netgenium

Javascriptové funkce

Framework NET Genium

1 Proměnné

- nID – ID aktuálně otevřeného záznamu v editačním formuláři (není doporučeno používat #id#)
- Page_IsPostBack – Stránka je načtena po provedení postbacku
- dateSeparator – Oddělovač datumu
- domLib_isIE – Aktuální prohlížeč je Internet Explorer
- domLib_isIE56 – Aktuální prohlížeč je Internet Explorer verze 5 nebo 6
- domLib_isIE7 – Aktuální prohlížeč je Internet Explorer verze 7
- domLib_isIE8 – Aktuální prohlížeč je Internet Explorer verze 8
- domLib_isIE9 – Aktuální prohlížeč je Internet Explorer verze 9
- domLib_isIE56789 – Aktuální prohlížeč je Internet Explorer ve verzích 5, 6, 7, 8 nebo 9
- domLib_isIE10 – Aktuální prohlížeč je Internet Explorer verze 10
- domLib_isIE11 – Aktuální prohlížeč je Internet Explorer verze 11
- domLib_isChrome – Aktuální prohlížeč je Google Chrome, Microsoft Edge, nebo Opera verze 15 a vyšší
- domLib_isFirefox – Aktuální prohlížeč je Mozilla Firefox
- domLib_isSafari – Aktuální prohlížeč je Safari
- domLib_isOpera – Aktuální prohlížeč je Opera verze 12.15 nebo nižší
- domLib_isMobileDevice – Aktuální prohlížeč je mobilní zařízení
- domLib_isCE – Aktuální prohlížeč zařízení Windows CE

2 Funkce

void _loadUrl(object o, string url)

- ***Popis:*** *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. Parametr „o“ je pomocný objekt, do kterého jsou ukládány všechny informace potřebné k úspěšnému dokončení asynchronního ajaxováho volání, a tímto objektem může být cokoliv v objektovém modelu webové stránky nebo nově vytvořený objekt. Pomocí jednoho objektu je možné v jednu chvíli volat pouze jedno asynchronní volání. Pokud by došlo k opětovnému zavolání funkce „_loadUrl“ se stejným pomocným objektem, u kterého dosud nebylo nedokončeno asynchronní volání, k novému asynchronnímu volání nedojde. Pokud je nutné paralelní volání funkce „_loadUrl“, jako pomocný objekt „o“ je ideální zvolit „new Object()“.*

- ***Příklad:***

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

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

- ***Popis:*** *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. callbacku definovaného parametrem „f“. Výsledný HTML kód stránky je funkci „f“ předán jako první parametr. Parametr „o“ je pomocný objekt, do kterého jsou ukládány všechny informace potřebné k úspěšnému dokončení asynchronního ajaxováho volání, a tímto objektem může být cokoliv v objektovém modelu webové stránky nebo nově vytvořený objekt. Pomocí jednoho objektu je možné v jednu chvíli volat pouze jedno asynchronní volání. Pokud by došlo k opětovnému zavolání funkce „_loadUrl“ se stejným pomocným objektem, u kterého dosud nebylo nedokončeno asynchronní volání, k novému asynchronnímu volání nedojde. Pokud je nutné paralelní volání funkce „_loadUrl“, jako pomocný objekt „o“ je ideální zvolit „new Object()“.*

- ***Příklad:***

```
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)

Popis: *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. callbacku definovaného parametrem „f“. Výsledný HTML kód stránky je funkci „f“ předán jako první parametr. Volitelný parametr „id“ je funkci „f“ předán jako druhý parametr. Parametr „o“ je pomocný objekt, do kterého jsou ukládány všechny informace potřebné k úspěšnému dokončení asynchronního ajaxováho volání, a tímto objektem může být cokoliv v objektovém modelu webové stránky nebo nově vytvořený objekt. Pomocí jednoho objektu je možné v jednu chvíli volat pouze jedno asynchronní volání. Pokud by došlo k opětovnému zavolání funkce „_loadUrl“ se stejným pomocným objektem, u kterého dosud nebylo nedokončeno asynchronní volání, k novému asynchronnímu volání nedojde. Pokud je nutné paralelní volání funkce „_loadUrl“, jako pomocný objekt „o“ je ideální zvolit „new Object()“.*

Příklad:

```
function ajaxTest(id)
{
  var p0 = 'ěščřžýáíé', 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)

- ***Popis:*** *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. callbacku definovaného parametrem „f“. Výsledný HTML kód stránky je funkci „f“ předán jako první parametr. Volitelný parametr „id“ je funkci „f“ předán jako druhý parametr. Parametr „params“ je spolu s asynchronním voláním odeslán metodou POST, a jde o seznam párů klíč vs. hodnota oddělených pomocí znaku „&“. Parametr „o“ je pomocný objekt, do kterého jsou ukládány všechny informace potřebné k úspěšnému dokončení asynchronního ajaxováho volání, a tímto objektem může být cokoliv v objektovém modelu webové stránky nebo nově vytvořený objekt. Pomocí jednoho objektu je možné v jednu chvíli volat pouze jedno asynchronní volání. Pokud by došlo k opětovnému zavolání funkce „_loadUrl“ se stejným pomocným objektem, u kterého dosud nebylo nedokončeno asynchronní volání, k novému asynchronnímu volání nedojde. Pokud je nutné paralelní volání funkce „_loadUrl“, jako pomocný objekt „o“ je ideální zvolit „new Object()“.*

- ***Příklad:***

```
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)

- ***Popis:*** *Funkce připojí funkci „listener“ k události objektu „o“. „Type“ definuje typ události, např. load, scroll apod.*

- ***Příklad:***

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

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

void bt_Click()

Popis: *Funkce vyvolá znovunačtení formuláře (postback). Při této události dojde ke znovunačtení dat všech tabulek, kalendářů, grafů apod. Tato funkce byla nahrazena funkcí „form_Update“.*

Příklad:

```
bt_Click();
```

void bt_Click(string id)

Popis: *Funkce vyvolá programové kliknutí na tlačítko zadané parametrem „id“. Funkce se nejdříve snaží vyhledat prvek s daným ID, a pokud nenajde, tak s daným textem.*

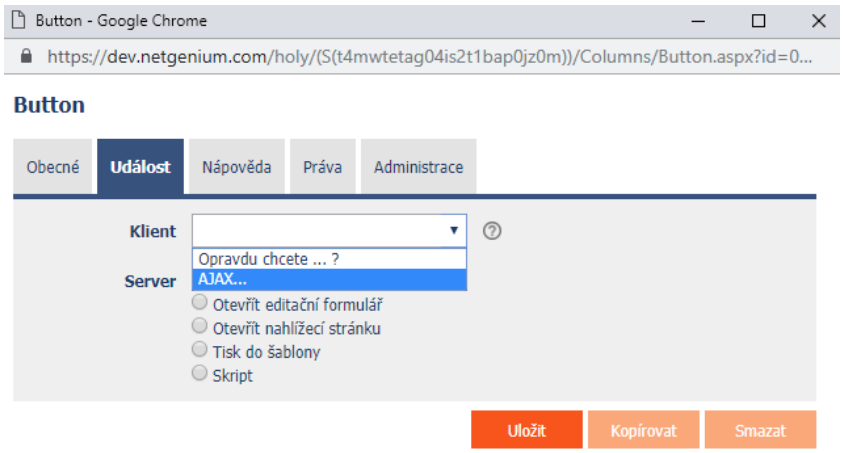
Příklad:

```
bt_Click(BT123);
bt_Click('Uložit');
```

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

Popis: *Funkce při stisknutí tlačítka načte stránku (typicky externí funkci) definovanou parametrem „url“, a pomocí validující funkce definované parametrem „f“ ověří načtený HTML kód. Validující funkce musí vracet logickou hodnotu „true“ nebo „false“, která určuje, zda se má pokračovat při zpracování „onclick“ události. Chování funkce „bt_Eval“ lze definovat při určení JavaScriptu tlačítka (viz obrázek). A dále v javascriptu příslušné nahlížecí stránky nebo editačního formuláře.*

Příklad:



***Příklad** - 1. varianta:*

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

***Příklad** - 2. varianta:*

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

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

***Příklad** - 3. varianta:*

```
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)

Popis: *Funkce vrací tlačítko s titulkem zadaným parametrem „text“. Parametr „index“ je volitelný a říká, kolikátý výskyt tlačítka s titulkem „index“ má funkce vrátit. Tento parametr má význam v editačních formulářích, kde je umístěno více tlačítek se stejným titulkem. Číslování probíhá od nuly, první tlačítko má tedy index 0.*

Příklad:

```
var bt = bt_FindByText('Dnes', 2); // Vrací objekt tlačítka, které je nazvané „Dnes“ a je ve
formuláři jako třetí v pořadí s tímto názvem
```

bool bt_Icon(object bt, string url)

- ***Popis:*** *Funkce nastaví na pozadí tlačítka obrázek definovaný parametrem „url“.*
- ***Příklad:***

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

void button_Disable(o)

- ***Popis:*** *Funkce disabluje tlačítko zadané parametrem „o” v editačním formuláři.*
- ***Příklad:***

```
button_Disable(BT123);
```

void buttons_Disable()

- ***Popis:*** *Funkce disabluje všechna tlačítka v editačním formuláři.*
- ***Příklad:***

```
buttons_Disable();
```

void button_Enable(o)

- ***Popis:*** *Funkce enabluje tlačítko zadané parametrem „o” v editačním formuláři.*
- ***Příklad:***

```
button_Enable(BT123);
```

void buttons_Enable()

- ***Popis:*** *Funkce enabluje všechna tlačítka v editačním formuláři.*
- ***Příklad:***

```
buttons_Enable();
```

void cb_Disable(object cb)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „ComboBox“ zadaného parametrem „cb“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void cb_Enable(object cb)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „ComboBox“ zadaného parametrem „cb“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

string Color2Hex(string value)

- ***Popis:*** *Funkce formátuje barvu zadanou parametrem „value“ do hexadecimální podoby ze vstupního formátu „R,G,B“.*
- ***Příklad:***

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

void control_Disable(object o)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku definovaného parametrem „o“.*
- ***Příklad:***

```
control_Disable(#ng_ovladaciprvek#);
```

void control_Disable(object o, bool disableLink)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku definovaného parametrem „o“. Volitelný parametr „disableLink“ (true/false) určuje, zda se má u ovládacího prvku „ForeignKey“ deaktivovat odkaz na vybraný záznam.*

Příklad:

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

void control_Enable(object o)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku definovaného parametrem „o“.*

Příklad:

```
control_Enable(#ng_ovladaciprvek#);
```

string control_GetValue(object o)

- ***Popis:*** *Funkce vrací hodnotu databázového ovládacího prvku zadaného parametrem „o“. Hodnotu ovládacího prvku je možné zjistit také zjednodušeně pomocí zápisu „#ng_ovladaciprvek#.value“, tento zápis se ale nedá použít u následujících ovládacích prvků:*
  - TextBox typu Datum se zapnutým časem – čas je umístěn v jiném textovém poli než samotné datum, takže „#ng_textbox#.value“ vrací pouze datumovou složku bez času, jediný možný způsob pro vrácení datumu i s časem je „control_GetValue(#ng_textbox#)“
  - MultiListBox – jediný možný způsob pro vrácení seznamu vybraných hodnot oddělených tabulátorem je „control_GetValue(#ng_multilistbox#)“
  - Radio – jediný možný způsob pro vrácení vybrané hodnoty je „control_GetValue(#ng_radio#)“
  - CheckBox – vedle funkce „control_GetValue(#ng_checkbox#)“, která vrací textovou hodnotu uloženou do databáze v případě zaškrtnutí tlačítka, nebo prázdný textový řetězec v případě nezaškrtnutí tlačítka, je možné používat logickou hodnotu „#ng_checkbox#.checked“, která vrací zaškrtnutí/nezaškrtnutí tlačítka

Příklad:

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

void control_ResizeTable(object o, int width)

- ***Popis:*** *Funkce nastaví šířku tabulky, ve které je zobrazen ovládací prvek zadaný parametrem „o“, na novou šířku zadanou parametrem „width“.*

Příklad:

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

void control_Save(object o)

- ***Popis:*** *Funkce uloží hodnotu databázového ovládacího prvku do databáze pomocí ajaxu. Při uložení se nevykonávají události „OnBeforeSave“ ani „OnAfterSave“. Parametr „o” určuje databázový ovládací prvek, jehož hodnota bude do databáze uložena.*

Příklad:

```
control_Save(#ng_ovladaciprvek#);
```

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

- ***Popis:*** *Funkce uloží hodnotu databázového ovládacího prvku do databáze pomocí ajaxu. Při uložení se nevykonávají události „OnBeforeSave“ ani „OnAfterSave“. Parametr „o” určuje databázový ovládací prvek, jehož hodnota bude do databáze uložena. Parametr „f“ udává název funkce, která se má spustit po načtení odpovědi ze serveru. Tato odpověď je buď „OK“ (záznam se podařilo uložit) nebo „ERROR“ (záznam se nepodařilo uložit). Parametr „errorMessage“ definuje text chybového hlášení, který se zobrazí v případě, že se při ukládání vyskytne chyba.*

Příklad:

```
control_Save(#ng_ovladaciprvek#, ajaxResponse, 'Vyskytla se chyba');

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

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

- ***Popis:*** *Funkce vloží do databázového ovládacího prvku zadaného parametrem „o“ hodnotu definovanou parametrem „value“.*
  - Jde-li o „TextBox“ s validací „Date“ nebo „DatePicker“, je možné použít parametr „value“ jako textový řetězec formátovaný ve tvaru „dd.MM.rrrr HH:mm“, nebo javascriptový objekt typu „Date“. Parametr

„value2“ je volitelný a určuje čas ve formátu „HH:mm“. Parametr má význam pouze u „TextBoxu“ s validací „Date“ a zvoleným zadáváním času.

- Je-li parametr „value“ javascriptový objekt typu „Date“, není nutné parametr „value2“ uvádět, NET Genium si automaticky z parametru „value“ načte i časový údaj.
- Jde-li o „CheckBox“, je možné použít dva typy parametrů „value“:
  - Je-li parametr „value“ logická hodnota, je řečeno, zda zaškrtávací tlačítko zatrhnout či nikoliv.
  - Je-li parametr „value“ textový řetězec, bude jeho hodnota nejprve porovnávána s hodnotou nastavenou ve vlastnostech ovládacího prvku, a podle výsledku tohoto testu bude teprve zaškrtávací tlačítko zatrženo či nikoliv.

Příklad:

```
control_SetValue(#ng_ovladaciprvek#, '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()

Popis: *Funkce zakáže editaci všech ovládacích prvků (kromě buttonů) v editačním formuláři.*

Příklad:

```
controls_Disable();
```

void controls_Disable(bool disableLinks)

Popis: *Funkce zakáže editaci všech ovládacích prvků (kromě buttonů) v editačním formuláři. Volitelný parametr „disableLinks“ (true/false) určuje, zda se má u ovládacích prvků „ForeignKey“ deaktivovat odkaz na vybraný záznam.*

Příklad:

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

void controls_Enable()

- ***Popis:*** *Funkce povolí editaci všech ovládacích prvků (kromě buttonů) v editačním formuláři.*
- ***Příklad:***

```
controls_Enable();
```

void controls_Join(object[] array)

- ***Popis:*** *Funkce spojí ovládací prvky definované parametrem „array“ do jedné skupiny, která zobrazuje jen ovládací prvky s vyplněnou hodnotu, plus jeden další.*
- ***Příklad:***

```
controls_Join([#ng_ovladaciprvek1#, #ng_ovladaciprvek2#, #ng_ovladaciprvek3#]);
controls_Join(new Array(#ng_ovladaciprvek1#, #ng_ovladaciprvek2#, #ng_ovladaciprvek3#));
```

void controls_Save(object[] array)

- ***Popis:*** *Funkce uloží hodnoty databázových ovládacích prvků do databáze pomocí ajaxu. Při uložení se nevykonávají události „OnBeforeSave“ ani „OnAfterSave“. Parametr „array” určuje seznam databázových ovládacích prvků, jejichž hodnoty budou do databáze uloženy.*
- ***Příklad:***

```
controls_Save([#ng_ovladaciprvek1#, #ng_ovladaciprvek2#]);
controls_Save(new Array(#ng_ovladaciprvek1#, #ng_ovladaciprvek2#));
```

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

- ***Popis:*** *Funkce uloží hodnoty databázových ovládacích prvků do databáze pomocí ajaxu. Při uložení se nevykonávají události „OnBeforeSave“ ani „OnAfterSave“. Parametr „array” určuje seznam databázových ovládacích prvků, jejichž hodnoty budou do databáze uloženy. Parametr „f“ udává název funkce, která se má spustit po načtení odpovědi ze serveru. Tato odpověď je buď „OK“ (záznam se podařilo uložit) nebo „ERROR“ (záznam se nepodařilo uložit). Parametr „errorMessage“ definuje text chybového hlášení, který se zobrazí v případě, že se při ukládání vyskytne chyba.*

- ***Příklad:***

```
controls_Save([#ng_ovladaciprvek1#, #ng_ovladaciprvek2#], ajaxResponse, 'Vyskytla se chyba');

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

string cookie_Get(string name)

- ***Popis:*** *Funkce vrací obsah cookie zadané parametrem „name“.*

- ***Příklad:***

```
var s = cookie_Get('loginname'); // Vrací hodnotu cookie s názvem „loginname“
```

void cookie_Set(string name, string value)

- ***Popis:*** *Funkce nastaví obsah cookie zadané parametrem „name“ na hodnotu zadanou parametrem „value“.*

- ***Příklad:***

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

bool copyToClipboard(string value, string info)

- ***Popis:*** *Funkce zkopíruje hodnotu zadanou parametrem „value“ do clipboardu. Parametr „info“ definuje text informativního hlášení, které bude zobrazeno ihned po úspěšném vložení hodnoty do clipboardu. Pokud vložení nebude úspěšné, funkce vrací hodnotu „false“.*

Příklad:

```
copyToClipboard('Hodnota, která bude vložena do schránky', 'Hodnota byla úspěšně vložena do schránky.');
```

string Date2Str(object date)

- ***Popis:*** *Funkce vrací textový řetězec ve formátu „dd.MM.rrrr HH:mm“ z data zadaného parametrem „date“. Pokud je čas 00:00, funkce vrací pouze řetězec ve formátu „dd.MM.rrrr“.*

Příklad:

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

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

- ***Popis:*** *Funkce nahradí texty položek rozbalovacího seznamu zadaného parametrem „ddl“ hodnotami definovanými v poli „array“. Pole musí obsahovat hodnotu, která je ukládána do databáze, a zástupný text, který se má zobrazit v rozbalovacím seznamu místo hodnoty. Poslední parametr „separator“ je nepovinný, a funguje jako oddělovač mezi hodnotou a jejím zástupným textem. Nahrazení textů pomocí této funkce nemá vliv na hodnotu, která je ukládána do databáze.*

Příklad *(vyžaduje ovládací prvek ComboBox s položkami 1, 2 a 3):*

```
var array = [];
array.push(['1', 'První položka']);
array.push(['2', 'Druhá položka']);
array.push(['3', 'Třetí položka']);
ddl_ReplaceTexts(#ng_combobox#, array);
// ddl_ReplaceTexts(#ng_combobox#, array, ' - ');
```

void ddl_SortByText(object ddl)

- ***Popis:*** *Funkce seřadí abecedně položky rozbalovacího seznamu „ddl“ podle textů jednotlivých položek.*
- ***Příklad:***

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

void ddl_SortByText2(object ddl)

- ***Popis:*** *Funkce seřadí numericky položky rozbalovacího seznamu „ddl“ podle textů jednotlivých položek.*
- ***Příklad:***

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

void ddl_SortByValue(object ddl)

- ***Popis:*** *Funkce seřadí abecedně položky rozbalovacího seznamu „ddl“ podle hodnot jednotlivých položek.*
- ***Příklad:***

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

void ddl_SortByValue2(object ddl)

- ***Popis:*** *Funkce seřadí numericky položky rozbalovacího seznamu „ddl“ podle hodnot jednotlivých položek.*
- ***Příklad:***

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

string ddl_SelectedItem_Text(object ddl)

- ***Popis:*** *Funkce vrací text vybrané položky v rozbalovacím seznamu zadaným parametrem „ddl“.*
- ***Příklad:***

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

string ddl_SelectedItem_Value(object ddl)

- ***Popis:*** *Funkce vrací hodnotu vybrané položky v rozbalovacím seznamu zadaným parametrem „ddl“.*
- ***Příklad:***

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

bool ddl_SelectText(object ddl, string text)

- ***Popis:*** *Funkce vyhledá položku v rozbalovacím seznamu zadaným parametrem „ddl“ podle jejího textu (parametr „text“), a pokud ji nalezne, vybere jí, a zavolá událost „onchange“. Pokud hodnota nebyla nalezena v seznamu hodnot ovládacího prvku, funkce vrací „false“.*
- ***Příklad:***

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

bool ddl_SelectText_NoChange(object ddl, string text)

- ***Popis:*** *Funkce vyhledá položku v rozbalovacím seznamu zadaným parametrem „ddl“ podle jejího textu (parametr „text“), a pokud ji nalezne, vybere jí. Funkce z události „onchange“ nejsou volány. Pokud hodnota nebyla nalezena v seznamu hodnot ovládacího prvku, funkce vrací „false“.*
- ***Příklad:***

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

bool ddl_SelectValue(object ddl, string text)

- ***Popis:*** *Funkce vyhledá položku v rozbalovacím seznamu zadaným parametrem „ddl“ podle její hodnoty (parametr „text“), a pokud ji nalezne, vybere jí, a zavolá událost „onchange“. Pokud hodnota nebyla nalezena v seznamu hodnot ovládacího prvku, funkce vrací „false“.*
- ***Příklad:***

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

bool ddl_SelectValue_NoChange(object ddl, string text)

- ***Popis:*** *Funkce vyhledá položku v rozbalovacím seznamu zadaným parametrem „ddl“ podle její hodnoty (parametr „text“), a pokud ji nalezne, vybere jí. Funkce z události „onchange“ nejsou volány. Pokud hodnota nebyla nalezena v seznamu hodnot ovládacího prvku, funkce vrací „false“.*
- ***Příklad:***

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

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

- ***Popis:*** *Funkce odpojí funkci „listener“ od události objektu „o“. „Type“ definuje typ události, např. load, scroll apod.*
- ***Příklad:***

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

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

void dg_Controls(object dg)

- ***Popis:*** *Funkce vrací pole všech editovatelných ovládacích prvků uvnitř datagridu „dg“.*
- ***Příklad:***

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

void dg_Disable()

- ***Popis:*** *Funkce zakáže editaci všech ovládacích prvků určených pro hromadnou editaci, které se nacházejí uvnitř všech datagridů.*
- ***Příklad:***

```
dg_Disable();
```

void dg_Disable(object dg)

- ***Popis:*** *Funkce zakáže editaci všech ovládacích prvků určených pro hromadnou editaci, které se nacházejí uvnitř konkrétního datagridu.*
- ***Příklad:***

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

void dg_Enable()

- ***Popis:*** *Funkce povolí editaci všech ovládacích prvků určených pro hromadnou editaci, které se nacházejí uvnitř všech datagridů.*
- ***Příklad:***

```
dg_Enable();
```

void dg_Enable(object dg)

- ***Popis:*** *Funkce povolí editaci všech ovládacích prvků určených pro hromadnou editaci, které se nacházejí uvnitř konkrétního datagridu.*
- ***Příklad:***

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

bool dg_HideColumn(object dg, int cid)

- ***Popis:*** *Funkce skryje sloupec zadaný parametrem „cid“ v datagridu definovaném parametrem „dg“. ID sloupce je stejné jako ID ovládacího prvku, kterým je sloupec definován (např. ID TextBoxu).*
- ***Příklad:***

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

void dg_HideHeader(object dg)

- ***Popis:*** *Funkce skryje nadpis datagridu definovaného parametrem „dg“. Parametr „dg“ může být i samotný nadpis datagridu.*

Příklad:

```
dg_HideHeader(DG123);
dg_HideHeader('Nadpis tabulky');
```

void dg_SearchColumns(object dg, array cids)

- ***Popis:*** *Funkce zaškrtne vyhledávání sloupců definovaných parametrem „cids“ v datagridu definovaném parametrem „dg“. Sloupce určené k vyhledávání se definují jako pole identifikátorů, kde ID sloupce je stejné jako ID ovládacího prvku, kterým je sloupec definován (např. ID TextBoxu).*

Příklad:

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

bool dg_ShowColumn(object dg, int cid)

- ***Popis:*** *Funkce zobrazí sloupec (který byl skrytý) definovaný parametrem „cid“ v datagridu definovaném parametrem „dg“. ID sloupce je stejné jako ID ovládacího prvku, kterým je sloupec definován (např. ID TextBoxu).*

Příklad:

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

void dg_VisibleColumns(object dg, array cids)

- ***Popis:*** *Funkce zobrazí sloupce definované parametrem „cids“ v datagridu definovaném parametrem „dg“. Sloupce určené k zobrazení se definují jako pole identifikátorů, kde ID sloupce je stejné jako ID ovládacího prvku, kterým je sloupec definován (např. ID TextBoxu).*

Příklad:

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

void dp_Disable(object dp)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „DatePicker" zadaného parametrem „dp". Alternativou je obecná funkce „control_Disable".*
- ***Příklad:***

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

void dp_Enable(object dp)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „DatePicker" zadaného parametrem „dp". Alternativou je obecná funkce „control_Enable".*
- ***Příklad:***

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

object el(string id)

- ***Popis:*** *Funkce vrací objekt na webové stránce podle jeho ID.*
- ***Příklad:***

```
var bt = el(BT1665);
// Vrací objekt tlačítka s databázovým ID 1665
```

void evalGoogleCaptcha()

- ***Popis:*** *Funkce spouští validaci služby Google reCAPTCHA, která rozpoznává uživatele od robotů.*
- ***Příklad:***

```
evalGoogleCaptcha();
```

void file_Disable(object file)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „File“ zadaného parametrem „file“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void file_Enable(object file)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „File“ zadaného parametrem „file“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

void fireEvent2(object o, string type)

- ***Popis:*** *Funkce vyvolá událost na objektu „o“. Parametr „type“ je typ události (např. blur, click, onchange, keyup, mouseout, mouseover…).*
- ***Příklad:***

```
fireEvent2(#ng_ovladaciprvek#, 'change');
// Vyvolá událost „onchange“ na prvku „ng_ovladaciprvek“
```

void fk_Disable(object fk, bool disableLink)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „ForeignKey“ zadaného parametrem „fk“. Volitelný parametr „disableLink“ (true/false) určuje, zda se má deaktivovat odkaz na vybraný záznam. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void fk_Enable(object fk)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „ForeignKey“ zadaného parametrem „fk“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

void fk_Update(object fk)

- ***Popis:*** *Funkce aktualizuje ajaxem hodnotu ovládacího prvku „ForeignKey“ zadaného parametrem „fk“. Funkce má význam pouze v případě, že je nastavena nulová šířka ovládacího prvku, a byla použita funkce „control_SetValue“ pro nastavení nové hodnoty ovládacího prvku.*
- ***Příklad:***

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

void form_DisableUpdate()

- ***Popis:*** *Funkce zamezí automatické aktualizaci editačního formuláře po zkopírování souborových příloh nebo obrázků pomocí pickeru, případně po zkopírování hodnot do rozbalovacích seznamů nebo radio buttonů, které kopírovanou hodnotu dosud neobsahují.*
- ***Příklad:***

```
form_DisableUpdate();
```

void form_Change()

- ***Popis:*** *Funkce je zavolána pokaždé, kdy dojde ke změně hodnoty ovládacího prvku v editačním formuláři, a zajistí, aby se při pokusu o opuštění editačního formuláře uživateli zobrazilo hlášení, zda chce formulář skutečně opustit bez předchozího uložení provedených změn.*

Příklad č. 1:

```
form_Change();
```

Příklad č. 2:

```
var saveObject = new Object();

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

bool form_Picker()

- ***Popis:*** *Funkce vrací logickou hodnotu, je-li formulář ve stavu volání javascriptových událostí „onchange“ u ovládacích prvků, jejichž hodnoty byly právě převzaty pomocí pickeru.*

Příklad:

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

void form_Save()

- ***Popis:*** *Funkce uloží právě otevřený záznam v editačním formuláři pomocí ajaxu. Při uložení se nevykonávají události „OnBeforeSave“ ani „OnAfterSave“.*

Příklad:

```
form_Save();
```

void form_Save(string f)

- ***Popis:*** *Funkce uloží právě otevřený záznam v editačním formuláři pomocí ajaxu. Při uložení se nevykonávají události „OnBeforeSave“ ani „OnAfterSave“. Parametr „f“ udává název funkce, která se má spustit po načtení odpovědi ze serveru. Tato odpověď je buď „OK“ (záznam se podařilo uložit) nebo „ERROR“ (záznam se nepodařilo uložit).*

Příklad:

```
form_Save(ajaxResponse);

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

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

- ***Popis:*** *Funkce nastaví okraje webové stránky.*

Příklad:

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

void form_Update()

- ***Popis:*** *Funkce aktualizuje právě otevřený editační formulář, především z důvodu aktualizace dat v datagridech nebo zdrojích dat jednotlivých ovládacích prvků v editačním formuláři.*

Příklad:

```
form_Update();
```

string Hex(int n)

- ***Popis:*** *Funkce převede decimální tvar čísla „n“ na hexadecimální.*

Příklad:

```
var s = Hex(123);
// Vrací „7B“
```

string htmlEncode(string value)

- ***Popis:*** *Funkce nahradí všechny HTML znaky obsažené v parametru „value“ za odpovídající sekvenci znaků (character entities).*

Příklad:

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

void chb_Disable(object chb)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „CheckBox“ zadaného parametrem „chb“. Alternativou je obecná funkce „control_Disable“.*

Příklad:

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

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

- ***Popis:*** *Funkce skryje/zobrazí element zadaný parametrem „o“ podle toho, zda je checkbox definovaný parametrem „chb“ zaškrtnutý či nikoliv. Parametr „reverse“ určuje „true“ nebo „false“ podle toho, zda se má funkce chovat inverzně. Funkce má využití převážně v události „OnClick“ u checkboxu.*

Příklad:

```
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)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „CheckBox“ zadaného parametrem „chb“. Alternativou je obecná funkce „control_Enable“.*

Příklad:

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

void image_Disable(object image)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „Image“ zadaného parametrem „image“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void image_Enable(object image)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „Image“ zadaného parametrem „image“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

int innerHeight2()

- ***Popis:*** *Funkce vrací vnitřní výšku okna objektu window.*
- ***Příklad:***

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

int innerWidth2()

- ***Popis:*** *Funkce vrací vnitřní šířku okna objektu window.*
- ***Příklad:***

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

void InsertTime(object tb)

- ***Popis:*** *Funkce vloží aktuální čas ve formátu „HH:mm“ do textového pole definovaného parametrem „tb“.*
- ***Příklad:***

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

string Int2Time(int minutes)

- ***Popis:*** *Funkce vrací textový řetězec ve formátu „HH:mm“ vyjádřený z počtu minut zadaného parametrem „minutes“.*
- ***Příklad:***

```
var s = Int2Time(123);
// Vrací „02:03“
var s = Int2Time(-123);
// Vrací „-02:03“
```

bool isImage(string filename)

- ***Popis:*** *Funkce vrací „true“ nebo „false“ podle přípony souboru, která identifikuje obrázek.*
- ***Příklad:***

```
var ok = isImage('filename.txt'); // Vrací „false“
```

string jsCompanyWithoutSro(string company)

- ***Popis:*** *Funkce odstraní koncovku typu společnosti (např. s.r.o., a.s., k.s., v.o.s., spol., a další) z názvu firmy zadaného parametrem „company“.*
- ***Příklad:***

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

string jsDisplay(bool value)

- ***Popis:*** *Funkce vrací '' (prvek bude viditelný) nebo 'none' (prvek nebude viditelný) podle obsahu parametru „value“. Funkce má využití pro nastavení atributu „style.display“ jednotlivých ovládacích prvků editačního formuláře.*

Příklad:

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

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

- ***Popis:*** *Funkce formátuje číslo zadané parametrem „number“ do textové podoby, která je vyžadovaná pro správné uložení číselné hodnoty ovládacího prvku do databáze – funkce zajistí použití správného oddělovače desetinných míst.*

Příklad:

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

string jsFormatCurrency(double number, string symbol)

- ***Popis:*** *Funkce formátuje číslo zadané parametrem „number“ do textové podoby – s merezou jako oddělovačem tisíců, s desetinnou čárkou nebo tečkou v závislosti na jazykovém nastavení aktuálně přihlášeného uživatele, zaokrouhlené na dvě desetinná místa, a se symbolem zadaným parametrem „symbol“.*

Příklad:

```
var s = jsFormatCurrency(1100, 'Kč');
// Vrací „1 100,00 Kč“
var s = jsFormatCurrency(1100.5, 'Kč');
// Vrací „1 100,50 Kč“
var s = jsFormatCurrency(1100.5412, 'Kč');
// Vrací „1 100,54 Kč“
```

string jsFormatDate(object date)

Popis: *Funkce formátuje datum zadané parametrem „date“ do textové podoby na základě formátu data z nastavení NET Genia a jazykového nastavení aktuálně přihlášeného uživatele:*

- „dd/mm/yyyy“ nebo
- „mm/dd/yyyy“.

Pokud je jazyk uživatele angličtina:

- „dd/mm/yyyy“ nebo
- „mm/dd/yyyy“.

Pokud je jazyk uživatele jiný jazyk než angličtina:

- „dd.mm.yyyy“ nebo
- „mm.dd.yyyy“.

Příklad:

```
var s = jsFormatDate(new Date()));
// Vrací aktuální datum ve formátu „dd.mm.yyyy“ (jazyk uživatele je jiný než angličtina)
var s = jsFormatDate(new Date()));
// Vrací aktuální datum ve formátu „dd/mm/yyyy“ (jazyk uživatele je angličtina)
var s = jsFormatDate(Str2Date('01.01.2016 13:00')));
// Vrací „01.01.2016“ (jazyk uživatele je jiný než angličtina)
var s = jsFormatDate(Str2Date('01/01/2016 13:00'));
// Vrací „01/01/2016“ (jazyk uživatele je angličtina)
```

string jsFormatDouble(double number, string symbol)

Popis: *Funkce formátuje číslo zadané parametrem „number“ do textové podoby – s merezou jako oddělovačem tisíců, s desetinnou čárkou nebo tečkou v závislosti na jazykovém nastavení aktuálně přihlášeného uživatele, a se symbolem zadaným parametrem „symbol“.*

Příklad:

```
var s = jsFormatDouble(1100, 'kg'));
// Vrací „1 100 kg“
var s = jsFormatDouble(1100.5, 'kg'));
// Vrací „1 100,5 kg“
var s = jsFormatDouble(1100.5412, 'kg'));
// Vrací „1 100,5412 kg“
```

string jsFormatSize(int size)

Popis: *Funkce formátuje velikost souboru zadanou parametrem „size“ do textové podoby.*

Příklad:

```
var s = jsFormatSize(1);
// Vrací „1 b“
var s = jsFormatSize(1024);
// Vrací „1 kB“
var s = jsFormatSize(102456407);
// Vrací „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)

Popis: *Funkce vrací HTML kód tabulky formátované ve vzhledu datagridu. Parametr „value“ definuje pole hodnot tabulky, parametr „title“ definuje nadpis tabulky, parametr „width“ definuje šířku zobrazeného datagridu. Parametr „showGrid“, který může být „true“ nebo „false“ určuje, zda se má v tabulce zobrazit mřížka. Parametr „noEntries“ definuje, jaký text se má zobrazit v případě, že tabulka neobsahuje žádné záznamy. Parametr „pageSize“ definuje, kolik záznamů má být obsaženo na jedné stránce tabulky, a parametr „tb_pageIndex“ definuje ovládací prvek, do kterého je uloženo číslo aktuální stránky.*

Příklad:

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

<script type="text/javascript">

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

el('polozky').innerHTML = jsFormatTable(array, 'Uživatelé', 400, true, 'Nebyly nalezeny žádné
záznamy');
// el('polozky').innerHTML = jsFormatTable(array, 'Uživatelé', 400, true, 'Nebyly nalezeny
žádné záznamy', 1, #ng_textbox#);

</script>
```

string jsFormatTime(object date)

Popis: *Funkce formátuje čas zadaný parametrem „date“ do textové podoby ve formátu „HH:mm“.*

Příklad:

```
var s = jsFormatTime(new Date());
// Vrací aktuální čas ve formátu „HH:mm“
var s = jsFormatTime(Str2Date('1.1.2016 13:00'));
// Vrací „13:00“
var s = jsFormatTime(Str2Date('01/01/2016 13:00'));
// Vrací „13:00“
```

string jsFormatTime2(object date)

- ***Popis:*** *Funkce formátuje čas zadaný parametrem „date“ do textové podoby ve formátu „HH:mm:ss“.*
- ***Příklad:***

```
var s = jsFormatTime(new Date());
// Vrací aktuální čas ve formátu „HH:mm:ss“
var s = jsFormatTime(Str2Date('1.1.2016 13:00'));
// Vrací „13:00:00“
var s = jsFormatTime(Str2Date('01/01/2016 13:00'));
// Vrací „13:00:00“
```

string jsFormatTime3(object date)

- ***Popis:*** *Funkce formátuje čas zadaný parametrem „date“ do textové podoby ve formátu „HH:mm:ss.SSS“.*
- ***Příklad:***

```
var s = jsFormatTime(new Date());
// Vrací aktuální čas ve formátu „HH:mm:ss:SSS“
var s = jsFormatTime(Str2Date('1.1.2016 13:00'));
// Vrací „13:00:00.000“
var s = jsFormatTime(Str2Date('01/01/2016 13:00'));
// Vrací „13:00:00.000“
```

bool jsIsNumber(object o)

- ***Popis:*** *Funkce vrací logickou hodnotu, je-li hodnota zadaná parametrem „o“ číslo či nikoliv.*
- ***Příklad:***

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

double jsMeasureHours(object startdate, object enddate)

- ***Popis:*** *Funkce vrací nezaokrouhlený počet hodin z časového úseku zadaného parametry „startdate“ a „enddate“. Oba parametry musí být javascriptové objekty typu „Date“.*
- ***Příklad:***

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

double jsMeasureMinutes(object startdate, object enddate)

- ***Popis:*** *Funkce vrací nezaokrouhlený počet minut z časového úseku zadaného parametry „startdate“ a „enddate“. Oba parametry musí být javascriptové objekty typu „Date“.*
- ***Příklad:***

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

double jsMeasureWorkDays(object startdate, object enddate)

- ***Popis:*** *Funkce vrací počet celých pracovních dnů z časového úseku zadaného parametry „startdate“ a „enddate“.*
  Oba parametry musí být javascriptové objekty typu „Date“. Funkce zohledňuje státní svátky evidované v základní aplikaci „Nastavení“.
- ***Příklad:***

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

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

- ***Popis:*** *Funkce vrací počet celých pracovních dnů z časového úseku zadaného parametry „startdate“ a „enddate“. Parametr „workinghours“ určuje počet hodin běžného pracovního dne. Funkce vrací desetinné číslo – poměrnou část pracovního dne – pokud je „startdate“ a „enddate“ ve stejný den, a pokud je délka časového úseku v hodinách kratší než počet hodin běžného pracovního dne.*
- ***Příklad:***

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

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

- ***Popis:*** *Funkce převede objekt „o“ na číslo.*
- ***Příklad:***

```
var n = jsN(114.12); // Vrací „114.12“
var n = jsN(114,12); // Vrací „114.12“
```

string jsNetGeniumStart(string date)

- ***Popis:*** *Funkce vrací čas prvního spuštění NET Genia aktuálně přihlášeným uživatelem ve dni zadaném textovým parametrem „date“ ve formátu „dd.MM.rrrr“. Informaci o čase spuštění NET Genium ukládá do cookies.*
- ***Příklad:***

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

string jsngdph()

- ***Popis:*** *Funkce vrací aktuální sazby DPH oddělené středníkem.*
- ***Příklad:***

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

double jsngdph(int rate)

- ***Popis:*** *Funkce vrací sazbu DPH definovanou parametrem „rate“. Parametr je definován číslem 2 pro zvýšenou sazbu DPH, číslem 1 pro sníženou sazbu DPH a číslem 0 pro nulovou sazbu DPH.*
- ***Příklad:***

```
var n = jsngdph(1);
// Vrací aktuální sníženou sazbu DPH, např. „15“
```

string jsngdph(object date)

- ***Popis:*** *Funkce vrací sazby DPH oddělené středníkem ke dni uvedeném v parametru „date“.*
- ***Příklad:***

```
var s = jsngdph(Str2Date('1.1.2000'));
// Vrací sazby DPH oddělené středníkem platné k danému dni, např. „22;5;0“.
```

double jsngdph(int rate, object date)

- ***Popis:*** *Vrací sazbu DPH definovanou parametrem „rate“. Parametr je definován číslem 2 pro zvýšenou sazbu DPH, číslem 1 pro sníženou sazbu DPH a číslem 0 pro nulovou sazbu DPH. Sazba je určena ke dni uvedeném v parametru „date“.*

Příklad:

```
var n = jsngdph(2, Str2Date('1.1.2000'));
// Vrací zvýšenou sazbu DPH k danému dni, např. „22“
```

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

- ***Popis:*** *Funkce spočítá v závislosti na parametru „type2“ buď částku, nebo hodnotu DPH. Pro výpočet je použita základní částka definovaná parametrem „base“ a sazba DPH definovaná parametrem „rate“. Parametr „type1“ specifikuje blíže hodnotu parametru „base“.*

  Možné hodnoty parametru „type1“:

  - bezdph – parametr 1 je suma bez DPH
  - sdph – parametr 1 je suma s DPH

  Možné hodnoty parametru „type2“:

  - bezdph – vypočítá základ daně
  - dph – vypočítá hodnotu DPH
  - zdph – vypočítá hodnotu DPH zaokrouhlenou na celé číslo
  - dph1 – vypočítá hodnotu DPH snížené sazby
  - zdph1 – vypočítá hodnotu DPH snížené sazby zaokrouhlenou na celé číslo
  - dph2 – vypočítá hodnotu DPH zvýšené sazby
  - zdph2 – vypočítá hodnotu DPH zvýšené sazby zaokrouhlenou na celé číslo
  - sdph – vypočítá částku s DPH
  - zsdph – vypočítá částku s DPH zaokrouhlenou nahoru na celé číslo
  - zsdph-1 – vypočítá částku s DPH zaokrouhlenou matematicky na celé číslo
  - zsdph-2 – vypočítá částku s DPH zaokrouhlenou na 50 haléřů

Příklad:

```
var n = jsngdph(-0.55, 'sdph', 19, 'dph');
// Vrací „-0,99“
```

int jsngvatindex(double rate)

- ***Popis:*** *Funkce vrací index aktuální sazby DPH zadané parametrem „rate". Funkce vrací hodnotu „0" v případě nulové sazby DPH, hodnotu „1" v případě nižší sazby DPH, hodnotu „2" v případě vyšší sazby DPH a hodnotu „3" v případě druhé snížené sazby DPH.*

Příklad:

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

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

- ***Popis:*** *Funkce vrací index sazby DPH zadané parametrem „rate" v zemi zadané parametrem „country" (CZ, DE, HU, SK, …) a ke dni zadaném parametrem „date". Funkce vrací hodnotu „0" v případě nulové sazby DPH, hodnotu „1" v případě nižší sazby DPH, hodnotu „2" v případě vyšší sazby DPH a hodnotu „3" v případě druhé snížené sazby DPH.*

Příklad:

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

string jsngvatrates(string country, object date)

- ***Popis:*** *Funkce vrací sazby DPH oddělené středníkem v dané zemi (CZ, DE, HU, SK,…) zadané parametrem „country" k danému datu zadaném parametrem „date".*

Příklad:

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

string jsNumberInWords(double number, string language)

Popis: *Funkce vrací slovní vyjádření číselné hodnoty zadané parametrem „number“. Parametr „language“ může obsahovat jednu z hodnot „cs“, „de“, „en“, „fr“ nebo „sk“. Funkce zaokrouhluje na celá čísla.*

Příklad:

```
var s = jsNumberInWords(12345.678, 'cs');
// Vrací „dvanácttisíctřistačtyřicetšest“
```

object jsO(object o)

Popis: *Funkce převede objekt „o“ na datový typ objekt. Funkce má využití při testování práv uživatele k příslušnému ovládacímu prvku.*

Příklad správného použití:

```
var o = jsO(#ng_tb#);
// Pokud uživatel nemá na ovládací prvek „#ng_tb#“ právo, dojde k nahrazení „#ng_tb#“ za hodnotu uloženou v tomto prvku v databázi
if (o != null)
{
}
```

Příklad chybného použití:

```
var o = #ng_tb#;
// Pokud uživatel nemá na ovládací prvek „#ng_tb#“ právo, může dojít k nahrazení „#ng_tb#“ za prázdnou hodnotu. Výsledek pak bude syntaktická chyba „var o = ;“
```

void jsOpen(string url)

Popis: *Funkce otevře webovou stránku zadanou parametrem „url“ v novém okně webového prohlížeče.*

Příklad:

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

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

- ***Popis:*** *Funkce vrací zaokrouhlenou hodnotu čísla zadaného parametrem „number“.*
- ***Příklad:***

```
var n = jsR(21.232);
// Vrací „21“
```

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

- ***Popis:*** *Funkce vrací číslo zadané parametrem „number“ zaokrouhlené na počet desetinných míst zadaných parametrem „decimals“.*
- ***Příklad:***

```
var n = jsR(2.354, 2);
// Vrací „2.35“
```

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

- ***Popis:*** *Funkce vrací číslo zadané parametrem „number“ zaokrouhlené na padesátihaléře.*
- ***Příklad:***

```
var n = jsR_50(123.456);
// Vrací „123,5“
```

bool jsUserInGroup(string name)

- ***Popis:*** *Funkce vrací logickou hodnotu, je-li aktuálně přihlášený uživatel členem uživatelské skupiny zadané parametrem „name“ či nikoliv. „Name“ může být id uživatelské skupiny nebo její jméno.*
- ***Příklad:***

```
alert('Uživatel ' + (jsUserInGroup(2) ? 'je' : 'není') + ' členem skupiny <b>Users</b>');
```

```
alert('Uživatel ' + (jsUserInGroup('Users') ? 'je' : 'není') + ' členem skupiny
<b>Users</b>');
```

bool jsValidateEmail(string value)

- ***Popis:*** *Funkce přijímá prostřednictvím parametru „value" e-mailovou adresu a ověřuje, jestli je adresa validní (ve správném formátu pro e-mail). Poté vrací „true" nebo „false".*
- ***Příklad:***

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

bool jsValidateEmails(string value)

- ***Popis:*** *Funkce přijímá prostřednictvím parametru „value" e-mailové adresy oddělené čárkou nebo středníkem*
  a ověřuje, jestli jsou zadané adresy validní (ve správném formátu pro e-mail). Poté vrátí „true" nebo „false".
  V hodnotě „value" se mohou nacházet mezery mezi jednotlivými e-maily.
- ***Příklad:***

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

string jsVisibility(bool value)

- ***Popis:*** *Funkce vrací „visible" (prvek bude viditelný) nebo „hidden" (prvek nebude viditelný) podle obsahu parametru „value". Funkce má využití pro nastavení atributu „style.visibility" jednotlivých ovládacích prvků editačního formuláře.*
- ***Příklad:***

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

bool jsWait(string url, string pleaseWait)

- ***Popis:*** *Funkce skryje obsah webové stránky, zobrazí hlášku definovanou parametrem „pleaseWait“, a přesměruje uživatele na novou webovou stránku s adresou zadanou parametrem „url“. Funkce vrací „false“.*
- ***Příklad:***

```
jsWait('https://www.netgenium.com', 'Čekejte prosím…');
```

void lb_Disable(object lb)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „ListBox“ zadaného parametrem „lb“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void lb_Enable(object lb)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „ListBox“ zadaného parametrem „lb“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

bool letterOrDigit(string value)

- ***Popis:*** *Funkce vrací hodnotu „true“, pokud text zadaný parametrem „value“ začíná písmenem nebo číslicí. Funkce vrací hodnotu „false“, pokud text zadaný parametrem „value“ začíná jiným znakem, než je písmeno nebo číslice.*
- ***Příklad:***

```
letterOrDigit('NET Genium'); // Vrací „true“
letterOrDigit('-a**'); // Vrací „false“
```

void loadUrl(string url)

- ***Popis:*** *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. V jednu chvíli je možné volat pouze jedno asynchronní volání pomocí funkce „loadUrl“. Pokud by došlo k opětovnému zavolání funkce „loadUrl“ v případě, že předchozí asynchronní volání dosud nebylo nedokončeno, k novému asynchronnímu volání nedojde. Pro potřeby paralelního volání je určena funkce „_loadUrl“.*

Příklad:

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

void loadUrl(string url, string f)

- ***Popis:*** *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. callbacku definovaného parametrem „f“. Výsledný HTML kód stránky je funkci „f“ předán jako první parametr. V jednu chvíli je možné volat pouze jedno asynchronní volání pomocí funkce „loadUrl“. Pokud by došlo k opětovnému zavolání funkce „loadUrl“ v případě, že předchozí asynchronní volání dosud nebylo nedokončeno, k novému asynchronnímu volání nedojde. Pro potřeby paralelního volání je určena funkce „_loadUrl“.*

Příklad:

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

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

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

- ***Popis:*** *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. callbacku definovaného parametrem „f“. Výsledný HTML kód stránky je funkci „f“ předán jako první parametr. Volitelný parametr „id“ je funkci „f“ předán jako druhý parametr. V jednu chvíli je možné volat pouze jedno asynchronní volání pomocí funkce „loadUrl“. Pokud by došlo k opětovnému zavolání funkce „loadUrl“ v případě, že předchozí asynchronní volání dosud nebylo nedokončeno, k novému asynchronnímu volání nedojde. Pro potřeby paralelního volání je určena funkce „_loadUrl“.*

- ***Příklad:***

```
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)

- ***Popis:*** *Funkce načte stránku definovanou parametrem „url“ pomocí asynchronního ajaxového volání, resp. načte HTML kód této stránky. Funkce nečeká bezprostředně na odpověď serveru, po jejím zavolání pokračuje provádění javascriptového kódu. Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. callbacku definovaného parametrem „f“. Výsledný HTML kód stránky je funkci „f“ předán jako první parametr. Volitelný parametr „id“ je funkci „f“ předán jako druhý parametr. Parametr „params“ je spolu s asynchronním voláním odeslán metodou POST, a jde o seznam párů klíč vs. hodnota oddělených pomocí znaku „&“. V jednu chvíli je možné volat pouze jedno asynchronní volání pomocí funkce „loadUrl“. Pokud by došlo k opětovnému zavolání funkce „loadUrl“ v případě, že předchozí asynchronní volání dosud nebylo nedokončeno, k novému asynchronnímu volání nedojde. Pro potřeby paralelního volání je určena funkce „_loadUrl“.*

- ***Příklad:***

```
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()

- ***Popis:*** *Funkce vyvolá přihlášení do účtu Facebooku na nové záložce webového prohlížeče. Úspěšné přihlášení do Facebooku nastaví hodnotu proměnné „#facebook_access_token#“. Funkce vrací „false“, pokud je uživatel do Facebooku již přihlášen.*

Příklad:

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

bool loginByGoogle(string scope, bool offline)

- ***Popis:*** *Funkce vyvolá přihlášení do účtu Google na nové záložce webového prohlížeče, a předá hodnotu parametru „scope“, který definuje rozsah přístupových oprávnění. Úspěšné přihlášení do Google nastaví hodnotu proměnné „#google_access_token#“. Parametr „offline“ určuje, zda má úspěšné přihlášení nastavit také hodnotu proměnné „#google_refresh_token#“. Funkce vrací „false“, pokud je uživatel do Google již přihlášen.*

Příklad:

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

bool loginByMicrosoft(string scope, bool offline)

- ***Popis:*** *Funkce vyvolá přihlášení do účtu Microsoftu na nové záložce webového prohlížeče, a předá hodnotu parametru „scope”, který definuje rozsah přístupových oprávnění. Úspěšné přihlášení do Microsoftu nastaví hodnotu proměnné „#microsoft_access_token#”. Parametr „offline” určuje, zda má úspěšné přihlášení nastavit také hodnotu proměnné „#microsoft_refresh_token#”. Funkce vrací „false”, pokud je uživatel do Microsoftu již přihlášen.*

Příklad:

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

void mlb_Disable(object mlb)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „MultiListBox“ zadaného parametrem „mlb“. Alternativou je obecná funkce „control_Disable“.*

Příklad:

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

void mlb_Enable(object mlb)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „MultiListBox“ zadaného parametrem „mlb“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

string Now()

- ***Popis:*** *Funkce vrací aktuální datum a čas ve formátu „dd.mm.yyyy hh:mm“.*
- ***Příklad:***

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

string Now2()

- ***Popis:*** *Funkce vrací aktuální datum a čas ve formátu „dd.mm.yyyy hh:mm:ss“.*
- ***Příklad:***

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

int offsetLeft2(object o)

- ***Popis:*** *Funkce vrací x-ovou souřadnici elementu „o“ od levého horního rohu webové stránky.*
- ***Příklad:***

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

int offsetTop2(object o)

- ***Popis:*** *Funkce vrací y-ovou souřadnici elementu „o“ od levého horního rohu webové stránky.*
- ***Příklad:***

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

void picker_Open(object o)

- ***Popis:*** *Funkce otevře picker ovládacího prvku zadaného parametrem „o“.*
- ***Příklad:***

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

string queryString(string value)

- ***Popis:*** *Funkce vrací parametr „value“ z adresy „window.location“.*
- ***Příklad:***

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

string queryString(string value, string url)

- ***Popis:*** *Funkce vrací parametr „value“ z adresy zadané parametrem „url“.*
- ***Příklad:***

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

void rbl_Disable(object rbl)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „Radio“ zadaného parametrem „rbl“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void rbl_Enable(object rbl)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „Radio“ zadaného parametrem „rbl“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

void rbl_EnabledItems(object rbl, array items)

- ***Popis:*** *Funkce povolí možnost výběru pouze konkrétních položek u ovládacího prvku „Radio“ zadaného parametrem „rbl“. Seznam položek je definován parametrem „items“.*
- ***Příklad:***

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

int rbl_SelectedIndex(object rbl)

- ***Popis:*** *Funkce vrací index vybrané položky ovládacího prvku „Radio“. V případě, že není vybrána žádná položka, vrací „-1“.*
- ***Příklad:***

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

string rbl_SelectedItem_Text(object rbl)

- ***Popis:*** *Funkce vrací text přiřazený vybrané položce v ovládacím prvku „Radio“.*
- ***Příklad:***

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

string rbl_SelectedItem_Value(object rbl)

- ***Popis:*** *Funkce vrací hodnotu přiřazenou vybrané položce v ovládacím prvku „Radio“. Pokud není vybrána žádná položka, funkce vrací prázdný string.*
- ***Příklad:***

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

bool rbl_SelectText(object rbl, string value)

- ***Popis:*** *Funkce vyhledá položku v ovládacím prvku „Radio“ podle jejího textu, a pokud ji nalezne, vybere jí. Pokud hledaná položka nebude nalezena, funkce vrací „false“.*
- ***Příklad:***

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

bool rbl_SelectValue(object rbl, string value)

- ***Popis:*** *Funkce vyhledá položku v ovládacím prvku „Radio“ podle její hodnoty, a pokud ji nalezne, vybere jí. Pokud hledaná položka nebude nalezena, funkce vrací „false“.*
- ***Příklad:***

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

string removeDiacritics(string value)

- ***Popis:*** *Funkce odebere diakritiku z řetězce zadaného parametrem „value“.*
- ***Příklad:***

```
var s = removeDiacritics('ěščřžýáíé'); // Vrací „escrzyaie“
```

string removeDoubleCharacters(string value, string ch)

Popis: *Funkce odstraní z řetězce zadaného parametrem „value“ zdvojené znaky definované parametrem „ch“.*

Příklad:

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

string removeDoubleLines(string value)

Popis: *Funkce odstraní z víceřádkového řetězce zadaného parametrem „value“ zdvojené prázdné řádky.*

Příklad:

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

string removeDoubleSpaces(string value)

Popis: *Funkce odstraní z řetězce zadaného parametrem „value“ zdvojené mezery.*

Příklad:

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

string removeDuplicates(string value, string separator)

Popis: *Funkce odebere všechny duplicitní položky z položek zadaných parametrem „value“. Oddělovač, který je použit při specifikování záznamů, je definován parametrem „separator“.*

Příklad:

```
var s = removeDuplicates('aa;bb;aa;cc', ';'); // Vrací „aa;bb;cc“
```

string removeNonAlphanumericCharacters(string value)

- ***Popis:*** *Funkce odstraní z řetězce zadaného parametrem „value“ všechny znaky kromě alfanumerických znaků.*
- ***Příklad:***

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

string removeNonDigits(string value)

- ***Popis:*** *Funkce odstraní z řetězce zadaného parametrem „value“ všechny znaky kromě číslic.*
- ***Příklad:***

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

string removeTags(string html)

- ***Popis:*** *Funkce odstraní z řetězce zadaného parametrem „html“ všechny tagy, a nahradí tagy „ “, „
“, „<hr>“, „</p>“ a „
“ za prázdné řádky pomocí podřetězce „\r\n“.*
- ***Příklad:***

```
var s = removeTags('<b>Ano</b>');
// Vrací „Ano“
```

string removeTags2(string html)

- ***Popis:*** *Funkce odstraní všechny tagy z řetězce zadaného parametrem „html“.*
- ***Příklad:***

```
var s = removeTags2('<p>abc</p><p>abc</p>');
// Vrací „abcabc“
```

string removeUnnecessarySpaces(string value)

- ***Popis:*** *Funkce odstraní z řetězce zadaného parametrem „value“ všechna nadbytečná prázdná místa – všechna prázdná místa na začátku řetězce a na konci řetězce, a dále zdvojené mezery uprostřed nahradí za jednoduché mezery.*
- ***Příklad:***

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

string removeWhiteSpaces(string value)

- ***Popis:*** *Funkce odstraní z řetězce zadaného parametrem „value“ všechna prázdná místa (mezery a tabulátory).*
- ***Příklad:***

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

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

- ***Popis:*** *Funkce vrací textový řetězec zadaný parametrem „value“, ve kterém byly nahrazeny všechny výskyty řetězce uloženého v parametru „oldvalue“ (vzor) řetězcem uloženým v parametru „newvalue“ (náhrada).*
- ***Příklad:***

```
var s = replaceAll('Lálá', 'á', 'a');
// Vrací „Lala“
```

void rtb_Disable(object rtb)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „RichTextBox“ zadaného parametrem „rtb“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void rtb_Enable(object rtb)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „RichTextBox“ zadaného parametrem „rtb“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

void rtb_Focus(object rtb)

- ***Popis:*** *Funkce nastaví focus do ovládacího prvku „RichTextBox“ definovaného parametrem „rtb“.*
- ***Příklad:***

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

void rtb_Insert(object rtb, string value)

- ***Popis:*** *Funkce vloží do ovládacího prvku „RichTextBox“ hodnotu zadanou parametrem „value“ na aktuální místo kurzoru.*
- ***Příklad:***

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

string script_Error(string value)

- ***Popis:*** *Tato javascriptová funkce je spuštěna vždy při vyvolání skriptového přerušení v editačním formuláři nebo na nahlížecí stránce. Do parametru „value" je předán text přerušení tak jak je nastaven v návrháři skriptů. Pro snadnou identifikaci typu přerušení je doporučeno do textu přerušení přidat například číselný kód, a tento kód se následně snažit v javascriptu vyhledat. Javascriptová funkce „script_Error" musí být v editačním formuláři nebo na nahlížecí stránce definována jako celek, tj. včetně „function script_Error(value)".*

Příklad:

```
Přerušení „Nastala chyba(123)"

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

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

- ***Popis:*** *Funkce naskroluje editační formulář nebo nahlížecí stránku na pozici definovanou parametry „x" a „y". Volitelný parametr „save" značí, že má ihned dojít k zapamatování této pozice, obvykle když funkci „scroll_To" volá tlačítko se serverovým skriptem.*

Příklad:

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

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

- ***Popis:*** *Funkce zavolá jinou funkci definovanou v parametru „f" po uplynutí časového úseku definovaného v parametru „timeout". Funkce se vždy váže na konkrétní objekt zadaný parametrem „o". Pokud během stanoveného časového úseku dojde k opětovnému zavolání funkce „setTimeout2" se stejnými parametry, dojde ke zrušení předchozího volání, a naplánování nového opět po uplynutí časového úseku definovaného v parametru „timeout". Hodnota parametru „timeout" je uváděna v milisekundách.*

Příklad:

```
var saveObject = new Object();

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

string StringBuilder()

- ***Popis:*** *„StringBuilder" představuje ekvivalent stejnojmenné třídy v C# určené k optimalizovanému sčítání dlouhých stringů.*

Příklad:

```
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)

- ***Popis:*** *Funkce vrací datum konvertované z textového řetězce zadaného parametrem „text" ve formátu „dd.MM.rrrr" nebo „dd.MM.rrrr HH:mm".*

Příklad:

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

void ta_Disable(object ta)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „TextArea“ zadaného parametrem „ta“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void ta_Enable(object ta)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „TextArea“ zadaného parametrem „ta“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

void ta_Insert(object ta, object value)

- ***Popis:*** *Funkce vloží do ovládacího prvku „TextArea“ zadaného parametrem „ta“ hodnotu „value“ na aktuální místo kurzoru.*
- ***Příklad:***

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

void tab_Disable(int n)

- ***Popis:*** *Funkce zakáže otevření záložky zadané indexem „n“. Indexy jsou číslované od nuly.*
- ***Příklad:***

```
tab_Disable(0);
```

void tab_Enable(int n)

- ***Popis:*** *Funkce povolí otevření záložky zadané indexem „n“. Indexy jsou číslované od nuly.*
- ***Příklad:***

```
tab_Enable(0);
```

void tab_Open(int n)

- ***Popis:*** *Funkce otevře záložku s daným indexem – parametr „n“. Indexy jsou číslované od nuly.*
- ***Příklad:***

```
tab_Open(0);
```

void tb_Disable(object tb)

- ***Popis:*** *Funkce zakáže editaci ovládacího prvku „TextBox“ zadaného parametrem „tb“. Alternativou je obecná funkce „control_Disable“.*
- ***Příklad:***

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

void tb_Enable(object tb)

- ***Popis:*** *Funkce povolí editaci ovládacího prvku „TextBox“ zadaného parametrem „tb“. Alternativou je obecná funkce „control_Enable“.*
- ***Příklad:***

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

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

Popis: *Funkce inicializuje vlastní ajaxový picker, který je definovaný v externí funkci „ngef“. Ovládací prvek, pro který bude picker inicializován, je definován parametrem „tb“. Parametr „f“ definuje funkci, která se použije k načtení pickeru. Parametr „width“ určuje šířku pickeru v pixelech. Parametr „left“ určuje posun okna pickeru na ose x vzhledem k umístění příslušného ovládacího prvku. Parametr „focus“ určuje, zda se má picker zobrazit již při kliknutí na ovládací prvek (true) nebo až při zadávání dat (false).*

Příklad klientské části:

```
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));
}
```

Příklad serverové části (zdrojový kód stránky 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)

- ***Popis:*** *Funkce vrací počet minut konvertovaný z textového řetězce – času – zadaného parametrem „time“ ve formátu „HH:mm“.*

Příklad:

```
var n = Time2Int('01:23');
// Vrací „83“
var n = Time2Int('-01:24');
// Vrací „-84“
```

string translate(string value)

- ***Popis:*** *Funkce vrací jazykový překlad vícejazyčného termínu zadaného parametrem „value“ na základě jazykového nastavení aktuálně přihlášeného uživatele.*

Příklad:

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

string translate(string value, string language)

- ***Popis:*** *Funkce vrací jazykový překlad vícejazyčného termínu zadaného parametrem „value“ na základě jazyka definovaného parametrem „language“.*

Příklad:

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

string urlDecode(string text)

- ***Popis:*** *Funkce dekóduje všechny URL hexadecimální znaky v textovém řetězci zadaném parametrem „text“.*
- ***Příklad:***

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

string urlEncode(string text)

- ***Popis:*** *Funkce zakóduje všechny URL nevalidní znaky (např. mezera) v textovém řetězci zadaném parametrem „text“ za odpovídající sekvenci znaků v hexadecimální podobě.*
- ***Příklad:***

```
var s = urlEncode('Příliš žluťoučký kůň pěl ďábelské ódy');
```

string ZInt(int n)

- ***Popis:*** *Funkce vrací textovou podobu celého čísla „n“ o minimální délce 2 znaků tak, že před číslo menší než 10 umístí nulu.*
- ***Příklad:***

```
var s = ZInt(8);
// Vrací „08“
```