Utils

Apr 05, 2024 22:01

Supported Clients

SmartClient WebClient NGClient MobileClient

Methods Summary

Array	base64ToBytes(base64String:)	Return the byte array representation of the Base64 value
String	base64ToString(base64String:)	Return the String representation of the Base64 value
String	bytesToBase64(byteArray:)	Return the Base64 value representation of the byteArray
String	bytesToHex(bytearray)	
String	bytesToString(byteArray:)	Return the string conversion of the byteArray
String	dateFormat(date, format)	Format a date object to a text representation.
String	dateFormat(date, format, timezone)	Format a date object to a text representation using the format and timezone given.
String	dateFormat(date, format, language, country)	Format a date object to a text representation.
String	dateFormat(date, format, timezone, language, country)	Format a date object to a text representation using the format, timezone, language and country given.
String	getUnicodeCharacter (unicodeCharacterNumber)	Returns a string containing the character for the unicode number.
Boolean	hasRecords(foundset)	Returns true if the (related)foundset exists and has records.
Boolean	hasRecords(record, relationString)	Returns true if the (related)foundset exists and has records.
Array	hexToBytes(hex)	
String	hexToString(hex)	
Boolean	isMondayFirstDayOfWeek()	Returns true when Monday is the first day of the week for your current locale setting.
String	numberFormat(number, digits)	Format a number to have a defined fraction.
String	numberFormat(number, digits, language, country)	Format a number to have a defined fraction.
String	numberFormat(number, format)	Format a number to specification.
String	numberFormat(number, format, language, country)	Format a number to specification.
Date	parseDate(date, format)	Parse a string to a date object.
Date	parseDate(date, format, timezone)	Parse a string to a date object.
Date	parseDate(date, format, language, country)	Parse a string to a date object.
Date	parseDate(date, format, timezone, language, country)	Parse a string to a date object.
String	stringEscapeMarkup(textString)	Returns the escaped markup text (HTML/XML).
String	stringEscapeMarkup(textString, escapeSpaces)	Returns the escaped markup text (HTML/XML).
String	stringEscapeMarkup(textString, escapeSpaces, convertToHtmlUnicodeEscapes)	Returns the escaped markup text (HTML/XML).
String	stringFormat(text_to_format, parameters)	Formats a string according to format specifiers and arguments.
String	stringFormat(text, format)	Format a string using mask.
String	stringIndexReplace(text, i_start, i_size, replacement_text)	Replaces a portion of a string with replacement text from a specified index.
String	stringInitCap(text)	Returns all words starting with capital chars.
String	stringLeft(text, i_size)	Returns a string with the requested number of characters, starting from the left.
String	stringLeftWords(text, numberof_words)	Returns the number of words, starting from the left.
String	stringMD5HashBase16(textString)	Returns the md5 hash (encoded as base16) for specified text.
String	stringMD5HashBase64(textString)	Returns the md5 hash (encoded as base64) for specified text.
String	stringMiddle(text, i_start, i_size)	Returns a substring from the original string.
String	stringMiddleWords(text, i_start, numberof_words)	Returns a substring from the original string.
String	stringPBKDF2Hash(textString)	Returns the PBKDF2 hash for specified text.
String	stringPBKDF2Hash(textString, iterations)	Returns the PBKDF2 hash for specified text.
Number	stringPatternCount(text, toSearchFor)	Returns the number of times searchString appears in textString.
Number	stringPosition(textString, toSearchFor, i_start, i_occurrence)	Returns the position of the string to search for, from a certain start position and occurrence.
String	stringReplace(text, search_text, replacement_text)	Replaces a portion of a string with replacement text.
String	stringReplaceTags(text, scriptable)	Returns the text with %%tags%% replaced, based on provided record or foundset or form.
String	stringRight(text, i_size)	Returns a string with the requested number of characters, starting from the right.
String	stringRightWords(text, numberof_words)	Returns the number of words, starting from the right.
String	stringToBase64(string:)	Return the Base64 representation of the string
Array	stringToBytes(string:)	Return the byte array representation of the string
String	stringToHex(string)	
Number	stringToNumber(textString)	Filters characters out of from a string and leaves digits, returns the number.

Number	stringToNumber(textString, decimalSeparator)	Filters characters out of from a string and leaves digits, returns the number.
String	stringTrim(textString)	Returns the string without leading or trailing spaces.
Number	stringWordCount(text)	Returns the number of words in the text string.
Date	timestampToDate(date)	Returns a datestamp from the timestamp (sets hours,minutes,seconds and milliseconds to 0).
Boolean	validatePBKDF2Hash(password, hash)	Validates the given password against the given hash.

Methods Details

base64ToBytes(base64String:)

```
Return the byte array representation of the Base64 value
```

Parameters

Object base64String: the Base64 encoded string to convert to byte array

Returns

Array byteArray representation of the base64 string using UTF-8 charset for conversion

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var string = utils.base64ToBytes(base64String);
```

base64ToString(base64String:)

```
Return the String representation of the Base64 value
```

Parameters

Object base64String: the Base64 value to convert to String

Returns

String String decoded representation of the Base64 value using UTF-8 charset for conversion

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var string = utils.base64ToString(base64Value);
```

bytesToBase64(byteArray:)

```
Return the Base64 value representation of the byteArray
```

Parameters

Object byteArray: the byte array to convert to Base64 value

Returns

String Base64 representation of the byte array using UTF-8 charset for conversion

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var string = utils.bytesToBase64(byteArray);
x`
```

bytesToHex(bytearray)

Parameters

Array bytearray the byte array to convert to hex encoded string

Returns

String returns hex encoded string from bytearray

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var string = utils.bytesToHex(byteArray);
```

bytesToString(byteArray:)

Return the string conversion of the byteArray

Parameters

Object byteArray: the byte array to convert to

Returns

String string representation of the byte array using UTF-8 charset for conversion

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var string = utils.bytesToString(byteArray);
```

dateFormat(date, format)

Format a date object to a text representation.
This will format with the system timezone for the webclient
For NGClient it will use the timezone of the client, the same goes for the Smartclient (but that is the system timezone)
see #dateFormat(Date,String,String) for using the actual clients timezone.

Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.yyyy'.
Symbols meaning is:

G	era designator
y	year
Y	week year
M	month in year
d	day in month
h	hour in am/pm (1~12)
H	hour in day (0~23)
m	minute in hour
s	second in minute
S	millisecond
E	day in week
D	day in year
F	day of week in month
w	week in year
W	week in month
a	am/pm marker
z	time zone
k	hour in day (1~24)
K	hour in am/pm (0~11)

Parameters

Date date the date
String format the format to output

Returns

String the date as text

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var formattedDateString = utils.dateFormat(dateobject,'EEE, d MMM yyyy HH:mm:ss');
```

dateFormat(date, format, timezone)

Format a date object to a text representation using the format and timezone given.
If the timezone is not given the timezone of the client itself will be used.
see i18n.getAvailableTimeZoneIDs() to get a timezone string that can be used.

Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.yyyy'
Symbols meaning is:

G	era designator
y	year
Y	week year
M	month in year
d	day in month
h	hour in am/pm (1~12)
H	hour in day (0~23)
m	minute in hour
s	second in minute
S	millisecond
E	day in week
D	day in year
F	day of week in month
w	week in year
W	week in month
a	am/pm marker
z	time zone
k	hour in day (1~24)
K	hour in am/pm (0~11)

Parameters

Date	date	the date
String	format	the format to output
String	timezone	The timezone string to use to parse the date (like GMT+3), if null then the timezone of the current client is used.

Returns

String the date as text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var formattedDateString = utils.dateFormat(dateobject,'EEE, d MMM yyyy HH:mm:ss', "UTC");
```

dateFormat(date, format, language, country)

Format a date object to a text representation.
This will format with the system timezone for the webclient
With language and/or country the locale will be created.
For NGClient it will use the timezone of the client, the same goes for the Smartclient (but that is the system timezone)
see #dateFormat(Date,String,String) for using the actual clients timezone.

Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.yyyy'.
Symbols meaning is:

G	era designator
y	year
Y	week year
M	month in year
d	day in month
h	hour in am/pm (1~12)
H	hour in day (0~23)
m	minute in hour
s	second in minute
S	millisecond
E	day in week
D	day in year
F	day of week in month
w	week in year
W	week in month
a	am/pm marker
z	time zone
k	hour in day (1~24)
K	hour in am/pm (0~11)

Parameters

Date	date	the date
String	format	the format to output
String	language	language used to create locale
String	country	country used along side language to create the locale

Returns

String the date as text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var formattedDateString = utils.dateFormat(dateobject,'EEE, d MMM yyyy HH:mm:ss', "fr", "CA");
```

dateFormat(date, format, timezone, language, country)

```
Format a date object to a text representation using the format, timezone, language and country given.
With the language and country given, the locale will be created.
If the timezone is not given the timezone of the client itself will be used.
see i18n.getAvailableTimeZoneIDs() to get a timezone string that can be used.

Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.
yyyy'
Symbols meaning is:
 G        era designator
 y        year
 Y        week year
 M        month in year
 d        day in month
 h        hour in am/pm (1~12)
 H        hour in day (0~23)
 m        minute in hour
 s        second in minute
 S        millisecond
 E        day in week
 D        day in year
 F        day of week in month
 w        week in year
 W        week in month
 a        am/pm marker
 z        time zone
 k        hour in day (1~24)
 K        hour in am/pm (0~11)
```

Parameters

Date date the date
String format the format to output
String timezone the timezone to use the format, if null then current client timezone is used.
String language language used to create locale
String country country used along side language to create the locale

Returns

String the date as text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var formattedDateString = utils.dateFormat(dateobject,'EEE, d MMM yyyy HH:mm:ss', "UTC", "fr", "CA");
```

getUnicodeCharacter(unicodeCharacterNumber)

```
Returns a string containing the character for the unicode number.
```

Parameters

Number unicodeCharacterNumber the number indicating the unicode character

Returns

String a string containing the unicode character

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns a big dot
var dot = utils.getUnicodeCharacter(9679);
```

hasRecords(foundset)

Returns true if the (related)foundset exists and has records.
Another use is, to pass a record and qualified relations string to test multiple relations/foundset at once

Parameters

JSFoundSet foundset the foundset to be tested

Returns

Boolean true if exists

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
//test the orders_to_orderitems foundset
if (%%elementName%%.hasRecords(orders_to_orderitems))
{
        //do work on relatedFoundSet
}
//test the orders_to_orderitems.orderitems_to_products foundset to be reached from the current record
//if (%%elementName%%.hasRecords(foundset.getSelectedRecord(),'orders_to_orderitems.orderitems_to_products'))
//{
//        //do work on deeper relatedFoundSet
//}
```

hasRecords(record, relationString)

Returns true if the (related)foundset exists and has records.
Another use is, to pass a record and qualified relations string to test multiple relations/foundset at once

Parameters

JSRecord record A JSRecord to test.
String relationString The relation name.

Returns

Boolean true if the foundset/relation has records.

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
//test the orders_to_orderitems foundset
if (%%elementName%%.hasRecords(orders_to_orderitems))
{
        //do work on relatedFoundSet
}
//test the orders_to_orderitems.orderitems_to_products foundset to be reached from the current record
//if (%%elementName%%.hasRecords(foundset.getSelectedRecord(),'orders_to_orderitems.orderitems_to_products'))
//{
//        //do work on deeper relatedFoundSet
//}
```

hexToBytes(hex)

Parameters

String hex hex encoded string to be decoded into a byte array.

Returns

Array a byte array from hex encoded string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var array = utils.hexToBytes(hex);
```

hexToString(hex)

Parameters

String hex ;

Returns

String returns decoded string from hex

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var string = utils.hexToString(hex);
```

isMondayFirstDayOfWeek()

Returns true when Monday is the first day of the week for your current locale setting.

Returns

Boolean true if Monday is first day of the week in current locale

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
if(utils.isMondayFirstDayOfWeek())
{
        //a date calculation
}
```

numberFormat(number, digits)

Format a number to have a defined fraction.

Parameters

Number number the number to format
Number digits nr of digits

Returns

String the resulting number in text

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var textalNumber = utils.numberFormat(16.749, 2); //returns 16.75
```

numberFormat(number, digits, language, country)

Format a number to have a defined fraction.

Parameters

Number number the number to format
Number digits nr of digits
String language language used to create locale
String country country used along side language to create the locale

Returns

String the resulting number in text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var textalNumber = utils.numberFormat(16.749, 2, "fr", "CA"); //returns 16.75
```

numberFormat(number, format)

Format a number to specification.

Parameters

Number number the number to format
String format the format

Returns

String the resulting number in text

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var textalNumber2 = utils.numberFormat(100006.749, '#,###.00'); //returns 100,006.75
```

numberFormat(number, format, language, country)

Format a number to specification.

Parameters

Number	number	the number to format
String	format	the format
String	language	language used to create locale
String	country	country used along side language to create the locale

Returns

String the resulting number in text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var textalNumber2 = utils.numberFormat(100006.749, '#,###.00', "fr", "CA"); //returns 100,006.75
```

parseDate(date, format)

Parse a string to a date object. This parses the date using the TimeZone of the server
Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.yyyy'

Parameters

String	date	the date as text
String	format	the format to parse the date

Returns

Date the date as date object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var parsedDate = utils.parseDate(datestring,'EEE, d MMM yyyy HH:mm:ss');
```

parseDate(date, format, timezone)

Parse a string to a date object. Using the timezone that is given, if null then it formats it with the clients timezone.

see i18n.getAvailableTimeZoneIDs() to get a timezone string that can be used.

Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.yyyy'

Parameters

String	date	the date as text
String	format	the format to parse the date
String	timezone	The timezone string to use to parse the date (like GMT+3)

Returns

Date the date as date object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var parsedDate = utils.parseDate(datestring,'EEE, d MMM yyyy HH:mm:ss');
```

parseDate(date, format, language, country)

Parse a string to a date object. Using language and country that are given,
if null then it formats it with the locale of the client

Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.yyyy'

Parameters

String date the date as text
String format the format to parse the date
String language language used to create locale
String country country used along side language to create the locale

Returns

Date the date as date object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var parsedDate = utils.parseDate(datestring,'EEE, d MMM yyyy HH:mm:ss', 'en', 'US');
```

parseDate(date, format, timezone, language, country)

Parse a string to a date object. Using the timezone, language and country that are given,
if null then it formats it with the timezone and locale of the client

see i18n.getAvailableTimeZoneIDs() to get a timezone string that can be used.

Format can be a string like: 'dd-MM-yyyy' , 'dd-MM-yyyy HH:mm' , 'MM/dd/yyyy', 'MM/dd/yyyy hh:mm aa', 'dd.MM.yyyy'

Parameters

String date the date as text
String format the format to parse the date
String timezone The timezone string to use to parse the date (like GMT+3)
String language language used to create locale
String country country used along side language to create the locale

Returns

Date the date as date object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var parsedDate = utils.parseDate(datestring,'EEE, d MMM yyyy HH:mm:ss', 'GMT+3', 'en', 'US');
```

stringEscapeMarkup(textString)

Returns the escaped markup text (HTML/XML).

Parameters

String textString the text to process

Returns

String the escaped text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var escapedText = utils.stringEscapeMarkup('<html><body>escape me</body></html>')
```

stringEscapeMarkup(textString, escapeSpaces)

Returns the escaped markup text (HTML/XML).

Parameters

String	textString	the text to process
Boolean	escapeSpaces	indicating to escape spaces

Returns

String the escaped text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var escapedText = utils.stringEscapeMarkup('<html><body>escape me</body></html>')
```

stringEscapeMarkup(textString, escapeSpaces, convertToHtmlUnicodeEscapes)

Returns the escaped markup text (HTML/XML).

Parameters

String	textString	the text to process
Boolean	escapeSpaces	indicating to escape spaces
Boolean	convertToHtmlUnicodeEscapes	indicating to use unicode escapes

Returns

String the escaped text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var escapedText = utils.stringEscapeMarkup('<html><body>escape me</body></html>')
```

stringFormat(text_to_format, parameters)

Formats a string according to format specifiers and arguments.

Parameters

String	text_to_format	the text to format
Array	parameters	the array with parameters

Returns

String the formatted text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// the  format specifier has the syntax: %[argument_index$][flags][width][.precision]conversion
// argument index is 1$, 2$ ...
// flags is a set of characters that modify the output format
// typical values: '+'(The result will always include a sign), ','(The result will include locale-specific
grouping separators)
// width is a non-negative decimal integer indicating the minimum number of characters to be written to the
output
// precision is a non-negative decimal integer usually used to restrict the number of characters
// conversion is a character indicating how the argument should be formatted
// typical conversion values: b(boolean), s(string), c(character), d(decimal integer), f(floating number), t
(prefix for date and time)
// Date/Time Conversions (used after 't' prefix):
                // 'H'         Hour of the day for the 24-hour clock, formatted as two digits with a leading
zero as necessary i.e. 00 - 23.
                // 'I'         Hour for the 12-hour clock, formatted as two digits with a leading zero as
necessary, i.e. 01 - 12.
                // 'k'         Hour of the day for the 24-hour clock, i.e. 0 - 23.
                // 'l'         Hour for the 12-hour clock, i.e. 1 - 12.
                // 'M'         Minute within the hour formatted as two digits with a leading zero as necessary,
i.e. 00 - 59.
                // 'S'         Seconds within the minute, formatted as two digits with a leading zero as
necessary, i.e. 00 - 60 ("60" is a special value required to support leap seconds).
                // 'L'         Millisecond within the second formatted as three digits with leading zeros as
necessary, i.e. 000 - 999.
                // 'p'         Locale-specific morning or afternoon marker in lower case, e.g."am" or "pm". Use
of the conversion prefix 'T' forces this output to upper case.
                // 'z'         RFC 822 style numeric time zone offset from GMT, e.g. -0800.
                // 'Z'         A string representing the abbreviation for the time zone.
                // 'B'         Locale-specific full month name, e.g. "January", "February".
                // 'b'         Locale-specific abbreviated month name, e.g. "Jan", "Feb".
                // 'h'         Same as 'b'.
                // 'A'         Locale-specific full name of the day of the week, e.g. "Sunday", "Monday"
                // 'a'         Locale-specific short name of the day of the week, e.g. "Sun", "Mon"
                // 'C'         Four-digit year divided by 100, formatted as two digits with leading zero as
necessary, i.e. 00 - 99
                // 'Y'         Year, formatted as at least four digits with leading zeros as necessary, e.g.
0092 equals 92 CE for the Gregorian calendar.
                // 'y'         Last two digits of the year, formatted with leading zeros as necessary, i.e. 00 -
99.
                // 'j'         Day of year, formatted as three digits with leading zeros as necessary, e.g. 001
- 366 for the Gregorian calendar.
                // 'm'         Month, formatted as two digits with leading zeros as necessary, i.e. 01 - 13.
                // 'd'         Day of month, formatted as two digits with leading zeros as necessary, i.e. 01 -
31
                // 'e'         Day of month, formatted as two digits, i.e. 1 - 31.

                // common compositions for date/time conversion
                // 'R'         Time formatted for the 24-hour clock as "%tH:%tM"
                // 'T'         Time formatted for the 24-hour clock as "%tH:%tM:%tS".
                // 'r'         Time formatted for the 12-hour clock as "%tI:%tM:%tS %Tp". The location of the
morning or afternoon marker ('%Tp') may be locale-dependent.
                // 'D'         Date formatted as "%tm/%td/%ty".
                // 'F'         ISO 8601 complete date formatted as "%tY-%tm-%td".
                // 'c'         Date and time formatted as "%ta %tb %td %tT %tZ %tY", e.g. "Sun Jul 20 16:17:00
EDT 1969".

utils.stringFormat('%s Birthday: %2$tm %2$te,%2$tY',new Array('My',new Date(2009,0,1))) // returns My Birthday:
01 1,2009
utils.stringFormat('The time is: %1$tH:%1$tM:%1$tS',new Array(new Date(2009,0,1,12,0,0))) // returns The time
is: 12:00:00
utils.stringFormat('My %s: %2$.0f, my float: %2$.2f',new Array('integer',10)) // returns My integer: 10, my
float: 10.00
utils.stringFormat('Today is: %1$tc',new Array(new Date())) // returns current date/time as:  Today is: Fri Feb
20 14:15:54 EET 2009
utils.stringFormat('Today is: %tF',new Array(new Date())) // returns current date as: Today is: 2009-02-20
```

stringFormat(text, format)

Format a string using mask.

Parameters

String text the string to format
Object format the format

Returns

String the resulting text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var formattedString = utils.stringFormat('0771231231', '(###)####-###'); //returns (077)1231-231
```

stringIndexReplace(text, i_start, i_size, replacement_text)

Replaces a portion of a string with replacement text from a specified index.

Parameters

String text the text to process
Number i_start the start index to work from
Number i_size the size of the text to replace
String replacement_text the replacement text

Returns

String the changed text string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'this was a test'
var retval = utils.stringIndexReplace('this is a test',6,2,'was');
```

stringInitCap(text)

Returns all words starting with capital chars.

Parameters

String text the text to process

Returns

String the changed text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'This Is A Test'
var retval = utils.stringInitCap('This is A test');
```

stringLeft(text, i_size)

Returns a string with the requested number of characters, starting from the left.

Parameters

String text the text to process
Number i_size the size of the text to return

Returns

String the result text string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'this i'
var retval = utils.stringLeft('this is a test',6);
```

stringLeftWords(text, numberof_words)

Returns the number of words, starting from the left.

Parameters

String text to process
Number numberof_words to return

Returns

String the string with number of words form the left

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'this is a'
var retval = utils.stringLeftWords('this is a test',3);
```

stringMD5HashBase16(textString)

Returns the md5 hash (encoded as base16) for specified text.

NOTE: MD5 (Message-Digest Algorythm 5) is a hash function with a 128-bit hash value, for more info see:
http://en.wikipedia.org/wiki/MD5

Parameters

String textString the text to process

Returns

String the resulting hashString

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var hashed_password = utils.stringMD5HashBase16(user_password)
```

stringMD5HashBase64(textString)

Returns the md5 hash (encoded as base64) for specified text.

NOTE: MD5 (Message-Digest Algorythm 5) is a hash function with a 128-bit hash value, for more info see:
http://en.wikipedia.org/wiki/MD5

Parameters

String textString the text to process

Returns

String the resulting hashString

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var hashed_password = utils.stringMD5HashBase64(user_password)
```

stringMiddle(text, i_start, i_size)

Returns a substring from the original string.

Parameters

String text the text to process
Number i_start the start index to work from
Number i_size the size of the text to return

Returns

String the result text string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'his'
var retval = utils.stringMiddle('this is a test',2,3);
```

stringMiddleWords(text, i_start, numberof_words)

Returns a substring from the original string.

Parameters

String	text	to process
Number	i_start	start word index
Number	numberof_words	the word count to return

Returns

String the string with number of words form the left and

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'is a'
var retval = utils.stringMiddleWords('this is a test',2,2);
```

stringPBKDF2Hash(textString)

Returns the PBKDF2 hash for specified text. This method is preferred above the old MD5 hash for enhanced security.
It uses a default of 9999 iterations. The string that is returned can only be used in the utils.validatePBKDF2Hash(password,thisReturnValue)
to check if this hash is a result of that password.
This will always be false: utils.stringPBKDF2Hash("test") == utils.stringPBKDF2Hash("test"). Because for the same string in multiply calls it will not generate the same hash.
So you can only check it like this: utils.validatePBKDF2Hash("test",utils.stringPBKDF2Hash("test"))

NOTE: PBKDF2 is the key hash function for the PKCS (Public-Key Cryptography) standard, for more info see: http://en.wikipedia.org/wiki/PBKDF2

Parameters

String textString the text to process

Returns

String the resulting hashString

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var hashed_password = utils.stringPBKDF2Hash(user_password)
```

stringPBKDF2Hash(textString, iterations)

Returns the PBKDF2 hash for specified text. This method is preferred above the old MD5 hash for enhanced security.

NOTE: PBKDF2 is the key hash function for the PKCS (Public-Key Cryptography) standard, for more info see: http://en.wikipedia.org/wiki/PBKDF2

Parameters

String	textString	the text to process
Number	iterations	how many hash iterations should be done, minimum should be 1000 or higher.

Returns

String the resulting hashString

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var hashed_password = utils.stringPBKDF2Hash(user_password,5000)
```

stringPatternCount(text, toSearchFor)

Returns the number of times searchString appears in textString.

Parameters

String	text	the text to process
String	toSearchFor	the string to search for

Returns

Number the occurrenceCount that the search string is found in the text

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 2 as count
var count = utils.stringPatternCount('this is a test','is');
```

stringPosition(textString, toSearchFor, i_start, i_occurrence)

Returns the position of the string to search for, from a certain start position and occurrence.

Parameters

String	textString	the text to process
String	toSearchFor	the string to search
Number	i_start	the start index to search from
Number	i_occurrence	the occurrence

Returns

Number the position

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 4 as position
var pos = utils.stringPosition('This is a test','s',1,1)
```

stringReplace(text, search_text, replacement_text)

Replaces a portion of a string with replacement text.

Parameters

String	text	the text to process
String	search_text	the string to search
String	replacement_text	the replacement text

Returns

String the changed text string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'these are cow 1 and cow 2.'
var retval = utils.stringReplace('these are test 1 and test 2.','test','cow');
```

stringReplaceTags(text, scriptable)

Returns the text with %%tags%% replaced, based on provided record or foundset or form.

Parameters

String	text	the text tags to work with
Object	scriptable	the javascript object or foundset,record,form to be used to fill in the tags

Returns

String the text with replaced tags

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//Next line places a string in variable x, whereby the tag(%%TAG%%) is filled with the value of the database
column 'company_name' of the selected record.
var x = utils.stringReplaceTags("The companyName of the selected record is %%company_name%% ", foundset)
//var otherExample = utils.stringReplaceTags("The amount of the related order line %%amount%% ",
order_to_orderdetails);
//var recordExample = utils.stringReplaceTags("The amount of the related order line %%amount%% ",
order_to_orderdetails.getRecord(i);
//Next line places a string in variable y, whereby the tag(%%TAG%%) is filled with the value of the form
variable 'x' of the form named 'main'.
//var y = utils.stringReplaceTags("The value of form variable is %%x%% ", forms.main);
//The next sample shows the use of a javascript object
//var obj = new Object();//create a javascript object
//obj['x'] = 'test';//assign an named value
//var y = utils.stringReplaceTags("The value of object variable is %%x%% ", obj);//use the named value in a tag
```

stringRight(text, i_size)

Returns a string with the requested number of characters, starting from the right.

Parameters

String text the text to process
Number i_size the size of the text to return

Returns

String the result text string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'a test'
var retval = utils.stringLeft('this is a test',6);
```

stringRightWords(text, numberof_words)

Returns the number of words, starting from the right.

Parameters

String text to process
Number numberof_words to return

Returns

String the string with number of words form the right

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'is a test'
var retval = utils.stringRightWords('this is a test',3);
```

stringToBase64(string:)

Return the Base64 representation of the string

Parameters

Object string: the string to convert to Base64

Returns

String Base64 encoded representation of the string using UTF-8 charset for conversion

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var string = utils.stringToBase64('This is A test');
```

stringToBytes(string:)

Return the byte array representation of the string

Parameters

Object string: the string to convert to bytes

Returns

Array byte array representation of the string using UTF-8 charset for conversion

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var byteArray = utils.stringToBytes('This is A test');
```

stringToHex(string)

Parameters

String string String to be encoded into hex

Returns

String returns hex encoded string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var hex = utils.stringToHex(string);
```

stringToNumber(textString)

Filters characters out of from a string and leaves digits, returns the number. Uses locale decimal separator.

Parameters

String textString the text to process

Returns

Number the resulting number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 65567
var retval = utils.stringToNumber('fg65gf567');
```

stringToNumber(textString, decimalSeparator)

Filters characters out of from a string and leaves digits, returns the number. Decimal separator is specified as parameter.

Parameters

String textString the text to process
String decimalSeparator decimal separator

Returns

Number the resulting number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 65.567
var retval = utils.stringToNumber('fg65gf.567','.');
```

stringTrim(textString)

Returns the string without leading or trailing spaces.

Parameters

String textString the text to process

Returns

String the resulting trimmed string

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns 'text'
var retval = utils.stringTrim('   text   ');
```

stringWordCount(text)

Returns the number of words in the text string.

Parameters

String text the text to process

Returns

Number the word count

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//returns '4' as result
var retval = utils.stringWordCount('this is a test');
```

timestampToDate(date)

Returns a datestamp from the timestamp (sets hours,minutes,seconds and milliseconds to 0).

Parameters

Date date object to be stripped from its time elements

Returns

Date the stripped date object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var date = utils.timestampToDate(application.getTimeStamp());
```

validatePBKDF2Hash(password, hash)

Validates the given password against the given hash. The hash should be generated by one of the stringPBKDF2Hash (password [,iteration]) functions. If hash is null or empty string the method will return false.

NOTE: PBKDF2 is the key hash function for the PKCS (Public-Key Cryptography) standard, for more info see: http://en.wikipedia.org/wiki/PBKDF2

Parameters

String password the password to test against
String hash the hash the password needs to validate to.

Returns

Boolean true if his hash is valid for that password

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
if (utils.validatePBKDF2Hash(user_password, hashFromDb)) {
   // logged in
}
```