

## Kliens oldali script nyelvek - JavaScript

# Áttekintés

- **Bevezetés**
- HTML DOM
- JavaScript beszúrása
- JavaScript szintaxis
- JavaScript események
- Egyéb JavaScript beépített objektumok

# Áttekintés

- Bevezetés
- **HTML DOM**
- JavaScript beszárasa
- JavaScript szintaxis
- JavaScript események
- Egyéb JavaScript beépített objektumok

# Áttekintés

- Bevezetés
- HTML DOM
- **JavaScript beszúrása**
- JavaScript szintaxis
- JavaScript események
- Egyéb JavaScript beépített objektumok

# Áttekintés

- Bevezetés
- HTML DOM
- JavaScript beszárasa
- **JavaScript szintaxis**
- JavaScript események
- Egyéb JavaScript beépített objektumok

# Áttekintés

- Bevezetés
- HTML DOM
- JavaScript beszárasa
- JavaScript szintaxis
- **JavaScript események**
- Egyéb JavaScript beépített objektumok

# Áttekintés

- Bevezetés
- HTML DOM
- JavaScript beszárasa
- JavaScript szintaxis
- JavaScript események
- Egyéb JavaScript beépített objektumok

# JavaScript

- a JavaScript a Web-en használt legnépszerűbb script nyelv
- az elterjedtebb böngészők nagyrésze ismeri (Internet Explorer, Mozilla, Firefox, Netscape, Opera)
- értelmező (interpreter) nyelv – a kód értelmezése dinamikusan történik
- szintaxisa hasonlít a Java programozási nyelvéhez



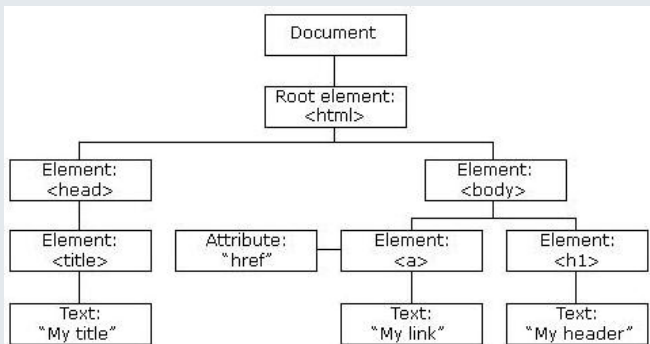
# JavaScript

## amit nyújt:

- a HTML oldalak készítői számára egy programozási eszközt biztosít (használatja egyszerű)
- módosíthatja a HTML tartalmát, kinézetét (a HTML DOM objektumaihoz való hozzáférés által)
- eseményekre tud reagálni (pl. oldal betöltése, kattintás egy elemre, stb.)
- a bevitt adat helyességének ellenőrzésére ad lehetőséget (mielőtt elküldenénk a szerverre)
- megvizsgálhatjuk a böngésző típusát, és ennek függvényében más-más, böngésző-specifikus tartalmat tölthetünk be
- sütiket (cookies) hozhatunk létre a kliens gépén való információ-tárolás érdekében

# HTML Document Object Model

- A HTML Dokumentum Objektum Modell (HTML DOM) lehetőséget teremt a dokumentum elemeihez való hozzáférésre és ezek módosítására
- a DOM a HTML dokumentumnak egy fa-szerkezetet feleltet meg.  
Csomópontok:
  - dokumentum csomópont – a teljes dokumentum
  - elem csomópont – minden HTML elem (tag)
  - szöveg csomópont – az egyes HTML elemek törzse
  - attribútum csomópont – a HTML attribútumok
  - megjegyzés csomópontok – HTML kommentek
- DOM objektum modell részei (alap, **HTML DOM**, XML DOM)



```
<html>
<head> <title>My title</title></head>
<body>
  <a href="...">My link</a>
  <h1>My header</h1>
</body>
</html>
```

# DOM

## DOM objektumok közti kapcsolat:

- szülő (parent)
- gyerek (child)
- testvér (sibling)
- előd (ancestor)
- utód (descendant)

## csomópontokhoz való hozzáférés:

- **getElementById()**, **getElementsByTagName()** metódusok segítségével
- egy elem csomópont **parentNode**, **childNodes**, **firstChild**, **lastChild**, **previousSibling**, **nextSibling**, ... mezőit használva
- sajátos csomópont: **document.body**

# DOM

## egy csomópont standard mezői:

- információt nyújtanak az illető csomópontról
  - nodeName – csomópont neve (tag-név, attribútumnév, #text, #document)
  - nodeValue – szöveg- illetve attribútum csomópontok esetén a szöveget illetve attribútum értékét tartalmazza
  - .nodeType – csomópont típusa
- megj.: egy-egy csomópont számos más mezővel is rendelkezik, illetve hozzárendelt metódusokkal

# DOM (JavaScript) objektumok

- **window** – a JavaScript hierarchiában a legfelső szinten levő objektum. A böngészőablaknak felel meg.
  - kollekció: `frames[]`
  - mezők: `document`, `history`, `location`; `outerHeight`, `outerWidth`; `statusbar`, `scrollbars`, `toolbar` (láthatóak legyenek vagy sem); `self`, `top`, `status`
  - metódusok: `blur()`, `focus()`; `alert()`, `confirm()`, `prompt()`; `open()`, `close()`; `setTimeout()`, `clearTimeout()`, `setInterval()`, `clearInterval()`; `scrollBy()`, `scrollTo()`; `moveBy()`, `moveTo()`; `resizeTo()`, `resizeBy()`

## előugró ablakok

- figyelmeztető ablak (alert box): `alert("szoveg")`  
inkább hibakeresés (debug) céljából használják
- jóváhagyó ablak (confirm box): `confirm("szoveg")`  
jóváhagyás kérése "fontosabb" művelet végrehajtása előtt
- adatbekérő ablak (prompt box): `prompt("szoveg", "alapertelmezett")`  
pl. adatbekérés az oldal betöltése előtt (ritkán használt)

# DOM (JavaScript) objektumok

## a window objektum alobjektumai:

- **history** – az illető böngészőablakból meglátogatott URL-eket tartalmazza (`window.history`)  
mezők, metódusok: `length`, `back()`, `forward()`, `go()`
- **location** – információ az aktuális URL-ről (`window.location`)  
mezők, metódusok: `href` (teljes URL), `protocol`, `hostname`, `port`, `pathname`, `hash` (# utáni rész), `search` (? utáni rész); `assign()`, `reload()`, `replace()`
- **document** ... →

# DOM objektumok

## document objektum

- a teljes HTML dokumentumhoz való hozzáférést teszi lehetővé
- a window JavaScript-objektum része (`window.document`)
- kollekciók:
  - `anchors[]`, `forms[]`, `images[]`, `links[]`
- mezők:
  - `body`, `cookie`, `domain`, `lastModified`, `referrer`, `title`, `URL`
- metódusok:
  - `getElementById()`, `getElementsByName()`,  
`getElementsByTagName()`, `open()`, `close()`, `write()`, `writeln()`

## HTML elemeknek megfelelő objektumok esetén:

- beállítható, átállítható bármely attribútum értéke
- az elem törzse az *innerHTML* mezőn keresztül érhető el  
lásd: [Peldak/csere.htm](#)
- bővebb információért → HTML DOM referencia



## JavaScript beágyazása HTML oldalba

```
<script LANGUAGE="JavaScript1.5">  
...  
</script>
```

## JavaScript-et nem használó régebbi böngészők esetén:

```
<script LANGUAGE="JavaScript">  
<!--  
...  
//-->  
</script>
```

## Külső script-állomány beszárasa

```
<script src="kulso.js"></script>
```

- több HTML oldal is használhatja
- átláthatóbb kód: HTML illetve JavaScript szétválasztása

## JavaScript elhelyezése:

- elvileg bárhova elhelyezhető (script elembe ágyazva)
- a script kiértékelése az *oldal betöltése közben* történik.
- függvények, globális változók deklarálása a **head** elemben ajánlott
  - az itt megírt függvényeket pl. meghívhatjuk a body egy elemének valamilyen eseményére

## a nyelv szintaxisa

- Java-hoz hasonló (if, if else, while, switch, for, while, do while, break, continue, stringkezelés ...)

## változók

- a változók típusának meghatározása dinamikusan történik
- a JavaScript különbséget tesz kis és nagybetűk közt
- a változók deklarálása nem kötelező (de megtehető a **var** kulcsszó segítségével, típus megadása nélkül)
- a változók neve betűvel vagy a ("\_") karakterrel kell kezdődjön
- a globális változók hatóköre deklarálásuktól (első használatuktól) a HTML oldal bezárásáig tart

## függvények

```
function fuggvenynev(valtozo1, valtozo2,...)  
{  
... return(ertek) // nem kötelező  
}
```

- meghívhatóak – a HTML oldal bármely pontjáról
  - egy esemény bekövetkeztekor
- deklarálása – a head elembe ajánlott

# Objektumok, tömbök használata

tömb létrehozása az Array beépített objektum segítségével:

```
var sorozat=new Array(); sorozat[0]="páros";  
sorozat[1]="páratlan";  
var sorozat=new Array("páros","páratlan")
```

- a JavaScript tömbök tulajdonképpen index-el ellátott mezőkkel rendelkező objektumokként vannak tárolva

általános objektum létrehozása

```
var obj=new Object(); // létrejön egy "üres" objektum
```

- a mezőket hozzáadhatjuk menet közben (ha eddig nem volt ilyen mező, érték-hozzárendeléskor automatikusan létre lesz hozva):  
obj.nev="Móricka"  
obj.eletkor=21
- a mezőkre az alábbi szintaxissal is hivatkozhatunk ("tömb jelleg"):  
obj['nev']="Móricka"  
előny: a szögletes zárójelen belül használhatunk kifejezést is

### új függvény hozzáadása az objektumhoz:

- inline módon:

```
obj.hanyEves=function(){  
    alert(this.nev+" "+this.eletkor+" éves")  
}
```

- előzőleg definiált függvény hozzáadása:

```
function koszon(){  
    alert("Szia "+this.eletkor)  
}
```

...

```
obj.koszon=koszon // vigyázat, nem obj.koszon=koszon()
```

### alobjektum hozzárendelése az objektumhoz:

```
var myLibrary= new Object();  
myLibrary.books=new Array();  
myLibrary.books[0]=new Object();  
myLibrary.books[0].title="Kétévi vakáció";  
myLibrary.books[0].authors=new Array();  
var vernegyula=new Object();  
vernegyula.name="Jules Verne";  
vernegyula.foglalkozas="ifjúsági regényíró";  
myLibrary.books[0].authors[0]=vernegyula;
```

- ...bonyolultabb struktúra felépítése kissé nehézkes ...
- gyorsabb megoldás: JSON (JavaScript Object Notation)

## JSON – JavaScript Object Notation

- tömb esetén  
`myLibrary.books=[konyv1, konyv2, konyv3]`
- objektum esetén (kapcsos zárójelben felsorolt “kulcs:érték” párosok):  
`myLibrary.books={  
 mese : konyv1,  
 regeny : konyv2,  
 zoldsegpucolo: konyv3  
}`
- komplex objektumhierarchia építhető fel ilyen módon
- a mezőknek való értékadáskor kifejezést is használhatunk (dinamikus tartalom)



## JSON –függvény-mező megadása egy objektumnak:

```
var barkacskonyv={
  cim: "Csinald magad "+kotetszam+".  kotet",
  szerzok:[{nev:"Valaki",ev:25},{nev:"Barki", ev:50}],
  megj: function(hossz){
    if(!hossz){hossz=10}
    var szov=this.cim+", "+this.szerzok[0].nev+" konyve n"
    for(i=0; i<hossz;i++){
      szov+="a"
    }
    szov+="gyon unalmas!"
    alert(szov);
  }
}
...
barkacskonyv.megj(5)
```

- a JSON jelölés illetve egyszerű JavaScript használata kiegészíthetik egymást

# Saját objektum létrehozása

Saját objektum létrehozása “konstruktor”-szerű függvény segítségével:

```
function MyObject(name, size){  
  this.name=name  
  this.size=size  
}
```

- később ennek segítségével létrehozhatjuk a MyObject típusú objektumot:

```
var myobj= new MyObject("nadragszija","5m");  
alert("a(z) "+myobj.name+" merete "+myobj.size);
```

- a konstruktoron belül bármilyen mezőt rendelünk hozzá a this-hez, az elérhető lesz az objektum mezőjeként

# Saját objektum + függvény

## függvény-típusú mező hozzáadása az objektumhoz

- a **prototype** (az objektumokhoz rendelt speciális mező) segítségével ajánlott

```
MyObject.prototype.tellSize=function(){  
    alert("a(z) "+this.name+" merete "+this.size)  
}
```

...

```
var myobj= new MyObject("nadragszija","5m");  
myobj.tellSize()
```

## példák

- Objektum\_JSON.htm, SajatObjektum.htm, SajatObjektum\_fuggvenymezo.htm, SajatObjektum\_prototype.htm
- Objektum mezőinek végigjárása **for in** ciklus segítségével: mezovegigjaras.htm

# JavaScript események

## JavaScript események:

- a JavaScript által felismert történések
- dinamikus (időben változó tartalmú) HTML oldalak létrehozását teszik lehetővé
- minden egyes HTML eleméhez vannak hozzárendelt események, melyek hatására JavaScript függvényt futtathatunk le
- a HTML tag-ben adjuk meg ezeket: pl.

```
<input type="button" value="megnyom"  
onClick="gombnyomas()" />
```

# JavaScript események

## néhány esemény:

- **onload, onUnload**: oldal vagy kép betöltése, oldal elhagyása  
(pl. böngésző típusának ellenőrzése, sütik tárolása)  
lásd: [informaciok.htm](#), [cookie.htm](#)
- **onClick**  
(tipikusan gombnyomás vagy más HTML elemre -pl. kép- való kattintásra meghívunk egy JavaScript függvényt)
- **onFocus, onBlur, onChange**  
(pl. form elem értékének ellenőrzése, ha változott a tartalma)
- **onSubmit**  
(pl. a form összes adatainak ellenőrzése a szerverre való küldés előtt)
- **onMouseOver, onMouseOut**  
(pl. animált gombok)

# Egyéb JavaScript beépített objektumok

- Navigator – információ a kliens böngészőjéről  
mezők: `navigator.appName` (böngésző típusa),  
`navigator.appVersion` (böngésző verziószáma)
- Screen – információ a kliens képernyőjéről  
mezők: `screen.width` (képernyő szélessége), `screen.height` (magassága)

# Egyéb JavaScript beépített objektumok

- String – minden string típusu változó tképpen String típusú objektum a JavaScriptben  
mező: length  
metódusok: toUpperCase(), toLowerCase(); substring(), charAt(), indexOf(); toString() (szám átalakítása String-é)
- Date – dátum/idő beállítás/lekérdezés  
objektum létrehozása:  

```
birthday = new Date();  
birthday = new Date("June 20, 1996 08:00:00");
```

  
metódusok: setDate(), getDate(), set/get: Month Year Time Hours Minutes Seconds
- Math – automatikusan létre van hozva (explicit Math objektum létrehozása nélkül használható)  
metódusok: kerekítés – ceil(), floor(), round(); véletlenszám generálás (0 és 1 között) – random()