LU08c - DOM-Methoden

Lernziele

1. Verschiedene DOM-Selektor-Methoden unterscheiden können: Sie kennen Zweck, Einsatzgebiet und Unterschiede von getElementById(), getElementsByClassName(), getElementsByTagName() und querySelector().
2. HTML-Elemente gezielt auswählen und verändern können: Sie wenden jede Methode korrekt im Quellcode an, greifen auf einzelne Elemente oder Elementgruppen zu und manipulieren Inhalte oder Styles.
3. Situationsgerecht den passenden Selektor auswählen können: Sie entscheiden selbstständig, welche Methode für eine konkrete Aufgabenstellung am sinnvollsten ist – effizient, korrekt und verständlich.

JavaScript DOM-Selektoren kompakte Übersicht

Die folgenden Methoden dienen dazu, HTML-Elemente im Dokument (DOM) zu finden und zu manipulieren. Jede Methode arbeitet etwas anders und eignet sich für unterschiedliche Zwecke.

- getElementByID
- getElementsByClassName
- getElementsByTagName
- querySelector

Die in der Demonstration verwendeten Beispielscripte finden Sie hier

lu08_demo.zip

---

1. document.getElementById()

- Findet genau ein Element anhand seiner ID.
- IDs sind im Dokument eindeutig – ideal ein Element gezielt angesprochen werden soll.

Syntax

```
document.getElementById("idName");
```

Codebeispiel

```
<p id="titel">Hallo Welt</p>
<script>
    document.getElementById("titel").innerHTML = "Text erfolgreich
geändert!";
</script>
```

2. document.getElementsByClassName()

- Liefert alle Elemente einer bestimmten CSS-Klasse als HTMLCollection (ähnlich wie ein Array).
- Für einzelne Elemente muss man über den Index zugreifen:

Syntax

```
document.getElementsByClassName("className")[index];
```

Codebeispiel

```
<p class="info">Erster Text</p>
<p class="info">Zweiter Text</p>
<script>
    document.getElementsByClassName("info")[0].innerHTML = "Ich bin
geändert!";
    document.getElementsByClassName("info")[1].style.color = "red";
</script>
```

3. document.getElementsByTagName()

- Gibt alle Elemente eines HTML-Tags zurück, ebenfalls als HTMLCollection.
- Praktisch bei der Bearbeitung ganzer Elementgruppen (<p>, <li>, <button> ...).

Syntax

```
document.getElementsByTagName("tagName")[index];
```

Codebeispiel

```
<p>Text A</p>
<p>Text B</p>
<script>
    let alleAbschnitte = document.getElementsByTagName("p");
    alleAbschnitte[1].innerHTML = "Ich bin der zweite Absatz – verändert!";
</script>
```

4. document.querySelector()

- Findet das erste Element, das zu einem CSS-Selektor passt.
- Modern, flexibel, unterstützt komplexe Selektoren (ID, Klasse, Attribute ...).

Syntax

```
document.querySelector("css-selector");
```

Codebeispiel

```
<p id="a">1</p>
<p class="b">2</p>
<p class="b">3</p>
<script>
    document.querySelector(".b").style.fontWeight = "bold";
    // triff nur das erste .b, alle anderen werden ignoriert
</script>
```

Codebeispiel

Damit wir alle p-Elemente verändern, braucht es eine Schleife.

```
    const alleP = document.querySelectorAll("p");
    // jedem die neue Klasse geben
    alleP.forEach(p => p.className = "beschreibungN");
```

Zusatzmaterial

- W3School - HTML-DOM
- SelfHTML- HTML DOM
