WEBFEJLESZTÉS 2. – ESEMÉNYEK, NYELVI ELEMEK, BEÉPÍTETT OBJEKTUMOK

Horváth Győző

Egyetemi adjunktus

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

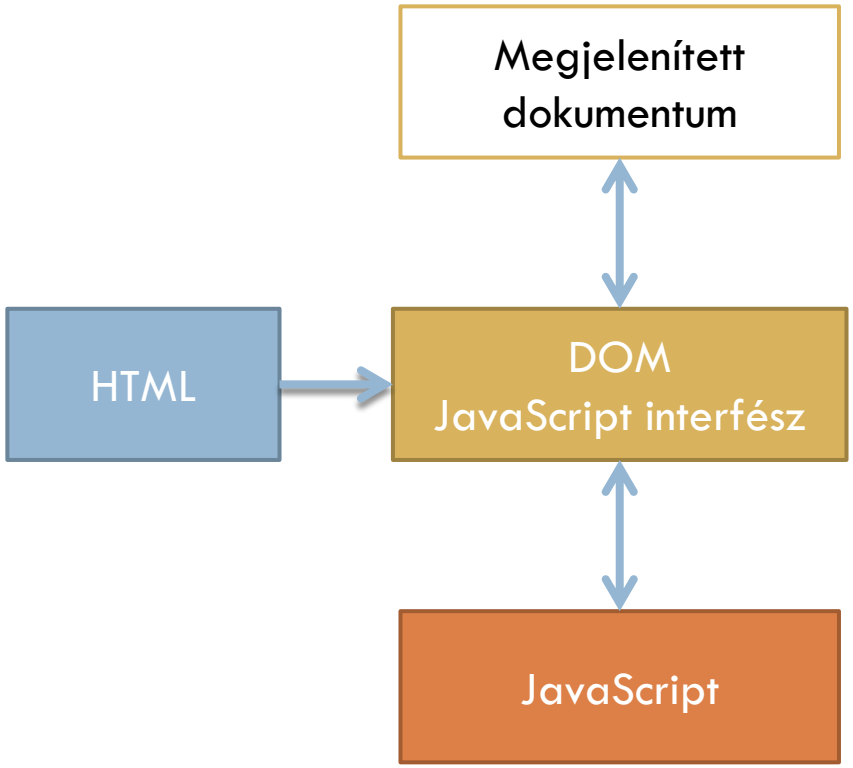
Ismétlés

Ismétlés – nyelvi elemek

- Gyengén típusos
- Interpretált nyelv
- Szintaxis C++-hoz hasonló
  - vezérlési szerkezetek
  - operátorok
- Adatszerkezetek
  - elemi típusok
  - összetett típusok (JSON)

Ismétlés

- HTML elem → DOM objektum
- DOM objektum <=> Megjelenített elem
- DOM objektum – JavaScript objektum



Ismétlés

- Elemek elérése
  - document.getElementById(id)
  - rövidítve: pl. $() függvény
- Elem (JavaScript objektum) tulajdonságai
  - Analóg módszer: Webfejlesztés 1. → Webfejlesztés 2.
  - Felfedező
  - Dokumentáció alapú (szabvány)
- Eseménykezelés
  - Eseménytípusok
  - Eseménykezelő modellek

Ismétlés

- HTML és JavaScript szétválasztása
  - HTML: <script src=""></script>
  - JavaScript: $(), window load, init()
- Feladatmegoldás lépései
  - HTML szükséglet
  - JavaScript adatszerkezet
  - Eseménykezelők hozzárendelése és logikája

Eseménykezelés

Eddig…

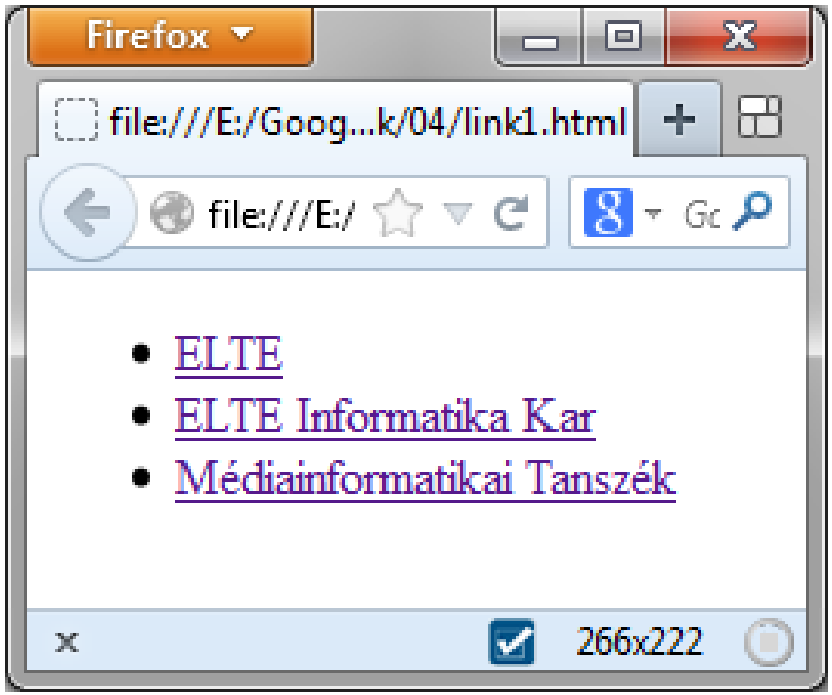
- Mik azok az események?
- Események típusai
- Eseménykezelők
- Eseménykezelők hozzárendelése, regisztrálása
  - inline (HTML attribútumként: `<elem onclick="fv()">`)
  - tradícionális (JS tulajdonságként: `elem.onclick = fv;`)
  - szabványos (`elem.addEventListener(típus, fv, irány);`)
  - Microsoft (`elem.attachEvent(típus, fv);`)

További kérdések

- ☑ Események típusai
- ☑ Eseménykezelő regisztrálása
- ☐ Eseményt kiváltó DOM objektum
- ☐ Eseményobjektum megszerzése
- ☐ Eseményobjektum tulajdonságai
- ☐ Alapértelmezett művelet megakadályozása
- ☐ Események buborékolása
- ☐ Eseményt először kiváltó DOM objektum

Példa

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



Példa – HTML


```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <script type="text/javascript" src="link.js"></script>
  </head>
  <body>
    <ul>
      <li><a href="http://www.elte.hu">ELTE</a></li>
      <li><a href="http://www.inf.elte.hu">ELTE Informatika
Kar</a></li>
      <li><a href="http://www.inf.elte.hu/mot">Médiainformatikai
Tanszék</a></li>
    </ul>
  </body>
</html>
```

1. megoldás

- Minden linkhez id
- Mindegyik link eseménykezelőjére külön függvényt készítünk

```
<ul>
  <li><a id="link1" href="http://www.elte.hu">ELTE</a></li>
  <li><a id="link2" href="http://www.inf.elte.hu">IK</a></li>
  <li><a id="link3" href="http://www.inf.elte.hu/mot">MOT</a></li>
</ul>
```

1. megoldás


```
function init() {
  $('link1').addEventListener('click', mutat1, false);
  $('link2').addEventListener('click', mutat2, false);
  $('link3').addEventListener('click', mutat3, false);
}
function mutat1() {
  var href = $('link1').href;
  console.log(href);
  alert('stop');
}
function mutat2() {
  var href = $('link2').href;
  console.log(href);
  alert('stop');
}
function mutat3() {
  var href = $('link3').href;
  console.log(href);
  alert('stop');
}
```

1. megoldás


```
function init() {
  $('link1').addEventListener('click', mutat1, false);
  $('link2').addEventListener('click', mutat2, false);
  $('link3').addEventListener('click', mutat3, false);
}
function mutatKozos(obj) {
  var href = obj.href;
  console.log(href);
  alert('stop');
}
function mutat1() {
  mutatKozos($('link1'));
}
function mutat2() {
  mutatKozos($('link2'));
}
function mutat3() {
  mutatKozos($('link3'));
}
```

1. megoldás

- Három függvény oka: eseménykezelőkben csak konkrét id-val tudunk egyelőre hivatkozni
- Ezért annyi függvény kell, ahány link – felesleges
- Kérdés: nem lehet az eseménykezelő függvényen belül az eseményt kiváltó objektumot elérni?
- Így elég lenne mindhárom link kattintására ugyanazt a függvényt futtatni.

Eseményt jelző DOM objektum

- Az eseményt jelző (vagy az eseménykezelőt meghívó) DOM objektum az eseménykezelő függvényen belül a this kulcsszóval érhető el.
- Sok elem eseményéhez ugyanaz az eseménykezelő rendelhető, mégis kinyerhető az eseményt kiváltó objektum

2. megoldás


```
function init() {
  $('link1').addEventListener('click', mutat, false);
  $('link2').addEventListener('click', mutat, false);
  $('link3').addEventListener('click', mutat, false);
}
function mutat() {
  console.log(this);
  var obj = this;
  var href = obj.href;
  console.log(href);
  alert('stop');
}
```

2. megoldás

- Honnan tudom, hogy a SHIFT billentyű le volt-e nyomva kattintás közben?
- Általában: honnan tudom
  - milyen billentyűt nyomtak le?
  - melyik egérgombot nyomták le?
  - a képernyőn hol van az egér pozíciója?
  - milyen típusú esemény váltódott ki?
- → eseményobjektum

Eseményobjektum megszerzése

- Szabványos (W3C)
  - Az eseménykezelő függvény első paramétereként érkezik automatikusan

```
function fv(e) {

}
```

Eseményobjektum tulajdonságai

- Milyen értékek érhetők el az eseményobjektumon keresztül?
  - egér (pozíció, gomb)
  - billentyűzet (gomb)
  - esemény (típus, stb)
  - elemek (kiváltó, eredeti, stb.)
- Szabványos
- Történetileg nagy eltérések

Eseményobjektum tulajdonságai

- Egérpozíció
  - clientX, clientY (DOM dokumentumhoz képest)
  - screenX, screenY (képernyőhöz képest)
- Egérgomb
  - button
- Módosítógombok (logikai)
  - shiftKey
  - altKey
  - ctrlKey

Eseményobjektum tulajdonságai

- Billentyűzet
  - keyCode
  - charCode
  - which
- Tulajdonság
  - type
  - target
  - currentTarget
  - bubbles
  - cancelable

3. megoldás


```
function init() {
  $('link1').addEventListener('click', mutat, false);
  $('link2').addEventListener('click', mutat, false);
  $('link3').addEventListener('click', mutat, false);
}
function mutat(e) {
  console.log(e);
  if (e.shiftKey) {
    var obj = this;
    var href = obj.href;
    console.log(href);
    alert('stop');
  }
}
```

3. megoldás

- SHIFT-et lenyomva ne jelenjen meg az új oldal, csak írjuk ki a konzolra
- alert nem megoldás
- Meg kellene akadályozni, hogy a böngésző végrehajtsa az oldalbetöltést
- →Általánosan: az alapértelmezett művelet megakadályozása szükséges

Alapértelmezett művelet megakadályozása

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

```
e.preventDefault();
```

  - hamis értékkel visszatérni a függvényből

```
return false;
```

Alapértelmezett művelet megakadályozása

- preventDefault()
  - szabványos
  - bárhol szerepelhet az eseménykezelő függvényen belül
- return false
  - nem szabványos
  - mindenhol működik
  - az utolsó sorba kell tenni

4. megoldás


```
function init() {
  $('link1').addEventListener('click', mutat, false);
  $('link2').addEventListener('click', mutat, false);
  $('link3').addEventListener('click', mutat, false);
}
function mutat(e) {
  if (e.shiftKey) {
    e.preventDefault();
    var obj = this;
    var href = obj.href;
    console.log(href);

  }
}
```

4. megoldás


```
function init() {
  $('link1').addEventListener('click', mutat, false);
  $('link2').addEventListener('click', mutat, false);
  $('link3').addEventListener('click', mutat, false);
}
function mutat(e) {
  if (e.shiftKey) {

    var obj = this;
    var href = obj.href;
    console.log(href);
    return false;
  }
}
```

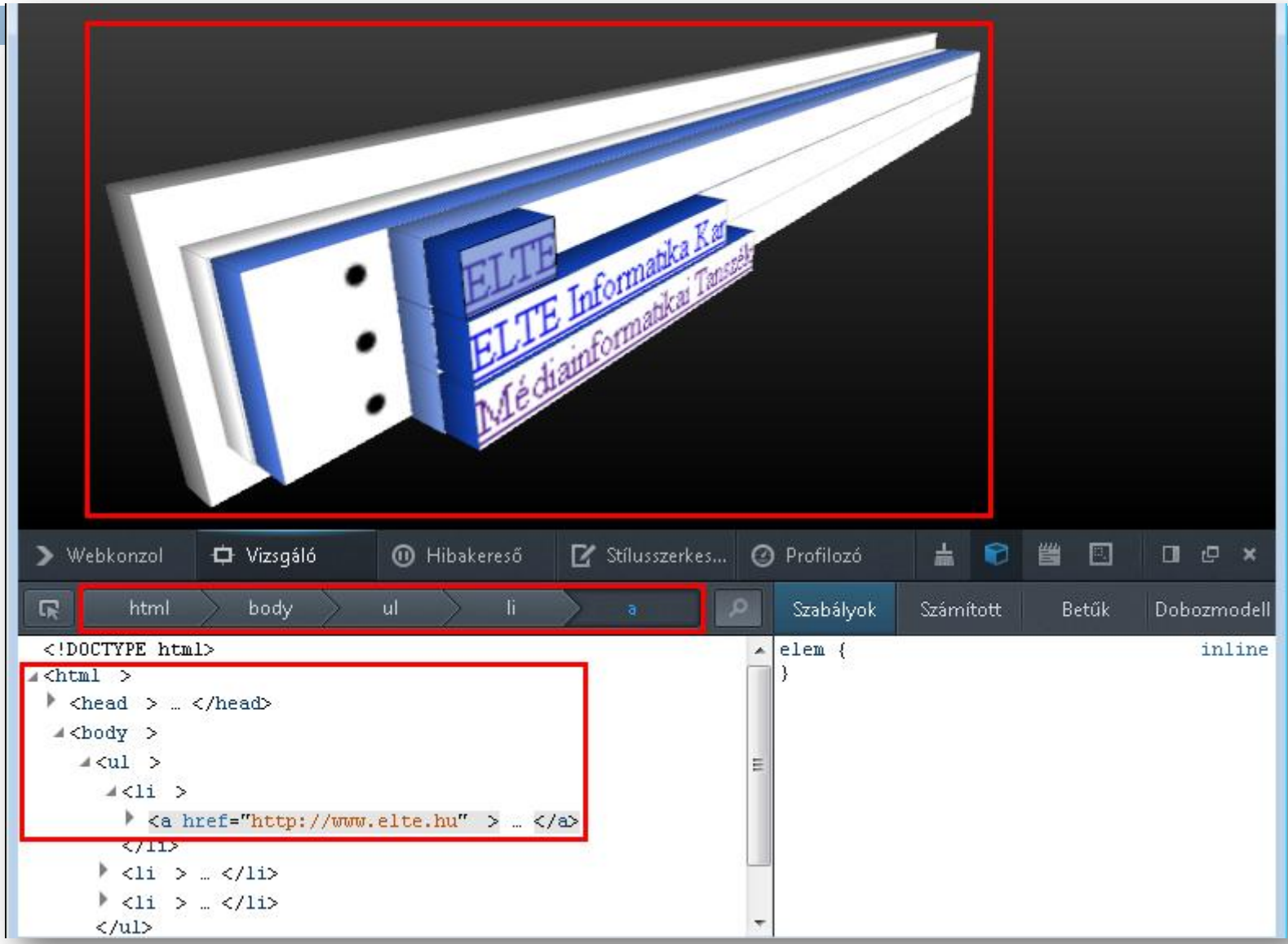
4. megoldás

- Az eseménykezelő függvény OK
- Az eseménykezelő függvény hozzárendelése egyesével történik az érintett elemeken
  - feleslegesen ismétlődik
  - új link beszúrása?
- Ciklusban hozzárendelni
  - félmegoldás és egyéb bonyodalmak lehetnek
- → Események buborékolása

Események buborékolása

- Egy esemény nemcsak a közvetlenül érintett elemen hívódik meg, hanem mindegyik szülőelemen is.
- Mindig elérhető az eredeti kiváltó objektum
  - target (szabvány)
  - srcElement (Internet Explorer)

Események buborékolása



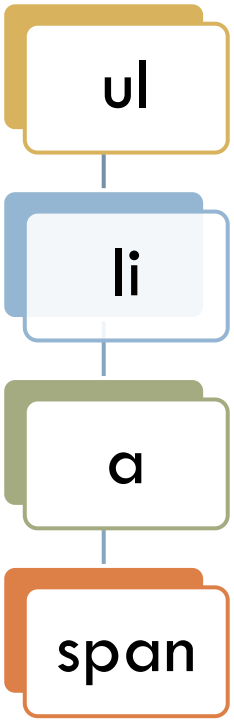
5. megoldás


```
<ul id="lista">
  <li><a href="http://www.elte.hu">ELTE</a></li>
  <li><a href="http://www.inf.elte.hu">ELTE Informatika
Kar</a></li>
  <li><a href="http://www.inf.elte.hu/mot">Médiainformatikai
Tanszék</a></li>
</ul>
```

```
function init() {
  $('lista').addEventListener('click', mutat, false);
}
function mutat(e) {
  if (e.target.tagName === 'A') {
    if (e.shiftKey) {
      e.preventDefault();
      var obj = e.target;
      var href = obj.href;
      console.log(href);
    }
  }
}
```

5. megoldás

```
<ul id="ul">
  <li><a id="link1" href="http://www.elte.hu"><span>ELTE</span></a></li>
  <li><a id="link2" href="http://www.inf.elte.hu">ELTE <span>Informatika
Kar</span></a></li>
  <li><a id="link3" href="http://www.inf.elte.hu/mot">Médiainformatikai Tanszék</a></li>
</ul>
```




```
function mutat(e) {
  var target = e.target;

  while (target && target.tagName !== 'A') {
    if (target === $('ul')) {
      return;
    }
    target = target.parentNode;
  }

  if (e.shiftKey) {
    e.preventDefault();
    var href = target.href;
    console.log(href);
  }
}
```

5. megoldás

- dinamikus
- rövid kód
- egy elem, egy eseménykezelő
- delegálás
- e.stopPropagation() – megállítja a buborékolást

Böngészőfüggetlenség

Eseménykezelők regisztrációja, eseményobjektum megszerzése, eseményobjektum tulajdonságai

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, kezelo);
  }
  //Tradicionális módszer
  else {
    elem['on' + tipus] = kezelo;
  }
}
```

```
function init() {
  kezelotRegisztral($('lista'), 'click', mutat);
}
```

Eseményobjektum megszerzése

- Szabványos (W3C)
  - Az eseménykezelő függvény első paramétereként érkezik automatikusan

```
function fv(e) {

}
```

- Internet Explorer (régebbi)
  - Az eseménykezelő függvényen belül a globális window.event objektum az

```
function fv() {
  e = window.event;
}
```

Eseményobjektum megszerzése

- Egységes megoldás
  - Ha nem jön paraméterként, akkor a globális értéket kell venni.
  - Minden eseménykezelőnek így kell kezdődnie

```
function fv(e) {
  if (!e) e = window.event;
  //... e felhasználása
}
```

Eseményobjektum megszerzése


```
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;
  }
}
```

Eseményobjektum tulajdonságai

Szabványos

- type
- target
- button (értékek)
- keyCode

Internet Explorer

- type
- srcElement
- button (értékek)
- which

Eseményobjektum tulajdonságai

- Egységes megoldás – karakter kódja
- Megnézni, melyik tulajdonság tartalmaz értelmes értéket – feature detection

```
function fv(e) {
  var kod;
  if (!e) var e = window.event;
  if (e.keyCode) kod = e.keyCode;
  else if (e.which) kod = e.which;
}
```

Eseményobjektum tulajdonságai

- Egységes megoldás – eseményt kiváltó objektum
- feature detection
- Safari böngészőhiba kezelése

```
function fv(e) {
  var obj;
  if (!e) var e = window.event;
  if (e.target) obj = e.target;
  else if (e.srcElement) obj = e.srcElement;
  if (obj.nodeType == 3) { // Safari bug
    obj = obj.parentNode;
  }
}
```

Nyelvi elemek

Globális függvények, JavaScript beépített objektumai

Globális függvények

- Számokkal kapcsolatos
  - isNaN(s)
  - isFinite(s) – véges-e a szám
  - parseInt(s, számrendszer)
  - parseFloat(s)
- Kódolással kapcsolatos
  - decodeURI
  - decodeURIComponent
  - encodeURI
  - encodeURIComponent

NaN

- Not a Number
- Speciális érték, ha egy művelet nem értelmezhető a számok halmazán
- Vizsgálata: isNaN(érték)

```
var a = 'alma' / 2;
isNaN(a);  //true
```

Infinity

- Végtelen jelölésére szolgál
- +Infinity, -Inifity
- Vizsgálata: isFinite()

```
var a = 10/0;
isFinite(a);  //false
```

Szövegből szám

- parseInt(szöveg, számrendszer)
- parseFloat(szöveg)

```
console.log( parseInt('123') );     //123
console.log( parseInt('123', 10) ); //123
console.log( parseInt('0101', 2) ); //5
console.log( parseInt('alma', 10) ); //NaN
console.log( parseInt('5alma', 10) ); //5

console.log( parseFloat('4.54') ); //4.54
```

Szövegkódolás

- Speciális karakterek
  - tárolása
  - küldése
- Régi: escape(), unescape()
  - ne használjuk
- encodeURI(), decodeURI()

```
var kod = encodeURI('árvíztûrõtükörfúrógép');
//"%C3%A1rv%C3%ADzt%C5%B1r%C5%91t%C3%BCk%C3%B6rf%C3%BAr%C3%B3g%C3%A9p"
var dekod = decodeURI(kod);
//"árvíztûrõtükörfúrógép"
```

Dátum függvények

- Date objektum
- Aktuális időpont
  - new Date()
- Megadott időpont
  - new Date(year, month, day [, hour, minute, second, millisecond]);
- Műveletcsoportok
  - getterek: getMonth(), getDay(), stb
  - setterek: setMonth(), setDay(), stb.
  - toString metódusok: pl. toLocaleString(), toGMTString()

Dátum példa


```
var most = new Date();
console.log(most.toLocaleString());
console.log(most.getFullYear());
most.setFullYear(2011);
console.log(most.toLocaleString());

var maskor = new Date(2011, 2, 28);
console.log(maskor.toLocaleString());
//"2011. március 28. 0:00:00"

console.log(maskor - most); //ms-okban
```

String műveletek

- charAt(i), charCodeAt(i)
  - egy karakter vagy kódja
- indexOf(mit, honnan), lastIndexOf(mit, honnan)
  - keresés elölről, hátulról, indexet ad vissza
- localeCompare(mivel)
  - összehasonlítás, -1, 0, 1
- search(regexp), replace(mit, mivel), match(regexp)
  - keresés, csere, reguláris kifejezésekkel is
- substr(honnan, hossz), substring(mettől, meddig), slice(mettől, meddig)
  - részszöveg
- split(szeparátor)
  - szöveg felbontása → tömb

String példa


```
console.log('piros alma'.charAt(2)); //"r"
console.log('piros alma'.charCodeAt(2)); //114
console.log('piros alma'.indexOf('alma')); //6
console.log('piros alma'.localeCompare('piros körte')); //-1
console.log('piros alma'.replace('piros', 'sárga')); //"sárga alma"
console.log('piros alma'.substr(2, 3)); //"ros"
console.log('piros alma'.split(' ')); //["piros","alma"]
```

Math objektum

- Matematikai műveletek
  - sin(rad), asin(sz), cos(rad), acos(sz), tan(rad), atan(sz)
  - pow(alap, kit), exp(n), log(n), stb.
  - random()
  - round(n), floor(n)
- Konstansok
  - PI
  - E
  - SQRT2
  - LN2, stb.

Math példa


```
console.log(Math.PI);  //3.141592653589793
console.log(Math.sin(90));  //0.8939966636005579
console.log(Math.sin(90 * Math.PI / 180));  //1
console.log(Math.random());  //pl. 0.47057095554085615
console.log(Math.random());  //pl. 0.5286946792885748
console.log(Math.round(1.6));  //2
console.log(Math.floor(1.6));  //1
```

Tömb műveletek

- pop(), push(e), shift(e), unshift()
  - végéről, végére, elejére,elejéről
- reverse()
  - megfordít
- splice(honnan, mennyit)
  - kivág
- join(szeparátor)
  - szöveggé fűz

Tömb példa


```
var t = [1, 2, 3, 4, 5];
t.push(6);
console.log(t);  //[1, 2, 3, 4, 5, 6]
console.log(t.pop());  //6
console.log(t);  //[1, 2, 3, 4, 5]
t.unshift(0);
console.log(t);  //[0, 1, 2, 3, 4, 5]
console.log(t.shift());  //0
console.log(t);  //[1, 2, 3, 4, 5]
t.reverse();
console.log(t);  //[5, 4, 3, 2, 1]
t.splice(2, 1);
console.log(t);  //[5, 4, 2, 1]
console.log(t.join('###'));  //"5###4###2###1"
```

Összefoglalás

- Eseménykezelés
  - Eseményt jelző objektum
  - Eseményobjektum
  - Buborékolás
  - Eseményt eredetileg jelző objektum
- Nyelvi elemek
  - Globális függvények
  - JavaScript beépített objektumai