1. HTML DOM

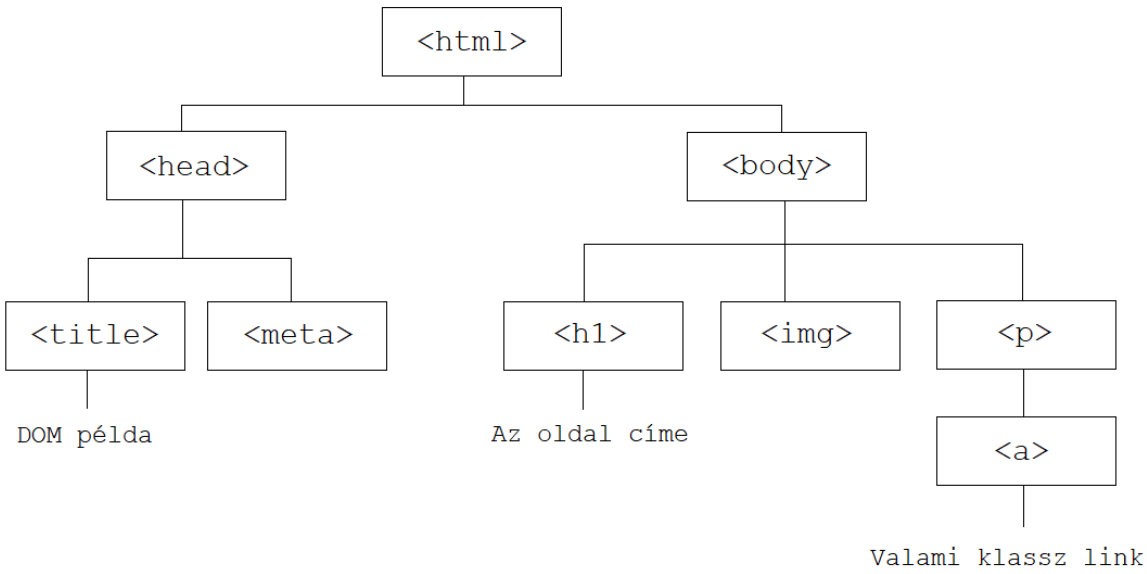
A webfejlesztésben a **HTML** nyelvet használjuk weboldalak létrehozására. Ennek a nyelvnek a segítségével mondhatjuk meg, hogy mi az, amit egy weboldalon látni szeretnénk (pl. szövegek, képek, táblázatok, űrlapok, multimédia stb.), illetve lehetőségünk van a weboldalon megjelenő tartalom strukturálására is különféle szakaszok, tartalmi egységek kialakításával.

A HTML dokumentumok úgy épülnek fel, hogy HTML objektumokat (úgynevezett **tageket**) ágyazunk egymásba. Ezek az objektumok egy hierarchikus fastruktúrát alkotnak a dokumentumban.

Amikor egy weboldal betöltődik, akkor a weboldalon található HTML objektumokból elkészíti az úgynevezett **dokumentum-objektum modell**t, avagy röviden a **DOM**-ot. A **DOM fa** (DOM tree) segítségével könnyen szemléltethetjük a weboldalon található HTML elemek hierarchikus viszonyait. Nézzünk példát egy DOM fára!

Példa: Egy egyszerű HTML kód és az ahhoz tartozó DOM fa

```html
<!DOCTYPE html>
<html lang="hu">
   <head>
      <title>DOM példa</title>
      <meta charset="UTF-8"/>
   </head>
   <body>
      <h1>Az oldal címe</h1>
      <img src="my-image.png" alt="Valami klassz kép"/>
      <p>
         <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ">Valami klassz link</a>
      </p>
   </body>
</html>
```



Megjegyzés: A `DOCTYPE` nem egy HTML tag, ezért a DOM fában sem szerepel.

1.1. HTML elemek DOM-beli viszonyai

Ha az A objektum (nem feltétlen közvetlenül) tartalmazza a B objektumot, akkor azt mondjuk, hogy az A objektum a B objektum **ős**e, a B objektum pedig A-nak **leszármazott**ja. Amennyiben ez a tartalmazás **közvetlen**, akkor A-t a B **szülő**jének, B-t pedig az A **gyerek**ének nevezzük.

Néhány példa a fenti kódból és az ahhoz tartozó DOM fából:

- A `<body>` objektum leszármazottjai a `<h1>`, `<img>`, `<p>` és `<a>` objektumok, illetve a „Valami klassz link” és „Az oldal címe” szöveges csomópontok.
- A `<body>` objektum gyerekei a `<h1>`, `<img>` és `<p>` objektumok. Másképp mondva: a `<h1>`, `<img>` és `<p>` objektumok szülője a `<body>`.
- A `<body>` objektumnak az `<a>` objektum **nem** gyereke, csak leszármazottja, hiszen itt a tartalmazás nem közvetlen (van még a fában egy `<p>` elem is közöttük).

Ha az A és B objektumok szülője megegyezik, akkor A és B egymás **testvér**ei. Például a fenti kódban és az ahhoz tartozó DOM fában a `<h1>`, `<img>` és `<p>` objektumok egymás testvérei, hiszen mindhárom objektum szülője a `<body>`.

A DOM fa tetején lévő, szülővel nem rendelkező elemet **gyökérelem**nek nevezzük. A teljes HTML DOM-ban a gyökérelem mindig a `<html>` objektum lesz (ugyanis ebbe ágyazunk be minden további HTML elemet).

1.2. A DOM egyik gyakorlati jelentősége: DOM alapú CSS szelektorok

Amikor a weboldalunk tartalmát CSS-ben formázzuk, akkor használhatunk olyan szelektorokat (kijelölőket), amelyek a DOM-beli viszonyaik alapján jelölnek ki HTML objektumokat. Néhány példa DOM alapú CSS szelektorokra, a teljesség igénye nélkül:

- `div p`: kijelöl minden olyan `<p>`-t, amely egy `<div>` leszármazottja
- `div > p`: kijelöl minden olyan `<p>`-t, amely egy `<div>` gyereke
- `div ~ p`: kijelöl minden olyan `<p>`-t, amely egy `<div>` testvére, és ezen `<div>` után szerepel
- `p:first-child`: kijelöl minden olyan `<p>`-t, amely a szülőjének legelső gyereke
- `p:nth-child(n)`: kijelöl minden olyan `<p>`-t, amely a szülőjének `n`-edik gyereke
- `p:last-child`: kijelöl minden olyan `<p>`-t, amely a szülőjének utolsó gyereke
- `p:first-of-type`: kijelöl minden olyan `<p>`-t, amely a `<p>` típusú testvérei közül a legelső
- `p:nth-of-type(n)`: kijelöl minden olyan `<p>`-t, amely a `<p>` típusú testvérei közül az `n`-edik
- `p:last-of-type`: kijelöl minden olyan `<p>`-t, amely a `<p>` típusú testvérei közül az utolsó.

1.3. A DOM egy másik gyakorlati jelentősége: JavaScript DOM manipuláció

A webes világban gyakran előfordul, hogy a DOM fát manipulálni szeretnénk (pl. szeretnénk egy objektumot módosítani vagy törölni, esetleg egy új objektumot akarunk a fába beszúrni). Erre biztosítanak lehetőséget a JavaScript **DOM manipulációs műveletei**.

Néhány valós életbeli példa arra, amikor a DOM fát manipulálni szeretnénk:

- Adott egy HTML űrlap, amit a felhasználó kitölt. A szervernek való továbbítás előtt kliensoldali ellenőrzéseket szeretnénk végezni JavaScriptben, hogy megbizonyosodjunk arról, hogy a megadott űrlapadatok helyesek (pl. e-mail cím formátuma megfelelő, életkor nem negatív stb.). Ekkor JavaScriptben DOM műveletekkel lekérjük az űrlapadatokat és ellenőrizzük őket.
- Lekérdezünk adatokat egy szervertől, és meg szeretnénk azokat jeleníteni a weboldalunkon. Mivel a HTML oldal már rég betöltődött addigra, amire az adatok megjöttek, ezért azokat úgy tudjuk megjeleníteni a weboldalon, hogy utólag szúrogatjuk be őket a DOM fába.
- Szeretnék elérni, hogy egy gombra kattintva válthasson a felhasználó világos és sötét téma között. Ha a weboldal jelenleg világos témájú, akkor a gombra kattintva sötét stílust adunk az oldalnak. Ha a weboldal sötét témájú, akkor a gombra kattintva világos témájúvá tehetjük azt.

2. A JavaScript DOM manipulációs lehetőségei

A JavaScript DOM manipulációs műveletei tehát lehetőséget biztosítanak arra, hogy dinamikusan módosítsuk a HTML dokumentumaink szerkezetét, tartalmát és stílusát. Ebben a fejezetben megismerkedünk néhány fontosabb DOM manipulációs lehetőséggel.

A későbbiekben sokszor fogjuk használni a **`document`** objektumot. Ez lényegében a böngésző által megnyitott HTML dokumentumot reprezentálja, és hozzáférést biztosít a DOM fához.

2.1. HTML objektumok megkeresése a DOM fában

A DOM fában való kereséshez használhatjuk az alábbi metódusokat:

- `document.getElementById()`: visszaadja az adott `id` értékkel rendelkező elemet (ennek a használata a legegyszerűbb, hiszen ez mindig csak 1 elemet ad vissza, mert az `id` értéke egyedi)
- `document.getElementsByTagName()`: visszaadja az adott tagnévvel rendelkező elemeket (minden esetben egy tömböt ad vissza)
- `document.getElementsByClassName()`: visszaadja az adott `class` értékkel rendelkező elemeket (minden esetben egy tömböt ad vissza).

A következő oldalon nézünk egy-egy példát ezeknek a DOM metódusoknak a használatára.

Példa: Írassuk ki a konzolra alábbi HTML kódból i.) az `id="second"` attribútummal rendelkező bekezdés szövegét, ii.) az első bekezdés szövegét, iii.) az első `class="important"` attribútummal rendelkező bekezdés szövegét! A szöveges tartalom lekéréséhez használjuk az `innerText` property-t!

```
<!DOCTYPE html>
<html lang="hu">
   <head>
      <title>DOM példa</title>
      <meta charset="UTF-8"/>
   </head>
   <body>
      <p id="first">Ez itt az első bekezdés.</p>
      <p id="second" class="bold">Ez itt a második bekezdés.</p>
      <p id="third" class="important">Ez itt a harmadik bekezdés.</p>
      <p id="fourth" class="important">Ez itt a negyedik bekezdés.</p>

      <!-- Egy lehetséges megoldás -->

      <script>
        console.log(document.getElementById("second").innerText);                // i.)
        console.log(document.getElementsByTagName("p")[0].innerText);            // ii.)
        console.log(document.getElementsByClassName("important")[0].innerText); // iii.)
      </script>
   </body>
</html>
```