Autocomplete function in the smartform

A autocomplete function is a function, that, when inputting into a textfield, makes suggestions, which then can be selected. An example would be a field for selecting a employee.

jQuery

For the first step the JavaScript library jQuery has to be integrated. This already contains possibilities for integrating autocomplete functions. The integration has to happen in the initMethod of the smartform, so that jQuery is available at all times. The integration occurs via the following line:

```
jq = (this.form.ownerDocument.defaultView!=null) ?
this.form.ownerDocument.defaultView.jQuery :
this.form.ownerDocument.parentWindow.jQuery;
```

The “$” operator is now available as the “jq” operator and is needed to use jQuery functionalties.

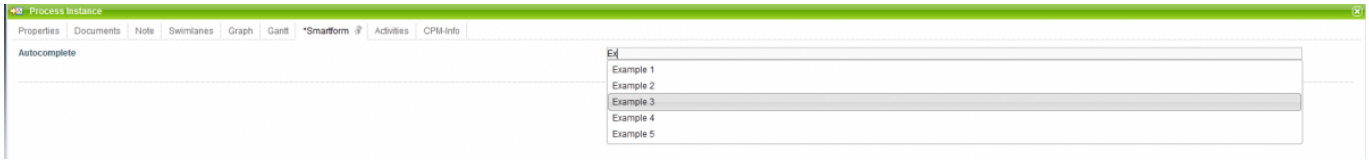
Autocomplete

Any textfield can now be selected to include a autocomplete. THerefor the field has to be found via a selector. A field can be found by its ID via the following line:

```
jq('#ID_OF_FIELD');
```

The first extension stage will be shown with a simple example. Here the autocomplete is put on the field, whereby an array (describes a list of values) is used as a source.

```
jq( "#ID_OF_FIELD" ).autocomplete({
        source: [ "Example 1", "Example 2", "Example 3", "Example 4",
"Example 5"]
});
```



In the next step the function has to be extended, so that the selected option is saved as well. Therefor the code has to be adjusted as follows. A 'change' function has to be added, which is called when selection an option. In this method the nam and value is passed on to the so-called 'entity'. Now it is possible for TIM to save the processvariable. It is also possible to include a valdiate function in the change method, that checks, if a selection was made.

```
jq = (this.form.ownerDocument.defaultView!=null) ?
this.form.ownerDocument.defaultView.jQuery :
this.form.ownerDocument.parentWindow.jQuery;
```

```
jq( "#auto" ).autocomplete({
     source: [ "Example 1", "Example 2", "Example 3", "Example 4", "Example
5"],
     change:function(ui){
               if(ui.item == null){
                  alert("Please choose an entry from the list!");
               } else {
              window['entity'].setValue(this.id, ui.item.value);
               }
     }
});
```

Additional parameter can be set for the autocomplete. To expand the exmaple, we add that suggestions will only be shown after inputting 3 characters. This has the advantage that when using a large data set the suggestion list is not too long. To implement this we add the setting “minLength”. Additional paramters can be seen in the API of jQuery

```
jq = (this.form.ownerDocument.defaultView!=null) ?
this.form.ownerDocument.defaultView.jQuery :
this.form.ownerDocument.parentWindow.jQuery;
jq( "#auto" ).autocomplete({
      source: [ "Example 1", "Example 2", "Example 3", "Example 4", "Example
5"],
      minLength:3,
      change:function(ui){
         if(ui.item == null){
                  alert("Please choose an entry from the list!");
               } else {
              window['entity'].setValue(this.id, ui.item.value);
               }
      }
});
```

Autocomplete with CSV file as source

The suggestions for the autocomplete can of course be more complexe and include more information. Therefor the autocomplete has to be passed a text file. For this example we are using a example file with personnel data. It contains first name, last name, email and phonenumber in the columns of the CSV. The file is located in this exmaple in the custom folder. Firstly the file has to be called and a object has to be created from the content. This happens via AJAX request. The file's content will be split on ever linebreak and then on ever semicolon. Now an object “employee” with data from the text file is generated. The object is then passed on to the autocomplete as a source. If an option is selected, first name, last name, email and phonenumber are displayed in a popup. If the text file cannot be loaded, an error message is shown.

```
jq.ajax({
        type: "post",
        url: "/loom-portal/custom/tim/personaldaten.txt",
```

```
        dataType: "html",
        success: function(data){
            var employees = new Array();
            var allRows = data.split("\n");
            for(var r=0;r<allRows.length;r++){
                var columns = allRows[r].split(";");
                employees[r]={
                    value: columns[0] + ", " + columns[1],
                    firstname: columns[0],
                    lastname: columns[1],
                    email: columns[2],
                    phone: columns[3]
                };
            }
            jq( "#ID_OF_FIELD" ).autocomplete({
                source: employees,
                minLength: 0,
                select: function(event, ui){
                    this.value = ui.item.value;
                    window['entity'].setValue(this.id,ui.item.value);
                    window['entity'].mergeLocal(true);
                    alert("Suche: " + ui.item.value + "\nFirstname: " +
ui.item.firstname +"\nLastname: " + ui.item.lastname + "\nEmail: " +
ui.item.email + "\nPhone: " + ui.item.phone);
                },
                change: function(event, ui){
                    if(ui.item == null){
                        alert("Please choose an entry from the list!");
                        this.value="";
                    }
                }
            });
        },
        error: function(){
            alert('Could not load data');
        }
    });
```