

## Exercise 1

Selektieren Sie das <h1>-Element durch die Variante „Drilling into the DOM“ aus und speichern Sie das gesamte Element in einer Variablen mit einem Namen „h1Element“.

## Exercise 2

Selektieren Sie ausgehend von der Variablen „h1Element“ dessen „Parent-Element“.

## Exercise 3

Selektieren Sie ausgehend von der Variablen „h1Element“ dessen nächsten „Sibling-Element“.

## Exercise 4

Selektiere das <h1> Element mit getElementById und speichere das Element in der Variable „h1Element“.

## Exercise 5

Selektiere das zweite <p> Element mit „querySelector“ und speichere es in der Variable „highlightedParagraph“.

## Exercise 6

Ändere den Text des zweiten <p> (siehe Exercise 5) auf „This was changed via JavaScript!“.

## Exercise 7

Füge ein neues Element zum DOM hinzu! Gehe dazu schrittweise vor:

- 1) Erstelle ein neues <a> Element. Der Link soll auf <https://google.com> verweisen. Der Link soll mit „This leads to Google!“ beschrieben sein.
- 2) Der Link soll in den ersten <p> eingefügt werden. Selektiere deshalb das <p> Element und speichere das Element in der Variable „firstParagraph“.
- 3) Füge das neue <a> Element ein.

Die Seite sollte mittlerweile so aussehen:

# JS und der DOM

Erster Absatz. [This leads to Google!](#)

This was changed via JavaScript!

## Exercise 8

Entferne die Überschrift (= <h1> Element) aus dem DOM mit Hilfe von JavaScript!

## Exercise 9

Verändere die Reihenfolge! Tausche die Reihenfolge der beiden <p>. Recherchiere nach einer möglichen Lösung.

This was changed via JavaScript!

Erster Absatz. [This leads to Google!](#)

### Exercise 10

Beantworte folgende Fragen (kurz und in eigenen Worten):

Was ist der DOM? Warum ist das Konzept DOM in der JS-Entwicklung so wichtig?

### Exercise 11

Erweitere das HTML-File um einen dritten Absatz (<p>). Im Absatz soll „Ich bin klickbar!“ stehen. Klickt man auf den Absatz, soll sich der Text auf „Clicked!“ ändern. Gehe Schrittweise vor:

- 1) Füge den Absatz in das HTML File hinzu
- 2) Selektiere den Absatz via JS
- 3) Füge den passenden Event-Listener hinzu.
- 4) Schreibe eine Funktion, die auf den Event reagiert und den Text ändert.

### Exercise 12

Erweitere das Beispiel aus Exercise 11. Zeigt der Absatz „Ich bin klickbar!“ soll nach dem Klick „Clicked!“ im <p>-Element sichtbar sein. Zusätzlich soll nun, wenn erneut auf den <p> geklickt wird, der Text wieder zurück auf „Ich bin klickbar!“ wechseln.

### Exercise 13

Erweitere das HTML-File mit einem <input>-Element mit dem type="text".

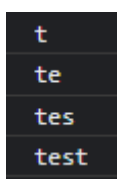
This was changed via JavaScript!

Ich bin klickbar!

Erster Absatz. [This leads to Google!](#)

Füge einen Event-Listener hinzu, der auf die Eingabe in das Input Feld „reagiert“.

Gibt man einen Wert in das Input-Element ein, soll dieser auf der Konsole ausgegeben werden.

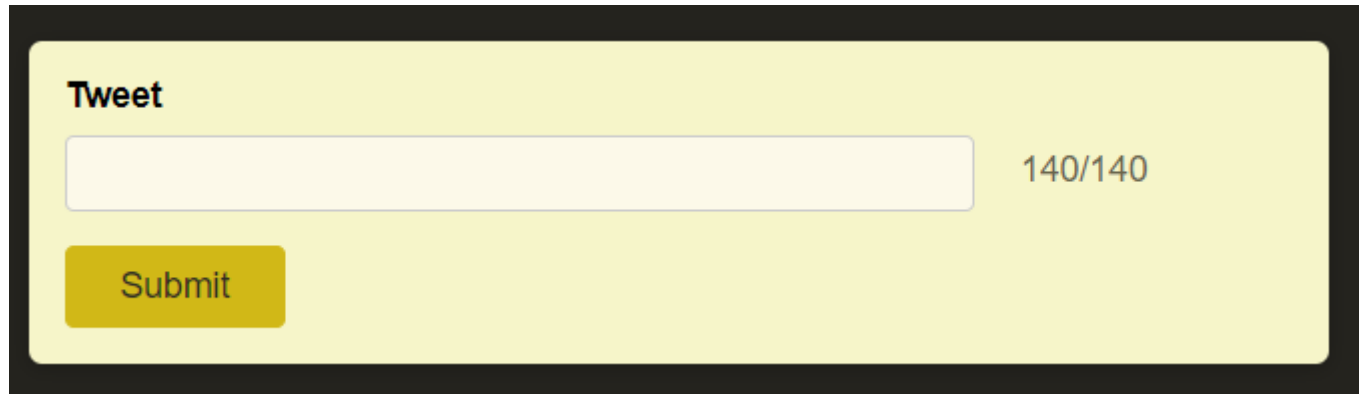


## Exercise 14

Was ist falsch an folgendem Code:

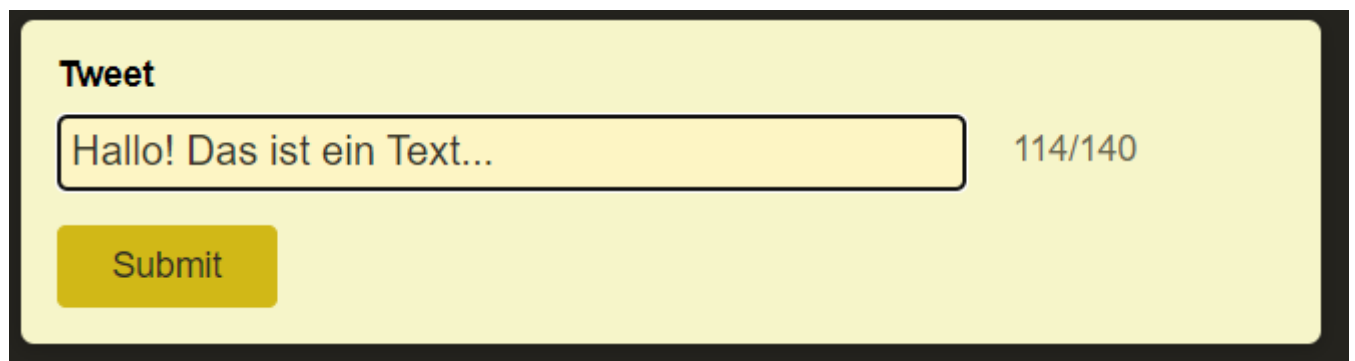
```
inputElement.addEventListener('click', doSomething());
```

## Exercise 15 - Twitter



A screenshot of a Twitter form. It has a yellow background with a black border. At the top left, the word "Tweet" is written in bold. Below it is a large, empty text input field. To the right of the input field, the text "140/140" is displayed. Below the input field is a yellow button with the word "Submit" in black text.

Ein Tweet ist mit 140 Zeichen begrenzt. Der User soll während der Eingabe wissen, wie viele Zeichen noch verfügbar sind.



A screenshot of the same Twitter form, but now the input field contains the text "Hallo! Das ist ein Text...". The character count on the right has updated to "114/140". The "Submit" button remains yellow with black text.

- 1) Erstelle einen neuen Ordner und füge folgende Dateien ein.
  - a. Index.html
  - b. Styles.css
  - c. App.js
- 2) Erstelle das HTML Dokument
- 3) Implementiere die Logik mit JS
  - a. Selektiere das Input-Feld
  - b. EventListner
  - c. Callback-Funktion mit Logik
- 4) Styling mit CSS