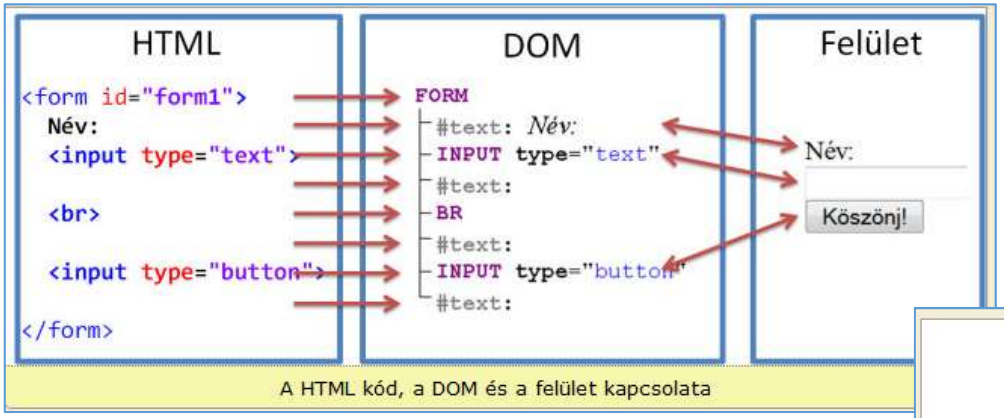
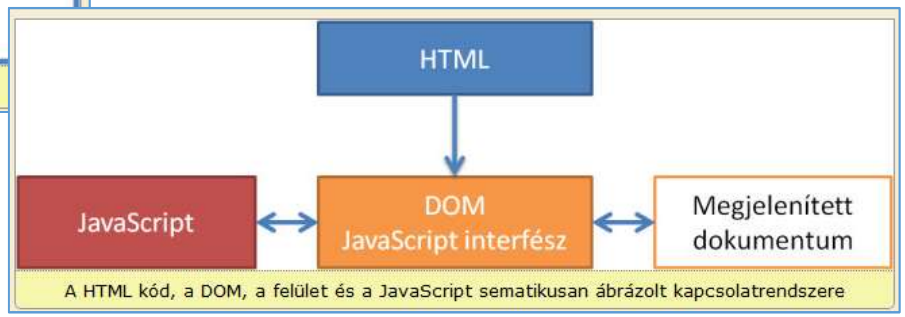
# Dokumentum Objektum Modell (DOM)

A HTML elemekhez való hozzáférés JavaScriptben egy szabványos interfészen, a *Dokumentum Objektum Modellen* (röviden ***DOM***) keresztül történik.

- A HTML dokumentum egy fastruktúra, ahol HTML elemek egymás mellett vagy egymásba ágyazva jelennek meg. Minden elemnek legfeljebb egy szülője van, aki tartalmazza őt, és minden elemnek lehetnek gyerekei, akiket az ő nyitó- és záróeleme tartalmaz. Az elemmel egy szinten lévő más elemeket pedig testvéreknek nevezzük.
- A szöveges HTML állomány betöltésekor a böngésző minden egyes HTML elemhez (még a szövegesekhez is) létrehoz egy JavaScript objektumot. Ezeket az objektumokat ugyanolyan fastruktúrába rendezi, mint ahogy az az eredeti HTML elemeknél volt. A folyamat eredményeképpen kialakul egy objektumhierarchia, ez a DOM.
- A böngésző az oldal megjelenítését a DOM alapján végzi el. A DOM és a kirajzolt oldal között nagyon szoros kapcsolat van. Ha változik a DOM, akkor az az oldalon azonnal érvényesül. Ugyanez visszafele is igaz: ha valamit változtatunk az oldalon, akkor a változások rögtön megjelennek a DOM-ban is.



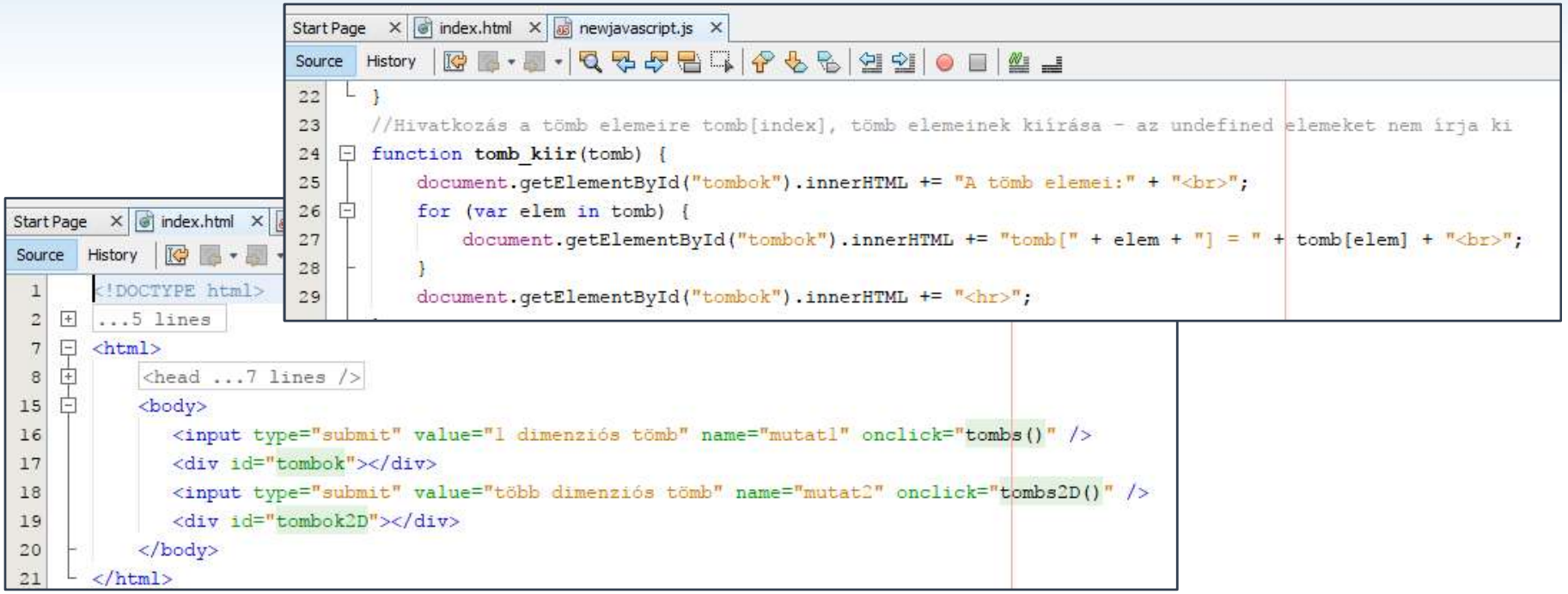
A HTML kód, a DOM és a felület kapcsolata



A HTML kód, a DOM, a felület és a JavaScript sematikusan ábrázolt kapcsolatrendszere

# Dokumentum Objektum Modell (DOM)

- **Hivatkozás a HTML elemekre**
  - **DOM-művelettel:**
    - a DOM segítségével egy adott HTML-beli elemet úgy lehet elérni, hogy a megfelelő elemet egyedi azonosítóval látjuk el az ***id*** *attribútumán* keresztül, és a létrejövő objektumhierarchiában megkeressük az adott azonosítójú elemet.
    - A keresést az egész DOM gyökérelemén, a ***document*** objektumon kell kezdeményeznünk a *getElementById()* paranccsal, aminek paraméterének a keresendő elem azonosítóját kell megadni, a rá való hivatkozást pedig a függvény visszatérési értéke adja meg

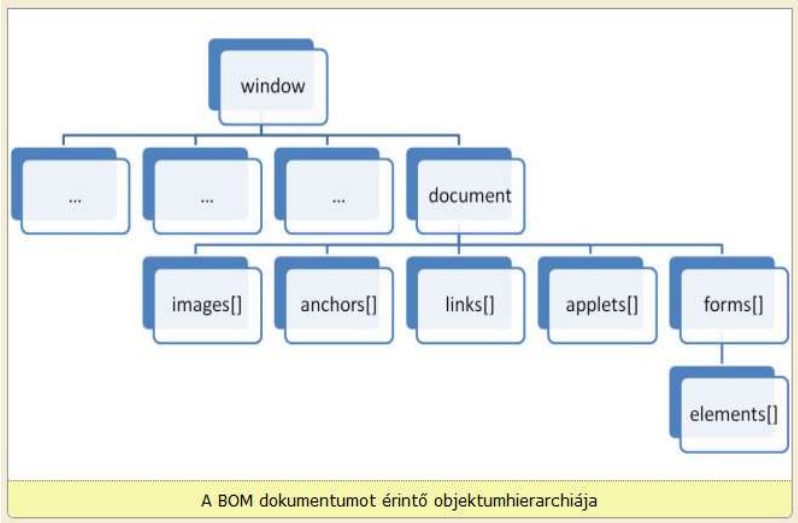


# Dokumentum Objektum Modell (DOM)

- **Hivatkozás a HTML elemekre**

  **B. BOM-művelettel:**

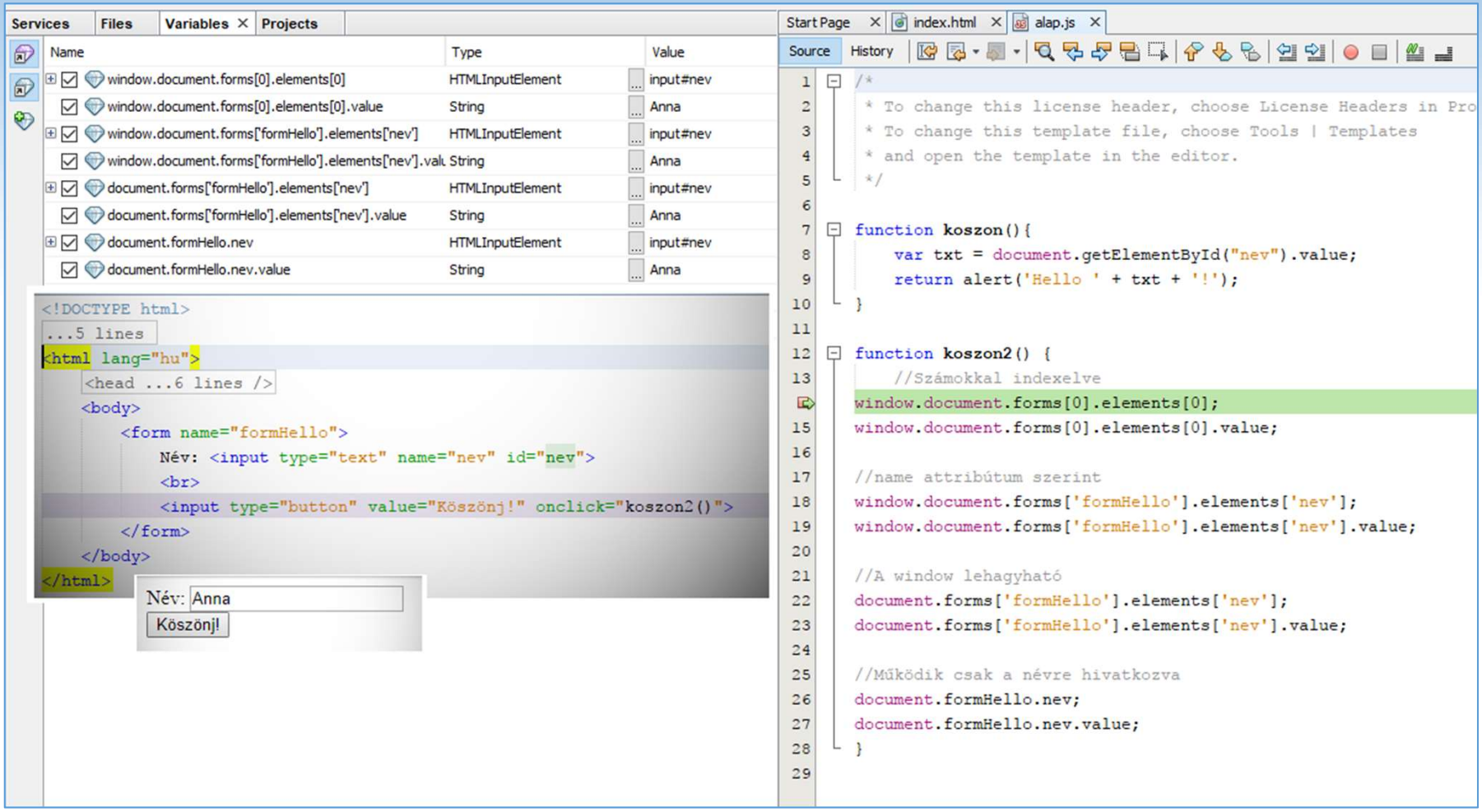
  - A DOM megszületéséig és szabványossá válásáig a másik alternatíva az elemek elérésére a *Böngésző Objektum Modell (BOM)* volt.
  - A BOM nem szabványos, de szinte minden böngésző által egyformán támogatott objektumhierarchia, amelyen keresztül a böngésző egyes elemeihez lehet programozottan hozzáférni. A BOM a dokumentum elemeihez is korlátozott hozzáférést ad olyan értelemben, hogy nem minden elem érhető el, csak azok, amelyek az interfészen megjelennek.
  - A hierarchia csúcsán a *window* objektum áll. Alatta számos egyéb mellett megjelenik a HTML dokumentumnak megfelelő *document* objektum (ez egyébként maga a DOM).
  - A *document-en* keresztül számos gyűjtemény érhető el: képek az images, hivatkozások a links, űrlapok a forms tömbön keresztül. Az űrlapelemek pedig a forms megfelelő eleme alatt vannak az elements tömbben összegyűjtve.
  - Az egyes tömböket vagy számokkal indexelhetjük (ha tudjuk, hányadik az oldalon belül), vagy az elemek name attribútumával.



A BOM dokumentumot érintő objektumhierarchiája

# Dokumentum Objektum Modell (DOM)

- **Hivatkozás a HTML elemekre**
  - **BOM-művelettel:**



# Dokumentum Objektum Modell (DOM)

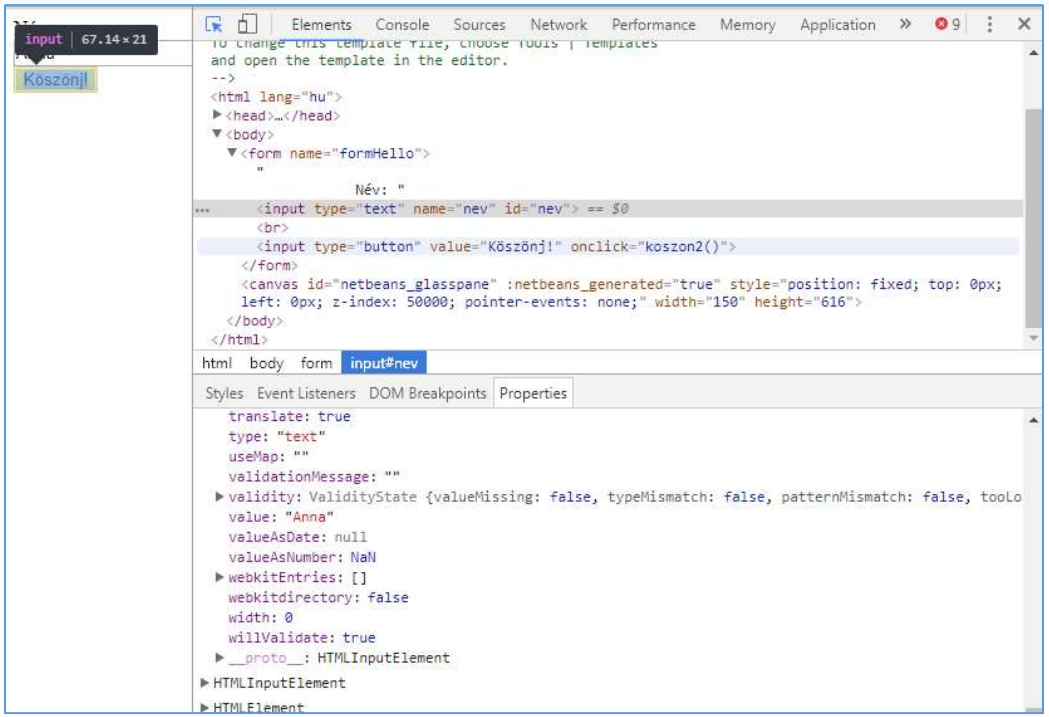
- **Elemek tulajdonságai**

A HTML elemeknek megfelelő DOM objektumok a böngésző által létrehozott JavaScript objektumok. Mint minden JavaScript objektumnak, ezeknek is vannak adattagjai és metódusai, amelyeket a DOM szabvány határoz meg. Ezekről többféleképpen szerezhetünk tudomást.

- a HTML DOM leírásból - a Mozilla gondozásában is találhatunk referenciát a HTML elemek DOM interfészéről.
- Az egyes adattagok elnevezésekor azzal elnevezési konvencióval éltek, hogy a HTML attribútumoknak megfelelő DOM tulajdonságokat vezettek be egy-egy HTML elemnél. Az átírásnál a „camel case” használták, ennek az a lényege, hogy minden adattag és metódus kis betűvel kezdődik, de összetett szavaknál a második szótól kezdve nagy betűvel kezdődnek a szavak.

1. A JavaScript konzolon keresztül. (felfedező módszer)

| HTML attribútum | DOM tulajdonság |
|---|---|
| id | id |
| name | name |
| type | type |
| value | value |
| readonly | readOnly |
| maxlength | maxLength |



# Dokumentum Objektum Modell (DOM)

- **Események és eseménykezelők**
  - Egy weboldalon rengeteg történés, kiváltó ok lehet, amire reagálhatunk.
    Pl.: a felhasználó kattint, mozgatja az egeret, lenyom egy billentyűt vagy a böngésző betölt egy oldalt, vagy egy képet, megváltozik a böngészőablak mérete, … stb.

    A HTML dokumentum elemei, mögött álló JavaScript objektumok érzékelik ezeket a történéseket, és jelzik ezek bekövetkezését. Ezeket a jelzéseket hívjuk **esemény**nek.
  - A bekövetkező eseményekre lehet reagálni JavaScriptből az ún. **eseménykezelő** függvényekkel.
    Ezek a függvények csak az események bekövetkeztekor futnak le, és programozottan kezelik az események történéseit. Az egyes eseményekhez hozzá kell rendelni az eseménykezelő függvényeket. Az *eseményekre feliratkoznak az eseménykezelő függvényre* vagy az *eseménykezelőket regisztráljuk az eseményekhez.*

- **Események típusai**

**Egéresemények**

*click*: kattintás
*dblclick*: duplakattintás
*mousedown*: egér gombjának lenyomása
*mouseup*: egér gombjának felengedése
*mouseover*: az egérkurzor belépése egy elem fölé
*mousemove*: az egérkurzor mozog egy elem fölött
*mouseout*: az egérkurzor elhagyja az elemet

**Billentyűzetesemények**

*keydown*: billentyű lenyomása
*keypress*: billentyű megnyomása (pl. hosszan nyomva tartásnál többször meghívódik)
*keyup*: billentyű felengedése

**Érintés események**

*touchstart, touchend*
*touchmove, touchcancel*

**Oldal események**

*load (window, frame, object, image*): az adott objektum betöltődött
*unload (window, frame):* az adott objektum bezárul (lecserélődik)
*abort (image, object):* a betöltés megszakadt
error (object, image, frame): hiba a betöltés során
resize (document): a dokumentum mérete megváltozik
scroll (document): a dokumentumban görgetnek

**Űrlap és űrlapelem események**

submit (form): az űrlap elküldésre kerül
reset (form): az űrlap alaphelyzetbe áll
select (input, textarea): szöveg kijelölése
change (input, select, textarea): az űrlapmező értékének megváltozása
focus (input, select, textarea, label, button): az űrlapelem megkapta fókuszt (nem csak űrlapelemeknél használható, hanem szinte az összes elemnél, pl. hivatkozásoknál)
blur (input, select, textarea, label, button): a fókusz elhagyása (ld. az előző megjegyzést

# Dokumentum Objektum Modell (DOM)

- **Eseménykezelők regisztrálása**
- A web fejlődése során több módszer is kialakult az eseménykezelők regisztrálására. Ezeket nézzük végig, mindegyiknél megemlítve az előnyeit és hátrányait, valamint azt, hogy ajánlott-e a használatuk.

- **Inline módszer:** HTML attribútumként jelenik meg az eseménykezelő, annál az elemnél kell felvenni, amelyen az esemény bekövetkezik.

```
<elem ontípus="JavaScript kód">        <-- -->        <input type="button" onclick="hello()">
```

ez a megoldás minden böngészőben működik, de mivel a HTML kódban jelenik meg, ezért nincs lehetőség eseménykezelők programozott beállítására

**Tradicionális módszer:** programkódból rendelhetünk egy eseménykezelő függvényt egy eseményhez, ekkor eseménykezelő függvényt a kiválasztott elem megfelelő adattagjához kell rendelni

```
elem.ontípus = függvény;
```

Az adattag neve kötelezően végig kisbetű, *on+típus* felépítésű. ***Az értékadás jobb oldalán függvényreferenciának, és nem függvényhívásnak kell szerepelnie !!!***

```
HTML:   <input type="button" id="gomb">
JS:     document.getElementById('gomb').onclick = hello;
```

ez a megoldás minden böngészőben működik, de egy elem egy eseményéhez csak egy függvény regisztrálható.

# Dokumentum Objektum Modell (DOM)

- **Eseménykezelők regisztrálása**
- **Szabványos módszer:**
  A szabvány két függvényt vezetett be minden DOM objektumhoz:
  - az **addEventListener** függvénnyel feliratkozhatunk egy elem adott típusú eseményére,
  - a **removeEventListener** függvény pedig megszünteti a hozzárendelést.

  Mindkét függvénynek három paramétere van:
  1. *az esemény típusa*, szövegként megadva;
  2. *az eseménykezelő függvény referenciája*;

```
//Feliratkozás
elem.addEventListener('típus', függvény, false);

//Leiratkozás
elem.removeEventListener('típus', függvény, false);
```

-->

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

//vagy egyben, de így nagyon hosszú
document.getElementById('gomb').addEventListener('click', hello, false);
```

egy elem adott eseményére akárhány függvény feliratkozhat és az esemény bekövetkez-tekor a böngésző mindegyik függvényt egymás után meghívja valamilyen sorrendben.

**Microsoft módszer:** A Microsoft saját eseménykezelő-regisztrálási módszert dolgozott ki, amely a szabványhoz hasonlóan ugyancsak több függvény regisztrálását teszi lehetővé, de nem támogatja az elkapás irányának meghatározását, így valamennyivel kevesebb annál. Itt is egy-egy függvény *attachEvent* és *detachEvent*. (régebbi Internet Explorerek esetén)

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

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

# Dokumentum Objektum Modell (DOM)

- **Kiírás a dokumentumba**

1. *console.log* : a JavaScript konzolra ír, az átlagos felhasználónak nem is látszik, fejlesztéskor használjuk
2. *alert()* csúnya és felugró ablakkal zavarja meg az oldal használatának menetét
3. *document.writeln()* lezárt dokumentumra nem működik: újat nyit a dokumentum betöltése során szabad csak használni
4. *innerHTML*
   - az elemeken belüli HTML tartalom
   - írható, olvasható

```
var p = document.querySelector('p');

// Olvasás
console.log(p.innerHTML);

// Írás
p.innerHTML = 'Tetszőleges <i>HTML</i> szöveg';
```

```
<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!';
```

# Dokumentum Objektum Modell (DOM)

- **Finomítások:**
  - *A $() függvény:*

  Kliensoldali szkriptekben gyakran fordul elő a *document.getElementById()* vagy a *document.querySelector()* függvény használata.

  - a hosszúsága miatt olvashatatlanná teszi a kódot
  - Könnyen elgépelhető

  Az olvasható kód és a hibamentesség érdekében bevezetünk egy ***$()*** nevű függvényt, amely ugyanazt adja vissza, mint fenti függvények.Ezt csomagoló függvénynek is nevezzük.

- *$$() függvény*

  a *document.getElementById()* vagy a *document.querySelector()* függvények csomagoló függvénye

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

function $$(tag) {
    return document.getElementsByTagName(tag);
}
```

```
function $(selector) {
    return document.querySelector(selector);
}

function $$(selector) {
    return document.querySelectorAll(selector);
}
```

# Dokumentum Objektum Modell (DOM)

```
<!DOCTYPE html>
...5 lines
<html lang="hu">
    <head ...6 lines />
    <body>
        <form>
            Név: <input type="text" id="nev">
            <br>
            <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
            <br>
            <span id="kimenet"></span>
        </form>
        <script>
            function $(selector) {
                return document.querySelector(selector);
            }
            function nevbolUdvozles(nev) {
                return `Hello ${nev}!`;
            }
            function hello() {
                var nev = $('#nev').value;
                var udvozles = nevbolUdvozles(nev);
                $('#kimenet').innerHTML = udvozles;
            }
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
...5 lines
<html lang="hu">
    <head ...6 lines />
    <body>
        <form>
            Név: <input type="text" id="nev">
            <br>
            <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
            <br>
            <span id="kimenet"></span>
        </form>
        <script>
            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>
    </body>
</html>
```

# Dokumentum Objektum Modell (DOM)

- **Szerkezet és viselkedés szétválasztása:**
  - Ahogy a HTML-t és a CSS-t állományszinten is elválasztottuk egymástól
  - A JavaScriptet is el kell választanunk a HTML-től.
    - JavaScript kódot külön állományba tenni
    - HTML kódban csak hivatkozás maradhat külső állományra

Így az egyes állományok csak egyféle funkciójú kódért lesznek felelősek.

**HTML:** az oldal szemantikai szerkezetéért

**CSS:** az oldal megjelenéséért

**JavaScript:** az oldal viselkedéséért

# Dokumentum Objektum Modell (DOM)

- **Szerkezet és viselkedés szétválasztása:**
  - Lépések:
    1. Lépés: Függvénydefiníciók külön állományba helyezése
    2. Lépés: szétválasztás –
       1. Inline eseménykezelők megszüntetése
          1. Programozott hozzárendelés – szabványos modell használata
       2. Szkript a **<head>**-ben az eseménykezelő regisztrálásakor - az elemek még nem elérhetők
          1. Az oldal betöltése után kell hozzárendelni az eseménykezelőket a </body> elé
          2. *window* objektum *load* eseménykezelő függvényben oldjuk meg az inicializálást
       3. Szkript a **</body>** előtt közvetlenül
          1. minden elem elérhető
          2. nincs további teendő
       4. Különböző funkciójú kódrészletek a JavaScript állományon belül
          1. segédfüggvények (pl. $),
          2. eseménykezelő függvények (pl. hello és init), valamint
          3. feldolgozó függvények (pl. nevbolUdvozles).

# Dokumentum Objektum Modell (DOM)

```html
<!doctype html>
<html lang="hu">
    <head>
        <meta charset="utf-8">
        <title>Üdvözlés</title>
        <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>
    </head>
    <body>
        <form>
            Név: <input type="text" id="nev">
            <br>
            <input type="button" value="Köszönj!" id="gomb" onclick="hello()">
            <p id="kimenet"></p>
        </form>
    </body>
</html>
```

1.

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

```js
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;
}
```

2.1

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

```js
//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);
```

2.2/3

```js
//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;
}
function init() {
    //Eseménykezelők regisztrálása
    $('gomb').addEventListener('click', hello, false);
}
window.addEventListener('load', init, false);
```

3.

```js
//Segédfüggvények
function $(id) { /*...*/ }

//Feldolgozó függvények
function nevbolUdvozles(nev) { /*...*/ }

//Eseménykezelő függvények
function hello() { /*...*/ }
function init() { /*...*/ }
window.addEventListener('load', init, false);
```