netgenium

Javascript functions

Framework NET Genium

1 Variables

- nID – ID of the currently open record in the edit form (it is not recommended to use #id#)
- Page_IsPostBack – The page is loaded after a postback
- dateSeparator – Date separator
- domLib_isIE – The current browser is Internet Explorer
- domLib_isIE56 – The current browser is Internet Explorer version 5 or 6
- domLib_isIE7 – The current browser is Internet Explorer version 7
- domLib_isIE8 – The current browser is Internet Explorer version 8
- domLib_isIE9 – The current browser is Internet Explorer version 9
- domLib_isIE56789 – The current browser is Internet Explorer version 5, 6, 7, 8 or 9
- domLib_isIE10 – The current browser is Internet Explorer version 10
- domLib_isIE11 – The current browser is Internet Explorer version 11
- domLib_isChrome – The current browser is Google Chrome, Microsoft Edge or Opera version 15 and above
- domLib_isFirefox – The current browser is Mozilla Firefox
- domLib_isSafari – The current browser is Safari
- domLib_isOpera – The current browser is Opera version 12.15 or lower
- domLib_isMobileDevice – The current browser is a mobile device
- domLib_isCE – The current Windows CE device browser

2 Function

void _loadUrl(object o, string url)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. The "o" parameter is an auxiliary object in which all the information needed to successfully complete an asynchronous ajax call is stored, and this object can be anything in the object model of the website or a newly created object. Only one asynchronous call can be called at a time using one object. If the "_loadUrl" function is called again with the same helper object for which the asynchronous call has not yet been completed, a new asynchronous call will not occur. If a parallel call to the "_loadUrl" function is required, it is ideal to select "new Object()" as the auxiliary object "o".*

- ***Example:***

```
var p0 = 'ěščřžýáié', p1 = 'ĚŠČŘŽÝÁÍÉ';
_loadUrl(new Object(), 'ngef.aspx?MyFirstFunction,' + urlEncode(p0) + ',' + urlEncode(p1));
```

void _loadUrl(object o, string url, string f)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. When the answer then arrives, the execution of the main context of the code is suspended (sooner or later depending on the priority of the performed task) and the so-called callback defined by the "f" parameter. The resulting HTML code of the page is passed to the "f" function as the first parameter. The "o" parameter is an auxiliary object in which all the information needed to successfully complete an asynchronous ajax call is stored, and this object can be anything in the object model of the website or a newly created object. Only one asynchronous call can be called at a time using one object. If the "_loadUrl" function is called again with the same helper object for which the asynchronous call has not yet been completed, a new asynchronous call will not occur. If a parallel call to the "_loadUrl" function is required, it is ideal to select "new Object()" as the auxiliary object "o".*

- ***Example:***

```
function ajaxTest()
{
  var p0 = 'ěščřžýáié', p1 = 'ĚŠČŘŽÝÁÍÉ';
  _loadUrl(new Object(), 'ngef.aspx?MyFirstFunction,' + urlEncode(p0) + ',' + urlEncode(p1),
ajaxResponse);
}

function ajaxResponse(html)
{
  alert(html);
}

for (var i = 0; i < 5; i++)
{
```

```
  ajaxTest();
}
```

void _loadUrl(object o, string url, string f, string id)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. When the answer then arrives, the execution of the main context of the code is suspended (sooner or later depending on the priority of the performed task) and the so-called callback defined by the "f" parameter. The resulting HTML code of the page is passed to the "f" function as the first parameter. The optional parameter "id" is passed to the "f" function as the second parameter. The "o" parameter is an auxiliary object in which all the information needed to successfully complete an asynchronous ajax call is stored, and this object can be anything in the object model of the website or a newly created object. Only one asynchronous call can be called at a time using one object. If the "_loadUrl" function is called again with the same helper object for which the asynchronous call has not yet been completed, a new asynchronous call will not occur. If a parallel call to the "_loadUrl" function is required, it is ideal to select "new Object()" as the auxiliary object "o".*

- ***Example:***

```
function ajaxTest(id)
{
  var p0 = 'ěščřžýáié', p1 = 'ĚŠČŘŽÝÁÍÉ';
  _loadUrl(new Object(), 'ngef.aspx?MyFirstFunction,' + urlEncode(p0) + ',' + urlEncode(p1),
ajaxResponse, id);
}

function ajaxResponse(html, id)
{
  alert(id + ': ' + html);
}

for (var i = 0; i < 5; i++)
{
  ajaxTest(i);
}
```

void _loadUrl(object o, string url, string f, string id, string params)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. When the answer then arrives, the execution of the main context of the code is suspended (sooner or later depending on the priority of the performed task) and the so-called callback defined by the "f" parameter. The resulting HTML code of the page is passed to the "f" function as the first parameter. The optional parameter "id" is passed to the "f" function as the second parameter. The parameter "params" is sent together with the asynchronous call by the POST method, and it is a list of key vs. pair pairs. value separated by a "&" character. The "o" parameter is an auxiliary object in which all the information needed to successfully complete an asynchronous ajax call is stored, and this object can be anything in the object model of the website or a newly created object. Only one asynchronous call can be called at a time using one object. If the "_loadUrl" function is called again with the same helper object for which the asynchronous call has not yet been completed, a new asynchronous call will not occur. If a parallel call to the "_loadUrl" function is required, it is ideal to select "new Object()" as the auxiliary object "o".*

- ***Example:***

```
function ajaxTest(id)
{
  var p0 = 'ěščřžýáíé', p1 = 'ĚŠČŘŽÝÁÍÉ';
  _loadUrl(new Object(), 'ngef.aspx?MyFirstFunction', ajaxResponse, id, 'p0=' + urlEncode(p0)
+ '&p1=' + urlEncode(p1));
}

function ajaxResponse(html, id)
{
  alert(id + ': ' + html);
}

for (var i = 0; i < 5; i++)
{
  ajaxTest(i);
}
```

void attachEvent2(object o, string type, function listener)

- ***Description:*** *The function appends the "listener" function to the "o" object event. "Type" defines the type of event, eg load, scroll, etc.*

- ***Example:***

```
attachEvent2(window, 'load', function () { alert('onload1'); });
attachEvent2(window, 'load', test);

function test()
{
 alert('onload2');
}
```

void bt_Click()

Description: *The function invokes the form reload (postback). During this event, the data of all tables, calendars, graphs, etc. will be reloaded. This function has been replaced by the "form_Update" function.*

Example:

```
bt_Click();
```

void bt_Click(string id)

Description: *The function triggers a program click on the button specified by the "id" parameter. The function first tries to find an element with the given ID, and if it can't find it, with the given text.*

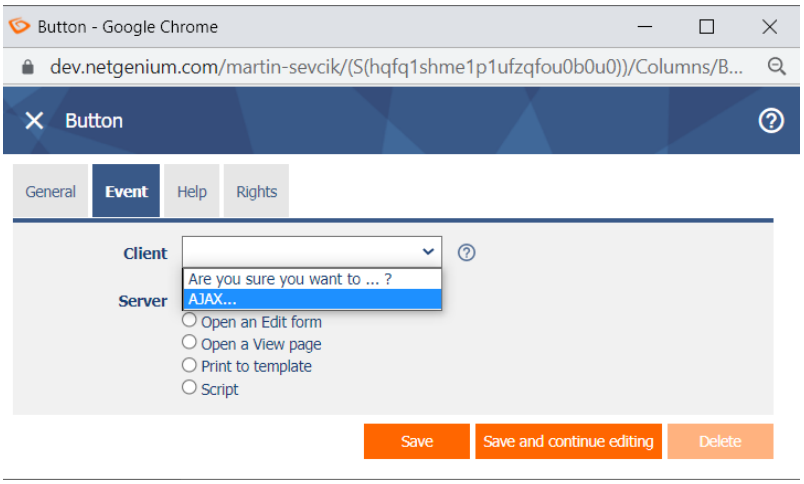
Example:

```
bt_Click(BT123);
bt_Click('Save');
```

bool bt_Eval(object bt, string url, string f)

Description: *When the button is pressed, the function loads the page (typically an external function) defined by the "url" parameter, and uses the validating function defined by the "f" parameter to verify the loaded HTML code. The validating function must return a logical value of "true" or "false", which determines whether to continue processing the "onclick" event. The behavior of the "bt_Eval" function can be defined when specifying the JavaScript of the button (see figure). And also in the javascript of the relevant view page or edit form.*

Example:



***Example** – 1st variant:*

```
return bt_Eval(this, 'ngef.aspx?test', 'evalFunction');
```

***Example** – 2nd variant:*

```
return bt_Eval(this, 'ngef.aspx?test', ajaxResponse);

function ajaxResponse(html)
{
  return confirm('Response: ' + html);
}
```

***Example** – 3rd variant:*

```
return test(this);

function test(bt)
{
  if (!bt_Eval(bt, 'ngef.aspx?test', ajaxResponse)) return false;
  return confirm('Continue?');
}

function ajaxResponse(html)
{
  return confirm('Response: ' + html);
}
```

object bt_FindByText(string text, int index)

Description: *The function returns a button with the title specified by the "text" parameter. The "index" parameter is optional and tells you how many times the button with the title "index" should return the function. This parameter is important in edit forms where multiple buttons with the same title are located. Numbering is from zero, so the first button has an index of 0.*

Example:

```
var bt = bt_FindByText('Today', 2); // Returns a button object that is named "Today" and is
third in the form with that name
```

bool bt_Icon(object bt, string url)

Description: *The function sets the image defined by the "url" parameter in the background of the button.*

Example:

```
bt_Icon(el(BT123), 'Images/MD/Content/ic_add_white_18dp.png');
```

void button_Disable(o)

Description: *The function disables the button specified by the "o" parameter in the edit form.*

Example:

```
button_Disable(BT123);
```

void buttons_Disable()

Description: *The function disables all buttons in the edit form.*

Example:

```
buttons_Disable();
```

void button_Enable(o)

Description: *The function enables the button specified by the "o" parameter in the edit form.*

Example:

```
button_Enable(BT123);
```

void buttons_Enable()

Description: *The function enables all buttons in the edit form.*

Example:

```
buttons_Enable();
```

void cb_Disable(object cb)

- ***Description:*** *The function disables the editing of the "ComboBox" control specified by the "cb" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
cb_Disable(#ng_combobox#);
```

void cb_Enable(object cb)

- ***Description:*** *The function enables editing of the "ComboBox" control specified by the "cb" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
cb_Enable(#ng_combobox#);
```

string Color2Hex(string value)

- ***Description:*** *The function formats the color specified by the "value" parameter into hexadecimal from the input format "R,G,B".*
- ***Example:***

```
var s = Color2Hex(control_GetValue(#ng_color#));
```

void control_Disable(object o)

- ***Description:*** *The function disables editing of the control defined by the "o" parameter.*
- ***Example:***

```
control_Disable(#ng_control#);
```

void control_Disable(object o, bool disableLink)

- ***Description:*** *The function disables editing of the control defined by the "o" parameter. The optional "disableLink" parameter (true/false) determines whether the reference to the selected record is to be deactivated for the "ForeignKey" control.*

- ***Example:***

```
control_Disable(#ng_control#);
control_Disable(#ng_control#, false);
```

void control_Enable(object o)

- ***Description:*** *The function enables editing of the control defined by the "o" parameter.*

- ***Example:***

```
control_Enable(#ng_control#);
```

string control_GetValue(object o)

- ***Description:*** *The function returns the value of the database control specified by the "o" parameter. The value of the control can also be found simply using the notation "#ng_control#.value", but this notation cannot be used for the following controls:*

    - TextBox of type Date with time on – the time is placed in a different text field than the date itself, so "#ng_textbox#.value" returns only a date folder without time, the only possible way to return the date and time is "control_GetValue(#ng_textbox#)"
    - MultiListBox – the only possible way to return a tab-delimited list of selected values is "control_GetValue(#ng_multilistbox#)"
    - Radio – the only possible way to return the selected value is "control_GetValue(#ng_radio#)"
    - CheckBox – next to the function "control_GetValue(#ng_checkbox#)", which returns a text value stored in the database if the button is checked, or an empty text string if the button is not checked, it is possible to use the logical value "#ng_checkbox#.checked", which returns a check/do not check the button

- ***Example:***

```
var s = control_GetValue(#ng_control#);
```

void control_ResizeTable(object o, int width)

- ***Description:*** *The function sets the width of the table in which the control specified by the "o" parameter is displayed to the new width specified by the "width" parameter.*

Example:

```
control_ResizeTable(#ng_control:Table#, 60);
```

void control_Save(object o)

- ***Description:*** *The function stores the value of the database control in the database using ajax. "OnBeforeSave" and "OnAfterSave" events are not executed when saved. The "o" parameter specifies the database control whose value will be stored in the database.*

Example:

```
control_Save(#ng_control#);
```

void control_Save(object o, string f, string errorMessage)

- ***Description:*** *The function stores the value of the database control in the database using ajax. "OnBeforeSave" and "OnAfterSave" events are not executed when saved. The "o" parameter specifies the database control whose value will be stored in the database. The "f" parameter specifies the name of the function to run after the response is retrieved from the server. This response is either "OK" (record saved) or "ERROR" (record failed). The "errorMessage" parameter defines the text of the error message that is displayed if an error occurs during saving.*

Example:

```
control_Save(#ng_control#, ajaxResponse, 'An error has occured');

function ajaxResponse(status)
{
  alert(status);
}
```

void control_SetValue(object o, var value [, var value2])

- ***Description:*** *The function inserts the value defined by the "value" parameter into the database control specified by the "o" parameter.*
    - If it is a "TextBox" with "Date" or "DatePicker" validation, it is possible to use the "value" parameter as a text string formatted in the form "dd.MM.yyyy HH:mm", or a javascript object of the "Date" type. The

"value2" parameter is optional and specifies the time in the format "HH:mm". The parameter is only relevant for "TextBox" with "Date" validation and selected time entry.

- If the "value" parameter is a javascript object of the "Date" type, it is not necessary to specify the "value2" parameter, NET Genium will automatically read the time data from the "value" parameter.
- If it is a "CheckBox", it is possible to use two types of "value" parameters:
  - If the "value" parameter is a logical value, it is said whether to check the check box or not.
  - If the "value" parameter is a text string, its value will first be compared with the value set in the control's properties, and depending on the result of this test, the check box will be checked or not.

Example:

```
control_SetValue(#ng_control#, 'Text');
control_SetValue(#ng_textbox#, 20);
control_SetValue(#ng_textbox#, '1.1.2000');
control_SetValue(#ng_textbox#, '1.1.2000', '12:34');
control_SetValue(#ng_textbox#, new Date());
control_SetValue(#ng_datepicker#, '1.1.2000');
control_SetValue(#ng_datepicker#, '1.1.2000 12:34');
control_SetValue(#ng_datepicker#, new Date());
control_SetValue(#ng_checkbox#, true);
control_SetValue(#ng_checkbox#, 'x');
control_SetValue(#ng_foreignkey#, 20);
```

void controls_Disable()

Description: *The function disables editing of all controls (except buttons) in the edit form.*

Example:

```
controls_Disable();
```

void controls_Disable(bool disableLinks)

Description: *The function disables editing of all controls (except buttons) in the edit form. The optional "disableLinks" parameter (true/false) determines whether the reference to the selected record is to be deactivated for the "ForeignKey" controls.*

Example:

```
controls_Disable();
controls_Disable(false);
```

void controls_Enable()

- ***Description:*** *The function allows editing of all controls (except buttons) in the edit form.*
- ***Example:***

```
controls_Enable();
```

void controls_Join(object[] array)

- ***Description:*** *The function combines the controls defined by the "array" parameter into one group, which displays only the controls with the filled value, plus one more.*
- ***Example:***

```
controls_Join([#ng_control1#, #ng_control2#, #ng_control3#]);
controls_Join(new Array(#ng_control1#, #ng_control2#, #ng_control3#));
```

void controls_Save(object[] array)

- ***Description:*** *The function stores the values of database controls in a database using ajax. "OnBeforeSave" and "OnAfterSave" events are not executed when saved. The "array" parameter specifies a list of database controls whose values will be stored in the database.*
- ***Example:***

```
controls_Save([#ng_control1#, #ng_control2#]);
controls_Save(new Array(#ng_control1#, #ng_control2#));
```

void controls_Save(object[] array, string f, string errorMessage)

- ***Description:*** *The function stores the values of database controls in a database using ajax. "OnBeforeSave" and "OnAfterSave" events are not executed when saved. The "array" parameter specifies a list of database controls whose values will be stored in the database. The "f" parameter specifies the name of the function to run after the response is retrieved from the server. This response is either "OK" (record saved) or "ERROR" (record failed). The "errorMessage" parameter defines the text of the error message that is displayed if an error occurs during saving.*

- ***Example:***

```
controls_Save([#ng_control1#, #ng_control2#], ajaxResponse, 'An error has occured');

function ajaxResponse(status)
{
  alert(status);
}
```

string cookie_Get(string name)

- ***Description:*** *The function returns the content of the cookie specified by the "name" parameter.*

- ***Example:***

```
var s = cookie_Get('loginname'); // Returns the value of a cookie named "loginname"
```

void cookie_Set(string name, string value)

- ***Description:*** *The function sets the content of the cookie specified by the "name" parameter to the value specified by the "value" parameter.*

- ***Example:***

```
cookie_Set('TestCookie', 'Test cookie');
```

bool copyToClipboard(string value, string info)

- ***Description:*** *The function copies the value specified by the "value" parameter to the clipboard. The "info" parameter defines the text of the information message, which will be displayed immediately after the successful insertion of the value into the clipboard. If the insert is not successful, the function returns "false".*

- ***Example:***

```
copyToClipboard('Value to be pasted into clipboard', 'Value successfully pasted into
clipboard.');
```

string Date2Str(object date)

- ***Description:*** *The function returns a text string in the format "dd.MM.yyyy HH:mm" from the date specified by the "date" parameter.*
  If the time is 00:00, the function returns only a string in the format "dd.MM.yyyy".

- ***Example:***

```
var s = Date2Str(new Date());
```

void ddl_ReplaceItems(object ddl, object array)

- ***Description:*** *The function replaces the items of the dropdown list specified by the "ddl" parameter with the values defined in the "array" field. The field must contain a value that is stored in the database and a placeholder text to be displayed in the dropdown instead of the value.*

- ***Example*** *(requires a ComboBox control with items 1, 2 and 3):*

```
var array = [];
array.push(['1', 'First item']);
array.push(['2', 'Second item']);
array.push(['3', 'Third item']);
ddl_ReplaceItems(#ng_combobox#, array);
```

void ddl_ReplaceTexts(object ddl, object array, string separator)

- ***Description:*** *The function replaces the texts of the items in the drop-down list specified by the "ddl" parameter with the values defined in the "array" field. The field must contain the value that is stored in the database and the placeholder text that you want to appear in the drop-down list instead of the value. The last parameter "separator" is optional, and acts as a separator between the value and its placeholder text. Replacing texts with this function does not affect the value that is stored in the database.*

- ***Example*** *(requires ComboBox control with items 1, 2 and 3):*

```
var array = [];
array.push(['1', 'First item']);
array.push(['2', 'Second item']);
array.push(['3', 'Third item']);
ddl_ReplaceTexts(#ng_combobox#, array);
// ddl_ReplaceTexts(#ng_combobox#, array, ' - ');
```

void ddl_SortByText(object ddl)

- ***Description:*** *The function sorts the items in the "ddl" drop-down list alphabetically according to the texts of the individual items.*

- ***Example:***

```
ddl_SortByText(#ng_combobox#);
```

void ddl_SortByText2(object ddl)

- ***Description:*** *The function numerically sorts the items in the "ddl" drop-down list according to the texts of the individual items.*

- ***Example:***

```
ddl_SortByText2(#ng_combobox#);
```

void ddl_SortByValue(object ddl)

- ***Description:*** *The function sorts the "ddl" drop-down list items alphabetically according to the values of the individual items.*
- ***Example:***

```
ddl_SortByValue(#ng_combobox#);
```

void ddl_SortByValue2(object ddl)

- ***Description:*** *The function numerically sorts the items in the "ddl" drop-down list according to the values of the individual items.*
- ***Example:***

```
ddl_SortByValue2(#ng_combobox#);
```

string ddl_SelectedItem_Text(object ddl)

- ***Description:*** *The function returns the text of the selected item in the drop-down list specified by the "ddl" parameter.*
- ***Example:***

```
var s = ddl_SelectedItem_Text(#ng_combobox#);
```

string ddl_SelectedItem_Value(object ddl)

- ***Description:*** *The function returns the value of the selected item in the drop-down list specified by the "ddl" parameter.*
- ***Example:***

```
var s = ddl_SelectedItem_Value(#ng_combobox#);
```

bool ddl_SelectText(object ddl, string text)

- ***Description:*** *The function searches for an item in the drop-down list specified by the "ddl" parameter according to its text (the "text" parameter), and if it finds it, selects it and calls the "onchange" event. If the value is not found in the control's list of values, the function returns "false".*

- ***Example:***

```
var ok = ddl_SelectText(#ng_combobox#, 'Prague');
```

bool ddl_SelectText_NoChange(object ddl, string text)

- ***Description:*** *The function searches for an item in the drop-down list specified by the "ddl" parameter according to its text ("text" parameter) and, if it finds it, selects it. Functions from the "onchange" event are not called. If the value is not found in the control's list of values, the function returns "false".*

- ***Example:***

```
var ok = ddl_SelectText_NoChange(#ng_combobox#, 'Prague');
```

bool ddl_SelectValue(object ddl, string text)

- ***Description:*** *The function searches for an item in the drop-down list specified by the "ddl" parameter according to its value (the "text" parameter), and if it finds it, selects it and calls the "onchange" event. If the value is not found in the control's list of values, the function returns "false".*

- ***Example:***

```
var ok = ddl_SelectValue(#ng_combobox#, 'Prague');
```

bool ddl_SelectValue_NoChange(object ddl, string text)

- ***Description:*** *The function searches for an item in the drop-down list specified by the "ddl" parameter according to its value (the "text" parameter) and, if it finds it, selects it. Functions from the "onchange" event are not called. If the value is not found in the control's list of values, the function returns "false".*

- ***Example:***

```
var ok = ddl_SelectValue_NoChange(#ng_combobox#, 'Prague');
```

void detachEvent2(object o, string type, function listener)

- ***Description:*** *The function disconnects the "listener" function from the "o" object event. "Type" defines the type of event, eg load, scroll, etc.*

Example:

```
attachEvent2(window, 'load', test);
detachEvent2(window, 'load', test);

function test()
{
  alert('onload');
}
```

void dg_Controls(object dg)

- ***Description:*** *The function returns an array of all editable controls within the "dg" datagrid.*

Example:

```
var array = dg_Controls(el(DG123));
```

void dg_Disable()

- ***Description:*** *The function disables the editing of all controls intended for mass editing, which are located inside all datagrids.*

Example:

```
dg_Disable();
```

void dg_Disable(object dg)

- ***Description:*** *The function disables the editing of all controls intended for mass editing, which are located within a specific datagrid.*

Example:

```
dg_Disable(el(DG123));
```

void dg_Enable()

- ***Description:*** *The function allows the editing of all controls intended for mass editing, which are located inside all datagrids.*

Example:

```
dg_Enable();
```

void dg_Enable(object dg)

- ***Description:*** *The function allows the editing of all controls intended for mass editing, which are located within a specific datagrid.*

Example:

```
dg_Enable(el(DG123));
```

bool dg_HideColumn(object dg, int cid)

- ***Description:*** *The function hides the column specified by the "cid" parameter in the datagrid defined by the "dg" parameter. The column ID is the same as the control ID by which the column is defined (eg TextBox ID).*

Example:

```
dg_HideColumn(el(DG123), 123);
dg_HideColumn(DG123, 123);
```

void dg_HideHeader(object dg)

- ***Description:*** *The function hides the title of the datagrid defined by the "dg" parameter. The "dg" parameter can also be the datagrid title itself.*

Example:

```
dg_HideHeader(DG123);
dg_HideHeader('Table title');
```

void dg_SearchColumns(object dg, array cids)

- ***Description:*** *The function checks the search of columns defined by the "cids" parameter in the datagrid defined by the "dg" parameter. The columns to be searched are defined as an identifier field, where the column ID is the same as the control ID by which the column is defined (eg TextBox ID).*
- ***Example:***

```
dg_SearchColumns(el(DG123), [1, 2, 3]);
dg_SearchColumns(DG123, [1, 2, 3]);
```

bool dg_ShowColumn(object dg, int cid)

- ***Description:*** *The function displays the column (which was hidden) defined by the "cid" parameter in the datagrid defined by the "dg" parameter. The column ID is the same as the control ID by which the column is defined (eg TextBox ID).*
- ***Example:***

```
dg_ShowColumn(el(DG123), 123);
dg_ShowColumn(DG123, 123);
```

void dg_VisibleColumns (object dg, array cids)

- ***Description:*** *The function displays the columns defined by the "cids" parameter in the datagrid defined by the "dg" parameter. The columns to be displayed are defined as an array of identifiers, where the column ID is the same as the control ID by which the column is defined (eg TextBox ID).*
- ***Example:***

```
dg_VisibleColumns(el(DG123), [1, 2, 3]);
dg_VisibleColumns(DG123, [1, 2, 3]);
```

void dp_Disable(object dp)

- ***Description:*** *The function disables editing of the "DatePicker" control specified by the "dp" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
dp_Disable(#ng_datepicker#);
```

void dp_Enable(object dp)

- ***Description:*** *The function enables editing of the "DatePicker" control specified by the "dp" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
dp_Enable(#ng_datepicker#);
```

object el(string id)

- ***Description:*** *The function returns an object on the web page according to its ID.*
- ***Example:***

```
var bt = el(BT1665);
// Returns a button object with database ID 1665
```

void evalGoogleCaptcha()

- ***Description:*** *This feature triggers Google reCAPTCHA validation, which recognizes users from robots.*
- ***Example:***

```
evalGoogleCaptcha();
```

void file_Disable(object file)

- ***Description:*** *The function disables editing of the "File" control specified by the "file" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
file_Disable(#ng_file#);
```

void file_Enable(object file)

- ***Description:*** *The function allows editing of the "File" control specified by the "file" parameter. An alternative is the general function "control_Enable".*

Example:

```
file_Enable(#ng_file#);
```

void fireEvent2(object o, string type)

- ***Description:*** *The function raises an event on the "o" object. The "type" parameter is the type of event (eg blur, click, onchange, keyup, mouseout, mouseover…).*

Example:

```
fireEvent2(#ng_control#, 'change');
// Raises the "onchange" event on the "ng_control" element
```

void fk_Disable(object fk, bool disableLink)

- ***Description:*** *The function disables editing of the "ForeignKey" control specified by the "fk" parameter. The optional "disableLink" parameter (true/false) determines whether the link to the selected record is to be deactivated. An alternative is the general function "control_Disable".*

Example:

```
fk_Disable(#ng_foreignkey#);
fk_Disable(#ng_foreignkey#, false);
```

void fk_Enable(object fk)

- ***Description:*** *The function allows editing of the "ForeignKey" control specified by the "fk" parameter. An alternative is the general function "control_Enable".*

Example:

```
fk_Enable(#ng_foreignkey#);
```

void fk_Update(object fk)

- ***Description:*** *The function updates the value of the "ForeignKey" control specified by the "fk" parameter with ajax. The function is only relevant if the control width is set to zero and the "control_SetValue" function has been used to set a new control value.*

- ***Example:***

```
fk_Update(#ng_foreignkey#);
```

void form_DisableUpdate()

- ***Description:*** *The function prevents automatic editing of the edit form after copying file attachments or images using a pickup, or after copying values to drop-down lists or radio buttons that do not yet contain the copied value.*

- ***Example:***

```
form_DisableUpdate();
```

void form_Change()

- ***Description:*** *The function is called whenever the value of a control in the edit form changes, and ensures that when the user tries to leave the edit form, a message is displayed indicating whether he really wants to leave the form without first saving the changes.*

- ***Example n. 1:***

```
form_Change();
```

- ***Example n. 2:***

```
var saveObject = new Object();

function form_Change()
{
  setTimeout2(saveObject, function() { form_Save(); }, 1000);
}
```

bool form_Picker()

- ***Description:*** *The function returns a boolean value if the form is in the "onchange" javascript event call state of the controls whose values have just been taken over by the picker.*

Example:

```
var ok = form_Picker();
```

void form_Save()

- ***Description:*** *The function saves the currently open record in the edit form using ajax. "OnBeforeSave" and "OnAfterSave" events are not executed when saved.*

Example:

```
form_Save();
```

void form_Save(string f)

- ***Description:*** *The function saves the currently open record in the edit form using ajax. "OnBeforeSave" and "OnAfterSave" events are not executed when saved. The "f" parameter specifies the name of the function to run after the response is retrieved from the server. This response is either "OK" (record saved) or "ERROR" (record failed).*

Example:

```
form_Save(ajaxResponse);

function ajaxResponse(status)
{
  alert(status);
}
```

void form_SetMargins(int topMargin, int rightMargin, int bottomMargin, int leftMargin)

- ***Description:*** *The function sets the margins of the web page.*

Example:

```
form_SetMargins(0, 0, 0, 0);
```

void form_Update()

- ***Description:*** *The function updates the currently open edit form, mainly due to updating data in datagrids or data sources of individual controls in the edit form.*
- ***Example:***

```
form_Update();
```

string Hex(int n)

- ***Description:*** *The function converts the decimal form of the number "n" to hexadecimal.*
- ***Example:***

```
var s = Hex(123);
// Returns "7B"
```

string htmlEncode(string value)

- ***Description:*** *The function replaces all HTML characters contained in the "value" parameter with the corresponding character sequence.*
- ***Example:***

```
var s = htmlEncode('<>');
```

void chb_Disable(object chb)

- ***Description:*** *The function disables the editing of the "CheckBox" control specified by the "chb" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
chb_Disable(#ng_checkbox#);
```

void chb_Display(object chb, object o, bool reverse)

- ***Description:*** *The function hides/shows the element specified by the "o" parameter depending on whether the checkbox defined by the "chb" parameter is checked or not. The "reverse" parameter specifies "true" or "false" depending on whether the function should behave inversely. The function is mainly used in the "OnClick" event at the checkbox.*

Example:

```
chb_Display(this, 'D1');
chb_Display(this, 'D1', true);
chb_Display(this, #ng_textbox#);
chb_Display(this, #ng_textbox#, true);
chb_Display(#ng_checkbox#, 'D1');
chb_Display(#ng_checkbox#, #ng_textbox#);
```

void chb_Enable(object chb)

- ***Description:*** *The function enables editing of the "CheckBox" control specified by the "chb" parameter. An alternative is the general function "control_Enable".*

Example:

```
chb_Enable(#ng_checkbox#);
```

void image_Disable(object image)

- ***Description:*** *The function disables editing of the "Image" control specified by the "image" parameter. An alternative is the general function "control_Disable".*

Example:

```
image_Disable(#ng_image#);
```

void image_Enable(object image)

- ***Description:*** *The function allows editing of the "Image" control specified by the "image" parameter. An alternative is the general function "control_Enable".*

Example:

```
image_Enable(#ng_image#);
```

int innerHeight2()

- ***Description:*** *The function returns the inner height of the window of the window object.*
- ***Example:***

```
var n = innerHeight2();
```

int innerWidth2()

- ***Description:*** *The function returns the inner width of the window of the window object.*
- ***Example:***

```
var n = innerWidth2();
```

void InsertTime(object tb)

- ***Description:*** *The function inserts the current time in the format "HH:mm" in the text field defined by the "tb" parameter.*
- ***Example:***

```
InsertTime(#ng_textbox#);
```

string Int2Time(int minutes)

- ***Description:*** *The function returns a text string in the format "HH:mm" expressed from the number of minutes specified by the "minutes" parameter.*
- ***Example:***

```
var s = Int2Time(123);
// Returns "02:03"
var s = Int2Time(-123);
// Returns "-02:03"
```

bool isImage(string filename)

- ***Description:*** *The function returns "true" or "false" depending on the file extension that identifies the image.*
- ***Example:***

```
var ok = isImage('filename.txt'); // Returns "false"
```

string jsCompanyWithoutSro(string company)

- ***Description:*** *The function removes the suffix of the company type (eg s.r.o., a.s., k.s., v.o.s., spol., etc.) from the company name specified by the "company" parameter.*
- ***Example:***

```
var s = jsCompanyWithoutSro('Company, v.o.s.');
```

string jsDisplay(bool value)

- ***Description:*** *The function returns '' (element will be visible) or 'none' (element will not be visible) according to the content of the "value" parameter. The function is used to set the "style.display" attribute of individual controls of the edit form.*
- ***Example:***

```
#ng_textbox:Table#.style.display = jsDisplay(#ng_checkbox#.checked);
D1.style.display = jsDisplay(#ng_checkbox#.checked);
```

string jsFN(double n)
string jsFormatNumber(double number)

- ***Description:*** *The function formats the number specified by the "number" parameter into the text form, which is required for the correct storage of the numeric value of the control in the database – the function ensures the use of the correct decimal separator.*
- ***Example:***

```
control_SetValue(#ng_textbox#, jsFN(1.23));
#ng_textbox#.value = jsFN(1.23);
```

string jsFormatCurrency(double number, string symbol)

Description: *The function formats the number specified by the "number" parameter into text – with a measure as a thousands separator, with a decimal point or a dot, depending on the language setting of the currently logged in user, rounded to two decimal places, and with a symbol specified by the "symbol" parameter.*

Example:

```
var s = jsFormatCurrency(1100, 'CZK');
// Returns "1 100,00 CZK"
var s = jsFormatCurrency(1100.5, 'CZK');
// Returns "1 100,50 CZK"
var s = jsFormatCurrency(1100.5412, 'CZK');
// Returns "1 100,54 CZK"
```

string jsFormatDate(object date)

Description: *The function formats the date specified by the "date" parameter into text based on the date format from the NET Genium settings and the language settings of the currently logged in user:*

- "dd/mm/yyyy" or
- "mm/dd/yyyy".

If the user's language is English:

- "dd/mm/yyyy" or
- "mm/dd/yyyy".

If the user's language is a language other than English:

- "dd.mm.yyyy" or
- "mm.dd.yyyy".

Example:

```
var s = jsFormatDate(new Date());
// Returns the current date in the format "dd.mm.yyyy" (the user´s language is English)
var s = jsFormatDate(new Date());
// Returns the current date in the format "dd/mm/yyyy" (the user´s language is English)
var s = jsFormatDate(Str2Date('01.01.2016 13:00'));
// Returns "01.01.2016" (the user´s language is other than English)
var s = jsFormatDate(Str2Date('01/01/2016 13:00'));
// Returns "01/01/2016" (the user´s language is English)
```

string jsFormatDouble(double number, string symbol)

- ***Description:*** *The function formats the number specified by the "number" parameter into text – with a scale as a thousands separator, with a decimal point or a dot, depending on the language setting of the currently logged in user, and with a symbol specified by the "symbol" parameter.*

- ***Example:***

```
var s = jsFormatDouble(1100, 'kg'));
// Returns "1 100 kg"
var s = jsFormatDouble(1100.5, 'kg'));
// Returns "1 100,5 kg"
var s = jsFormatDouble(1100.5412, 'kg'));
// Returns "1 100,5412 kg"
```

string jsFormatSize(int size)

- ***Description:*** *The function formats the file size specified by the "size" parameter into text.*

- ***Example:***

```
var s = jsFormatSize(1);
// Returns "1 b"
var s = jsFormatSize(1024);
// Returns "1 kB"
var s = jsFormatSize(102456407);
// Returns "97.71 MB"
```

string jsFormatTable(array value, string title, int width, bool showGrid, string noEntries)

string jsFormatTable(array value, string title, int width, bool showGrid, string noEntries, int pageSize, object tb_pageIndex)

- ***Description:*** *The function returns the HTML code of the table formatted in the appearance of the datagrid. The "value" parameter defines the field of table values, the "title" parameter defines the table title, the "width" parameter defines the width of the displayed datagrid. The "showGrid" parameter, which can be "true" or "false", determines whether to display a grid in the table. The "noEntries" parameter defines what text should be displayed if the table does not contain any records. The "pageSize" parameter defines how many records should be contained on one table page, and the "tb_pageIndex" parameter defines the control in which the current page number is stored.*

- ***Example:***

```
<div id="items"></div>

<script type="text/javascript">

var array = [];
array.push(['Name', 'Surname']);
array.push(['A', 'B']);
array.push(['C', 'D']);

el('items').innerHTML = jsFormatTable(array, 'Users', 400, true, 'No records found');
// el('items').innerHTML = jsFormatTable(array, 'Users', 400, true, 'No records found', 1,
#ng_textbox#);

</script>
```

string jsFormatTime(object date)

- ***Description:*** *The function formats the time specified by the "date" parameter into text in the format "HH:mm".*

- ***Example:***

```
var s = jsFormatTime(new Date());
// Returns the current time in the format "HH:mm"
var s = jsFormatTime(Str2Date('1.1.2016 13:00'));
// Returns "13:00"
var s = jsFormatTime(Str2Date('01/01/2016 13:00'));
// Returns "13:00"
```

string jsFormatTime2(object date)

- ***Description:*** *The function formats the time specified by the "date" parameter into text in the format "HH:mm:ss".*

- ***Example:***

```
var s = jsFormatTime(new Date());
// Returns the current time in the format "HH:mm:ss"
var s = jsFormatTime(Str2Date('1.1.2016 13:00'));
// Returns "13:00:00"
var s = jsFormatTime(Str2Date('01/01/2016 13:00'));
// Returns "13:00:00"
```

string jsFormatTime3(object date)

- ***Description:*** *The function formats the time specified by the "date" parameter into a text format in the format "HH:mm:ss.SSS".*

- ***Example:***

```
var s = jsFormatTime(new Date());
// Returns the current time in the format "HH:mm:ss:SSS"
var s = jsFormatTime(Str2Date('1.1.2016 13:00'));
// Returns "13:00:00.000"
var s = jsFormatTime(Str2Date('01/01/2016 13:00'));
// Returns "13:00:00.000"
```

bool jsIsNumber(object o)

- ***Description:*** *The function returns a logical value if the value specified by the "o" parameter is a number or not.*

- ***Example:***

```
var ok = jsIsNumber(#ng_textbox#.value);
```

double jsMeasureHours(object startdate, object enddate)

- ***Description:*** *The function returns the unrounded number of hours from the time period specified by the "startdate" and "enddate" parameters. Both parameters must be javascript objects of type "Date".*

Example:

```
var n = jsMeasureHours(Str2Date('1.1.2000'), Str2Date('1.2.2000'));
```

double jsMeasureMinutes(object startdate, object enddate)

- ***Description:*** *The function returns the unrounded number of minutes from the time period specified by the "startdate" and "enddate" parameters. Both parameters must be javascript objects of type "Date".*

Example:

```
var n = jsMeasureMinutes(Str2Date('1.1.2000'), Str2Date('1.2.2000'));
```

double jsMeasureWorkDays(object startdate, object enddate)

- ***Description:*** *The function returns the number of whole working days from the time period specified by the "startdate" and "enddate" parameters.*
  Both parameters must be javascript objects of type "Date". The function takes into account public holidays registered in the basic application "Settings".

Example:

```
var n = jsMeasureWorkDays(Str2Date('1.1.2000'), Str2Date('1.2.2000'));
```

double jsMeasureWorkDays(object startdate, object enddate, double workinghours)

- ***Description:*** *The function returns the number of whole working days from the time period specified by the "startdate" and "enddate" parameters. The "workinghours" parameter specifies the number of hours of a normal working day. The function returns a decimal number – a proportional part of the working day – if "startdate" and "enddate" are on the same day, and if the length of the time period in hours is shorter than the number of hours of a normal working day.*

Example:

```
var n = jsMeasureWorkDays(Str2Date('1.1.2000'), Str2Date('1.2.2000'), 7.5);
```

double jsN(object o)
double jsNumber(object o)

- **Description:** *The function converts the "o" object to a number.*
- **Example:**

```
var n = jsN(114.12); // Returns "114.12"
var n = jsN(114,12); // Returns "114.12"
```

string jsNetGeniumStart(string date)

- **Description:** *The function returns the time of the first start of NET Genium by the currently logged in user on the day specified by the text parameter "date" in the format "dd.MM.yyyy". NET Genium stores information about the start time in cookies.*
- **Example:**

```
var s = jsNetGeniumStart('#today#');
```

string jsngdph()

- **Description:** *The function returns the current VAT rates separated by a semicolon.*
- **Example:**

```
var s = jsngdph();
```

double jsngdph(int rate)

- **Description:** *The function returns the VAT rate defined by the "rate" parameter. The parameter is defined by number 2 for the increased VAT rate, number 1 for the reduced VAT rate and number 0 for the zero VAT rate.*
- **Example:**

```
var n = jsngdph(1);
// Returns the current reduced VAT rate, e.g. "15"
```

string jsngdph(object date)

- ***Description:*** *The function returns VAT rates separated by a semicolon on the day specified in the "date" parameter.*
- ***Example:***

```
var s = jsngdph(Str2Date('1.1.2000'));
// Returns VAT rates separated by a semicolon valid on the given day, e.g. "22;5;0".
```

double jsngdph(int rate, object date)

- ***Description:*** *Returns the VAT rate defined by the "rate" parameter. The parameter is defined by number 2 for the increased VAT rate, number 1 for the reduced VAT rate and number 0 for the zero VAT rate. The rate is determined on the day specified in the "date" parameter.*
- ***Example:***

```
var n = jsngdph(2, Str2Date('1.1.2000'));
// Returns the increased VAT rate on a given day, e.g. "22"
```

double jsngdph(double base, string type1, double rate, string type2)

- ***Description:*** *The function calculates either the amount or the VAT value, depending on the "type2" parameter. The basic amount defined by the "base" parameter and the VAT rate defined by the "rate" parameter are used for the calculation. The "type1" parameter specifies the value of the "base" parameter.*

  Possible values for the "type1" parameter:

  - bezdph – parameter 1 is the sum without VAT
  - sdph – parameter 1 is the amount with VAT

  Possible values for the "type2" parameter:

  - bezdph – calculates the tax base
  - VAT – calculates the value of VAT
  - zdph – calculates the VAT value rounded to the nearest whole number
  - dph1 – calculates the VAT value of the reduced rate
  - zdph1 – calculates the value of the reduced rate VAT rounded to the nearest whole number
  - dph2 – calculates the VAT value of the increased rate
  - zdph2 – calculates the VAT value of the increased rate rounded to an integer
  - sdph – calculates the amount with VAT
  - zsdph – calculates the amount with VAT rounded up to a whole number
  - zsdph-1 – calculates the amount with VAT rounded mathematically to an integer
  - zsdph-2 – calculates the amount with VAT rounded to 50 pennies

Example:

```
var n = jsngdph(-0.55, 'sdph', 19, 'dph');
// Returns "-0,99"
```

int jsngvatindex(double rate)

Description: *The function returns the index of the current VAT rate specified by the "rate" parameter. The function returns the value "0" in the case of a zero VAT rate, the value "1" in the case of a lower VAT rate, the value "2" in the case of a higher VAT rate and the value "3" in the case of a second reduced VAT rate.*

Example:

```
var n = jsngvatindex(0);
// Returns "0"
var n = jsngvatindex(15);
// Returns "1"
var n = jsngvatindex(21);
// Returns "2"
var n = jsngvatindex(10);
// Returns "3"
```

int jsngvatindex(double rate, string country, object date)

Description: *The function returns the VAT rate index specified by the "rate" parameter in the country specified by the "country" parameter (CZ, DE, HU, SK, …) and on the day specified by the "date" parameter. The function returns the value "0" in the case of a zero VAT rate, the value "1" in the case of a lower VAT rate, the value "2" in the case of a higher VAT rate and the value "3" in the case of a second reduced VAT rate.*

Example:

```
var n = jsngvatindex(0, 'EN', Str2Date('1.1.2020'));
// Returns "0"
var n = jsngvatindex(15, 'EN', Str2Date('1.1.2020'));
// Returns "1"
var n = jsngvatindex(21, 'EN', Str2Date('1.1.2020'));
// Returns "2"
var n = jsngvatindex(10, 'EN', Str2Date('1.1.2020'));
// Returns "3"
```

string jsngvatrates(string country, object date)

- ***Description:*** *The function returns VAT rates separated by a semicolon in the given country (CZ, DE, HU, SK, …) specified by the "country" parameter on the given date specified by the "date" parameter.*

Example:

```
var s = jsngvatrates('DE', Str2Date('1.1.2013'));
```

string jsNumberInWords(double number, string language)

- ***Description:*** *The function returns a verbal expression of the numeric value specified by the "number" parameter. The "language" parameter can contain one of the values "cs", "de", "en", "fr" or "sk". The function rounds to integers.*

Example:

```
var s = jsNumberInWords(12345.678, 'cs');
// Returns "twelvethousandfourhundredfortysix"
```

object jsO(object o)

- ***Description:*** *The function converts the "o" object to an object data type. The function is used to test the user's rights to the relevant control.*

Example of correct use:

```
var o = jsO(#ng_tb#);
// If the user does not have the rights to the control "#ng_tb#", it will be replaced
"#ng_tb#" by the value stored in this element in the database
if (o != null)
{
}
```

Example of misuse:

```
var o = #ng_tb#;
// If the user does not have the rights to the control "#ng_tb#", it may be replaced "#ng_tb#"
with an empty value. The result will be a synthetic error "var o = ;"
```

void jsOpen(string url)

- ***Description:*** *The function opens the web page specified by the "url" parameter in a new web browser window.*
- ***Example:***

```
jsOpen('https://www.netgenium.com');
```

int jsR(double number)
int jsRound(double number)

- ***Description:*** *The function returns the rounded value of the number specified by the "number" parameter.*
- ***Example:***

```
var n = jsR(21.232);
// Returns "21"
```

double jsR(double number, int decimals)
double jsRound(double number, int decimals)

- ***Description:*** *The function returns the number specified by the "number" parameter, rounded to the number of decimal places specified by the "decimals" parameter.*
- ***Example:***

```
var n = jsR(2.354, 2);
// Returns "2.35"
```

double jsR_50(double number)
double jsRound_50(double number)

- ***Description:*** *The function returns the number specified by the "number" parameter, rounded to fifty pennies.*
- ***Example:***

```
var n = jsR_50(123.456);
// Returns "123,5"
```

bool jsUserInGroup(string name)

- ***Description:*** *The function returns a logical value if the currently logged in user is a member of the user group specified by the "name" parameter or not. "Name" can be the id of the user group or its name.*

- ***Example:***

```
alert('User ' + (jsUserInGroup(2) ? 'is' : 'is not') + ' group member <b>Users</b>');
alert('User ' + (jsUserInGroup('Users') ? 'is' : 'is not') + ' group member <b>Users</b>');
```

bool jsValidateEmail(string value)

- ***Description:*** *The function receives an e-mail address via the "value" parameter and verifies that the address is valid (in the correct format for e-mail). It then returns "true" or "false".*

- ***Example:***

```
var ok = jsValidateEmail('info@netgenium.com');
```

bool jsValidateEmails(string value)

- ***Description:*** *The function accepts e-mail addresses separated by a comma or a semicolon via the "value" parameter*
  and verifies that the addresses entered are valid (in the correct format for e-mail). Then it returns "true" or "false". The "value" can contain spaces between individual e-mails.

- ***Example:***

```
var ok = jsValidateEmails('info@netgenium.com');
var ok = jsValidateEmails('i@netgenium.com, n@netgenium.com;  f@netgenium.com');
```

string jsVisibility(bool value)

- ***Description:*** *The function returns "visible" (element will be visible) or "hidden" (element will not be visible) according to the content of the "value" parameter. The function is used to set the "style.visibility" attribute of individual controls of the edit form.*

- ***Example:***

```
#ng_textbox:Table#.style.visibility = jsVisibility(#ng_checkbox#.checked);
D1.style.visibility = jsVisibility(#ng_checkbox#.checked);
```

bool jsWait(string url, string pleaseWait)

- ***Description:*** *The function hides the content of the web page, displays the message defined by the "pleaseWait" parameter, and redirects the user to a new web page with the address specified by the "url" parameter. The function returns "false".*
- ***Example:***

```
jsWait('https://www.netgenium.com', 'Please wait…');
```

void lb_Disable(object lb)

- ***Description:*** *The function disables the editing of the "ListBox" control specified by the "lb" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
lb_Disable(#ng_listbox#);
```

void lb_Enable(object lb)

- ***Description:*** *The function enables editing of the "ListBox" control specified by the "lb" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
lb_Enable(#ng_listbox#);
```

bool letterOrDigit(string value)

- ***Description:*** *The function returns "true" if the text specified by the "value" parameter begins with a letter or number. The function returns "false" if the text specified by the "value" parameter begins with a character other than a letter or number.*
- ***Example:***

```
letterOrDigit('NET Genium'); // Returns "true"
letterOrDigit('-a**'); // Returns "false"
```

void loadUrl(string url)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. Only one asynchronous call can be called at a time using the "loadUrl" function. If the "loadUrl" function is called again if the previous asynchronous call has not yet been completed, the new asynchronous call will not take place. The "_loadUrl" function is intended for parallel calling.*

- ***Example:***

```
var p0 = 'ěščřžýáíé', p1 = 'ĚŠČŘŽÝÁÍÉ';
loadUrl('ngef.aspx?MyFirstFunction,' + urlEncode(p0) + ',' + urlEncode(p1));
```

void loadUrl(string url, string f)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. When the answer then arrives, the execution of the main context of the code is suspended (sooner or later depending on the priority of the performed task) and the so-called callback defined by the "f" parameter. The resulting HTML code of the page is passed to the "f" function as the first parameter. Only one asynchronous call can be called at a time using the "loadUrl" function. If the "loadUrl" function is called again if the previous asynchronous call has not yet been completed, the new asynchronous call will not take place. The "_loadUrl" function is intended for parallel calling.*

- ***Example:***

```
var p0 = 'ěščřžýáíé', p1 = 'ĚŠČŘŽÝÁÍÉ';
loadUrl('ngef.aspx?MyFirstFunction,' + urlEncode(p0) + ',' + urlEncode(p1), ajaxResponse);

function ajaxResponse(html)
{
  alert(html);
}
```

void loadUrl(string url, string f, string id)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. When the answer then arrives, the execution of the main context of the code is suspended (sooner or later depending on the priority of the performed task) and the so-called callback defined by the "f" parameter. The resulting HTML code of the page is passed to the "f" function as the first parameter. The optional parameter "id" is passed to the "f" function as the second parameter. Only one asynchronous call can be called at a time using the "loadUrl" function. If the "loadUrl" function is called again if the previous asynchronous call has not yet been completed, the new asynchronous call will not take place. The "_loadUrl" function is intended for parallel calling.*

- ***Example:***

```
var p0 = 'ěščřžýáíé', p1 = 'ĚŠČŘŽÝÁÍÉ';
loadUrl('ngef.aspx?MyFirstFunction,' + urlEncode(p0) + ',' + urlEncode(p1), ajaxResponse,
'test');

function ajaxResponse(html, id)
{
  alert(id + ': ' + html);
}
```

void loadUrl(string url, string f, string id, string params)

- ***Description:*** *The function loads the page defined by the "url" parameter using an asynchronous ajax call, resp. loads the HTML code of this page. The function does not wait for a response from the server, after calling it, the execution of javascript code continues. When the answer then arrives, the execution of the main context of the code is suspended (sooner or later depending on the priority of the performed task) and the so-called callback defined by the "f" parameter. The resulting HTML code of the page is passed to the "f" function as the first parameter. The optional parameter "id" is passed to the "f" function as the second parameter. The parameter "params" is sent together with the asynchronous call by the POST method, and it is a list of key vs. pair pairs. value separated by a "&" character. Only one asynchronous call can be called at a time using the "loadUrl" function. If the "loadUrl" function is called again if the previous asynchronous call has not yet been completed, the new asynchronous call will not take place. The "_loadUrl" function is intended for parallel calling.*

- ***Example:***

```
var p0 = 'ěščřžýáíé', p1 = 'ĚŠČŘŽÝÁÍÉ';
loadUrl('ngef.aspx?MyFirstFunction', ajaxResponse, 'test', 'p0=' + urlEncode(p0) + '&p1=' +
urlEncode(p1));

function ajaxResponse(html, id)
{
  alert(id + ': ' + html);
}
```

bool loginByFacebook()

- ***Description:*** *The function triggers a login to the Facebook account on a new web browser tab. Successful login to Facebook sets the value of the variable "#facebook_access_token#". The function returns "false" if the user is already logged in to Facebook.*

- ***Example:***

```
var ok = loginByFacebook();
```

bool loginByGoogle(string scope, bool offline)

- ***Description:*** *The function invokes a login to the Google Account on a new web browser tab, and passes the value of the "scope" parameter, which defines the scope of access rights. Successful login to Google will set the value of the variable "#google_access_token#". The "offline" parameter determines whether a successful login should also set the value of the "#google_refresh_token#" variable. The function returns "false" if the user is already logged in to Google.*

- ***Example:***

```
var ok = loginByGoogle('https://www.googleapis.com/auth/youtube', false);
```

bool loginByMicrosoft(string scope, bool offline)

- ***Description:*** *The function invokes a login to the Microsoft account on a new web browser tab, and passes the value of the "scope" parameter, which defines the scope of access rights. Successful login to Microsoft sets the value of the variable "#microsoft_access_token#". The "offline" parameter determines whether the successful login should also set the value of the "#microsoft_refresh_token#" variable. The function returns "false" if the user is already logged in to Microsoft.*

- ***Example:***

```
var ok = loginByMicrosoft('openid email', false);
```

void mlb_Disable(object mlb)

- ***Description:*** *The function disables editing of the "MultiListBox" control specified by the "mlb" parameter. An alternative is the general function "control_Disable".*

- ***Example:***

```
mlb_Disable(#ng_multilistbox#);
```

void mlb_Enable(object mlb)

- ***Description:*** *The function enables editing of the "MultiListBox" control specified by the "mlb" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
mlb_Enable(#ng_multilistbox#);
```

string Now()

- ***Description:*** *The function returns the current date and time in the format "dd.mm.yyyy hh:mm".*
- ***Example:***

```
var s = Now();
```

string Now2()

- ***Description:*** *The function returns the current date and time in the format "dd.mm.yyyy hh:mm:ss".*
- ***Example:***

```
var s = Now2();
```

int offsetLeft2(object o)

- ***Description:*** *The function returns the x-coordinate of the "o" element from the upper left corner of the web page.*
- ***Example:***

```
var n = offsetLeft2(#ng_textbox#);
```

int offsetTop2(object o)

- ***Description:*** *The function returns the y-coordinate of the "o" element from the upper left corner of the web page.*
- ***Example:***

```
var n = offsetTop2(#ng_textbox#);
```

void picker_Open(object o)

- ***Description:*** *The function opens the picker of the control specified by the "o" parameter.*
- ***Example:***

```
picker_Open(#ng_textbox#);
```

string queryString(string value)

- ***Description:*** *The function returns the parameter "value" from the address "window.location".*
- ***Example:***

```
var s = queryString('id');
```

string queryString(string value, string url)

- ***Description:*** *The function returns the "value" parameter from the address specified by the "url" parameter.*
- ***Example:***

```
var s = queryString('id', 'https://www.netgenium.com?id=123');
// Returns "123"
var s = queryString('id', window.location.tostring());
```

void rbl_Disable(object rbl)

- ***Description:*** *The function disables the editing of the "Radio" control specified by the "rbl" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
rbl_Disable(#ng_radio#);
```

void rbl_Enable(object rbl)

- ***Description:*** *The function enables editing of the "Radio" control specified by the "rbl" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
rbl_Enable(#ng_radio#);
```

void rbl_EnabledItems(object rbl, array items)

- ***Description:*** *The function allows the possibility to select only specific items for the "Radio" control specified by the "rbl" parameter. The list of items is defined by the "items" parameter.*
- ***Example:***

```
rbl_EnabledItems(#ng_radio#, ['a', 'b']);
```

int rbl_SelectedIndex(object rbl)

- ***Description:*** *The function returns the index of the selected "Radio" control item. Returns "-1" if no item is selected.*
- ***Example:***

```
var n = rbl_SelectedIndex(#ng_radio#);
```

string rbl_SelectedItem_Text(object rbl)

- ***Description:*** *The function returns the text assigned to the selected item in the "Radio" control.*
- ***Example:***

```
var s = rbl_SelectedItem_Text(#ng_radio#);
```

string rbl_SelectedItem_Value(object rbl)

- ***Description:*** *The function returns the value assigned to the selected item in the "Radio" control. If no item is selected, the function returns an empty string.*
- ***Example:***

```
var s = rbl_SelectedItem_Value(#ng_radio#);
```

bool rbl_SelectText(object rbl, string value)

- ***Description:*** *The function searches for an item in the "Radio" control according to its text and selects it if it finds it. If the searched item is not found, the function returns "false".*
- ***Example:***

```
rbl_SelectText(#ng_radio#, 'Prague');
```

bool rbl_SelectValue(object rbl, string value)

- ***Description:*** *The function searches for an item in the "Radio" control by its value and selects it if it finds it. If the searched item is not found, the function returns "false".*
- ***Example:***

```
rbl_SelectValue(#ng_radio#, 'Prague');
```

string removeDiacritics(string value)

- ***Description:*** *The function removes diacritics from the string specified by the "value" parameter.*
- ***Example:***

```
var s = removeDiacritics('ěščřžýáié'); // Returns "escrzyaie"
```

string removeDoubleCharacters(string value, string ch)

- ***Description:*** *The function removes duplicate characters defined by the "ch" parameter from the string specified by the "value" parameter.*
- ***Example:***

```
var s = removeDoubleCharacters('a  b  c', ' ');
```

string removeDoubleLines(string value)

- ***Description:*** *The function removes duplicate blank lines from the multiline string specified by the "value" parameter.*
- ***Example:***

```
var s = removeDoubleLines('a\r\n\r\n\r\nbc');
```

string removeDoubleSpaces(string value)

- ***Description:*** *The function removes double spaces from the string specified by the "value" parameter.*
- ***Example:***

```
var s = removeDoubleSpaces('a  b  c');
```

string removeDuplicates(string value, string separator)

- ***Description:*** *The function removes all duplicate items from the items specified by the "value" parameter. The separator that is used when specifying records is defined by the "separator" parameter.*
- ***Example:***

```
var s = removeDuplicates('aa;bb;aa;cc', ';'); // Returns "aa;bb;cc"
```

string removeNonAlphanumericCharacters(string value)

- ***Description:*** *The function removes all characters except alphanumeric characters from the string specified by the "value" parameter.*
- ***Example:***

```
var s = removeNonAlphanumericCharacters('1:a:2');
```

string removeNonDigits(string value)

- ***Description:*** *The function removes all characters except numbers from the string specified by the "value" parameter.*
- ***Example:***

```
var s = removeNonDigits('1.2.3');
```

string removeTags(string html)

- ***Description:*** *The function removes all tags from the string specified by the "html" parameter, and replaces the " ", "
", "<hr>", "</p>" and "
" after blank lines using the substring "\r\n".*
- ***Example:***

```
var s = removeTags('<b>Yes</b>');
// Returns "Yes"
```

string removeTags2(string html)

- ***Description:*** *The function removes all tags from the string specified by the "html" parameter.*
- ***Example:***

```
var s = removeTags2('<p>abc</p><p>abc</p>');
// Returns "abcabc"
```

string removeUnnecessarySpaces(string value)

- ***Description:*** *The function removes all extra white spaces from the string specified by the "value" parameter – all white spaces at the beginning of the string and at the end of the string, and replaces the double spaces in the middle with single spaces.*
- ***Example:***

```
var s = removeUnnecessarySpaces(' a b c ');
```

string removeWhiteSpaces(string value)

- ***Description:*** *The function removes all spaces (spaces and tabs) from the string specified by the "value" parameter.*
- ***Example:***

```
var s = removeWhiteSpaces('a b c');
```

string replaceAll(string value, string oldvalue, string newvalue)

- ***Description:*** *The function returns the text string specified by the "value" parameter, in which all occurrences of the string stored in the "oldvalue" parameter were replaced by the string stored in the "newvalue" parameter.*
- ***Example:***

```
var s = replaceAll('Lálá', 'á', 'a');
// Returns "Lala"
```

void rtb_Disable(object rtb)

- ***Description:*** *The function disables editing of the "RichTextBox" control specified by the "rtb" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
rtb_Disable(#ng_richtextbox#);
```

void rtb_Enable(rtb object)

- ***Description:*** *The function allows editing of the "RichTextBox" control specified by the "rtb" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
rtb_Enable(#ng_richtextbox#);
```

void rtb_Focus(object rtb)

- ***Description:*** *The function sets the focus to the "RichTextBox" control defined by the "rtb" parameter.*
- ***Example:***

```
rtb_Focus(#ng_#);
```

void rtb_Insert(object rtb, string value)

- ***Description:*** *The function inserts the value specified by the "value" parameter at the current cursor position into the "RichTextBox" control.*
- ***Example:***

```
rtb_Insert(#ng_richtextbox#, 'Text');
```

string script_Error(string value)

- ***Description:*** *This javascript function is always executed when a script interrupt is called in the edit form or on the view page. The interrupt text is passed to the "value" parameter as set in the script designer. To easily identify the type of interrupt, it is recommended to add, for example, a numeric code to the text of the interrupt, and then try to find this code in javascript. The javascript function "script_Error" must be defined as a whole in the edit form or on the view page, i.e. including "function script_Error(value)".*

- ***Example:***

```
Interruption "Error occurred(123)"

function script_Error(value)
{
  if (value.indexOf('(123)') != -1)
  {
    // ...
  }
}
```

void scroll_To(int x, int y, bool save)

- ***Description:*** *The function scrolls the edit form or view page to the position defined by the "x" and "y" parameters. The optional "save" parameter means that this position should be remembered immediately, usually when the "scroll_To" function is called by the server script button.*

- ***Example:***

```
scroll_To(0, 0);
scroll_To(0, 0, true);
```

void setTimeout2(object o, function f, int timeout)

- ***Description:*** *The function calls another function defined in the "f" parameter after the time period defined in the "timeout" parameter has elapsed. The function always binds to the specific object specified by the "o" parameter. If the "setTimeout2" function with the same parameters is called again during the specified time period, the previous call is canceled and a new one is scheduled again after the time period defined in the "timeout" parameter has elapsed. The value of the "timeout" parameter is given in milliseconds.*

Example:

```
var saveObject = new Object();

function form_Change()
{
  setTimeout2(saveObject, function() { form_Save(); }, 1000);
}
```

string StringBuilder()

- ***Description:*** *"StringBuilder" is the equivalent of the C# class of the same name, designed for optimized addition of long strings.*

Example:

```
var sb = new StringBuilder();
for (var i = 0; i < 10; i++)
{
  if (!sb.isEmpty()) sb.append(';');
  sb.append(i);
}
sb.appendLine();
sb.appendLine('abc');
var s = sb.toString();
```

object Str2Date(string text)

- ***Description:*** *The function returns the date converted from the text string specified by the "text" parameter in the format "dd.MM.yyyy" or "dd.MM.yyyy HH:mm".*

Example:

```
var d = Str2Date(#ng_date#.value);
var d = Str2Date('#today#');
var d = Str2Date('#now#');
```

void ta_Disable(object ta)

- ***Description:*** *The function disables the editing of the "TextArea" control specified by the "ta" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
ta_Disable(#ng_textarea#);
```

void ta_Enable(object ta)

- ***Description:*** *The function allows editing of the "TextArea" control specified by the "ta" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
ta_Enable(#ng_textarea#);
```

void ta_Insert(object ta, object value)

- ***Description:*** *The function inserts the value "value" at the current cursor position into the "TextArea" control specified by the "ta" parameter.*
- ***Example:***

```
ta_Insert(#ng_textarea#, 'Text');
```

void tab_Disable(int n)

- ***Description:*** *The function disables the opening of the bookmark specified by the index "n". The indices are numbered from zero.*
- ***Example:***

```
tab_Disable(0);
```

void tab_Enable(int n)

- ***Description:*** *The function allows the opening of the bookmark specified by the index "n". The indices are numbered from zero.*
- ***Example:***

```
tab_Enable(0);
```

void tab_Open(int n)

- ***Description:*** *The function opens a tab with the given index – parameter "n". The indices are numbered from zero.*
- ***Example:***

```
tab_Open(0);
```

void tb_Disable(object tb)

- ***Description:*** *The function disables the editing of the "TextBox" control specified by the "tb" parameter. An alternative is the general function "control_Disable".*
- ***Example:***

```
tb_Disable(#ng_textbox#);
```

void tb_Enable(object tb)

- ***Description:*** *The function enables editing of the "TextBox" control specified by the "tb" parameter. An alternative is the general function "control_Enable".*
- ***Example:***

```
tb_Enable(#ng_textbox#);
```

void tb_InitAjax(object tb, function f, int width, int left, bool focus)

Description: *The function initializes its own ajax picker, which is defined in the external function "ngef". The control for which the picker will be initialized is defined by the "tb" parameter. The "f" parameter defines the function that will be used to load the picker. The "width" parameter specifies the width of the picker in pixels. The "left" parameter determines the displacement of the picker window on the x-axis with respect to the location of the respective control. The "focus" parameter determines whether the picker is to be displayed when the control is clicked (true) or when entering data (false).*

Example of the client part:

```
tb_InitAjax(#ng_tb#, function(e) { startSearch(e, 'test1', 'test2'); }, 150, 0, false);

function startSearch(e, p0, p1)
{
  // tb_LoadUrl(e, 'ngef.aspx?TestPicker,' + urlEncode(#ng_tb#.value) + ',' + p0 + ',' + p1);
  tb_LoadUrl(e, 'TestPicker.aspx?' + urlEncode(#ng_tb#.value));
}
```

Example server part (source code of TestPicker.aspx):

```
StringBuilder sb = new StringBuilder();

DataTable data = new DataTable();
data.Columns.Add("value");

for (char ch1 = 'A'; ch1 <= 'C'; ch1++)
  for (char ch2 = 'A'; ch2 <= 'C'; ch2++)
    for (char ch3 = 'A'; ch3 <= 'C'; ch3++)
    {
      data.Rows.Add(ch1.ToString() + ch2.ToString() + ch3.ToString());
    }

DataView view = new DataView(data, args[0].Length != 0 ? "value LIKE " +
ParserSql.ToString(args[0], true, true, DbDriver.DataView) : "", "value",
DataViewRowState.CurrentRows);
int n = 0;

foreach (DataRowView row in view)
{
  if (n == 20)
  {
    sb.Append("<br>...");
    break;
  }

  if (sb.Length != 0) sb.Append("<br>");

  sb.Append("<a class=\"lf\" href=\"javascript:\" onclick=\"el('");
  sb.Append(args[1]);
  sb.Append("').value = '");
  sb.Append(row[0]);
```

```
  sb.Append("'; return false;\">");
  sb.Append(row[0]);
  sb.Append("</a>");
  n++;
}

if (sb.Length == 0)
{
  sb.Append("OK");
}

Html.FlushAjaxContent(sb.ToString());
```

int Time2Int(string time)

- ***Description:*** *The function returns the number of minutes converted from the text string – time – specified by the "time" parameter in the format "HH:mm".*

- ***Example:***

```
var n = Time2Int('01:23');
// Returns "83"
var n = Time2Int('-01:24');
// Returns "-84"
```

string translate(string value)

- ***Description:*** *The function returns the language translation of the multilingual term specified by the "value" parameter based on the language setting of the currently logged in user.*

- ***Example:***

```
var s = translate('Approval has been canceled#en:Approval has been canceled');
```

string translate(string value, string language)

- ***Description:*** *The function returns the language translation of the multilingual term specified by the "value" parameter based on the language defined by the "language" parameter.*

- ***Example:***

```
var s = translate('Approval has been canceled#en:Approval has been canceled', '#language#');
```

string urlDecode(string text)

- ***Description:*** *The function decodes all URL hexadecimal characters in the text string specified by the "text" parameter.*

Example:

```
var s = urlDecode('%c4%9b%c5%a1%c4%8d');
```

string urlEncode(string text)

- ***Description:*** *The function encodes all URL invalid characters (eg a space) in the text string specified by the "text" parameter after the corresponding sequence of characters in hexadecimal.*

Example:

```
var s = urlEncode('Too yellowish horse sand devilish ode');
```

string ZInt(int n)

- ***Description:*** *The function returns the text form of an integer "n" with a minimum length of 2 characters by placing a zero before a number less than 10.*

Example:

```
var s = ZInt(8);
// Returns "08"
```