**Rekursīvas funkcijas**

Dažos gadījumos funkcija būs jāizsauc no pašas funkcijas. Tas var būt lielisks risinājums dažām diezgan sarežģītām problēmām. Tomēr ir dažas lietas, kas jāpatur prātā. Ko, jūsuprāt, dara šis kods?

**Izmēģini piemēru Nr.1 darbībā!**

**Piemērs Nr. 1**

Attēls, kurā ir teksts, fonts, ekrānuzņēmums, dizains

Mākslīgā intelekta ģenerēts saturs var būt nepareizs.

Kods izdrukā vērtību 3, pēc tam to samazina — un funkcija turpina darboties. Kāpēc tā neapstājas? Jo mēs neesam norādījuši, kad tai apstāties. Piemērā Nr. 2 ir uzlabota versija:

**Piemērs Nr. 2**

Attēls, kurā ir teksts, fonts, ekrānuzņēmums

Mākslīgā intelekta ģenerēts saturs var būt nepareizs.

Funkcija turpinās izsaukt pati sevi tik ilgi kamēr parametra **nr** vērtība būs lielāka par 0.

Kad mēs funkciju izsaucam rekursīvi, katrā funkcijas izpildes reizē tā kļūst par vienu funkciju “dziļāka”. Pirmais funkcijas izsaukums tiks izpildīts kā pēdējais.

**Piemērs Nr. 3**

Attēls, kurā ir teksts, ekrānuzņēmums, fonts

Mākslīgā intelekta ģenerēts saturs var būt nepareizs.

**Darbības shēma piemēram Nr.3:**

Rekursīva funkcija **rekinat2(3)** saņemta;

Rekursīva funkcija **rekinat2(2)** saņemta;

Rekursīva funkcija **rekinat2(1)** saņemta;

Rekursīva funkcija **rekinat2(0)** saņemta;

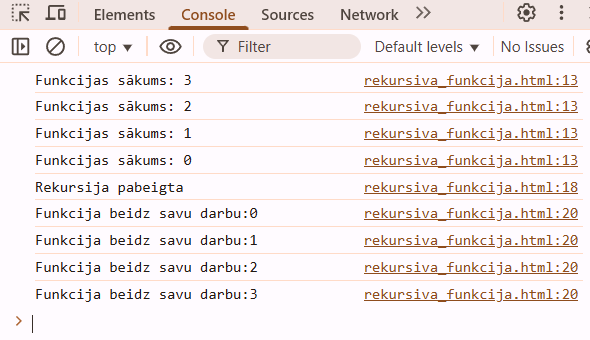
**rekinat2(0)** izpilde pabeigta;

**rekinat2(1)** izpilde pabeigta;

**rekinat2(2)** izpilde pabeigta;

**rekinat2(3)** izpilde pabeigta;

Konsolē tiks rādīts sekojošs rezultāts:



Dažās situācijās rekursīvas funkcijas ir lielisks risinājums. Ja jums ir nepieciešams izsaukt vienu un to pašu funkciju atkal un atkal ciklā, varat apsvērt rekursijas izmantošanu. Rekursija ir noderīga arī kaut ko meklējot (veicot meklēšanu). Tā vietā, lai cikliski veiktu visas darbības vienas funkcijas ietvaros, varat sadalīt funkciju un izsaukt to vairākas reizes funkcijas iekšienē.

Tomēr ir svarīgi paturēt prātā, ka kopumā rekursijas veiktspēja ir nedaudz sliktāka nekā parasta cikla iterācijas veiktspēja. Tāpēc, ja rekursijas izmantošana palēnina jūsu lietojumprogrammu, ir vērts apsvērt citu pieeju.

**Apskatiet faktoriāla aprēķināšanu, izmantojot rekursīvās funkcijas, šajā uzdevumā!**

**Uzdevums: faktoriāla aprēķināšana**

Bieži sastopama problēma, ko varam atrisināt, izmantojot rekursiju, ir faktoriālu aprēķināšana.

Īsumā atsvaidzināsim savas matemātiskās zināšanas par faktoriāliem.

|  |
| --- |
| Matemātikā par naturāla skaitļa **n ≥ 1** faktoriālu sauc visu naturālo **skaitļu no** **1 līdz n reizinājumu**. To apzīmē ar **n!**.  **Piemēram, 3! = 1 · 2 · 3 = 6, 4! = 1 · 2 · 3 · 4 = 24 utt.** |

Kā rekursīvās funkcijas var palīdzēt mums atrast faktoriālu? Šajā vingrinājumā mēs izmantosim rekursiju, lai aprēķinātu skaitliskas vērtības faktoriāla rezultātu, kas dots kā funkcijas arguments. Mēs izsauksim funkciju zemāk un zemāk, līdz sasniegsim 0.

1. Izveidojiet funkciju, kas satur nosacījumu, kurš pārbauda, ​​vai argumenta vērtība ir vienāda ar 0.
2. Ja parametrs ir 0 (nulle), funkcijai jāatgriež vērtība 1. Pretējā gadījumā tai jāatgriež argumenta vērtība, kas reizināta ar pašas funkcijas atgriezto vērtību, no argumenta vērtības atņemot vienu. Tas sāks bloku, kas darbosies līdz vērtība kļūs par 0.
3. Izsauciet funkciju, kā argumentu norādot jebkuru skaitli, kura faktoriālu vēlaties uzzināt. Kodam jādarbojas ar jebkuru skaitli, kas sākotnēji tika nodots funkcijai, samazinot vērtību līdz 0 un parādot aprēķina rezultātus ekrānā. Varat arī iekļaut komandu console.log(), lai izdrukātu funkcijas argumenta pašreizējo vērtību funkcijas izsaukšanas brīdī.
4. Mainiet skaitli, lai redzētu, kā tas ietekmē funkcijas rezultātus.

Uzdevuma izpildes rezultāts:

Attēls, kurā ir teksts, ekrānuzņēmums, fonts

Mākslīgā intelekta ģenerēts saturs var būt nepareizs.