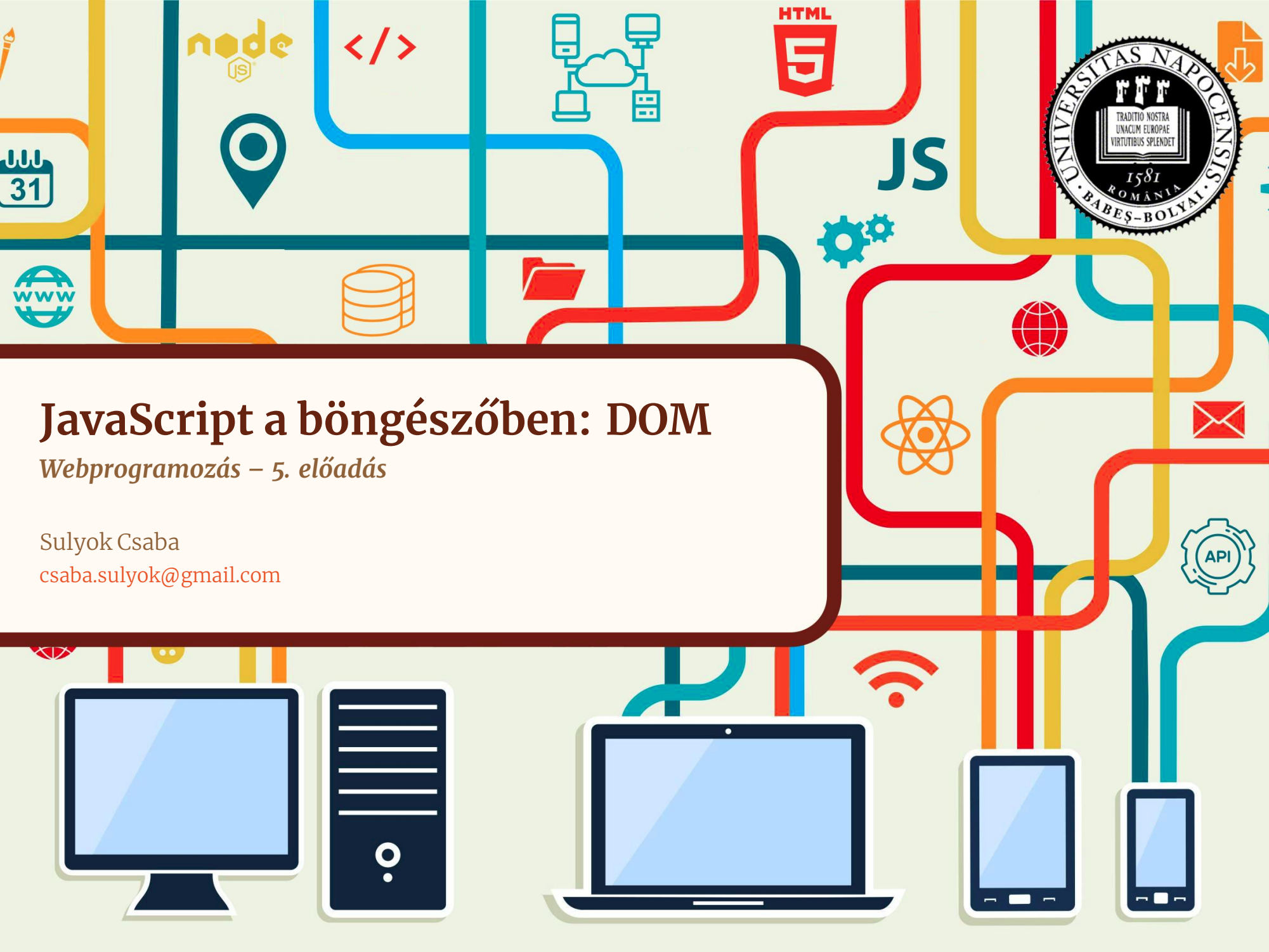



# JavaScript a böngészőben: DOM

*Webprogramozás – 5. előadás*

Sulyok Csaba
csaba.sulyok@gmail.com

# JavaScript beszúrása HTML-be

- ahogyan `style`-lal szúrtuk be a CSS kódot, JavaScriptet a `script` elemmel szúrhatunk be:

```
<script>

</script>
```

- régebbi példákban:

```
<script type="text/javascript">
...
</script>
```

- külső script-állomány beszúrása:

```
<!-- relative path -->
<script src="other.js"></script>
<!-- absolute path -->
<script src="https://externalhost/library.js"></script>
```

- ezutóbbit több HTML oldal is használhatja
- átláthatóbb kód: HTML és JavaScript szétválasztása
- cache-elhető
- **nem használható önbezáró tag, szükséges a záró `</script>`**

# JavaScript beszúrása HTML-be

- a `script` elem HTML dokumentumon belül elvileg bárhova elhelyezhető
- a kiértékelése az *oldal betöltése közben* történik (**vigyázat:** nem biztos hogy az ablak minden része be van töltve, amikor lefut).
- leghasznosabb helyek:
  - `head` elemben
  - közvetlenül a `body` befejező tagje elé

```
<!DOCTYPE html>
<html>
    <head>
        <!-- ... -->
    </head>
    <body>
        <!-- ... -->

        <script>
            console.log("Hello, World!");
        </script>
    </body>
</html>
```

# ES modulok HTML-ben

- ECMAScript modulok a `type="module"` attribútum megadásával jelezhetőek. Lehetnek mind külsők, mind belsők.
- Eredetileg a modulként működő JS állományoknak az `mjs` kiterjesztést tartották fenn, de kevés eszköz adoptálta a használatát. Ha ezt használjuk, nem szükséges kiterjesztést megadni az URL-eknek.

```
<script type="module" src="main.js"></script>
<script type="module">
    import stuff from './somewhere';
    // ...
</script>
```

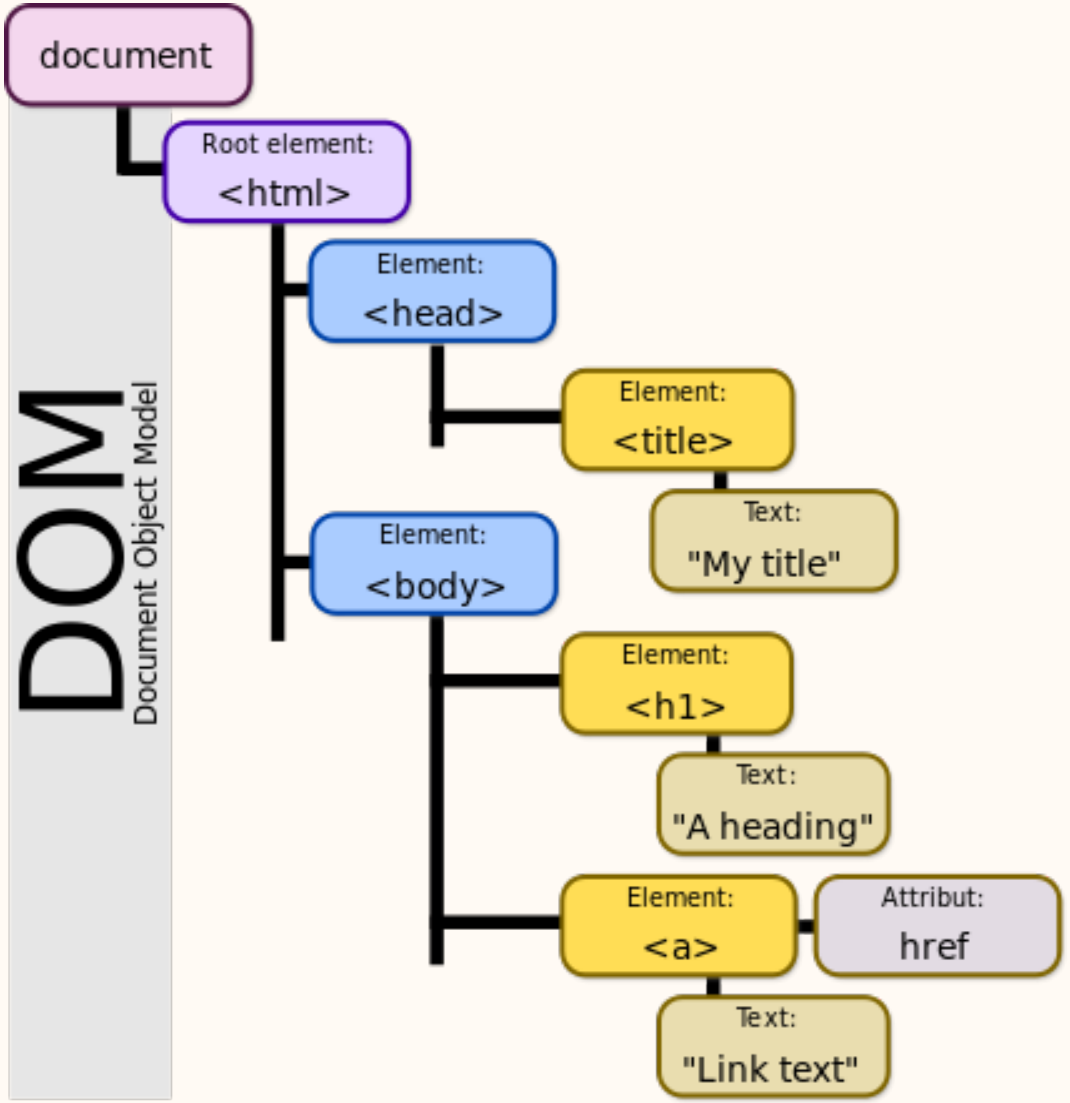
- Az egyszerű (nem modul) JS szkriptek által deklarált változók elérhetőek a böngésző JS konzoljában, míg a modulokban használtak nem.

# HTML Document Object Model

- A **HTML Dokumentum Objektum Modell** (HTML DOM) lehetőséget teremt a dokumentum elemeihez való hozzáférésre és ezek módosítására JavaScript segítségével
- A DOM platform- és nyelvfüggetlen interfész
- a DOM a HTML dokumentumnak egy fa-szerkezetet feleltet meg. Legfontosabb csomópont típusok:
  - dokumentum csomópont - a teljes dokumentum
  - elem csomópont - minden HTML elem (tag)
  - szöveg csomópont - az egyes HTML elemek törzse
  - attribútum csomópont - a HTML attribútumok
  - megjegyzés csomópontok - HTML kommentek

# HTML Document Object Model

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



# DOM

- DOM objektumok közti kapcsolat:
  - szülő (parent)
  - gyerek (child)
  - testvér (sibling)
  - előd (ancestor)
  - utód (descendant)
- központi dokumentum elem: `document`
- csomópontokhoz való hozzáférés:
  - `document.` ...: `getElementById(id)`, `getElementsByTagName(name)`, `getElementsByClassName(class)` metódusok segítségével
  - egy elem csomópont `parentNode`, `childNodes`, `firstChild`, `lastChild`, `previousSibling`, `nextSibling`, `attributes`, ... mezőit használva
  - dinamikusan értelmezett HTML kód adható meg egy csomópont `innerHTML` mezőjének beállításával
  - sajátos csomópontok: `document.body`, `document.documentElement` (a HTML elemnek felel meg)

# DOM

- egy csomópont standard mezői információt nyújtanak az illető csomópontról:
  - `nodeName` (read only) - csomópont neve (tag-név nagybetűkkel, attribútumnév, `#text`, `#document`, `#comment`)
  - `nodeType` (read only) - csomópont típusa, numerikus érték, pár példa:
    - 1 ⇒ `Node.ELEMENT_NODE`
    - 2 ⇒ `Node.ATTRIBUTE_NODE`
    - 3 ⇒ `Node.TEXT_NODE`
    - 8 ⇒ `Node.COMMENT_NODE`
  - `nodeValue` - szöveg- illetve attribútum csomópontok esetén a szöveget illetve attribútum értékét tartalmazza
- *megj.*: egy-egy csomópont számos más mezővel is rendelkezik. Csomópont típusok listája
- Dokumentációk:
  - W3C DOM4
  - WHATWG Living Standard

# DOM: egy *x* csomópont metódusai

- `x.getElementsByTagName(name)` – megadott elemnévvel rendelkező elemek listája
- `x.appendChild(node)` – csomópont hozzáadás (ld. `beszuras.html`)
- `x.removeChild(node)` – csomópont törlés
- `x.replaceChild(new_node, old_node)` – csomópont lecserélése
- `x.insertBefore(new_node, existing_node)` – beszúr egy új csomópontot egy már létező elé
- `x.createAttribute(attrname)` – létrehoz egy attrib. csomópontot
- `x.createElement(nodename)` – létrehoz egy elemcsomópontot
- `x.createTextNode(text)` – létrehoz egy szöveg-csomópontot
- `x.getAttribute(attrname)` – visszatéríti a keresett attribútum értéket
- `x.setAttribute(name,value)` – beállítja/átállítja a megadott attribútum értékét

# DOM példa

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML example</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <p id="hello">Hello, World!</p>

        <script>
            const node = document.getElementById("hello");
            // check node type
            if (node.nodeType === Node.ELEMENT_NODE) {
                console.log("Node is element");
            }
            // check child node type
            const textNode = node.childNodes[0];
            if (textNode.nodeType === Node.TEXT_NODE) {
                console.log("Child is text node");
            }
            // change content
            textNode.nodeValue = "Dynamically changed value";
        </script>
    </body>
</html>
```

# Browser Object Model

- `window` – a JavaScript hierarchiában legfelső szinten levő objektum. A böngészőablaknak felel meg.
  - kollekció: `frames[]`
  - mezők: `document`, `history`, `location`, `screen`, `innerHeight`, `innerWidth`, `self`, `top`
  - metódusok: `blur()`, `focus()`, `alert()`, `confirm()`, `prompt()`, `open()`, `close()`, `setTimeout()`, `clearTimeout()`, `setInterval()`, `clearInterval()`, `scrollBy()`, `scrollTo()`, `moveBy()`, `moveTo()`, `resizeTo()`, `resizeBy()`
- előugró ablakok (a `window` objektum metódusai)
  - figyelmeztető ablak (alert box): – `alert("szoveg")` – inkább hibakeresés (debug) céljából használják
  - jóváhagyó ablak (confirm box): – `confirm("szoveg")` – jóváhagyás kérése "fontosabb" művelet végrehajtása előtt
  - adatbekérő ablak (prompt box): – `prompt("szoveg", "alapertelmezett")` – pl. adatbekérés az oldal betöltése előtt (ritkán használt)

# Browser Object Model

- a `window` objektum alobjektumai:
  - `history` – az illető böngészőablakból meglátogatott URL-ket tartalmazza
    - mezők, metódusok: `length`, `back()`, `forward()`, `go()`
  - `location` – információ az aktuális URL-ről
    - mezők: `href` (teljes URL), `protocol`, `hostname`, `port`, `pathname`, `hash` (`#` utáni rész), `search` (`?` utáni rész)
    - metódusok: `assign()`, `reload()`, `replace()`
  - `navigator` – információ a kliens böngészőjéről
    - mezők: `appName` (böngésző típusa), `appVersion` (böngésző verziószáma)
  - `screen` – információ a kliens képernyőjéről
    - mezők: `width` (képernyő szélessége), `height` (magassága)
  - `document` $\longrightarrow$

# Browser Object Model

- `document` objektum
  - a teljes HTML dokumentumhoz való hozzáférést teszi lehetővé
  - a window JavaScript-objektum része (`window.document`)
  - kollekciók: `anchors[]`, `forms[]`, `images[]`, `links[]`
  - mezők: `body`, `cookie`, `domain`, `referrer`, `title`, `URL`, `lastModified`
  - metódusok: `getElementById()`, `getElementsByTagName`, `getElementsByName()`, `open()`, `close()`, `write()`, `writeln()`
- HTML elemeknek megfelelő objektumok esetén:
  - beállítható, átállítható bármely attribútum értéke
  - az elem törzse az `innerHTML` mezőn keresztül érhető el; lásd: `csere.html`

# JavaScript események

- a JavaScript által felismert történések
- dinamikus (időben változó tartalmú) HTML oldalak létrehozását teszik lehetővé
- minden egyes HTML elemhez vannak hozzárendelt események, melyek hatására JavaScript függvényt futtathatunk le
- HTML elemek attributumaiként adjuk meg, pl.

```
<input type="button" value="Nyomj meg!" onclick="gombNyomas()" />
```

- az attributum értéke JavaScript kódként van kezelve s futtatva

# JavaScript események

- néhány esemény (*XHTML esetén csupa kisbetűvel*):
  - `onLoad`, `onUnload` - oldal vagy kép betöltése, oldal elhagyása (pl. böngésző típusának ellenőrzése... $\longrightarrow$ elavult szokás, sütik tárolása, példák: `informaciok.html`, `cookie.html`
  - `onClick` - tipikusan gombnyomás vagy más HTML elemre - pl. kép - való kattintásra meghívunk egy JavaScript függvényt
  - `onFocus`, `onBlur`, `onChange` - pl. form elem értékének ellenőrzése, ha változott a tartalma
  - `onSubmit` - form leadásakor váltódik ki, pl. hasznos az összes adatainak ellenőrzése a szerverre való küldés előtt
  - `onMouseOver`, `onMouseOut` - pl. animált gombok

# JavaScript események példa

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML example</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <a href="#" id="hello" onclick="sayHello()">Click me!</a>

        <script>
            const sayHello = () => { console.log("Hello"); }
        </script>
    </body>
</html>
```

# onload

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML example</title>
        <meta charset="UTF-8">
    </head>
    <body onload="loadFinished()">
        <p>Not quite loaded yet...</p>

        <script>
            const loadFinished = () => {
                document.getElementsByTagName("p")[0].innerHTML = "<b>Loading done!</b>";
            };
        </script>
    </body>
</html>
```