WEBFEJLESZTÉS 2. – ELEMEK ELÉRÉSE, ESEMÉNYEK

Horváth Győző

Egyetemi adjunktus

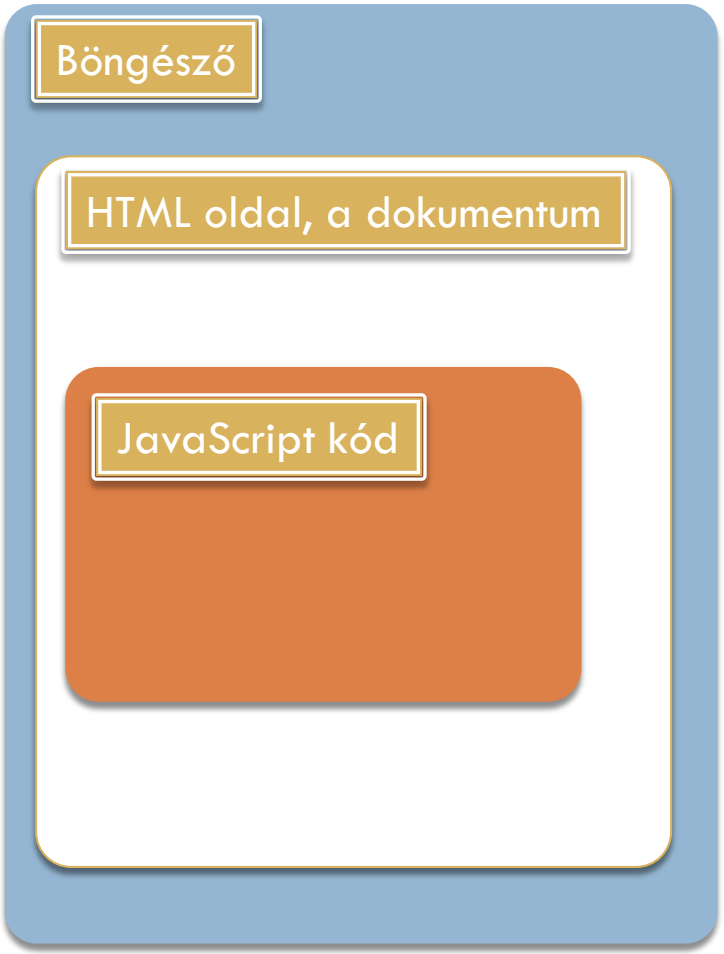
1117 Budapest,
Pázmány Péter sétány 1/C, 2.420
Tel: (1) 372-2500/1816

Ismétlés

- JavaScript nyelvi elemei
  - elemi és összetett típusok
  - vezérlési szerkezetek
  - operátorok
- Alapvető kommunikáció
  - beolvasás (prompt, confirm)
  - kiírás (console.log, alert, document.writeln)

Felépítés

- JavaScript nyelv
- JavaScript és HTML kapcsolata



Egyszerű példa: Helló név!

- Feladat
  - Bekérni a nevet, pl. "Senki bácsi"
  - Kiírni: "Hello Senki bácsi!"
- Eddigi tudásunkkal
  - prompt és alert
  - … de nem így szokták megoldani

```
var nev = prompt('Hogy hívnak?', 'Senki bácsi');
alert('Hello ' + nev + '!');
```

Egyszerű példa: Helló név!

- Űrlap
  - Gomb lenyomása
  - Érték kiolvasása
  - Eredmény megjelenítése


```
<form>
  Név: <input type="text">
  <br>
  <input type="button" value="Köszönj!">
</form>
```

Megoldás lépései

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

Kérdések a megoldáshoz

- Hogyan lehet a gomb lenyomására reagálni?
- Hogyan tudom a szöveges mező értékét kiolvasni? (valaminek a valamije)
  - JavaScriptből hogyan érem el a szöveges mezőt? (tetszőleges HTML dokumentumbeli elemre hivatkozás)
  - Ha elértem a szöveges mezőt, akkor hogyan tudom a benne lévő értéket kiolvasni? (tulajdonság)
- Hogyan tudom az eredményt nem felugró ablakkal megjeleníteni, hanem a dokumentumban?

Elemek elérése

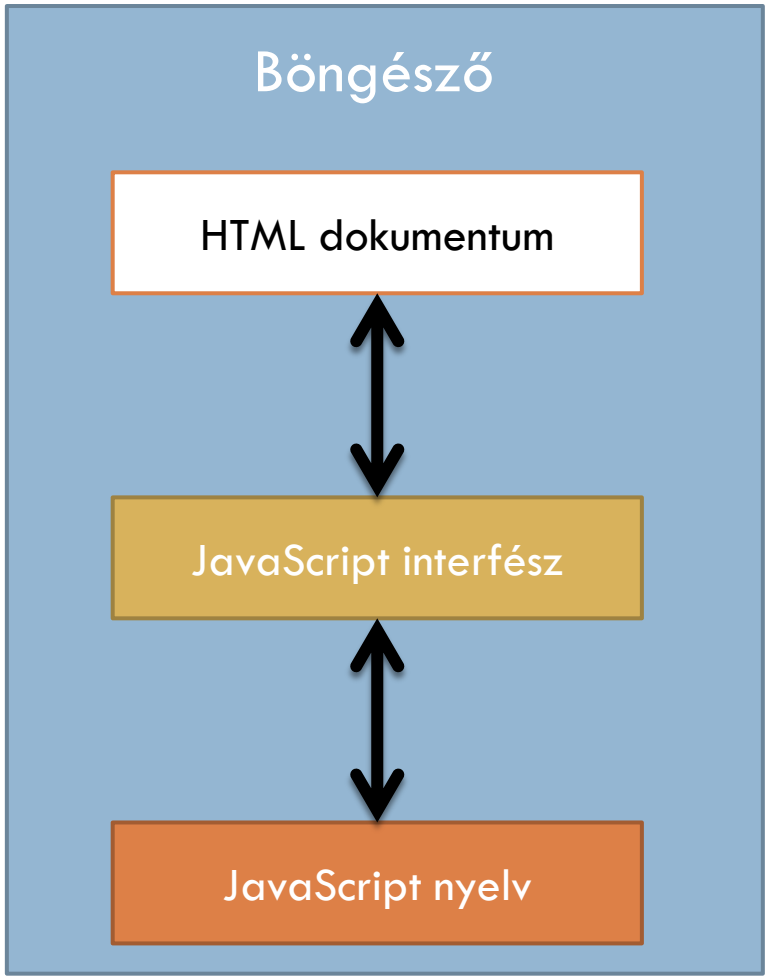
DOM

Elemek elérése

- Hogyan tud a JavaScript egy tetszőleges HTML elemet elérni?
- Ez nem a JavaScript nyelvben van definiálva.
- Ezt a JavaScriptet futtató környezet biztosítja, ami most a böngésző
- A böngésző tesz lehetővé olyan kapcsolódási pontokat (interfészeket), amelyen keresztül a HTML dokumentum JavaScripttel módosítható.

Elemek elérése

- Kétféle interfész
  - DOM
  - (BOM)



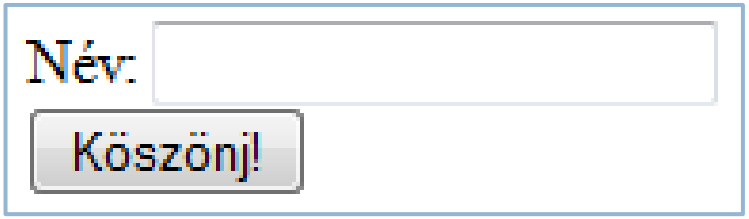
HTML mint faszerkezet


```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <form id="form1">
      Név: <input type="text" name="nev" id="nev">
      <br>
      <input type="button" value="Köszönj!" name="gomb"
id="nev">
    </form>
  </body>
</html>
```

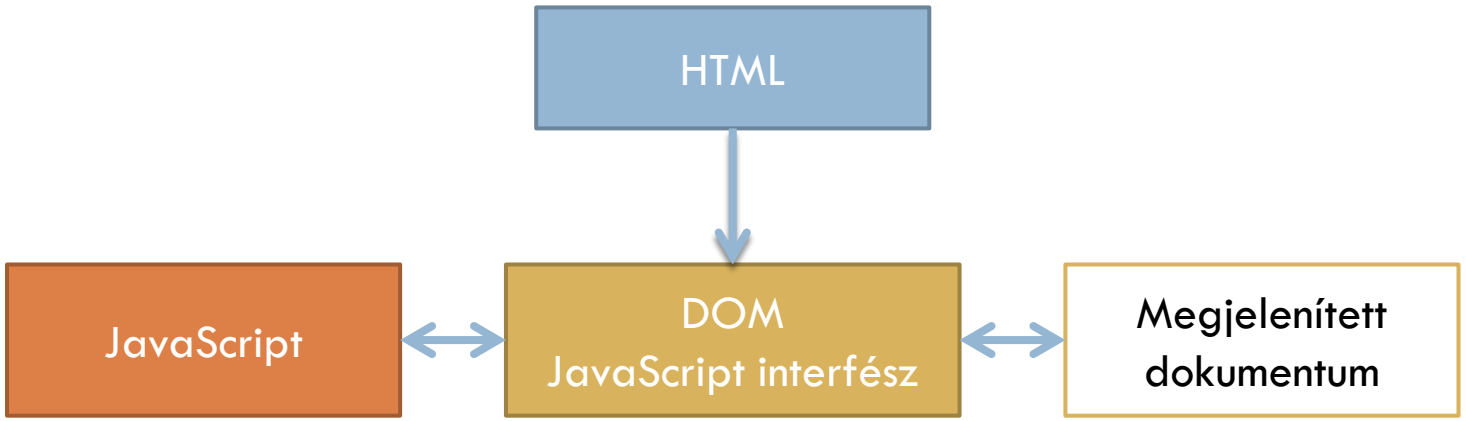
DOM


```
DOCTYPE: html
HTML lang="en"
  HEAD
    #text:
    META charset="utf-8"
    #text:
    TITLE
      #text:
  #text:
  BODY
    #text:
    FORM id="form1"
      #text: Név:
      INPUT type="text" id="nev" name="nev"
      #text:
      BR
      #text:
      INPUT type="button" id="nev" name="gomb" value="Köszönj!"
      #text:
    #text:
```

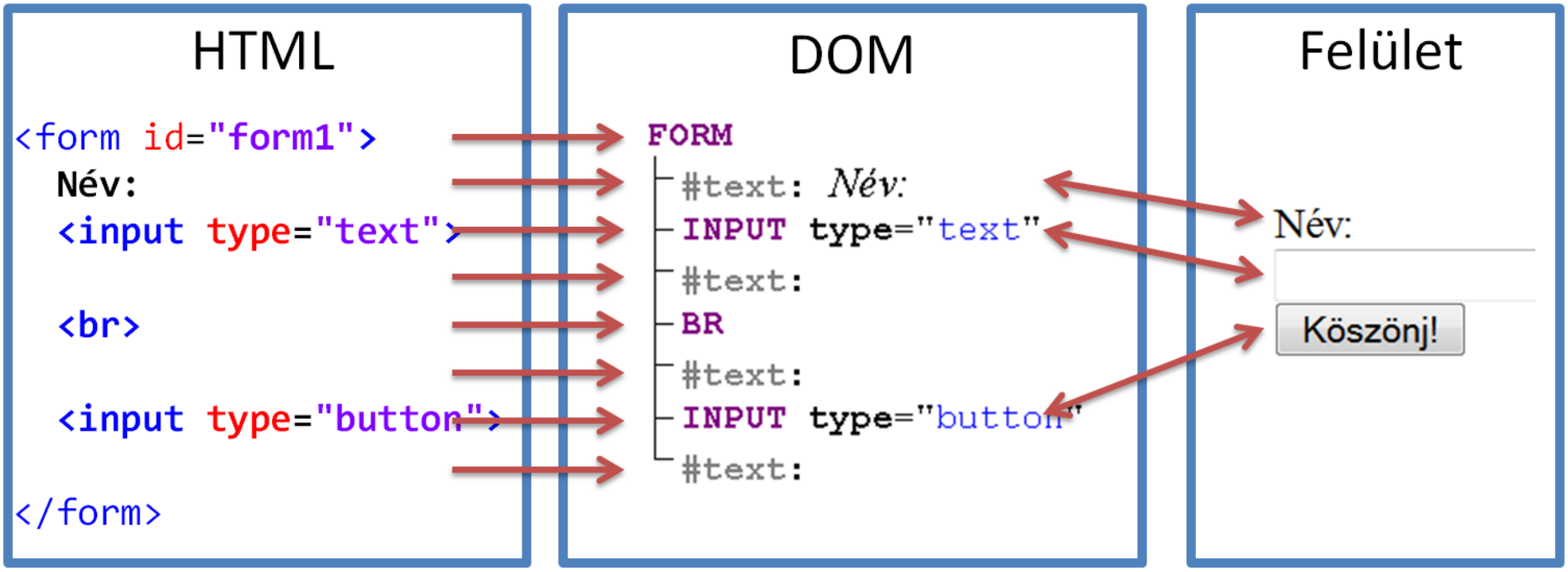
Felület



Sematikus kapcsolatrendszer



Élő kapcsolat



Document Object Model – DOM

- Dokumentum Objektum Modell
- Korszerű, minden böngésző támogatja
- Szabványos interfész fastruktúra alapú hierarchiához
  - DOM Level 1 (1998)
  - DOM Level 2 (2000)
  - DOM Level 3 (2004)
  - DOM Events Level 2 (2000)
- Tulajdonságok, metódusok és események
- HTML, XML

DOM műveletek

- Rengeteg művelet (elemek, attribútumok)
  - bejárás
  - létrehozás
  - módosítás
  - törlés
- DOM gyökere: document
- Elem elérése azonosító (id) alapján történik

```
document.getElementById('id')
```

Elemek elérése – összefoglalás

- BÁRMELYIK elem elérhető JavaScriptből
- id-t kell adni neki
- document.getElementById(id)

- Példa:

```
Név: <input type="text" id="nev">
```

```
document.getElementById('nev')

//vagy változóba kimentve
var nevObj = document.getElementById('nev');
```

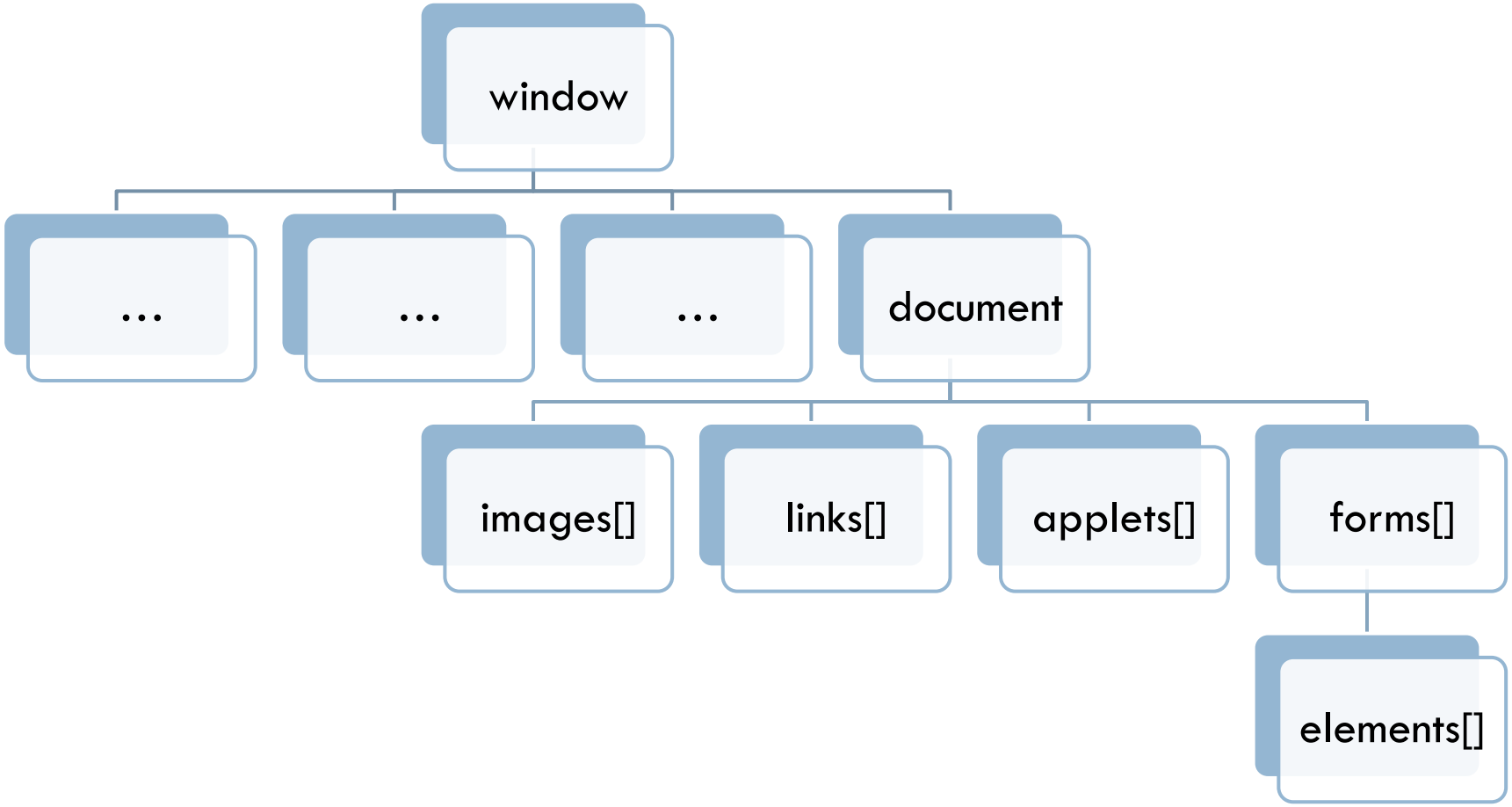
Elemek elérése

BOM

Browser Object Model – BOM

- Nem szabványos, de minden böngésző támogatja
- Bővebben később lesz szó róla
- Elemek elérésének módja elavult benne.
- Limitált hozzáférés a dokumentum elemeihez
- Régi megközelítés
  - Mind a mai napig sok ilyen kóddal lehet találkozni
  - Ne használjuk!
- A name attribútum alapján működik

BOM



BOM

- Tömbök indexelése
  - számokkal
  - name attribútummal

```
<form name="form1">
  Név: <input type="text" name="nev">
  <br>
  <input type="button" value="Köszönj!" name="gomb">
</form>
```

```
window.document.forms[0].elements[0]
window.document.forms['form1'].elements['nev']
document.forms['form1'].elements['nev']
document.form1.nev
```

Tulajdonságok

Analóg, felfedező, referencia alapú módszer

Mit tud egy DOM objektum?

- A szabvány meghatározza, hogy egy DOM objektum mit tud
- Objektum
  - tulajdonságok
  - metódusok
- Minket főleg a tulajdonságok érdekelnek

Szabványos módszer

- DOM HTMLElement
- http://www.w3.org/TR/DOM-Level-2-HTML/html.html
- https://developer.mozilla.org/en-US/docs/Gecko_DOM_Reference
- https://developer.mozilla.org/en-US/docs/Gecko_DOM_Reference#HTML_interfaces

Analóg módszer

- Elnevezési konvenció: általában minden DOM objektumnak olyan tulajdonságai vannak, amilyen attribútumai a neki megfelelő HTML elemnek.
- Átírási szabály: camel case
  - ezEgyÖsszetettSzó
  - elsőKicsiÖsszetételnélNagyBetűk
  - pl. getElementById

Analóg módszer – input mező

HTML attribútum	JavaScript tulajdonság
id	id
name	name
type	type
value	value
readonly	readOnly
maxlength	maxLength

Felfedező módszer

- JavaScript konzolban vizsgáljuk meg az elemet
- DOM fül
- Böngésző specifikus tulajdonságok is megjelenhetnek → óvatosan ezzel a módszerrel

Felfedező módszer

Név:

alma

Köszöni!

Konzol HTML CSS Szkript DOM Net Sütik

window > input#nev

readOnly	false
required	false
scrollHeight	19
scrollLeft	0
scrollTop	0
scrollWidth	150
selectionEnd	4
selectionStart	4
size	20
spellcheck	false
src	""
style	CSS2Properties { length=0, MozAppearance="", MozOutlineRadius="", több... }
tabIndex	0
tagName	"INPUT"
textContent	""
textLength	4
title	""
type	"text"
useMap	""
value	"alma"

Példa: input mező értéke

- HTML

```
<input type="text" name="nev" id="nev">
```

- JavaScript

```
document.getElementById('nev').value
```

- Például:

```
var nev = document.getElementById('nev').value;
```

- Írható olvasható tulajdonság

```
document.getElementById('nev').value = 'Győző';
```

Eseménykezelés

Inline, tradícionális, szabványos, IE specifikus módszer

Esemény és eseménykezelő

- Akció-reakció elve
- Akció – kiváltó ok
  - Környezet (böngésző) eleme (pl. oldalbetöltődés)
  - Felhasználó cselekvése (pl. kattintás, egérmozgatás)
- Reakció = **események**
  - HTML elemek jelzései (gomb: kattintás történt)
- Ezekre az eseményekre lehet feliratkozni **eseménykezelő** függvényekkel

Események típusai

- Egéresemények
  - click
  - dblclick
  - mouseup
  - mousedown
  - mouseover
  - mousemove
  - mouseout

Események típusai

- Billentyűzetesemények
  - keydown
  - keyup
  - keypress
- Oldal események
  - load (window, frame, object, image)
  - unload (window, frame)
  - abort (image, object)
  - error (object, image, frame)
  - resize (document)
  - scroll (document)

Események típusai

- Űrlap és űrlapelem események
  - submit (form)
  - reset (form)
  - select (input, textarea)
  - change (input, select, textarea)
  - focus (input, select, textarea, label, button)
  - blur (input, select, textarea, label, button)

Események típusai

- Érintés események
  - touchstart
  - touchend
  - touchmove
  - touchenter
  - touchleave
  - touchcancel

Eseménykezelő regisztrálása

- Inline módszer
- Tradícionális
- Szabványos
- IE specifikus

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

```
elem.ontípus = fuggveny;
```

- És NEM:

```
elem.ontípus = fuggveny();
```

- Törlés:

```
elem.ontípus = null;
```

- Például:

```
document.getElementById('gomb').onclick = hello;
```

Tradícionális módszer

- Előnye
  - Minden böngésző ismeri
- Hátránya
  - Nem szabványos
  - Egy elemhez csak egy eseménykezelő köthető

Szabványos módszer

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

```
elem.addEventListener('típus', fuggveny, false);

elem.removeEventListener('típus', fuggveny, false);
```

- Például

```
var gomb = document.getElementById('gomb');
gomb.addEventListener('click', hello, false);
```

Microsoft módszer

- Hasonló a szabványhoz, de kevesebbet tud
- attachEvent, detachEvent

```
elem.attachEvent('ontípus', fuggveny);

elem.detachEvent('ontípus', fuggveny);
```

Példa: gombra kattintás


```
<form id="form1">
  Név: <input type="text" id="nev">
  <br>
  <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
</form>

<script type="text/javascript">
function hello() {
  //...
}
</script>
```

Kiírás a dokumentumba

innerHTML

Kiírás a dokumentumba

- console.log nem látszik
- alert() csúnya
- document.writeln() lezárt dokumentumra nem működik: újat nyit
  - a dokumentum betöltése során szabad csak használni
- ÚJ: innerHTML
  - az elemeken belüli HTML tartalom
  - írható, olvasható

```
elem.innerHTML = 'Tetszőleges HTML szöveg'; //írás
console.log(elem.innerHTML); //olvasás
```

Példa – innerHTML


```
<form id="form1">
  Név: <input type="text" id="nev">
  <br>
  <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
  <br>
  <span id="kimenet"></span>
</form>
```

```
document.getElementById('kimenet').innerHTML = 'Hello világ!';
```

Példafeladat megoldása

Megoldás


```
<form id="form1">
  Név: <input type="text" id="nev“> <br>
  <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
  <br>
  <span id="kimenet"></span>
</form>
<script type="text/javascript">
function nevbolUdvozles(nev) {
  return 'Hello ' + nev + '!';
}
function hello() {
  //Beolvasás
  var nev = document.getElementById('nev').value;
  //Feldolgozás
  var udvozles = nevbolUdvozles(nev);
  //Kiírás
  document.getElementById('kimenet').innerHTML = udvozles;
}
</script>
```

Finomítások

$() függvény, HTML és JavaScript szétválasztása

$() függvény bevezetése

- document.getElementById()
  - hosszú, olvashatatlanná teszi a kódot
  - könnyen elgépelhető
- $() függvény
  - document.getElementById() függvényt csomagolja be
  - rövidebb forma

```
function $(id) {
  return document.getElementById(id);
}
```

```
document.getElementById(id)  --->  $(id)
```

Példa


```
<form id="form1">
  Név: <input type="text" id="nev">
  <br>
  <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
  <br>
  <span id="kimenet"></span>
</form>
<script type="text/javascript">
function $(id) {
  return document.getElementById(id);
}
function nevbolUdvozles(nev) {
  return 'Hello ' + nev + '!';
}
function hello() {
  var nev = $('nev').value;
  var udvozles = nevbolUdvozles(nev);
  $('kimenet').innerHTML = udvozles;
}
</script>
```

Szétválasztás

- HTML – szerkezet
- CSS – megjelenés
- JavaScript – viselkedés

- HTML és CSS szétválasztása
- HTML és JavaScript szétválasztása
  - JavaScript kódot külön állományba tenni
  - HTML kódban csak hivatkozás maradhat külső állományra

Szétválasztás – 1. lépés

- Függvénydefiníciók külön állományba

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title></title>
    <script type="text/javascript" src="hello.js"></script>
  </head>
  <body>
    <form id="form1">
      Név: <input type="text" id="nev">
      <br>
      <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
      <br>
      <span id="kimenet"></span>
    </form>
  </body>
</html>
```

Szétválasztás – 2. lépés

- Inline eseménykezelők megszüntetése
  - Programozott hozzárendelés – szabványos modell
  - A head-ben az elemek még nem elérhetők
  - Az oldal betöltése után kell hozzárendelni az eseménykezelőket
  - window objektum load eseménykezelő függvényben

Szétválasztás – HTML


```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title></title>
    <script type="text/javascript" src="hello.js"></script>
  </head>
  <body>
    <form id="form1">
      Név: <input type="text" id="nev">
      <br>
      <input type="button" value="Köszönj!" id="gomb">
      <br>
      <span id="kimenet"></span>
    </form>
  </body>
</html>
```

Szétválasztás – JavaScript

- <head>-ben → még nem működik

```
//Függvénydefiníciók
function $(id) {
        return document.getElementById(id);
}
function nevbolUdvozles(nev) {
        return 'Hello ' + nev + '!';
}
function hello() {
        var nev = $('nev').value;
        var udvozles = nevbolUdvozles(nev);
        $('kimenet').innerHTML = udvozles;
}
//Eseménykezelők regisztrálása
$('gomb').addEventListener('click', hello, false);
```

Szétválasztás – JavaScript


```
//Segédfüggvények
function $(id) {
	return document.getElementById(id);
}
//Feldolgozó függvények
function nevbolUdvozles(nev) {
	return 'Hello ' + nev + '!';
}
//Eseménykezelő függvények
function hello() {
	var nev = $('nev').value;
	var udvozles = nevbolUdvozles(nev);
	$('kimenet').innerHTML = udvozles;
}
function init() {
	//Eseménykezelők regisztrálása
	$('gomb').addEventListener('click', hello, false);
}
window.addEventListener('load', init, false);
```

Szétválasztás

- Ez az elvárás innentől!
- Külön HTML és JavaScript állomány
- $ és init függvény mindig kell!

Feladatmegoldás lépései

1. Tervezés
   - Specifikáció
   - Felületterv
2. JavaScript adatszerkezetek kialakítása
3. Feldolgozó függvények (felületfüggetlen)
4. HTML szükségletek kialakítása
5. Eseménykezelők kialakítása
   - Melyik elem melyik eseményére kell mit reagálni