5 DOM and Events

In order to keep overview of our work, we start with new files.
Create a html file “table-overview.html”. Add page structure (e.g. navigation) as before.
Create a js file “table-overview.js” and include in the new html file. You can include all other js-files in table-overview.html if necessary.

5.1 Creating elements via Javascript

5.1.1 Syntax

We can add, update and remove html elements using javascript.

```
const footer = document.createElement("footer");
footer.innerHTML = "some text"
document.querySelector("body").appendChild(footer);
footer.className = "message";
```

First, element footer is created. The element is filled with the text “Some text”. Then footer is added in the DOM as last child of html element <body>. Finally, the class-attribute “message” is added to the footer.
Remark we use “querySelector()” and not “getElementById()”: querySelector returns the DOM element with given (CSS)-selector (here “body”), see also https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector .

5.1.2 Exercise: Create elements div#status and h3 in that div

Write javascript code that creates a div with id “status” in the (new) HTML-file. Add an element “h3” with content “Status” to the new div.

The result should look like

```
<main id="main">
  <h2>My Games</h2>
  <div id="status">
    <h3>Status</h3>
  </div>
</main>
```

Evaluation criteria: In the file overview-table.js you should have

- ❑ Created elements “div” and h3

- ❑ Added the id “status” to the new div
- ❑ Added some text to h3
- ❑ Appended h3 to the new div and the div to the element main.

5.1.3 Exercise: Create new table

Likewise, we can also create a new table where we can keep track of our animals. This is cleaner than just writing our content in paragraphs.

Add a new table in the table-overview.html file.

The table should have 4 headings: Name, Type, and Rating.
For now, the tbody element should be empty, and should have id “my-games-table-body”.

The result should look like this:

```
<main>
  <h2>My Games</h2> overflow
  <table> overflow
    <thead>
      <tr>
        <th>Name</th> overflow
        <th>Type</th> overflow
        <th>rating</th> overflow
      </tr>
    </thead>
    <tbody id="my-games-table-body"></tbody> overflow
  </table>
  <div id="status">…</div>
</main>
```

Evaluation criteria: In the file table-overview.html you should have

- ❑ Created a new table element
- ❑ Added the required html to the table element
- ❑ Made sure that the tbody element has id “my-games-table-body”

5.1.4 Exercise: Populating our table with game data

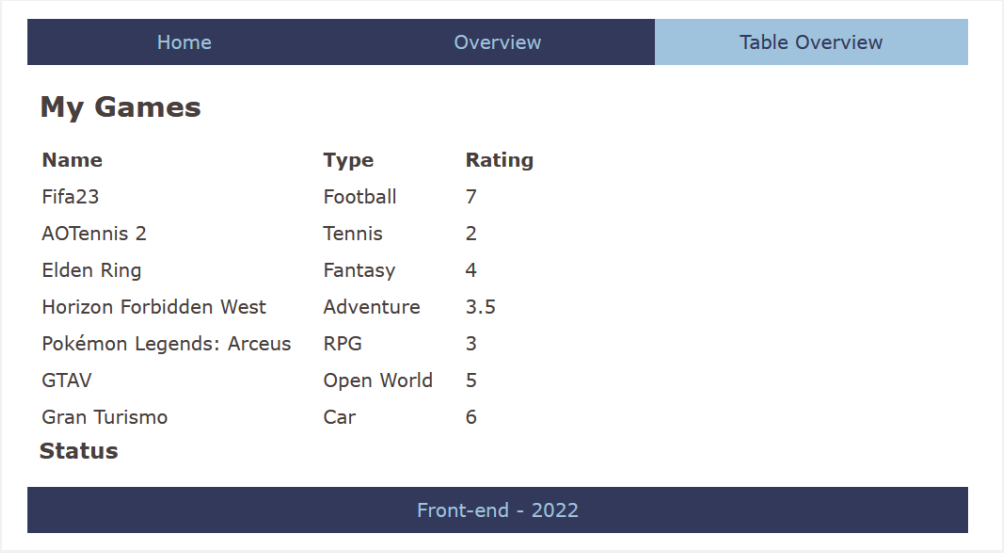
Now that we have our table, we can create a function that will automatically populate it with game data. We can use forEach to create a new row in tbody for each game that we have.

Write a function, renderGames, that accepts a games array as parameter and creates a new table row for each game inside the array.

Copy the games array from “games.js” and add it to the top of the file “table-overview.js”, so that the array can be reused.

Call the function with the games array.

The result should look like this:



Evaluation criteria: In the file overview-table.js you should have

- ❑ Created a new function with parameter “games”, that will create a new table row for each game in the games array.
- ❑ Copied the games array from “games.js” to “table-overview.js”
- ❑ Called the newly created function with the games array as parameter.

5.2 Adding events

With javascript the webpage can be made more interactive. You can define effects that happen when user performs some action like hovering with mouse, clicking etc.

5.2.1 Syntax

```
tableRow.addEventListener("click", () => selectAnimal(animal));
    tableRow.addEventListener("mouseover", () => tableRow.className =
"select");
    tableRow.addEventListener("mouseout", () => tableRow.className = "");
```

The method “addEventListener” makes that something fires when user e.g. clicks on the element. The first parameter (“click”) is the type of the event. The second parameter is the function we want to call when the event occurs.

5.2.2 Exercise: add hover-effect on div#status

When user hovers over the (newly created) div#status, background-color is changed. Take care that background-color disappears when mouse is removed from div.

Evaluation criteria: In the file overview-table.js you should have

- ❑ Added mouseover- and mouseout-event to div#status
- ❑ Added and removed attribute style with setAttribute() and removeAttribute()
- ❑ Or added and removed some class-attribute

5.2.3 Exercise: add click-effect on h2

When the user clicks on the h2 title on our page, javascript should choose a random textcolor for that element. It is not allowed to add an id- or class-attribute to the h2 element.

Use

- Math.floor() and Math.random() to create random number between 0 and 365
- Hsl(number, 100%, 50%) to define a color

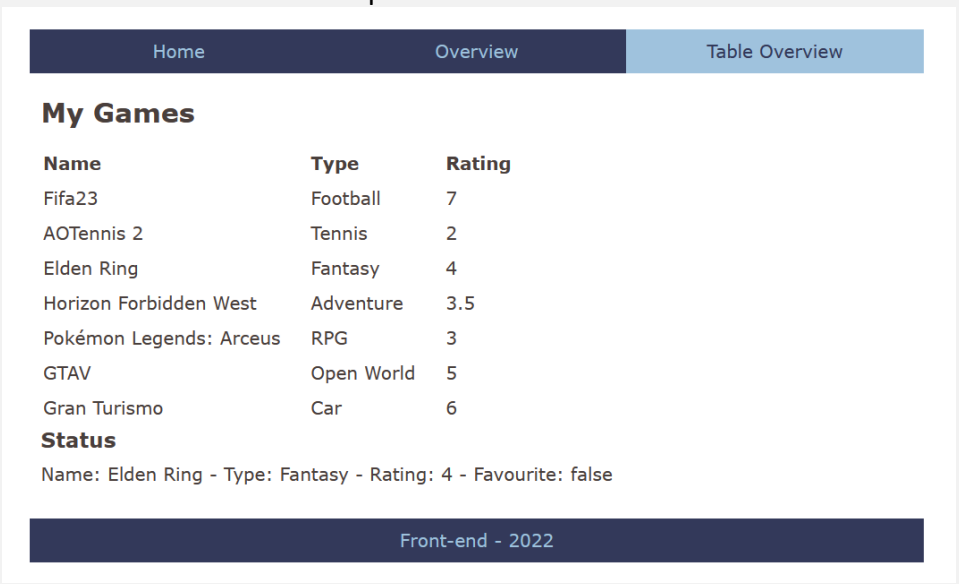
Evaluation criteria: In the file overview-table.js you should have

- ❑ Used querySelector() to select element h2;
- ❑ Defined a function createColor = () => that returns a random color in hsl-format
- ❑ Added a click-event to h2

5.2.1 Exercise: Add click-action on each table row

Our table is visible now, but we want to show all game information in div#status when a user clicks on a game.

Add an event listener to each table row (inside the renderGames function) that will display the games toString information in the “div#status” element whenever a user clicks on that specific row.



Evaluation criteria: In the file overview-table.js you should have

- ❑ Updated the renderGames function to add a “click” event listener to each created row
- ❑ The event listener should display all the games information in the “div#status” element.
- ❑ When second row is clicked, information of first game is removed.

5.3 Adding filtering

Let's make the overview page more interactive. We can add some input field to the overview page. When user types some characters in it, only animals with name including these characters are shown.

```
const renderAllAnimals = () => {
  const chars = document.getElementById("field").value;
  animals
    .filter((animal)=>animal.name.includes(chars))
    .forEach((animal) => {…}
}
```

The variable "chars" reads the characters user typed in the input field with id "field". Then only those animals with name containing chars are filterd by the higher order function "filter". Finally, something happens with each of those animals (defined in "{…}").

5.3.1 Exercise: Show only favorite games

Create a button in overview-table.html with title "Show my Favourites". When user clicks on this button, only favourite games are shown in the table. Also create a button "Show all" that resets the list.

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

- ❑ Created two button-elements in overview-table.js
- ❑ Added click-event on both buttons
- ❑ Added parameter “filterFunction” on the function renderGames
- ❑ Added filter on array: games.filter(()=>)
- ❑ When no filterFunction is passed to renderGames function, all games are shown

5.3.2 Exercise: Show games with rating larger than given value

Create input field in overview-table.html. When the user fills out some number, only games with a rating higher than the input value are shown. The values are shown in real time, i.e. while typing.

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

- ❑ Added an input event listener to the rating input field. When the user types a new rating, call the renderGames function with a filterFunction that uses the entered rating.