Das DOM verstehen

Lernziel

Nach dieser Lektion weißt du, was das **DOM** ist und warum JavaScript damit sichtbare Inhalte einer Webseite lesen und verändern kann. Du verstehst außerdem, dass eine HTML-Seite im Browser als Baumstruktur dargestellt wird.

Was bedeutet DOM?

DOM steht für *Document Object Model*, auf Deutsch etwa: „Dokument-Objekt-Modell“.

Wenn der Browser eine HTML-Datei lädt, liest er deren Elemente ein und erstellt daraus eine Struktur im Speicher. Diese Struktur nennt man DOM. JavaScript kann auf sie zugreifen.

Aus diesem HTML-Code:

```
<body>
  <h1>Willkommen</h1>
  <p>JavaScript macht Webseiten interaktiv.</p>
</body>
```

entsteht gedanklich eine Baumstruktur:

```
document
└── html
    ├── head
    └── body
        ├── h1
        └── p
```

Jedes HTML-Element wird dabei zu einem Objekt im DOM. JavaScript kann diese Objekte untersuchen, verändern, hinzufügen oder entfernen.

> **Merke:** HTML beschreibt den ursprünglichen Aufbau der Seite. Das DOM ist die Version dieser Seite, mit der der Browser und JavaScript gerade arbeiten.

Das globale Objekt `document`

JavaScript erhält über das Objekt `document` Zugriff auf die geladene Webseite.

```
console.log(document);
```

Gib diesen Befehl in der Browser-Konsole ein. Du siehst die Struktur des aktuellen Dokuments.

`document` steht für die gesamte HTML-Seite. Es enthält unter anderem:

- das `<html>`-Element,
- den Bereich `<head>`,
- den sichtbaren Bereich `<body>`,
- alle Überschriften, Absätze, Bilder, Buttons und Formulare.

Du kannst beispielsweise den Seitentitel auslesen:

```
console.log(document.title);
```

Der Titel stammt aus dem `<title>`-Element im `<head>`:

```
<head>
  <title>Meine erste Webseite</title>
</head>
```

Ändere nun den Titel per JavaScript:

```
document.title = "JavaScript üben";
```

Im Browser-Tab erscheint sofort der neue Titel.

HTML und DOM sind nicht ganz dasselbe

HTML ist der Code, den du in einer Datei schreibst:

```
<p>Hallo!</p>
```

Das DOM entsteht daraus erst, wenn der Browser die Datei verarbeitet. JavaScript verändert normalerweise das DOM — nicht direkt die gespeicherte HTML-Datei.

Schau dir dieses Beispiel an:

```
<h1 id="ueberschrift">Willkommen</h1>
```

Wenn JavaScript später den Text verändert, sieht die Seite anders aus. Die ursprüngliche HTML-Datei auf deinem Computer bleibt jedoch unverändert.

```
document.querySelector("#ueberschrift").textContent = "Schön, dass du da bist!";
```

Nach dem Neuladen der Seite wird wieder der ursprüngliche HTML-Code geladen, sofern du die HTML-Datei nicht selbst gespeichert und geändert hast.

Elemente, Attribute und Inhalte

HTML-Elemente bestehen häufig aus mehreren Teilen:

```
<a href="https://example.com" class="link">Mehr erfahren</a>
```

Teil	Bedeutung
`<a>`	Das Element beziehungsweise der Elementtyp
`href`	Ein Attribut mit dem Ziel des Links
`class`	Ein Attribut für CSS oder JavaScript
`Mehr erfahren`	Der Textinhalt des Elements

Im DOM kann JavaScript all diese Informationen lesen und später verändern.

Zum Beispiel hat ein Button einen Text und eine CSS-Klasse:

```
<button class="kaufen-button">Jetzt kaufen</button>
```

JavaScript könnte später:

- den Text in „Bestellt“ ändern,
- eine Klasse ergänzen,
- den Button deaktivieren,
- auf einen Klick reagieren.

Wie du einzelne Elemente gezielt auswählst und veränderst, lernst du in den nächsten Lektionen.

Die Seite im Browser untersuchen

Die Entwicklerwerkzeuge des Browsers helfen dir, das DOM sichtbar zu machen.

1. Öffne eine Webseite im Browser.
2. Drücke **F12** oder klicke mit der rechten Maustaste auf ein Element.
3. Wähle „Untersuchen“ oder „Element untersuchen“.
4. Öffne den Bereich **Elements** oder **Elemente**.

Dort zeigt der Browser die aktuelle DOM-Struktur an. Klappst du Elemente auf, erkennst du ihre Verschachtelung.

Angenommen, deine Seite enthält:

```
<main>
  <h1>Meine Aufgaben</h1>
  <ul>
    <li>Einkaufen</li>
    <li>JavaScript lernen</li>
  </ul>
</main>
```

Dann liegt das `<h1>`-Element innerhalb von `<main>`. Die beiden `<li>`-Elemente liegen innerhalb der Liste `<ul>`.

Diese Beziehungen sind wichtig:

- `<main>` ist das **Elternelement** von `<h1>` und `<ul>`.
- `<h1>` und `<ul>` sind **Kindelemente** von `<main>`.

- Die beiden `<li>`-Elemente sind Kindelemente von `<ul>`.
- Die `<li>`-Elemente sind zueinander **Geschwisterelemente**.

Warum die Baumstruktur wichtig ist

Die Baumstruktur bestimmt, welche Elemente zu welchem Bereich gehören. Das ist besonders bei größeren Webseiten hilfreich.

```
<section class="produkt">
  <h2>Rucksack</h2>
  <p>Praktisch für Alltag und Reise.</p>
  <button>In den Warenkorb</button>
</section>
```

Der Button gehört hier zu einem bestimmten Produktbereich. Eine interaktive Webseite soll später beispielsweise genau diesen Bereich aktualisieren, wenn jemand auf den Button klickt.

Das DOM erlaubt JavaScript, solche Zusammenhänge zu erkennen und gezielt zu nutzen.

Ein erstes vollständiges Beispiel

Lege eine Datei namens `index.html` an:

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>DOM-Beispiel</title>
    <script src="script.js" defer></script>
  </head>
  <body>
    <h1>Meine Lernseite</h1>
    <p>Ich lerne das DOM kennen.</p>
  </body>
```

```
</html>
```

Erstelle daneben die Datei `script.js`:

```
console.log(document);
console.log(document.title);
console.log(document.body);
```

Öffne anschließend die HTML-Datei im Browser und rufe die Konsole auf.

Du solltest Folgendes beobachten:

- `document` repräsentiert das gesamte Dokument.
- `document.title` gibt den Seitentitel aus.
- `document.body` gibt den sichtbaren Inhaltsbereich aus.

Das Attribut `defer` im `<script>`-Element ist hier wichtig:

```
<script src="script.js" defer></script>
```

Es sorgt dafür, dass der Browser zuerst das HTML verarbeitet und das JavaScript danach ausführt. So steht das DOM bereit, wenn dein Skript darauf zugreift.

Veränderungen im DOM beobachten

Öffne in den Entwicklerwerkzeugen den Bereich **Elements**. Gib danach in der Konsole diesen Code ein:

```
document.body.style.backgroundColor = "lightblue";
```

Die Hintergrundfarbe der Seite ändert sich sofort. Im Bereich „Elements“ kannst du sehen, dass der Browser eine Stilinformation zum `<body>`-Element ergänzt hat.

Du hast damit das DOM zur Laufzeit verändert.

Lade die Seite neu. Die Farbe verschwindet wieder, denn die Änderung war nur im aktuellen DOM vorhanden und wurde nicht in deine HTML-Datei geschrieben.

> **Wichtig:** Änderungen durch JavaScript bleiben normalerweise nur bestehen, bis die Seite neu geladen wird.

Zusammenfassung

- Das **DOM** ist die Objektstruktur einer geladenen HTML-Seite im Browser.
- JavaScript greift über `document` auf diese Struktur zu.
- HTML-Elemente bilden im DOM eine Baumstruktur aus Eltern-, Kind- und Geschwisterelementen.
- Über das DOM kann JavaScript Inhalte, Attribute, Klassen und Styles verändern.
- Die Entwicklerwerkzeuge zeigen dir die aktuelle DOM-Struktur.
- Mit `defer` wird dein JavaScript ausgeführt, nachdem der Browser das HTML verarbeitet hat.

Übungen

1. Das Dokument erkunden

Öffne eine beliebige Webseite und gib in der Konsole diese Befehle ein:

```
console.log(document);
console.log(document.title);
console.log(document.body);
```

Frage: Was geben die drei Befehle jeweils aus?

---

2. Den Seitentitel ändern

Erstelle eine HTML-Seite mit einem `<title>`-Element. Ändere den Titel in deiner JavaScript-Datei:

```
document.title = "Mein DOM-Training";
```

Aufgabe: Beobachte, wo sich der Titel im Browser ändert.

3. Die Baumstruktur erkennen

Betrachte diesen HTML-Code:

```
<article>
  <h2>Neuigkeiten</h2>
  <p>Heute beginnt das JavaScript-Training.</p>
  <p>Viel Erfolg beim Lernen!</p>
</article>
```

Fragen:

1. Welches Element ist das Elternelement der beiden Absätze?
2. Welche Elemente sind Geschwister?
3. Welches Element enthält den Text „Neuigkeiten“?

Lösungen:

1. Das `<article>`-Element ist das Elternelement der Absätze.
2. `<h2>` sowie beide `<p>`-Elemente sind Geschwister, weil sie direkt innerhalb von `<article>` liegen.
3. Das `<h2>`-Element enthält den Text „Neuigkeiten“.

Im nächsten Schritt lernst du, wie du einzelne Elemente aus diesem DOM gezielt mit `querySelector()` auswählst.

Revision #1
Created 2026-07-24 13:44:23 UTC by art10m
Updated 2026-07-24 13:44:23 UTC by art10m