DOM-Modell und JavaScript

Öffnen Sie folgende HTML-Datei (01_Uebungen_DOM_vorlage)

index.html > ...

```html
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Javascript Start</title>
    <link rel="stylesheet" href="styles.css">

</head>
<body id="bd">
    <div id="a1">
        <div id="a2">Dies ist ein <em>Text</em>...</div>

    </div>
    <div id="b1"></div>

</body>
</html>
```

1. Woran erkennt man, dass es sich um eine HTML5-Datei handelt?
2. Stellen Sie mit Hilfe z. B. der Chrome DevTools die HTML-Datei dar.
3. Stellen Sie die HTML-Datei auf einem Blatt strukturiert als DOM-Baum dar. Unterscheiden Sie dabei die HTML-Elemente (Objekte) und die Text-Knoten.
4. Mit HTML werden Dokumente strukturiert. Um das Aussehen zu bestimmen, verwenden Sie Cascading Style Sheets (CSS). CSS ist eine einfache Formatierungssprache, die prinzipiell aus *Selektoren, Eigenschaften und Werten* besteht. Versehen Sie Ihr Beispiel mit den folgenden Styles:

styles.css > #a1

```css
body {
    font-family: Calibri, 'Trebuchet MS', sans-serif;
    font-size: 1.4em;
    color: #006;
}
#a1 {
    background-color: rgb(228, 144, 158);
    width: 300px;
    padding: 10px;
    border: 1px solid #006;
}
```

Dies müsste zu folgendem Aussehen führen:

Dies ist ein *Test*...

6. Wie lassen sich die einzelnen Elemente unseres Beispiels ansprechen? Am einfachsten geht, wenn sie eine ID haben. Dann können die Elemente mit der folgenden Methode des Document-Objekts angesprochen werden: getElementById(). Wir probieren dies an unserem Beispiel aus.

7. Erstellen Sie eine JavaScript-Datei und fügen Sie die Verlinkung zur Datei vor dem Ende des bodys hinzu.

```
15      <div id="b1"></div>
16    <script src="javascript.js"></script>
17  </body>
18  </html>
```

```
JS javascript.js > ...
1   "use strict"
2
3   document.addEventListener("DOMContentLoaded", () => {
4       console.log("Hello World!");
5       console.log(document.getElementById("a1").innerHTML);
6   })
```

8. Eleganter funktioniert das natürlich mit Funktion und Variable. Vergleichen Sie anschließend. Was bedeutet getElementById(„a1“) und was bedeutet getElementById(„a1“).innerHTML?

_____

_____

Lösung:

```
JS javascript.js > ...
1   "use strict"
2
3   document.addEventListener("DOMContentLoaded", () => {
4       console.log("Hello World!");
5       let ausgabe = document.getElementById("a1");
6       console.log(ausgabe);
7       console.log(ausgabe.innerHTML);
8   })
```

9. InnerHTML kann auch verändert werden. Dies ist ein einfacher Weg, um HTML-Code an einer bestimmten Stelle einzufügen. Bezogen auf unser Beispiel:

JS javascript.js > ...

```
1   "use strict"
2
3   document.addEventListener("DOMContentLoaded", () => {
4       console.log("Hello World!");
5       let ausgabe = document.getElementById("a1");
6       console.log(ausgabe);
7       console.log(ausgabe.innerHTML);
8       ausgabe.innerHTML += "<br>und das finde ich toll!";
9       console.log(ausgabe.innerHTML);
10  })
```

10. Lassen Sie sich nun alle Elemente mit getElementsbyTagName(„div“) in der Console anzeigen? Was ist das Ergebnis?

```
let container = document.getElementsByTagName("div");
console.log(container);
```

11. Lassen Sie sich über eine alert-Box jeweils folgendes ausgeben:

```
alert(container[0].id);
alert(container[0].innerHTML);
alert(container[0].innerText);
alert(container[0].offsetHeight);
```

12. Betrachten Sie auch in der Console alle Eigenschaften.

13. Erstellen Sie eine Schleife, so dass Sie folgende Ausgabe haben:

Dies ist ein *Test* ...

Das 1te Element hat die id a1
Das 2te Element hat die id a2
Das 3te Element hat die id b1

Elemente verändern

Mit JavaScript kann man z. B.

1. Inhalte von HTML-Inhalten dynamisch ändern

```
let x1=document.getElementById("a2");
let y1=x1.childNodes[1].childNodes[0];
y1.nodeValue="Versuch";
```

2. das Aussehen von Inhalten dynamisch ändern

```
x1.style.fontSize="1.8em";
x1.style.color = "#b80926";
```

3. neue HTML-Objekte hinzufügen

   Mit JavaScript kann man einem DOM-Baum neue Knoten hinzufügen oder auch bestehende Knoten entfernen. Die Methode createTextNode() erzeugt einen neuen Textknoten. Dieser wird zunächst nicht angezeigt, da er dafür zuerst in den DOM-Baum eingebunden werden muss. Dafür sorgt die Methode appendChild().

```
let b1 = document.getElementById("b1");
let content = document.createTextNode("Hier steht der Inhalt von B1");
let paragraph = document.createElement("p");
paragraph.appendChild(content);
b1.appendChild(paragraph);
```

4. HTML-Objekte an eine andere Stelle verschieben

   Die Methode appendChild() kann auch im Zusammenhang mit Objekten, die bereits im DOM-Baum eingebunden sind, verwendet werden. Möchten Sie die Objekte an einer anderen Stelle einfügen, können Sie die Methode insertBefore() verwenden.

5. HTML-Objekte aus der Hierarchie löschen

   removeChild

Übung