JavaScript alapok

A JavaScript főbb tulajdonságai

- *Script nyelv* – azaz futás közben értelmezett, **interpretált** nyelv
- Legfőbb alkalmazási területe: a HTML dokumentumok **dinamikussá, interaktívvá tétele**
- *Gyengén típusos* nyelv: Nem adhatjuk meg a változók típusát. Az a nekik adott értéktől függ; menet közben változhat is.
- Szintaxisa a C nyelven alapul; sokban hasonlít a Java-ra is.

JavaScript és HTML

Négyféleképpen adhatunk JavaScript kódot egy HTML dokumentumhoz:

- Külön fájlban tárolt kód csatolása:
  ```
  <script src="file.js"></script>
  ```
- Kód beágyazása Script node-dal:
  ```
  <script>
  // Ide jön a JavaScript kód
  </script>
  ```
- HTML elemek eseménykezelőiként:
  ```
  <input type="button" onclick="alert('na!')" />
  ```
- Hivatkozás URL-jeként:
  ```
  <a href="javascript:alert('hahó')">
  ```

Szintaxis

- Az utasításokat pontosvessző (;) vagy újsor jel határolja.
- A { és } jelek közé zárva **utasítás-blokk**okat hozhatunk létre.
- Megjegyzések:

```
// egysoros megjegyzés
/* többsoros
   megjegyzés */
```

- A JavaScript érzékeny a kis- és nagybetűk közti különbségre.

Változók, értékadás

- deklarálás:

```
var a,b;
var c=3; // deklarálás kezdőérték adásával
var s="Szöveg";
var z='másik szöveg';
```

- Nem kötelező deklarálni a változókat; az első értékadással mindenképpen létrejönnek.
- Függvényen belül deklarált változó csak ott érhető el. Ha esetleg van azonos nevű globális változó, azt *elfedi* a függvény futása idejére

Vezérlési szerkezetek

- elöl tesztelős ciklus:

```
while (feltétel) utasítás;
```

vagy:

```
while (feltétel) { utasítások;...; }
```

Az utasítás (vagy a blokkban lévő utasítások) újra és újra végrehajtódnak, amíg a feltétel (egy logikai érték) igaz.

Vezérlési szerkezetek

- Hátul tesztelős ciklus

```
do {
  utasítások; ...;
} while (feltétel);
```

Az utasítás (vagy a blokkban lévő utasítások) újra és újra végrehajtódnak, amíg a feltétel igaz, de legalább egyszer.

Vezérlési szerkezetek

- `for` ciklus:

```
for (utasítás1; feltétel; utasítás2) {
  ciklusmag;
};
```

A gép először végrehajtja `utasítás1`-et, aztán mindaddig ismétli a ciklusmag utasításait, amíg a `feltétel` igaz. A ciklusmag minden utasítása után végrehajtja `utasítás2`-t is.

A `for` ciklust leggyakrabban a ciklusmag valahányszor való ismételt végrehajtására használjuk. Ilyenkor használunk egy ciklusváltozót, melynek értékét folyamatoan léptetjük:

```
for (int f=1,i=2;i<=10;i++)
  f=f*i;
```

Vezérlési szerkezetek

- Elágazások:

```
if (feltétel) {
  utasítások;
} else {
  utasítások;
}
```

Ha `feltétel` igaz, akkor az első blokk hajtódik végre, ha nem, akkor a második.

A második rész el is hagyható, ilyenkor nincs ott az `else` sem.

Típusok

- boolean (logikai)
  ```
  a=true; b=false; c=(x>15);
  ```
- number (szám)
  ```
  d=1; e=2.34; f=1.56E15; o=015; h=0xff;
  ```
- string (szöveg)
  ```
  s1='alma'; s2="körte";
  ```
- array (tömb)
  ```
  t=[2,3, 'alma'];
  ```
- object (objektum)
  ```
  o={ fi: 47.5, lambda: 19.1 };
  ```
- function (függvény)
  ```
  f=function(x) { return x*x+3*x-1; };
  ```

A **typeof** operátorral tudhatjuk meg egy változó aktuális típusát.

Tömbök

A tömbök olyan összetett változók, melyek elemeit azok indexével érhetjük el. Az indexelés 0-val kezdődik:

```
var t=[2,4,6,7];
alert(t[2]); // 6-ot ír ki.
```

A tömb ismeri a hosszát:

```
alert(t.length); // 4-et ír ki
```

A tömb veremként és sorként is használható:

```
t.push(11); // betesz egy elemet a tömb végére
x=t.pop(); // kiveszi az utolsó elemet (most: 11)
y=t.shift(); // kiveszi az első elemet (most: 2)
```

Objektumok

Az objektumok olyan összetett változók, melyek név: érték párokat tartalmaznak.

```
hely={ fi:47.5, la:19.1,
nev:'Budapest' };
```

Az objektumok tagjaira `objektumnév.tagnév` módon hivatkozhatunk:

```
hely.fi=47.6;
```

Egy objektum önmagára a `this` kulcsszóval hivatkozhat.

Konstruktorok és a new operátor

Objektumok létrehozhatók ún. **konstruktorfüggvény** definiálásával, és utána a **new** operátor használatával:

```
function Pont(lat,lon,nev)
{
  this.fi=lat; this.la=lon; this.nev=nev;
  this.kiir=function()
    { alert(this.fi+','+this.la+':'+this.nev); }
}

var p1=new Pont(51,0,'London');
var p2=new Pont(47.5,19.1,'Budapest');
p2.kiir();
```

A DOM

Egy böngészőben a JavaScript számára a HTML dokumentumot az ún. DOM (Document Object Model) reprezentálja. Ebben minden HTML elemnek egy-egy objektum felel meg, melyek hierarchikus rendbe szerveződnek.

A DOM elemeit módosíthatjuk, így futás közben dinamikusan megváltoztathatjuk a HTML dokumentumot.

A DOM használata

A DOM „gyökere” a `document` nevű objektum. Ennek metódusai használhatók a DOM manipulációjára.

- Ismert ID-jú elemhez tartozó objektum:
  ```
  o=document.getElementById('valami');
  ```
- Új elem létrehozása:
  ```
  i=document.createElement('img');
  i.src='kep.jpg';
  ```
- Elem csatolása a DOM egy meglévő eleméhez:
  ```
  o.appendChild(i);
  ```
- Elem tartalmának megadása/változtatása:
  ```
  o.innerHTML+='<p>Még egy bekezdés</p>';
  ```