Exercises for Session 4

Goals

We're going to put together some puzzle pieces and build our first **real** app. A search for Star Wars characters (or beers if you want to rather do that 🍻).

Building the HTML, attaching some listeners and manipulate it

Build the following HTML layout and sprinkle some CSS on top make it look nice:

```
<form>
    <input />
    <button>Search</button>
</form>
```

- Select the form element using `document.getElementById` (you need to give it an id)
- Add a `submit` listener to the form element
- The function that is executed when the listener is called should do the following:
  - Select the `input` (e.g. again using `getElementById`)
  - `console.log("You searched for:", inputElement.value)`

> Tip: to use the form more dynamically and prevent the default behaviour of browsers to send forms to a remote server your handle should look like this:

```
formElement.addEventListener('submit', event => {
    event.preventDefault()
    // rest of your code
})
```

Using fetch to get some real data

- Instead of using `console.log` to print the search query we're sending to an API with the following url: `https://swapi.co/api/people/?search=${searchQuery}`
- Transform the response into json as soon as you receive it (`then(response => response.json()))`
- Once you have the json data from the API log it once again: `console.log("Server response:", searchResult)`

> Tip: if you want to see the search result in your browser enter the url in a new tab (e.g. https://swapi.co/api/people/?search=luke)

Displaying the result dynamically

- The data we want to display is the array `searchResult.results`. Use `.map()` to create a link from every element in the array:

```
searchResult.results.forEach(result => {
    // create element "a"
    // a.href = url of result
    // a.innerText = name of the result
    // append the a to the body of the document
})
```

- Add some CSS to make the links appear each on a new line and have some margin between each other.

> Liked the exercise but rather want to search for beers? Use `$ curl https://api.punkapi.com/v2/beers?beer_name=${searchQuery}` to get some refreshing hop juices instead of Star Wars characters.

---

localStorage

We're going to enhance the mini-app we just build with some data storage

- After a user searches for something take the value and save it as value into localStorage
    - First read and parse the current value from localStorage: `let queries = JSON.parse(localStorage.getItem('queries'))`
    - If that data is `null` set it to an empty array `[]`
    - `push()` the current value to the array
    - Save the array into localStorage (`localStorage.setItem('queries', queries))
        - The line will not work as you need to `stringify()` `queries` when writing it into localStorage
- You can see the values being written into localStorage by checking the `Application` tab of your browser's dev tools. Refresh the page to ensure the data is still there.
- To remind the user of what they already searched for we want to display the past searches as text below the search form
    - In a `script` tag at the end of the body, read and parse all `queries` from localStorage (see above on how to do that)
    - if the `length` of `queries` is larger than 0 create string from all queries using `join`: `const pastSearches = queries.join(',')`
    - create a text node `document.createTextNode("Previous searches: " + pastSearches)` and append it to the body