Javascript

Domonkos Sándor Balázs Phd hallgató

Programozási alapismeretek felzárkóztató kurzus

Jelen tananyag a Szegedi Tudományegyetemen készült az Európai Unió támogatásával.

Projekt azonosító: EFOP-3.4.3-16-2016-00014

Szegedi Tudományegyetem
Cím: 6720 Szeged, Dugonics tér 13.
www.u-szeged.hu
www.szechenyi2020.hu

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

 Oktató: **Domonkos Sándor Balázs**	 **Programozási alapismeretek**	 Becsült olvasási idő: **15 perc**

Javascript Negyedik Lecke

Eddig áttekintettük,hogy hogyan és nagyvonalakban mire tudjuk használni a JS-t HTML felületi elemeken módosításokat végrehajtani. Átnéztük a változó típusokat azaz miből tudunk és miként építkezni. Most áttekintjük az bejövő adatok után, hogyan tudjuk ezeket megjeleníteni illetve az alap vezérlési szerkezetek megfelelő alkalmazását.

Kimenet Kiíratás hova és miként?

Akár információt akarunk **közölni**, akár **hibakereséshez**:

- HTML document.write()
- HTML ELEM
- ALERT BOX window.alert()
- CONSOLE console.log()
- Print

HTML document.write()

```
<!DOCTYPE html>
<html>
<body>

<h2>Weblapom</h2>
<p>Hello Világ!</p>

<p>document.write parancsot sose hívjuk meg miután már betöltött a dokumentum, mivel az egész
dokumentumot letörli és csak a benne lévő szövegeket írja ki, viszont betöltés közben mint itt a
következő részben lehet alkalmazni.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Weblapom

Hello Világ!

document.write parancsot sose hívjuk meg miután már betöltött a dokumentum, mivel az egész dokumentumot letörli és csak a benne lévő szövegeket írja ki, viszont betöltés közben mint itt a következő részben lehet alkalmazni.

11

a document.write paranccsal tudunk **kiíratni** a HTML-be magára **új felületi elemeket**, de csak a dokumentum generálása közben, szóval **gombra rákötve** csak az ott lévő szövegeket fogja kiírni és kitörli a többit a dokumentumról.

Viszont mint a példa is mutatja a **kiíratásba tudunk logikát beépíteni** és az ott lévő parancsok is végrehajtódnak és a végeredményt fogjuk majd látni pl 5+6 a beírt parancs és 11 a kimenet amit kiírat végül.

HTML ELEM

```
<!DOCTYPE html>
<html>
<body>

<h2>Weblapom</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Weblapom

11

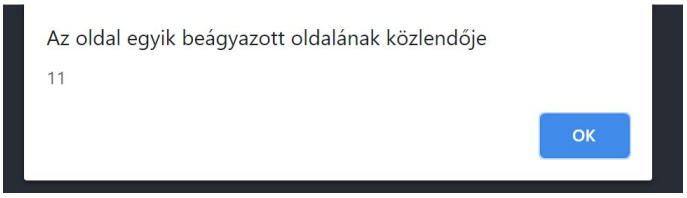
Itt is látszódik,hogy a **kimeneti logika végrehajtódik** és az összeadás végeredményét írja be szövegként a megfelelő id= demo html alkotóba.

ALERT BOX window.alert()

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

My First Web Page

My first paragraph.



Felugró értesítő ablakban a **dokumentum legenerálása előtt** már megjelenik az üzenet mielőtt bármit is látnánk a html dokumentumból és amíg nem kattintunk az OK gombra nem fog tovább menni.

CONSOLE console.log()

Legtöbbször ezt használjuk **hibakeresésnél** is, mert a debug eszközökkel ez a legjobban olvasható és legjobban használható.

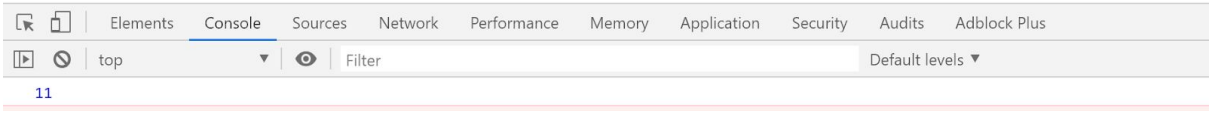
```
<!DOCTYPE html>
<html>
<body>

<h2>Debugging</h2>

<p>FNyomd meg az F12 gombot chromeban</p>
<p>Válaszd ki a console tabot a menüpontok közül</p>
<p>Nyomd meg a Run gombot bármelyik JS demo futattó környezetben és látni fogod a 11-et a console üzeneteknél.</p>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

Debugging

FNyomd meg az F12 gombot chromeban

Válaszd ki a console tabot a menüpontok közül

Nyomd meg a Run gombot bármelyik JS demo futattó környezetben és látni fogod a 11-et a console üzeneteknél.



JavaScript Print

Külön nem tudunk nyomtatókat célozni de a beépített adott oldal nyomtatás parancsot megtudjuk hívni JS eseményként is.

```
<!DOCTYPE html>
<html>
<body>


<p>Aktuális oldal nyomtatása.</p>

<button onclick="window.print()">Nyomtat</button>

</body>
</html>
```

Aktuális oldal nyomtatása.

Nyomtat

Utasítások és kódolási stílusok a JS kód alapjai

Egy számítógépes program egy utasítások listája amiket a számítógép hajt végre megadott sorrendben. Egy programozási nyelvben megtalálhatóak a programozási nyelv saját utasításai
Egy javascript program ilyen programozási utasítások egy listáját tartalmazza és ezeket folytatólagosan adott sorrendben hajtja végre.

Építő elemei:

Értékek,operátorok,kifejezések,kulcsszavak és kommentek.

Pontosvessző ;

Ez választja el az egyes utasításokat egymástól, egy adott utasítás végét jelöli
Több féle stílusban is lehet használni:

egyik féle verzió:
var a, b, c;
a = 5;
b = 6;
c = a + b;

és másként használva:
var a, b, c; a = 5; b = 6; c = a + b;

Az eltérő stílus **csak vizuálisan néz ki máshogy**,de az eredmény a **kód futtatásával azonos** lesz.

White Space vagyis üres karakter.

A JS nem foglalkozik azzal,hogy 1 vagy több üres karaktert ütünk a kódban egymás mögé ezeket 1 üres karakterként kezeli így csak a **kód stílusos formázásában van szerepe** annak,hogy 1 vagy 2 vagy 3 spacet ütünk az utasítások közé a jobb láthatóság kedvéért.

var person = "Hege";
var person="Hege";

Ez a kettő kód ugyanúgy hajtódik végre.
Általában operátorok + - = / * elé és mögé szokás 2-2 spacet ütni 1 helyett.

Sorhosszúság és sortörés

A fordító nem foglalkozik a kódban lévő **sortörésekkel** adott utasítás vagy sor végét számára egyedül a ; jel jelöli. Így az alábbi kódot pl problémamentesen végrehajtja:

document.getElementById("demo").innerHTML =
"Hello World!";

Stílus segédletként javasolt 80-100 karakterenként sortörést alkalmazni ez monitor mérettől függetlenül is **jól olvasható kódot** fog létrehozni.

Kód Blokkok

Különböző utasításokat egybe lehet fűzni és így ezeket funkcióknak nevezzük amiknek paramétereket is tudunk átadni és a { } jel közötti rész fog végrehajtódni sorrendben.

function name(parameter1, parameter2, parameter3) { kód rész
}
A name a kód blokk neve a mögötte lévő rész pedig az átadott paraméter lista lehet a { } közötte részbe kerülhet pedig maga a végrehajtandó kód.

```
function myFunction(param1,param2) {
  document.getElementById("demo1").innerHTML = "Hello!";
  document.getElementById("demo2").innerHTML = "Számolj velem!";
return ( param1+param2);
}
```

meghívása pedig így történik pl:
document.getElementById("demo").innerHTML = myFunction(4, 3);

```
<!DOCTYPE html>
<html>
<body>

<h2>JS functions</h2>

<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>

<script>
function myFunction(param1,param2) {
  document.getElementById("demo1").innerHTML = "Hello!";
  document.getElementById("demo2").innerHTML = "Számolj velem!";
return ( param1+param2);
}

document.getElementById("demo3").innerHTML = myFunction(4, 3);
</script>

</body>
</html>
```


Ezen a komplexebb példán 3db html elem helyére írjuk be a megfelelő szövegeket amiket a js scriptünk hajt végre demo1 demo2 demo3 helyekre a funkció meghívásával és a 4,3 paraméterek átadásával.

Javascript Keywords és alap vezérlési szerkezetek áttekintése

A keywords úgynevezett fentartott szavak a javascript nyelvben amiket sehol máshol nem lehet alkalmazni (nem lehetnek változó és eljárás nevek sem stb..) , speciális utasítások a végrehajtásra vonatkozóan, a legnépszerűbbeket itt felsorolom egy rövid példával szemléltetve.

for
Elöltesztelő ciklus a for ciklus feltétele már a ciklus végrehajtása előtt lefut , így előfordulhat,hogy egyszer sem fut le a ciklusmagja.

```
for(var a=0; a<=10; a++) {
document.write("The loop is running for " + a + " times");
}
```

Kiíratja az a változó értékét amíg az a nem éri el a 10-es értéket.
a változó 0-ról indít , 10-es értékig fut és a a++ azt jelenti minden ciklus végén eggyel növeli az a értékét.

if ... else
Az előző elöltesztelő ciklus kiegészített verziója, itt megadhatjuk,hogy ha nem teljesül az if feltétel akkor mi fusson le “egyéb” esetre.

```
var date = new Date();
var day = date.getDay();
if(day==6 | |  day==7  ) {
alert("Hetvege!");
} else {
alert("Hetkoznap!");
```

Lekérjük a Date függvénnyel az adott napot a date változóba, a date változót a .getday funkcióval a day változóba lekérjük numerikusan. Az if ciklus azt vizsgálja,hogy a nap száma a héten belül 6 vagy 7 || jelenti a VAGY operátort ez esetben kírjuk,hogy hétvége van. Egyéb esetben azt,hogy hétköznap van.

while
Ismétlési szerkezet az úgynevezett hátul tesztelős verzió, itt a ciklus magja egyszer legalább mindenképpen lefut és a feltétel tesztelés a ciklus végén hajtódik le.

```
var a=10;
while(a <= 10)
{
document.write("loop is running for " + a + "times</p>");
a++;
}
```

Kiíratja az a változó értékét és hiába 10 az a változó értéket egyszer mindenképpen lefut.

switch
Több esetet lehet megadni,hogy adott esetre vonatkozóan mi történjen és mi hajtódjon végre.

```
var date = new Date();
switch(date.getDay()) {
case 6:
alert("Hetvege ");
break;
case 7:
alert("Hetvege ");
default:
alert("Hetkoznap");
break;
}
```

Adott példánál a switchben az értéket a date.getday() adja meg ami a héten belül az adott nap számát adja meg így a hatos eset és a 7-es eset hétvégével tér vissza az összes többi (default) eset pedig azzal,hogy hétköznap.

break
Egy ismétlési szerkezetek végrehajtását tudjuk megszakítani vele.

```
for(var a=0; a<=10; a++) {
if(a == 5)
break;
document.write("The loop is running for " + a + " times");
}
```

Itt például a for 10 alkalommal futna le de a if a==5 miatt 5 ismétlésnél a break parancs megszakítja a végrehajtását a loopnak.

continue
Ismétlési szerkezetben átugorja a rákövetkező részt és utána folytatja a loop következő elemmel.

```
 for (i = 0; i < 10; i++) {
  if (i === 3) { continue; }
  text += "The number is " + i + "<br>";
}
```

Szóval adott példánál ha a i=3-nál járunk akkor a continue miatt a kiíratás nem fog végrehajtódni és halad továbba 4-re.

debugger
Megállítja az adott javascript futását és behívja a debuggert ha elérhető, a chrome esetében feldobja a debugger consolet.

```
var prod = 10 * 10;
debugger;
document.getElementbyId("id").innerHTML = prod;
```

Meghívja a debuggert és a prod változó értékét beleírja.

function
Egy kód blokk egység elnevezésére használható, ami után ezzel tudjuk meghívni majd a kód blokkot.

```
var func = function(){
```

return "Hello";
}
alert(func());

Adott példában a func kódblokkot definiálljuk amiben egy hello karakter sorozat kiíratását adjuk meg azzal tér vissza mint érték. A végén az alert()-ben meghívjuka func nevű kódblokkunk ahol így az történik,hogy a func helyébe behelyettesítődik a hello string és azt íratja ki.
alert(func()); -> alert(“Hello”);

return
Visszaadja az adott értéket amit egy “ “ string is lehet illetve változó is .
var func = function(){
return "Hello";
}

var
Változó deklarálás és értékadás.

var prod = 10

Ellenőrző kérdések

1. Tudunk-e JS-ből nyomtatási parancsot hívni?
2. Sorolj fel 3 lehetőséget output kiíratásra!
3. Mit jelent a kód blokk kifejezés mit tartalmazhat?
4. Mi a lényegi különbség a for és a while ciklusok között?
5. Milyen karakterrel tudjuk az egyes utasításokat elválasztani egymástól?

Válaszok

Tudunk-e JS-ből nyomtatási parancsot hívni?
Igen megoldható a beépített böngésző nyomtatási ablak meghívása

Sorolj fel 3 lehetőséget output kiíratásra!
Print
Alertbox
Console
HTML elem

Mit jelent a kód blokk kifejezés mit tartalmazhat?
JS-ben ez a function megvalósítása egy függvényt lehet így létrehozni amiben több kódsort is elhelyezhetünk illetve paramétereket adhatunk át nekik.

Mi a lényegi különbség a for és a while ciklusok között?
Egyik elől a másik pedig hátul tesztelő, a while a hátul tesztelő úgy ott a ciklus magja egyszer mindenképp lefut a for pedig elől tesztelő így ott nem fut le a ciklus magja ha nem igaz a feltétel.

Milyen karakterrel tudjuk az egyes utasításokat elválasztani egymástól?
; az elválasztó