

smartformHelper

Ansammlung nützlicher Funktionen welche oftmals im Einsatz sind. Die Funktionen sind größtenteils so gehalten, dass sie bei anderen Kunden ohne viel Aufwand verwendet werden können.

Darin enthalten sind:

- **setValue:** Setzt einen Wert als Prozessvariable und befüllt ein evtl. vorhandenes Feld damit (egal ob Input, Textarea, Radiobuttongroup, Select usw)
- **getValue:** Holt Wert für Prozessvariable. Wenn undefined wird Leerstring zurückgegeben
- **ajaxCall:** Führt Ajaxcall aus und gibt Response an eine Callback weiter (Params: Type, URL, dataType, contentType, callback)
- **stringContains:** Prüft ob Zeichenkette vars in Wert var vorhanden ist. vars kann ein String oder ein Array aus mehreren Strings sein (Params: variable1, variable(s)2)
- **selectUsersByGroup:** Füllt Dropdown mit Usern einer Gruppe oder allen TIM Usern
HTML: `<select name="INSERT A NAME" id="INSERT AN ID" class="INSERT A CLASSNAME" group="INSERT A TIM GROUP"></select>`
Params: classname
- **getUserName:** Returns Firstname, Lastname from user got. If undefined returns username. (Param: User)
- **selectUsersByGroup:** Fill dropdowns with all users or users from group. Exepts Classname of the dropdowns
HTML: `<select name="INSERT A NAME" id="INSERT AN ID" class="INSERT A CLASSNAME" group="INSERT A TIM GROUP"></select>`
Params: classname
- **getUserName:** Returns Firstname, Lastname from user got. If undefined returns username. (Param: User)
- **getUserDepartment:** Returns department of userGot (Param: user)
- **getDateString:** returns date as string in format dd.MM.yyyy (Params: Date)
- **getTime:** returns time as string in format hh:mm (Params: Date)
- **populateSelectFromArray:** fills an Selectfield with the given arrays
Params: id of the select
array: array filled plain text or objects. If objects, all attributes are taken
attributeList: if only certain attributes of objects should be taken
- **populateSelectFromCSV:** fills an Selectfield with the CSV File from URL
`<select id="testSelect" csvFile= "/loom-portal/custom/test/Ressources/test.csv" csvCols="SpalteA;SpalteB" name="testSelect" class="form-control"></select>`
`#####papaparse.min.js mandatory#####`
Params: -
- **checkValidInput:** checks if regex from field param matches. alerts message and clears field if not. Sets value on change
example:
`<input type="text" name="name" id="id" regex="^\d*$" message="only digits"/>`
`checkValidInput($(".fields"))`
Params: filds

»Achtung: jQuery wird dringend benötigt!!

Sourcen

TODO

Beispiel HTML

```
<!--
    select Users By Group
-->
<select name="BELIEBIG" id="BELIEBIG" group="TIM Gruppe"
class="BELIEBIG">&#160;</select>

<!--
    populateSelectFromCSV
-->
<select id="BELIEBIG" csvFile= "/loom-
portal/custom/test/Ressources/test.csv"  csvCols="SpalteA;SpalteB"
name="BELIEBIG" class="form-control"></select>

<!--
    checkValidInput
-->
<input type="text" name="BELIEBIG" id="BELIEBIG" regex="^d*$" message="Hier
steht die Meldung die ausgegeben wird sollte die Regex false sein"/>
```

Beispiel Custom

```
var head = document.getElementsByTagName("head")[0];
var customUrl = "/loom-portal/custom/BELIEBIGERORDNER/";
scriptTag = document.createElement('script');
scriptTag.setAttribute("type", "text/javascript");
scriptTag.setAttribute("src", customUrl + "smartformHelper.js");
head.appendChild(scriptTag);

(function($, scope){

    var helper = gadget.functions.initFormularFunctions.bind(scope)();

    /*
        Beispiele Funktionsaufrufe:
    */

    //setValue
    helper.setValue("ID", "WERT");

    //getValue
```

```
var value = helper.getValue("ID");

//ajaxCall
helper.ajaxCall(Type, URL, dataType, contentType, callback);

//stringContains
var bool = helper.stringContains("SUCHWERT", "SUCHMENGE");
var bool2 = helper.stringContains("SUCHWERT",
["ARRAY", "AUS", "STRINGS", "DAS", "DURCHSUCHT", "WERDEN", "SOLL"]);

//selectUsersByGroup
helper.selectUsersByGroup( $("select") );

//forceRedraw
helper.forceRedraw( $("select") );

//getUserUserName
var firstname_lastname = helper.getUserUserName(
gadget.getEntity("currentUser") );

//getUserUserNameReverse
var lastname_firstname = helper.getUserUserNameReverse(
gadget.getEntity("currentUser") );

//getUserDepartment
var department = helper.getUserDepartment(
gadget.getEntity("currentUser") );

//getDateString
var ddMMyyyy = helper.getDateString( new Date() );

//getTime
var hhmm = helper.getTime( new Date() );

//populateSelectFromArray
helper.populateSelectFromArray("ID", ["Wert1", "Wert2"], ["Wert1"]);

//populateSelectFromCSV
helper.populateSelectFromCSV();

//checkValidInput
var bool = helper.checkValidInput($("input"));

//alert
helper.alert("TITLE", "MESSAGE", function(){
    //tu was
})

//confirm
helper.confirm("TITLE", "TEXT", "BUTTON OK TEXT", "BUTTON ABBRECHEN
TEXT", function(decision){
```

```

        if(decision) // tu was

    })

})( this.form.ownerDocument.defaultView != null) ?
this.form.ownerDocument.defaultView.jquery :
this.form.ownerDocument.parentWindow.jquery , this);

```

Dabei zu beachten sind Folgende Punkte:

- Aus Performancegründen wird für die selectUsersByGroup auf Queries zurückgegriffen. Diese sind im JavaScript mit definiert und können im Supermandanten angelegt werden.
- Für populateSelectFromCSV wird PAPAParse.min.js benötigt.

smartformHelper Code

smartformHelper.js

```

/*  Method Index
    setValue:                Sets value, merges local and inserts value in
smartformfield if there is one (Params: id, value)
    -----
    getValue:                Returns blank if undefined (Params: id)
    -----
    ajaxCall:                Führt Ajaxcall aus und gibt Response an eine
Callback weiter (Params: Type, URL, dataType, contentType, callback)
    -----
    stringContains:          Prüft ob Zeichenkette vars in Wert var vorhanden
ist. vars kann ein String oder ein Array aus mehreren Strings sein
(PParams: variable1, variable(s)2)
    -----
    selectUsersByGroup:      Fill dropdowns with all users or users from
group. Exepts Classname of the dropdowns
HTML: <select name="INSERT A NAME" id="INSERT AN
ID" class="INSERT A CLASSNAME" group="INSERT A TIM GROUP"></select>
Params: classname
    -----
    getUserName:             Returns Firstname, Lastname from user got. If
undefined returns username. (Param: User)
    -----
    getUserDepartment:       Returns department of userGot (Param: user)
    -----
    getDateString:           returns date as string in format dd.MM.yyyy
(Params: Date)
    -----
    getTime:                 returns time as string in format hh:mm (Params:
Date)
    -----
    populateSelectFromArray:  fills an Selectfield with the given arrays
Params: id of the select

```

```

        array: array filled plain text or objects. If
objects, all attributes are taken
        attributeList: if only certain attributes of
objects should be taken
-----
populateSelectFromCSV:        fills an Selectfield with the CSV File from
URL
                                <select id="testSelect" csvFile= "/loom-
portal/custom/test/Ressources/test.csv" csvCols="SpalteA;SpalteB"
name="testSelect" class="form-control"></select>
                                #####papaparse.min.js mandatory#####
                                Params: -

-----
checkValidInput:              checks if regex from field param matches.
alerts message and clears field if not. Sets value on change
                                example:
                                <input type="text" name="name" id="id"
regex="^d*$" message="only digits"/>
                                checkValidInput( $(".fields") )
                                Params: filds

*/

gadget.functions.initFormularFunctions = function(){

    var $ = (this.form.ownerDocument.defaultView!=null) ?
this.form.ownerDocument.defaultView.jQuery :
this.form.ownerDocument.parentWindow.jQuery;
    var scope = this;

    var helper = {

        /*
        Sets value, merges local and inserts value in smartformfield if
there is one
        Params: id, value
        */
        setValue: function(idGot, value){
            if(typeof value == "undefined" || value == null) return
false;

            value = value.toString();

            var identifier =
idGot.replace(/\[/g,"\\[").replace(/\]/g,"\\]").replace(/\?/g,"\\?");
            var field = $("#"+identifier);

            if($(field).find("option:selected").attr("default")){

                scope.entity.setValue(idGot,"");

```

```
        }else{

            scope.entity.setValue(idGot,value);

        }
        scope.entity.mergeLocal(true);

        if(field.length>0){
            switch($(field).prop("tagName")){
                case "INPUT":
                    if($(field).attr("type")==="radio")
$("input[name="+identifier+"])[value="+value+"]").prop("checked",true);
                    if($(field).attr("type")==="checkbox"){
                        if(value == "true")
                            $(field).prop("checked",true);
                        else
                            $(field).prop("checked",false);
                    }
                    else
                        $(field).val(value);
                    break;
                case "SPAN":
                    $(field).prop("innerHTML",value);
                    break;
                case "TEXTAREA":
                    $(field).val(value);
                    break;
                case "SELECT":
                    $(field).val(value);
                default:
                    break;
            }
        }
    },

    /*
    Returns blank if undefined
    Params: id
    */
    getValue: function(idGot){
        return ( typeof scope.entity.getValue(idGot) ==
"undefined") ? "" : scope.entity.getValue(idGot);
    },

    /*
    Führt Ajaxcall aus und gibt Response an eine Callback weiter
    Params: Type, URL, dataType, contentType, callback
    */
    ajaxCall: function(type, url, dataType, contentType, callback){
        $.ajax({
            type: type,
```

```

        url: url,
        dataType: dataType,
        contentType: contentType
    }).done(function(response){
        if(typeof callback == "function") {
            callback.bind(this)(response)
        } else{
            return response;
        }
    }).bind(this)).fail(function(){
        if(typeof callback == "function") {
            callback.bind(this)(null)
        } else{
            return null;
        }
    }).bind(this));
    },

    /*
    Prüft ob Zeichenkette vars in Wert var vorhanden ist. vars kann ein
String oder ein Array aus mehreren Strings sein
Params: variable1, variable(s)2
    */
    stringContains: function(va, vars){
        var c=false;
        switch(typeof vars){
            case "object":
                $(vars).each(function(i, v){
                    if(va.indexOf(v)>-1){
                        c=true;
                        return false;
                    }
                });
                break;
            case "string":
                if(va.indexOf(vars)>-1){
                    c=true;
                }
                break;
            default:
                c = false;
        }
        return c;
    },

    /*
    Fill dropdowns with all users or users from group. Exepts Classname
of the dropdowns
    HTML: <select name="INSERT A NAME" id="INSERT AN ID" class="INSERT A
CLASSNAME" group="INSERT A TIM GROUP"></select>
    Params: classname

```

FOR PERFORMANCE REASONS ADD QUERIES TO SUPER/ADMIN

Name: getUsersByGroup

Query:

```

SELECT i.ID_, i.NAME_, i.NAMEFIRST, i.NAMELAST, i.EMAIL,
c.DEPARTMENT, c.COMPANYID, c.COMPANYNAME
from
loom_identity i,
loom_identity m,
loom_identity g,
loom_systemconfiguration c
where i.CLASS = 'USER'
and i.ARCHIV_ '0'
and i.USERPROFILE_ID_ = c.ID_
and i.CLIENT_ID_ = ${SYS.CURRENT_CLIENT}
and g.CLASS = 'GROUP'
and m.CLASS = 'MEMBERSHIP'
and m.PARENT_ID_ = g.ID_
and m.USER_ID_ = i.ID_
and g.NAME_ = ?

```

Name: getAllUsers

Query:

```

SELECT i.ID_, i.NAME_, CASE WHEN i.NAMEFIRST IS NULL THEN '' ELSE
i.NAMEFIRST END NAMEFIRST, CASE WHEN i.NAMELAST IS NULL THEN '' ELSE
i.NAMELAST END NAMELAST, i.EMAIL, c.DEPARTMENT, c.COMPANYID, c.COMPANYNAME
FROM
loom_identity i,
loom_systemconfiguration c
where i.CLASS = 'USER'
and i.CLIENT_ID_ = ${SYS.CURRENT_CLIENT}
and i.USERPROFILE_ID_ = c.ID_
and i.ARCHIV_ = '0'
ORDER BY NAMELAST,NAMEFIRST desc

```

*/

```

selectUsersByGroup: function(classGot){
    // Init all selects
    $('.'+classGot).each(function(i, select){
        $(select).attr("loaded","false");
        $(select).empty();
        var val = helper.getValue(
$(select).attr("id") + "RealName" );
        if(val) {
$(this).append($("<option>").prop({"innerHTML": val + " (" +helper.getValue(
$(select).attr("id") + "Expression" ).replace("user(", "").replace(")", "")
+")", "selected":true}));
        }
        else{
            var $option =
$("<option>").attr({"value":"Bitte auswählen...","Expression":"","

```



```

>Email":""," RealName":"","default":"default")).prop({"innerHTML":"Bitte
auswählen..."});

                                if(typeof getSingleTranslation ==
"function"){

                                if(typeof getSingleTranslation
== "function"){
                                $option.text(
getSingleTranslation(null, "select" ,"Bitte auswählen...") );
                                }
                                }

                                $(this).append( $option );
                                }
                                });

                                /*
                                On focus check if loaded and insert users
                                Fills Select with all users if parameter
                                */
                                var allSelectUsers = null;

                                $('.'+classGot).hover(function(){
                                //var val =
$(this).children(":first").prop("innerHTML");
                                if($(this).attr("loaded") == "false"){

                                var selectedGroup =
$.trim($(this).attr("group"));
                                if(!helper.stringContains(selectedGroup,"Wild Card") ||
!helper.stringContains(selectedGroup, "ALLUSERS")){

                                var fillDropDown =
function(selectedGroup){

                                webService.DbConnectionManager.getResultSetAsWsEntity("getUsersByGroup",
selectedGroup, function(usersGot){

                                $(this).attr("loaded","true");

                                $(this).empty();

                                var $option =
$("<option>").attr({"value":"Bitte auswählen...","Expression":"","
>Email":""," RealName":"","default":"default",
"QNumber":""}).prop({"innerHTML":"Bitte auswählen..."});

                                if(typeof
getSingleTranslation == "function"){

                                var translation =

```

```
getSingleTranslation(null, "select" ,"Bitte auswählen...");
                                if(typeof
translation != "undefined")
$option.prop("innerHTML", translation);
                                }

                                $(this).append($option);
                                for(var
u=0;u<usersGot.length;u++){
                                var user =
usersGot[u];
                                var selected =
false;
                                if(typeof
helper.getValue($(this).attr("id")) != "undefined"){
if(helper.getValue($(this).attr("id")) == user.ID_)
                                selected =
true;
                                }

$(this).append($(".<option>").attr({"value":user.ID_,"Expression":"user("+user.NAME_ + ")", "Email":user.EMAIL, "RealName":user.NAMELAST + " " +
user.NAMEFIRST, "QNumber": user.NAME_}).prop({"innerHTML":user.NAMELAST + " ",
" + user.NAMEFIRST + "(" + user.NAME_ + ")", "selected":selected}));
                                }

helper.forceRedraw(this);

                                }.bind(this));

                                }

fillDropDown.bind(this)(selectedGroup);
                                }

                                else{
                                var fillAllUsersDropdown =

function(usersGot){
$(this).attr("loaded","true");

                                $(this).empty();
                                var $option =
$(".<option>").attr({"value":"Bitte auswählen...","Expression":"","
>Email":""," "RealName":"","default":"default",
"QNumber":""}).prop({"innerHTML":"Bitte auswählen..."});

                                if(typeof
getSingleTranslation == "function"){
                                var translation =
getSingleTranslation(null, "select" ,"Bitte auswählen...");
                                if(typeof translation !=
"undefined")
$option.prop("innerHTML", translation);
                                }
                                }
```

```

$(this).append( $option );

for(var
u=0;u<usersGot.length;u++){
    var user = usersGot[u];
    var selected = false;
    if(typeof
helper.getValue($(this).attr("id")) != "undefined"){
if(helper.getValue($(this).attr("id")) == user.ID_)
selected = true;
    }

$(this).append($("<option>").attr({"value":user.ID_,"Expression":"user("+user
r.NAME_ + ")", "Email":user.EMAIL, "RealName":user.NAMELAST + " " +
user.NAMEFIRST, "QNumber": user.NAME_}).prop({"innerHTML":user.NAMELAST + ",
" + user.NAMEFIRST + "(" + user.NAME_ + ")", "selected":selected}));
    }

    helper.forceRedraw(this);
}

webService.DbConnectionManager.getResultSetAsWsEntity("getAllUsers", "",
function(usersGot){

    usersGot.sort(function(a, b)

    {
        var nameA = null;
        var nameB = null;
        if (!a.LASTNAME)
            nameA = "ZZZ";
        else
            nameA =

a.LASTNAME.toUpperCase();

        if(!b.LASTNAME)
            nameB = "ZZZ";
        else
            nameB =

b.LASTNAME.toUpperCase();

        if (nameA <nameB)
            return -1;
        if (nameA> nameB)
            return 1;
        return 0;

    });

fillAllUsersDropdown.bind(this)(usersGot);

    }.bind(this));

```

```

        }
    }
});

/* On change create Variables for
Expression, Email and RealName */
$('.'+classGot).change(function(){
    var option =
$(this).children()[$(this).prop("selectedIndex")];
helper.setValue($(this).attr("id"),$(this).val());
helper.setValue($(this).attr("id")+"Expression",$(option).attr("Expression")
);
helper.setValue($(this).attr("id")+"Email",$(option).attr("Email"));
helper.setValue($(this).attr("id")+"RealName",$(option).attr("RealName"));
helper.setValue($(this).attr("id")+"Group",$(this).attr("group"));
helper.setValue($(this).attr("id")+"GroupExpression", "group(" +
$(this).attr("group") + ")");
helper.setValue($(this).attr("id")+"QNumber", $(option).attr("qnumber"));

    });
},

forceRedraw: function(element){

    if (!element) { return; }
    var n = document.createTextNode(' ');
    var disp = element.style.display; // don't worry about
previous display style

    element.appendChild(n);
    //element.style.display = 'none';

    setTimeout(function(){
        element.style.display = disp;
        if(n.parentNode != null)
            n.parentNode.removeChild(n);
    },20); // you can play with this timeout to make it as short
as possible
    },

    /*
    Returns Firstname, Lastname from user got. If undefined returns
username.
    Param: User
    */
    getUsername: function(userGot){
        var name = "";
        name += ( typeof userGot.namefirst != "undefined" )
? userGot.namefirst + ", " : "";
        name += ( typeof userGot.namelast != "undefined" ) ?
userGot.namelast : "";

```

```
        if(name == "")
            name = userGot.name;
        return name;
    },
    getUserReverse: function(userGot){
        var name = "";
        name += ( typeof userGot.namelast != "undefined" ) ?
userGot.namelast + ", " : "";
        name += ( typeof userGot.namefirst != "undefined" ) ?
userGot.namefirst : "";

        if(name == "")
            name = userGot.name;
        return name;
    },

    /*
    Returns department of userGot
    Param: user
    */
    getUserDepartment: function(userGot){
        if(typeof userGot.userProfile[0].department != "undefined")
            return userGot.userProfile[0].department;
        else
            return "";
    },

    /* Deletes all CBA and Bootstrap CSS from DOM*/
    removeCBAStyles: function(){
        var $cssFiles = $('.customCss,link[rel=stylesheet]');
        var $parent = $cssFiles.parent();

        $cssFiles.each(function(i, css){
            var href = $(css).attr("href").toLowerCase();
            if(href.indexOf("cba")>-1 || href.indexOf("bootstrap")>-1){
                $parent.remove(css);
            }
        });
    },

    /*
    returns date as string in format dd.MM.yyyy
    Params: Date
    */
    getDateString: function(dateGot){
        var day = dateGot.getDate().toString();
        day = (day.length == 1) ? "0"+day : day;

        var month = (parseInt( dateGot.getMonth() ) + 1).toString();
        month = (month.length == 1) ? "0"+month : month;
```

```
        return day + "." + month + "." + dateGot.getFullYear();
    },

    /*
    returns time as string in format hh:mm
    Params: Date
    */
    getTime: function(dateGot){
        var h = dateGot.getHours().toString();
        h = (h.length == 1) ? "0"+h : h;

        var m = dateGot.getMinutes().toString();
        m = (m.length == 1) ? "0"+m : m;

        return h+":"+m;
    },

    /*
    fills an Selectfield with the given arrays
    Params: id of the select
    array: array filled plain text or objects. If objects, all
attributes are taken
    attributeList: if only certain attributes of objects should be
taken

    */
    populateSelectFromArray: function(idGot, array, attributeList){
        $('#'+idGot).find('option').remove();
        $('#'+idGot).append($('
```

```

                count++;
            } else if(attributeListBool){
                currentString += " - " + e[key];
                count++;
            }
        }
    });
    if(currentString)
        $('#'+ idGot).append($("#<option
/>").val(currentString).text(currentString));
        break;
    default:
        //if values are plain text
        $('#'+ idGot).append($("#<option
/>").val(e).text(e));
    }
}
if(typeof(scope.entity.getValue(idGot)) != 'undefined' &&
scope.entity.getValue(idGot)){
    $("#"+ idGot).val(scope.entity.getValue(idGot));
} else {
    scope.entity.setValue(idGot, "-");
    scope.entity.mergeLocal(true);
}
},

/*
fills an Selectfield with the CSV File from URL
<select id="testSelect" csvFile= "/loom-
portal/custom/test/Ressources/test.csv" csvCols="SpalteA;SpalteB"
name="testSelect" class="form-control"></select>
###PAPAparse.min.js benötigt
Params: -
*/

populateSelectFromCSV: function(){
    csvConfig = {
        skipEmptyLines: true,
        header: true
    }
    $('[csvFile]').each(function(){
        var currentField = this;
        $.get($(currentField).attr("csvFile")).then(function(csv){
            var verwendung = Papa.parse(csv,csvConfig);
            var attributeList = (typeof
$(currentField).attr("csvCols") != "undefined" ?
$(currentField).attr("csvCols").split(";") : undefined);
            helper.populateSelectFromArray(currentField.id,verwendung.data,attributeList
);
        })
    });
};

```

```
    },

    /*
    checks if regex matches on change
    params fields

    example: <input type="text" name="name" id="id" regex="^\d*$"
message="only digits"/>
    */
    checkValidInput: function(fields){
        $(fields).unbind("change");
        $(fields).change(function(){
            var regex = new RegExp( $(this).attr("regex") );
            var message = $(this).attr("message");
            if(!regex.test($(this).val())){
                alert(message);
                helper.setValue($(this).attr("id"), "");
            }
            else
                helper.setValue($(this).attr("id"), $(this).val());
        });
    },

    alert: function(titleGot, textGot, callback){
        $("<div>").dialog({
            draggable:false,
            modal: true,
            resizable: false,
            show: {effect: "fadeIn"},
            title: titleGot,
            maxwidth: 600,
            open: function(){
                $(this).html(textGot);
                $(".ui-dialog-titlebar-close").css("display","none");
            },
            buttons: {
                Ok: function(){
                    $(this).dialog("close");
                    $(this).dialog("destroy");
                    if(typeof callback == "function")
                        eval(callback());
                }
            },
            create:function () {
                $(this).closest(".ui-dialog").find(".ui-dialog-buttonset
button").addClass("btn btn-primary");
            }
        });
    },
},
```



```
confirm: function(titleGot, textGot, btnOKText, btnCancelText,
callback){
    //bootbox.alert(textGot);
    $("<div>").dialog({
        draggable:false,
        modal: true,
        resizable: false,
        show: {effect: "fadeIn"},
        title: titleGot,
        open: function(){
            $(this).html(textGot);
            $(".ui-dialog-titlebar-close").css("display","none");
        },
        buttons: {
            Ok: function(){
                $(this).dialog("close");
                $(this).dialog("destroy");
                if(typeof callback == "function")
                    eval(callback(true));
            },
            Abbrechen: function(){
                $(this).dialog("close");
                $(this).dialog("destroy");
                if(typeof callback == "function")
                    eval(callback(false));
            }
        },
        create:function () {
            $(this).closest(".ui-dialog").find(".ui-dialog-buttonset
button").first().addClass("btn btn-primary");
            $(this).closest(".ui-dialog").find(".ui-dialog-buttonset
button").first().next().addClass("btn btn-danger");
            if(btnOKText)
                $(this).closest(".ui-dialog").find(".ui-dialog-
buttonset button").first().html(btnOKText);
            if(btnCancelText)
                $(this).closest(".ui-dialog").find(".ui-dialog-
buttonset button").first().next().html(btnCancelText);
        }
    });
}

return helper;
}
```

From:

<https://wiki.tim-solutions.de/> - **TIM Wiki** / [NEW TIM 6 Documentation](#)

Permanent link:

<https://wiki.tim-solutions.de/doku.php?id=smartformhelper&rev=1522939954>

Last update: **2021/07/01 10:01**

