

WEBPROGRAMOZÁS

FELÜLETI ELEMÉK PROGRAMOZÁSA: A DOM

Horváth Győző

egyetemi docens

horvath.gyozo@inf.elte.hu

1117 Budapest, Pázmány Péter sétány 1/c., 2.408

Tel: (1) 372-2500/8469

ISMÉTLÉS

JAVASCRIPT NYELVI ELEMEK

- Alapok
 - C-alapú szintaxis
 - vezérlési szerkezetek
 - elemi típusok (szöveg, szám, logikai)
- Függvények
 - funkcionális aspektus
 - függvénydeklaráció
 - függvényliterál
 - függvény mint paraméter
- Tömbök
 - tömbfüggvények
- Objektumok
 - OOP aspektus

PÉLDA

```
// Adatszerkezetek
const books = [
  { title: 'Anna Karenina', year: 1877 },
  { title: 'Harry Potter és a bölcsek köve', year: 1997 },
];

// Függvények
function getBooksOf1800(books) {
  const result = [];
  for (const book of books) {
    if (book.year >= 1800 && book.year < 1900) {
      booksOf1800.push(book);
    }
  }
  return result;
}
const booksOf1800 = getBooksOf1800(books)

// Tömbfüggvények
const booksOf1900 = books.filter(book => book.year >= 1900 && book.year < 2000)
```

PÉLDA

```
class Library {
  books = [
    {title: 'A case for Christ', author: 'Lee Strobel'},
    // ...
  ]
  booksByAuthor(author) {
    return this.books.filter(book => book.author === author)
  }
}

const lib = new Library()
const strobels = lib.booksByAuthor('Lee Strobel')
for (const b of strobels) {
  console.log(b.title);
}
```

FELÜLETI ELEMÉK PROGRAMOZÁSA

PÉLDA

- Feladat
 - Bekérni a nevet, pl. “Senki bácsi”
 - Kiírni: “Hello Senki bácsi!”
- Úrlap
 - Gomb lenyomása
 - Érték kiolvasása
 - Eredmény megjelenítése

```
<form>
  Name: <input type="text" id="name">
  <input type="button" value="Say hello!" id="hello">
  <span id="output"></span>
</form>
```

Name:

Say hello!

```
<form>
  Name: <input type="text" id="name">
  <input type="button" value="Say hello!" id="hello">
  <span id="output"></span>
</form>
```

MEGOLDÁS LÉPÉSEI

- Reagálni a gomb lenyomására.
- Kiolvasni a szöveges beviteli mező értékét (beolvasás).
- Előállítani a bemenet alapján a kimenetet, azaz az üdvözlő szöveget (feldolgozás).
- Megjeleníteni az üdvözlő szöveget (kiírás).

MEGOLDÁS LÉPÉSEI (1)

- Reagálni a gomb lenyomására.
- Kiolvasni a szöveges beviteli mező értékét (beolvasás).
- Előállítani a bemenet alapján a kimenetet, azaz az üdvözlő szöveget (feldolgozás).
- Megjeleníteni az üdvözlő szöveget (kiírás).

```
function greet(name) {  
    return `Hello ${name}!`;  
}
```

FELHASZNÁLÓI FELÜLET

- Interaktivitás
 - adatok megadása (beolvasás)
 - információ megjelenítése (kiírás)
- Elemei
 - **HTML**: oldal szerkezetének leírása
 - **CSS**: megjelenés, stílus
 - **JavaScript**: program a felület működtetésére
- → **PROGRAMOZÁSI INTERFÉSZ A HTML ELEMEKHEZ**
- *Platform-specifikum*

DOKUMENTUM OBJEKTUM MODELL (DOM)

- HTML elemek belső ábrázolása
- HTML elemeknek megfelelő JS objektumok hierarchiája

DOKUMENTUM OBJEKTUM MODELL (DOM)

- A forrás és a DOM eltérhet egymástól.

DOKUMENTUM OBJEKTUM MODELL (DOM)

- Programozási interfész (a felület felé)
- Bemeneti-kimeneti interfész

DOM CORE

- csomópontok fája
- ábrázolás és műveletek

HTML DOM

- HTML elemek fája

DOKUMENTUM OBJEKTUM MODELL

- HTML és XML dokumentumok programozási felülete
- Szabványos interfész fa struktúra alapú hierarchiához
- DOM gyökere: `document`
- DOM csomópontok:
 - dokumentum
 - elem
 - attribútum
 - szöveges csomópont

SZÖVEGES CSOMÓPONTOK

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <p>Lorem ipsum.</p>
    <p class="active">
      Dolor sit amet.
    </p>
  </body>
</html>
```

```
- DOCTYPE: html
- HTML
- #text:
- META charset="utf-8"
- #text:
- TITLE
- #text:
- #text:
- BODY
- #text:
- P
- #text: Lorem ipsum.
- P class="active"
- #text: Dolor sit amet.
#text:
```

MŰVELETCSOPORTOK

- Elemek kiválasztása
- Szerkezet bejárása
- Szerkezet módosítása
 - attribútumok
 - új elem/attribútum létrehozása
 - módosítás
 - törlés

ELEM(EK) KIVÁLASZTÁSA

ELEM(EK) KIVÁLASZTÁSA

CSS szelektorral:

Egy elem: `document.querySelector(sel)`

Több elem: `document.querySelectorAll(sel)`

```
<form>
  Név:
  <input id="name" value="Luke">
  <button>Click me!</button>
</form>
<script>
  console.log( document.querySelector("#name") );
  console.log( document.querySelectorAll("form > *") );
</script>
```

CSS SZELEKTOROK

névvel

button

azonosítóval

#navbar

stílusosztályval

.important

attribútummal

[name=password]

univerzális

*

kombinálva

input.error[type=text]

közvetlen gyerek

form > input

leszármazott

#wrapper div

következő testvér

ul + p

utána jövő testvérek

ul ~ p

TÖBB ELEM KIVÁLASZTÁSA

```
<ul>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
</ul>
```

```
const listItems = document.querySelectorAll('ul > li')

// Array-like object (NodeList)
for (const li of listItems) { /*...*/ }      // OK
listItems.forEach(li => { /*...*/ })        // OK
listItems.map(li => { /*...*/ })           // Wrong: not an array

// Transforming to array
const listItemsArray = Array.from(listItems);
listItemsArray.map(li => { /*...*/ });       // OK
```

KIVÁLASZTÁS LESZÁRMAZOTTAK KÖZÜL

- elem.querySelector(sel)
- elem.querySelectorAll(sel)

```
<div class="first">
  <span>Apple</span>
</div>
<div class="second">
  <span>Pear</span>
</div>
```

```
const firstDiv = document.querySelector('div.first')
const spanInFirstDiv = firstDiv.querySelector('span')
```

PÉLDAFELADAT

```
<form>
  Name: <input type="text" id="name">
  <input type="button" value="Say hello!" id="hello">
  <span id="output"></span>
</form>
```

```
const nameInput = document.querySelector('#name')
const nameInput = document.querySelector('form input[type=text]')
```

DOM ELEM TULAJDONSÁGAI ÉS METÓDUSAI

DOM ELEM TULAJDONSÁGAI ÉS METÓDUSAI

- **Szabvány**
- **Dokumentáció**
- Analóg módszer (HTML, az esetek 80%-ban OK)
- Felfedező módszer (fejlesztői eszköztár, óvatosan)

ANALÓG MÓDSZER

- camelCase átírási mód
- Tulajdonságok

HTML attribútum DOM tulajdonság

type	type
value	value
readonly	readOnly
maxlength	maxLength

TOVÁBBI FONTOSABB TULAJDONSÁGOK

- `innerHTML`, `innerText`: az elem nyitó- és záróelem közötti HTML vagy szöveg

```
<p>This is a <span>text</span></p>
```

```
const p = document.querySelector('p');
// reading
p.innerHTML; // 'This is a <span>text</span>'
p.innerText; // 'This is a text'
// writing
p.innerHTML = 'This is a <strong>new</strong> text';
```

PÉLDAFELADAT

```
<form>
  Name: <input type="text" id="name">
  <input type="button" value="Say hello!" id="hello">
  <span id="output"></span>
</form>
```

```
// Reading
const input = document.querySelector('#name');
const name = input.value;
// Writing
const output = document.querySelector('#output');
output.innerHTML = greet(name);
```

MEGOLDÁS LÉPÉSEI (2)

- Reagálni a gomb lenyomására.
- ✓ Kiolvasni a szöveges beviteli mező értékét (beolvasás).
- ✓ Előállítani a bemenet alapján a kimenetet, azaz az üdvözlő szöveget (feldolgozás).
- ✓ Megjeleníteni az üdvözlő szöveget (kiírás).

```
// Reading
const input = document.querySelector('#name');
const name = input.value;
// Writing
const output = document.querySelector('#output');
output.innerHTML = greet(name);
```

ELEMI ESEMÉNYKEZELELÉS

ESEMÉNYKEZELÉS

1. Az eseményt kiváltó elem (pl. egy gomb)
2. Az esemény típusa (pl. kattintás)
3. Az eseményt kezelő függvény

```
elem.addEventListener('eventType', eventHandler)
```

```
const button = document.querySelector('button');

button.addEventListener('click', handleButtonClick);

function handleButtonClick() {
    console.log('clicked');
}
```

MEGOLDÁS LÉPÉSEI (3)

```
<form>
  Name: <input type="text" id="name"> <br>
  <input type="button" value="Say hello!" id="hello">
  <br>
  <span id="output"></span>
</form>
```

```
function greet(name) {
  return `Hello ${name}!`;
}
function handleHelloClick() {
  const name = input.value;
  const greeting = greet(name);
  output.innerHTML = greeting;
}

const input = document.querySelector('#name');
const output = document.querySelector('#output');
const hello = document.querySelector('#hello');

hello.addEventListener('click', handleHelloClick);
```

MEGOLDÁS LÉPÉSEI (3)

-  Reagálni a gomb lenyomására.
-  Kiolvasni a szöveges beviteli mező értékét (beolvasás).
-  Előállítani a bemenet alapján a kimenetet, azaz az üdvözlő szöveget (feldolgozás).
-  Megjeleníteni az üdvözlő szöveget (kiírás).

A DOM MINT I/O INTERFÉSZ

A DOM MINT I/O

- A felhasználó a felhasználói felületen keresztül lép kapcsolatba a programmal
- A programbeli elérés a DOM-on keresztül lehetséges
- A program számára a DOM az I/O

```
// Typical JavaScript program:  
  
// Reading (from DOM)  
// Processing (independent from I/O)  
// Writing (to DOM)
```

A DOM MINT I/O

- Beolvasás: DOM objektum tulajdonságának lekérdezése
- Kiírás: DOM objektum tulajdonságának módosítása

```
<!-- Reading -->
<input type="checkbox" id="accept" checked>
<script>
  const elfogad = document.querySelector("#accept").checked;
</script>

<!-- Writing -->
<img src="" id="image">
<script>
  const url = "http://images.io/example.png";
  const image = document.querySelector("#image");
  image.src = url;
</script>
```

PÉLDA DOM I/O

```
<input type="radio" name="gender" value="male" checked> Male
<input type="radio" name="gender" value="female"> Female
Maiden name: <input id="maidenName">
<script>
  // Reading
  const femaleRadio =
    document.querySelector("[name=gender][value=female]");
  const isFemale = femaleRadio.checked;
  // Writing
  document.querySelector("#maidenName").hidden = !isFemale;
</script>
```

ÚJ ELEMEK LÉTREHOZÁSA

- A kiírás egy speciális formája
- A HTML kód szövegesen
- `innerHTML` tulajdonság írása
- Akár egyszerre több elemet is

```
<div id="output"></div>
<script>
  const greeting = "<h1>Hello <em>World</em></h1>";
  const output = document.querySelector("#output");
  output.innerHTML = greeting;
</script>
```

HTML GENERÁLÁS

```
// Rövid statikus szöveg megadása
const html11 = `<h1>Hello there!</h1>`;

// Többsoros statikus szöveg megadása
const html12 = `
  <div>
    <p>No, <strong>I</strong> am your father!</p>
  </div>
`;

// Változók behelyettesítése
const callsign = 'Red 5';
const html13 = `${callsign}, standing by.`;

// Tömbök kiírása Leképezéssel
const callsigns = ["Red 10", "Red 7", "Red 3", "Red 6", "Red 9"];
const html14 = `
  <p>All wings, report in.</p>
  <ul>
    ${callsigns.map(callsign => `
      <li>${callsign}, standing by.</li>
    `).join("")}
  </ul>
`;
```

ÚJ ELEMEK LÉTREHOZÁSA

Programozottan:

- Létrehozás:
 - `document.createElement(elem)`
- Beszúrás:
 - `parent.appendChild(child)`: szülő gyerekeihez utolsóként hozzáadja az új elemet
 - `parent.insertBefore(newChild, refChild)`: referencia előtt beszúrja az új elemet

ÚJ ELEMEK BESZÚRÁSA

```
<body>
  <ul>
    <li>First</li>
    <li>Second</li>
    <li>Third</li>
  </ul>
</body>
```

```
const p = document.createElement('p');
document.body.appendChild(p);

const newLi = document.createElement('li');
const ul = document.querySelector('ul');
const refLi =
  ul.querySelector('li:nth-of-type(3)');
ul.insertBefore(newLi, refLi);
```

STÍLUSATTRIBÚTUMOK PROGRAMOZÁSA

STÍLUSATTRIBÚTUM, STÍLUSOSZTÁLY, STÍLUSLAP

ISMÉTLÉS

```
<div class="rodian bounty-hunter" style="bottom: 72in">  
    Greedo  
</div>
```

STÍLUSATTRIBÚTUM PROGRAMOZÁSA

`style` tulajdonság olvasása/írása

CSS stílustulajdonság

`left`

`background-color`

`border-bottom-width`

`border-top-left-radius`

style objektum tulajdonsága

`left`

`backgroundColor`

`borderBottomWidth`

`borderTopLeftRadius`

STÍLUSATTRIBÚTUM PROGRAMOZÁSA

```
<div style="position: absolute" id="movingElement"></div>
<script>
  document.querySelector("#movingElement").style.top  = "25px";
  document.querySelector("#movingElement").style.left = "42px";
</script>
```

STÍLUSOBJEKTUM

- `elem.style`: `CSSStyleDeclaration` objektum
- Az összes stílustulajdonságot tartalmazza
- Tetszőleges tulajdonság beállítható (írás)
- Lekérdezhető stílustulajdonságok (olvasás):
 - a `style` attribútumon keresztül voltak megadva;
 - JavaScriptből határoztuk meg az értéküket.

PÉLDA

```
<style>
.box {
  position: absolute;
  width: 100px; height: 100px;
}
</style>
<div class="box" style="left: 20px"></div>
```

```
const box = document.querySelector("div");
box.style.top = "30px";

box.style.top;      // "30px"  <-- JS
box.style.left;    // "20px"  <-- style attribute
box.style.width;   // ""
box.style.position; // ""
```

SZÁMÍTOTT STÍLUS

- `window.getComputedStyle(elem)`
- A böngésző által nyilvántartott stílustulajdonságok
- A rövidítések (pl. `border`, `background`, stb.) nem érhető el, csak az elemi tulajdonságok.

PÉLDA

```
<style>
.box {
  position: absolute;
  width: 100px; height: 100px;
}
</style>
<div class="box" style="left: 20px"></div>
```

```
const box = document.querySelector("div");
box.top = "30px";

const computedStyle = window.getComputedStyle(box);
computedStyle.top      // "30px"
computedStyle.left     // "20px"
computedStyle.width    // "100px"
computedStyle.position // "absolute"
```

STÍLUSOSZTÁLY PROGRAMOZÁSA

`classList` tulajdonság

- `add(osztály)`
- `remove(osztály)`
- `toggle(osztály)`
- `contains(osztály)`

```
<div class="rodian bounty-hunter">Greedo</div>
```

STÍLUSOSZTÁLY PROGRAMOZÁSA

`add`, `remove`, `toggle`, `contains`

```
<div class="red green blue">
```

```
const div = document.querySelector('div');
div.classList.remove("green");
div.classList.add("pink");

// változatás
div.classList.toggle("pink");

// feltételes megjelenítés
div.classList.toggle("pink", i < 10);

// van-e adott stílusosztály
div.classList.contains("red"); // true

// több hozzáadása egyszerre
div.classList.add("orange", "yellow");
```

STÍLUSLAPOK

- `document.styleSheets` → tömb (`StyleSheet`)
- Fontosabb tulajdonságok
 - `type`
 - `disabled` (stíluslap dinamikus ki-bekapcsolása)
 - `href`
 - `title`
 - `cssRules`: szabályok módosítása
 - `cssText`: a szabály szöveges formája
 - `selectorText` (nem szabványos)
 - `style`: az adott szabály stíluslistája (mint az inner style)

DOM TULAJDONSÁGOK ÉS METÓDUSOK

ATTRIBÚTUM PROGRAMOZÁS

- `getAttribute(name)`
- `setAttribute(name, value)`
- `hasAttribute(name)`
- `removeAttribute(name)`
- `toggleAttribute(name[, predicate])`

```
<button type="submit" data-id="10" disabled>A button</button>
```

```
const b = document.querySelector('button')
// reading
b.getAttribute('type')      // 'submit'
b.getAttribute('data-id')   // '10'
b.getAttribute('disabled')  // ''
b.getAttribute('foo')       // null or ''
b.hasAttribute('disabled') // true
// writing
b.setAttribute('name', 'send')
b.setAttribute('hidden', '') // boolean attribute
b.toggleAttribute('disabled') // toggle boolean attribute
```

ATTRIBÚTUM PROGRAMOZÁS

`data-*` attribútum → `dataset` tulajdonság

```
<div  
  data-id="10"  
  data-some-text="foo"  
></div>
```

```
const div = document.querySelector('div')  
// reading  
div.dataset.id      // "10"  
div.dataset.someText // "foo"  
// writing  
div.dataset.someText = "bar"  
div.dataset.user = "John"           // --> data-user="John"  
div.dataset.dateOfBirth = "2000-12-12" // --> data-date-of-birth="2000-12-12"  
// removing  
delete div.dataset.id  
// checking  
'someText' in el.dataset // true  
'id' in el.dataset      // false
```

ATTRIBÚTUM PROGRAMOZÁS

- `attributes`: élő lista, tömbszerű objektum
- `getAttributeNames()`: szövegtömb

SZERKEZET BEJÁRÁSA

- gyerekek
 - `childNodes`, `firstChild`, `lastChild`
 - `children`, `firstElementChild`, `lastElementChild`
- szülő/ős
 - `parentNode`, `parentElement`
 - `closest(cssSelector)`
- testvérek
 - `nextSibling`, `previousSibling`
 - `nextElementSibling`, `previousElementSibling`

Csomópont típusa

- `nodeType === 1`: elem
- `nodeType === 3`: szöveges csomópont

SZERKEZET MÓDOSÍTÁSA

- `parent.appendChild(elem)`: beszúrás
- `parent.insertBefore(elem, ref)`: beszúrás
- `parent.removeChild(elem)`: törlés
- `parent.replaceChild(elem, oldElem)`: csere
- Ha `elem` létezik, akkor mozgatás

SZERKEZET LEKÉRDEZÉSE

- `parent.contains(element)`: tartalmazza-e

PÉLDA: HÁNYADIK ELEM

```
<ul>
  <li>első</li>
  <li>második</li>
  <li>harmadik</li>
  <li>negyedik</li>
  <li>ötödik</li>
</ul>
```

```
function getPosition(element) {
  let count = 0;
  while ( null != el ) {
    el = el.previousElementSibling;
    count++;
  }
  return count;
}

const secondChild = document.querySelector('ul > li:nth-child(2)');
const position = getPosition(secondChild); // 2
```

PÉLDA: HÁNYADIK ELEM

```
<ul>
  <li>első</li>
  <li>második</li>
  <li>harmadik</li>
  <li>negyedik</li>
  <li>ötödik</li>
</ul>
```

```
function getPosition(element) {
  const parent = element.parentNode;
  const children = Array.from(parent.children);

  return children.indexOf(element) + 1;
}

const secondChild = document.querySelector('ul > li:nth-child(2)');
const position = getPosition(secondChild); // 2
```

RÉGI MEGOLDÁSOK

BÖNGÉSZŐ I/O

- `alert(text)`
- `confirm(text)`
- `prompt(text, default)`

```
alert('The Force will be with you. Always.');

const kerdes = confirm('Judge me by my size, do you?');

const target = prompt(`  
You would prefer another target?  
A military target?  
Then name the system!  
`, 'Dantooine');

console.log(`  
${target} is too remote to make an effective demonstration!
`);
```

ELEMEK KIVÁLASZTÁSA

- `document`
 - `getElementById(id)`
 - `getElementsByName(name)`
- `document/elem`
 - `getElementsByTagName(tagName)`
 - `getElementsByClassName(className)`
 - `querySelector(css_selector)`
 - `querySelectorAll(css_selector)`

STÍLUSOSZTÁLY PROGRAMOZÁSA

- `className` tulajdonság
- `class` attribútum szöveges értéke
- olvasható/írható
- elavult

```
<div class="human hero">Aragorn</div>
```

```
const div = document.querySelector('div')
div.className // 'human hero'
div.className = 'human king'
```

TESZTELÉS

KERETRENDSZER HASZNÁLATA

```
<!-- keretrendszer betöltése -->
<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.4.0/jasmine.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.4.0/jasmine.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.4.0/jasmine-html.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jasmine/3.4.0/boot.min.js"></script>

<!-- az alkalmazás és teszt betöltése -->
<script src="app.js"></script>
<script src="app.test.js"></script>
```

```
// app.test.js
describe('factorial', () => {
  it('0! should be 1', () => {
    expect(factorial(0)).toBe(1);
  })
})
```

```
describe('factorial', () => {
  it('0! should be 1', function () {
    expect(factorial(0)).toBe(1);
  });
  it('1! should be 1', function () {
    expect(factorial(1)).toBe(1);
  });
  it('5! should be 120', function () {
    expect(factorial(5)).toBe(120);
  });
});
```

Factorial page

3.4.0

Options

• • •

3 specs, 0 failures, randomized with seed 06774

finished in 0.999s

```
factorial
  factorial
    • 0! should be 1
    • 1! should be 1
    • 5! should be 120
```

PÉLDA – HIBA

```
describe('factorial', () => {
  /* ... */
  it('5! should be 120', function () {
    expect(factorial(5)).toBe(1200);
  });
});
```

Factorial page

3.4.0

Options

✖ • •

3 specs, 1 failure, randomized with seed 77429

finished in 0.994s

[Spec List](#) | [Failures](#)

factorial > factorial > 5! should be 120

Expected 120 to be 1200.

```
Error: Expected 120 to be 1200.  
  at <Jasmine>  
  at r.<anonymous> (http://webprogramozas.inf.elte.hu/assets/codes/02/app.test.2.js:11:34)  
  at <Jasmine>
```

ÖSSZEFOGLALÁS

- DOM a felhasználói felület programozási interfésze
- Elemek kiválasztása:

```
document.querySelector(css-selector)
```

- Tulajdonságai: analóg módszer
- Alap eseménykezelés:

```
elem.addEventListener('click', onClick)
```