JAVASCRIPT YORDAMIDA TUB SONLAR KETMA-KETLIGINI ANIQLASH ALGORITMLARI

To`xtayeva Robiya Ravshanbek qizi
Andijon davlat universiteti talabasi

Annotatsiya: Tub sonlar, faqat 1 ga va o'ziga bo'linadigan butun sonlar turli sohalarda, jumladan, matematika va informatikada hal qiluvchi rol o'ynaydi. Ushbu maqolada biz tub sonlar ketma-ketligini aniqlashning samarali usullarini o'rganish uchun JavaScript sohasini o'rganamiz. Asosiy algoritmlarni tushunish va JavaScript kuchidan foydalanish orqali biz ilovalarimizdagi tub sonlarning imkoniyatlarini ochishimiz mumkin.

Kalit so`zlar: Algaritm, statistika, Iterativ yondashuv, funksiannallik, web-sahifa, kutubxona

JavaScript: Ko'pincha Internet tili sifatida qaraladigan JavaScript veb-sahifalarda dinamik va interaktiv tarkibni yaratish imkoniyatini beradi. Bu brauzerda ishlaydigan ko'p qirrali til bo'lib, ishlab chiquvchilarga Document Object Modelini (DOM) boshqarish va foydalanuvchilarning o'zaro ta'siriga javob berish imkonini beradi.

Shu nuqtayi nazardan, tub sonlar ketma-ketligi uchun dastur yaratishga sayohatimiz uchun zarur bo'lgan JavaScriptning asosiy tushunchalarini o'rganamiz. Biz bu dasturni tuzish uchun algoritmdan ham foydalanishimiz kerak. Keling birgalikda Algoritm o`zi nima? va nima uchun kerakligini ko`rib chiqamiz.

Tub sonlarni yaratish uchun eng qadimiy, ammo samarali algoritmlardan biri Eratosfen elakidir. Qadimgi yunon matematigi Eratosfen nomi bilan atalgan bu algoritm tizimli ravishda har bir tub sonning ko‘paytmalarini yo‘q qiladi va oxirida faqat tub sonlarni qoldiradi.

Keling, JavaScript-da Eratosthenes elakini amalga oshiramiz:

```javascript
function sieveOfEratosthenes(limit) {
    const primes = [];
    const isPrime = Array(limit + 1).fill(true);

    for (let num = 2; num <= Math.sqrt(limit); num++) {
        if (isPrime[num]) {
            primes.push(num);

            for (let multiple = num * num; multiple <= limit; multiple += num) {
                isPrime[multiple] = false;
            }
        }
    }

    for (let num = Math.max(2, Math.sqrt(limit) + 1); num <= limit; num++) {
        if (isPrime[num]) {
            primes.push(num);
        }
    }

    return primes;
}

const primeSequence = sieveOfEratosthenes(50);
console.log("Prime Numbers:", primeSequence);
```

Ushbu JavaScript funksiyasi belgilangan chegaragacha tub sonlar ketma-ketligini yaratish uchun Eratosfen elakidan foydalanadi.

Javascript va Tub sonlar: Ko'pincha veb-ishlab chiqish bilan bog'liq bo'lgan JavaScript ko'p qirrali dasturlash tili bo'lib, turli vazifalar, jumladan, matematik hisoblashlar uchun mos keladi. U tub sonlar bilan muammosiz ishlash uchun zarur vositalar va moslashuvchanlikni ta'minlaydi.

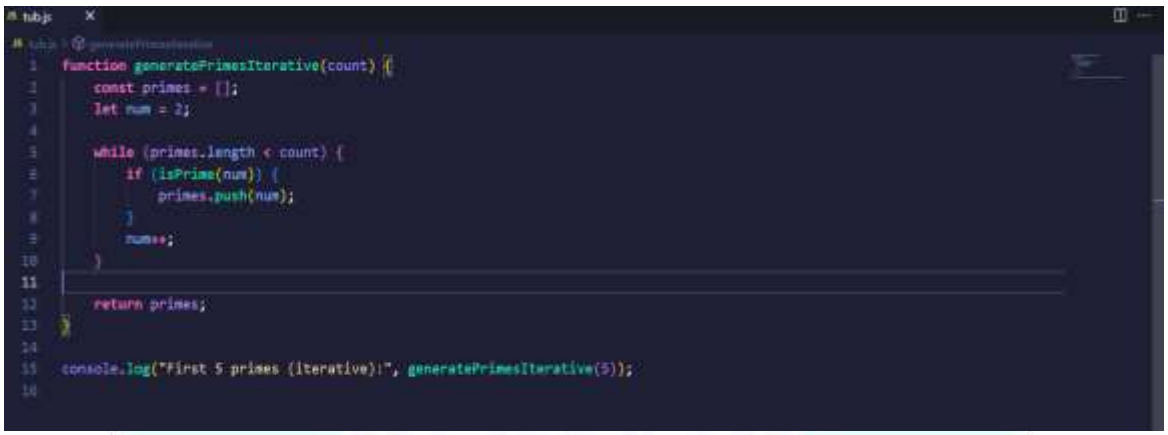
Mana, berilgan raqamning asosiy ekanligini tekshirish uchun JavaScriptning asosiy funksiyasi.

```javascript
function isPrime(number) {
    if (number <= 1) return false;
    if (number <= 3) return true;

    if (number % 2 === 0 || number % 3 === 0) return false;

    for (let i = 5; i * i <= number; i += 6) {
        if (number % i === 0 || number % (i + 2) === 0) {
            return false;
        }
    }

    return true;
}

console.log("Is 17 prime?", isPrime(17));
console.log("Is 25 prime?", isPrime(25));
```

Bu funksiya asosiy birlamchilikni tekshirishdan foydalanadi va JavaScriptda tub sonlar ketma-ketligi bilan ishlashda muhim ahamiyatga ega.

Bosh sonlar ketma-ketligini amalga oshirish. Bosh sonlar ketma-ketligini yaratish turli usullar va yondashuvlarni o'rganishni o'z ichiga oladi. Keling, JavaScriptda interaktiv va rekursiv echimlarni ko'rib chiqaylik.

Interaktiv yondashuv:



Bu funksiya interaktiv ravishda birinchi "hisoblash" tub sonlarini hosil qiladi.

JavaScriptda tub sonlar bilan ishlashda ba'zi qiyinchiliklar paydo bo'lishi mumkin. Keling, umumiy muammolarni va ularning yechimlarini muhokama qilaylik.

Katta raqamlar bilan ishlash. JavaScript butun sonlar aniqligi chegarasiga ega. Juda katta tub sonlar bilan ishlashda BigInt kabi ixtisoslashtirilgan kutubxonalardan foydalanish yoki katta sonlar ustida arifmetik operatsiyalar uchun maxsus algoritmlarni amalga oshirish haqida o'ylab ko'ring.

Ishlashni optimallashtirish. Bosh sonlar ketma-ketligining o'lchami ortib borishi bilan algoritmlarning ishlashi hal qiluvchi ahamiyatga ega bo'ladi. Parallellashtirish kabi optimallashtirishlarni amalga oshirish tub sonlarni yaratish tezligini sezilarli darajada oshirishi mumkin.

Xotira iste'moli. Xotiradan foydalanish, ayniqsa katta tub sonlar ketma-ketligini yaratishda tashvish tug'dirishi mumkin. Samarali ma'lumotlar tuzilmalari va algoritmlari xotirani ehtiyotkorlik bilan boshqarish bilan birga ushbu muammoni hal qilishga yordam beradi.

Xulosa qilib aytganda, JavaScript tub sonlar ketma-ketligini aniqlash uchun ko'p qirrali platformani taqdim etadi. Eratosfen elaklari kabi algoritmlarni amalga oshirish va JavaScript-dan foydalanish orqali.

Foydalanilgan adabiyotlar:

1. Algaritm va dasturlash asoslari. Cho'lpon nomidagi nashriyot-matbaa ijodiy uyi: Toshkent — 2013.
2. Matematikadan qo`llanma: Toshkent «Yangi asr avlodi» 2006.
3. “Zamonaviy Javascript darsligi” Ilya Kantor.
4. https://www.wikipedia.org/
5. https://www.w3schools.com/