**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:

Ein Bild, das Text, drinnen enthält.

Automatisch generierte Beschreibung

**Exercise 8**

Entferne die 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.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**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-Listner 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.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Füge einen Event-Listner 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.



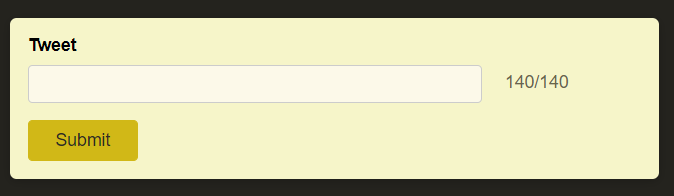


**Exercise 14**

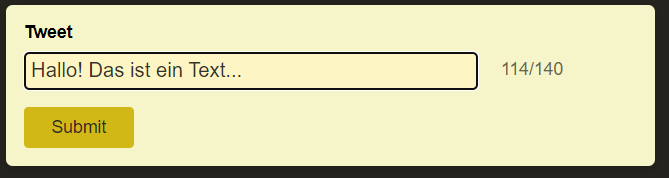
Was ist falsch an folgendem Code:



**Exercise 15 - Twitter**



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



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