offset, count, arg)

void replaceData(offset, count, arg) W3C DOM Level 1 CharacterData method. This method replaces text in the CharacterData object. The offset parameter is of type unsigned long. The count parameter is of type unsigned long. The arg parameter is of type String. A CharacterData also has all the properties and methods of Node.

substringData(offset, count)

String substringData(offset, count) W3C DOM Level 1 CharacterData method. This method returns a substring of the CharacterData object. The offset parameter is of type unsigned long. The count parameter is of type unsigned long. A CharacterData also has all the properties and methods of Node.

NodeList

length

W3C DOM Level 1 NodeList property. This property returns the number of nodes in a NodeList object.

avg('[NodeList]')

Number avg('[NodeList]') Composer aggregate extension method. Returns a number equal to the average value in the NodeList. The NodeList parameter of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. e.g. Input.XPath("rootElem/childElem").avg() NOTE: The function argument should be under single quote and must be escaped in case of nested calls.

count('[NodeList]')

Number count('[NodeList]') Composer aggregate extension method. Returns a number equal to a count of the nodes in the NodeList that have data. Nodes without data or nodes with only child elements will not be counted. To count all nodes, use the .length property on a nodeList object. The optional NodeList parameter is of type XPath. If no parameter is supplied (the usual case), then the current NodeList/GroupName is used. e.g. Input.XPath("rootElem/childElem").count() NOTE: The function argument should be under single quote and must be escaped in case of nested calls.

item(index)

Node item(index) W3C DOM Level 1 NodeList method. This method returns the indicated Node from the NodeList. The index parameter is of type unsigned long. The Index is 0-based.

min('[NodeList]')

Number min('[NodeList]') Composer aggregate extension method. Returns a number equal to the lowest value in the NodeList. The NodeList parameter of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. e.g. Input.XPath("rootElem/childElem").min() NOTE: The function argument should be under single quote and must be escaped in case of nested calls.

max(['NodeList]')

Number max('[NodeList]') Composer aggregate extension method. Returns a number equal to the highest value in the NodeList. The NodeList parameter of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. e.g. Input.XPath("rootElem/childElem").max() NOTE: The function argument should be under single quote and must be escaped in case of nested calls.

sum('[NodeList]')

Number sum('[NodeList]') Composer aggregate extension method. Returns a number equal to the sum of the values in NodeList. The NodeList parameter of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. e.g. Input.XPath("rootElem/childElem").sum() NOTE: The function argument should be under single quote and must be escaped in case of nested calls.

where(XPathType asPattern)

NodeList where(String asPattern) Composer extension method. Get a NodeList of nodes matching the XPath pattern.

toNumber()

toNumber() Converts the data of the first instance in the NodeList to an ECMAScript Number object. Any alphabetic characters or embedded spaces in data returns NaN. Leading and trailing spaces are okay. Example:

```
var myNum = Input.XPath("Invoice/Amount").toNumber()
```

NamedNodeMap

length

length W3C DOM Level 1 NamedNodeMap property. This property returns the number of nodes in a NamedNodeMap.

getNamedItem(name)

Node getNamedItem(name) W3C DOM Level 1 NamedNodeMap method. This method returns all selected Nodes of the indicated name. The name parameter is of type String.

getNamedItemNS(namespaceURI, localName)

Node getNamedItemNS(namespaceURI, localName) W3C DOM Level 2 NamedNodeMap method. Returns a node specified by local name and namespace URI. Parameters: namespaceURI—a string of the namespaceURI of the Node to retrieve. localName—is a string of the localName of the Node to retrieve. Example: Temp.XPath("A/B").item(0).getAttributes() .getNamedItemNS("someURI", "anAttrName")

item(index)

Node item(index) W3C DOM Level 1 NamedNodeMap method. This method returns the indicated Node from the NamedNodeMap. The index parameter is of type unsigned long. The index is 0-based.

removeNamedItem(name)

Node removeNamedItem(name) W3C DOM Level 1 NamedNodeMap method. This method removes the indicated node from the NamedNodeMap and returns an unattached node. The name parameter is of type String.

removeNamedItemNS(namespaceURI, localName)

Node removeNamedItemNS(namespaceURI, localName) W3C DOM Level 2 NamedNodeMap method. Removes and returns the node specified by namespace URI and local name. Parameters: namespaceURI—a string of the namespaceURI of the Node to remove. localName—is a string of the localName of the Node to remove. Example:

```
Temp.XPath("A/B").item(0).getAttributes() .removeNamedItemNS("someURI",
"anAttrName")
```

setNamedItem(arg)

Node setNamedItem(arg) W3C DOM Level 1 NamedNodeMap method. This method returns a Node. The arg parameter is of type Node.

setNamedItemNS(Node arg)

Node setNamedItemNS(arg) W3C DOM Level 2 NamedNodeMap method. If the new Node replaces an existing node the replaced Node is returned, otherwise null is returned. Example:

```
var item = Temp.XPath("A/B").item(0);
item.getAttributes().setNamedItemNS(aNodeObject)
```

Text

splitText(offset)

Text splitText(offset) W3C DOM Level 1 Element method. This method removes the text up to the offset and creates an unattached text node with the removed text. The offset parameter is of type unsigned long. A Text also has all the properties and methods of CharacterData.

CDATASection

(none currently defined) W3C DOM Level 1. CDATASection also has all the properties and methods of Text.

DocumentType

name

W3C DOM Level 1 DocumentType property. This property returns a String representing the document type name.

entities

W3C DOM Level 1 DocumentType property. This property returns a NamedNodeMap of the entities defined in the document.

internalSubset

W3C DOM Level 2 DocumentType property. This property returns a String representing the internal subset as a string.

notations

W3C DOM Level 1 DocumentType property. This property returns a NamedNodeMap of the notations defined in the document.

publicId

W3C DOM Level 2 DocumentType property. This property returns a String representing the public identifier of the external subset.

systemId

W3C DOM Level 2 DocumentType property. This property returns a String representing the system identifier of the external subset.

Comment

(none currently defined) W3C DOM Level 1 Comment. Comment also has all the properties and methods of CharacterData.

DOMImplementation

createDocument(namespaceURI, qualifiedName, doctype)

Document createDocument(namespaceURI, qualifiedName, doctype) W3C DOM Level 2 DOMImplementation method. Creates an XML Document object of the specified type with its document element. Parameters: namespaceURI—a string of the namespace URI of the document element to create. qualifiedName—a string of the name of the document element to create. doctype—is the type of document to create or null. Note: qualifiedName = namespaceprefix + : + localName

createDocumentType(qualifiedName, publicID, systemID)

DocumentType createDocumentType(qualifiedName, publicID, systemID) W3C DOM Level 2 DOMImplementation method. Creates an empty DocumentType node. Parameters: qualifiedName—is a string of the name of the document type to create. publicID is the external subset public identifier. systemID is the external subset system identifier. Note: qualifiedName = namespaceprefix + : + localName

hasFeature(feature, version)

boolean hasFeature(feature, version) W3C DOM Level 1 DOMImplementation method. This method returns a boolean. The feature parameter is of type String. The version parameter is of type String.

DocumentFragment

(none currently defined) W3C DOM Level 1 object. DocumentFragment also has all the properties and methods of Node.

Notation

publicId

W3C DOM Level 2 This property returns a String representing the public identifier of the external subset.

systemId

W3C DOM Level 2 This property returns a String representing the system identifier of the external subset.

Entity

publicId

W3C DOM Level 2 This property returns a String representing the public identifier of the external subset.

systemId

W3C DOM Level 2 This property returns a String representing the system identifier of the external subset.

notationName

W3C DOM Level 1 Entity property. This property is of type String. An Entity also has all the properties and methods of Node.

EntityReference

(none currently defined) W3C DOM Level 1 EntityReference. An EntityReference also has all the properties and methods of Node.

ProcessingInstruction

target

W3C DOM Level 1 ProcessingInstruction property. This property is a String representation of the target part of a Processing Instruction.

data

W3C DOM Level 1 ProcessingInstruction property. This property is a String representation of the data part of a Processing Instruction.

4 ECMAScript Core Language

Array Object

Array(item0, item1, . . .)

Array() Constructor

join(separator)

Array join(separator) The elements of the array are converted to strings, and these strings are then concatenated, separated by occurrences of the separator. If no separator is provided, a single comma is used as the separator.

length

Array length The length property of this Array object

pop()

Object pop() Removes tail item from array and returns it. The array is permanently modified: It is shortened in length by one item.

push()

push(Object object) Adds an item to the end of the array. The array grows in length by one.

reverse()

reverse() The elements of the array are rearranged so as to reverse their order. The operation is done in-place, meaning that the original array is modified.

shift()

Object shift() Returns the first (actually, the zeroth) element of the array *and removes it from the array.* The array is shortened by one item. All remaining items are "shifted" such that each item's index is one less than before. Example:

```
var ar = [ 'a','b','c' ];
var firstItem = ar.shift(); // 'a'
var item = ar[0]; // 'b'
```

sort(comparefn)

Array sort() The elements of this array are sorted. The sort is not necessarily stable. If comparefn is supplied, it should be a function that accepts two arguments x and y and returns a negative value if x < y, zero if x = y, or a positive value if x > y.

toString()

Array toString() The elements of this object are converted to strings, and these strings are then concatenated, separated by comma characters. The result is the same as if the built-in join method were invoked for this object with no argument.

unshift()

unshift(value, [...]) Adds one or more values to the *start* of the array. If a single argument is passed, it becomes item zero of the array and the array grows in length by one.

Boolean Object

There is seldom a need to use the object version of Boolean in place of true/false literal values. This object is provided for completeness. It is specified in ECMA-262.

Boolean()

Boolean([true/false]) Constructor. Optionally takes one of `true` or `false` as an argument.

toString()

Boolean toString() If this boolean value is true, then the string "true" is returned. Otherwise, this boolean value must be false, and the string "false" is returned.

valueOf()

Boolean valueOf() Returns this boolean value.

Date Object

Date()

Date() constructor of the Date may have various signatures. The date constructor format can accept up to 7 parameters. Here is the format: new Date(year,month,date,hrs,mins,secs,ms)

getDate()

getDate() Method returns DateFromTime(LocalTime(t)).

getDay()

getDay() Method returns WeekDay(LocalTime(t)). The days of week are numbered from 0 -6. The number 0 represents Sunday and 6 represents Saturday.

getFullYear()

getFullYear() Method returns YearFromTime(LocalTime(t)).

getHours()

getHours() Method returns HourFromTime(LocalTime(t)).

getMilliseconds()

getMilliseconds() Method returns msFromTime(LocalTime(t)).

getMinutes()

getMinutes() Method returns MinFromTime(LocalTime(t)).

getMonth()

getMonth() Method returns MonthFromTime(LocalTime(t)). The months are returned as an integer value from 0-11. The number 0 represents January and 11 represents December.

getSeconds()

getSeconds() Method returns SecFromTime(LocalTime(t)).

getTime()

getTime() Method returns a number, which is this time value. The number value is a millisecond representation of the specified Date object.

getTimezoneOffset()

getTimezoneOffset() Method returns (t * LocalTime(t)) / msPerMinute. The difference is in minutes between (GMT) and local time.

getUTCDate()

getUTCDate() Method returns DateFromTime(t).

getUTCDay()

getUTCDay() Method returns WeekDay(t). The days of week are numbered from 0 -6. The number 0 represents Sunday and 6 represents Saturday.

getUTCFullYear()

getUTCFullYear() Method returns YearFromTime(t). There does not exist a getYearUTC method, therefore this method must be used to obtain a year from an UTC Date object.

getUTCHours()

getUTCHours() Method returns HourFromTime(t).

getUTCMilliseconds()

getUTCMilliseconds() Method returns msFromTime(t).

getUTCMinutes()

getUTCMinutes() Method returns MinFromTime(t).

getUTCSeconds()

getUTCSeconds() Method returns SecFromTime(t).

getYear()

getYear() Method returns YearFromTime(LocalTime(t))—1900. \The function getFullYear() is much to be preferred for nearly all purposes, because it avoids the year 2000 problem.

parse(string)

parse(string) Method applies the ToString operator to its argument and interprets the resulting string as a date; it returns a number, the UTC time value corresponding to the date. The string may be interpreted as a local time, a UTC time, or a time in some other time zone, depending on the contents of the string.

setDate(date)

setDate(date) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 30 or 31, the [Value] of this will then be added to the existing date value, not set.

setFullYear(year[,mon[,date]])

setFullYear(year[,mon[,date]]) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value.

setHours(hour[,min[,sec[,ms]]])

setHours(hour[,min[,sec[,ms]]]) Method sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of the this value. When entering a value for hours, an hour value greater than 23 will be added on to the exisitng hour value, not set.

setMilliseconds(ms)

setMilliseconds(ms) Method computes UTC from argument and sets the [Value] property of this value to TimeClip(calculatedUTCtime). Returns the value of the [Value] property of the this value.

setMinutes(min[,sec[,ms]])

setMinutes(min[,sec[,ms]]) Method sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of the this value.

setMonth(mon[,date])

setMonth(mon[,date]) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 11, the [Value] property for this will be added to the existing month, not set

setSeconds(sec [, ms])

setSeconds(sec [, ms]) Method sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of the this value.

setTime(time)

setTime(time) Method sets the [Value] property of the this to TimeClip(time). Returns the value of the [Value] property of the this value. The [Value] property of this is a millisecond value that is converted by the TimeClip(time) method.

setUTCDate(date)

setUTCDate(date) Method sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 30 or 31, the [Value] of this will then be added to the existing date value, not set.

setUTCFullYear(year[,mon[,date]])

setUTCFullYear(year[,mon[,date]]) Method sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of the this value.

setUTCFullYear(year[,mon[,date]])

toGMTString() Returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in UTC. NOTE: This function is for backwards compatibility only. Its use is not recomended. Use toUTCString() instead.

setUTCHours(min[,sec[,ms]])

setUTCHours(min[,sec[,ms]]) Method sets the [Value] property of this value to time. Returns the value of the [Value] property of the this value. When entering a value for hours, an hour value greater than 23 will be added on to the exisitng hour value, not set.

setUTCMilliseconds(ms)

setUTCMilliseconds(ms) Method sets the [Value] property of this value to time and returns the value of the [Value] property of the this value.

setUTCMinutes(min[,sec[,ms]])

setUTCMinutes(min[,sec[,ms]]) Method sets the [Value] property of this value to time. Returns the value of the [Value] property of the this value.

setUTCMonth(mon[,date])

setUTCMonth(mon[,date]) Method sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of the this value. If the [Value] property of this exceeds 11, the [Value] property for this will be added to the existing month, not set

setUTCSeconds(sec [, ms])

setUTCSeconds(sec [, ms]) Method sets the [Value] property of this value to time. Returns the value of the [Value] property of the this value.

setYear(year)

setYear(year) Method sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of the this value.

toLocaleString()

toLocaleString() Method returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form appropriate to the geographic or cultural locale.

toString()

toString() Method returns this string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in the current time zone.

toUTCString()

toUTCString() Method returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in UTC.

UTC()

UTC() Method, which may accept different number of arguments. UTC function differs from the Date constructor in two ways: it returns a time value as a number, rather than creating a Date object, and it interprets the arguments in UTC rather than as local time.

valueOf()

valueOf() Method returns a number, which is this time value. The valueOf() function is not generic. Therefore it will generate a runtime error if the object is not a Date object.

Function Object

Function(p1, p2, . . . , pn, body)

Function Constructor. The last argument specifies the body (executable code)

of a function; any preceding arguments specify formal parameters.

length

The value of the length property is usually an integer that indicates the "typical" number of arguments expected by the function. However, the language permits the function to be invoked with some other number of arguments. The behaviour of a function when invoked on a number of arguments other than the number specified by its length property depends on the function.

toString()

String toString() An implementation-dependent representation of the function is returned. This representation has the syntax of a FunctionDeclaration. Note in particular that the use and placement of whitespace, line terminators, and semicolons within the representation string is implementation-dependent.

Math Object

All of the Math object's properties and methods are static, which means you should prepend "Math" to the property or method name in your code. For example, use "Math.PI," not simply "PI."

E

The number value for *e*, the base of the natural logarithms, which is approximately 2.7182818284590452354.

LN10

The number value for the natural logarithm of 10, which is approximately 2.302585092994046.

LN2

The number value for the natural logarithm of 2, which is approximately 0.6931471805599453.

LOG2E

The number value for the base-2 logarithm of e, the base of the natural logarithms; this value is approximately 1.4426950408889634. (Note that the value of Math.LOG2E is approximately the reciprocal of the value of Math.LN2.)

LOG10E

The number value for the base-10 logarithm of e, the base of the natural logarithms; this value is approximately 0.4342944819032518. (Note that the value of Math.LOG10E is approximately the reciprocal of the value of Math.LN10.)

PI

The number value for *, the ratio of the circumference of a circle to its diameter, which is approximately 3.14159265358979323846.

SQRT1_2

The number value for the square root of 1/2, which is approximately 0.7071067811865476. (Note that the value of Math.SQRT1_2 is approximately the reciprocal of the value of Math.SQRT2.)

SQRT2

The number value for the square root of 2, which is approximately 1.4142135623730951.

abs(x)

Number abs(x) This function returns the absolute value of the argument x; in general, the result has the same magnitude as the argument but has positive sign. The input value x can be any number value. Example: Math.abs(-123.23940) = 123.23940

acos(x)

Number acos(x) This function returns an implementation-dependent approximation to the arc cosine of the argument. The result is expressed in radians and ranges from +0 to +PI(3.14159...)radians. The input value x must be a number between -1.0 and 1.0. Example: PI/4 = 0.785 Math.acos(0.785) = 0.6681001997570769.

asin(x)

Number asin(x) This function returns an implementation-dependent approximation to the arc sine of the argument. The result is expressed in radians and ranges from -PI/2 to +PI/2. The input value x must be a number between -1.0 and 1.0. Example: PI/4 = 0.785 Math.asin(0.785) = 0.9026961270378197.

atan(x)

Number atan(x) This function returns an implementation-dependent approximation to the arc tangent of the argument. The result is expressed in radians and ranges from -PI/2 to +PI/2. The input value x can be any number. Example: 3PI/4 = 2.355 Math.atan(2.355) = 1.1692404275454853.

atan2(x,y)

Number atan2(x,y) This function returns an implementation-dependent approximation to the arc tangent of the quotient y/x of the arguments y and x, where the signs of the arguments are used to determine the quadrant of the result. Note that it is intentional and traditional for the two-argument arc tangent function that the argument named y be first and the argument named x be second. The result is expressed in radians and ranges from -PI to +PI. The input value x is the x-coordinate of the point. The input value y is the y-coordinate of the point. Example: PI/2 = 1.57 Math.atan2(1.57,-1.57) = 2.356194490192345.

ceil(x)

Number ceil(x) This function returns the smallest (closest to -infinity) number value that is not less than the argument and is equal to a mathematical integer. If the argument is already an integer, the result is the argument itself. The input value x can be any numeric value or expression. The Math.ceil(x) function property is the same as -Math.floor(-x). Example:

```
Math.ceil(123.78457) = 123.
```

cos(x)

Number cos(x) This function returns an implementation-dependent approximation to the cosine of the argument. The argument must be expressed in radians.

exp(x)

Number exp(x) This function returns an implementation-dependent approximation to the exponential function of the argument (e raised to the power of the argument, where e is the base of the natural logarithms). The input value x can be any numeric value or expression greater than 0. Example:

```
Math.exp(10) = 22026.465794806718.
```

floor(x)

Number floor(x) This function returns the greatest (closest to +infinity)number value that is not greater than the argument and is equal to a mathematical integer. If the argument is already an integer, the result is the argument itself. The input value x can be any numeric value or expression.

Example:

```
Math.floor(654.895869)=654.
```

log(x)

Number log(x) This function returns an implementation-dependent approximation to natural logarithm of the argument. The input value x can be any numeric value or expression greater than 0. Example:

```
Math.log(2) = 0.6931471805599453.
```

max(x,y)

Number max(x,y) This function returns the larger of the two arguments. The input values x and y can be any numeric values or expressions. Example:

```
Math.max(12.345,12.3456)= 12.3456.
```

min(x,y)

Number min(x,y) This function returns the smaller of the two arguments. The input values x and y can be any numeric values or expressions. Example:

```
Math.min(-12.457,-12.567)= -12.567.
```

pow(x,y)

Number pow(x,y) This function returns an implementation-dependent approximation to the result of raising x to the power of y. The input value x must be the number raised to a power. The input value y must be the power that x is to be raised to. Example:

```
Math.pow(2,4) = 16.
```

random()

Number random() This function returns a positive floating-point number in the unit interval (0..1). The number value is chosen randomly or pseudo randomly with approximately uniform distribution over that range, using an implementation-dependent algorithm or strategy. This function takes no arguments. Example:

```
Math.random()=0.9545176397178535.
```

round(x)

Number round(x) This function returns the number value that is closest to the argument and is equal to a mathematical integer. If two integer number values are equally close to the argument, then the result is the number value that is closer to +infinity. If the argument is already an integer, the result is the argument itself. The input value x can be any number. Example:

```
Math.round(13.53) = 14.
```

sin(x)

Number sin(x) This function returns an implementation-dependent approximation to the sine of the argument. The argument is expressed in radians. The input value x must be an angle measured in radians.

sqrt(x)

Number sqrt(x) This function returns an implementation-dependent approximation to the square root of the argument. The input value x must be any numeric value or expression greater than or equal to 0. If the input value x is less than zero, the string "NaN" is returned. The string "NaN" stands for "Not a Number". Example:

```
Math.sqrt(25) = 5.
```

tan(x)

Number tan(x) This function returns an implementation-dependent approximation to the tangent of the argument. The argument is expressed in radians. The input value x must be an angle measured in radians.

Number Object

MAX_VALUE

Number.MAX_VALUE The largest positive finite value of the number type, which is approximately 1.7976931348623157e308

MIN_VALUE

Number.MIN_VALUE The smallest positive nonzero value of the number type, which is approximately 5e-324

NaN

Number.NaN The primitive value NaN represents the set of IEEE Standard "Not-a-Number" values.

NEGATIVE_INFINITY

Number.NEGATIVE_INFINITY The value of negative infinity.

Number()

Number() Constructor of Number has two forms: Number(value) and Number().

POSITIVE_INFINITY

Number.POSITIVE_INFINITY The value of positive infinity.

toString(radix)

toString() If the radix is the number 10 or not supplied, then this number value is given as an argument to the ToString operator; the resulting string value is returned. If the radix is supplied and is an integer from 2 to 36, but not 10, the result is a string, the choice of which is implementation-dependent. The toString function is not generic; it generates a runtime error if its this value is not a Number object. Therefore, it cannot be transferred to other kinds of objects for use as a method.

valueOf()

valueOf() Returns this number value. The valueOf function is not generic; it generates a runtime error if its this value is not a Number object. Therefore, it cannot be transferred to other kinds of objects for use as a method.

Object

Object( )

Constructor for Object.

toString()

Object toString() When the toString method is called on an arbitrary object, the following steps are taken: 1. Get the [[Class]] property of this object. 2. Compute a string value by concatenating the three strings "[object ", Result(1), and "]". 3. Return Result(2).

valueOf()

Object valueOf() As a rule, the valueOf method for an object simply returns the object; but if the object is a "wrapper" for a host object, as may perhaps be created by the Object constructor, then the contained host object should be returned.

String Object

String(x)

String(x) constructor of the string.

charAt(pos)

charAt(pos) Method returns a string containing the character at position pos in the string resulting from converting this object to a string. If there is no character at that position, the result is the empty string. The result is a string value, not a string object.

charCodeAt(pos)

charCodeAt(pos) Method returns a number (a nonnegative integer less than 2^16) representing the Unicode code point encoding of the character at position pos in the string resulting from converting this object to a string. If there is no character at that position, the result is NaN.

fromCharCode(char0, char1, . . .)

fromCharCode(char0, char1, . . .) returns a string value containing as many characters as the number of arguments. Each argument specifies one character of the resulting string, with the first argument specifying the first character, and so on, from left to right. An argument is converted to a character by applying the operation ToUint16 and regarding the resulting 16-bit integer as the Unicode code point encoding of a character. If no arguments are supplied, the result is the empty string.

indexOf(searchString, pos)

indexOf(searchString, pos) If the given searchString appears as a substring of the result of converting this object to a string, at one or more positions that are at or to the right of the specified position, then the index of the leftmost such position is returned; otherwise, -1 is returned. If position is undefined or not supplied, 0 is assumed, so as to search all of the string.

lastIndexOf(searchString, pos)

lastIndexOf(searchString, pos) If the given searchString appears as a substring of the result of converting this object to a string, at one or more positions that are at or to the left of the specified position, then the index of the rightmost such position is returned; otherwise, -1 is returned. If position is undefined or not supplied, the length of the string value is assumed, so as to search all of the string.

length

length property equals to the number of characters in the String value represented by this string object.

match(RegExp)

String match(RegExp) Takes a regular expression object as argument. It returns an Array of matches, else null.

replace(RegExp, String)

String replace(RegExp, String) Takes a regular expression and a replacement string. Returns original string with replacements accomplished.

search(RegExp)

String search(RegExp) Takes a regular expression as the sole arg and returns the offset of the first substring that matches, or -1 on no match.

split(separator)

split(separator) Method returns an Array object, into which substrings of the result of converting this object to a string have been stored. The substrings are determined by searching from left to right for occurrences of the given separator; these occurrences are not part of any substring in the returned array, but serve to divide up the string value. The separator may be a string of any length.

substring(start, end)

substring(start, end) Method returns a substring of the result of converting this object to a string, starting from character position start and running to the position end of the string. If second parameter is not present end position is considered end of the string. The result is a string value, not a string object.

toLowerCase()

toLowerCase() Method returns a string equal in length to the length of the result of converting this object to a string. The result is a string value, not a string object. Every character of the result is equal to the corresponding character of the string, unless that character has a Unicode 2.0 lowercase equivalent, in which case the lowercase equivalent is used instead. (The canonical Unicode 2.0 case mapping shall be used, which does not depend on implementation or locale.

toString()

toString() Method returns this string value. When concerned with the placement and use of whitespace line terminators, and semicolons within the representation string is implementation-dependent.

toUpperCase()

toUpperCase() Method returns a string equal in length to the length of the result of converting this object to a string. The result is a string value, not a string object. Every character of the result is equal to the corresponding character of the string, unless that character has a Unicode 2.0 uppercase equivalent, in which case the uppercase equivalent is used instead. The canonical Unicode 2.0 case mapping shall be used, which does not depend on implementation or locale.

valueOf()

valueOf() Method returns this string value. The valueOf() function is not generic. Therefore it will generate a runtime error if the object is not a String object.

Top-Level

ECMAScript provides certain “top-level” methods and properties, so-called because they are available from any context: They are not parented off any particular object.

escape(string)

String escape() The escape function computes a new, URL-legal version of a string in which certain URL-illegal characters have been replaced by hexadecimal escape sequences.

eval(x)

eval() When the eval function is called with one argument x, the following steps are taken: 1. If x is not a string value, return x. 2. Parse x as an ECMAScript Program. If the parse fails, generate a runtime error. 3. Evaluate the program from step 2. 4. If Result(3) is "normal" completion after value "V", return the value V. 5. Return undefined.

Infinity

A special primitive value representing positive infinity.

isFinite(number)

isFinite() Applies Number( ) to its argument, then returns false if the result is NaN, +*, or **, and otherwise returns true.

isNaN(value)

isNan() Returns true if the argument evaluates to NaN (“not a number”), otherwise returns false.

NOTE: Any form of logical comparison of NaN against anything else, *including itself*, returns false. Use isNaN( ) to determine whether a variable (or a return value, etc.) is equal to NaN.

NaN

The primitive value NaN represents the set of IEEE standard "Not-a-Number" values.

parseFloat(string)

number parseFloat() Produces a floating-point number by interpretation of the contents of the string argument. If the string cannot be converted to a number, the special value NaN (see above) is returned.

parseInt(string, radix)

number parseInt() Produces an integer value dictated by interpretation of the contents of the string argument, according to the specified radix.

unescape(string)

String unescape() The unescape function computes a new version of a string value in which each escape sequences of the sort that might be introduced by the escape function is replaced with the character that it represents.