Interaktivitás, interaktív grafika

Az eddig kódokkal az a probléma, hogy mindig ugyanazt és ugyanúgy csinálják. Nem tudjuk például megadni a rajzolandó kör sugarát vagy a kiírandó szöveg méretét, stb. Kellene egy eszköz, hogy adatokat adhassunk meg a weboldal számára. Erre két módszer is létezik. Az egyik a prompt metódus, mellyel a billentyűzetről adatokat tudunk megadni. A másik az űrlap, erről majd később "emlékezünk meg". Nézzétek át a következő kódot!

```
<!DOCTYPE html>
<html>
<body>
<script>
    document.write('Hello, user'+"<br>");
    var a=prompt("Kérek egy számot");
    var b=prompt("Kérek még egy számot");
    document.write("<strong>A két szám összege: ");
    document.write(a+b);
    document.write("</strong>");
    document.write("<br>A két szám összege: ");
    document.write(parseInt(a)+parseInt(b));
</script>
</body>
</html>
```

A program két számot kér be, és rögtön tárolja is az a nevű és a b nevű változókba. Majd a weblapra kiírja a két szám összegét. A gond az, hogy hiábra adunk meg számot, a JavaScript valamiért szöveggként tárolja a bekért adatokat. Így amikor csak az a+b-t íratjuk ki, akkor a két szám nem összeadódik, hanem egymás mellé kerül, ezt úgy mondjuk "magyarul", hogy konkatenálódnak, azaz összefűzésre kerülnek mint két szöveg, A második kiírásnál már használtuk a parseInt metódust, amely a szövegből számot csinál. Adódik a kérdés: hogyan lehet ezt a programot többször is futtatni? A válasz: CTRL-R, azaz frissítsük az oldalt! De van ettől jobb megoldás is. Tegyük a kódot egy metódusba, tegyünk ki egy gombot az oldalra, és ha arra kattint a felhasználó, akkor hívjuk meg a metódust. Ezt mutatjuk meg a következő kódban.

```
<!DOCTYPE html>
<html>
<body>
<script>
  function osszead()
  {
      document.write('Hello, user'+"<br>");
      var a=prompt("Kérek egy számot");
      var b=prompt("Kérek még egy számot");
      document.write("<strong>A két szám összege: ");
      document.write(a+b);
      document.write("</strong>");
      document.write("<br>A két szám összege: ");
      document.write(parseInt(a)+parseInt(b));
  }
</script>

<form>
  <input type="button" value="Kattints rám" onClick="osszead()">
</form>
```

```
</body>
</html>
```

Ez már majdnem jó, csak éppen a document.write "törli" a gombot is. Azaz megint nem lehet többször is futtatni a programot. A helyes megoldás:

```
<!DOCTYPE html>
<html>
<body>
<script>
   function osszead()
   {
        var bekezdes=document.getElementById("eredmeny");
        bekezdes.innerHTML='Hello, user'+"<br>";
        var a=prompt("Kérek egy számot");
        var b=prompt("Kérek egy mág egy számot");
        bekezdes.innerHTML+="<br>A két szám összege: ";
        bekezdes.innerHTML+=parseInt(a)+parseInt(b);
   }
</script>

<p id="eredmeny">Ide kerül az eredmény</p>

<form>
  <input type="button" value="Kattints rám" onclick="osszead()">
</form>
</body>
</html>
```

A bekezdes változóba lekérjük az eredmeny bekezdést. Majd a bekezdés innerHTML tulajdonsága tartalmazza a bekezdés tartalmát. A "Hello, user...." szöveggel ezt felülírjuk, majd a +=-vel a korább tartalomhoz hozzáadjuk a "A két szám..." szöveget majd az eredményt is!

De mi történik akkor, ha a felhasználó nem számot ad meg? Az is érdekes eredményt ad, hogy ha osztani akarunk és a felhasználó az osztónak 0-at ad meg. A megoldáshoz elágazást, "magyarul" szelekciót fogunk használni!

Elágazás

Szelekcióhoz (elágazás) már olyan utasításokat használunk, amelyek valamilyen logikai kifejezés értékétől függően eltérő utasításblokkot hajtanak végre. Megvalósításához használhatjuk az if (logikai kifejezés){ } utasítást. Nézzük előbb egy egyszerű példát!

```
<!DOCTYPE html>
<html>
<head>
<title>Elágazás</title>
<meta charset="UTF-8">
</head>
<body>
<script>
     var a = prompt("Kérek egy számot!");
     var b = prompt("Kérek egy másik számot!");
```

```
     document.write("<h1>"+ a + " vagy " + b + " a nagyobb?" + "</h1><br
/>");
     if (a>b) {
       document.write(a+" nagyobb mint " + b);
     }
     else if (b>a) {
            document.write(b+" nagyobb mint " + a);
          }
          else {
            document.write(a+ " egyenlő"+ b + "<br />");
         }
</script>
</body>
</html>
```

Figyeljétek meg, hogy az igaz és a hamis ágakban is blokkba tesszük az utasításokat, azaz {} közé zárjuk. A hamis ágba a példánkban egy másik if-et tettük.

Egy megjegyzés a stílusról: az igaz és a hamis ág utasításait is beljebb szokás írni. Nem kötelező, de így szebb, átláthatóbb a kód. A blokkot jelentő kezdő kapcsos zárójelet vagy az if-fel és az else-el egy sorba szokták írni vagy a következő sorba. Az előbbit ajánlják Java kódok írására. Mi keverni fogjuk a kettőt.:-)

Nézzük most meg a korábbi példánkat, de most már figyelünk a hibás adatokra!

```
<!DOCTYPE html>
<html>
<body>
<script>
    document.write('Hello, user'+"<br>");
    var a=prompt("Kérek egy számot");
    if (isFinite(a))  //ha szám
    {
        var b=prompt("Kérek még egy számot");
        if (isFinite(b) && b!=0) //ha szám és nem nulla
        {
            document.write("<strong>A két szám hányadosa: ");
            document.write(parseInt(a)/parseInt(b));
        }
        else
           document.write("<strong>Ez nem jó szám!");
    }
    else
        document.write("<strong>Ez nem szám!");
</script>
</body>
</html>
```

Az isFinite függvény igazat ad vissza, hogy a paraméterként kapott érték egy "valódi" szám. (Így teszteljük, hogy szöveget vagy számot adott meg a felhasználó.) Azt, hogy az osztó (b változó) értéke nem nulla, azt a != operátorral vizsgáltuk meg. Arra is van lehetőség, hogy a hibára egy felugró ablakkal figyelmeztessük a felhasználót. Erre szolgált az alert() vagy a window.alert() metódusok.

Feladatok

1. A Bitek világában az tanítás szigorúan életkorhoz van kötve. Mindenki pontosan 6 évesen kezdi az általános iskolát, pontosan 14 évesen a középiskolát, és pontosan 18 évesen érettségizik. Készítsetek egy programot, ami bekér egy életkort, és kiírja hogy azzal az életkorral egy bitdiák az iskola előtt jár, az általános iskolában, a középiskolában, vagy a középiskola után. (10100|2 pont)

2. Készítsetek egy programot, amely 3 bekért szám közül meghatározza a legtöbb számjegyből állót, ha több ilyen is van, akkor azok közül a legkisebbet! (11110|2 pont)

3. Kérjétek be egy háromszög oldalainak hosszát, majd írassátok ki, hogy tompa-, derék-, vagy hegyesszögű! Ha az c^2 -a^2 -b^2 összefüggés egyenlő nulláva, akkor derékszögű, ha nagyobb nullánál, akkor tompaszögű, és ha kisebb nullánál, akkor hegyesszögű. (10100|2 pont)

4. Készítsetek egy programot, amely beolvas 4 számot, majd az összes lehetséges módon vonja ki egymásból a számokat, és jelenítse meg az eredményt (12 lehetőség lesz). Pl. ha a számok 1, 3, 6, és 10, akkor ilyen sorok jelenjenek meg (10100|2 pont):
1-3=-2
3-1=2
10-3=7
3-10=-7
...

5. A kodolosuli.hu kezdőoldalán találtok egy animációt (ha mégsem, akkor F5), amelyben egy lézerágyú tüzel az ellenséges űrhajókra, ha azok hatótávolságon belül vannak. Készítsetek olyan programot, amely az ágyú kör alakú hatótávolságának ismeretében megmondja két bekért számról, hogy ha azok egy érkező űrhajó aktuális koordinátái, akkor az űrhajó hatótávolságon belül van vagy kívül. Legyen az ágyú az origóban! Példa: ha a hatótávolság 10, az ellenséges űrhajó koordinátái: 7 és 6, akkor igen, mivel 7*7+6*6=85 <=100. De ha az aktuális koordináta 8 és 7, akkor 64+49 >=100, tehát nem érdemes rá lőni. A program szemléltesse is egy rajzzal és felirattal az aktuális esetet! (1111|2 pont)

Iteráció

Az iteráció (ismétlődés, ciklus) teszi lehetővé bizonyos feltételek teljesülése esetén egy utasításblokk többszöri végrehajtását. Ilyenkor a ciklusmagban használt változók értéke a feladattól függően folyamatosan változhat. Awhile (logikai kifejezés){ } elöltesztelő ciklusnál az ismétlési feltételeket a ciklus mag előtt allítjuk be, míg a do { } while (logikai kifejezés) hátultesztelő ciklus esetén a ciklusmag utasításblokkját követően. Létezik még az úgynevezett növekményes (előírt lépésszámú) ciklusszervező utasítás, a for (ismétlések) { } . Itt az ismétlések tényleges számát már a ciklus indulásakor meg kell adnunk. Figyeljétek meg az alábbi mintaprogramban az utóbbi felépítését!

```
<!DOCTYPE html>
<html>
<head>
<title>Vezérlőszerkezetek</title>
<meta charset="UTF-8">
</head>
<body>
```

```
<script>
    var a=1;
    var b=12;
    document.write("<h1>Egész számok "+a+" és "+b+" között</h1>");
    for (i=a;i<=b;i++) {
       document.write(i + ", ");
    }
    document.write("<br />");
    document.write("<h1>Páros számok "+a+" és "+b+" között</h1>");
    var s = ''; // üres, karakter típusú változó
    for (i=1; i <= 100; ++i)
    {
       s += '<tr><td>'+i+'</td><td>'+i*i+'</td><td>'+Math.sqrt(i)+
'</td></tr>';
    }
    document.write('<table border="1px">',s,'<table>');

</script>
</body>
</html>
```

Az első megoldásban az i kezdőértéke az a változó kezdőértékét veszi fel. Aztán lesz egy tesztelés, az i értéke kisebb vagy egyenlő mint a b értéke? Ha igen, akkor a ciklus belsejében folytatódik a végrehajtás, azaz megtörténik az i értékének a kiírása. Aztán vissza a léptetésre, azaz i++. Ez azt jelenti, hogy i értéke eggyel nő (mint a tét:). Majd ismét teszt: i értéke kisebb vagy egyenlő mint b értéke? Ha igen, akkor ismét a kiírás következik, majd léptetés. Ha a feltétel nem lesz igaz, akkor fejeződik be a ciklus végrehajtása. A második ciklusban pedig kiírjuk az első 100 szám négyzetgyökét, melyet mindig a Math.sqrt() függvény ad vissza.

Egyébként az i+=2 2-vel növelni az i értékét, az i+=3 pedig hárommal, stb. Az i-- pedig eggyel csökkentené, az i-=2 pedig kettővel csökkentené az i változó értékét.

Feladatok

6. Készítsetek egy programot, amely bekér egy számot, majd megjelenít minden 400-nál kisebb számot, ami osztható a bekért számmal!(1111|2 pont)
7. Készítsetek programot, amely beolvas két számot, majd kiírja a kettő között az összes páros, egész számot, de visszafelé. Ehhez jól jöhet a -- operátor! (pl. i--) (1111|2 pont)
8. Készítsetek olyan programot, amely 3 (a, b, c) számot olvas be, majd az első kettő szám (a,b) között megtaláltató minden c-edik számot írja ki. Például: a=10, b=20, c=3, akkor 13, 16, 19.(1111|2 pont)

A következő két kód a másik két ciklus működését mutatja be.

```
<!DOCTYPE html>
<html>
<head>
<title>Vezérlőszerkezetek</title>
<meta charset="UTF-8">
</head>
<body>
<script>
   var a=10;
   var b=20;
```

```
   var j=a;
   document.write("<br />");
   document.write("<h1>Páratlan számok "+a+" és "+b+" között</h1>");
   while(j<=b) {
     if(j%2==1) {
       document.write(j + ", ");
     }
     j++;
   }
   var n; // a változónak még nincs értéke (így típusa sincs)
   while (!isFinite(n)) { // addig ismétlődjön a ciklus, amíg "n" nem szám
típusú
           n = prompt('Kérek egy számot','0');
   }
   document.write('A megadott szám=', n, "<br>");
   var n=42; // a változónak VAN értéke és az szám típusú
   do { // ciklus kezdete. Nincs feltétel.
      n = prompt('Kérek egy számot','0');
   } while (!isFinite(n)); // a végén értékel, hogy kell-e ismételnie
   document.write('A megadott szám=', n, "<br>");
   var j=a;
   document.write("<br />");
   document.write("<h1>Páratlan számok "+a+" és "+b+" között</h1>");
   do {
     if(j%2!=0) {
       document.write(j + ", ");
    }
    j++;
   }  while(j<=b)
   document.write("<br />");

</script>
</body>
</html>
```

A while ciklus addig hatja végre a {} közötti utasításokat, azaz a ciklusmagot, amíg a feltétel igaz. Ha már a belépés előtt sem igaz, akkor egyszer sem hatjódik végre a ciklusmag. A do while pedig a végén tesztel, azaz 1x biztosan végrehajtja a ciklusmagot. Az első esetben tehát a vizsgálat: j<=b? Ha igen, akkor jöhet a ciklusmag végrehajtás. Ha nem igaz, akkor a ciklus utáni (var n) utasítás hajtódik végre.

Gondoltam egy számot! A gép kitalálja, hogy melyik volt az. Figyeljétek meg a megoldást!

```
<!DOCTYPE html>
<html>
<head>
<title>Vezérlőszerkezetek</title>
<meta charset="UTF-8">
<script>
       function myFunction() {
          var numberComputer = Math.floor((Math.random()*10)+1);
          var num = prompt("Gondoltam egy számra egy és tíz között. Találd
ki!");
          var ok = true;
          while(ok) {
             if(isNaN(num)){
                num = prompt("Csak számot adhatsz meg és legyen egy és tíz
között!");
```

```
            }
            else if(num < numberComputer) {
                  num = prompt("Nagyobbra gondoltam!");
                }
                else if(num > numberComputer) {
                    num = prompt("Kisebbre gondoltam!");
                  }
                  else if(num == numberComputer) {
                      alert("Igen, erre a számra gondoltam: " +
numberComputer);
                      ok = false;
                  }
               }
          }
   </script>
</head>
<body>
<h1>Gondoltam egy számot!</h1>
<button type="button" onclick="myFunction()">Találd ki!</button>
<p id="demo"></p>
</body>
</html>
```

A kódban most az isNaN függvénnyel vizsgáltuk meg, hogy számot adott-e meg a felhasználó. A kód többször is futtatható a gombra kattintással. A megoldást az alert() jeleníti majd meg. A véletlenszámot pedig a Math.random metódussal generáltatjuk, de ez egy 0-1 közötti valós számot generál, ezért "kicsit besegítünk" neki, hogy 1-10 közé essen.:-)

Feladatok

9. A háromszög-számokat úgy lehet generálni, hogy elindulunk a 0-tól, hozzáadunk 1-et, majd minden alkalommal eggyel többel növeljük a számot, mint azelőtt (1-gyel, majd 2-vel, majd 3-mal, stb.). Így a következő sorozat jön ki: 0, 1, 3, 6, 10, 15, 21, 28, ... Készítsetek egy programot, ami felsorolja az 1000-nél kisebb háromszög-számokat. (100011|2)

Interaktív grafika

Térjünk vissza az első forduló utolsó témájához! Mi lenne, ha olyan köröket kellene rajzolni, amelynek a helyét és a sugarát a felhasználó adja meg?

```
<!DOCTYPE html>
<html>
<body>
<canvas id="vaszon" width="400" height="400" style="border:1px solid
#c3c3c3;">
</canvas>
<script>
   function rajzol()
   {

     var c = document.getElementById("vaszon");
     var ctx = c.getContext("2d");
```

```
     var x=document.getElementById("x").value;
     var y=document.getElementById("y").value;
     var r=document.getElementById("r").value;
     ctx.beginPath();
     ctx.arc(x,y,r,0,2*Math.PI);
     ctx.stroke();
    }
    function torles()
    {
         var c = document.getElementById("vaszon");
         var ctx = c.getContext("2d");
         ctx.clearRect(0, 0, c.width, c.height);
    }
</script>
<p id="eredmeny">Ide kerül az eredmény</p>
<form>
  X: <input type="text" id="x"> Y: <input type="text" id="y"> R: <input
type="text" id="r"></br>
     <input type="button" value="Rajzolás" onclick="rajzol()">
     <input type="button" value="Törlés" onclick="torles()">
</form>
</body>
</html>
```

Figyeljétek meg a kódot! Három beviteli mezőnk van, az id-vel azonosítottuk őket. Két gombunk van, az egyik a rajzol() a másik a torles() metódust hívja meg. A rajzol metódusban lekérdezzük a beviteli mezők tartalmát. Ehhez kell a value, azaz pl. a weboldalon lévő x elem értékét tároljuk az x változóba. A torles() metódusban a vásznat töröljük teljes szélességében és magasságában. Ha az eljárás utolsó sorát áttenénk a rajzol() 2. sora után, akkor minden rajzolás során törölnénk a vászon tartalmát, azaz mindig csak az új kör látszódnak.

Feladatok

10. Egy informatikus szobájában biztosan van hűtő, viszont "szerencsétekre" általban nem sok minden van benne:, de kóla, sör, vaj, dobozban süti biztosan. :-) Rajzoljátok meg alakzatok segítségével a hűtő elrendezését úgy, hogy azonosíthatóak legyenek benne az "ételek" és italok! Távirányító (gombok) segítségével ki lehessen nyitni a hűtőt. Ekkor lehessen látni a tartalmát! (1 1110|2 pont)

11. A korábbi olimpiai ötkarikás programotokat kell módosítani. Legyen az oldalon 5 gomb, a gombok színe egy-egy karika színével egyezzen meg! Amelyik gombra kattintunk, az a karika ne jelenjen meg az oldalon. Ha megint kattintunk a gombra, akkor a hozzá tartozó karika ismét legyen kirajzolva. (111100||2)