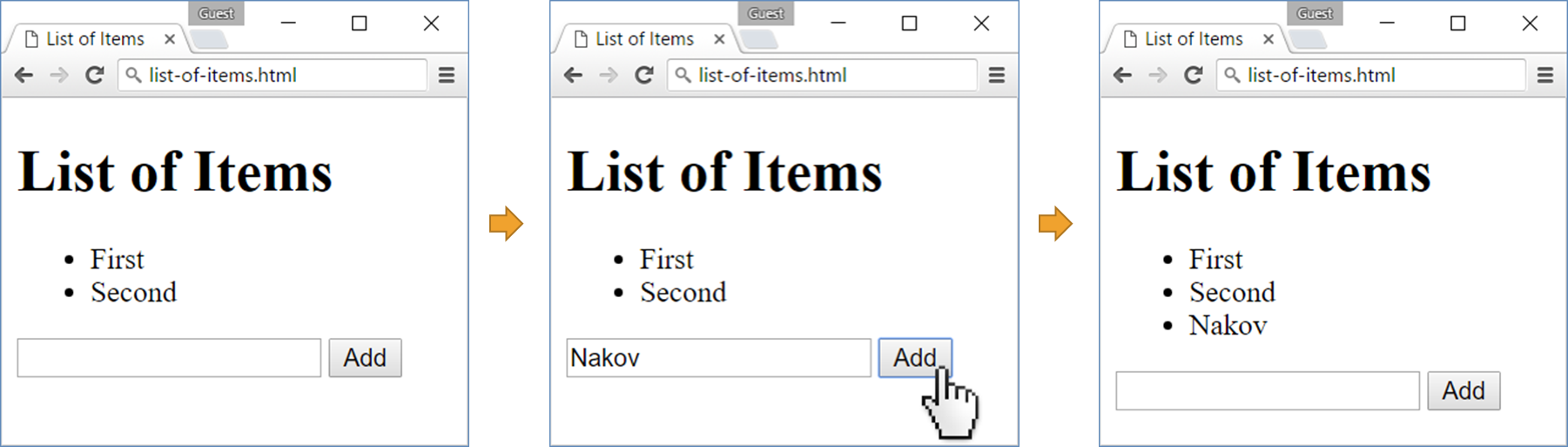
# Exercises: jQuery and AJAX

Problems for exercises and homework for the [“Software Technologies” course @ SoftUni](https://softuni.bg/courses/software-technologies).

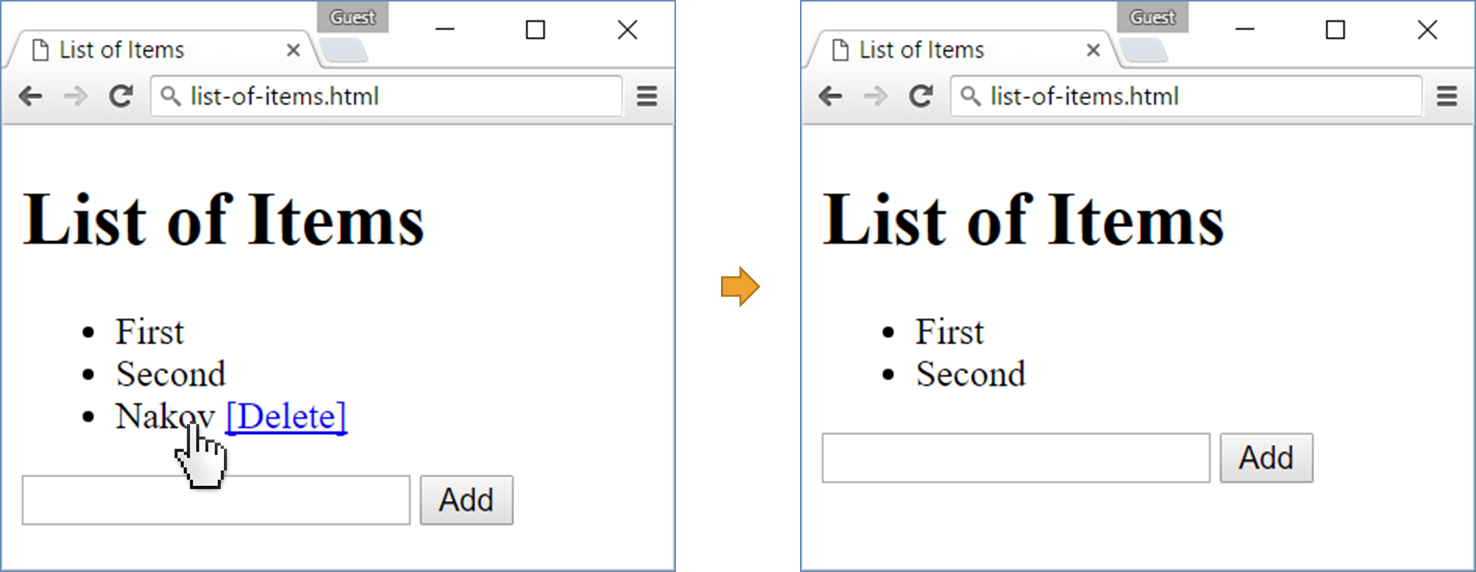
## Add / Delete Items

Create a dynamic HTML page to hold and edit a **list of items**. Implement **add** + **delete** functionality in JS.

**Adding** items could look like this:

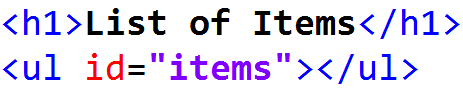


**Deleting** items could happen by showing a **[Delete]** link when the mouse is over some item:



### Hints

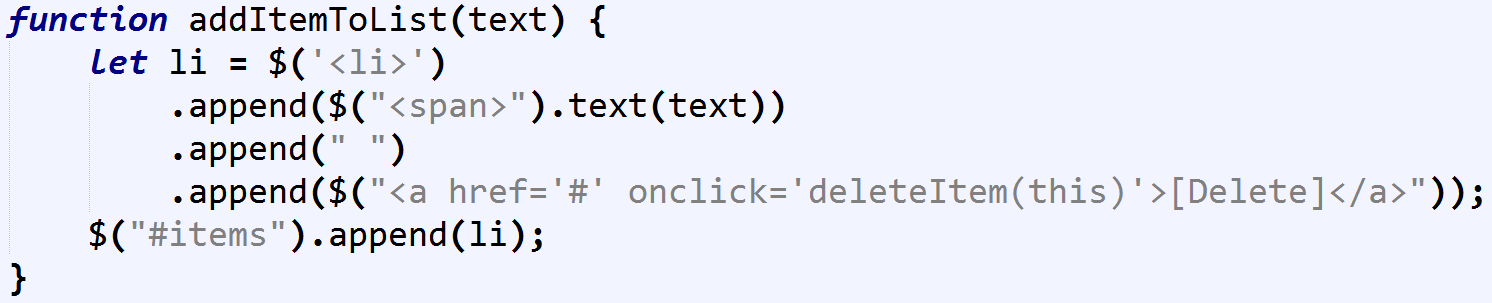
* Create your HTML page, e.g. items.html. Put an empty items list in the body:



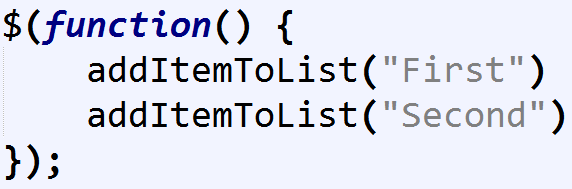
* Include the **jQuery** library to simplify the DOM manipulation:
  + Download the jQuery code from <https://code.jquery.com/jquery-3.0.0.min.js>.
  + Save jquery-3.0.0.min.js in the current folder (where items.html stays).
  + Reference jQuery in the <head> section of your HTML:  
    
* For each item in the list you should add in the DOM tree the **text** and **“delete” link** after it:
  + **<span>*item text*</span>**
  + **<a href="#" onclick="deleteItem(this)>[Delete]</a>**

Note that the function deleteItem() takes this as parameter, which holds the clicked hyperlink object from the DOM tree. It will be used later to find the list item for deleting.

**Adding a new item** to the list could be done by the following JS code using jQuery:

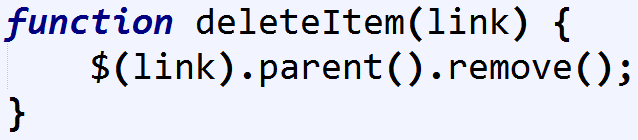


* Add some items in the list to test the above function. You may use the following code:

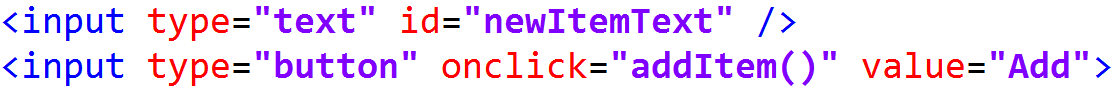


This function will be executed after the HTML page is fully loaded by the browser.

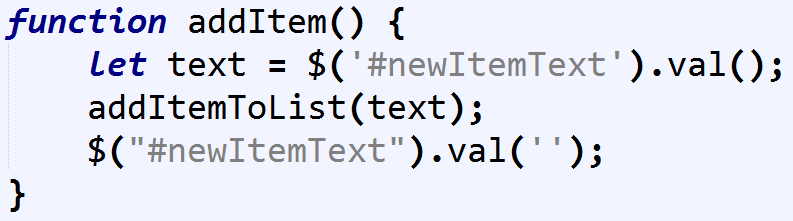
* On click on the **delete link**, delete its parent. Note that the function takes the hyperlink object from the DOM tree as input parameter:



* Test the delete link for the items on the page by clicking on it. It should work correctly.
* Now add the “**create new item**” functionality. First create the **input text box** and **[Add] button**:

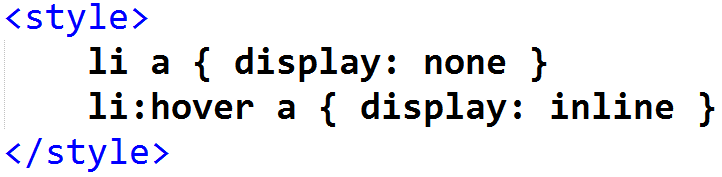


Next, write the JavaScript code to handle the **[Add] button** click:

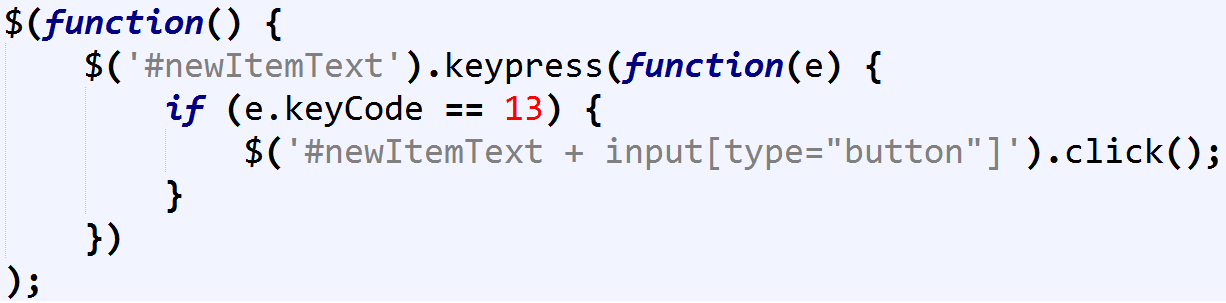


It finds the text typed in the text box and calls the addItemToList(text) function to add new item to the list, then clears the text box (puts empty text in it).

* Finally, make the **[Delete] links** invisible unless the mouse is positioned over some of the list items. The easiest way to do this is by adding the following **CSS style** in the HTML <head> section:



* Optionally, make pressing **[Enter]** key in the text box to add new item without clicking on the **[Add]** button:



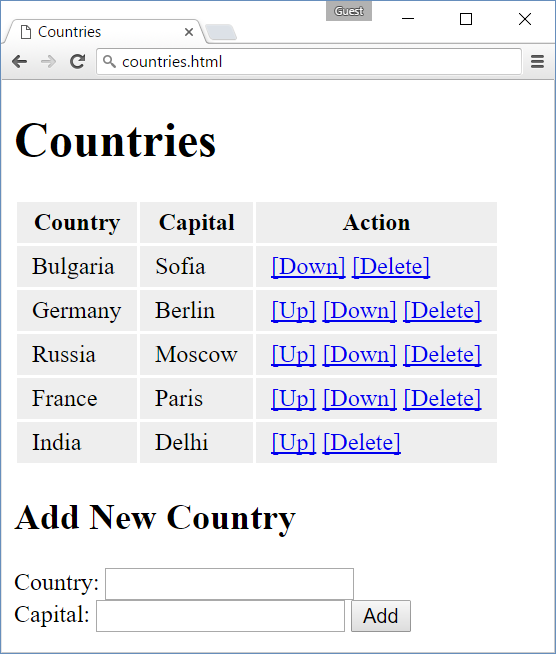
The above code attaches to handle the **“key press” event** for the input text box and when the **[Enter]** key is pressed (key code == 13), it finds the **submit button** on the right on the text box and clicks it.

## Countries Table

Create a dynamic HTML page to hold **countries** with their **capitals** in a table (see the screenshot). Implement the following functionality:

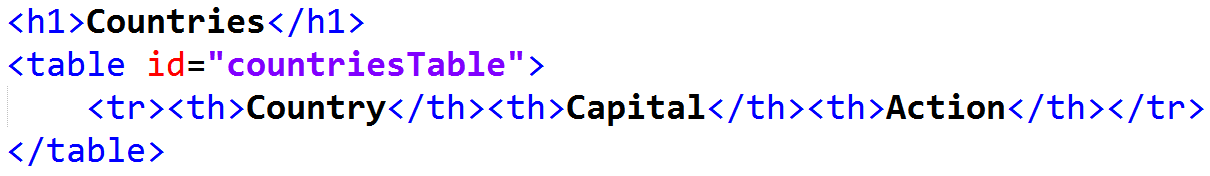
* Users can **add** new country, move country **up** / **down** and **delete** existing country.
* Each country can be **moved up** (except the first), **moved down** (except the last) and **deleted** (after confirmation).

Use **JavaScript** and **jQuery** to implement the requested functionality.



### Hints

* Create your HTML page, e.g. countries.html. Put an empty table with column headers in the body:

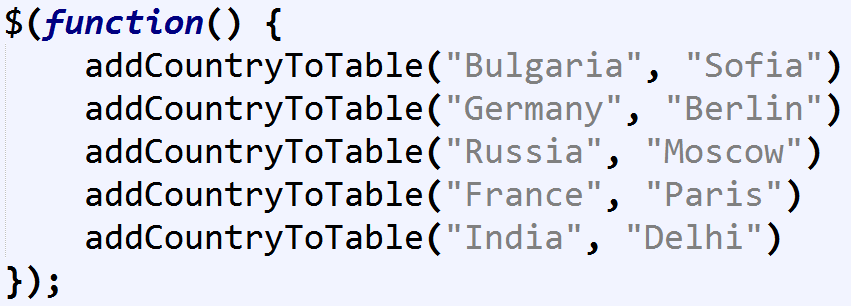


* Include the **jQuery** library to simplify the DOM manipulation:
  + Download the jQuery code from <https://code.jquery.com/jquery-3.0.0.min.js>.
  + Save jquery-3.0.0.min.js in the current folder (where items.html stays).
  + Reference jQuery in the <head> section of your HTML:  
    
* Create a JavaScript function to **add a country** to the table (country + capital + action links):

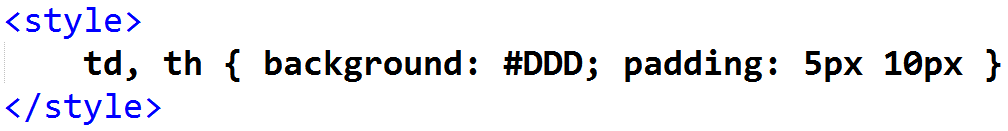


It returns the new added row as a result. You may need it later.

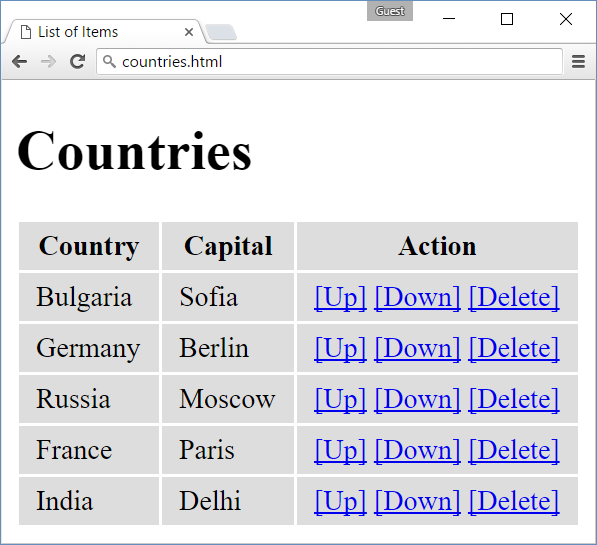
* Add a few countries to test the above code:



* Add some CSS styles in the table **<head>** section to make the table look better:

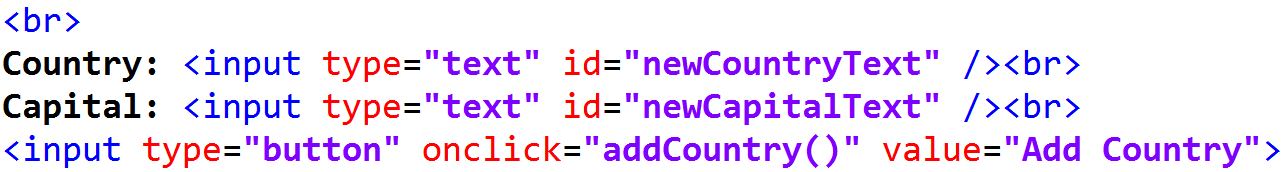


* **Test your code** to ensure it works (even when it is not yet finished):

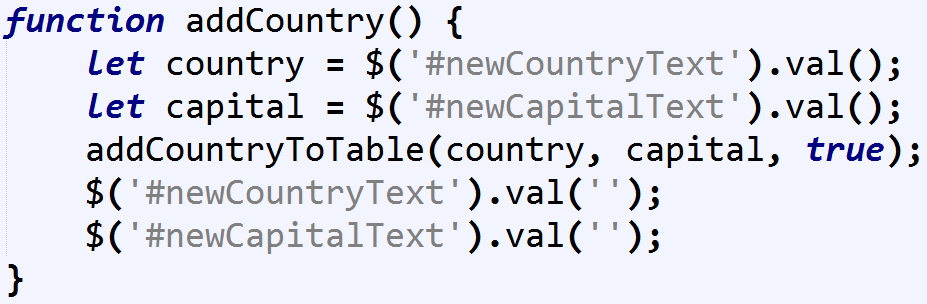


Still the action links will not work, but the countries should be shown in the table.

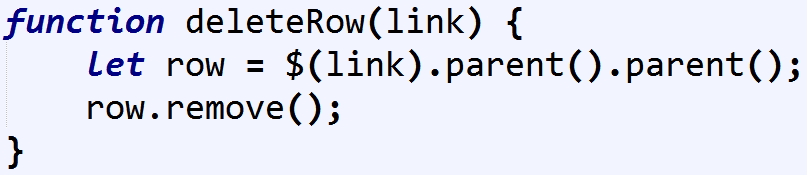
* Now, let’s **add the “create new country” functionality**. First add the input fields and **[Add Country]** button:



Then, write the **JS code** for adding a country in the table:

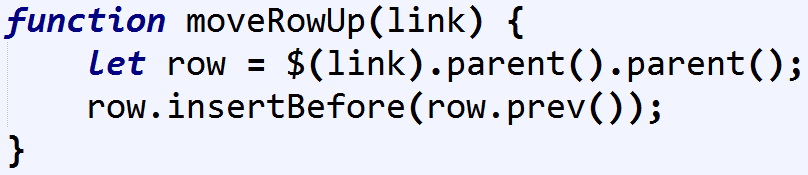


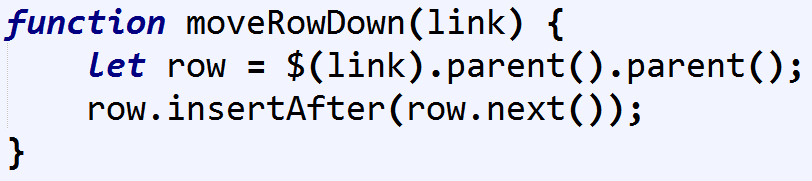
* Let’s implement the “**delete row**” functionality. Write the JS function to handle the **[Delete]** button click:



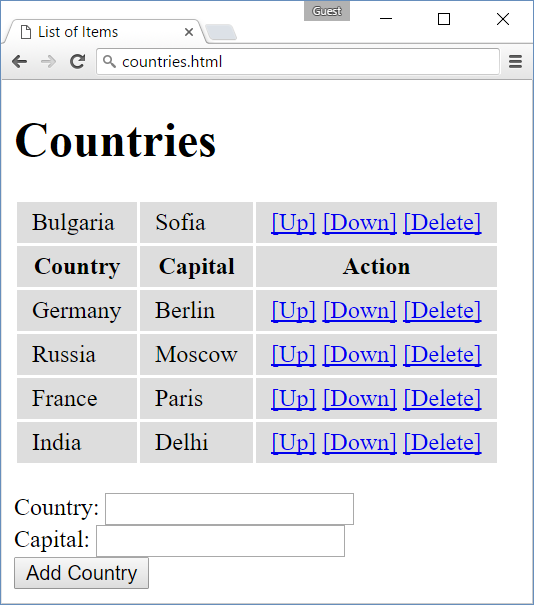
**How it works?** The function takes as input the DOM element holding the **[Delete]** hyperlink. Its parent is the table cell (<td>). Its parent is the table row for deleting (<tr>). It removes the row from the DOM tree.

* Next, let’s implement “**move row up**” and “**move row down**” functionalities:

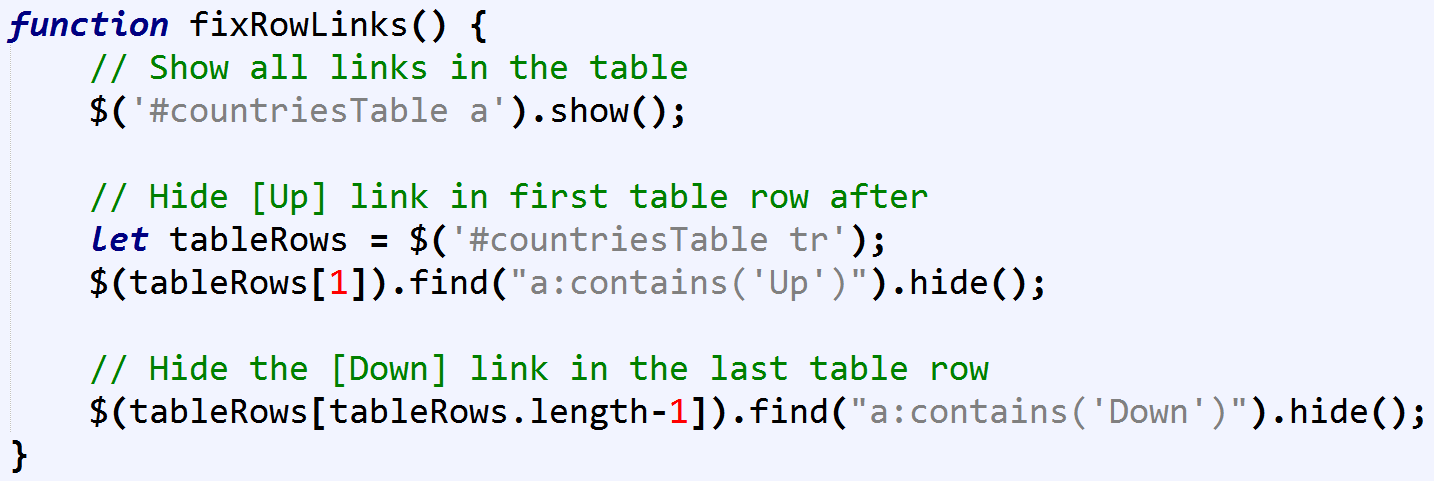




* Now **test your code** again. It is almost ready, but still needs some details to be improved. It does not hide the link **[Down]** at the last row and the link **[Up]** at the first row after the table header:



* Let’s implement **hiding** the **[Up]** and **[Down]** links when not needed. This JS function will do the job:

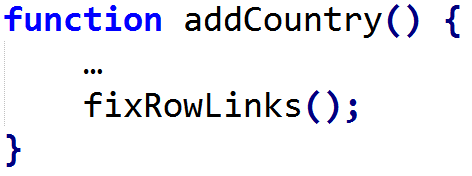
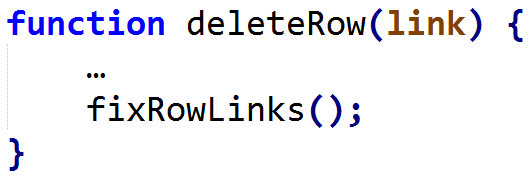


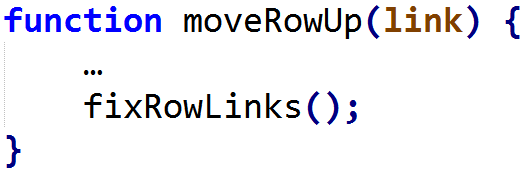
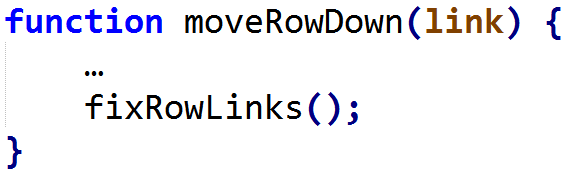
**How it works?** First, show all links in the table. Select them with jQuery selector and invoke show().

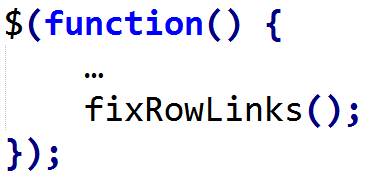
Later, in the first table row after the heading row (row 1), find all links holding the text “Up” and hide them.

Finally, in the last table row (length-1), find all links holding the text “Down” and hide them.

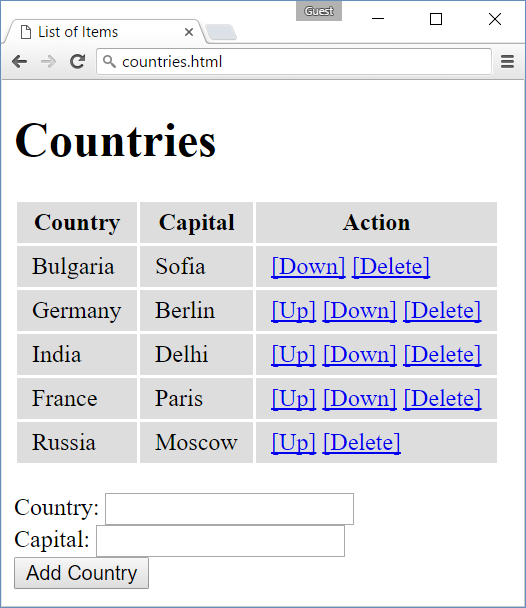
* Now we have the code that **hides** the unneeded **[Up]** and **[Down]** links. Put it at the end of all functions that **modify the table**:

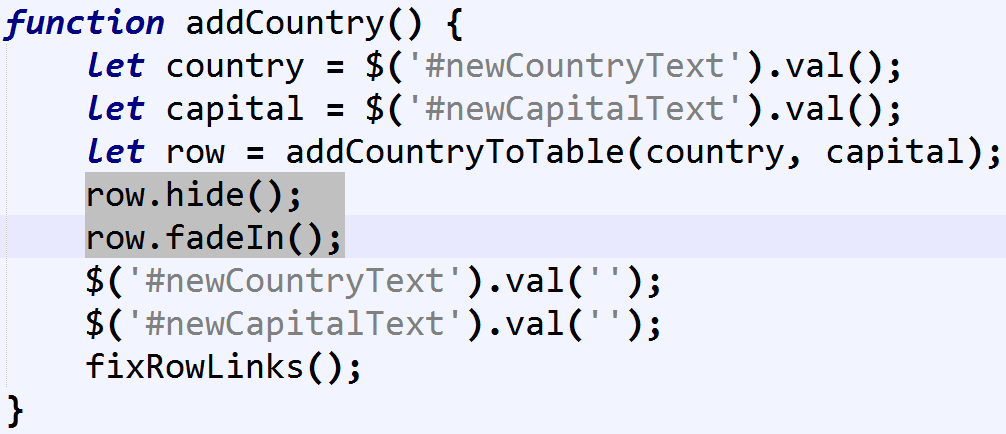
 

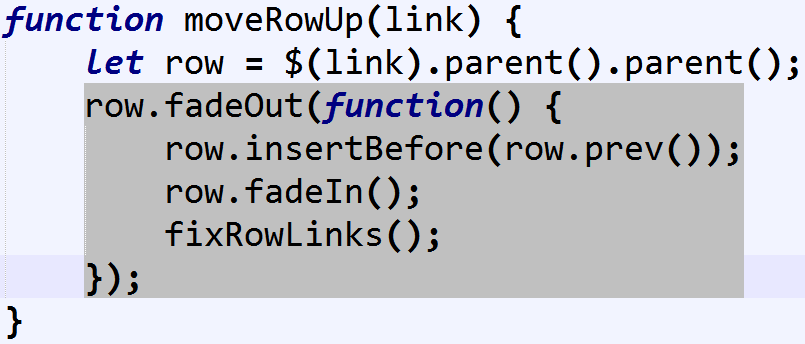


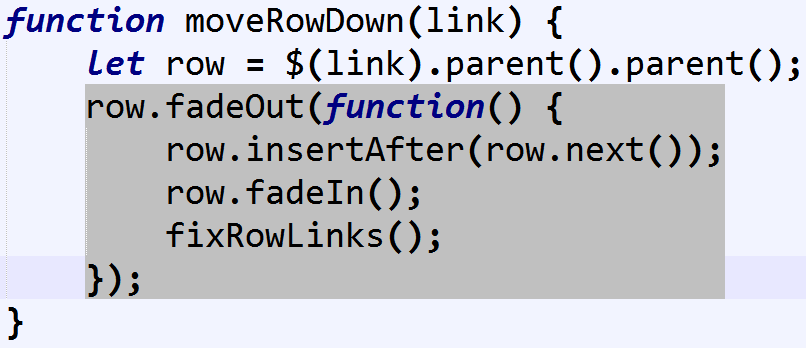
* **Test your code** again. It should work correctly. The first and last data rows should have no links for moving up and down respectively. This works correctly after using **[Delete]** and **[Add Country]**. Test is well.

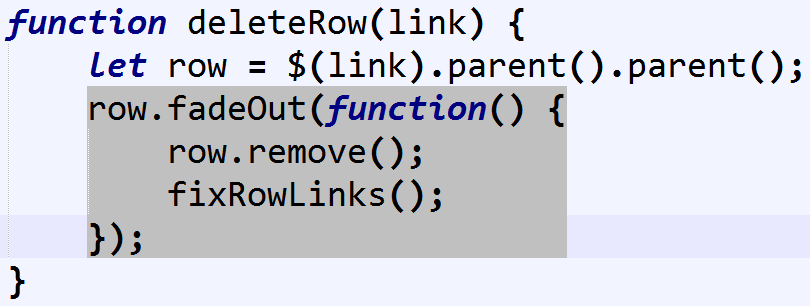


* Finally, let’s make things look and behave better. Add **transition effects (animation)** after **add** / **delete** / **move up** / **move down**. The **user experience** will be improved very much, try it! We shall use the jQuery fadeIn() and fadeOut() functions which run asynchronously and take a onComplete function callback to perform some action after the fade effect is completed:









* Test you code and **enjoy the fading effects**!

## Book Library

Register in Kinvey and create a “BookLibrary” app. Create a collection “books” to hold book **title** + **author** + **description**. Create a HTML page with several buttons (links) with the following functionality:

* **Login**
  + Login in Kinvey with existing username + password.
  + In case of success, a session key should be stored somewhere in the application.
* **Register New User**
  + Register in Kinvey with existing username + password.
  + In case of success, a session key should be stored somewhere in the application.
* **Logout**
  + Logout from Kinvey and forget the session key from the application.
  + This button is available after successful login only.
* **List All Books**
  + Load the books from Kinvey and display them in the HTML page (in some predefined container).
  + This button is available after successful login only.
* **Create a New Book**
  + Shows a **“create new book” form**. When the form is submitted, a new book is added in Kinvey.
  + This button is available after successful login only.

This is how your HTML page may look like:

**TODO**

## \* Book Comments

Add functionality to **add comments** for each book. Save the comments in each book object as array of strings.