

WEBPROGRAMOZÁS

ALKALMAZÁSFJEJLESZTÉSI ALAPELVEK, KÓDSZERVEZÉS

Horváth Győző

egyetemi docens

horvath.gyozo@inf.elte.hu

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

Tel: (1) 372-2500/8469

ISMÉTLÉS

ISMÉTLÉS - NYELVI ELEMEK

- ✓ Dinamikusan típusos
- ✓ Interpretált nyelv
- ✓ Szintaxis: { utasítások }
- ✓ Adatszerkezetek (elemi, összetett)
 - [1, 2, 3] – tömb
 - { nev: "Győző" } – objektum
- ✓ Funkcionális aspektus
- ✓ OOP-s aspektus

ISMÉTLÉS - DOM

-  HTML elemek belső ábrázolása
-  Programozási interfész (API)
-  Bemeneti-kimeneti interfész

ISMÉTLÉS – DOM

✓ Elemek kiválasztása

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

✓ Elem (JavaScript objektum) tulajdonságai

- Analógia: Webfejlesztés → Webprogramozás
- írás/olvasás
- tulajdonságok (pl. `innerHTML`)
- metódusok

✓ Eseménykezelés

- Eseménytípusok

ISMÉTLÉS – ESEMÉNYKEZELÉS

- ✓ `elem.addEventListener(type, handler)`
- ✓ Interakció eszköze
- ✓ Mini-programok
- ✓ Eseményobjektum (`event`)
- ✓ Alapértelmezett műveletek megakadályozása
(`preventDefault`)
- ✓ Buborékolás
- ✓ Delegálás (`delegate`)

ALKALMAZÁSFEJLESZTÉSI ALAPELVEK

EDDIG

- JavaScript nyelvi elemek
 - változók, konstansok
 - függvények
 - tömbök, objektumok, osztályok
- DOM
- Eseménykezelés
 - eseményobjektum
 - delegálás
- **SZERVEZŐ ELV?**

TANULSÁGOK

- Eseménykezelő függvény
 - beolvasás
 - feldolgozás
 - kiírás
- Biztosan dolgozik a DOM-mal
- Kapcsolatot teremt a DOM és a nyelvi elemek között
- Hol legyen az adat?

ADATTÁROLÁS HELYE

LEHETŐSÉGEK

```
// ✓ Ebből könnyű HTML-t generálni
let count = 3
const movies = [
  { id: 1, title: 'The Shack', year: 2017, seen: true },
  { id: 2, title: 'Fireproof', year: 2008, seen: true },
  { id: 3, title: 'Courageous', year: 2011, seen: false },
]
```

```
<!-- ✓ Előfordulnak olyan adatok, amelyeket a HTML attribútumban
      kell tárolni -->
<div data-id="1">The Shack</div>
```

```
<!-- ✗ Ebből nehéz az adatot kinyerni -->
<ul>
  <li data-id="1" class="seen" >The Shack (2017)</li>
  <li data-id="2" class="seen" >Fireproof (2008)</li>
  <li data-id="3" class="unseen">Courageous (2011)</li>
</ul>
```

TODO LISTA



TÁROLÁS A DOM-BAN (1)

```
<input id=" newItem">
<button id=" addItem">Add</button>
<ul id=" todoList">
  <li>Buy milk</li>
  <li>Learn JavaScript</li>
</ul>
```

```
const todoList = document.querySelector("#todoList");
const button = document.querySelector("#addItem");
const input = document.querySelector("#newItem");

function handleButtonClick() {
  const newItem = input.value;
  todoList.innerHTML += `<li>${newItem}</li>`;
}

button.addEventListener("click", handleButtonClick);
```

TÁROLÁS A DOM-BAN (2)

```
<input id=" newItem">
<button id=" addItem">Add</button>
<ul id=" todoList"></ul>
```

```
const todoList = document.querySelector("#todoList");
const button = document.querySelector("#addItem");
const input = document.querySelector("#newItem");

function handleButtonClick() {
    const newItem = input.value;
    const newListItem = document.createElement("li");
    newListItem.innerHTML = newItem;
    todoList.appendChild(newList);
}

button.addEventListener("click", handleButtonClick);
```

TÁROLÁS A DOM-BAN (3)

Todo lista elemei

```
const input = document.querySelector("input");
const button = document.querySelector("button");
const listElement = document.querySelector("ul");
const listItems = document.querySelectorAll("li");

function handleButtonClick() {
    // Input
    const newItem = input.value;
    const listContent = Array.from(list).map(li => li.innerText);
    // Process
    listContent.push(newItem);
    // Output
    const newListItem = document.createElement("li");
    newListItem.innerHTML = newItem;
    todoList.appendChild(newListItem);
}

button.addEventListener("click", handleButtonClick);
```

TÁROLÁS A DOM-BAN

TÁROLÁS A DOM-BAN (ÖSSZEFoglalás)

- Az adatot a DOM-ban tároljuk
- Mindig onnan kell kiolvasni
- Nem alap nyelvi elemekkel dolgozunk
- Adat és feldolgozó függvény szétválik
- Probléma: egységbe zárás, tárolás

ADAT ÉS MEGJELENÉS SZÉTVÁLASZTÁSA

- **Alkalmazásfejlesztési alapelv**
- Minél kisebb érintkezési felület az adat és felület között
- Könnyebb egységbe zárás
- Alapvető nyelvi elemeket használ
- A DOM (I/O) lassú

ADAT ÉS MEGJELENÉS SZÉTVÁLASZTÁSA

ADAT ÉS MEGJELENÉS SZÉTVÁLASZTÁSA

```
const list = [];

const input = document.querySelector("input");
const button = document.querySelector("button");
const listElement = document.querySelector("ul");

function handleButtonClick() {
    // Input
    const newItem = input.value;
    // Process
    list.push(newItem);
    // do something else with the data
    // Output
    const newElement = document.createElement("li");
    newElement.innerHTML = newItem;
    listElement.appendChild(newElement);
}

button.addEventListener("click", handleButtonClick);
```

(Imperatív felületkezelés)

ADAT ÉS MEGJELENÉS SZÉTVÁLASZTÁSA

A felület mint az adat leképezése (függvény)

```
const list = [];  
  
function renderList(list) {  
  return list.map(item => `<li>${item}</li>`).join("");  
}  
  
// ...  
  
function handleButtonClick() {  
  // Input  
  const newItem = input.value;  
  // Process  
  list.push(newItem);  
  // Output  
  listElement.innerHTML = renderList(list);  
}  
  
button.addEventListener("click", handleButtonClick);
```

(Deklaratív felületkezelés)

MEGOLDÁS RÉSZEI

FELÜLET KEZELÉSE

KIÍRÁS A DOM-BA

▪ Imperatív megközelítés

- Ha szükséges megőrizni az adott DOM elemet
(van belső állapotuk, pl. input, animáció, DOM-ban tárolás)
- Direkt manipuláció

▪ Deklaratív megközelítés

- Ha az elemek újragenerálhatóak (nincs belső állapotuk)
- Adat leképezése felületi elemekre
- UI = render(data)
- HTML generálók

IMPERATÍV

```
function handleButtonClick() {
  const newItem = input.value;
  list.push(newItem);
  // Output
  const newElement = document.createElement("li");
  newElement.innerHTML = newItem;
  listElement.appendChild(newElement);
}
```

DEKLARATÍV

```
function renderList(list) {
  return list.map(item => `<li>${item}</li>`).join("");
}

const input = document.querySelector("input");
const button = document.querySelector("button");
const listElement = document.querySelector("ul");

function handleButtonClick() {
  // Input
  const newItem = input.value;
  // Process
  list.push(newItem);
  // Output
  listElement.innerHTML = renderList(list);
}
```

FIZIKAI ÉS LOGIKAI EGYSÉGEK

KÓDSZERVEZÉS

- Fizikai
 - külön fájl
 - *modul*
- Logikai
 - függvény
 - osztály (egységbe zárás)
 - *modul*

FIZIKAI CSOPORTOSÍTÁS

- Külön fájlokba funkció szerint
- Függőségek kézi kezelése

```
// math.js
const add = (a, b) => a + b;
```

```
// app.js
console.log(add(40, 2));
```

```
<body>
  <!-- ... -->
  <script src="math.js"></script>
  <script src="app.js"></script>
</body>
```

FIZIKAI CSOPORTOSÍTÁS

- Külön fájlokba: **modul**
- Függőségek automatikus kezelése

```
// math.js
const add = (a, b) => a + b;
export { add };
```

```
// app.js
import { add } from "./math.js";
console.log(add(40, 2));
```

```
<body>
<!-- ... -->
<script type="module" src="app.js"></script>
</body>
```

MODULOK – EXPORT

```
// in-place export
export const add = (a, b) => a + b;
export const multiply = (a, b) => a * b;

// separate export
const add = (a, b) => a + b;
const multiply = (a, b) => a * b;
export { add, multiply };

// default export
export default const add = (a, b) => a + b;

// rename exports
const add = (a, b) => a + b;
const multiply = (a, b) => a * b;
export { add as customAdd, multiply as customMultiply };

// export from module
export * from "./other.js";
```

Referencia

MODULOK – IMPORT

```
// import entities
import { add, multiply } from "./math.js";

// import defaults
import add from "./math.js";

// rename imports
import { add as mathAdd } from "./math.js";

// import module object
import * as MyMath from "./math.js";

// import just for side effects
import "./something.js";

// import URL
import * as MyMath from "http://some.where.hu/math.js"
```

Referencia

MODULOK

- Strict mode
- Nem globális scope
 - nehezebb debugolás
(konzolon nem érhető el)
- File protocol-on nem működik
- Node.js `serve` package
 - Node.js és npm installálása
 - `npx serve`
 - `http://localhost:5000`
- VS Code Live Server extension

LOGIKAI CSOPORTOSÍTÁS

- Függvények
 - elemi egység
 - műveletek strukturálása

```
// Helper/utility function
function range(n) {
  return Array.from({length: n}, (e, i) => i + 1);
}

// HTML generator
function genList(list) {
  return `<ul>${list.map(e => `<li>${e}</li>`).join('')}</ul>`;
}

// Business logic
const add = (a, b) => a + b;

// Event handler
function onClick(e) {
  // ...
}
```

LOGIKAI CSOPORTOSÍTÁS

- Osztályok
 - magasabb szintű egység
 - műveletek és adatok egységbe zárása

```
class Tile {  
    constructor(x, y) {  
        this.x = y;  
        this.y = y;  
    }  
  
    get coords() {  
        return {x, y};  
    }  
  
    distance(tile) {  
        return Math.sqrt(  
            (tile.x - this.x) ** 2 + (tile.y - this.y) ** 2  
        );  
    }  
}
```

EGYSÉGBE ZÁRÁS

EGYSÉGBEZÁRÁS 1.

Globális változók és metódusok

```
let number = 0;

function increase() {
    number++;
}

function init() {
    number = 0;
}

increase();
console.log(number); // 1
```

EGYSÉGBEZÁRÁS 2.

Objektum, ~névtér

```
const game = {
  number: 0,
  increase: function() {
    this.number++;
  },
  init: function () {
    this.number = 0;
  }
}

game.init();
game.increase();
console.log(game.szam); // 1
```

EGYSÉGBEZÁRÁS 3.

Osztály

```
class Game {  
    constructor() {  
        this.number = 0;  
    }  
    increase() {  
        this.number++;  
    }  
}  
  
const game = new Game();  
game.increase();  
console.log(game.number); // 1
```

EGYSÉGBEZÁRÁS 4.

Függvény, saját hatókörrel (felfedő modul minta), IIFE

```
const game = (function () {
    let number = 0;

    function increase() {
        number++;
    }

    function init() {
        number = 0;
    }

    function getNumber() {
        return number;
    }

    return { increase, getNumber};
})();

game.increase();
console.log(game.getSzam()); // 1
```

EGYSÉGBEZÁRÁS 5.

Modul, függvény

```
// game.js
let number = 0;

export function increase() {
    number++;
}

export function init() {
    number = 0;
}

export function getNumber() {
    return number;
}
```

```
// main.js
import { increase, init, getSzam } from "./game.js";

increase();
console.log(getSzam()); // 1
```

EGYSÉGBEZÁRÁS 6.

Modul, osztály

```
// game.js
export class Jatek {
  constructor() {
    this.number = 0;
  }

  increase() {
    this.counter += 1;
  }
}
```

```
import { Jatek } from './game.js';

const jatek = new Jatek();
jatek.increase();
console.log(jatek.number);
```

ALKALMAZÁSFEJLESZTÉS LÉPÉSEI

FELADATMEGOLDÁS LÉPÉSEI

1. Felhasználói felület (UI)

- Tervezés
- HTML, CSS

2. Üzleti logika (BL)

- adatok
- függvények

3. Eseménykezelő függvények

- beolvasás (DOM)
- feldolgozás (BL)
- kiírás (DOM)

PÉLDA

SZÁMKITALÁLÓS JÁTÉK

SZÁMKITALÁLÓS JÁTÉK

Számkitalálós játék

Gondoltam egy számra 1 és 100 között. Találd ki!

FELÜLETI TERV

```
<input id="tipp">
<button id="tippGomb">Tipp!</button>
<ul id="tippek"></ul>
```

ÁLLAPOTTÉR

```
let kitalalandoSzam = 42;
let vege = false;
const tippek = [50, 25, 38, 44];
```

ÁLLAPOT VÁLTOZÁSA

```
function tipp(tippeltSzam) {  
    tippek.push(tippeltSzam);  
    vege = (tippeltSzam === kitalaldoSzam);  
}
```

SEGÉDFÜGGVÉNYEK

```
function veletlenEgesz(min, max) {  
    const veletlen = Math.random();  
    const tartomany = max - min + 1;  
    return Math.trunc(veletlen * tartomany) + min;  
}
```

ESEMÉNYKEZELŐ FÜGGVÉNYEK

```
const tippInput = document.querySelector("#tipp");
const gomb = document.querySelector("#tippGomb");
const tippLista = document.querySelector("#tippek");

gomb.addEventListener("click", tippeles);
function tippeles(e) {
    // beolvasás
    const tippeltSzam = parseInt(tippInput.value);
    // feldolgozás
    tipp(tippeltSzam);
    // kiírás
    // deklaratív felületkezelés
    tippLista.innerHTML = tippek.map(szam =>
        `<li>${szam} (${hasonlit(szam, kitalandoSzam)})</li>`)
    .join("");
    // imperatív felületkezelés
    gomb.disabled = vege;
}
function hasonlit(szam, kitalandoSzam) {
    if (szam < kitalandoSzam) return "nagyobb";
    if (szam > kitalandoSzam) return "kisebb";
    return "egyenlő".
```

HTML GENERÁTOROK

```
function genLista(tippek, kitalaldoSzam) {
    return tippek.map(szam =>
        `<li>${szam} (${hasonlit(szam, kitalaldoSzam)})</li>`
    ).join("");
}
```

```
function tippeles(e) {
    // beolvasás
    const tippeltSzam = parseInt(tippInput.value);
    // feldolgozás
    tipp(tippeltSzam);
    // kiírás
    tipplista.innerHTML = genLista(tippek, kitalaldoSzam);
    gomb.disabled = vege;
}
```

PÉLDA

MEMÓRIAJÁTÉK

MEMÓRIAJÁTÉK

n =

m =

FELÜLET, HTML

```
<div id="main">
  <form action="">
    n = <input type="number" id="n" value="3"> <br>
    m = <input type="number" id="m" value="4">
    <button type="button">Start new game</button>
  </form>
  <div id="board"></div>
  <div id="status"></div>
</div>
<script src="game.js"></script>
<script src="index.js"></script>
```

```
<!-- Inside #board -->
<table>
  <tr>
    <td>
      <div class="card">
        <div class="front">1</div>
        <div class="back"></div>
      </div>
    </td>
  </tr>
</table>
```

ÁLLAPOTTÉR

```
let board = []
let gameState = 0 // 0, 1, 2

function init() {
  board = []
  gameState = 0
}
function initBoard(n, m) {
  const numbers = Array(n * m / 2).fill(0).map((e, i) => i + 1)
  const values = [...numbers, ...numbers].sort((a, b) => Math.random() < 0.5 ? 1
  board = Array(n).fill(0).map(() => Array(m).fill(0).map(() => ({
    value: values.shift(),
    flipped: false,
    solved: false,
  })))
  gameState = 1
}
function selectCard(i, j) {
  const flipped = flippedCards()
  if (isFlipped(i, j) || isSolved(i, j) || flipped.length === 2) {
    return
  }
}
```

ESEMÉNYKEZELŐK

```
const form = document.querySelector('form')
const button = form.querySelector('button')
const boardDiv = document.querySelector('#board')
const statusDiv = document.querySelector('#status')

function xyCoord(card) {
    const td = card.closest('td')
    const x = td.cellIndex
    const tr = td.parentNode
    const y = tr.sectionRowIndex
    return { x, y }
}

button.addEventListener('click', onGenerate)
function onGenerate(e) {
    e.preventDefault()
    const n = form.querySelector('#n').valueAsNumber
    const m = form.querySelector('#m').valueAsNumber
    if (n * m % 2 !== 0) {
        return
    }
```

PÉLDA – DEKLARATÍV

n =
m =

IMPERATÍV KIEGÉSZÍTÉSEK

```
function onSelectCard(e) {
    const card = e.target.closest('.card')
    if (boardDiv.contains(card)) {
        if (flippedCards().length === 2) {
            return
        }
        const {x, y} = xyCoord(card)
        selectCard(y, x)
        update()
        if (flippedCards().length === 2) {
            setTimeout(turnBackAndRender, 1000)
        }
    }
}
function turnBackAndRender() {
    turnBack()
    update()
}
function render() {
    renderBoard(board)
    renderStatus(countSolved(), gameState)
```

PÉLDA – IMPERATÍV

n =
m =

MODULARIZÁLÁS

```
<!-- ... -->
<script type="module" src="index.js"></script>
```

```
import { xyCoord } from "./helper.js";
import { Game } from "./game.js";
import { render, update } from "./render.js";

const form = document.querySelector('form')
const button = form.querySelector('button')
const boardDiv = document.querySelector('#board')
const statusDiv = document.querySelector('#status')

const game = new Game()

button.addEventListener('click', onGenerate)
function onGenerate(e) {
    // ...
    game.initBoard(n, m)
    render(game)
}

boardDiv.addEventListener('click', onSelectCard)
function onSelectCard(e) {
    const card = e.target.closest('.card')
    if (boardDiv.contains(card)) {
```

GAME.JS

```
export class Game {
  constructor() {
    this.board = []
    this.gameState = 0 // 0, 1, 2
  }

  init() {
    this.board = []
    this.gameState = 0
  }
  initBoard(n, m) {
    const numbers = Array(n * m / 2).fill(0).map((e, i) => i + 1)
    const values = [...numbers, ...numbers].sort((a, b) => Math.random() < 0.5 ?
      this.board = Array(n).fill(0).map(() => Array(m).fill(0).map(() => ({
        value: values.shift(),
        flipped: false,
        solved: false,
      })))
      this.gameState = 1
    }
    selectCard(i, j) {
      ...
    }
  }
}
```

RENDER.JS

```
const boardDiv = document.querySelector('#board')
const statusDiv = document.querySelector('#status')

export function render(game) {
  renderBoard(game.board)
  renderStatus(game.countSolved(), game.gameState)
}

export function update(game) {
  updateBoard(game.board)
  renderStatus(game.countSolved(), game.gameState)
}

export function updateBoard(board) {
  board.forEach((row, i) => row.forEach((card, j) => {
    document.querySelector(`table tr:nth-child(${i+1}) td:nth-child(${j+1}) .card`)
    document.querySelector(`table tr:nth-child(${i+1}) td:nth-child(${j+1}) .card`)
  }))
}

export function renderBoard(board) {
  boardDiv.innerHTML =
    <table>
```

HELPER.JS

```
export function xyCoord(card) {
  const td = card.closest('td')
  const x = td.cellIndex
  const tr = td.parentNode
  const y = tr.sectionRowIndex
  return { x, y }
}
```

ÖSSZEFOGLALÁS

- Alkalmazásfejlesztési alapelvek
 - adat és megjelenítés szétválasztása
 - adatréteg, eseménykezelők, HTML generátorok
 - felület deklaratív és imperatív kezelése
- Kódszervezés
 - fizikai, logikai
 - egységbe zárás
 - adattárolás helye