# LU01e - DOM

Der Document Object Model (DOM) ist eine Programmierschnittstelle für HTML und XML-Dokumente. Es stellt das Dokument als Baumstruktur dar, bei dem jedes Element (wie ein HTML-Tag) ein Knoten ist. Durch den DOM können wir auf HTML-Elemente zugreifen und sie dynamisch mit JavaScript verändern. Jedes HTML-Element kann als ein Objekt betrachtet werden, das verschiedene Eigenschaften und Methoden hat, mit denen man es verändern kann.

## Zugriff auf DOM-Elemente

Um DOM-Elemente zu modifizieren, müssen wir zuerst auf sie zugreifen. Dies geschieht meist über die Methode document in JavaScript.

**getElementById()** Diese Methode wird verwendet, um ein Element anhand seiner id-Eigenschaft auszuwählen. Jede ID im HTML-Dokument muss einzigartig sein.

```
const element = document.getElementById("meinElement");
```

**getElementsByClassName()** Diese Methode wählt alle Elemente aus, die eine bestimmte Klasse haben. Es gibt eine Sammlung (Array-ähnlich) von Elementen zurück.

```
const elements = document.getElementsByClassName("meineKlasse");
```

**querySelector() und querySelectorAll()** Diese Methoden erlauben eine flexible Auswahl von DOM-Elementen mittels CSS-Selektoren. querySelector() gibt das erste passende Element zurück, während querySelectorAll() eine Liste aller passenden Elemente liefert.

```
const element = document.querySelector(".meineKlasse #meinElement");
```

## Modifizieren von DOM-Elementen

### Ändern von Inhalten

Um den Text oder HTML-Inhalt eines Elements zu ändern, gibt es zwei Hauptmethoden:

**textContent** Verändert den reinen Text eines Elements, ohne HTML-Tags zu interpretieren.

```
document.getElementById("meinElement").textContent = "Neuer Text";
```

**innerHTML** Erlaubt das Setzen von HTML-Inhalten innerhalb eines Elements.

```
document.getElementById("meinElement").innerHTML = "<strong>Fetter
Text</strong>";
```

### Ändern von Attributen

Mit der Methode setAttribute() können wir Attribute von HTML-Elementen ändern. Zum Beispiel, um das src-Attribut eines Bildes zu ändern:

```
document.getElementById("meinBild").setAttribute("src", "neuesBild.jpg");
```

Ebenso können wir das class- oder id-Attribut eines Elements ändern:

```
document.getElementById("meinElement").setAttribute("class", "neueKlasse");
```

### CSS-Stile ändern

Mit der Eigenschaft style können wir die CSS-Stile eines Elements direkt in JavaScript verändern.

```
document.getElementById("meinElement").style.backgroundColor = "blue";
```

Mann kann so ziemlich jeden CSS-Stil über JavaScript ändern. Wichtig: In JavaScript werden mehrteilige CSS-Eigenschaften wie background-color in Kamel-Schreibweise (camelCase) geschrieben, also backgroundColor.

### Elemente hinzufügen und entfernen

Um ein neues Element in den DOM einzufügen, nutzt man die Methode createElement() und anschließend appendChild(), um das Element an eine bestehende Struktur anzuhängen.

```
const neuesElement = document.createElement("p");
neuesElement.textContent = "Dies ist ein neuer Absatz.";
document.body.appendChild(neuesElement);
```

Um ein Element zu entfernen, nutzt man removeChild():

```
const element = document.getElementById("meinElement");
element.parentNode.removeChild(element);
```

## Ereignisse (Events) und Funktionen mit Buttons auslösen

DOM-Elemente können auf Benutzereingaben reagieren, indem sie Events (Ereignisse) auslösen. Ein Event ist eine Aktion, die im Browser stattfindet, wie z.B. ein Mausklick oder ein Tastendruck.

Häufig genutzte Events:

- **click:** Wird ausgelöst, wenn ein Element angeklickt wird.
- **input:** Wird ausgelöst, wenn der Benutzer Eingaben in ein Formularfeld macht.
- **mouseover:** Tritt auf, wenn der Mauszeiger über ein Element bewegt wird.

Um auf ein Event zu reagieren, verknüpfen wir eine Funktion (auch als Event-Handler bezeichnet) mit einem DOM-Element. Dies geschieht meistens durch die Methode addEventListener().

```
<button id="meinButton">Klick mich</button>
```

```
document.getElementById("meinButton").addEventListener("click", function() {
  alert("Button wurde geklickt!");
});
```

Die Methode addEventListener() wird verwendet, um ein Ereignis (in diesem Fall click) mit einem DOM-Element zu verknüpfen. Wenn der Button angeklickt wird, wird die Funktion ausgeführt, die ein Popup mit der Nachricht „Button wurde geklickt!“ ausgibt.