Übung 03

1 Einführung JavaScript

JavaScript ist eine Skriptsprache, welche es ermöglicht die Inhalte von HTML und CSS dynamisch zu gestalten. Hierzu gehören unter anderem das nachträgliche verändern, neuladen oder entfernen von Inhalten. Als kleine Einführung in JavaScript schauen wir uns folgenden Code für eine Auflistung von Nachrichten ähnlich einer E-Mail Inbox an.

Listing 1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Zahlenratespiel</title>
  <style>
    .highlight {
      color: red;
      font-style: italic;
    }
  </style>
</head>
<body>
  <div>
    <p>Inbox</p>
    <ul>

    </ul>
  </div>
  <script>
    var messages = [
      {
        "subject": "first message",
        "flag": ""
      },{
        "subject": "second message",
        "flag": ""
      },{
        "subject": "third message",
```

```
      "flag": "seen"
    },{
      "subject": "spam message",
      "flag": ""
    },{
      "subject": "another message",
      "flag": "seen"
    }
  ];
</script>
</body>
</html>
```

Erweitern wir das JavaScript im `<script>`-Tag zunächst um eine Funktion, die für jede Nachricht im Array `messages` einen Listeneintrag `<li>` zur ungeordneten Liste hinzufügt. Wir nutzen hierbei das Konzept des Data Attributes, um jedem Listeneintrag zusätzlich die Information über den Lesestatus anzuhängen (`flags`). Wir beginnen mit der Erstellung eines Listenelements in der die Nachrichten angezeigt werden sollen. Hierzu wählen wir das bereits vorhandene `<ul>` aus dem `body`-Element unseres HTML-Dokuments aus.

```
var list = document.querySelector("div ul");
```

Für eine genauere Beschreibung der Funktion `querySelector()` schlagen Sie diese im Bereich Hilfestellungen nach. Im nächsten Schritt wollen wir nun die `messages` zu unserer `list` hinzufügen. Zu diesem Zweck nutzen wir eine `forEach`-Schleife, die über alle Elemente in dem Array iteriert.

```
messages.forEach( function(messages) {
// Inhalt der for-Schleife wird hier eingefügt
});
```

Die anonyme Funktion wird hierbei von jeder Nachricht in dem Array `messages` aufgerufen. Erstellen wir nun eine Funktionsinterne Hilfsvariable in der wir die Nachrichten und dessen `flags` als Listeneintrag `<li>` speichern, um diese anschließend in unserer Webseite anzuzeigen. Da die Hilfsvariable nur innerhalb der Funktion benötigt wird, nutzen wir die Deklaration `let`.

```
messages.forEach( function(messages) {
  let listItem = document.createElement("li");
  listItem.textContent = message.subject;
  listItem.dataset.flag = message.flag;
  list.appendChild(listItem);
});
```

Die letzte Funktion `list.appendChild(listItem)` fügt den Listeneintrag der Liste innerhalb des HTML-Dokuments (DOM) hinzu. Somit werden nun unsere Nachrichten aus `messages` auf der Webseite innerhalb einer ungeordneten Liste angezeigt. Aufbauend darauf wollen wir nun eine Funktion, die für alle Listeneinträge überprüft, ob diese eine gelesene

oder ungelesene Nachricht darstellen, implementieren. Ungelesene Nachrichten sollen mit der CSS-Klasse `highlight` versehen werden, nachdem sie auf die Webseite geladen wurden. Für die Umsetzung implementieren wir daher eine Funktion `highlightUnread()`, in der wir zuerst alle Listenelemente `<li>` selektieren.

```
var listItems = document.querySelectorAll("li");
```

Anschließend iterieren wir über alle selektierten Listenelemente und prüfen, mithilfe eine if-Bedingung, ob diese den `flag` ungelesen"besitzen. Ist dies der Fall, setzen wir die CSS-Klasse des Elements auf `highlight`.

```
listItems.forEach(function(listItem) {
  if (listItem.dataset.flag != "seen") {
    listItem.classList.add("highlight");
    }
 });
```

Zuletzt müssen wir unsere neue Funktion noch aufrufen, nachdem die Nachrichten an der Webseite hinzugefügt werden. Das vollständige Script sieht anschließend wie folgt aus:

Listing 2

```
<body>
  <div>
    <p>Inbox</p>
    <ul>

    </ul>
  </div>
  <script>
    var messages = [
      {
        "subject": "first message",
        "flag": ""
      },
      {
        "subject": "second message",
        "flag": ""
      },
      {
        "subject": "third message",
        "flag": "seen"
      },
      {
        "subject": "spam message",
        "flag": ""
      },
      {
        "subject": "another message",
        "flag": "seen"
      }
    ];

    function highlightUnread() {
```

```
      var listItems = document.querySelectorAll("li"); // Selektiere alle
      ↪ Listenelemente
      listItems.forEach(function(listItem) { // Iteriere alle Listenelemente
        if (listItem.dataset.flag != "seen") { // Prüfe data-flag
          listItem.classList.add("highlight"); // Setze Klasse 'highlight'
        }
      });
    }

    // Erzeuge Listenelemente für Nachrichten
    var list = document.querySelector("div ul");
    messages.forEach(function(message) {
      let listItem = document.createElement("li");
      listItem.textContent = message.subject; // Betreffzeile der Nachricht
      listItem.dataset.flag = message.flag; // Hänge data-flag an
      list.appendChild(listItem); // Hänge Listenelement an DOM an
    });

    highlightUnread(); // Markiere Nachrichtenelemente
  </script>
</body>
```

2 Aufgaben

Bearbeiten Sie nun die folgenden Aufgaben, nutzen Sie hierfür den zuvor erstellten Code oder erstellen Sie ein neues HTML-Dokument. Beachten Sie die Hilfestellungen und nutzen Sie `var`, `let` und `const` auf sinnvolle Art.

Aufgabe 1

Implementieren Sie ein Zahlenratespiel. Ein Eingabefeld soll die Eingabe einer Zahl ermöglichen und bei Klick auf einen Button „Prüfen“ soll die Eingabe mit einer beim Laden der Seite zufällig generierten Zahl zwischen 1 und 100 verglichen werden.

Errät der Nutzer die Zahl nicht innerhalb von zehn Versuchen, ist das Spiel verloren. Alle Eingaben sollen auf der Seite angezeigt werden, sowie eine Information darüber, ob die letzte eingebene Zahl größer oder kleiner als die zu erratende Zahl ist.

Aufgabe 2

Erweitern Sie die Implementierung aus Aufgabe 2, sodass eine Fehlermeldung geworfen wird wenn die eingegebene Zahl nicht zwischen 1 und 100 liegt oder keine Zahl eingegeben wur-

de. Nutzen Sie dafür eine `try-catch`-Anweisung. Geben Sie für beide Fälle unterschiedliche Fehlermeldungen aus.

3 Hilfestellungen

Nutzen Sie folgende Hilfestellungen zur Lösung der Aufgaben:

- `element = document.querySelector(selectors);`
  Erlaubt den direkten Zugriff auf ein DOM-Element über die Variable `element`. Der Query-String kann dabei beispielsweise eine ID (`#`), eine Klasse (`.`) oder den Elementtyp selektieren.

- `elementList = parentNode.querySelectorAll(selectors);`
  Gibt eine Liste von allen Elementen zurück auf die der übergebene Query-String zutrifft.

- `arrayObj.forEach(function(element) {...});`
  Ermöglicht die Iteration über alle Elemente eines Arrays. Die anonyme Funktion wird dabei für jedes `element` des Arrays aufgerufen.

- `domElement.dataset.<value>`
  Erlaubt die Zweisung und das Auslesen von Data Attributes. Data Attributes speichern nur Strings (kompatible Datentypen werden implizit konvertiert) und verändern nicht die Darstellung eines DOM-Elements.

- `domElement.classList.add(<string>)`
  Fügt einem DOM-Element eine CSS-Klasse hinzu. DOM-Elemente können mehreren CSS-Klassen zugewiesen werden.

- `document.createElement(<string>);`
  Erstellt ein DOM-Objekt, definiert durch den übergebenen String, beispielsweise `"div"`.

- `domElement.appendChild(childElement);`
  Hängt einem DOM-Element ein Kindelement an, z.B. um ein Listenelement zu einer Liste hinzuzufügen.

- `{element.addEventListener('click', func);`
  Bindet einen Event-Listener an `element`. `func` ist dabei die Funktion, die beim Auslösen des Events aufgerufen wird.

- Eine Zufallszahl lässt sich mit Hilfe von `Math.random()` und `Math.floor()` gene-

rieren. `random()` erzeugt eine Fließkommazahl zwischen 0 und 1. `floor()` gibt die nächstkleinere Ganzzahl zu einer Fließkommazahl aus. Denken Sie an den geforderten Wertebereich der Zufallszahl zwischen 1 und 100.

- `element.value`
  Gibt den eigegebenen Wert für ein Eingabefeld (element) zurück.

- `element.textContent`
  Erlaubt, den Inhalt eines DOM-Elements mit einer Zuweisung (`=`) zu setzen, oder mit `+=` zusätzlichen Inhalt anzuhängen.

- `Number(value)`
  Ist ein Objektkonstruktor der Number Wrapperklasse und kann zur Typkonvertierung für einen String mit numerischem Inhalt genutzt werden.