

# Ötödik fejezet

- Függvények és eljárások
- Eljárások (szubrutinok) deklarációja
- Függvények
- Változók élettartama
- Kiszolgálóoldali beágyazások (Server Side Include, SSI)
- A VBScript beépített függvényei
- A felhasználói adatbevitel elemei: A HTML-űrlapok (FORM-ok)

## *Függvények és eljárások*

A függvények és eljárások olyan scriptutasítások csoportját jelentik, amelyek a program többi részétől jól elkülöníthetők és névvel hivatkozhatunk rájuk. Ezt a hivatkozást *eljárás* illetve *függvényhívás*nak nevezzük. A hivatkozás tehát nem más, mint az eljárás vagy a függvény utasításainak végrehajtását jelenti minden egyes helyen, ahol azt meghívtuk. Függvények és eljárások használatával lehetőségünk nyílik ismétlődő programrészletek megadására. Ezért bárhol, ahol ezekre a scriptutasításokra szükség van, csak a függvény vagy az eljárás nevét kell ismételtelen megadnunk.

***Az eljárás és függvény közti különbség:** Az eljárás abban különbözik a függvénytől, hogy nincsen visszatérési értéke. Akkor használunk függvényeket, ha olyan eljárásra van szükségünk, amelynek valamilyen visszatérési értékkel kell rendelkeznie. Ha egy ASP oldal nem tartalmaz függvényt vagy eljárást, az ASP parancsvégrehajtója egyszerűen feldolgozza a fájlt az elejétől a végéig. A függvényeket és eljárásokat a program csak akkor hajtja végre, ha az adott eljárást vagy függvényt meghívjuk.*

Ha egy függvénynek vagy eljárásnak bemeneti információkat szeretnénk adni, akkor azt paraméterként kell megadnunk, úgy, hogy a változót a függvény vagy az eljárás neve után zárójelbe írjuk. Ha több bemeneti paraméterrel is rendelkezik egy eljárás vagy függvény, akkor a változókat egymástól vesszővel elválasztva kell megadnunk.

## *Eljárások (szubrutinok) deklarálása*

Ejrásokat a következőképpen definiálhatunk:

**Sub** eljárásnév ([paraméterek])  
utasítások  
[feltétel] [**Exit Sub**]  
**End Sub**

Az eljárást a **Call eljárásnév** utasítással, vagy csak egyszerűen az eljárás nevével lehet meghívni. Ha valamilyen esemény hatására az eljárás végrehajtását szeretnénk félbeszakítani, akkor azt az **Exit Sub** paranccsal tehetjük meg.

*Lássunk egy egyszerű példát eljárásra. (sub.asp)*

```
<BODY>
<HTML>
<%
Dim txt, i
txt = "Helló!"

'Első eljárás definiálása
Sub Write()
  Response.Write txt & "<br>"
End Sub

'Második eljárás definiálása
Sub Repeat()
For i=1 to 5
  Write 'Write eljárás hívása
Next
End Sub

'Eljárás meghívása
Repeat()
%>
</BODY>
</HTML>
```

A példaprogramban két eljárást definiáltunk. Az első eljárásnak mindössze az a feladata, hogy a "Helló!" szöveget kiírja és sort emeljen. A második eljárás pedig meghívja ötször a kiíró (Write) eljárást.

## *Függvények*

**Function** függvénynév ([paraméterek])

utasítások

[feltétel] [**Exit Function**]

függvénynév=érték

**End Function**

Ahogy korábban említettem, a függvény abban különbözik az eljárástól, hogy van visszatérési értéke. A függvény visszatérési értékét úgy tudjuk definiálni, hogy a függvényen belül valahol egyenlővé tesszük a függvénynevet valamilyen változóval vagy más függvény visszatérési értékével. (Ez az érték bármilyen típusú lehet.)

*Nézzünk egy példát függvényhívásra: (**func.asp**)*

```
<BODY>
<HTML>
<%
Dim a, b

'Fgv. definiálása
Function Calculate(a,b)
    Calculate = a + b
End Function

'Fgv. hívása bemenő paraméterekkel
Response.Write Calculate(5,10)
%>
</BODY>
</HTML>
```

*Bonyolítsuk a feladatot egy harmadik paraméter megadásával: (**func2.asp**)*

```
<HTML>
<BODY>
<%
'Fgv. definiálása három paraméterrel
Function Calculate(a,b,op)
```

```

Select Case op
  Case "+"
    Calculate=a+b
  Case "-"
    Calculate=a-b
  Case "*"
    Calculate=a*b
  Case "/"
    Calculate=a/b
  Case Else
    Response.Write ("Nem értelmezhető műveleti jel!")
End Select
End Function
%>
A és B Összege: <%=Calculate(10, 5,"+")%><br>
A és B Különbsége: <%=Calculate(10, 5,"-")%><br>
A és B Szorzata: <%=Calculate(10, 5,"*")%><br>
A és B Hányadosa: <%=Calculate(10, 5, "/" )%><br>
</BODY>
</HTML>

```

## Változók élettartama

A változók élettartalmát tekintve kétféle típust tudunk megkülönböztetni: **Globális változókat** a függvényen vagy eljáráson kívül kell deklarálni. Általában ezeket a script vagy utasításcsoport elején szokták a könnyebb áttekinthetőség miatt. **Lokális változókat** egy-egy függvényen vagy eljáráson belül alkalmazunk. Ezekre a változókra csak a függvényen illetve eljáráson belül hivatkozhatunk, ezért ezek élettartalma csak az eljárás vagy függvény végéig tart.

A globális változók értékét bármely ASP-lapon lévő scriptparancs használhatja vagy módosíthatja. Ha egy scriptben globális és lokális változókat is használunk, figyeljünk arra, hogy egy globális változó neve ne egyezzen meg egy lokális változó nevével. Két külön függvényben vagy eljárásban már megengedett két azonos nevű változó használata.

**Megjegyzés:** a változó hatókörének az adott eljárásra történő korlátozásával javul a teljesítmény, ezért érdemes használni.

## *Kiszolgálóoldali beágyazások (Server Side Include, SSI)*

Az ASP-ben történő fejlesztéseknél a kódok modularitása és újrafelhasználhatósága döntő fontosságú lehet. Ha minden .html vagy .asp lapban szeretnénk azonos szövegfájlt, képeket vagy függvényeket használni, akkor az ASP biztosítja a kiszolgálóoldali beágyazás lehetőségét. Ilyenkor ha valamit módosítani szeretnénk, akkor azt elegendő egyetlen fájlban megtennünk, és nem szükséges minden egyes lapot külön-külön kijavítanunk.

A fájlbeágyazás szintaxisa a következő:

```
<!-- #include elérési_út_típusa="fájlnév" -->
```

Az *elérési\_út\_típusa* paraméter egy kulcsszót tartalmaz, ami a FILE vagy VIRTUAL lehet.

**FILE** esetében az *include* direktívát tartalmazó dokumentumot tároló könyvtár relatív elérési útját tartalmazza. Ilyenkor a beágyazott fájl csak a vele azonos könyvtárban vagy egy alkönyvtárban lehet, de nem lehet az *include* direktívát tartalmazó oldalnál magasabb szintű könyvtárban.

**VIRTUAL** esetében a fájl a webhely virtuális könyvtárában lévő teljes virtuális elérési utat jelenti. A beágyazandó fájloknak elvileg bármilyen kiterjesztést lehet adni, de tanácsos az **.inc** kiterjesztést használni.

Nézzünk mindkét beágyazási módszerre egy-egy példát.

A beágyazott fájl a szülőfajllal azonos könyvtárban található:

```
<!-- #include file = "adat.inc" -->
```

A beágyazott fájl a *data* virtuális könyvtárban található:

```
<!-- #include virtual = "/data/adat.inc" -->
```

Figyeljünk arra, hogyha egy ASP-script által beágyazott fájl nagy mennyiségben tartalmaz olyan függvényeket és változókat, amelyeket a beágyazó script éppen nem használ, akkor a nem használt szerkezetek által lefoglalt erőforrástöbblet hátrányosan befolyásolhatja a teljesítményt. Ezért tanácsos a beágyazandó fájlokat felbontani kisebb fájlokra, és csak azokat beágyazni, amelyekre a kiszolgálóoldali scriptnek valóban szüksége van.

Esetenként szükséges lehet egy kiszolgálóoldali script beágyazása a <SCRIPT>..</SCRIPT> HTML-kódok segítségével. Az alábbi script például olyan fájlt ágyaz be (relatív elérési út segítségével), amit a kiszolgáló hajthat végre:

```
<SCRIPT LANGUAGE="VBScript" runat="SERVER"
src="utils\data.inc"></SCRIPT>
```

### *A VBScript beépített függvényei*

A programozási munka megkönnyítése céljából a VBScript előre "beépített" függvényeket bocsát a rendelkezésünkre, amelyek közvetlenül használhatóak általános programozási feladatokhoz. A függvények többsége hasonlít más programozási nyelvek standard függvénytáráiban megtalálható függvényekhez. Ilyen beépített függvényekkel már találkoztunk az előző fejezetekben. (pl. *now*, *date*, *formatdatetime*).

Az alábbi táblázat további hasznos beépített függvényeket sorol fel.

<b>Függvény neve</b>	<b>Feladata</b>
Abs (szám)	A kifejezés abszolút értékét adja vissza.
Asc (sztring)	A karakterlánc első karakterének ASCII kódját adja vissza.
Atn (szám)	A kifejezés árkusz tangensét adja vissza.
cBool (szám)	Boolean értéké alakít át egy változót.
cByte (szám)	Egy 0..255 tartományban lévő számot bájtértékké alakít át.

cDbl (szám)	Egy számot duplapontosságú lebegőpontos értéké alakít át.
Chr (karakter)	Egy karakter ASCII kódját adja vissza.
cInt (szám)	Számot integer (egész) értéké alakít át.
CLng (szám)	Számot hosszú integerré alakít át.
Cos (szám)	Radiánban megadott szög koszinuszát adja vissza.
cSng (szám)	Egy számot egyszeres pontosságú lebegőpontos értéké alakít át.
cStr (szám)	Sztring formátummá konvertál egy számot.
DateSerial (év,hó,nap)	Az aktuális dátumhoz viszonyított dátumot adja vissza.
DateValue (dátum)	Olyan formátumba alakítja át a dátumot, amely az összehasonlításoknál jobban használható.
Erase	Törli egy tömb tartalmát.
Fix (szám)	Visszatér egy szám egész értékével.
Hex (szám)	Hexadecimális formátumba alakít egy számot.
Int (szám)	Szám egész részét adja vissza.
inStr ([x],s1,s2 [hasonlítás])	Visszatér a második string előfordulási pozíciójával. s1: Egész karakterlánc, amelyben keresünk (kötelező megadni) s2: Keresendő karakterlánc (kötelező megadni) x: Keresés kezdő pozíciója (opcionális) hasonlítás: Egy számérték (opcionális). Ha 0, akkor bináris összehasonlítás, ha 1, akkor szövegszintű összehasonlítás.
isArray (változó)	Igaz értéket ad vissza, ha az adott változó tömb. Ellenkező esetben hamis értékkel tér vissza.
isDate (változó)	Visszatér egy logikai változóval, amely igaz, ha a változó dátum típusú.
isEmpty (változó)	Igaz értéket ad vissza, ha az adott változó még nem lett inicializálva, ellenkező esetben hamis lesz az értéke

isNull (változó)	Igaz értéket ad vissza, ha az adott változó értéke nulla, ellenkező esetben hamis értékkel tér vissza.
isNumeric (változó)	Igaz értéket ad vissza, ha az adott változó numerikus értéket tartalmaz, ellenkező esetben hamis értékkel tér vissza.
isObject (változó)	Igaz értéket ad vissza, ha az adott változó objektumnak felel meg.
Join (tömb)	Egy tömbben szereplő adatokat fűz egybe.
Lcase (sztring)	Kisbetűsre alakít egy karakterláncot.
Left (sztring, n)	Karakterlánc részláncát adja vissza; a részlánc a karakter bal szélén kezdődik és <i>n</i> karakterből áll.
Len (sztring)	Visszatér a sztring hosszával.
Ltrim (sztring)	Levágja egy karakterlánc bal széléről a vezető szóközöket, és az így kapott részláncot adja vissza.
Mid (sztring, start [hossz])	A karakterlánc belsejében lévő részláncot adja vissza. <i>sztring</i> : Az egész karakterlánc (kötelező megadni) <i>start</i> : Egy szám, honnantól kezdve (kötelező megadni) <i>hossz</i> : Egy szám (opcionális), milyen hosszú karakterláncot adjon vissza.
Oct (szám)	Oktális formátumba alakít egy karakterláncot.
Right (sztring, n)	Karakterlánc részláncát adja vissza; a részlánc a karakterlánc job szélén kezdődik és <i>n</i> karakterből áll.
Rtrim (sztring)	Levágja egy karakterlánc jobb széléről a vezető szóközöket, és az így kapott részláncot adja vissza
Round (szám)	A kapott számot tizedes egész számmá alakítja.
Sgn (szám)	Adott érték előjelét adja vissza. (-1, ha negatív, 0 ha nulla, és 1, ha pozitív)
Sin (szám)	Radiánban megadott szög szinuszát adja vissza.

Space (szám)	Visszatér n darab space (szóköz) karakterből álló sztringgel.
Sqr (szám)	Adott érték négyzetgyökét adja vissza.
StrComp (s1, s2 [hasonlítás])	Két sztringet (s1 és s2) hasonlít össze. Visszatérő értéke: 1 ha $s1 < s2$ , 0 ha egyenlőek, -1 ha $s1 > s2$ <i>hasonlítás</i> : egy szám (opcionális), 0=bináris, 1=szöveg szintű összehasonlítás.
StrReverse (sztring)	Visszafelé írja ki a paraméterként megadott sztringet.
String (szám, karakter)	Olyan karakterláncot ad vissza, amely az adott karaktert n-szer tartalmazza.
Tan (szám)	Radiánban megadott szög tangensét tartalmazza.
TimeSerial (óra,perc,msp)	Az aktuális időhöz viszonyított időt adja vissza.
TimeValue (idő)	Olyan formátumba alakítja az időt, amely az összehasonlításoknál jobban kezelhető.
Trim (sztring)	Levágja egy karakterlánc elejétől és a végétől a szóközöket, és az így kapott karakterláncot adja vissza.
Ucase (sztring)	Nagybetűssé alakít egy sztringet.
VarType (változó)	A változó típusát jelző értéket adja vissza.

### *A felhasználói adatbevitel elemei: A HTML- űrlapok (FORM-ok)*

A HTML-űrlapok használata a weblapú információk bevitelének leggyakoribb módja. Olyan speciálisan elrendezett HTML-kódokat tartalmaznak, amelyek a felhasználói felület elemeit a weblapon jelenítik meg. Ezek az elemek (szövegmezők, gombok, jelölőnégyzetek stb.) teszik lehetővé a weblapok interaktív használatát, valamint adatok továbbítását a webkiszolgálónak. Ilyenkor a HTML-oldalon belül az egyes elemeket összefoglaljuk egy-egy FORM-ba és ezt küldjük el az adatfeldolgozást végző alkalmazásnak.

Az FORM (űrlap) leggyakrabban használt deklarációja a következő:

```
<FORM name="formnév" action="fájlnev/url"
method="POST/GET">
...
Elemek
...
</FORM>
```

Megfigyelhető, hogy három jellemzőt kell megadnunk egy űrlapnak:

1. **Name:** Ezzel a névvel hivatkozhatunk rá, amennyiben kliensoldali scriptekkel szeretnénk megváltoztatni az elemek értéket vagy tulajdonságait.
2. **Action:** Az Action kulcsszó után annak a fájlnek vagy URL-címnek a nevét kell megadni, amellyel szeretnénk feldolgoztatni az űrlap által küldött információ(ka)t.
3. **Method:** A method jellemző után definiálhatjuk, hogy milyen formátumban szeretnénk az action részben megadott fájlnek továbbítani az adatokat. Ilyenkor kétféle lehetőség közül választhatunk: **POST** vagy **GET**.

A POST esetében a **Request.Form("elemnév")** módszerrel, GET esetén a **Request.QueryString("elemnév")** módszerrel tudjuk elérni a továbbított információt. Utóbbi esetben a hivatkozási fájl neve kiegészül a formok elemeinek megfelelő értékeivel az alábbiak szerint.

```
adat.asp?elem1=érték1&elem2=érték2&...
```

Ha hosszú és összetett űrlapok webkiszolgálóhoz történő továbbítására a GET metódust alkalmazzuk, akkor az az adatok elvesztését okozhatja. Egyes webkiszolgálók ugyanis korlátozhatják az URL-lekérdezés karakterláncának hosszát, így a GET metódussal továbbított terjedelmes űrlapadatokat megcsönkíthatják. Ezért ha nagy mennyiségű adat továbbítása szükséges, akkor indokolt a POST metódust használata.

Nézzünk meg azokat az elemeket, amelyeket használhatunk a Form-on belül:

Form elemek:	Jelentésük:
<code>&lt;INPUT type="text" name="szoveg"&gt;</code>	Szöveges beviteli mező.
<code>&lt;TEXTAREA name="szovegdoboz" rows="magasság" cols="szélesség"&gt;&lt;/TEXTAREA&gt;</code>	Szövegdoboz, text típusú szövegek tárolására.
<code>&lt;INPUT type="hidden" name="rejtett"&gt;</code>	Rejtett beviteli mező.
<code>&lt;INPUT name="jelszo" type="password"&gt;</code>	Szöveges beviteli mező, de a bevitt szöveg karakterei helyett * (csillag) jelenik meg.
<code>&lt;INPUT name="ellenorzo" type="checkbox" [value="érték"]&gt;</code>	Ellenőrző mező, amely <b>on</b> értéket ad vissza, ha be van jelölve. Ha opcionálisan adunk meg értéket, akkor ezt az értéket adja vissza, ha a bekapcsolt állapotban van.
<code>&lt;INPUT name="valaszto" type="radio"&gt;</code>	Választógomb.
<code>&lt;SELECT name="valaszto" [size="érték"]&gt;&lt;OPTION [selected] value="érték1"&gt;név1 &lt;/OPTION&gt;&lt;OPTION value="érték2"&gt;név2&lt;/OPTION&gt;&gt;...&lt;/SELECT&gt;</code>	Legördülő választódoboz, ahol az <i>option</i> kifejezésekkel adhatunk újabb mezőt a meglévőekhez. A <i>value="érték<sub>n</sub>"</i> kerül elküldésre, ha valamelyik mezőt kiválasztottunk. Az <i>option</i> után írt <i>"név"</i> jelenik meg a dokumentumban. Az alapértelmezettként kiválasztott mezőt a <i>"selected"</i> kifejezés jelzi. Ha opcióként a <i>size</i> értékét is megadjuk, akkor beállítható, hogy hány sora legyen a választódoboznak.

<code>&lt;INPUT name="elkuld" type="submit"&gt;</code>	Egy Submit gomb, amely lenyomására az adatokat lehet elküldeni.
<code>&lt;INPUT name="torol" type="reset"&gt;</code>	Reset gomb, ha lenyomjuk minden kitörlődik, amit eddig az űrlapba írtunk.

*Készítsünk egy olyan űrlapot (**form.htm**), amely bekéri a felhasználó kereszt-és vezetéknévét és életkorát. Választógomb segítségével kérjük be a felhasználó nemét (férfi vagy nő), majd egy legördülő listából válasszuk ki a kedvenc színét. A Submit (küldés) gomb megnyomására, az adatokat küldjük el egy másik oldalnak (**form.asp**), amely kiírja a kapott adatokat. Az űrlapon legyen egy Reset (törlés) gomb is.*

Az **form.htm** forráskódja:

```
<HTML>
<BODY>
<FORM name="urlap" action="form.asp" method="POST">
<table border=1 cellpadding=0 cellspacing=0>
<tr>
<td>Vezetékeve:</td>
<td><input type="text" name="surname"></td></tr>
<tr>
<td>Keresztneve:</td>
<td><input type="text" name="firstname"></td></tr>
<tr>
<td>Kora:</td>
<td><input type="text" name="age"></td></tr>
<tr>
<td>Neme:</td>
<td>Férfi<input name="rd" type="radio" ↵
value="férfi">
<br>Nő<input name="rd" type="radio" ↵
value="nő">
</td>
</tr>
<tr>
<td>Érdeklődési köre:</td></tr>
```

```
<td>Számítógép<input name="cbox1" type="checkbox" ↵
    value="igen">
<br>Könyv<input name="cbox2" type="checkbox" ↵
    value="igen">
<br>Zene<input name="cbox3" type="checkbox" ↵
    value="igen">
<br>Autók<input name="cbox4" type="checkbox" ↵
    value="igen">
<br>Sport<input name="cbox5" type="checkbox" ↵
    value="igen">
</td>
</tr>
<tr>
<td>Kedvenc színe:</td>
<td><select name="colour">
    <option selected value="piros">Piros</option>
    <option value="kék">Kék</option>
    <option value="zöld">Zöld</option>
    <option value="sárga">Sárga</option>
    <option value="fekete">Fekete</option>
    <option value="fehér">Fehér</option>
</select></td>
</tr>
<tr>
<td>Megjegyzés:</td>
<td>
<textarea name="comment" rows="4" cols="20"> ↵
</textarea></td></tr>
<tr>
<td>Töröl:</td>
<td><input name="reset1" type="reset" ↵
    value=Reset (Töröl) ></td></tr>
<tr>
<td>Elküld:</td>
<td><input name="submit1" type="submit" ↵
    value=Submit (Elküld) ></td></tr>
</table>
</FORM>
</BODY>
</HTML>
```

**A form.asp forráskódja:**

```
<HTML>
<BODY>
Az elküldött értékek: <br>
<%
'A Request.Form() utasítással elkérjük a
'megfelelő form-elemtől az értéket és kiírjuk

Response.Write "Vezetékneve: " &
Request.Form("surname")
Response.Write "<br>Keresztneve: " &
Request.Form("firstname")
Response.Write "<br>Kora: " & Request.Form("age")
Response.Write "<br>Neme: " & Request.Form("rd")
Response.Write "<br>Számítógép: " & ↵
Request.Form("cbox1")
Response.Write "<br>Könyv: " & Request.Form("cbox2")
Response.Write "<br>Zene: " & Request.Form("cbox3")
Response.Write "<br>Autók: " & Request.Form("cbox4")
Response.Write "<br>Sport: " & Request.Form("cbox5")
Response.Write "<br>Színe: " & Request.Form("colour")
Response.Write "<br>Megjegyzés: " & ↵
Request.Form("comment")
%>
</BODY>
</HTML>
```

Az előző példában a POST metódust alkalmaztunk az adatok elküldésére. Amennyiben GET-et használtuk volna (*form2.htm*, *form2.asp*), csak annyiban különbözne a forráskód, hogy a Request.Form helyett Request.QueryString szerepelne. Ilyenkor a webszerver az alábbi URL-kérést kapná:

```
http://localhost/form2.asp?surname=Nagy&
firstname=Béla&age=40&rd=férfi.....
```

*Készítsünk most egy olyan HTML űrlapot (**emailcheck.htm**), amely bekér egy email-címet, majd elküldi ezt egy másik asp oldalnak (**emailcheck.asp**), amely ellenőrzi az email-cím formátumának helyességét. Ha nem felel meg a követelményeknek, akkor arról küldjön egy hibaüzenetet.*

### **emailcheck.htm**

```
<HTML>
<BODY>
<FORM action="emailcheck.asp" method="POST">
  Kérem az Email-címet:<br>
  <input type="text" name="email" size="30"></input>
  <input type="submit" value="Ellenőriz"></input>
  <input type="reset" value="Töröl"></input>
</FORM>
</HTML>
</BODY>
```

### **emailcheck.asp**

```
<%
mail = Request.Form("email")
If mail = "" Or Not IsValid(mail) Then
  Response.Write ("HELYTELEN az email-cím ↪
                    formátuma.")
Else
  Response.Write "Helyes az email-cím formátuma."
End If
%>
<%
' Email-cím ellenőrző fgv.
Function IsValid(mailvalid)
Dim valid
valid = True
'Email-cím formátumának ellenőrzése
'Nem lehet 5 karakternél rövidebb
If Len(mailvalid) < 5 Then
  valid = False
Else
  If Instr(1, mailvalid, " ") <> 0 Then
```

```
valid = False
Else
  ' @ nem lehet az első karakter
  If InStr(1, mailvalid, "@", 1) < 2 Then
    valid = False
  Else
    'Pont nem lehet közvetlen a @ után
    If InStrRev(mailvalid, ".") < InStr(1, ↵
      mailvalid, "@", 1) + 2 Then
      valid = False
    End If
  End If
End If
IsValid = valid
End Function
%>
```

