

# WEBPROGRAMOZÁS

## INTERAKTÍV PROGRAMOK KÉSZÍTÉSE A BÖNGÉSZŐBEN: ESEMÉNYKEZELÉS

**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 C++-hoz hasonló
- ✓ 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
- `elem.addEventListener(típus, fgv)`

# ISMÉTLÉS – DOM

```
<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}!`;
}

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

function handleHelloClick() {
  const name = input.value;
  const greeting = greet(name);
  output.innerHTML = greeting;
}

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

# ESEMÉNYKEZELÉS

## INTERAKTÍV PROGRAMOK

# FOGALMAK

- Interaktív programok
- Felhasználói tevékenység
- Esemény
- Eseménykezelő függvények
- → **ESEMÉNYVEZÉRELT PROGRAMOZÁS**

# ESEMÉNYKEZELŐK MINT MINI PROGRAMOK

```
function handleEvent() {  
    // Read  
    // Process  
    // Write  
}
```

# ESEMÉNYEK

- `click`: egérkattintás
- `mousemove`: egérmozgatás
- `mousedown`: egér gombjának lenyomása
- `mouseup`: egér gombjának felenegedése
- `input`: input mező értékének megváltozása
- `keydown`: billentyűzet gombjának lenyomása
- `keyup`: billentyűzet gombjának felengedése
- `keypress`: billentyűzet gombjának megnyomása
- `submit`: űrlap elküldése
- `scroll`: görgetés az oldalon

## Események referenciaja

# ESEMÉNYKEZELŐ FÜGGVÉNYEK REGISZTRÁLÁSA

```
// Egyszerűbb esetekben
element.addEventListener(eventType, eventHandler);
element.removeEventListener(eventType, eventHandler);

// Például
const button = document.querySelector("button");

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

function handleButtonClick() {
    // mi történjen kattintáskor
}

// Helyben definiálva
element.addEventListener(eventType, function () {});
```

# ESEMÉNYKEZELŐ FÜGGVÉNYEK REGISZTRÁLÁSA

```
target.addEventListener(eventType, eventListener[, options]);
// `options` object:
// - capture: Boolean (elkapás iránya)
// - once: Boolean (egyszeri hívás, majd eltávolítás)
// - passive: Boolean (nincs preventDefault() hívás)

target.removeEventListener(eventType, eventListener[, options]);
// options object
// - capture: Boolean

// Csak a capture flag számít az eseménykezelő azonosításában
// a type és listener mellett
// Névtelen függvényt nem lehet eltávolítani (nincs rá referencia)
// kivéve: `once` option

target.addEventListener('click', onClick, { once: true });
```

# ESEMÉNYKEZELŐ FÜGGVÉNYEK REGISZTRÁLÁSA

```
// Egy elem egy eseményéhez több eseménykezelő függvény  
button.addEventListener("click", handleClick1);  
button.addEventListener("click", handleClick2);  
  
// Több eseményhez ugyanaz az eseménykezelő függvény  
button1.addEventListener("click", handleClick);  
button2.addEventListener("click", handleClick);
```

# ESEMÉNYKEZELŐ PÉLDA

```
<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}!`;
}

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

function handleHelloClick() {
  const name = input.value;
  const greeting = greet(name);
  output.innerHTML = greeting;
}

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

## PÉLDA

- Feladat:
  - Adott hivatkozások felsorolása
  - Cél: SHIFT gomb nyomva tartásával kattintva a konzolra kiírjuk a megnyitandó oldal URL-jét

- ELTE
- ELTE IK
- ELTE IK MOT

# KÉRDÉSEK

- Honnan tudjuk, hogy a SHIFT billentyű le van-e nyomva?
- Hogyan tiltom le a hivatkozás követését?
- Hogyan skálázható a megoldás a hivatkozások számával?

```
<ul>
  <li>
    <a href="http://www.elte.hu">ELTE</a>
  </li>
  <li>
    <a href="http://www.inf.elte.hu">ELTE IK</a>
  </li>
  <li>
    <a href="http://www.inf.elte.hu/mot">ELTE IK MOT</a>
  </li>
</ul>
```

# AZ ESEMÉNYOBJEKTUM

Az eseménykezelő függvények első paraméterként automatikusan megkapják

```
function handleEvent(event) {  
    console.log(event);  
}
```

# AZ ESEMÉNYOBJEKTUM

Tartalma az eseménytől függ:

Általános tulajdonságok pl. `type`, `target`

---

Egéresemény pl. `screenX`, `screenY`, `buttons`

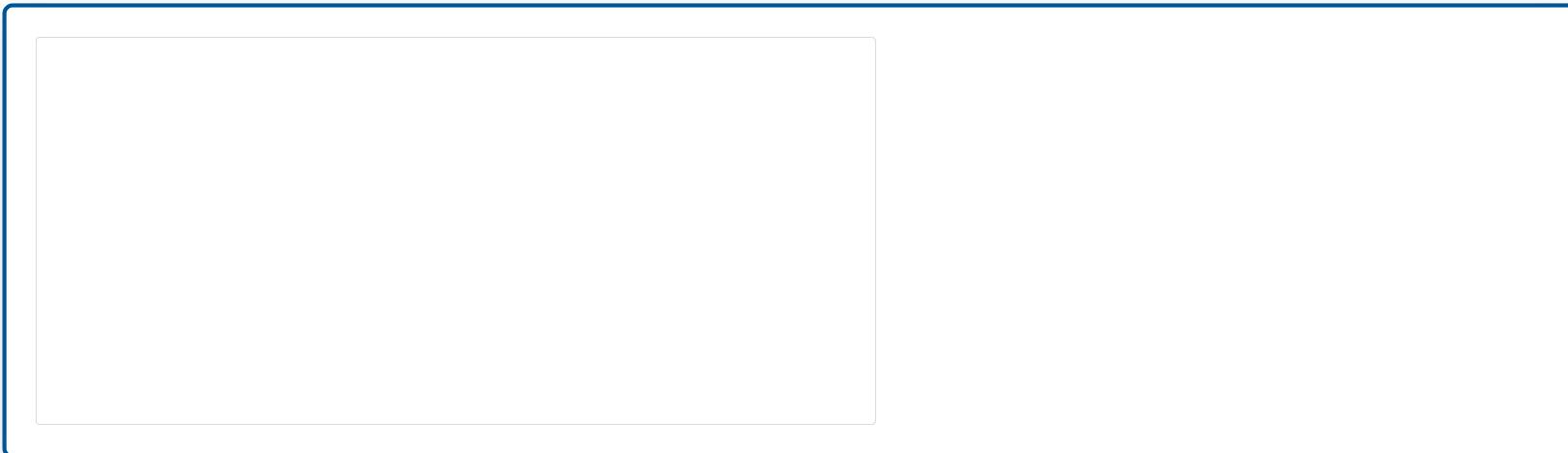
---

Billentyűzetesemény pl. `key`, `code`, `altKey`, `ctrlKey`

## Referencia

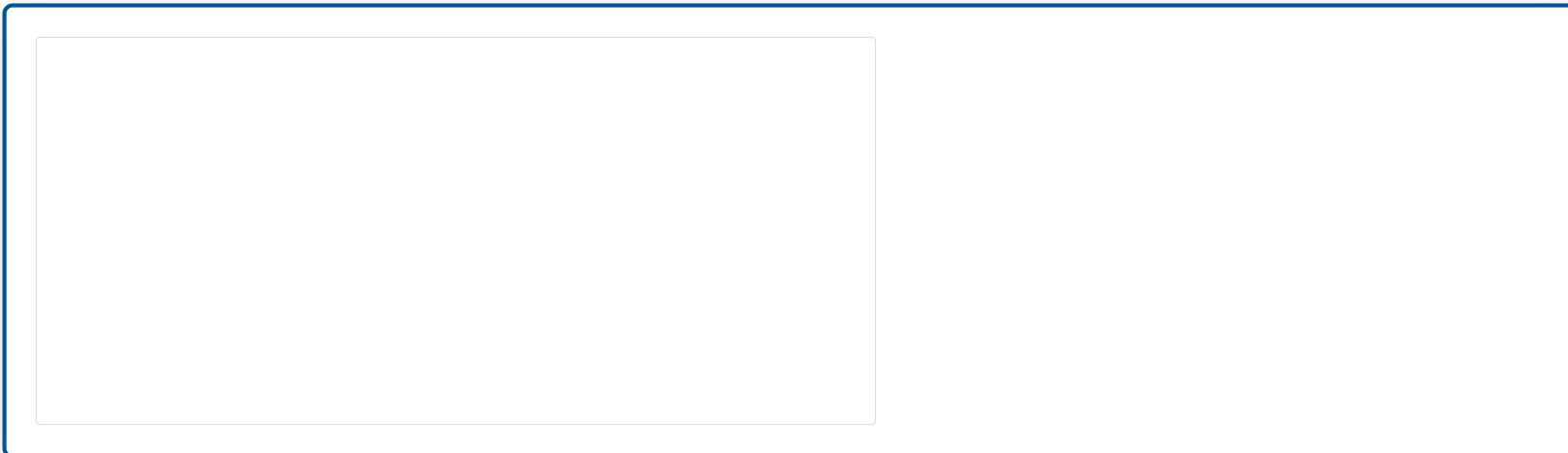
# BILLENTYÚZETESEMÉNY TULAJDONSÁGAI

key, code



# EGÉRESEMÉNY TULAJDONSÁGAI

`screenX`, `screenY`, `offsetX`, `offsetY`, `clientX`, `clientY`



## A MEGOLDÁS LÉPÉSEI

- SHIFT lenyomásának vizsgálata
- Alapértelmezett művelet letiltása
- Kattintott elem elérése
- Hozzárendelés az összes elemhez

# ESEMÉNYOBJEKTUM – PÉLDA

```
const link = document.querySelector('#link1');

function handleLinkClick(event) {
  if (!event.shiftKey) {
    return;
  }

  const href = link.href;
  console.log(href);
  alert('stop');
}

link.addEventListener('click', handleLinkClick);
```

- ELTE
- ELTE IK
- ELTE IK MOT

## A MEGOLDÁS LÉPÉSEI

-  SHIFT lenyomásának vizsgálata
-  Alapértelmezett művelet letiltása
-  Kattintott elem elérése
-  Hozzárendelés az összes elemhez

# ALAPÉRTELMEZETT MŰVELET MEGAKADÁLYOZÁSA

- Alapértelmezett műveletek:
  - Linkre kattintás → oldal betöltése
  - Submit gombra kattintás → űrlap elküldése
  - Beviteli mezőbe gépelés → karakterek beíródnak
- Megakadályozása
  - eseményobjektum `preventDefault()` metódusa

```
function handleEvent(event) {  
    event.preventDefault();  
}
```

# ALAPÉRTELMEZETT MŰVELET MEGAKADÁLYOZÁSA – PÉLDA

```
const link = document.querySelector('#link1');

function handleLinkClick(event) {
  if (!event.shiftKey) {
    return;
  }

  event.preventDefault();
  const href = link.href;
  console.log(href);
}

link.addEventListener('click', handleLinkClick);
```

- ELTE
- ELTE IK
- ELTE IK MOT

## A MEGOLDÁS LÉPÉSEI

- SHIFT lenyomásának vizsgálata
- Alapértelmezett művelet letiltása
- Kattintott elem elérése
- Hozzárendelés az összes elemhez

# HATÉKONYSÁGI MEGFONTOLÁSOK

```
const link1 = document.querySelector('#link1');
const link2 = document.querySelector('#link2');
const link3 = document.querySelector('#link3');

function handleLinkClick1(event) { /* ... */ }
function handleLinkClick2(event) { /* ... */ }
function handleLinkClick3(event) { /* ... */ }

link1.addEventListener('click', handleLinkClick);
link2.addEventListener('click', handleLinkClick2);
link3.addEventListener('click', handleLinkClick3);
```

# HATÉKONYSÁGI MEGFONTOLÁSOK

**this** az eseményre **figyelő** objektum (kezelőobjektum)

```
const link1 = document.querySelector('#link1');
const link2 = document.querySelector('#link2');
const link3 = document.querySelector('#link3');

function handleLinkClick(event) {
  if (!event.shiftKey) {
    return;
  }

  event.preventDefault();
  const href = this.href;
  console.log(href);
}

link1.addEventListener('click', handleLinkClick);
link2.addEventListener('click', handleLinkClick);
link3.addEventListener('click', handleLinkClick);
```

- **ELTE**
- **ELTE IK**
- **ELTE IK MOT**

## A MEGOLDÁS LÉPÉSEI

-  SHIFT lenyomásának vizsgálata
-  Alapértelmezett művelet letiltása
-  Kattintott elem elérése
-  Hozzárendelés az összes elemhez

# HATÉKONYSÁGI MEGFONTOLÁSOK

```
const linkList = document.querySelectorAll('#linkList li a');

function handleLinkClick(event) {
  if (!event.shiftKey) {
    return;
  }

  event.preventDefault();
  const href = this.href;
  console.log(href);
}

for (const link of linkList) {
  link.addEventListener('click', handleLinkClick);
}
```

- Ciklus?
- Nem hatékony
- Nem rugalmas
- 

- **ELTE**
- **ELTE IK**
- **ELTE IK MOT**

## A MEGOLDÁS LÉPÉSEI

- ✓ SHIFT lenyomásának vizsgálata
- ✓ Alapértelmezett művelet letiltása
- ✓ Kattintott elem elérése
- Hozzárendelés az összes elemhez 😐

# ESEMÉNYEK BUBORÉKOLÁSA ÉS DELEGÁLÁSA

- **Forrásobjektum:** az eseményt *kiváltó* objektum (`e.target`)
- **Események buborékolása:** az esemény a forrásobjektumtól kezdve sorban minden gyerek szülőjén is bekövetkezik
- Lehetséges magasabb szinten kezelní az eseményt, mint ahol bekövetkezik
- **Kezelőobjektum:** az eseményre *figyelő* objektum
  - az az objektum, amelyhez az eseménykezelő hozzá van rendelve (`this`)

```
forrás → szülő → szülő → ... → body → html → document → window
```

# ESEMÉNYEK BUBORÉKOLÁSA ÉS DELEGÁLÁSA

- **Delegálás:** az eseményt magasabb szinten kezeljük, de egy alacsonyabb szintű objektummal dolgozunk
- **Delegált objektum:** az az objektum, amellyel az eseménykezelőben dolgozni szeretnénk
- **Buborékolás megakadályozása:** `e.stopPropagation()`
- Pontosabban
  1. Föntről lefele (event capture)
  2. Lentről felülről (event bubbling)

```
forrás → szülő → szülő → ... → body → html → document → window
```

# ESEMÉNY DELEGÁLÁSA – PÉLDA

```
<ul id="linkList">
  <li>
    <a href="http://www.elte.hu">ELTE</a>
  </li>
  <li>
    <a href="http://www.inf.elte.hu">ELTE IK</a>
  </li>
  <li>
    <a href="http://www.inf.elte.hu/mot">ELTE IK MOT</a>
  </li>
</ul>
```

# ESEMÉNY DELEGÁLÁSA – PÉLDA

```
const linkList = document.querySelector('#linkList');

function handleListClick(event) {
  if (!event.shiftKey) {
    return;
  }

  const target = event.target;

  if (!target.matches('li a')) {
    return;
  }

  event.preventDefault();
  const href = target.href;
  console.log(href);
}

linkList.addEventListener('click', handleListClick);
```

# ESEMÉNY DELEGÁLÁSA – PÉLDA

```
<ul>
  <li>
    <a href="http://www.elte.hu">
      <span>ELTE</span>
    </a>
  </li>
  <li>
    <a href="http://www.inf.elte.hu">
      ELTE <span>IK</span>
    </a>
  </li>
  <li>
    <a href="http://www.inf.elte.hu/mot">
      Médiainformáció Tanszék
    </a>
  </li>
</ul>
```

A delegált elem közbülső elem!

- **ELTE**
- **ELTE IK**
- **Médiainformaticai Tanszék**

# ESEMÉNY DELEGÁLÁSA

```
function handleClick(event) {
  const handlerElement = this;
  const sourceElement = event.target;
  const selector = '.foo';

  let element = sourceElement;

  while (
    element !== handlerElement &&
    !element.matches(selector)
  ) {
    element = element.parentNode;
  }

  if (element !== handlerElement) {
    const targetElement = element;
    console.log(targetElement);
  }
}
```

# ESEMÉNY DELEGÁLÁSA

```
function handleClick(event) {
  const handlerElement = this;
  const sourceElement = event.target;
  const selector = '.foo';

  const closestElement =
    sourceElement.closest(selector);

  if (handlerElement.contains(closestElement)) {
    const targetElement = closestElement;
    console.log(targetElement);
  }
}
```

# ESEMÉNY DELEGÁLÁSA – PÉLDA

```
function onClick(event) {
    const handlerElement = this;
    const sourceElement = event.target;
    const selector = 'li a';

    const closestElement =
        sourceElement.closest(selector);

    if (handlerElement.contains(closestElement)) {
        const targetElement = closestElement;

        if (!event.shiftKey) {
            return;
        }

        event.preventDefault();
        console.log(targetElement.href);
    }
}
```

# “OKOS” **delegate** SEGÉDFÜGGVÉNY

```
function delegate(parent, type, selector, handler) {  
  
    function delegatedFunction(event) {  
        const handlerElement = this;  
        const sourceElement = event.target;  
  
        const closestElement =  
            sourceElement.closest(selector);  
  
        if (handlerElement.contains(closestElement)) {  
            const targetElement = closestElement;  
            handler.call(targetElement, event);  
        }  
    }  
  
    parent.addEventListener(type, delegatedFunction);  
}
```

# “RÖVID-OKOS” **delegate** SEGÉDFÜGGVÉNY

```
function delegate(parent, type, selector, handler) {  
    parent.addEventListener(type, function (event) {  
        const targetElement = event.target.closest(selector);  
  
        if (this.contains(targetElement)) {  
            handler.call(targetElement, event);  
        }  
    });  
}
```

# “BUTA” DELEGÁLÁS

```
function handleEvent(event) {
  if (!event.target.matches(selector)) {
    return;
  }
  // Do magic...
}
parent.addEventListener(type, handleEvent);
```

# ESEMÉNY DELEGÁLÁSA – PÉLDA

```
const list = document.querySelector('ul')

function handleListClick(event) {
  if (!event.shiftKey) {
    return;
  }

  event.preventDefault();
  // A this itt a delegate segédfüggvény által van beállítva
  console.log(this.href);
}

delegate(list, 'click', 'li a', handleListClick);
```

- ELTE
- ELTE IK
- Médiainformaticai Tanszék

## A MEGOLDÁS LÉPÉSEI

- ✓ SHIFT lenyomásának vizsgálata
- ✓ Alapértelmezett művelet letiltása
- ✓ Kattintott elem elérése
- ✓ Hozzárendelés az összes elemhez

# ESEMÉNYEK DELEGÁLÁSA

- Hatékony programozási minta  
(egy elem - egy eseménykezelő)
- Sok elemnél / valamelyen típusú elemre általánosan
- Dinamikusan beszúrt elemeknél
- Viselkedés hozzárendelése elemekhez deklaratívan

# ESEMÉNYEK DELEGÁLÁSA – PÉLDA

Viselkedés deklaratív hozzárendelése

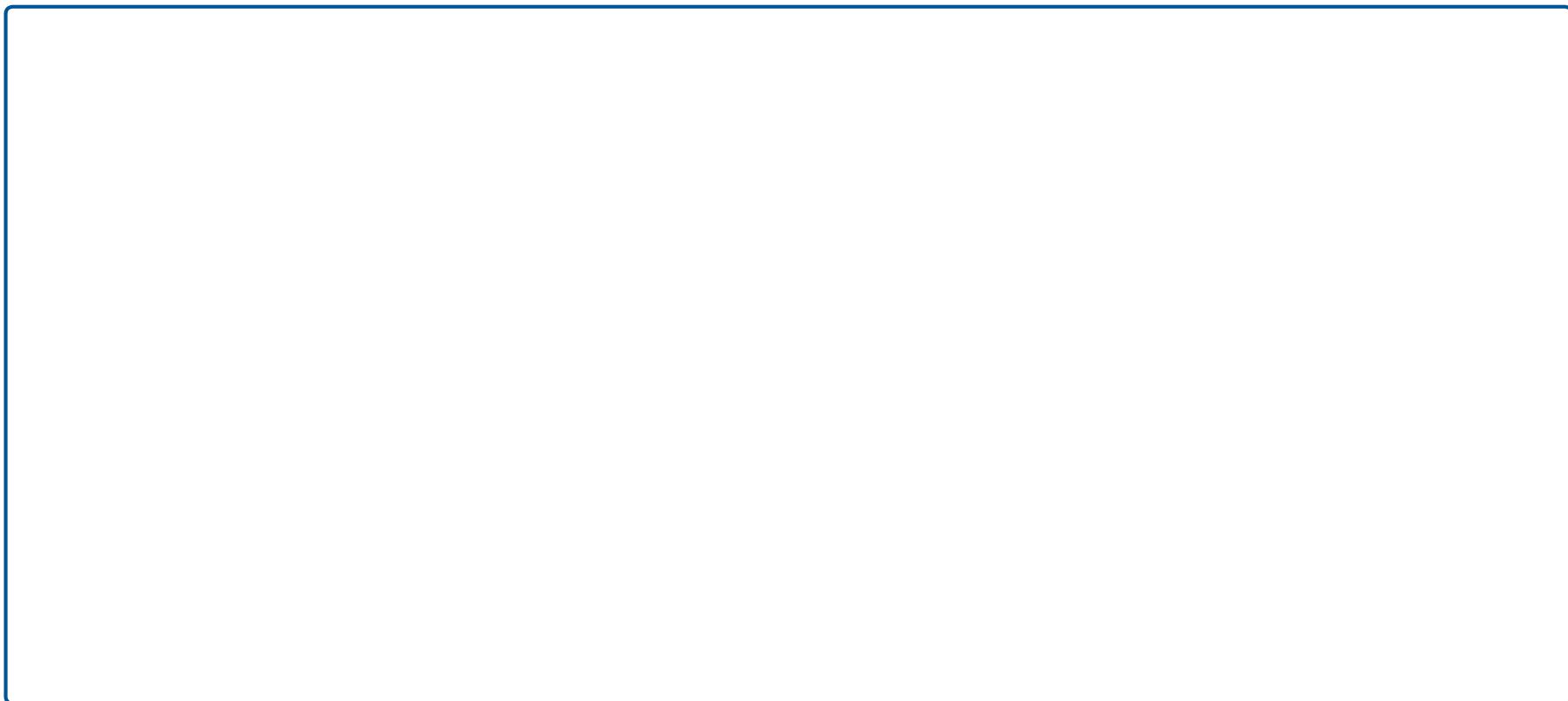
```
<a href="http://www.elte.hu" class="info"><span>ELTE</span></a>
<a href="http://www.inf.elte.hu">ELTE <span>Informatika Kar</span></a>
<a href="http://www.inf.elte.hu/mot" class="info">Médiainformatikai Tanszék</a>
```

```
delegate(document, 'click', 'a.info', handleClick);
function handleClick(event) {
    if (!event.shiftKey) {
        return;
    }
    event.preventDefault();
    console.log(this.href);
}
```

# PÉLDÁK AZ ESEMÉNYKEZELÉSRE

# CSILLAGOK

Rajzoljunk ki egy csillagot a képernyőnek azon pontjára, ahova kattintottunk!



# CSILLAGOK

```
<ul id="stars"></ul>
```

```
.star {  
  position: fixed;  
  list-style-type: none;  
}
```

```
document.addEventListener("click", onDocumentClick);  
function onDocumentClick(e) {  
  // input  
  const x = e.clientX;  
  const y = e.clientY;  
  // processing  
  const star =  
    `<li class="star" style="top:${y}px; left:${x}px;">*</li>`;  
  // output  
  document.querySelector("#stars").innerHTML += star;  
}
```

# TODO LISTA 1.

Egy listaelemre kattintva váltogassuk annak stílusosztályát!

- első
- második
- harmadik

# TODO LISTA 1.

```
<ul class="list">
  <li>first</li>
  <li>second</li>
  <li>third</li>
</ul>
```

```
.done::before {
  content: "✓ ";
}
```

```
const ul = document.querySelector("ul.list")
ul.addEventListener("click", onListClick);
function onListClick(e) {
  if (e.target.matches("li")) {
    // input
    const li = e.target;
    // output
    li.classList.toggle("done");
  }
}
```

## TODO LISTA 2.

Legyen lehetőség új elemet hozzáadni a listához!

Egy listaelemre kattintva változzassuk annak stílusosztályát!

Új elem:  Hozzáad

- első
- második
- harmadik

## TODO LISTA 2.

```
<style>
.done:before {
  content: "✓ ";
}
</style>
New todo:
<input>
<button>Add</button>
<ul>
  <li>first</li>
  <li>second</li>
  <li>third</li>
</ul>
<script>
document.querySelector("button").addEventListener("click", newTodoClick);
function newTodoClick(e) {
  // input
  const newTodo = document.querySelector('input').value;
  // output
  document.querySelector('ul').innerHTML += `<li>${newTodo}</li>`
}
```

# MEGJELENÍTŐ

Ha egy gombra kattintunk, és annak van egy speciális attribútuma, akkor az attribútumban lévő elem megjelenését változassuk!

A nyilatkozat mutatása

# MEGJELENÍTŐ

```
<button data-toggle-id="some-statement">  
    A nyilatkozat mutatása  
</button>  
<div id="some-statement" hidden>  
    Nyilatkozat  
</div>
```

```
document.addEventListener('click', onTogglerClick)  
function onTogglerClick(e) {  
    const id = e.target.dataset.toggleId;  
    if (!id) return;  
  
    const elem = document.getElementById(id);  
  
    elem.hidden = !elem.hidden;  
}
```

# ESEMÉNYEK PROGRAMOZOTT KIVÁLTÁSA

# BEÉPÍTETT ESEMÉNYEK KIVÁLTÁSA

```
const event = new MouseEvent('click', {  
    bubbles: true,  
    clientX: tx,  
    clientY: ty  
})  
elem.dispatchEvent(event)  
  
// vagy  
elem.dispatchEvent(new Event('change', {bubbles: true}))
```

- Lehetséges események
- MouseEvent konstruktor

# EGYEDI ESEMÉNYEK KIVÁLTÁSA ÉS FIGYELÉSE

```
const event = new Event('valami');
// Feliratkozás az eseményre
elem.addEventListener('valami', function (e) { /* ... */ });
// Esemény kiváltása
elem.dispatchEvent(event);

// vagy

const event = new CustomEvent('valami', { detail: "adat", bubbles: true });
elem.addEventListener('valami', eventHandler);
function eventHandler(e) {
  console.log(e.detail); // "adat"
}
```

## Referencia

# **TESZTELÉS**

**JASMINE**

# PÉLDA



Köszönj!

Hello !

3.4.0

• • •

3 specs, 0 failures, randomized with seed 61584

finished in 0.974s

Options

#### felület működése

- minden megjelenik
- szöveget írva be helyes üdvözlés lesz
- üresen hagyva Hello ! jelenik meg

# ELŐKÉSZÍTÉS

```
<!-- 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>

<input type="text" id="nev"> <br>
<input type="button" value="Köszönj!" id="gomb"> <br>
<span id="kimenet"></span>

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

# TESZTEK

```
describe('felület működése', () => {
  // saját $ függvény
  function $(sel) { return document.querySelector(sel); }
  // minden teszt előtt készítsük elő a felületet
  beforeEach(() => {
    $('#nev').value = '';
    $('#kimenet').innerHTML = '';
  });
  // tesztek
  it(`Mindenn megjelenik`, () => {
    expect($('input#nev')).not.toBeNull();
    expect($('input#gomb')).not.toBeNull();
    expect($('span#kimenet')).not.toBeNull();
  });
  it(`Üresen hagyva Hello ! jelenik meg`, () => {
    $('#gomb').click(); // elvégez
    expect($('#kimenet').textContent).toBe('Hello !'); // ellenőriz
  });
  it(`Szöveget írva be helyes üdvözlés lesz`, () => {
    $('#nev').value = 'alma'; // előkészít
    $('#gomb').dispatchEvent(new Event('click')); // elvégez
    expect($('#kimenet').textContent).toBe('Hello alma!'); // ellenőriz
  });
});
```

# **BÖNGÉSZŐ-FÜGGETLENSÉG**

**ÖRÖKÖLT KÓD MEGÉRTÉSÉHEZ**

Tulajdonság ellenőrzés

# INLINE MÓDSZER

- HTML attribútumként jelenik meg az eseménykezelő
- Történetileg az első módszer
- mindenhol működik

```
<!-- Általában -->
<elem ontípus="javascript kód">

<!-- Például -->
<input type="button" id="gomb" onclick="hello()">
```

# TRADÍCIONÁLIS MÓDSZER

- Programozottan rendelhető eseménykezelő egy elem eseményéhez
- Nem szabványos, de minden böngésző támogatja
- Egy elemhez csak egy eseménykezelő köthető

```
elem.ontípus = függvény;  
// És NEM:  
elem.ontípus = függvény();  
// Törlés  
elem.ontípus = null;  
// Például  
document.querySelector('#gomb').onclick = hello;
```

# SZABVÁNYOS MÓDSZER

- `addEventListener`, `removeEventListener`
- Paraméterek
  - Esemény típusa
  - Eseménykezelő függvény
  - Elkapás iránya

```
// Hozzáadás/regisztrálás
elem.addEventListener('típus', fuggveny);
// Elvétel/Leiratkozás
elem.removeEventListener('típus', fuggveny);

// Például
let gomb = document.querySelector('#gomb');
gomb.addEventListener('click', hello, false);
```

# ESEMÉNYKEZELŐK REGISZTRÁCIÓJA

## Tulajdonság ellenőrzés (feature detection)

```
function kezelotRegisztral(elem, tipus, kezelo) {
    //Szabványos módszer
    if (elem.addEventListener) {
        elem.addEventListener(tipus, kezelo, false);
    }
    //Microsoft módszer
    else if (elem.attachEvent) {
        elem.attachEvent('on' + tipus, function () {
            kezelo(window.event);
            //vagy
            // return kezelo.call(elem, window.event);
        });
    }
    //Tradicionális módszer
    else {
        elem['on' + tipus] = kezelo;
    }
}

// Használata
kezelotRegisztral($('#lista'), 'click', mutat);
```

# ÖSSZEFOGLALÁS

- Eseménykezelés
  - `elem.addEventListener(típus, fgv)`
  - Interakció eszköze
  - Eseményobjektum
  - Alapértelmezett műveletek megakadályozása
  - Buborékolás
  - Delegálás