Tartalom

Eljárások

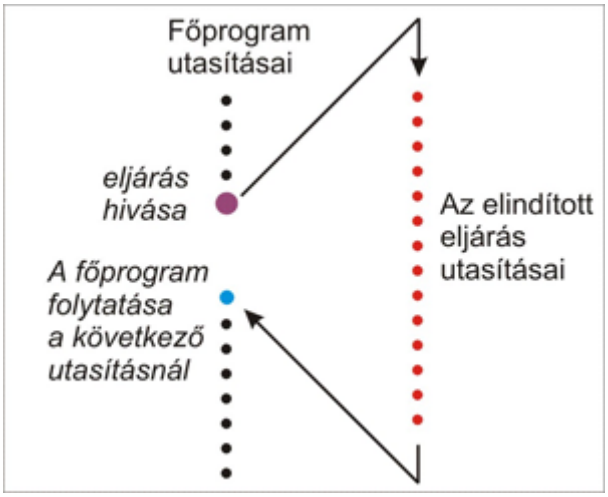
A programozás módszere, hogy bontsuk le a problémát apróbb, kezelhető részekre, majd oldjuk meg a részeket, végül rakjuk össze belőle a teljes megoldást.

A részekre bontás egyik módja az alprogramok kialakítása. Alprogramoknak nevezzük azokat az önálló feladattal rendelkező kódrészeket, melyeknek nevük (és esetleg paramétereik) vannak. Az alprogramban lévő kód futását az alprogram nevének segítségével kell kezdeményezni.

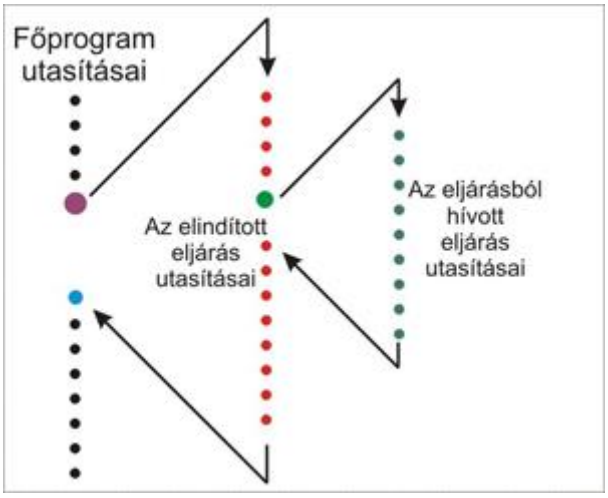
Eljárás elemei

- jelölni kell őket a **function** kulcsszóval
- van neve (azonosító),
- lehetnek paraméterei,
- van törzse, utasítások melyek lefutnak a meghívásakor
- futtatásához, meg kell hívni! (erről később)

Eljárások lefutásának menete



Természetesen van rá lehetőség az eljárásokból is további eljárásokat hívni:



Eljárások meghívása

Az eljárás meghívása igen könnyű, nem kel mást tennünk, mint megadni, annak nevét, valamint két kerek zárójellel jelezni, hogy a név, amit megadunk valójában egy eljárás.

```
function EljarasNev();
```

```
function IloveProgramming()
{
        document.write("Szeretem a programozás");
}
IloveProgramming();
```

Amennyiben nem hívjuk meg az eljárást, úgy az egyszerűen nem fog lefutni, csak készen áll egy „lehetséges későbbi használatra”.

Paraméterek kezelése

Amennyiben szeretnénk, hogy az eljárásunk használjon, bizonyos általunk megadott értékeket a futtatása során, úgy adhatunk meg ilyeneket is, paraméter formájában.

A paramétereket a korábban látott forma esetén a kerek zárójelekbe helyezzük el „()”.

Ezután az itt megadott értékeket az eljárás csak és kizárólag a futása közben felhasználhatja, a későbbiekben ezek nem használhatók fel, másik eljárásnál, hatáskörük, csak az eljárás programkódjára terjed ki, élettartamuk addig tart amíg futtatjuk az eljárásunkat.

```
function EljarasNev( elsoParaméter, masodikParaméter);
```

Természetesen a paraméterek száma tetszőleges, ha akarjuk egyet sem, ha szeretnénk, akár kettőt vagy tízet is megadhatunk!

```
function Osszead(a, b)
{
        let osszeg=a+b;
        document.write(`A két szám összege: ${osszeg}`);
}
Osszead(3,4);
```

Függvények

A függvény rokon fogalom az eljárással – egy apró különbséggel. A függvény egy olyan eljárás, amely olyan részfeladatot old meg, melynek pontosan egy végeredménye is van – egy érték.

Amennyiben az a részfeladat, hogy egy értékekkel már feltöltött tömb eleminek összegét kell kiszámítani, a végeredmény egyetlen számérték. Amennyiben ezt eljárás segítségével oldjuk meg, úgy a végeredményt egy megosztott változóba kell elhelyezni. Ez nem túl szerencsés megoldás, mert a függvény kötődik ezen megosztott változó nevéhez. Amennyiben egy másik programba szeretnénk ezt függvényt áthelyezni, úgy vele együtt kell mozgatni a megosztott változót is.

Megosztott változó, ami a kód bármely részéből elérhető, pl.: var, vagy const típusú

Az ilyen jellegű feladatokat megoldó alprogramokat függvények formájában írjuk meg.

Amennyiben függvényt akarunk írni, két NAGYON fontos dolgot kell szem előtt tartanunk:

- A függvények ezek után kötelesek minden esetben egy értéket vissza is adni! A függvény visszatérési értékét a **return** kulcsszó után írt **kifejezésben** kell feltüntetni.
- A függvények önmagukban csak egy értéket állítanak elő, a későbbi felhasználáshoz azt le kell tárolni egy változóba, a megjelenítéshez pedig szükségünk van egy megjelenítő parancs használatára.

Paraméter nélküli függvények

```
function RandomEgesz100()
{
	let generaltSzam=Math.round(Math.random()*100);
	return generaltSzam;
}
document.write(RandomEgesz100());
```

Paraméteres függvények

A függvényeknél ugyanúgy használhatunk paramétereket, mint az eljárások esetében!

```
function RandomEgesz(alsoHatar, felsoHatar)
{
	let generaltSzam=Math.round(Math.random()*(felsoHatar-alsoHatar))+alsoHatar;
	return generaltSzam;
}
document.write(RandomEgesz(1,50));
```

ArrowFunction

A függvényeket egy új feature-val van lehetőségünk leegyszerűsíteni, erre az egyszerűsítésre az „ArrowFunction”-t használjuk, a következőképpen.

A következő függvényt dupla néven hozzuk létre szám paramétert vár, és visszatér annak kétszeresével.

```
let dupla = function (szam){
        return szam*2;
}
```

Egyszerűsítésre „használjuk a => „arrow function jelet

```
let duplaV2 = (szam) =>{
        return szam*2;
}
```

Egysoros utasítás blokk esetén a zárójelek és a return kulcsszó is elhagyhatók…

```
let duplaV3 = (szam) => szam*2;
```

Egy paraméter esetén a paraméterek kerek zárójele is elhagyható!

```
let duplaV4 = szam => szam*2;
```

Rekurzió

A rekurzió egy programozási módszer. Azt az esetet nevezzük így, amikor egy eljárásban szerepló kód **önmagát** (ugyanazt az eljárást) hívja meg.

Faktoriáls JavaScript függvénnyel:

```
function fakt(n)
{
  if(n==0)
  {
        return 1;
  }
  else
  {
        return n * fakt( n - 1 );
  }
}
```

Callback

A Callback egy opcionálisan megadható paraméter, mellyel meghatározhatod, mi történjen az adott függvényed végrehajtását, követően!

Példa a megadására:

```
function udvozlo(nev) {
  alert ('Hello ' + nev);
}

function adatFeldolgozas(callback) {
  var nev = prompt ('Kérlek add meg a neved.');
  callback(nev);
}

adatFeldolgozas (udvozlo);
```

Ebben az esetben létezik a az **udvozlo()** és az **adatFeldolgozas()** függvényünk, és az **adatFeldogozas()** után közvetlenül végre hajtja a paraméterként megadott függvényen, 1 paraméter használatával a belső függvény lefuttatását.

Források

Function parameters(W3Schools)

ArrowFunction(Mozilla Developer)

ArrowFunction(W3Schools)

Prog Wiki(Rekurzió)

Callback(W3Schools)

Callback(Mozilla Developer)