DOM manipuláció JavaScript segítségével

A Document Object Model (DOM) a weboldalak szerkezetét írja le egy hierarchikus fa formájában, amelyen keresztül a JavaScript hozzáférhet és módosíthatja az oldal tartalmát és szerkezetét. A DOM manipuláció az egyik legfontosabb eszköz a dinamikus weboldalak fejlesztésében, mivel lehetővé teszi az elemek hozzáadását, eltávolítását, módosítását, valamint az eseménykezelést.

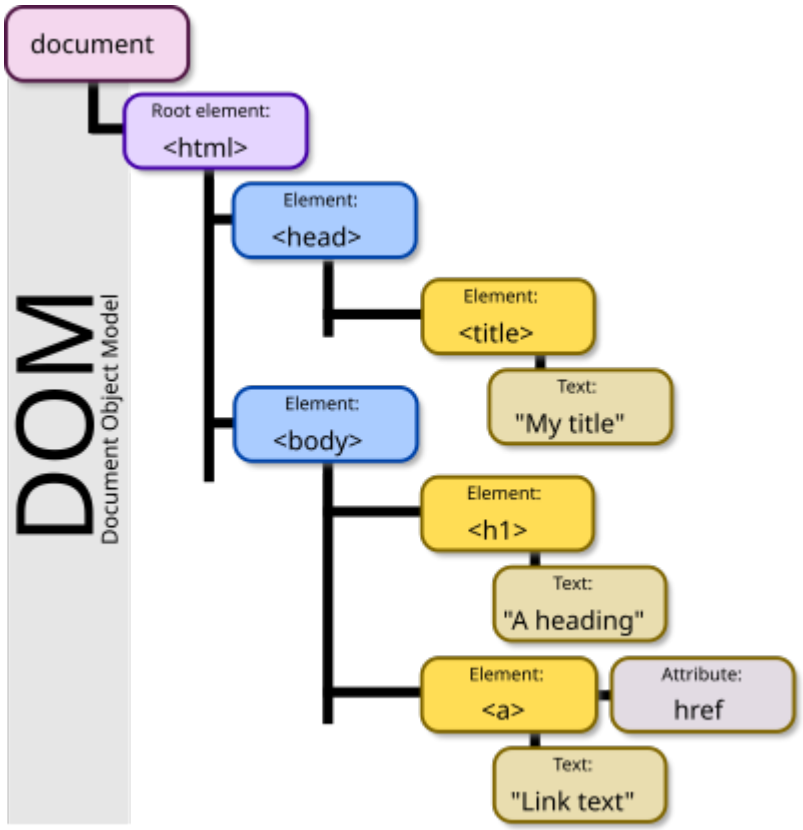
Document Object Model

A **DOM (Document Object Model)** egy platformfüggetlen, objektumalapú reprezentációja egy HTML- vagy XML-dokumentumnak. A DOM egy fa struktúrában írja le az oldal elemeit, ahol minden HTML-címke egy csomópontként (**node**) jelenik meg, és hierarchikus kapcsolatban áll a többi elemmel. A DOM-on keresztül JavaScript segítségével elérhetjük és módosíthatjuk az oldal tartalmát és szerkezetét.

A DOM egy objektummodell, amelyben minden HTML-elem egy objektumként létezik. Például egy egyszerű HTML-oldal:

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="#>Link text</a>
  </body>
</html>
```

A DOM ezt az oldalt a következő fastruktúraként kezeli:



A `document` objektum a teljes HTML-dokumentumot képviseli, és az egyik legfontosabb kiindulópont a DOM manipuláció során. Segítségével elérhetjük az oldal elemeit, új elemeket hozhatunk létre, módosíthatjuk a tartalmat, és kezelhetjük az eseményeket. A DOM többféle csomópontot tartalmaz, de a legfontosabbak a következők:

- **Node** (csomópont): Minden, ami a DOM fában található, egy csomópont (node). A csomópontok különböző típusúak lehetnek, például elemek, attribútumok vagy szövegek.
- **Element** (elem): Az element típusú csomópontok a tényleges HTML-elemeket képviselik (pl. `div`, `p`, `h1` stb.).
    - Példa: ** **Ez egy bekezdés.** **
- **Attribute** (attribútum): Az attribute típusú csomópontok egy HTML-elem tulajdonságait tárolják (pl. `id`, `class`, `src`, `href` stb.).
    - Példa: <p **id=“main-text”**>Ez egy fontos bekezdés.
- **Text** (szöveg): A text típusú csomópontok a HTML-elemek belső szöveges tartalmát tárolják.
    - Példa: **Ez egy bekezdés.**

A DOM lehetővé teszi, hogy JavaScript segítségével:

- Elemeket válasszunk ki és módosítsunk (pl. `document.getElementById()`, `document.querySelector()`)
- Új elemeket hozzunk létre és illesszünk be (pl. `createElement()`, `appendChild()`)
- Eseménykezelőket hozzunk létre (pl. `addEventListener()`)
- Stílusokat és attribútumokat módosítsunk (pl. `element.style`, `setAttribute()`)

2. Alapvető DOM manipulációs technikák bemutatása (25 perc) Elemek kiválasztása

getElementById getElementsByClassName getElementsByTagName querySelector querySelectorAll Tartalom módosítása

innerText, textContent innerHTML Attribútumok módosítása

setAttribute, getAttribute classList.add, classList.remove, classList.toggle Új elemek létrehozása és hozzáadása

createElement appendChild, prepend insertBefore Eseménykezelés

addEventListener Példa: gombra kattintáskor szöveg megváltoztatása Stílus módosítása JavaScriptből

element.style.property Dinamikus osztályváltás classList-tel

3. Példa (10 perc) Én mutatok egy példát, amely jól illusztrálja az eddig tanultakat.

Bemutatott feladat: Egyszerű interaktív lista Feladat:

Van egy <ul> lista pár kezdeti elemmel. Egy megnyomására egy új elem hozzáadódik a listához. Ha egy listaelemre kattintanak, az pirosra változik.

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Manipuláció</title>
    <style>
        .highlight { color: red; }
    </style>
</head>
<body>
    <ul id="myList">
        <li>Első elem</li>
        <li>Második elem</li>
    </ul>
    <button id="addItem">Új elem hozzáadása</button>

    <script>
        document.getElementById('addItem').addEventListener('click',
function() {
            let newItem = document.createElement('li');
            newItem.textContent = "Új elem";
            document.getElementById('myList').appendChild(newItem);
        });

        document.getElementById('myList').addEventListener('click',
function(event) {
            if (event.target.tagName === 'LI') {
                event.target.classList.toggle('highlight');
            }
        });
    </script>
</body>
</html>
```

Feladat

A feladat egy egyszerű, dinamikus teendő lista készítése.

- Legyen egy `<input>` szöveges mező, ahova a felhasználó beírhat egy új teendőt.
- Legyen egy „Hozzáadás” gomb, melynek lenyomásakor a beírt szöveg kerüljön fel egy `<ul>` listára.
- A listán szereplő elemekre kattintva azok legyenek áthúzva (CSS: `text-decoration: line-through`).
- Legyen egy „Törlés” gomb minden elem mellett, amely törli az adott teendőt listából.
- Az oldal elemeinek formázásához használj CSS stílusokat!

Az elkészült megoldást töltsd fel a GitHub repository-dba!

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:web_technologia_alapjai:js_dom?rev=1741611855

Last update: **2025/03/10 13:04**