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.

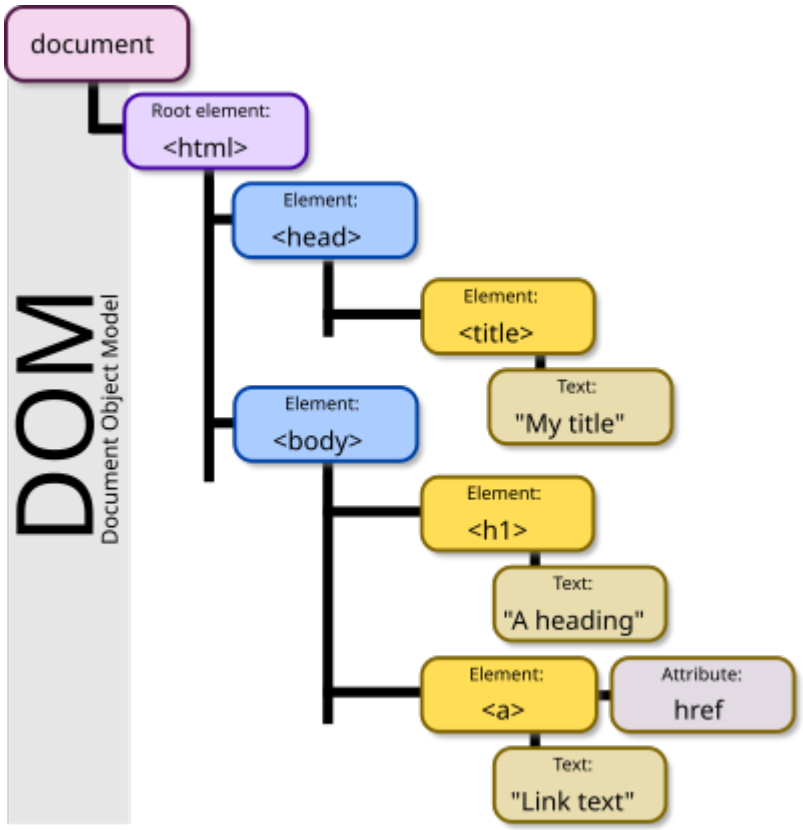
Document Object Model

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()`)

DOM manipulációs technikák

A DOM elérésére a globális `document` objektumon keresztül van lehetőség, melynek dokumentációja

itt található meg: https://developer.mozilla.org/en-US/docs/Web/API/Document

A következő bekezdések a teljesség igénye nélkül mutatnak be néhány gyakran használt metódust.

Elemek kiválasztása

A DOM-ban található csomópontok elérését a következő függvények biztosítják:

- `getElementById(id)` – Visszaadja az első olyan elemet, amelynek `id` attribútuma megegyezik a megadott értékkel. Ha nincs ilyen elem, null értéket ad vissza.
- `getElementsByClassName(className)` – Egy `HTMLCollection`-t ad vissza, amely tartalmazza az összes olyan elemet, amely rendelkezik a megadott osztálynévvel.
- `getElementsByTagName(tagName)` – Egy `HTMLCollection`-t ad vissza, amely tartalmazza az összes megadott típusú HTML elemet.
- `querySelector(selector)` – Az első olyan elemet adja vissza, amely illeszkedik a megadott CSS-szelektorhoz. Ha nincs találat, null-t ad vissza.
- `querySelectorAll(selector)` – Egy `NodeList`-et ad vissza, amely tartalmazza az összes olyan elemet, amely megfelel a megadott CSS-szelektornak.

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM - Elem kiválasztás</title>
</head>
<body>
    <h1 id="title">Főcím</h1>
    <p class="text">Ez egy bekezdés.</p>
    <p class="text">Ez egy másik bekezdés.</p>
    <div>
        <span>Span 1</span>
        <span>Span 2</span>
    </div>

    <script>
        // Egy adott ID-val rendelkező elem kiválasztása
        let titleElement = document.getElementById("title");
        console.log(titleElement.textContent); // Főcím

        // Osztály alapján több elem kiválasztása
        let textElements = document.getElementsByClassName("text");
        console.log(textElements[0].textContent); // Ez egy bekezdés.

        // Összes adott típusú elem kiválasztása
        let spans = document.getElementsByTagName("span");
        console.log(spans.length); // 2
```

```
        // Az első találat kiválasztása CSS-szelektorral
        let firstParagraph = document.querySelector(".text");
        console.log(firstParagraph.textContent); // Ez egy bekezdés.

        // Az összes találat kiválasztása CSS-szelektorral
        let allParagraphs = document.querySelectorAll(".text");
        allParagraphs.forEach(p => console.log(p.textContent));
        // Ez egy bekezdés.
        // Ez egy másik bekezdés.
    </script>
</body>
</html>
```

Tartalom módosítása

A DOM-ban található csomópontok tartalmát a következő tulajdonságok segítségével lehet elérni, illetve módosítani:

- `innerText` és `textContent` – Mindkettő a szöveges tartalom lekérésére vagy módosítására szolgál, de különbség köztük, hogy az `innerText` figyelembe veszi a CSS láthatósági beállításait, míg a `textContent` minden szöveget visszaad, függetlenül a stílusoktól.
- `innerHTML` – Az elem HTML tartalmának beállítására vagy lekérésére szolgál. Vigyázni kell vele, mert ha külső bemenetet tartalmaz, az alkalmazás sérülékennyé válhat az XSS támadásokkal szemben.

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM tartalom módosítása</title>
    <style>
        .hidden {
            display: none;
        }
    </style>
</head>
<body>
    <h1 id="title">Eredeti <span class="hidden">cím</span></h1>
    <p id="paragraph">Ez egy <strong>bekezdés</strong> HTML formázással.</p>
    <p id="hiddenText" class="hidden">Ez egy rejtett szöveg.</p>

    <script>
        // innerText: figyelembe veszi a szöveg egyes részeinek láthatóságát
        let title = document.getElementById("title");
        console.log(title.innerText); // Eredeti
        title.innerText = "Új cím";
```

```
        // textContent: nem veszi figyelembe a CSS láthatóságot
        let hiddenText = document.getElementById("hiddenText");
        console.log(hiddenText.textContent); // Ez egy rejtett szöveg

        // innerHTML: a HTML tartalmat is kezeli
        let paragraph = document.getElementById("paragraph");
        console.log(paragraph.innerHTML); // Ez egy
<strong>bekezdés</strong> HTML formázással.
        paragraph.innerHTML = "Ez egy <strong>új bekezdés</strong>
módosított HTML tartalommal.";
    </script>
</body>
</html>
```

Attribútumok és stílus módosítása

A DOM csomópontjainak attribútumait és stílusát a következő függvényekkel lehet elérni és módosítani:

- `setAttribute(name, value)`, `getAttribute(name)` – Az elem attribútumainak beállítására, valamint lekérdezésére szolgálnak.
- `classList.add(className)`, `classList.remove(className)`, `classList.toggle(className)` – Osztályok hozzáadására, eltávolítására és állapotváltására használható.
- `element.style` – Az elem CSS-stílusának közvetlen módosítására használható.

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Attribútumok és stílus módosítása</title>
    <style>
        .highlight {
            background-color: yellow;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <button id="my-button">Szövegkiemelés</button>
    <p id="my-paragraph" class="text">Ez egy bekezdés.</p>

    <script>
        let button = document.getElementById("my-button");
        let paragraph = document.getElementById("my-paragraph");

        // setAttribute és getAttribute használata
```

```
        button.setAttribute("title", "Ez a gomb kiemeli a bekezdést vagy
eltávolítja a kiemelést."); // Tooltip szöveg beállítása
        console.log(button.getAttribute("title")); // Ez a gomb kiemeli a
bekezdést vagy eltávolítja a kiemelést.

        // classList használata
        paragraph.classList.add("highlight"); // Kiemeli a bekezdést a
highlight osztály hozzáadásával
        setTimeout(() => {
            paragraph.classList.remove("highlight"); // 2 másodperc
elteltével eltávolítja a kiemelést
        }, 2000);

        button.addEventListener("click", () => {
            paragraph.classList.toggle("highlight"); // Ha rajta van a
highlight osztály, eltávolítja; ha nincs, hozzáadja
        });

        // style attribútum használata
        paragraph.style.color = "blue"; // Szöveg színének megváltoztatása
        paragraph.style.fontSize = "20px"; // Betűméret növelése
    </script>
</body>
</html>
```

Új elemek létrehozása és hozzáadása

A következő metódusokkal új csomópontokat hozhatunk létre, és adhatunk hozzá a DOM-hoz:

- `createElement(tagName)` – Egy új HTML elemet hoz létre a megadott tag-névvel.
- `appendChild(node)`, `prepend(node)` – Az `appendChild` az elem végéhez, a `prepend` az elejéhez ad hozzá egy új gyerekelemet.
- `insertBefore(newNode, referenceNode)` – Egy új elemet szúr be egy másik adott elem elé.

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Új elemek létrehozása és hozzáadása</title>
</head>
<body>
    <div id="container">
        <p>Ez az eredeti bekezdés.</p>
    </div>
```

```
    <button id="add-btn">Új elem hozzáadása</button>

    <script>
        let container = document.getElementById("container");
        let button = document.getElementById("add-btn");

        button.addEventListener("click", () => {
            // createElement használata
            let newParagraph = document.createElement("p");
            newParagraph.textContent = "Ez egy új bekezdés.";

            // appendChild használata (hozzáadja a végére)
            container.appendChild(newParagraph);

            // prepend használata (hozzáadja az elejére)
            let firstParagraph = document.createElement("p");
            firstParagraph.textContent = "Ez egy bekezdés az elején.";
            container.prepend(firstParagraph);

            // insertBefore használata (új elem beszúrása egy másik elé)
            let middleParagraph = document.createElement("p");
            middleParagraph.textContent = "Ez egy köztes bekezdés.";

            let referenceNode = container.children[1]; // Az első bekezdés
után helyezzük el
            container.insertBefore(middleParagraph, referenceNode);
        });
    </script>

</body>
</html>
```

Eseménykezelés

A DOM-ban található csomópontok eseményeinek (pl. kattintás, dupla kattintás, szövegmező fókuszba kerülése, drag&drop műveletek stb.) kezeléséhez a következő függvényt használhatjuk:

- `addEventListener(event, callback)` – Eseményfigyelőt (callback függvényt) ad az elemhez, amely a megadott esemény (event) bekövetkeztekor lefut.

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eseménykezelés példa</title>
</head>
<body>
```

```
    <input type="text" id="name-input" placeholder="Írd be a neved ...">
    <button id="submit-btn">Küldés</button>

    <script>
        // Szövegmező fókusz eseménykezelője
        document.getElementById("name-input").addEventListener("focus",
function() {
            this.style.borderColor = "blue";
        });

        // Gombra kattintás eseménykezelője
        document.getElementById("submit-btn").addEventListener("click",
function() {
            let inputValue = document.getElementById("name-input").value;
            alert("Hello, " + inputValue);
        });
    </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`).
- Az oldal elemeinek formázásához használj CSS stílusokat!

Az elkészült megoldást töltsd fel a GitHub repository-dba!