

JAVASCRIPT DASTURLASH TILIDA ZAMONAVIY VEB-ILOVALAR ARXITEKTURASI VA UNUMDORLIGINI OSHIRISHNING ILMIY-AMALIY ASOSLARI

Kuldasheva Feruza Kurdoshevna

TDIU "International Business" AL o'qituvchisi

Annotatsiya. Ushbu maqolada JavaScript dasturlash tili asosida veb-illovalar arxitekturasini barqaror, kengaytiriladigan va unumdor qilish muammosi tadqiq etiladi. Tadqiqotda adabiyotlar tahlili, taqqoslash va tajribaviy prototiplash qo'llanildi. Natijada modul tuzilma, asinxron boshqaruv va optimallashtirish tamoyillari tizimlashtirildi hamda amaliy tavsiyalar ishlab chiqildi. Maqolaning maqsadi JavaScript muhitida mijoz va server qismida ishlovchi ilovalarda samaradorlikni oshirishga xizmat qiladigan arxitektura yechimlarini asoslashdir. Metodlar sifatida konseptual tahlil, kod darajasidagi kuzatuv va ishlash ko'rsatkichlarini taqqoslash ishlatildi. Natijalar asinxron oqimlarni boshqarish, resurslarni tejash va modullashtirish bo'yicha yondashuvlarning ustunliklarini ko'rsatadi. Tadqiqot JavaScript ekotizimida keng tarqalgan arxitektura muammolari, xususan, murakkablikning o'sishi, kechikishlar va xotira sarfi bilan bog'liq masalalarni yoritadi. Metodologiya adabiy manbalar sharhi va prototip yechimlar orqali tekshirishga tayandi. Ilmiy hissa sifatida arxitektura qarorlarini tanlash mezonlari va optimallashtirishning izchil modeli taklif etildi.

Kalit so'zlar: JavaScript, veb-ilova, arxitektura, asinxron dasturlash, unumdorlik, modullashtirish, optimallashtirish

Abstract. This article examines the problem of making the architecture of web applications based on the JavaScript programming language stable, scalable and productive. The research used literature analysis, comparison and experimental prototyping. As a result, the principles of modular structure, asynchronous management and optimization were systematized and practical recommendations were developed. The purpose of the article is to substantiate architectural solutions that improve the performance of client-side and server-side applications running in a JavaScript environment. Conceptual analysis, code-level observation, and performance benchmarking were used as methods. The results demonstrate the advantages of asynchronous flow management, resource conservation, and modularization approaches. The study addresses common architectural problems in the JavaScript ecosystem, in particular, issues related to complexity growth, latency, and memory consumption. The methodology is based on a literature review and verification through prototype solutions. As a scientific contribution, criteria for selecting architectural decisions and a consistent model for optimization are proposed.

Keywords: JavaScript, web application, architecture, asynchronous management, performance, modularization, optimization

Kirish

JavaScript dasturlash tili so'nggi o'n yilliklarda veb texnologiyalarning markaziy komponentiga aylanishi natijasida nafaqat interfeys darajasidagi ssenariylar, balki server tomoni, real vaqt tizimlari va ko'p platformali ilovalar uchun ham universal muhit sifatida shakllandi. Bu holat amaliy jihatdan tezkor ishlab chiqish, boy kutubxonalar ekotizimi va bir xil til orqali to'liq stek yechimlar yaratish imkonini bersa- da, ilmiy va muhandislik nuqtayi nazaridan murakkablikning keskin ortishi, arxitektura qarorlarining standartlashmaganligi, hamda unumdorlik va ishonchlikka bevosita ta'sir qiluvchi asinxron jarayonlar boshqaruvi kabi muammolarni keskinlashtirdi. Veb-ilovalar ko'lamining o'sishi bilan JavaScript kod bazalarida bog'liqliklar zichlashadi, holatni boshqarish qiyinlashadi, tarmoq bilan ishlashdagi kechikishlar foydalanuvchi tajribasini yomonlashtiradi, hamda xotira va hisoblash resurslari sarfi barqarorlikka putur yetkazadi. Mazkur omillar JavaScript dasturlashni faqat sintaksis va kutubxona bilimi bilan cheklab bo'lmashligini, balki arxitektura tamoyillari va optimallashtirish strategiyalarini ilmiy asosda tahlil qilish zarurligini ko'rsatadi.

Mavzuning dolzarbligi shundaki, JavaScript ekotizimi doimiy yangilanib boradi va tilning o'zi ham, uni qo'llash muhiti ham katta tezlikda evolyutsiyalanadi. Natijada dasturchilar ko'pincha "tez ishlaydigan" yechimni emas, balki "tez yoziladigan" yechimni tanlaydi, bu esa keyinchalik texnik qarzning ortishiga olib keladi. Ilmiy adabiyotlarda veb arxitekturalari, asinxron dasturlash va ishlash ko'rsatkichlari bo'yicha qarashlar mavjud bo'lsa-da, amaliy loyihalash qarorlarini tanlash mezonlarini bir tizimga soluvchi, mijoz va server tomoni JavaScript o'rtasidagi konseptual uzviylikni ko'rsatadigan hamda optimallashtirishni faqat mikro darajadagi kod o'zgartirish emas, balki makro darajadagi arxitektura masalasi sifatida izohlaydigan yondashuvlar yetarli darajada umumlashtirilmagan [1; 4]. Shu sababli tadqiqot muammosi JavaScript asosidagi zamonaviy veb-ilovalarda arxitektura va asinxron boshqaruv qarorlarini qanday mezonlar asosida tanlash, hamda unumdorlikka erishish uchun qaysi optimallashtirish strategiyalarini bir-biri bilan uyg'unlashtirish mumkinligi masalasida jamlanadi.

Maqolaning maqsadi

JavaScript dasturlash tilida veb-ilovalar arxitekturasini barqaror va kengaytiriladigan qilish bilan birga, ishlash samaradorligini oshirishga xizmat qiladigan ilmiy-amaliy tamoyillarni asoslashdan iborat. Ushbu maqsad doirasida bir nechta vazifalar qo'yiladi: birinchidan, JavaScript muhitida arxitektura tanlovlariga ta'sir qiluvchi asosiy omillarni konseptual tahlil qilish; ikkinchidan, asinxron bajarilish modellari va ularning kechikish, xotira sarfi hamda javob qaytarish tezligiga ta'sirini tizimlashtirish; uchinchidan, modullashtirish, qatlamlash va holatni boshqarish bo'yicha

qarorlarning texnik qarz va testlanish darajasiga ta'sirini aniqlash; to'rtinchidan, amaliy prototiplash orqali optimallashtirish yo'nalishlarini umumlashtirib, tanlash mezonlarini taklif etish. Tadqiqotning ilmiy yangiligi JavaScript kontekstida arxitektura va unumdorlikni bir butun tizim sifatida ko'rib, optimallashtirishni arxitektura qarorlaridan ajratmasdan baholash modelini konseptual ravishda asoslash bilan belgilanadi.

Usullar

Tadqiqot metodologiyasi ko'p bosqichli va aralash yondashuvga tayangan bo'lib, unda nazariy tahlil, taqqoslash va tajribaviy prototiplash usullari uyg'unlashtirildi. Avvalo, JavaScript va veb arxitekturasi bo'yicha xalqaro, rus va o'zbek ilmiy-amaliy manbalar tanlab olindi hamda ularning konseptual pozitsiyalari, tavsiya etilgan arxitektura naqshlari va ishlash samaradorligini baholashga oid yondashuvlari mazmunan tahlil qilindi [2; 6]. Ushbu tahlil mavzuning ilmiy kontekstini aniqlash, mavjud qarashlar o'rtasidagi farqlarni ko'rsatish va tadqiqot bo'shlig'ini asoslash uchun zarur bo'ldi. Nazariy tahlilda "murakkablikni boshqarish", "asinxron oqimni nazorat qilish", "holatni izolyatsiyalash", "modul chegaralari" kabi tushunchalar aniqlashtirildi va ular asosida baholash mezonlari shakllantirildi.

Keyingi bosqichda taqqoslash metodi qo'llanilib, JavaScript muhitida amaliyotda keng uchraydigan arxitektura yondashuvlari, xususan, qatlamli arxitektura, komponentga yo'naltirilgan tuzilma, xizmatlar orqali ajratish va hodisaga asoslangan modelning afzallik hamda cheklovlari bir xil mezonlar bo'yicha solishtirildi [1; 7]. Taqqoslashning maqsadi bitta "eng yaxshi" arxitekturani e'lon qilish emas, balki kontekstga mos tanlovni asoslaydigan shartlarni ko'rsatishdan iborat bo'ldi. Shu sababli mezonlar sifatida o'qiluvchanlik va qo'llab-quvvatlash qulayligi, testlanish imkoniyati, kengaytirish narxi, hamda ishlash samaradorligiga bilvosita yoki bevosita ta'sir ko'rsatadigan omillar tanlandi.

Empirik qismda tajribaviy prototiplash yondashuvi qo'llanildi: tipik veb-ilova ssenariylariga mos keladigan kichik hajmli prototip modullar yaratilib, ularda asinxron chaqiriqlarni tashkil etish, resurslarni keshlash, kechikishlarni kamaytirish va xotira oqishlarini oldini olishga qaratilgan yechimlar sinab ko'rildi. Empirik kuzatuvlar natijalarini umumlashtirishda ishlash ko'rsatkichlari sifatida javob vaqti, asosiy oqimning band bo'lish darajasi, resurs so'rovlari soni va xotira iste'moli bilan bog'liq holatlar sifat jihatdan tahlil qilindi [3; 8]. Ushbu metodlar tanlovi tadqiqot maqsadiga mos keladi, chunki JavaScriptdagi unumdorlik muammolari faqat nazariy tavsif bilan to'liq ochilmaydi, shu bilan birga faqat o'lchov natijalarini keltirish ham arxitektura qarorlarining sabab-oqibat bog'lanishini anglatib bera olmaydi. Shuning uchun konseptual, taqqoslovchi va tajribaviy yondashuvlar birgalikda qo'llanib, xulosalarning asoslanganligi kuchaytirildi.

Natijalar va na'munalar

Tadqiqot natijalari JavaScript asosidagi veb-ilovalar arxitekturasi va unumdorligi o'rtasida

barqaror bog‘liqlik mavjudligini ko‘rsatdi: ko‘plab “sekin ishlash” holatlari mikro optimallashtirish yetishmasligidan emas, balki modul chegaralarining noaniqligi, holatning haddan tashqari globalizatsiyasi va asinxron oqimlarning boshqarilmasligi kabi arxitektura omillaridan kelib chiqadi. Birinchi natija sifatida arxitektura qarorlarini tanlash mezonlari tizimlashtirildi va ular uch darajaga ajratildi: domen mantiqini ajratish darajasi, bog‘liqliklarni boshqarish mexanizmi, hamda ishlashga ta’sir qiluvchi resurs oqimlari. Bu mezonlar asosida bir xil funksional talablar turlicha arxitektura bilan bajarilganda, qo‘llab-quvvatlash xarajati va ishlash ko‘rsatkichlarida sezilarli farq paydo bo‘lishi aniqlashtirildi.

Ikkinchi natija asinxron bajarilish modelining loyihaviy oqibatlarini aniqlash bilan bog‘liq bo‘ldi. JavaScriptning hodisalar aylanishi va vazifalar navbati bilan ishlashi, ayniqsa ko‘p tarmoqli so‘rovlar, fayl tizimi yoki tashqi servislar bilan integratsiya mavjud bo‘lgan holatlarda, kechikishlarni yashirin ravishda ko‘paytirishi mumkinligi ko‘rsatildi. Tadqiqot prototiplarida asinxron chaqiriqlarni tartibsiz zanjirlash o‘rniga, aniq oqim boshqaruvi va xatoliklarni markazlashtirilgan qayta ishlash qo‘llanganda, javob qaytarish vaqti barqarorlashishi va nosozliklar diagnostikasi soddalashishi kuzatildi. Natijada asinxron oqimlar uchun “kuzatuvchanlik” tamoyili, ya’ni bajarilishning muhim nuqtalarini mantiqiy monitoring qilish imkonini beradigan arxitektura qatlamini oldindan ko‘zda tutish, unumdorlikka bilvosita ijobiy ta’sir berishi tizimlashtirildi.

Uchinchi natija modullashtirish va paketlash strategiyalarining ishlashga ta’sirini ochib berdi. Kod bazasi kichik bo‘lganida modullashtirishning ortishi ba’zan ortiqcha abstraksiyaga o‘xshab ko‘rinsa-da, murakkab ilovalarda modul chegaralarining to‘g‘ri belgilanishi takroriy hisoblashlarni kamaytirishi, keraksiz qayta render yoki qayta hisoblashlarni cheklashi va sinovlarni izolyatsiyada o‘tkazish imkonini kengaytirishi aniqlandi. Prototiplash jarayonida modul chegaralari domen obyektlari va foydalanish holatlari atrofida qurilganda, bog‘liqliklar kamayib, o‘zgarishlarning tarqalish radiusi qisqargani natijasida regressiya xatolarining kamayishi bilan birga ishlashdagi tebranishlar ham pasayishi kuzatildi. Shu tariqa modullashtirish faqat tashkiliy qulaylik emas, balki resurslar oqimini boshqarish vositasi sifatida ham namoyon bo‘ldi.

To‘rtinchi natija keshlash va ma’lumot almashinuvi strategiyalariga taalluqli bo‘lib, veb-ilovalarda tarmoq so‘rovlari soni va ma’lumotni qayta ishlash chastotasi o‘rtasidagi muvozanatni topish zarurligini ko‘rsatdi. Tadqiqotda keshlashning bevosita foydasi bilan birga uning noto‘g‘ri tashkil etilishi holatning eskirishi, mos kelmaslik va xotira ortiqcha sarfi kabi xavflarni keltirib chiqarishi aniqlashtirildi. Shuning uchun natijalar keshlashni umumiy qoida sifatida emas, balki ma’lumotning yangilanish tabiati, foydalanuvchi ssenariylari va ko‘p oynali ishlash kabi shartlar bilan bog‘langan qaror sifatida ko‘rishni taklif etadi. Natijada “ma’lumot hayot sikli”ga asoslangan keshlash mezonlari umumlashtirildi: qaysi qatlamda keshlash qilish, keshlash muddati va invalidatsiya signallarini qayerdan olish masalalari arxitektura dizaynining ajralmas qismi sifatida

belgilandi.

Munozara

Olingan natijalar JavaScript ekotizimida unumdorlik masalasi ko‘pincha noto‘g‘ri talqin qilinishini ko‘rsatadi, chunki amaliyotda u ko‘proq sintaksis darajasidagi “tezlashtirish” usullariga bog‘lanadi, holbuki tadqiqot arxitektura tanlovi asosiy determinantlardan biri ekanini asoslaydi. Bu xulosa veb ishlashiga oid klassik yondashuvlarda ko‘rsatilgan “tizim darajasida fikrlash” g‘oyasiga mos keladi: resurs oqimlari, kechikish manbalari va izolyatsiya darajasi birgalikda ko‘rilmaganda, lokal optimallashtirishlar kutilgan natijani bermaydi [6]. Shu nuqtayi nazardan, maqolada taklif etilgan mezonlar veb-ilovalarni loyihalashni shunchaki komponentlar yig‘indisi emas, balki o‘zaro ta’sir qiluvchi qatlamlar tizimi sifatida ko‘rishga yordam beradi.

Asinxron boshqaruv bo‘yicha natijalar JavaScriptning bir oqimli bajarilishi haqidagi ilmiy-amaliy qarashlarni yanada aniqlashtiradi. Flanagan tomonidan yoritilgan asinxron mexanizmlar, va‘da va hodisalar modeli amaliy qulayliklar bersa-da, murakkab tizimlarda kuzatuvchanlik bo‘lmasa, nosozliklar “yashirin” qolishi va kechikishlar domino effekti bilan ko‘payishi mumkinligi ta’kidlanadi [2]. Bizning natijalar aynan shu nuqtani arxitektura tamoyili sifatida mustahkamlaydi: asinxronlik faqat til imkoniyati emas, balki boshqaruv intizomini talab qiladigan dizayn masalasidir. Shu bilan birga, Node.js muhitiga oid manbalarda keltirilgan “bloklamaslik” konsepsiyasi ko‘pincha to‘g‘ri tushunilmaydi, chunki bloklamaydigan I/O mavjud bo‘lsa ham, hisoblashga og‘ir vazifalar yoki noto‘g‘ri navbatlash asosiy oqim bandligini oshiradi [4]. Tadqiqot prototiplari bu fikrni qo‘llab-quvvatlab, arxitekturada og‘ir hisoblashlarni ajratish, kechikishni yashirish emas, balki boshqarish zarurligini ko‘rsatadi.

Modullashtirish bo‘yicha xulosalar dasturiy injiniringdagi klassik pozitsiyalar bilan uyg‘unlashadi, chunki Martinning bog‘liqlik va mas’uliyatni ajratish haqidagi qarashlari modul chegaralari noto‘g‘ri belgilansa, tizim ham qo‘llab-quvvatlash, ham ishlash nuqtayi nazaridan zarar ko‘rishini ko‘rsatadi [7]. Bizning natijalarda modullashtirishning unumdorlikka ta’siri bevosita “kod bo‘linishi” bilan emas, balki holatni izolyatsiyalash va takroriy qayta hisoblashlarni cheklash orqali amalga oshishi aniqlandi. Shu jihatdan, o‘zbek amaliy adabiyotlarida dasturlashni o‘qitish va loyihalash madaniyati bilan bog‘liq yondashuvlar, ayniqsa, tuzilmalilik va qayta foydalanish g‘oyalarini kuchaytirish zarurligini ko‘rsatadi [9]. Tadqiqot ushbu qarashlarni JavaScript kontekstida arxitektura mezonlari bilan bog‘lab, modullashtirishni faqat ta’limiy tavsiya emas, balki ishlashga ta’sir qiluvchi ilmiy-amaliy mexanizm sifatida talqin qiladi.

Keshlash va ma’lumot hayot sikli bo‘yicha natijalar esa veb ishlashiga doir xalqaro amaliyotlarda uchraydigan qarashlarni to‘ldiradi: Souders ishlarida resurslarni optimallashtirish va keshlash strategiyalarining katta ta’siri ko‘rsatiladi, biroq keshlashning noto‘g‘ri invalidatsiyasi muammosi ko‘pincha “ikkinchi darajali” deb qaraladi [6]. Bizning tahlil shuni ko‘rsatdiki, aynan JavaScript

ilovalarda holatni boshqarish murakkablashgani sari keshlash noto‘g‘ri tashkil etilsa, foydalanuvchi tajribasiga salbiy ta‘sir ko‘rsatadigan mantiqiy xatolar ko‘payadi va bu holatni keyinchalik faqat texnik optimallashtirish bilan tuzatish qiyinlashadi. Shuning uchun keshlashni arxitektura qatlamlaridagi mas‘uliyatlar taqsimoti bilan birga ko‘rish zarur. Bu yondashuv rus tilidagi amaliy qo‘llanmalarda uchraydigan tizimli tahlil ruhiga mos bo‘lib, muammoni “alomat” darajasida emas, “sababi” darajasida hal qilishga yo‘naltiradi [1].

Umuman olganda, muhokama natijalarni mavjud adabiyotlar bilan taqqoslaganda, maqolaning asosiy farqi JavaScript unumdorligini baholashni fragmentar tavsiyalar yig‘indisi sifatida emas, balki arxitektura tanlovi, asinxron boshqaruv intizomi va resurs oqimlarini uyg‘unlashtiruvchi model sifatida ko‘rsatganidadir. Bu model amaliy loyiha ishlarida qarorlarni tezroq asoslash, texnik qarzni oldindan baholash va testlanish bilan unumdorlikni bir-biriga qarama-qarshi qo‘ymasdan, yagona dizayn maqsadlari sifatida qo‘yish imkonini beradi.

Xulosa

Maqolada JavaScript dasturlash tili asosida yaratiladigan zamonaviy veb- ilovalarda arxitektura tanlovi va unumdorlik o‘rtasidagi uzviy bog‘liqlik ilmiy-amaliy jihatdan asoslandi. Tadqiqot natijalari shuni ko‘rsatdiki, ishlash muammolarining sezilarli qismi kod darajasidagi mayda optimallashtirishlardan ko‘ra, modul chegaralarini belgilash, holatni izolyatsiyalash va asinxron oqimlarni boshqarish kabi arxitektura omillariga borib taqaladi. Asinxron bajarilishning kuzatuvchanligini ta‘minlash, xatoliklarni markazlashtirilgan qayta ishlash, domen mantiqini qatlamlarga ajratish va ma‘lumot hayot sikliga tayangan keshlash qarorlari barqaror samaradorlikka erishishda asosiy tamoyillar sifatida umumlashtirildi. Ushbu yondashuvning nazariy ahamiyati JavaScript ekotizimida optimallashtirishni arxitektura modeli bilan birga ko‘rish mezonlarini ishlab chiqishdan iborat bo‘lsa, amaliy ahamiyati loyihalash bosqichidayoq texnik qarz va ishlash xatarlarini kamaytirish imkonini berishidadir. Kelgusida tadqiqotlar JavaScript ilovalarda kuzatuvchanlik metrikalarini standartlashtirish, real vaqt rejimidagi tizimlar uchun oqim boshqaruvining formal mezonlarini aniqlashtirish va ko‘p platformali muhitlarda resurslardan foydalanish profilini solishtirish yo‘nalishlarida davom ettirilishi maqsadga muvofiq.

Foydalanilgan adabiyotlar:

- [1]. Симонов А. А. JavaScript. Шаблоны. Санкт-Петербург: Питер, 2019. 352 с.
- [2]. Flanagan D. JavaScript: The Definitive Guide. Sebastopol: O’Reilly Media, 2020. 706 p.
- [3]. Osmani A. Learning JavaScript Design Patterns. Sebastopol: O’Reilly Media, 2017. 254 p.
- [4]. Cantelon M., Harter M., Holowaychuk T., Rajlich N. Node.js in Action. Shelter Island: Manning Publications, 2017. 352 p.
- [5]. Кузнецов С. Д. Проектирование программных систем. Москва: Бином, 2018. 416 с.
- [6]. Souders S. High Performance Web Sites. Sebastopol: O’Reilly Media, 2007. 166p.

- [7]. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Pearson, 2017. 432 p.
- [8]. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing. Los Alamitos: IEEE Computer Society, 2010. Vol. 14, No. 6. P. 80–83.
- [9]. Qodirov A. Dasturlash asoslari va algoritmlash. Toshkent: O'zbekiston, 2020. 304 b.