Ellenőrzések és számolások megadása

Újdonság: Esettanulmányok hatékonyságnöveléshez
Esettanulmányok ellenőrzések és számolások megfogalmazásához

Példák a számolásokhoz
Az oldal csatolmányaként letölthető egy kérdőívsablon XML, melyben a számoláskora találhatóak példák. *(Tools/Attachements/szamolas_peldak.xml)*

Számolások

Egy adott mezőhöz tartozó számolások megadása két féle módon történhet: 1. egyetlen kifejezéssel (return nélkül), 2. bonyolultabb számolás esetén több sorban, a végén return segítségével.

A legegyszerűbb számolás

A legegyszerűbb számolás, amely a komponens saját értékét írja vissza ugyanebbe a komponensbe:

```
ertek
```

Számolás megadása egyetlen kifejezéssel

Egyszerű számolások megadására elég lehet egy kifejezés is, ilyen esetekben nem kötelező a return használata, de használható is, így az alábbi két kódrészlet ugyanazt eredményezi:

```
2*kerdoiv.getErtek("mezo1")
return 2*kerdoiv.getErtek("mezo1")
```

Bonyolultabb számolás megadása

Bonyolultabb számolások implementálására procedurális eszközöket is használhatunk ugyanolyan formában, mint ahogy azt az ellenőrzéseknél is tesszük. A számolást egy return kifejezéssel kell befejeznünk, amelyben visszatérünk a számolás eredményével.

Automatikusan létrejövő változók

Az "ertek" változó

Az "ertek" változó automatikusan deklarálódik, és annak az aktuális komponensnek az értékét tartalmazza, amelyen a számolást megadtuk. Ez olyan esetekben lehet hasznos, amikor a számolásnak nem kell módosítania az aktuális mező értékét, csak a mellékhatás miatt írunk számolást.

Példa: egy olyan számolás, amely másik mező értékét módosítja annak a komponensnek az értékére, amelyre a számolást tettük.

```
kerdoiv.setErtek("mezo2", ertek);
return ertek;
```

Ez alapértelmezetten szöveges típusú értéket tartalmaz, ezért szükség esetén szám típusra konvertálni kell:

```
parseInt(ertek)("mezo1")
```

Üresen hagyott mezők értékének kezelése

Az egyeztetés eredményeként az egyes komponenstípusoknál az alábbi megállapodás szerint kapja vissza a javascript futtató az egyes üresen hagyott mezők értékét:

1) Rádiógomb: Vagy ki van jelölve, vagy nincs, és ennek megfelelően kizárólag a "Kiválasztott érték" vagy a "Nem kiválasztott érték" tartalmát adja vissza stringként. Ha valaki ezek között definiál üres stringet, akkor az lehet üres string, de null-t soha nem ad vissza.

2) Jelölő négyzet: A rádiógombbal megegyező módon.

3) Legördülő menü: Amíg senki nem állítja be és nincs megadva alapértelmezett érték, akkor üres stringet ad vissza. A kiválasztást követően a kiválasztott elem kódértékét adja vissza stringként. Megengedett, hogy kódként üres string-et definiáljon a szerkesztő egy legördülő menü esetében egy elemnél. Így lehetővé tehető a felhasználónak, hogy a kiválasztás után meggondolja magát, és mégis "törölje" a kiválasztott értéket. Fontos hogy nem lehet megkülönböztetni, hogy lett-e bármi kiválasztva, és törölve, vagy teljesen érintetlen a mező.

4) Nomenklatúra mező: A legördülő menüvel megegyező módon.

5) Egysoros szövegbeviteli mező: Ha nincs kitöltve akkor üres string a visszatérési értéke. Egyébként a beírt szöveg.

6) Többsoros szövegbeviteli mező: Az egysoros szövegbeviteli mezővel megegyező módon.

7) Szám beviteli mező: Ha nincs értékkel kitöltve (vagy kitörli a mezőt, vagy nulla nem beírható mezőbe nullát üt, túl nagy túl kicsi értéket ad meg, stb), akkor null a visszaadott érték, egyébként a mező által kijelzett számérték szám típussal.

8) Sorszám mező: Az adat XML-be írással megegyező szabály szerint. (Azaz, ha van maszkolás a mezőn, akkor a maszkolt szöveget stringként, ha nincs maszk, akkor a sorszám érték string típussal)

Az "azonosito" változó

Az "azonosito" változó annak a komponensnek a kérdőíven belül egyedi azonosítóját tartalmazza, amelyen a számolás fut. Ha a mező ismétlődésen belül található, akkor az azonosító tartalmazza az ismétlődésből eredő indexeket is, pl.: "a" azonosítójú elem esetén "a[0]", "a[1]", vagy több szintű ismétlődés esetén: "a[0][0]", "a[0][1]".

A "sorszam" változó

A "sorszam" változó az ismétlődésen belüli aktuális indexet jelzi, a sorszámozás 0-tól történik, pl. a 3. sorban sorszam = 2 lesz.

A következő példa minden második sorban az adott mező szerkeszthetőségét megszünteti:

```
if (sorszam%2==1)
{
   kerdoiv.setTulajdonsag(azonosito, "szerkesztheto", false);
}
return ertek;
```

Az "az ismétlődésen belüli aktuális index" definíciót tágan kell értelmezni. Jó példa erre az az eset, hogy olyan komponensnél használjuk a "sorszam"-ot, ami nincs ismétlődő paneleken belül, de ismétlődő fejezetben található. Ilyenkor a "sorszam" az adott fejezet előfordulásai közül adja meg az aktuális fejezetpéldány sorszámát.

Ha az adott elem semmilyen szinten nincs ismétlődésben, akkor a sorszam változó alapértelmezetten -1 -et ad vissza.

A "szuloSorszam" változó

A "szuloSorszam" változó 2 egymásba ágyazott ismétlődés esetén használatos. A belső ismétlődésen belüli adott komponens külső ismétlődésbeli szülő komponensének indexét jelzi. A sorszámozás 0-tól történik, pl. a 3. sorban szuloSorszam = 2 lesz.

A következő példa egy ellenőrzés keretében összehasonlítja minden külső ismétlődés esetén az összes belső ismétlődés 2 komponensének értékét:

```
if ( nvlNan($("SzamMezo2002["+szuloSorszam+"]["+sorszam+"]"),0) >
     nvlNan($("SzamMezo2001["+szuloSorszam+"]["+sorszam+"]"),0) ) {
  return "HIBAKOD";
}
else {
  return "OK";
}
```

Ha a magasabb szintű ismétlődés nem értelmezhető, akkor a szuloSorszam változó alapértelmezetten -1-et ad vissza. Pl.: egy ismétlődő fejezetben, de egyébként nem ismétlődő panelekben elhelyezett mezőnél a sorszam változó még értelmezhető, mert az ismétlődő fejezet sorszámát fogja visszaadni, míg a szuloSorszam már a kérdőív szintű ismétlődést tudná csak tükrözni, de az soha nem ismétlődhet, így az értéke -1-es default érték lesz.

A "sorszamok" tömb

A "sorszamok" tömb az adott komponens minden ismétlődésbeli sorszámait adja vissza egy tömbben, a legelső eleme a legkülső ismétlődésbeli index (pl.: ismétlődő fejezetnél a fejezet sorszáma), a legutolsó pedig a komponenshez legközelebb eső tartalmazó ismétlődésbeli index.

2 egymásba ágyazott ismétlődés esetén a külső ismétlődés 2. sorában és a belső ismétlődés 3. sorában lévő komponensen a "sorszamok" tömb lekérdezése az "1,2" értéket fogja mutatni.

```
kerdoiv.naploz(sorszamok); //1,2
```

Amennyiben a mező egyáltalán nincs ismétlődésben, akkor a "sorszamok" (tömb) változó egy üres tömböt ad vissza eredményként.

Az "esemeny" változó

Az "esemeny" változó tartalmazza a számolás lefutását kiváltó esemény azonosítóját. Jelenleg a következő eseményeket különböztetjük meg:

- ERTEK_VALTOZAS (megváltozott a mező értéke)
- MEZO_ELHAGYAS (a felhasználó kikattint a mezőből)
- SOR_HOZZAADAS (kizárólag ismétlődésnél definiált számolásnál vagy ellenőrzésnél)
- SOR_TORLES (kizárólag ismétlődésnél definiált számolásnál vagy ellenőrzésnél)
- ELLENORZESEK_UJRAFUTTATASA (a felhasználó a hibalista frissítését kéri)
- INICIALIZALAS (kérdőívsablon betöltésnél)

Példa az események figyelésére komponens ismétlődésnél megadott számolás esetében:

```
if (esemeny == "SOR_HOZZAADAS")
{
   kerdoiv.hozzaadSor("teszt_ismetlodo_panel", index);
}
else if (esemeny == "SOR_TORLES")
{
   kerdoiv.torolSor("teszt_ismetlodo_panel", index);
}
```

Az "index" változó

SOR_HOZZAADAS és SOR_TORLES események esetén a művelet által érintett sor indexét az index változó tartalmazza szám típusú értékként.

A "forras" változó

Az adatmezők kapcsán különböző funkcióknál szükség lehet annak ismeretére, hogy az adott mező értéke milyen forrásból származik. Ezt tartalmazza a "forras" változó. Két értéket adhat vissza:

- "M" - megszemélyesítésből származó adatok esetén,
- "A" - adatszolgáltató által bevitt adat, vagy abból számolással előállt értékeknél.

A következő példa a feltételes szerkeszthetőségét valósítja meg, a megszemélyesített adatot tartalmazó egyébként kitöltendő mező szerkeszthetőségét letiltja:

```
if (forras=="M")
{
   kerdoiv.setTulajdonsag(azonosito, "szerkesztheto", false);
}
return ertek;
```

Két táblázat szinkronizációja számolások segítségével

Számolások megadásával megoldható két, akár külön fejezetben található táblázat sorainak szinkronban tartása.

Lépések:
1. Tegyük fel, hogy a két ismételhető táblázatunkat "a_tablazat" és "b_tablazat" azonosítóval láttuk el, a_tablazat változásait szeretnék b felé delegálni. Mindkét táblázat első két "oszlopa" két nomenklatúra beviteli mező, melyek közül az egyik a kiválasztott nomenklatúra kódját (megjelenítés: KOD), a másik pedig ennek megnevezését (megjelenítés: MEGNEVEZES) jeleníti meg. Ezeket az egyszerűség kedvéért a példában a következőképpen nevezzük el: "a_nomenklaturamezo_kod", "a_nomenklaturamezo_megnevezes" valamint "b_nomenklaturamezo_kod" és "b_nomenklaturamezo_megnevezes".

2. Helyezzük el az "a_tablazat" ismétlődésén a következő számolást:

```
if (esemeny == "SOR_HOZZAADAS")
{
   kerdoiv.hozzaadSor("b_tablazat", index);
}
else if (esemeny == "SOR_TORLES")
{
   kerdoiv.torolSor("b_tablazat", index);
}
```

Figyeljük meg, hogy mivel a komponens, amelyen a számolást elhelyeztük, nem kitölthető komponens (hanem ismételhető komponens), ezért nem kötelező return kifejezés a számolás megadásánál.

3. Helyezzük el az "a_nomenklaturamezo_kod" mezőn a következő számolást:

```
kerdoiv.setErtek("b_nomenklaturamezo_kod[" + sorszam + "]", ertek);
return ertek;
```

4. Helyezzük el az "a_nomenklaturamezo_megnevezes" mezőn a következő számolást:

```
kerdoiv.setErtek("b_nomenklaturamezo_megnevezes[" + sorszam + "]", ertek);
return ertek;
```

5. Állítsuk a b_nomenklaturamezo_kod és b_nomenklaturamezo_megnevezes mezőket nem szerkeszthetőre.

Számolások táblázaton belül

Ismétlődéses táblázat különálló összesen sorral

Előnyomásos és előnyomás nélküli ismétlődő táblázatoknál az összesen sor megvalósítására a sztenderd módszer egy különálló sor felvétele erre a célra. Ezt úgy célszerű megoldni, hogy a táblázat ismétlődő sorait magában foglaló függőleges panel alá elhelyezünk egy vízszintes panelt, ami már maga lesz az összesen sor. Ismétlődés esetében az egy oszlopba tartozó cellák ugyanazt az azonosítót kapják és a kitöltő a sorok generálásakor ezeket az azonosítókat

indexel látja el, így tömböt képez belőlük. Mivel az összesen sort egy külön panelben helyeztük el, nem része az ismétlődésnek így ezen sor bármely komponensére írhatunk egy olyan számolást, ami a felette - külön panelben - elhelyezkedő oszlop (tömb) értékeit összeadja és az értékét visszaadja az adott mezőre.
A külön panelben elhelyezett összesen sor adott komponensére az alábbi számolást kell megírnunk:

```
return sum(kerdoiv.getErtekek("azonosito"));
```

A fenti példa SzámMező használatát feltételezi. Amennyiben EgysorosMező komponenst használunk, szükséges a parseFloat konverzió.

Ismétlődéses táblázat generált összesen sorral

Előnyomásos és előnyomás nélküli ismétlődő táblázatoknál az összesen sort különálló panel felvétele nélkül is kivitelezhetjük. Ismétlődés esetében az egy oszlopba tartozó cellák ugyanazt az azonosítót kapják és a kitöltő a sorok generálásakor ezeket az azonosítókat indexel látja el, így tömböt képez belőlük. Amennyiben nem szeretnénk külön panelt alkalmazni az összesen sorra, úgy az egyetlen látszódó sor adott celláira kell megírnunk a számolást. Olyan számolást kell írnunk, amely a felette - azonos panelben - elhelyezkedő oszlop (tömb) értékeit összeadja és az értékét visszaadja az összesen komponensre. Mivel az a sor fog ismétlődni, amelyben az összesen sor is van, egy feltétellel ellenőriznünk kell, hogy az összeadás teljes művelete csak az utolsó - vagy az összeadásra kijelölt - sorban fusson le. Szintén figyelnünk kell, nehogy az összesen komponens is bele kerüljön az összeadásba, mert az hurkot eredményez.
Az ismétlendő sor adott komponensére az alábbi számolást kell megírnunk:

```
if ( sorszam==13 )
{
  var osszeg=0;
  var tomb=kerdoiv.getErtekek("azonosito");

  for ( var i=0; i<13; i++)
  {
    osszeg += nvlNan(tomb[i],0);
  }
  return osszeg;
}
return ertek;
```

A fenti példa SzámMező használatát feltételezi. Amennyiben EgysorosMező komponenst használunk, szükséges a parseFloat konverzió.

Függő számolások futtatásának tiltása

A számolásokat frissítési szempontból két csoportra oszthatjuk. Egy adott "A" mezőt tekintve van a saját definíciójában szereplő számolás (saját számolás) másrészt a más mezők (pl.: "B") definíciójában szereplő olyan számolás, aminek a végeredménye függ az "A" értékétől (pl: B=2*(A+C) esetén a B számolása függ az "A" és a "C" mezők értékétől.). Utóbbiakat nevezzük függő számolásnak. Abban az esetben, ha nincs szükség arra, hogy egy adott mező értékétől függő számolások lefussanak a mező értékének változása után, a komponens fuggoSzamolasokTiltva tulajdonságát kell true értékre beállítani:

```
kerdoiv.setTulajdonsag("azonosito", "fuggoSzamolasokTiltva", false);
```

Ez különösen hasznos lehet a kérdőív gyorsaságának az optimalizációja esetén, ahol a mező különböző példányainak az értékével készül valamilyen számolás. Pl. az előnyomással készült ismétlődéseken megadott számolások optimalizációjánál, olyan esetekben, amikor az (rész)összegző sor is szerepel az előnyomásban, ezáltal az összesen mező azonosítója is pontosan ugyanaz, mint az összegzendő mezőké. Ilyenkor az összegző mezőn beállítva ezt a tulajdonságot, annak értékváltozásánál nem fut le újra az összes mező példányra a számolás, beleértve az összegzést is még egyszer (ami ezáltal végtelen ciklus lehetne, és csak a kitöltő végtelen ciklus megszakító algoritmusa leállítja a számolási hurkot). Általánosan elmondható, hogy teljesítmény szempontokból érdemesebb az összesen sorokat külön kiszerkeszteni, de ha valamilyen okból mégis az ismétlődésen belül kellene használni, akkor lehetőség van a függő számolásokat így letiltani.

Szabad paneles táblázat összesen sorral

Amennyiben a táblázatot saját kézzel vagy a komponensek generálása funkció segítségével hozzuk létre, nem lesz ismétlődésünk és az egész táblázatot magunk előtt látjuk. Ez esetben egy oszlop minden komponense más azonosítót kap, így az ismétlődéseknél megszokott tömbbe rendezhetőség nem itt nem fog működni. Ellenben, mivel valószínűleg az oszlop minden komponense azonos oszlop tulajdonságot kap, megtehetjük, hogy az oszlop azonosítóit az oszlop tulajdonság értéke alapján gyűjtjük tömbökbe. Így már könnyedén el tudjuk végezni az összeadási műveletet. Egyedül arra kell figyelni az oszlop alapján való tömbbe rendezés esetén, hogy az összesen komponens értéke ne kerüljön bele az összeadásba, mert az hurkot eredményez.
Az ismétlendő sor adott komponensére az alábbi számolást kell megírnunk:

```
var osszeg=0;
var tomb=kerdoiv.getErtekekByOszlop("oszlop","kontener_azonosito");

for ( var i=0; i<(tomb.length-1); i++)
{
  osszeg += nvlNan(tomb[i],0);
}
return osszeg;
```

A fenti példa SzámMező használatát feltételezi. Amennyiben EgysorosMező komponenst használunk, szükséges a parseFloat konverzió.

Oszlop összes elemének összeadása néhány kivételével

Ha bizonyos sorokat nem szeretnénk belevonni az összeadásba, azokat egyesével ki kell vonnunk. Az alábbi példában 2. és 5. sorokat vonjuk ki az összegből. A tömbök sorszámozása 0-tól indul, így a 2. sor: oszlop_azonositoja[1] ! A példa a klasszikus táblázat szerkezetet feltételezi, de kis átalakítással használható bármely táblázattípusnál.

```
var osszeg1 = sum(kerdoiv.getErtekek("oszlop_azonositoja"));
return (osszeg1 - kerdoiv.getErtek("oszlop_azonositoja[1]")
                - kerdoiv.getErtek("oszlop_azonositoja[4]"));
```

vagy 1 lépésben és rövidítve

```
return (sum(kerdoiv.getErtekek("oszlop_azonositoja"))
        - $("oszlop_azonositoja[1]")
        - $("oszlop_azonositoja[4]"));
```

A fenti példa SzámMező használatát feltételezi. Amennyiben EgysorosMező komponenst használunk, szükséges a parseFloat konverzió.

Oszlop egyes elemeinek összeadása

Amennyiben az oszlopnak csak egyes elemeit szeretnénk összeadni, a legegyszerűbb módszer, ha az elemeket egyenként adjunk össze. Az alábbi példában 2. és 5. sorokat adjuk össze. A példa a klasszikus táblázat szerkezetet feltételezi, de kis átalakítással használható bármely táblázattípusnál.

```
return (kerdoiv.getErtek("oszlop_azonositoja[1]") +
        kerdoiv.getErtek("oszlop_azonositoja[4]"));
```

vagy rövidítve

```
return ($("oszlop_azonositoja[1]") +
        $("oszlop_azonositoja[4]"));
```

A fenti példa SzámMező használatát feltételezi. Amennyiben EgysorosMező komponenst használunk, szükséges a parseFloat konverzió.

Végtelen ciklus és hurok megszakítás

Legsűrűbben a táblázatoknál használt összesen sor esetében fordul elő, de egyéb számolásból is adódhat olyan helyzet, ahol végtelen ciklus jön létre. Jellemző példája, amikor egy összeadásnál az összeadandó mezőt is belefoglaljuk a számolásba. Feltételekkel ajánlott figyelni, hogy számolásoknál csak a szükséges komponensekkel végezzük el a műveleteket. A felhasználói ellenőrzés mellé bekerült a kitöltőbe egy hurokfigyelő rendszer, ami megkísérli felismerni és letiltani a kialakuló végtelen ciklusokat. A beépített ellenőrzés heurisztika alapján működik. Előfordulhat, hogy a problémás számolás első körben lefut, így a kívánt műveletet a kitöltő elvégzi, ám a naplóban láthatjuk, hogy megakadályozta a további futásokat. Működéséből adódóan huroknak vélhet olyan ciklust is, ami nem az. Ez a függvények helyes működését nem gátolja, pusztán egy figyelmeztetést látunk a naplóban.

Ellenőrzések

A számolásokkal ellentétben az ellenőrzések megadásánál a megfelelő Hibaüzenet kódjával kell visszatérni, pl.:

```
if (parseInt(ertek)>3)
{
   return "NEM_LEHET_NAGYOBB_MINT_3"
}
return "OK";
```

A hibalistából való ugrás célmezőjének beállítása

Az ellenőrzés eredményének megadásánál definiálhatjuk az "Ugrás" gomb megnyomása utáni célmezőt is a következő formában egy objektum megadásával:

```
...
return {
   kod: "NEM_LEHET_NAGYOBB_MINT_3",
   ugras: "SorszamMezo3"
};
...
```

Az ugras tulajdonság értéke sztring típusú lehet és a célmező azonosítóját tartalmazza. Az azonosító megadásánál az ismétlődésbeli sorszámot is megadhatjuk:

```
...
return {
   kod: "NEM_LEHET_NAGYOBB_MINT_3",
   ugras: "SorszamMezo3[3]"
};
...
```

Az üzenetben szereplő paraméterek értékének megadása

A hibák üzeneteit felparaméterezve is megadhatjuk (a kérdőívszerkesztőben), pl.: "A {} nevű mező értéke nem lehet {}!".
Az ellenőrzés eredményének megadásánál ilyen esetekben megadhatjuk az üzenetbe behelyettesítendő értékeket is:

```
...
return {
   kod: "NEM_LEHET_NAGYOBB_MINT_3",
   parameterek: ["x", "y"]
};
...
```

A parameterek tulajdonság értéke mindig tömb, elemei pedig sztring vagy szám típusúak lehetnek.

Ismétlésben szereplő hibák csoportosításának megadása

Az ismételt mezőkön szereplő mezőkből származó hibák a hibalistában egy hibaként jelennek meg, de néhány esetben szükség lehet arra, hogy ezek ismétlődés-példányonként külön szerepeljenek a hibalistában. Ennek beállítását tehetjük meg a csoportositas tulajdonság használatával.

Ha azt szeretnénk, hogy a hibalistában csak a legutolsó alkalommal hibát okozó mező hibája jelenjen csak meg, akkor a csoportositas tulajdonságot állítsuk true-ra:

```
...
return {
   kod: "NEM_LEHET_NAGYOBB_MINT_3",
   csoportositas: true
};
...
```

A csoportositas tulajdonság értékének típusa boolean, alapértelmezett értéke: false, azaz nincs csoportosítás.

Ahol a hibákat csoportositas: true tulajdonsággal hozzuk létre, ott a hiba feloldására szolgáló return "OK" -ot is az alábbi módon kell megadni:

```
...
return {
   kod: "OK",
   csoportositas: true
};
...
```

Visszatérés több hibaüzenettel

Előfordulhat, hogy egy ellenőrzésből több hibaüzenettel is vissza kell térni, ezt a következő módon lehet megvalósítani:

```
...

var ellenorzesUzenetek = new Array();

...

// hibaüzenet hozzáadása
ellenorzesUzenetek.push({
   kod: "HIBA",
   parameterek: [a,">",b],
   ugras: ("GBAC002_"+i)
});

...

if (ellenorzesUzenetek.length > 0)
{
   return ellenorzesUzenetek;
}
return "OK";
```

Ilyen esetben a "csoportositas" paraméternek nincs hatása a működése, a tömbbel visszaadott üzenetek nem csoportosíthatóak.

API

Fontos információk

A reguláris kifejezések megadásának szabályai

A függvényekben használandó reguláris kifejezése megadása az alábbi oldalon elérhető szabályrendszer szerint alkalmazandó tekintettel az lenti megjegyzésekre:

http://help.adobe.com/en_US/as3/dev/WS5b3ccc516d4fbf351e63e3d118a9b90204-7ea9.html

```
Megjegyzések:
1) A g flag nincs értelmezve, tehát pl.: /abc/i és /abc/gi ugyanazt fogja
eredményezni.
2) A regexp normál kifejezésében "\"-ként definiálandó kifejezéseknél
transzformációs okokból "\\" kell használni.
```

Implicit konténer

Amennyiben a kérdőív objektum (érték, tulajdonság, stb. elérését biztosító) függvényei esetében nem adunk meg konténer azonosítót, az alapértelmezett konténer az aktuális fejezet lesz, ahol a számolást vagy ellenőrzést hordozó komponens található. Ha a keresett komponens nincs az aktuális fejezetben, akkor a keresés tovább folytatódik a teljes kérdőíven.

Ha tudjuk, hogy a keresett komponens egy másik fejezetben szerepel, akkor megadhatjuk a fejezet azonosítóját.

```
$("a", "F"); // Az a komponens keresése az F fejezetben.
```

Abban az esetben, ha a teljes kérdőívben szeretnénk a komponenseket összegyűjteni, akkor a KERDOIV konstanst kell használni.

```
$("/^a", KERDOIV); // Az "a"-val kezdődő azonosítójú összes komponens
értékének lekérdezése az egész kérdőívből.
```

A kérdőív objektum

Mező értékének lekérdezése

Normál alak:

kerdoiv.getErtek(azonosító teljes megadása vagy reguláris kifejezés)

kerdoiv.getErtek(azonosító vagy reguláris kifejezés, konténer azonosító)

kerdoiv.getErtekek(azonosito)

kerdoiv.getSzamErtek(azonosito) - kompatibilitási okokból

Rövidített alak:

$(azonosító teljes megadása vagy reguláris kifejezés)

$(azonosító vagy reguláris kifejezés, konténer azonosító)

A kerdoiv.getErtekek és a kerdoiv.getSzamertek nem rendelkezik rövidített alakkal.

Visszatérési érték:

Típusa: sztring, szám vagy logikai (igaz/hamis), a mező típusától függően. Nómenklatúra mező esetén a kiválasztott elem kódja sztringként, ha a nómenklatúra mezőben nincs kiválasztva elem, akkor üres sztring ("").

Ha a hívásnál olyan komponens azonosítóját adjuk meg, amely ismétlődésekben szerepel, és az azonosítóban nem adunk meg ismétlődésbeli sorszámozást (pl.: "a[0]" helyett csak "a"), akkor a visszatérési érték típusa tömb lesz. Ha a komponens ismétlődő fejezetben található, de más ismétlődésben nem szerepel és nem adunk meg konténert (tehát az aktuális fejezetpéldányon keressük a komponenst), vagy megadjuk a komponens fejezetpéldány azonosítóját (pl.: $("a", "f[0]") - keressük az "a"-t az "f" fejezet első példányán), akkor a visszatérési típus nem tömb lesz, hanem a mező típusának megfelelően sztring, szám vagy logikai.

Ha a hívásnál reguláris kifejezést adtunk meg, akkor a visszatérési érték tömb, ennek vizsgálatára bevezetjük az isArray(érték) függvényt. Reguláris kifejezések használata esetén a tömb heterogén is lehet, a benne szereplő értékek az adott mező típusától függően: sztring, szám vagy logikai.

Példa:

```
var x=kerdoiv.getErtek("mezo1");

Azonosító teljes megadásával:
kerdoiv.getErtek("szamlalo_ossz") vagy röviden $("szamlalo_ossz")

Reguláris kifejezéssel:
kerdoiv.getErtek("/^UEWH(\\d)+$/") vagy röviden $("/^UEWH(\\d)+$/")
(Az összes olyan mező értékének lekérdezése, mely azonosítója az "UEWH"
karaktersorozattal kezdődik és legalább egy, de akárhány számjeggyel
folytatódik.)
...
```

vagy speciálisan számmezők esetében használható még a:

```
var x=kerdoiv.getSzamErtek("mezo1");
```

Ha a mező pl. egy hexa számot tartalmaz, akkor a parseInt hívásnál megadhatjuk a számrendszer alapját is:

```
var x=parseInt(kerdoiv.getErtek("mezo1"), 16);
...
```

Mező értékének számolási célból korrigált lekérdezése

Normál alak:

kerdoiv.getNullaErtek(azonosító teljes megadása vagy reguláris kifejezés)

Rövidített alak:

Nincs.

Visszatérési érték:

A kerdoiv.getErtek() függvénynél specifikáltakkal megegyező attól a kivételes esettől eltekintve, ha a lekérdezett komponens(ek) típusa Számmező, és az értéke null. Ilyenkor nem nullt, hanem 0 értéket ad vissza a függvény.

Mező értékének lekérdezése a mező egy tulajdonságának értéke alapján

A kerdoiv.getTulajdonsag függvénynél definiált tulajdonságok szerint lekérdezhető a megfelelő mezők értéke.

Normál alak:

kerdoiv.getErtekekByTulajdonsag(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés)
kerdoiv.getErtekekByTulajdonsag(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés, konténer azonosító)

Rövidített alak:

$#(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés)
$#(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés, konténer azonosító)

Visszatérési érték:

Típusa: homogén tömb, minden elem típusa megegyezik (nincs két különböző mező típus, amelynek közös nevű tulajdonsága van, de típusuk különböző). A tömbbeli elemek típusa a tulajdonság típusától függ, lehet logikai (pl. szerkeszthető), sztring vagy szám. Az elemek sorrendje az automatikus tabsorrenddel (az elemek kérdőívsablon XML-beli sorrendjével) egyezik meg.

Példa:

```
kerdoiv.getErtekekByTulajdonsag("szerkeszthető", true) vagy röviden
$#("szerkeszthető", true)

Reguláris kifejezés használata az érték paraméternél:
kerdoiv.getErtekekByTulajdonsag("oszlop", "/^UEWH(\\d)+$/") vagy röviden
$#("oszlop", "/^UEWH(\\d)+$/")
(Minden olyan mező, ahol az oszlop tulajdonság értéke az "UEWH"
karaktersorozattal kezdődik és legalább egy, de akárhány számjeggyel
folytatódik.)

Reguláris kifejezés használata az érték paraméternél és konténer megadása:
kerdoiv.getErtekekByTulajdonsag("oszlop", "/^UEWH(\\d)+$/", "tablazat2")
vagy röviden $#("oszlop", "/^UEWH(\\d)+$/", "tablazat2")
(Minden olyan mező a "tablazat2" panelen belül, melynél az oszlop
tulajdonság értéke az "UEWH" karaktersorozattal kezdődik és legalább egy,
de akárhány számjeggyel folytatódik.)
```

Mező értékének lekérdezése a mező "Oszlop" tulajdonságának értéke alapján

Normál alak:

kerdoiv.getErtekekByOszlop(érték teljes megadása vagy reguláris kifejezés)
kerdoiv.getErtekekByOszlop(érték teljes megadása vagy reguláris kifejezés, konténer azonosító)

Rövidített alak:

$O(érték teljes megadása vagy reguláris kifejezés)
$O(érték teljes megadása vagy reguláris kifejezés, konténer azonosító)

Visszatérési érték:

Típusa: homogén tömb, minden elem típusa megegyezik (nincs két különböző mező típus, amelynek közös nevű tulajdonsága van, de típusuk különböző). A tömbbeli elemek típusa a tulajdonság típusától függ, lehet logikai (pl. szerkesztheto), sztring vagy szám. Az elemek sorrendje az automatikus tabsorrenddel (az elemek kérdőívsablon XML-beli sorrendjével) egyezik meg.

Mező azonosítóinak lekérdezése

Normál alak:

kerdoiv.getAzonositok(reguláris kifejezés)
kerdoiv.getAzonositok(reguláris kifejezés, konténer azonosító)

Rövidített alak:

@(reguláris kifejezés)
@(reguláris kifejezés, konténer azonosító)

Visszatérési érték:

Típusa: sztringek tömbje.

Példa:

```
kerdoiv.getAzonositok("/^UEWH(\\d)+$/")
(Minden olyan mező azonosítóját visszaadja egy tömbben, melynél az
azonosító az "UEWH" karaktersorozattal kezdődik és legalább egy, de
akárhány számjeggyel folytatódik.)

kerdoiv.getAzonositok("/^UEWH(\\d)+$/", "tablazat2")
(Ugyanaz, mint a felső, de csak a "tablazat2" panelen belüli azonosítókat
kapjuk vissza.)
```

Mező azonosítóinak lekérdezése tulajdonság alapján

A kerdoiv.getTulajdonsag függvénynél definiált tulajdonságok szerint lekérdezhető a megfelelő mezők azonosítója.

Normál alak:

kerdoiv.getAzonositokByTulajdonsag(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés)
kerdoiv.getAzonositokByTulajdonsag(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés, konténer azonosítója)

Rövidített alak:

@#(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés)
@#(tulajdonság azonosítója, érték teljes megadása vagy reguláris kifejezés, konténer azonosítója)

Visszatérési érték:

Típusa: sztringek tömbje.

Példa:

```
kerdoiv.getAzonositokByTulajdonsag("szerkesztheto", true) vagy röviden
@#("szerkesztheto", true)

Reguláris kifejezés használata az érték paraméternél:
kerdoiv.getAzonositokByTulajdonsag("oszlop", "/^UEWH(\\d)+$/") vagy röviden
$#("oszlop", "/^UEWH(\\d)+$/")
(Minden olyan mező azonosítója, ahol az oszlop tulajdonság értéke az "UEWH"
karaktersorozattal kezdődik és legalább egy, de akárhány számjeggyel
folytatódik.)

Reguláris kifejezés használata az érték paraméternél és konténer megadása:
kerdoiv.getAzonositokByTulajdonsag("oszlop", "/^UEWH(\\d)+$/", "tablazat2")
vagy röviden @#("oszlop", "/^UEWH(\\d)+$/", "tablazat2")
(Minden olyan mező azonosítója a "tablazat2" panelen belül, melynél az
oszlop tulajdonság értéke az "UEWH" karaktersorozattal kezdődik és legalább
egy, de akárhány számjeggyel folytatódik.)
```

Mező azonosítóinak lekérdezése az "Oszlop" tulajdonság alapján

Normál alak:

kerdoiv.getAzonositokByOszlop(érték teljes megadása vagy reguláris kifejezés)
kerdoiv.getAzonositokByOszlop(érték teljes megadása vagy reguláris kifejezés, konténer azonosítója)

Rövidített alak:

@O(érték teljes megadása vagy reguláris kifejezés)
@O(érték teljes megadása vagy reguláris kifejezés, konténer azonosítója)

Visszatérési érték:

Típusa: sztringek tömbje.

Mező értékének beállítása

Normál alak:

kerdoiv.setErtek(*mező azonosító*, *érték*)

A *mező azonosítóval* adjuk meg a mező azonosítóját, ha a mező ismétlésben szerepel, akkor meg kell adnunk a mező ismétlésbeli sorszámát is (lásd a példát).

Rövidített alak:

Nincs.

Visszatérési érték:

Nincs.

Példa:

```
var x=kerdoiv.setErtek("a[2]", 17);
...
```

A példában a számolás az "a" mezők közül a 3. elem értékét állítja be 17-re.

Tulajdonság értékének lekérdezése

Normál alak:

kerdoiv.getTulajdonsag(*mező azonosító*, *tulajdonságnév*)

A *mező azonosítóval* adjuk meg a mező azonosítóját, ha a mező ismétlésben szerepel, akkor meg kell adnunk a mező ismétlésbeli sorszámát is (lásd a példát). A *tulajdonságnév* a lekérdezendő tulajdonságot adja meg:

- Szerkeszthetőség: "szerkesztheto",
- Háttérszín: "hatterszin",
- Előtérszín: "eloterszin",
- Láthatóság (fejezet, panel vagy mező egyaránt): "lathatosag",
- Ismételhetőség (felhasználó általi) fejezet szinten: "ismetelheto",
- Címke szövege: "szoveg",
- Címke igazítása: "igazitas",
- X koordináta: "x",
- Y koordináta: "y",
- Szélesség: "szelesseg",
- Magasság: "magassag",
- Azonosító: "azonosito",
- Oszlop: "oszlop",
- Függő számolások tiltása: "fuggoSzamolasokTiltva".

Rövidített alak:

#(*mező azonosító*, *tulajdonságnév*)

Visszatérési érték:

Sztring.

Tulajdonság értékének beállítása

Normál alak:

kerdoiv.setTulajdonsag(*mező azonosító*, *tulajdonságnév*, *érték*)

A *mező azonosítóval* adjuk meg a mező azonosítóját, ha a mező ismétlésben szerepel, akkor meg kell adnunk a mező ismétlésbeli sorszámát is (lásd a példát). A *tulajdonságnév* a lekérdezendő tulajdonságot adja meg:

- Szerkeszthetőség: "szerkesztheto",
- Háttérszín: "hatterszin",
- Előtérszín: "eloterszin",
- Láthatóság (fejezet, panel vagy mező egyaránt): "lathatosag",
- Ismételhetőség (felhasználó általi) fejezet szinten: "ismetelheto".
- Címke szövege: "szoveg",
- Címke igazítása: "igazitas",
- X koordináta: "x",
- Y koordináta: "y",
- Szélesség: "szelesseg",
- Magasság: "magassag".

A fejezetek esetében a felhasználó általi ismételhetőség bővebben kifejtve azt jelenti, hogy a szerkesztéskor ismétlődőnek jelölt fejezethez a felhasználó adhat-e hozzá saját ismétlődést, azaz megismételheti-e a fejezetet. Ha az adott fejezet szerkesztőben nem rendelkezik ismétlődéssel, akkor az ismételhetőség állítása önmagában nem teszi lehetővé új példány hozzáadását. Azaz tervezéskor el kell dönteni, hogy egy fejezet valaha lehet-e ismétlődő, és ha igen, akkor hozzá kell adni az ismétlődés tulajdonságot.

Rövidített alak:

Nincs.

Visszatérési érték:

Nincs.

Példa:

```
kerdoiv.setTulajdonsag("SzamMezo1","hatterszin", 0xFF0000);
```

Sor hozzáadása ismétlődéshez

Számolásból vagy ellenőrzésből egy ismétlődéshez hozzáadhatunk egy üres sort a következő metódus segítségével:

Normál alak:

kerdoiv.hozzaadSor(*mező azonosító*, *index*)

Mivel a komponens, amelyen a számolást elhelyeztük, nem kitölthető komponens, ezért nem kötelező return kifejezés a számolás megadásánál.

Rövidített alak:

Nincs.

Visszatérési érték:

Nincs.

Sor törlése ismétlődésből

Számolásból vagy ellenőrzésből egy ismétlődésből törölhetünk egy üres sort a következő metódus segítségével:

Normál alak:

kerdoiv.torolSor(*mező azonosító*, *index*)

Mivel a komponens, amelyen a számolást elhelyeztük, nem kitölthető komponens, ezért nem kötelező return kifejezés a számolás megadásánál.

Rövidített alak:

Nincs.

Visszatérési érték:

Nincs.

Bizonylatszám lekérdezése

A kérdőív bizonylatszámának a megjelenítése szükséges lehet. A "kérdőív azonosító"-"verzió" értéket adja vissza

Normál alak:

kerdoiv.getBizonylatszam()

Rövidített alak:

Nincs.

Visszatérési érték:

Sztring

Kódtábla elem ellenőrzése

Normál alak:

kerdoiv.validal(validálandó karakterlánc, Nómenklatúra kódja, nómenklatúra változata, vizsgálandó mező {KOD, MEGNEVEZES, EGYEB_1, EGYEB_2, EGYEB_3, EGYEB_4, EGYEB_5}: alapértelmezetten KOD)

Visszatérési érték:

Típusa: boolean
Eredménye Igaz, ha az adott karakterlánc pontosan illeszkedik legalább 1 nómenklatúra elem vizsgálandó attribútumra
Egyébként a visszatérési érték: Hamis.
Ha ismeretlen a megadott nómenklatúra változat, vagy a vizsgálandó mező hivatkozás, akkor a visszatérési érték: Hamis

Példa

```
var e = kerdoiv.validal("1005", "M562", "B2");
if (e)
{
 return "OK";
}
return "NEM OK";
```

Kódtábla elemek száma

Normál alak:

Függvény fejléce: kerdoiv.validalDarab(validálandó karakterlánc, Nómenklatúra kódja, nómenklatúra változata, vizsgálandó mező {KOD, MEGNEVEZES, EGYEB_1, EGYEB_2, EGYEB_3, EGYEB_4, EGYEB_5}: alapértelmezetten KOD)

Visszatérési érték:

Típusa: number
Ha ismeretlen a megadott nómenklatúra változat, vagy a vizsgálandó mező hivatkozás, akkor a visszatérési érték: -1
Egyéb esetekben az adott karakterláncra pontosan illeszkedő nómenklatúra elemek (vizsgálandó attribútumra vizsgált egyezés) darabszáma: 0..n

Példa

```
var e = kerdoiv.validalDarab("Durumbúza", "M562", "B2", "MEGNEVEZES");
return e;
```

Sablonban szereplő OSAP szám lekérdezése

Normál alak:

Függvény fejléce: kerdoiv.getOSAP()

Visszatérési érték:

Típusa: Sztring
Ha nem találja meg az kerdoivsablon/mc01 tag-et a sablon XML-ben, akkor üres sztring-et ad vissza, egyébként a tag-ben szereplő karaktersorozatot.

Példa

```
return kerdoiv.getOSAP();
```

Sablonban szereplő tárgyév lekérdezése

Normál alak:

Függvény fejléce: kerdoiv.getMEV()

Visszatérési érték:

Típusa: Sztring
Ha nem találja meg a kerdoivsablon/mev tag-et a sablon XML-ben, akkor üres sztring-et ad vissza, egyébként a tag-ben szereplő karaktersorozatot.

Példa

```
return kerdoiv.getOSAP();
```

Kisegítő eszközök

PDF fejléc/lábléc oldalszámozása

A PDF generálás során a fejléc és a lábléc feltöltésekor az oldalszámozás megjelenítését adattal kiszolgáló függvény. A kitöltőben nem használható, mivel ott az oldalszámozás nem értelmezhető.

Normál alak:

pdf.getOldalszam()

Visszatérési érték:

Nemnegatív egész szám

Példa:

```
pdf.getOldalszam()
```

Színek konvertálása

Színek konvertálása hexadecimális formátumba.

Normál alak:

toHex(kerdoiv.getTulajdonsag(azonosito, "hatterszin"))

Visszatérési érték:

Szín kódja hexadecimális alakban.

Példa:

```
toHex(kerdoiv.getTulajdonsag("x", "hatterszin"))
```

vagy egyszerűbb formában:

```
toHex(#("x", "hatterszin"))
```

Naplózás

Számolások és ellenőrzések készítésénél hasznos lehet, ha bizonyos változók futás közbeni értékét meg lehet tekinteni. Ezzel az eljárással tetszőleges üzenetet a naplóba írhatunk.

Normál alak:

kerdoiv.naploz(*uzenet*);

Visszatérési érték:

Nincs.

Példa:

```
kerdoiv.naploz("szamolas indul");
for (var i=0; i<3; i++)
{
   kerdoiv.naploz("i=" + i);
}
kerdoiv.naploz("szamolas befejezodott");
return ertek;
```

A naplóban ennek hatására a következő bejegyzések jelennek meg:

```
szamolas indul
0
1
2
```

```
szamolas befejezodott
```

Tömb elemeinek összeadása

Normál alak:

Előre definiált tömbbel:
sum (tomb_azonosito)

Ismétlődő soros táblázatban indexelt azonosítókból tömbbé alakítással:
sum (kerdoiv.getErtek(azonosito))

Visszatérési érték:

Float

Példa:

Előre definiált tömbbel:

```
sum ("tomb1")
```

Ismétlődő soros táblázatban indexelt azonosítókból tömbbé alakítással:

```
sum ($("EgysorosMezo1"))
```

Oracle nvl

Amennyiben az első paraméter null vagy undefined, a második paramétert, egyébként az első paramétert adja vissza értékül.

Normál alak:

nvl(a, b)

Visszatérési érték:

Float

Példa:

```
nvl(null, 2);
```

A példa 2-t ad vissza értékül.

nvlNan

Amennyiben az első paraméter null, undefined vagy nem szám (Nan), a második paramétert, egyébként az első paramétert adja vissza értékül.

Normál alak:

nvlNan(a, b)

Visszatérési érték:

Float

Példa:

```
nvlNan("a",2);
```

A példa 2-t ad vissza értékül.

Számok kerekítése megadott számú tizedesjegyre

A num paraméterben kapott számot kerekíti a dec paraméterben kapott számú tizedesjegy pontossággal.

Normál alak:

roundNumber(num, dec)

Visszatérési érték:

Number

Példa:

```
var ertek=5.123123;
return roundNumber(ertek, 2);
```

A példa a 5.12-t ad vissza értékül.

Tömb kitöltött elemeinek a darabszáma

A paraméterben kapott tömb elemeit nézi végig. Ha a tömb objektum nem definiált vagy null, akkor 0-t ad vissza. Ha nem tömb, akkor 1-et. Egyébként a skalár elemeket tartalmazó tömb nem null, nem undefined és nem +/- végtelen értékű elemeinek a számosságát adja vissza.

Normál alak:

count(tomb)

Visszatérési érték:

Number

Példa:

```
var tomb=$("mutato_neve");
return count(tomb);
```

Tömb legnagyobb értéke

A legnagyobb értéket adja vissza a tömbből.

Normál alak:

greatest(tömb)

Visszatérési érték:

Float

Példa:

```
var t = new Array(1,3,2);
return greatest (t);
```

A példa a 3-at adja vissza értékül.

Esetszétválasztó függvény tetszőleges számú paramméterre

Ha oValue = oCase1, az eredmény oValue1 stb. Ha oValue egyik oCaseN értékkel sem egyenlő, az eredmény oValueDef, ill. ha oValueDef = ”::self::”, akkor oValue. Az összehasonlítások az értékek eredeti típusai szerint történnek, nincs implicit konverzió.

Normál alak:

decode(oValue, oCase1, oValue1, ..., oCaseN, oValueN, oValueDef)

Visszatérési érték:

Sztring, Float, Int, Tömb, Logikai

Példa:

```
var vizsgalando = 2;
return decode(vizsgalando, 1, "egy", 2, "kettő", "egyik sem");
```

Tömb vizsgálat

Elem vizsgálata, miszerint tömb típusú-e.

Normál alak:

isArray()

Visszatérési érték:

Logikai

Példa:

```
var t = 1;
if (isArray(t)) {
  return "true";
}
else {
  return "false";
}
```

Hossz lekérdezése

Visszaadja egy adott objektum hosszát. Szöveg és tömb esetén a beépített length tulajdonságot adja vissza, egyébként (pl. számok esetén) pedig az objektumot szöveggé konvertálja és annak a hosszát adja vissza. Nem definiált érték esetén 0 a visszaadott érték.

Normál alak:

getLength()

Visszatérési érték

Nemnegatív egész szám

Példa:

```
getLength(null); // = 0
getLength(99); // = 2
getLength(new Array(1,2,3,4)); // = 4
getLength('szoveg'); // 6
```

Kód egyediség vizsgálat

Tömbelemek egyediségének vizsgálata

Megvizsgálja, hogy a tömb elemei egyedi értékekkel rendelkeznek-e. True, amennyiben nincs két egyforma érték, egyébként false.

Normál alak:

egyedi(tömb)

Visszatérési érték:

Logikai

Példa:

```
var t = new Array(1,2,3);
if (egyedi(t)) {
  return "true";
}
else {
  return "false";
}
```

Tömbelemek egyediségének vizsgálata üres elemek kihagyásával

Megvizsgálja, hogy a tömb nem üres elemei egyedi értékekkel rendelkeznek-e. True, amennyiben nincs két egyforma (nem üres) érték, egyébként false.

Normál alak:

egyediNemUresek(tomb)

Visszatérési érték:

Logikai

Példa:

```
var t = new Array('a', 'b', '', '', '');
if (egyediNemUresek(t)) {
  // erre az agra kerul a vezerles
  return "true";
}
else {
  return "false";
}
```

Tömbelemek egyediségének vizsgálata üres és 0 értékű elemek kihagyásával

Megvizsgálja, hogy a tömb nem üres elemei egyedi értékekkel rendelkeznek-e. True, amennyiben nincs két egyforma (nem üres) érték, egyébként false.

Normál alak:

egyediNemNullak(tomb)

Visszatérési érték:

Logikai

Példa:

```
var t = new Array(1, 2, 3, 0, 0);
if (egyediNemNullak(t)) {
  // erre az agra kerul a vezerles
  return "true";
}
else {
  return "false";
}
```

Érték meglétének vizsgálata a tömbben

Megvizsgálja, hogy az érték szerepel-e a tömbben. True, ha az érték nem szerepel a tömb elemei között, egyébként false.

Normál alak:

egyedi(tömb, érték)

Visszatérési érték:

Logikai

Példa:

```
var t = new Array(1,2,3);
if (egyedi(t,4)) {
  return "true";
}
else {
  return "false";
}
```

Ütközések vizsgálata

Visszaadja a tömbben levő elemek közül azoknak az indexét, amelyek értéke pontosan megegyezik az értékkel.

Normál alak:

utkozesek(tömb, érték)

Visszatérési érték:

Ütköző tömbindexekből képzett vektor.

Példa:

```
var t = new Array(1,2,3,2);
return utkozesek(t,2);
```

Kitöltöttség ellenőrzés

Nullát vagy kitöltetlen mezőket figyel, ha mind üres, vagy null, akkor hamis, egyébként igaz.

A kitöltöttség ellenőrzésénél az alábbiakat veszi figyelembe:
* a mező szerkeszthető legyen és üres, vagy null értéket tartalmazzon;
Nem vizsgálja azokat a mezőket, ami
* nem szerkeszthető,
* megszemélyesítéssel, előnyomással vagy alapértelmezetten kapott értéket.
Két megjegyzés:
* ha nincs a feltételeknek megfelelő, vizsgálandó komponens (pl. nincs szerkeszthető komponens), akkor is true-t ad vissza
* azzal a feltételezéssel éltünk, hogy az előnyomott komponens nem szerkeszthető

Teljes kérdőív szinten ellenőriz az minden oszlopnevet tartalmazó mezőt

rekurzívan megvizsgálja a teljes kérdőív valamennyi szabálynak megfelelő mezőjét.

Normál alak:

kerdoiv.kitoltott()

Visszatérési érték:

Logikai

Példa:

```
if (kerdoiv.kitoltott()) {
  return "true";
}
else {
  return "false";
}
```

Adott komponensen belül ellenőriz minden oszlopnevet tartalmazó mezőt

rekurzívan megvizsgálja a megadott konténer (és gyerek konténereik) valamennyi szabálynak megfelelő mezőjét.

Normál alak:

kerdoiv.kitoltott(panel vagy fejezet azonosíto)

Visszatérési érték:

Logikai

Példa:

```
if (kerdoiv.kitoltott("SzabadPanel16")) {
  return "true";
}
else {
  return "false";
}
```

Adott komponensen belül ellenőrizi az adott oszlopnevű tartalmazó mezőt

rekurzívan megvizsgálja a megadott konténer (és gyerek konténereik) valamennyi megadott oszlopnevű, szabálynak megfelelő mezőjét. Az oszlopnév paraméter fix szöveg, vagy reguláris kifejezés egyaránt lehet.

Normál alak:

kerdoiv.kitoltott(oszlopnév, panel vagy fejezet azonosíto)

Visszatérési érték:

Logikai

Példa:

```
if (kerdoiv.kitoltott("OSZLOP2","fejezet0")) {
  return "true";
}
else {
  return "false";
}
```

Teljes kitöltöttség ellenőrzése

A kitöltöttség ellenőrzéshez hasonlóan létezik függvény annak vizsgálatára, hogy minden mező tekintetében teljesülnek-e a kitöltöttség ellenőrzésnél definiált kritériumok. A vizsgálati szempontok a két esetben azonosak, de itt csak akkor ad igaz értéket a függvény, ha minden mezőre teljesül, egyébként (ha csak egy üres is van) hamis.

A függvény fejléce: kerdoiv.mindKitoltott()

Paraméterezését tekintve a lehetőségek megegyeznek a kerdoiv.kitoltott() függvény korábban definiált változataival.

KF csomag függvényei – ellenőrzés függvények

Szám Ellentettje

a * (-1)

Normál alak:

kf.kivon(a)

Visszatérési érték:

Float

Példa:

```
kf.kivon(1)
```

Eredmény: -1

Tetszőleges számú paraméter összeadása

x1 + x2 + ... + xN

Normál alak:

kf.osszeg(x1,...,xN)

Visszatérési érték:

Float

Példa:

```
kf.osszeg(1,5,7,12
```

Eredmény: 25

Osztás

Két szám hányadosa

Normál alak:

kf.osztas(a,b)

Visszatérési érték:

Float

Példa:

```
kf.osztas(6,3)
```

Eredmény: 2

Szorzás

Két szám szorzata

Normál alak:

kf.szorzas(a,b)

Visszatérési érték:

Float

Példa:

```
kf.szorzas(4,5)
```

Eredmény: 20

Százalék számítás

Az első érték hány százaléka a másodiknak

Normál alak:

kf.szazalek(a,b)

Visszatérési érték:

Float

Példa:

```
kf.szazalek(2,4)
```

Eredmény: 50

Egyenlőség vizsgálat

Megvizsgálja, hogy a két érték egyenlő-e.

Normál alak:

kf.egyenlo(a,b)

Visszatérési érték:

Egyenlőség esetén 0, egyébként 1.

Példa:

```
kf.egyenlo(1,1)
```

Eredmény: 0

Egyenlőtlenség vizsgálat

Megvizsgálja, hogy a két érték egyenlőtlen-e.

Normál alak:

kf.negyenlo(a,b)

Visszatérési érték:

Egyenlőség esetén 1, egyébként 0.

Példa:

```
kf.negyenlo(1,1)
```

Eredmény: 1

Nagyobb vizsgálat

Megvizsgálja, hogy az első érték nagyobb-e a másodiknál

Normál alak:

kf.nagyobb(a,b)

Visszatérési érték:

ha a>b, akkor 0 egyébként 1

Példa:

```
kf.nagyobb(4,3)
```

Eredmény: 0

Nagyobb-Egyenlő vizsgálat

Megvizsgálja, hogy az első érték nagyobb vagy egyenlő-e a másodiknál

Normál alak:

kf.nagyobbe(a,b)

Visszatérési érték:

ha a>=b, akkor 0 egyébként 1

Példa:

```
kf.nagyobb(3,3)
```

Eredmény: 0

Kisebb vizsgálat

Megvizsgálja, hogy az első érték kisebb-e a másodiknál

Normál alak:

kf.kisebb(a,b)

Visszatérési érték:

ha a<b vagy nvl(a,0)=nvl(b,0), akkor 0 egyébként 1

Példa:

```
kf.kisebb(3,5)
```

Eredmény: 0

Intervallumban elhelyezkedés vizsgálat

Megvizsgálja, hogy az első érték a második és a harmadik között helyezkedik-e el a számegyenesen, illetve egyenlő-e valamelyikkel.

Normál alak:

kf.kozott(a, ah, fh)

Visszatérési érték:

Ha ah<=a<=fh, akkor 0 egyébként 1

Példa:

```
kf.kozott(12, 10, 20)
```

Eredmény: 0

Intervallumban elhelyezkedés vagy Üresség vizsgálat

Megvizsgálja, hogy az első érték a második és a harmadik között helyezkedik-e el a számegyenesen, egyenlő-e valamelyikkel, illetve 0-e mind a három változó.

Normál alak:

kf.kozott0(a, ah, fh)

Visszatérési érték:

Ha ah<=a<=fh vagy a=ah=fh=0/üres , akkor 0 egyébként 1

Példa:

```
kf.kozott0(0, 0, 0)
```

Eredmény: 0

Kitöltött vagy Nulla vizsgálat

Megvizsgálja tetszőleges számú paraméterre, hogy mind ki van-e töltve (vagy nagyobb, mint nulla), illetve egy sincs-e kitöltve.

Normál alak:

kf.kelladat(x1,...,xN)

Visszatérési érték:

Ha mind ki van töltve (vagy nagyobb, mint nulla) vagy egy sincs kitöltve, akkor 0, egyébként 1.

Példa:

```
kf.kelladat(1, 2 ,0)
```

Eredmény: 1

Negativitás vizsgálat

Megvizsgálja, hogy negatív-e az érték.

Normál alak:

kf.nemneg(a)

Visszatérési érték:

Ha a>=0, akkor 0 egyébként 1

Példa:

```
kf.nemneg(4)
```

Eredmény: 0

hkod

Megvizsgálja, hogy az érték nagyobb-e, mint 2.

Normál alak:

kf.hkod(a)

Visszatérési érték:

Ha a>2, akkor 0 egyébként a.

Példa:

```
kf.hkod(3)
```

Eredmény: 0

Abszolút-érték vizsgálat 1

Megvizsgálja, hogy az érték abszolút-értéke nagyobb-e, mint nulla.

Normál alak:

kf.van(a)

Visszatérési érték:

Ha abs(a) > 0 akkor0, egyébként 1

Példa:

```
kf.van(1)
```

Eredmény: 0

Abszolút-érték vizsgálat 2

Megvizsgálja, hogy az érték abszolút-értéke nagyobb-e, mint nulla.

Normál alak:

kf.vanb(a)

Visszatérési érték:

Az abs(a) > 0 logikai eredményét adja vissza (true/false)

Példa:

```
kf.vanb(1)
```

Eredmény: true

Februártól vizsgálat

Megvizsgálja, hogy az érték nagyobb-e mint 1 (a januári hónap sorszáma).

Normál alak:

kf.februartol(a)

Visszatérési érték:

Az a>1, illetve a>”01”, logikai eredményét adja vissza (true/false)

Példa:

```
kf.februartol(2)
```

Eredmény: true

KF2 csomag függvényei

A KF csomag használatát könnyítendő kidolgozásra került egy KF2 csomag is, ami pontosan ugyanazokat a függvényeket biztosítja azonos szintaktikával, mint a KF csomag, viszont nem közvetlenül vizsgálandó értékeket várnak paraméterként, hanem az azonosítókat sztring formátumban. A KF2 csomag függvényei a kapott azonosítókra a getErtek függvényt használják. Pl.:kf.kisebb(3,5) helyett kf.kisebb("azonosito3","azonosito5")

FF csomag függvényei – ellenőrzés feltétel függvények

Abszolút-érték vizsgálat 1

Megvizsgálja, hogy az érték abszolút-értéke nagyobb-e, mint nulla.

Normál alak:

ff.van(a)

Visszatérési érték:

Ha abs(a) > 0, akkor 0, egyébként 3

Példa:

```
ff.van(0)
```

Eredmény: 3

Abszolút-érték vizsgálat 2

Megvizsgálja, hogy az érték abszolút-értéke egyenlő-e nullával.

Normál alak:

ff.nincs(a)

Visszatérési érték:

Ha abs(a) = 0, akkor 0, egyébként 3

Példa:

```
ff.nincs(2)
```

Eredmény: 3

Üresség vizsálat

Megvizsgálja, hogy az érték "null", azaz üres-e.

Normál alak:

ff.ures(a)

Visszatérési érték:

Ha a = null, akkor 0 egyébként 3

Példa:

```
ff.ures(null)
```

Eredmény: 0

Nem-üresség vizsgálat

Megvizsgálja, hogy az érték "null", azaz üres-e.

Normál alak:

ff.nemures(a)

Visszatérési érték:

Ha a != null, akkor 0 egyébként 3

Példa:

```
ff.nemures(3)
```

Eredmény: 0

Egyenlőség vizsgálat

Megvizsgál 2 vagy 3 paramétert, hogy egyenlőek-e.

Normál alak:

ff.egyenlo(a,b,c)

Visszatérési érték:

2 vagy 3 paraméterre, teljes egyenlőség esetén 0, egyébként 3

Példa:

```
ff.egyenlo(2,2,3)
```

Eredmény: 3

Nem-egyenlőség vizsgálat

Megvizsgál 2 paramétert, hogy egyenlőek-e.

Normál alak:

ff.negyenlo(a,b)

Visszatérési érték:

Ha a=b, akkor 3, egyébként 0

Példa:

```
ff.negyenlo(1,1)
```

Eredmény: 3

Nagyobb vizsgálat

Megvizsgálja, hogy az első érték nagyobb-e a másodiknál.

Normál alak:

ff.nagyobb(a,b)

Visszatérési érték:

Ha a>b, akkor 0 egyébként 3

Példa:

```
ff.nagyobb(2,1)
```

Eredmény: 0

Nagyobb-Egyenlő vizsgálat

Megvizsgálja, hogy az első érték nagyobb vagy egyenlő-e a másodikkal.

Normál alak:

ff.nagyobbe(a,b)

Visszatérési érték:

Ha a>=b, akkor 0 egyébként 3

Példa:

```
ff.nagyobbe(3,4)
```

Eredmény: 3

FF2 csomag függvényei

Az FF csomag használatát könnyítendő kidolgozásra került egy FF2 csomag is, ami pontosan ugyanazokat a függvényeket biztosítja azonos szintaktikával, mint az FF csomag, viszont nem közvetlenül vizsgálandó értékeket várnak paraméterként, hanem az azonosítókat sztring formátumban. Az FF2 csomag függvényei a kapott azonosítókra a getErtek függvényt használják.

Kérdőívkitöltőben használt színkódok

Beviteli mezők háttérszínei:

- Normál: #F0F0F0
- Fókuszált: #FFF6D5
- Nem szerkeszthető mezők háttérszíne: #B7DBFF