# JAVASCRIPT ASINXRON DASTURLASH: PROMISE, ASYNC/AWAIT VA CALLBACK FUNKSIYALARINING TAHLILI

***To‘xtayeva Robiya Ravshanbek qizi***
*Andijon davlat universiteti, talaba*
*tukhtayeva_robiya@adu.uz*

**Anotatsiya:** Ushbu maqola JavaScript asinxron dasturlash texnikalariga bag‘ishlangan bo‘lib, unda Promise, async/await, va callback funksiyalarining ishlash prinsiplari, afzalliklari va kamchiliklari batafsil tahlil qilinadi. Maqolada dasturchilarga asinxron jarayonlarni boshqarishning turli usullari ko‘rsatilib, ushbu texnikalarni qo‘llashdagi samaradorlik va qulaylikni tushuntirish maqsad qilingan. Callback funksiyalarning zanjirli tuzilmalari va ular bilan bog‘liq muammolar, Promise ning kodni yanada strukturalashtirishdagi roli va async/await yordamida kodni yanada intuitivroq va o‘qilishi osonroq qilish usullari muhokama qilinadi.

**Kalit so‘zlar:** JavaScript, Asinxron dasturlash, Callback funksiyalar, Promise, async/await, Asinxron kodni boshqarish, JavaScript dasturlash texnikalari, Kod samaradorligi, Asinxron operatsiyalar, JavaScript o‘rganish

JavaScript dasturlash dunyosida asinxronlik muhim mavqega ega, chunki u bir vaqtning o‘zida bir nechta vazifalarni bajarish imkonini beradi. Asinxron dasturlash, ayniqsa, real vaqt rejimida ma’lumotlar olish, server bilan ishlash yoki murakkab hisoblashlarni bajarish uchun muhim ahamiyatga ega. Ushbu maqolada callback funksiyalar, Promise va async/await usullarining ishlash mexanizmlari, afzalliklari va kamchiliklari batafsil tahlil qilinadi.

**Callback Funksiyalar**: Callback funksiyasi boshqa funksiya tomonidan chaqiriladigan funksiyadir. JavaScript bir vaqtning o‘zida faqat bitta vazifani bajaradigan “Event Loop” mexanizmiga ega bo‘lgani sababli, callback funksiyalar kodni uzluksiz va asinxron bajarishga imkon beradi.

```
function fetchData(callback) {
    setTimeout(() => {
        const data = "Serverdan kelgan ma'lumot";
        callback(data);
    }, 2000);
}

fetchData((result) => {
    console.log(result);
});
```

Asinxron operatsiyalarni amalga oshirishda oddiy va samarali. Tez o‘rganish va ishlatish mumkin. Callback funksiyalarni zanjir sifatida ishlatish kodni murakkab va

o‘qilishi qiyin qiladi. Xatolarni boshqarish murakkabligi: Xatolarni kuzatish va ularni qayta ishlash ko‘p mehnat talab qiladi.

```
asyncOperation1((result1) => {
    asyncOperation2(result1, (result2) => {
        asyncOperation3(result2, (result3) => {
            console.log(result3);
        });
    });
});
```

Bu misol callback hell sifatida tanilgan murakkablikni ko‘rsatadi.

**Promise** - bu JavaScriptda asinxron operatsiyalarni boshqarish uchun yaratilgan obyekt bo‘lib, uning holati uch xil bo‘lishi mumkin:

1. Pending (kutish): Operatsiya hali yakunlanmagan.
2. Fulfilled (bajarilgan): Operatsiya muvaffaqiyatli yakunlandi.
3. Rejected (rad etilgan): Operatsiya muvaffaqiyatsizlikka uchradi.

Promise struktura kodni tartibli qilishga yordam beradi va callback hell muammosini bartaraf etadi.

```
const fetchData = new Promise((resolve, reject) => {
    setTimeout(() => {
        const success = true;
        if (success) {
            resolve("Ma'lumot muvaffaqiyatli olindi");
        } else {
            reject("Xato yuz berdi");
        }
    }, 2000);
});

fetchData
    .then((data) => console.log(data))
    .catch((error) => console.error(error));
```

Kod strukturasining yaxshilanishi: Chain (zanjir) usuli kodni tartibli qiladi. .catch() metodi orqali xatolarni osonlik bilan qayta ishlash mumkin.

Murakkab operatsiyalarni boshqarish uchun Promise ham unchalik qulay emas. Ketma-ket zanjirlangan operatsiyalar hali ham ko‘plab kod yozishni talab qiladi.

**Async/Await ES2017 (ES8)** versiyasida joriy etilgan va Promise asosida ishlaydi. U kodni sinxron operatsiyaga o‘xshash qilib yozishga imkon beradi, bu esa o‘qish va tushunishni ancha osonlashtiradi.

```
async function fetchData() {
    try {
        const response = await new Promise((resolve, reject) =>
            setTimeout(() => resolve("Ma'lumot muvaffaqiyatli o
        });
        console.log(response);
    } catch (error) {
        console.error(error);
    }
}

fetchData();
```

**Afzalliklari**: Kod o‘qilishi va tushunilishi oson: Async/Await sintaksisi callback va Promise’ga qaraganda intuitivroq. Xatolarni boshqarish osonligi: try-catch bloklari orqali xatolarni boshqarish imkoniyati.

**Kamchiliklari**: Faqat Promise bilan ishlaydi, bu esa ba’zi eski texnologiyalar bilan muvofiq bo‘lmasligi mumkin. await ning noto‘g‘ri ishlatilishi kodning samaradorligini pasaytirishi mumkin.

**JavaScript**-ning asinxron mexanizmi "Event Loop" deb nomlanadi. Bu mexanizm sinxron va asinxron vazifalarni boshqaradi, navbatni kuzatib boradi va fon vazifalarining bajarilishini ta’minlaydi.

```
console.log("1");
setTimeout(() => console.log("2"), 0);
console.log("3");
```

Yuqoridagi kodda, natija "1", "3", va keyin "2" bo‘ladi. Bu "Event Loop" sababli sodir bo‘ladi.

**Mikrova Zanjir (Microtask Queue):** Promise dan foydalanganda mikrova zanjiri ishlatiladi. Bu zanjir oddiy callback vazifalaridan yuqori ustuvorlikka ega.

```
console.log("1");
Promise.resolve().then(() => console.log("2"));
console.log("3");
```

Bu yerda natija "1", "3", va keyin "2" bo‘ladi, chunki Promise mikrova zanjirida ishlaydi.

Parallel va Sekvential Operatsiyalar: JavaScript kodini samarali ishlatish uchun parallel va ketma-ket operatsiyalarni tushunish muhimdir. Parallel: Bir vaqtning o‘zida bir nechta operatsiyalarni bajaradi. Sekvential: Operatsiyalarni ketma-ket bajaradi.

JavaScriptda asinxron dasturlash texnikalarining rivojlanishi dasturchilarga

yanada samarali va intuitiv kod yozish imkonini berdi. Callback funksiyalar oddiy operatsiyalar uchun mos bo‘lsa-da, murakkablik oshgani sayin ular samaradorligini yo‘qotadi. Promise bu muammolarni qisman hal qilsa, Async/Await kodni yanada qulayroq va tushunarliroq qiladi. Har bir usulning afzalliklari va kamchiliklarini tushunib, muayyan vaziyatga mosini tanlash dasturchining muvaffaqiyatiga hissa qo‘shadi.

**Foydalanilgan adabiyotlar:**

1. React Native Documentation – React Native rasmiy hujjatlari. Onlayn manba
2. Expo Documentation – Expo rasmiy hujjatlari, React Native uchun rivojlantirish muhitini taqdim etuvchi platforma. Onlayn manba
3. Brown, A. (2021). React Native Cookbook: Bringing the Web to Native Platforms. Packt Publishing.
4. Chinnathambi, A. (2020). React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications. Apress.
5. Kumar, N. (2019). Mastering React Native. Packt Publishing.