Handle DOM Events

Table of Contents

Outline

In this exercise we will focus on building several screen enhancements based on events.

Upon completion, we've built a page with a simple form and a table that automatically changes its look and feel upon the events triggered by the use of input elements on the screen.

The overall goal is to understand how can those events be handled by executing specific screen actions.

Resources

This exercise can be set up on a new application. For further mention on this exercise it is implicit to have a completely blank Reactive Web App with one single UI module.

Scenario

In this exercise, we'll start to create a new Reactive Web application as well a single module for UI purposes. **JavascriptFrontend** will be our app with a Reactive web Module named **DOM_Events**.

How-To

In this section, we'll show how to do this exercise step-by-step. **If you already finished the exercise on your own, great! You don't need to do it again.** If you didn't finish the exercise, that's fine! We are here to help you.

Getting Started

Create a new Application from scracth and name it **JavascriptFrontend**. Please feel free to choose any color in the app name and description form or optionally upload a custom icon. Next, please create a new Reactive Web Module type and name it **DOM_Events**.

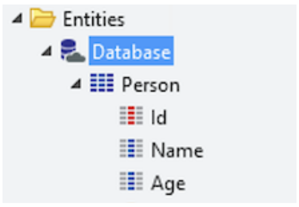
BootStrap Data

As our Module uses the Person Entity, we have created a small setup file to iniatilize our application data. In order to do so, open the **DOM_Events** module to:

1) on the Data tab, by right-click the Entities folder and choose the option *"Import New Entities from Excel..."*

2) choose **this** Excel file and follow all the Excel bootstrap steps until you've created the new Person entity with two attributes (Name and Age).

After you complete these steps the Database should be as the image below.



Publish your changes and afterwards check the records existing in the Person entity.

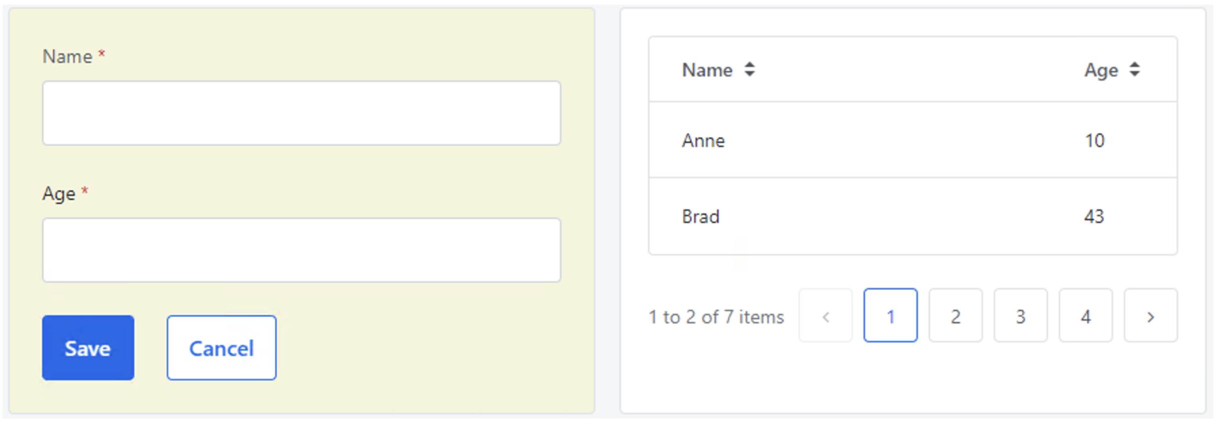
1. Screen content

In this chapter we start to prepare our screen that is split in two halfs. As the first half, on the left hand side, is a form to edit or create some person details, the right hand side shows a list of existing persons.

When the user clicks the form he can either create a new person or go to the list and choose an existing person details. If so, the form will be automatically updated.

Of all events triggered at the form or at the person list we will just be handling some of them. The goal is to improve the user experience at the page by updating some elements look and feel.

After you complete all the steps the screen should be as the image below.

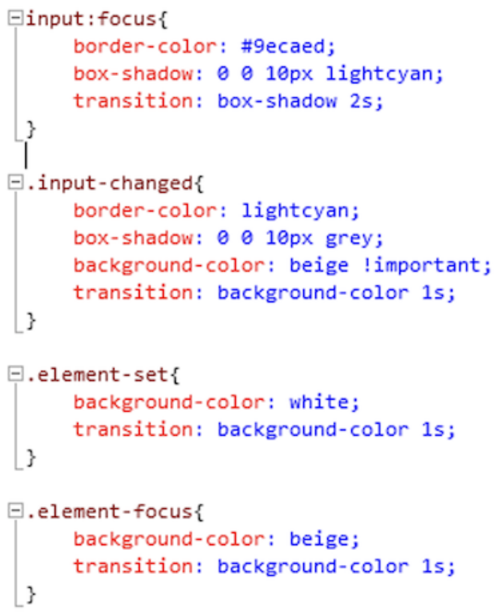


Name	Age
Anne	10
Brad	43

To do such, please go to the Interface tab and create a new Blank screen named **DOM_Events_Home**.

After the screen is created, it is necessary to prepare it's look and feel.

Please go to the screen Style Sheet Editor and place the following CSS classes in the editor.



```
input:focus{
    border-color: #9ecaed;
    box-shadow: 0 0 10px lightcyan;
    transition: box-shadow 2s;
}

.input-changed{
    border-color: lightcyan;
    box-shadow: 0 0 10px grey;
    background-color: beige !important;
    transition: background-color 1s;
}

.element-set{
    background-color: white;
    transition: background-color 1s;
}

.element-focus{
    background-color: beige;
    transition: background-color 1s;
}
```

1.1 Persons table

To create the Person table, drag and drop the Person entity into the **DOM_Events_Home** page MainContent area. Automatically, the table will be created, as well as the corresponding aggregate and other auxiliar local screen variables for the table pagination widget.

Create the following Screen local variables: **isInputSelected**, **isListSelected**, **isListVisible**, **isInputChanged**. All the variables are of *Boolean* Datatype, Default Value as *False*.

Afterwards, enclose all the new widgets in a single Container. Configure the container with:

- Layout Width to six columns (*6 col*)
- **Visible** property to **isListVisible** (local screen variable)

Also, at the screen change the local variable **MaxRecords** Default Value to 3. This variable was automatically created on the previous step of the Person table.

1.2 Person details Form

1) Drag a new Form widget into the **DOM_Events_Home** page MainContent area on the left hand side of the existing container.

2) Create the Form fields automatically, by dragging the Person Entity into the new empty Form.

3) Double-click the Form "Save" button to create a new Screen Action. Please name it **Save**.

4) Add a new Button into the Form with the text "Cancel". Place it next to the **Save** button and again double-click on it, as the "Cancel" screen action is also created. Please name it **Cancel**.

5) Shrink the Form Layout Width to six columns (*6 col*).

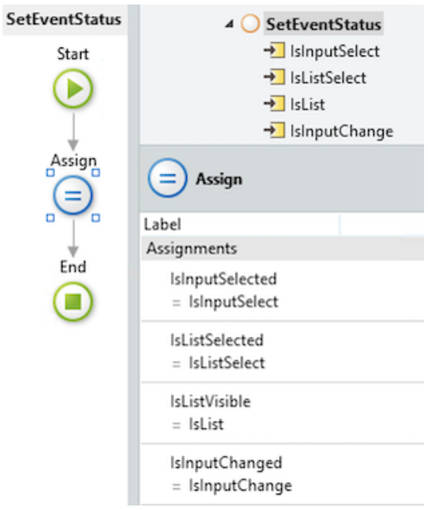
6) Place a new container with both buttons inside it and set it's **Visible** property to **isInputSelected**

Publish your changes and open the **DOM_Events_Home** page in the browser.

1.3. Dynamic behaviour

Let's improve the screen style to support the dynamic look and feel. To manage the several boolean control variables, we need to create a new screen action named **SetEventStatus**. It does one single assign to each one of the Boolean screen variables.

The action flow and input shoould be as shown in picture below.



SetEventStatus
- IsInputSelect
- IsListSelect
- IsList
- IsInputChange

Assign

Label

Assignments

IsInputSelected = IsInputSelect

IsListSelected = IsListSelect

IsListVisible = IsList

IsInputChanged = IsInputChange

The Form inputs will trigger some events but their behaviour will also simultaneously update in some cases. Let's update their style classes property:

1) At the Form, set it's Style Class Properties to the following expression *x.y*:

- *"form card" + If(isInputSelected, " element-focus", " element-set")*

2) Edit each Form input field Style Class Properties with the expression *x.y*:

- *"form-control" + If(isInputChanged, " input-changed","")*

3) Go back to the enclosing Container on the right hand side and edit it's Style Class Properties with the following expression *x.y*:

- *"card" + If(isListSelected, " element-focus", " element-set")*

2. Event handlers

The two main events events being handled when:

1) the user chooses an element in the table

2) any form input field is selected for writing text.

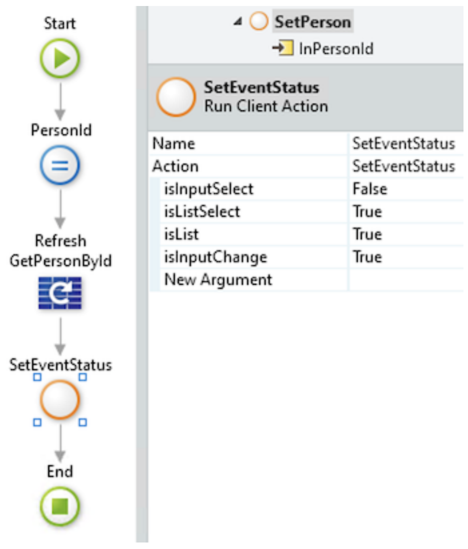
As the user clicks on any table cell, the form input's will loose focus and the table will be highlighted.

2.1 Table click

To handle this event on the table we set the **onclick** event on each expression *x.y* for the Name and Age and create a new Event handler action for the event **onclick**:

- name it **SetPerson**
- create a new input paramenter **InPersonId**, Person Identifier Data type, Mandatory.

This new action will be responsible for updating the Form data. So, edit the **setPerson** action flow as follows:



SetPerson
InPersonId

SetEventStatus
Run Client Action

Name	SetEventStatus
Action	SetEventStatus
isInputSelect	False
isListSelect	True
isList	True
isInputChange	True
New Argument	

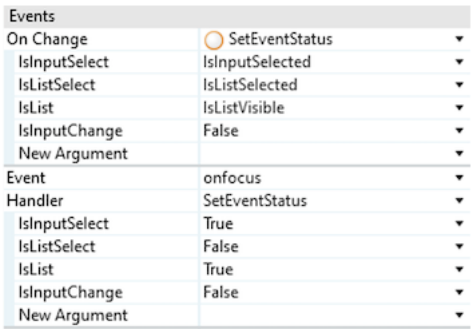
The first node is an assign to the screen local variable **PersonId** with the screen action input parameter.

2.2 Input focus

The focus event is triggered when the mouse cursor or pointer is clicks or is set on a widget, as it happens with input text, radio, or dropdown lists.

For our scenario we set the **onfocus** event on the form input fields. When any input is clicked by the users the table is automatically displayed so the user can verify if the info they were going to register already exists. Also, the users can decide last minute that they wish to update any of the record fields. When they do, the form is auto filled with the record info.

Both inputs for *Name* and *Age* will trigger the same behaviour as the **OnChange** and **onfocus** event handling is set up as follows:



Events	
On Change	SetEventStatus
IsInputSelect	IsInputSelected
IsListSelect	IsListSelected
IsList	IsListVisible
IsInputChange	False
New Argument	
Event	onfocus
Handler	SetEventStatus
IsInputSelect	True
IsListSelect	False
IsList	True
IsInputChange	False
New Argument	

Publish your changes and open the **DOM_Events_Home** page in the browser and test the behaviour of both halfs of the screen.

Extra tasks!

1) Complete the flow of the **Cancel** button screen action as it should reset the whole page back to it's initial state without using the Destination node.

2) Complete the flow of the **Save** button screen action as it should create or update existing info or create a new person. Do not forget the Form Validations before writing in the database. If the Save is successfull the page should return to it's initial state too without using the Destination node.