

7 Error handling

Up till now, the assumption is: a perfect user handles perfect data without errors. However, the world is not perfect. What happens when the back-end has no data? What if the user submits invalid data? In those cases, the user has to be informed of what went wrong.

7.1 No data in the back-end

When there is no data in backend, or the search-by-fetch on name has no matching games, you could limit yourself to only show an empty table. However, the user does not now that the empty table is a consequence of the absence of data, or that it is caused by an internal error. Therefore, the table should be hidden (e.g. by the CSS property “display: none”) and an appropriate message should be shown.

7.1.1 Exercise: Hide the table and show a message when there is no data



When data is fetched, your code can monitor the received number of data. When there is no data, the table with the overview is hidden and feedback is shown in `div#status`.

E.g. when you delete all games, application should show following:

My Games

Show my favourites

Show all

Show games with rating higher than:

Fetch games

from backend with name containing:

Status

No games in library



Evaluation criteria: You should have updated in the file `table-overview.js`

- ❑ Write a function “`hideTable({tableId})`” that hides the table with given id by adding a classname “`hidden`”. This class sets the style of the given element to “`display: none`”;

- ☐ Refactor the function “renderAllGames()” such that it behaves as follows when there are no games in library:
 - Hides the table
 - Adds an error message to div#status
- ☐ Check the application: remove all data. The message should appear. Add a new game. Is your application still working?

7.2 Input validation

In add-game.js we can add validation to the form, e.g. to inform the user that one of the fields is missing. For some checks (e.g. empty field), HTML-validation can be used (with the attribute “required” for instance). However, as an exercise, we make our own customized input validation.

7.2.1 Exercise: add input validation



Show appropriate messages when

- ☐ One or more fields are empty
- ☐ The length of a name is invalid: $2 \leq \text{length} \leq 64$
- ☐ The rating is not valid: $0 \leq \text{rating} \leq 10$
- ☐ The name is not unique in the library.

For this last item (name not unique): a GET-request to

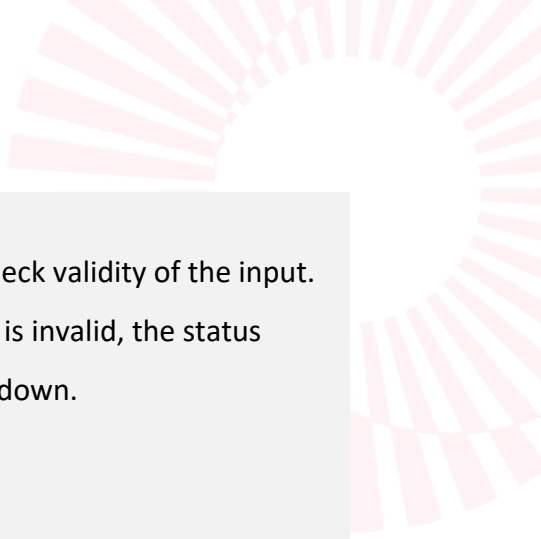
<http://localhost:3000/games/name/:name> returns the id of the game with given name and null otherwise.

Add a game	Add a game	Add a game
Name: <input type="text"/>	Name: <input type="text" value="bb"/>	Name: <input type="text" value="bb"/>
Type: <input type="text" value="ann"/>	Type: <input type="text" value="ann"/>	Type: <input type="text" value="ann"/>
Rating: <input type="text" value="2"/>	Rating: <input type="text" value="2"/>	Rating: <input type="text" value="15"/>
<input type="button" value="Add Game"/>	<input type="button" value="Add Game"/>	<input type="button" value="Add Game"/>
No empty values allowed for name, type and rating.	Game name must be unique	Rating must be between 0 and 10



Evaluation criteria: You should have updated in the file add-game.js

- ☐ Create function “addStatusError(status)”. This function clears the status message if necessary and adds a new message in red.

- 
- ❑ Read input from fields
 - ❑ *Before* you add the game to the backend: check validity of the input.
Use if/else statements: when the first check is invalid, the status message is created and the function breaks down.
 - ❑ Concerning the uniqueness of name:
 - Try in your browser the url
<http://localhost:3000/games/name/:name>. Replace “:name” by an existing game name. Check the output.
 - Write a function “nameIsUnique({gameName})”. This function returns the response of the request above as JavaScript. Do not forget to use async and await.
 - When “nameIsUnique(name) !== null”, the backend has found a game with the given name. You can get the id of that game with “nameIsUnique(name).id”.
 - Use this function in the if-statement in addGame(). Again, do not forget to use “await”!
 - ❑ Finally, add a valid game. Be aware that the status error disappears.

8 Polling

Open your games application in two browser tabs. The first tab shows the overview, the second the add-form. Add a new game. Note that the new game is shown in the second tab (where you added one), but not in the first one unless you refresh the page.

This problem can be solved by polling. This technique triggers the browser to regularly fetch data from server.

8.1.1 Syntax



```
setInterval( fetchAndRenderAnimals, 1000 );
```

The function `setInterval()` expects as first parameter a function that is called when a time interval is passed. That interval is the second parameter (in milliseconds).

8.1.2 Exercise: Use polling to synchronize the games overview



Add polling to your application to synchronize the games overview.

- ☐ Open the app in two browser tabs. Add an extra game in one tab and check that the overview page in the other tab is refreshed automatically.
- ☐ At this point, do not care about filter and search functionalities



Evaluation criteria: You should have updated in the file `table-overview.js`

- ☐ Add `setInterval(function,1000)` at the end of the file. The function *function* is your rendering function.
- ☐ If necessary, clear `tbody` to avoid repetition of table rows.

8.1.3 Exercise: repair your search-by-name functionality (if it's broken)



Refactor your code such that the search functionality (search by name) is working (exercise 6.4.1) (if it is not working).



Evaluation criteria: The solution depends on your program code. A possible solution is

- ☐ Do not call “setInterval()” on “fetchAndRender()”, but on “searchByFetchAndRender()”.

8.1.4 Exercise: repair your ‘filter on rating’ (if it's broken)



Now check the filter functionality on rating (exercise 5.3.2). If it is not working, refactor your code.



Evaluation criteria: The solution depends on your program code. A possible solution is

- ☐ Maybe you call in “searchByFetchAndRender()” the function “renderAllGames()”? In exercise 5.3.1 we suggested to pass a filter function as parameter to “renderAllGames()”. Can you pass the function that filters the rating as a parameter when you call renderAllGames() in searchByFetchAndRender()?

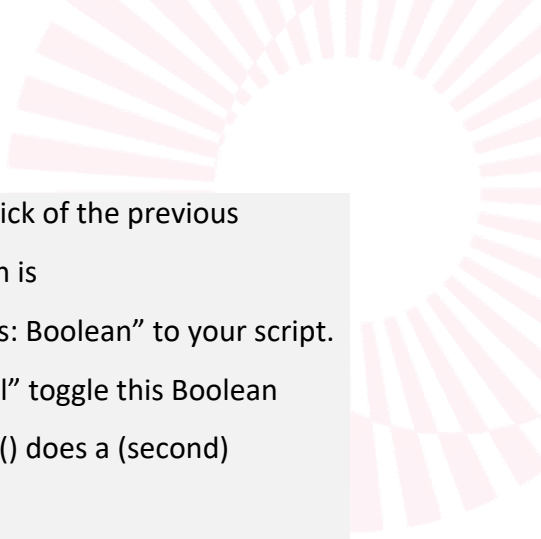
8.1.5 Exercise: repair ‘show favourites’ (if it's broken)



Now check the button “show my favourites”. This button should work in all cases: no filtering, filtering on rating, searching on name, combination of both. If it is not working, refactor your code.



Evaluation criteria:



The solution depends on your program code. The trick of the previous exercise is not working anymore. A possible solution is

- ☐ Introduce global parameter “showFavourites: Boolean” to your script.
- ☐ The click-events “show favourites”/“show all” toggle this Boolean
- ☐ Depending on the Boolean, renderAllGames() does a (second) filtering on favourites.