EDUMO ACADEMY

JAVA SCRIPT

O‘zbek o‘quvchilari uchun sodda qo‘llanma

Kubayev Javlonbek

Muqaddima

Bu kitob JavaScript dasturlash tilini o'rganish uchun mo'ljallangan mo'jazgina qo'llanmadir. JavaScript bugungi kunda veb-sahifalarni dinamik va interaktiv qilish, ma'lumotlar bilan ishlash, server bilan bog'lanish, interfeysni qurish, animatsiyalar yaratish va ko'plar turli xil muammolarni yechish uchun eng kuchli dasturlash tili hisoblanadi. Umid qilamizki, bu kitobni o'qib, siz JavaScript bo'yicha bilimlarni tezda o'zlashtrib, turli xil muammolarni hal qiladigan yangi loyihalarni hayotga tatbiq etasiz.

Kitobning Maqsadi

Bu kitob orqali siz quyidagi asosiy mavzularni o'rganasiz:

- JavaScriptning asosiy sintaksisi va buyruqlari
- Ob'ektlar, massivlar, funksiyalar va qo'shimcha JavaScript tushunchalari
- Veb-sahifalarda JavaScript kodini qo'llashning eng yaxshi usullari
- Asinxron dasturlash va AJAX
- Modullar dasturlash va ko'plab muammolarni yechish bo'yicha amaliyotlar

Kimlar uchun bu Kitob?

Bu kitob JavaScriptga boshlang'ich darajadagi dasturchilar uchun mo'ljallangan, lekin o'zlashtirilgan muammolar va ustuvor mavzular orqali ortiqcha bilimlarga erishishingiz mumkin. Agar siz veb-sahifalarni yaratishda va dinamik qilishda qiziqsangiz yoki dasturlash sohasida yangiliklarni o'rganishni istasangiz, bu kitob siz uchun.

Kitobni Ishlatish

Bu kitobni amaliy va nazariy darslik sifatida ishlatishingiz mumkin. Har bir bo'limda keyingi mavzularni o'rganishda sizga yordam berish uchun misollar va mashqlar keltirilgan.

Shuningdek, kitob davomida yaratilgan misollar, mashqlar va har xil qo'llanmalar sizni JavaScriptning bo'yicha bilimingizni rivojlantirishda yordam beradi.

Xulosa

Bu kitob JavaScript dasturlash tilini o'rganishga yo'l ochadi va sizga dunyoning yangi imkoniyatlarini ochadi. Umid qilamizki, bu kitob sizning dasturchi sifatidagi kasbingizni boshlashga va rivojlanishingizda muhim qadam bo'ladi.

Agar savollaringiz yoki takliflaringiz bo'lsa, bemalol biz bilan bog'laning: @javlon_developer (Telegram). Siz bilan birgalikda JavaScript dunyosiga sayohatimizni boshlaymiz!

Hurmat va ehtirom ila,

Kubayev Javlon

1. JavaScriptga Kirish

JavaScript Nima?

JavaScript — bu veb-sahifalarni interaktiv qilish uchun ishlatiladigan dasturlash tili. HTML va CSS bilan birgalikda, u veb-saytlarning muhim komponentlaridan biri hisoblanadi. HTML veb-sahifalarning tuzilishini, CSS esa ularning ko’rinishini belgilasa, JavaScript veb-sahifalarni dinamik qiladi, foydalanuvchilar bilan o’zaro aloqada bo’lish imkonini beradi.

JavaScript Tarixi va Rivojlanishi

JavaScript 1995-yilda Brendan Eich tomonidan Netscape kompaniyasida ishlab chiqilgan. Dastlab “Mocha” nomi bilan yaratilgan, keyinchalik “LiveScript” deb nomlangan, va nihoyat, “JavaScript” nomini olgan. JavaScriptning standarti ECMAScript deb ataladi va uning birinchi versiyasi 1997-yilda chiqarilgan. Hozirgi kunda ECMAScriptning turli versiyalari mavjud bo’lib, har bir yangi versiya JavaScript imkoniyatlarini kengaytiradi.

JavaScriptning Veb dasturlashdagi roli

JavaScript veb-ishlab chiqishning muhim qismi hisoblanadi. HTML veb-sahifalarning tuzilishini, CSS ularning dizaynini, JavaScript esa ularning interaktivligini ta’minlaydi. Masalan, foydalanuvchi tugmani bosganda yoki shaklni to’ldirganda, JavaScript bu harakatlarni kuzatishi va tegishli javoblarni berishi mumkin.

Ish Muhitini Sozlash

JavaScriptni yozish va ishlatish uchun oddiy matn muharriri va veb-brauzer kifoya qiladi. Quyida ish muhitini sozlash bo’yicha qo’llanma keltirilgan:

1. **Matn Muharriri**: VS Code, Sublime Text yoki Atom kabi matn muharririni yuklab oling va o’rnating. Masalan, VS Code-ni yuklab olish uchun bu yerni bosing.

2. **Brauzer**: Zamonaviy veb-brauzerlarni (Google Chrome, Firefox) o’rnating. Ushbu brauzerlarning barchasi JavaScript-ni to’g’ri ishlatadi.

3. **Birlamchi JavaScript Fayli**: Yangi HTML fayl yarating va unga JavaScript qo’shing. Quyida misol keltirilgan:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
```

```
    <title>Hello World</title>
</head>
<body>
    <h1>JavaScriptga Xush Kelibsiz!</h1>
    <script>
        console.log("Salom, Dunyo!");
    </script>
</body>
</html>
```

Bu kodni saqlang va brauzerda oching. Konsolda “Salom, Dunyo!” xabarini ko’rasiz.

Birinchi JavaScript Kodingiz

JavaScriptni boshlash uchun eng oddiy kod — “Hello, World!” xabarini chiqarish. Quyidagi misolni ko’rib chiqing:

```
console.log("Salom, Dunyo!");
```

Bu kodni brauzerning konsolida ishlatishingiz mumkin. Konsolni ochish uchun brauzerda F12 tugmasini bosing yoki o’ng tugmani bosib “Inspect” yoki “Qidiruv” tanlang, so’ngra “Console” bo’limiga o’ting.

Xulosa

Ushbu bo’limda siz JavaScriptning asosiy tushunchalari va tarixini o’rgandingiz. Endi sizda ish muhitini sozlash va birinchi JavaScript kodini yozish bo’yicha boshlang’ich bilimlar mavjud. Keyingi bo’limda biz JavaScriptning asosiy tushunchalari bilan tanishamiz.

O’zgaruvchilar (Variables)

> Bu bo’lim o’quvchilarga JavaScriptda o’zgaruvchilarni qanday yaratish, ularga qiymat berish va turli xil kalit so’zlardan qanday foydalanish haqida tushuncha beradi.

O’zgaruvchi nima?

JavaScriptda o’zgaruvchi - bu ma’lumotlarni saqlash uchun ishlatiladigan konteyner. O’zgaruvchi yaratish va unga qiymat berish uchun var, let yoki const kalit so’zlari ishlatiladi.

O’zgaruvchini yaratish

O’zgaruvchini yaratish uchun **var**, **let** yoki **const** kalit so’zlaridan foydalaniladi, so’ngra o’zgaruvchi nomi va qiymat tayinlanadi. Misollar:

```
var ism = "Ali";
let yosh = 25;
const tugilganYil = 1999;
```

Bu misolda:

`var` kalit so’zi bilan ism o’zgaruvchisi yaratilgan va unga “Ali” qiymati berilgan. `let` kalit so’zi bilan yosh o’zgaruvchisi yaratilgan va unga 25 qiymati berilgan. `const` kalit so’zi bilan tugilganYil o’zgaruvchisi yaratilgan va unga 1999 qiymati berilgan.

`var`, `let` va `const` o’rtasidagi farqlar

- **var**: Global va funksiya doirasida amal qiladi. U qayta belgilanishi va qayta o’zlashtirilishi mumkin.
- **let**: Blok doirasida amal qiladi. U qayta belgilanishi mumkin, lekin qayta e’lon qilinmaydi.
- **const**: Blok doirasida amal qiladi. U qayta belgilanishi va qayta o’zlashtirilishi mumkin emas. Bu o’zgaruvchining qiymati doimiy bo’lib qoladi.

Misollar:

```
var a = 10;
a = 20; // Bu ruxsat etiladi
```

```
let b = 30;
b = 40; // Bu ham ruxsat etiladi

const c = 50;
c = 60; // Bu xato beradi: TypeError: Assignment to constant
        variable.
```

O’zgaruvchi nomlash qoidalari

JavaScriptda o’zgaruvchilarni nomlashda quyidagi qoidalarga amal qilish kerak:

- O’zgaruvchi nomlari harf, _, $ bilan boshlanishi kerak.
- O’zgaruvchi nomlari raqam bilan boshlanmaydi.
- O’zgaruvchi nomlari har qanday uzunlikda bo’lishi mumkin va katta-kichik harflar farqlanadi.

To’g’ri misollar:

```
let ism = "Ali";
let _yosh = 25;
let $pul = 1000;
```

Noto’g’ri misollar:

```
let 1raqam = 10; // Xato: raqam bilan boshlanmaydi
let ism&familiya = "Ali Valiyev"; // Xato: noto'g'ri belgi
        ishlatilgan
```

O’zgaruvchilarga qiymat berish

O’zgaruvchilarga qiymat berish uchun tenglik (=) operatori ishlatiladi. O’zgaruvchining qiymatini yangilash mumkin (agar u const bo’lmasa).

Misollar:

```
let yosh = 25;
yosh = 26; // Endi yosh 26 ga teng

const tugilganYil = 1999;
// tugilganYil = 2000; // Xato: const o'zgaruvchining qiymatini
        o'zgartirib bo'lmaydi
```

O’zgaruvchilarning turlari

JavaScriptda o’zgaruvchilar turli xil ma’lumot turlariga ega bo’lishi mumkin:

1. `String`: Matnli qiymatlar (`"Ali"`, `"Salom"`)
2. `Number`: Raqamli qiymatlar (`25`, `3.14`)
3. `Boolean`: Mantiqiy qiymatlar (`true`, `false`)
4. `Array`: Massivlar (`[1, 2, 3]`, `["olma", "banan"]`)
5. `Object`: Ob’ektlar (`{ism: "Ali", yosh: 25}`)

Misollar:

```
let matn = "Salom dunyo"; // String
let raqam = 42; // Number
let haqiqat = true; // Boolean
let massiv = [1, 2, 3, 4, 5]; // Array
let obekt = {ism: "Ali", yosh: 25}; // Object
```

Izohlar (Comments)

Izohlar Nima?

Izohlar (comments) — bu dasturiy kod ichida yozilgan va dastur bajarilishi davomida e’tiborga olinmaydigan matn bo’laklari. Izohlar dasturchilarga kodni tushuntirishga, o’qilishini yaxshilashga va muhim eslatmalarni qoldirishga yordam beradi.

JavaScriptda izohlar ikki turga bo’linadi: 1. Bir qatorli izohlar 2. Ko’p qatorli izohlar

Bir Qatorli Izohlar

Bir qatorli izohlar `//` bilan boshlanadi. Bu belgidan keyingi barcha matn qator oxirigacha izoh sifatida qaraladi va bajarilmaydi.

```
// Bu bir qatorli izoh
let yosh = 25; // Bu ham bir qatorli izoh
```

Ko’p Qatorli Izohlar

Ko’p qatorli izohlar `/*` bilan boshlanadi va `*/` bilan tugaydi. Bu turdagi izohlar bir necha qatordan iborat bo’lishi mumkin.

```
/*
  Bu ko'p qatorli izoh.
  Bu izoh bir necha qatordan iborat bo'lishi mumkin.
*/
let ism = "Ali";
```

Izohlarni Foydalanish

Izohlar kodni tushunarli va o’qish uchun qulay qiladi. Quyida izohlarni qanday foydalanish bo’yicha ba’zi misollar keltirilgan:

Kodni Tushuntirish

Izohlar yordamida kodning qanday ishlashini tushuntirishingiz mumkin:

```
let radius = 5; // Aylana radiusi
let area = Math.PI * radius * radius; // Aylanani yuzini
        hisoblash
console.log("Aylanani yuzi: " + area);
```

Kod Bo’limlarini Belgilash

Kodni bo’limlarga ajratish va har bir bo’limni izoh bilan belgilash mumkin:

```
// Ma'lumotlarni e'lon qilish
let a = 10;
let b = 20;

// Hisoblash
let sum = a + b;

// Natijani chiqarish
console.log("Yig'indi: " + sum);
```

Murakkab Algoritmlarni Tushuntirish

Murakkab algoritmlarni va mantiqiy qadamlarni izohlash orqali boshqa dasturchilar yoki o’zingiz uchun tushunarli qilishingiz mumkin:

```
// Foydalanuvchidan kiritilgan sonning faktorialini hisoblash
function faktorial(n) {
    // Agar n 0 yoki 1 ga teng bo'lsa, faktorial 1 ga teng
    if (n === 0 || n === 1) {
        return 1;
    }

    // Aks holda, faktorialni rekursiv hisoblash
    return n * faktorial(n - 1);
}

console.log(faktorial(5)); // Natija: 120
```

Yaxshi Amaliyotlar

Izohlarni yozishda quyidagi yaxshi amaliyotlarga rioya qilish tavsiya etiladi: - **Aniq va qisqa bo’lish**: Izohlar aniq va qisqa bo’lishi kerak. - **Kerakli joyda izoh qo’shish**: Har bir qatorga izoh qo’shish kerak emas, faqat murakkab yoki tushunarsiz bo’lishi mumkin bo’lgan joylarga izoh qo’shing. - **Kod o’zgarishlaridan keyin izohlarni yangilash**: Kod o’zgarganda izohlarni ham yangilab turish muhim.

Xulosa

Ushbu bo’limda biz JavaScriptdagi izohlar bilan tanishdik. Izohlar kodni tushunarli va o’qilishini osonlashtiradi, shuningdek, dasturchilarga kod haqida eslatmalar va tushuntirishlar yozishga yordam beradi. Keyingi bo’limda biz JavaScriptdagi funksiyalarni o’rganamiz.

Ma’lumot turlari (Data Types)

> Bu bo’lim o’quvchilarga JavaScriptda turli ma’lumot turlari bilan qanday ishlash, ularni aniqlash va o’zgartirish haqida tushuncha beradi.

JavaScriptda ma’lumot turlari

JavaScriptda ma’lumotlar ikkita asosiy turga bo’linadi: primitiv (asosiy) turlar va ob’ekt turlari. Primitiv turlarga string, number, boolean, null, undefined va symbol kiradi. Ob’ekt turlariga esa array va object kiradi.

Primitiv turlar (Primitive Types)

String

String - matnli ma’lumotlarni saqlash uchun ishlatiladi. Stringlar qo’shtirnoq (“”) yoki birtirnoq (’’) ichida yoziladi.

```
let matn = "Salom, dunyo!";
```

Number

Number - raqamli ma’lumotlarni saqlash uchun ishlatiladi. U butun sonlar va o’nlik sonlarni o’z ichiga oladi.

```
let butunSon = 42;
let onlikSon = 3.14;
```

Boolean

Boolean - mantiqiy qiymatlar saqlaydi: `true` yoki `false`.

```
let rost = true;
let yolgon = false;
```

Null

Null - qiymat yo’qligini ifodalaydi. Bu qo’lda tayinlangan bo’sh qiymatdir.

```
let qiymatYoqligi = null;
```

Undefined

Undefined - qiymat tayinlanmagan o’zgaruvchilar uchun avtomatik ravishda beriladi.

```
let tayinlanmagan;
console.log(tayinlanmagan); // undefined
```

Symbol

Symbol - o’ziga xos va o’zgarmas identifikator yaratish uchun ishlatiladi.

```
let symbol = Symbol("unique");
```

Ob’ekt turlari (Object Types)

Array (Massiv)

Array - ma’lumotlar to’plamini tartiblangan holda saqlash uchun ishlatiladi. Array elementlari qavslar ichida yoziladi va vergul bilan ajratiladi.

```
let mevalar = ["olma", "banan", "apelsin"];
```

Object (Ob’ekt)

Object - kalit-qiymat juftliklari ko’rinishida murakkab ma’lumotlarni saqlash uchun ishlatiladi. Ob’ektlar jingalak qavslar ichida yoziladi.

```
let odam = {
  ism: "Ali",
  yosh: 30,
  kasb: "dasturchi"
};
```

Ma’lumot turini aniqlash (Type Detection)

Ma’lumot turini aniqlash uchun `typeof` operatoridan foydalaniladi.

```
console.log(typeof matn); // "string"
console.log(typeof butunSon); // "number"
console.log(typeof rost); // "boolean"
console.log(typeof
        qiymatYoqligi); // "object" (bu JavaScriptning xatosi)
console.log(typeof tayinlanmagan); // "undefined"
console.log(typeof symbol); // "symbol"
console.log(typeof odam); // "object"
console.log(typeof mevalar); // "object"
```

Ma’lumot turlarini o’zgartirish (Type Conversion)

JavaScriptda ma’lumot turlarini bir-biriga o’zgartirish mumkin.

Stringga o’zgartirish

```
let raqam = 123;
let matnRaqam = String(raqam); // "123"

let boolean = true;
let matnBoolean = String(boolean); // "true"
```

Raqamga o’zgartirish

```
let matn = "456";
let raqamMatn = Number(matn); // 456

let boolean = false;
let raqamBoolean = Number(boolean); // 0
```

Boolean ga o’zgartirish

```
let matn = "hello";
let booleanMatn = Boolean(matn); // true

let raqam = 0;
let booleanRaqam = Boolean(raqam); // false
```

Murakkab turlar bilan ishlash (Working with Complex Types)

Massivlar (Arrays)

Massivlar bilan ishlash uchun ko’plab metodlar mavjud:

```
let mevalar = ["olma", "banan", "apelsin"];
console.log(mevalar.length); // 3
mevalar.push("gilos"); // massiv oxiriga element qo'shadi
console.log(mevalar); // ["olma", "banan", "apelsin", "gilos"]
mevalar.pop(); // massiv oxiridan elementni olib tashlaydi
console.log(mevalar); // ["olma", "banan", "apelsin"]
```

Ob’ektlar (Objects)

Ob’ektlar bilan ishlash uchun quyidagi metodlardan foydalanish mumkin:

```
let odam = {
  ism: "Ali",
  yosh: 30,
  kasb: "dasturchi"
};
```

```
console.log(odam.ism); // "Ali"
odam.yosh = 31; // ob'ektning qiymatini o'zgartirish
console.log(odam.yosh); // 31
odam.shahar = "Toshkent"; // ob'ektga yangi xususiyat qo'shish
console.log(odam); // { ism: "Ali", yosh: 31, kasb: "dasturchi",
        shahar: "Toshkent" }
```

Stringlar (Strings)

> Bu bo’lim o’quvchilarga JavaScriptda stringlar bilan qanday ishlash, ularning uzunligini aniqlash, birlashtirish, kesish va boshqa ko’plab amallarni bajarish haqida tushuncha beradi.

String nima?

String - bu matnli ma’lumotlarni saqlash uchun ishlatiladigan ma’lumot turi. Stringlar qo’shtirnoq (“”) yoki birtirnoq (’’) ichida yoziladi.

String yaratish

String yaratish uchun qo’shtirnoq yoki birtirnoq ishlatiladi.

```
let matn1 = "Salom, dunyo!";
let matn2 = 'Salom, JavaScript!';
```

String uzunligi (Length)

Stringning uzunligini aniqlash uchun `.length` xususiyatidan foydalaniladi.

```
let matn = "Salom, dunyo!";
console.log(matn.length); // 12
```

Stringlarni birlashtirish (Concatenation)

Stringlarni birlashtirish uchun `+` operatori yoki `.concat()` metodi ishlatiladi.

```
let salom = "Salom";
let dunyo = "dunyo";
let natija1 = salom + ", " + dunyo + "!"; // "Salom, dunyo!"
let natija2 = salom.concat(", ", dunyo, "!"); // "Salom, dunyo!"
```

String shablonlari (Template Literals)

ES6 dan boshlab, backtick (`) belgilari ichida shablon stringlari yaratish mumkin. Bu string ichida o’zgaruvchilarni kiritish imkonini beradi.

```
let ism = "Ali";
let salomlash = `Salom, ${ism}!`;
console.log(salomlash); // "Salom, Ali!"
```

String metodlari (Methods)

JavaScriptda stringlar bilan ishlash uchun ko’plab metodlar mavjud.

`toUpperCase()` va `toLowerCase()`

Stringni katta harflarga yoki kichik harflarga o’zgartiradi.

```
let matn = "Salom, dunyo!";
console.log(matn.toUpperCase()); // "SALOM, DUNYO!"
console.log(matn.toLowerCase()); // "salom, dunyo!"
```

`trim()`

Stringning boshlanishi va oxiridagi bo’sh joylarni olib tashlaydi.

```
let matn = "   Salom, dunyo!   ";
console.log(matn.trim()); // "Salom, dunyo!"
```

`slice()`

Stringning ma’lum bir qismini kesib olish uchun ishlatiladi. Birinchi argument boshlanish indeksini, ikkinchi argument tugash indeksini bildiradi (tugash indeksi kiritilmagan bo’lsa, string oxirigacha olinadi).

```
let matn = "Salom, dunyo!";
let qism = matn.slice(7, 12);
console.log(qism); // "dunyo"
```

`substring()` va `substr()`

Stringning ma’lum bir qismini kesib olish uchun ishlatiladi. `substring()` boshlanish va tugash indekslarini oladi. `substr()` esa boshlanish indeksini va uzunlikni oladi.

```
let matn = "Salom, dunyo!";
console.log(matn.substring(7, 12)); // "dunyo"
console.log(matn.substr(7, 5)); // "dunyo"
```

`replace()`

Stringning ma’lum bir qismini almashtirish uchun ishlatiladi.

```
let matn = "Salom, dunyo!";
let yangiMatn = matn.replace("dunyo", "JavaScript");
console.log(yangiMatn); // "Salom, JavaScript!"
```

`includes()`

String ichida ma’lum bir substring bor yoki yo’qligini tekshiradi.

```
let matn = "Salom, dunyo!";
let mavjudmi = matn.includes("dunyo");
console.log(mavjudmi); // true
```

indexOf() va lastIndexOf()

String ichida ma’lum bir substringning indeksini qaytaradi (topilgan birinchi yoki oxirgi indeksni).

```
let matn = "Salom, dunyo!";
console.log(matn.indexOf("dunyo")); // 7
console.log(matn.lastIndexOf("o")); // 10
```

charAt() va charCodeAt()

Ma’lum bir indeksdagi belgini yoki uning Unicode qiymatini qaytaradi.

```
let matn = "Salom";
console.log(matn.charAt(2)); // "l"
console.log(matn.charCodeAt(2)); // 108
```

split()

Stringni ma’lum bir delimiter bo’yicha bo’lib, massivga o’zgartiradi.

```
let matn = "Salom, dunyo!";
let qismlar = matn.split(", ");
console.log(qismlar); // ["Salom", "dunyo!"]
```

Raqamlar (Numbers)

> Bu bo’lim o’quvchilarga JavaScriptda raqamlar bilan qanday ishlash, ularni aniqlash, o’zgartirish va turli matematik amallarni bajarish haqida tushuncha beradi.

Raqam nima?

JavaScriptda raqamlar - bu butun sonlar va o’nlik sonlarni ifodalash uchun ishlatiladigan ma’lumot turi. JavaScriptda raqamlar uchun yagona ma’lumot turi mavjud, ya’ni `number`.

Raqam yaratish

Raqamlarni oddiygina o’zgaruvchiga tayinlash orqali yaratish mumkin.

```
let butunSon = 42;
let onlikSon = 3.14;
```

Arifmetik amallar

JavaScriptda raqamlar ustida turli arifmetik amallarni bajarish mumkin.

Qo’shish

```
let natija = 10 + 5; // 15
```

Ayirish

```
let natija = 10 - 5; // 5
```

Ko’paytirish

```
let natija = 10 * 5; // 50
```

Bo’lish

```
let natija = 10 / 5; // 2
```

Qoldiqni hisoblash

```
let natija = 10 % 3; // 1
```

Darajaga ko’tarish

```
let natija = 2 ** 3; // 8
```

Raqam metodlari va xususiyatlari

JavaScriptda raqamlar bilan ishlash uchun bir nechta foydali metodlar va xususiyatlar mavjud.

`toString()`

Raqamni stringga o’zgartiradi.

```
let raqam = 123;
let stringRaqam = raqam.toString(); // "123"
```

`toFixed()`

Raqamni berilgan o’nlik kasrlar soni bilan stringga o’zgartiradi.

```
let raqam = 3.14159;
let qisqaRaqam = raqam.toFixed(2); // "3.14"
```

`toPrecision()`

Raqamni belgilangan umumiy raqamlar soni bilan stringga o’zgartiradi.

```
let raqam = 3.14159;
let aniqlikRaqam = raqam.toPrecision(4); // "3.142"
```

`parseInt()` va `parseFloat()`

Stringni butun son yoki o’nlik songa o’zgartiradi.

```
let butunSon = parseInt("123"); // 123
let onlikSon = parseFloat("3.14"); // 3.14
```

`isNaN()`

Qiymat `NaN` (Not-a-Number) ekanligini tekshiradi.

```
let natija = isNaN("hello"); // true
let natija2 = isNaN(123); // false
```

`isFinite()`

Qiymat cheksiz yoki `NaN` emasligini tekshiradi.

```
let natija = isFinite(123); // true
let natija2 = isFinite(Infinity); // false
```

Math obyekti

JavaScriptda matematik amallarni bajarish uchun `Math` obyekti mavjud. Bu obyekt ko’plab foydali metodlar va xususiyatlarni o’z ichiga oladi.

Math.PI

Pi qiymatini qaytaradi.

```
let pi = Math.PI; // 3.141592653589793
```

Math.round()

Raqamni eng yaqin butun songa yaxlitlaydi.

```
let yaxlitRaqam = Math.round(4.6); // 5
```

Math.ceil()

Raqamni yuqoriga yaxlitlaydi.

```
let yuqoriRaqam = Math.ceil(4.2); // 5
```

Math.floor()

Raqamni pastga yaxlitlaydi.

```
let pastRaqam = Math.floor(4.8); // 4
```

Math.sqrt()

Raqamning kvadrat ildizini hisoblaydi.

```
let ildiz = Math.sqrt(16); // 4
```

Math.abs()

Raqamning absolyut qiymatini qaytaradi.

```
let absolyut = Math.abs(-42); // 42
```

Math.pow()

Raqamni belgilangan darajaga ko’taradi.

```
let daraja = Math.pow(2, 3); // 8
```

Math.max() va Math.min()

Berilgan argumentlar orasidan eng katta va eng kichik qiymatni qaytaradi.

```
let engKatta = Math.max(1, 2, 3, 4, 5); // 5
let engKichik = Math.min(1, 2, 3, 4, 5); // 1
```

Math.random()

0 va 1 orasida tasodifiy raqam qaytaradi.

```
let tasodifiy = Math.random();
```

Massivlar (Arrays)

> Bu bo’lim o’quvchilarga JavaScriptda massivlar bilan qanday ishlash, ularni yaratish, o’zgartirish, element qo’shish va olib tashlash, shuningdek, turli massiv metodlaridan foydalanish haqida tushuncha beradi.

Massiv nima?

Massiv - bu tartiblangan ma’lumotlar to’plami bo’lib, har bir element indeks bilan belgilanadi. Massivlar JavaScriptda bir nechta qiymatlarni bitta o’zgaruvchida saqlash imkonini beradi.

Massiv yaratish

Massivni yaratish uchun kvadrat qavslar (`[]`) ishlatiladi.

Bo’sh massiv yaratish

```
let boshMassiv = [];
```

Elementlar bilan massiv yaratish

```
let mevalar = ["olma", "banan", "apelsin"];
```

Massiv elementlariga murojaat qilish

Massiv elementlariga ularning indekslari orqali murojaat qilinadi. Indekslar 0 dan boshlanadi.

```
let birinchiMeva = mevalar[0]; // "olma"
let ikkinchiMeva = mevalar[1]; // "banan"
```

Massiv uzunligi (Length)

Massivning uzunligini aniqlash uchun `.length` xususiyatidan foydalaniladi.

```
let uzunlik = mevalar.length; // 3
```

Massivga element qo’shish

Massivga yangi element qo’shish uchun `push()` metodidan foydalaniladi.

```
mevalar.push("gilos");
console.log(mevalar); // ["olma", "banan", "apelsin", "gilos"]
```

Massivdan elementni olib tashlash

Massivdan oxirgi elementni olib tashlash uchun `pop()` metodidan foydalaniladi.

```
mevalar.pop();
console.log(mevalar); // ["olma", "banan", "apelsin"]
```

Massivning boshiga element qo’shish

Massivning boshiga yangi element qo’shish uchun `unshift()` metodidan foydalaniladi.

```
mevalar.unshift("anor");
console.log(mevalar); // ["anor", "olma", "banan", "apelsin"]
```

Massivning boshidan elementni olib tashlash

Massivning boshidan elementni olib tashlash uchun `shift()` metodidan foydalaniladi.

```
mevalar.shift();
console.log(mevalar); // ["olma", "banan", "apelsin"]
```

Massivdagi elementlarni o’zgartirish

Massivdagi elementlarni indeks orqali o’zgartirish mumkin.

```
mevalar[1] = "nok";
console.log(mevalar); // ["olma", "nok", "apelsin"]
```

Massiv metodlari (Methods)

JavaScriptda massivlar bilan ishlash uchun ko’plab metodlar mavjud.

`concat()`

Ikki yoki undan ko’p massivlarni birlashtiradi.

```
let sabzavotlar = ["sabzi", "kartoshka"];
let oziqOvqatlar = mevalar.concat(sabzavotlar);
console.log(oziqOvqatlar); // ["olma", "nok", "apelsin", "sabzi",
        "kartoshka"]
```

slice()

Massivdan qism olish uchun ishlatiladi. Birinchi argument boshlanish indeksini, ikkinchi argument tugash indeksini bildiradi (tugash indeksi kiritilmagan bo’lsa, massiv oxirigacha olinadi).

```
let qism = mevalar.slice(1, 3);
console.log(qism); // ["nok", "apelsin"]
```

splice()

Massivdan elementlarni olib tashlash va yangi elementlarni qo’shish imkonini beradi.

```
mevalar.splice(1, 1, "behi", "anjir"); // 1-indeksdan boshlab 1
        ta element olib tashlanadi va yangi elementlar qo'shiladi
console.log(mevalar); // ["olma", "behi", "anjir", "apelsin"]
```

indexOf()

Massivda elementning indeksini qaytaradi, agar element topilmasa -1 qaytaradi.

```
let indeks = mevalar.indexOf("anjir");
console.log(indeks); // 2
```

includes()

Massivda element bor-yo’qligini tekshiradi.

```
let mavjudmi = mevalar.includes("apelsin");
console.log(mavjudmi); // true
```

forEach()

Massiv elementlarini ko’rib chiqish uchun ishlatiladi.

```
mevalar.forEach(function(element, indeks) {
  console.log(indeks + ": " + element);
});
// 0: olma
// 1: behi
// 2: anjir
// 3: apelsin
```

map()

Massiv elementlarini o’zgartirib, yangi massiv yaratadi.

```
let yangiMassiv = mevalar.map(function(element) {
  return element.toUpperCase();
});
console.log(yangiMassiv); // ["OLMA", "BEHI", "ANJIR", "APELSIN"]
```

filter()

Massiv elementlarini filtrlaydi va yangi massiv yaratadi.

```
let uzunMevalar = mevalar.filter(function(element) {
  return element.length > 4;
});
console.log(uzunMevalar); // ["behi", "anjir", "apelsin"]
```

reduce()

Massiv elementlarini bitta qiymatga qisqartiradi.

```
let raqamlar = [1, 2, 3, 4, 5];
let yigindi = raqamlar.reduce(function(acc, current) {
  return acc + current;
}, 0);
console.log(yigindi); // 15
```

Obyektlar

Obyektlarni Tushunish

JavaScriptda obyektlar – bu kalit-qiymat juftliklaridan iborat ma’lumot tuzilmasi. Obyektlar yordamida ma’lumotlarni guruhlab, tartibga solish va ularga nomlar berish mumkin.

Obyekt Yaratish

Obyekt yaratishning bir necha usullari mavjud. Eng oddiy usuli — obyekt literalidan foydalanish:

```
let odam = {
    ism: "Ali",
    yosh: 25,
    kasb: "dasturchi"
};
```

Bu yerda `odam` degan obyekt yaratildi, unda `ism`, `yosh`, va `kasb` degan kalitlar va ularning qiymatlari mavjud.

Obyekt Xususiyatlariga Murojaat Qilish

Obyekt xususiyatlariga nuqta (`.`) yoki qavs (`[]`) orqali murojaat qilish mumkin:

```
// Nuqta notatsiyasi
console.log(odam.ism); // "Ali"
console.log(odam.yosh); // 25

// Qavs notatsiyasi
console.log(odam["kasb"]); // "dasturchi"
```

Obyektga Yangi Xususiyat Qo’shish

Obyektga yangi xususiyat qo’shish juda oson:

```
odam.manzil = "Toshkent";
console.log(odam.manzil); // "Toshkent"
```

Obyekt Xususiyatlarini O’zgartirish va O’chirish

Obyekt xususiyatlarini o’zgartirish va o’chirish ham oson:

```
// Xususiyatni o'zgartirish
odam.yosh = 26;
console.log(odam.yosh); // 26

// Xususiyatni o'chirish
delete odam.kasb;
console.log(odam.kasb); // undefined
```

Obyekt Metodlari

Obyekt metodlari – bu obyekt ichidagi funksiyalar. Metodlar obyektning xatti-harakatlarini belgilaydi.

Metod Yaratish

Obyekt metodini yaratish uchun obyekt ichida funksiya belgilash kifoya:

```
let mashina = {
    model: "Tesla",
    yil: 2021,
    ovoz: function() {
        console.log("Bip Bip!");
    }
};

// Metodni chaqirish
mashina.ovoz(); // "Bip Bip!"
```

this Kalit So’zi

`this` kalit so’zi metod ichida obyektga murojaat qilish uchun ishlatiladi:

```
let talaba = {
    ism: "Dilshod",
    yosh: 22,
    salomBer: function() {
        console.log("Salom, mening ismim " + this.ism);
    }
};

talaba.salomBer(); // "Salom, mening ismim Dilshod"
```

Obyektlarni Birlashtirish

JavaScriptda obyektlarni birlashtirish uchun `Object.assign` metodidan foydalanish mumkin:

```
let maosh = {
    miqdor: 500,
    valyuta: "USD"
};

let birlashtirilganObyekt = Object.assign({}, odam, maosh);
console.log(birlashtirilganObyekt);
// { ism: "Ali", yosh: 26, manzil: "Toshkent", miqdor: 500,
        valyuta: "USD" }
```

Xulosa

Ushbu bo’limda siz JavaScriptda obyektlarni yaratish, ularga murojaat qilish, xususiyatlarini o’zgartirish va o’chirish, metodlarni qo’shish va `this` kalit so’zidan foydalanishni o’rgandingiz. Obyektlar JavaScriptda juda muhim tuzilma hisoblanadi va ular orqali dasturiy ta’minotning ko’p qismlarini tuzish mumkin.

window va history Ob’ektlari

window Ob’ekti

window ob’ekti brauzerdagi barcha ob’ektlar va funksiyalar uchun global konteyner hisoblanadi. U brauzer oynasini ifodalaydi va undagi barcha narsalarga kirish imkonini beradi.

window xususiyatlari va metodlari

window.alert()

Foydalanuvchiga ogohlantiruvchi xabar ko’rsatadi.

```
window.alert("Salom, dunyo!");
```

window.prompt()

Foydalanuvchidan ma’lumot kiritishni so’raydi.

```
let ism = window.prompt("Ismingizni kiriting:");
console.log("Salom, " + ism + "!");
```

window.confirm()

Foydalanuvchidan tasdiqlashni so’raydi.

```
let tasdiq = window.confirm("Siz rostdan ham davom etishni
        xohlaysizmi?");
if (tasdiq) {
  console.log("Foydalanuvchi davom etishni tanladi.");
} else {
  console.log("Foydalanuvchi davom etishni rad etdi.");
}
```

window.location

Brauzerning joriy URL manzilini olish yoki o’rnatish uchun ishlatiladi.

```
console.log(window.location.href); // Joriy URL manzilini
        ko'rsatadi

// Boshqa sahifaga o'tish
window.location.href = "https://www.example.com";
```

`window.setTimeout()` va `window.setInterval()`

Ma’lum bir vaqt o’tgach funksiyani bajarish yoki funksiyani muntazam ravishda bajarish uchun ishlatiladi.

```
// 3 soniyadan keyin xabarni chiqarish
window.setTimeout(() => {
  console.log("3 soniya o'tdi.");
}, 3000);

// Har 2 soniyada xabarni chiqarish
let interval = window.setInterval(() => {
  console.log("2 soniya o'tdi.");
}, 2000);

// Intervalni to'xtatish
window.clearInterval(interval);
```

`window` ob’ekti misollari

Oyna hajmini olish

```
let kenglik = window.innerWidth;
let balandlik = window.innerHeight;

console.log("Oyna kengligi: " + kenglik);
console.log("Oyna balandligi: " + balandlik);
```

Sahifa yuklanganda xabar chiqarish

```
window.onload = () => {
  console.log("Sahifa yuklandi.");
};
```

`history` Ob’ekti

`history` ob’ekti brauzerning tarixini boshqarish uchun ishlatiladi. U foydalanuvchining brauzer tarixida harakat qilishiga imkon beradi.

`history` xususiyatlari va metodlari

`history.back()`

Brauzerni bir bosqich orqaga qaytaradi (Back tugmasi kabi).

```
history.back();
```

`history.forward()`

Brauzerni bir bosqich oldinga qaytaradi (Forward tugmasi kabi).

```
history.forward();
```

`history.go()`

Brauzerni tarixda berilgan bosqichlar soniga qarab oldinga yoki orqaga qaytaradi.

```
// Bir bosqich orqaga qaytish
history.go(-1);

// Ikki bosqich oldinga qaytish
history.go(2);
```

`history.length`

Tarixdagi yozuvlar sonini ko’rsatadi.

```
console.log("Tarix yozuvlari soni: " + history.length);
```

`history` ob’ekti misollari

Foydalanuvchini oldingi sahifaga qaytarish

```
function oldingiSahifa() {
  if (history.length > 1) {
    history.back();
  } else {
    console.log("Orqaga qaytish uchun hech qanday sahifa yo'q.");
  }
}
```

O’yin davomida foydalanuvchini sahifada ushlab turish

```
window.onbeforeunload = () => {
  return "Sahifani tark etmoqchimisiz? Hozirgi o'yiningiz
        saqlanmaydi.";
};
```

Mashqlar

1. **Mashq**: Foydalanuvchiga “Saytga xush kelibsiz!” xabarini ko’rsatish uchun `window.alert()` funksiyasidan foydalaning.

Yechim:

```
window.alert("Saytga xush kelibsiz!");
```

2. **Mashq**: Foydalanuvchidan ismini so’rang va konsolda “Salom, [ism]!” xabarini chiqaring.

   Yechim:

   ```
   let ism = window.prompt("Ismingizni kiriting:");
   console.log("Salom, " + ism + "!");
   ```

3. **Mashq**: 5 soniyadan keyin konsolda xabar chiqarish uchun `window.setTimeout()` funksiyasidan foydalaning.

   Yechim:

   ```
   window.setTimeout(() => {
     console.log("5 soniya o'tdi.");
   }, 5000);
   ```

4. **Mashq**: Brauzer tarixidagi yozuvlar sonini konsolda chiqarish uchun `history.length` dan foydalaning.

   Yechim:

   ```
   console.log("Tarix yozuvlari soni: " + history.length);
   ```

Xulosa

Bu dars `window` va `history` ob’ektlari haqida batafsil ma’lumot beradi.

Operatorlar (Operators)

Bu bo’lim o’quvchilarga JavaScriptda operatorlardan qanday foydalanish, ularning turlari va amallarini qanday bajarish haqida tushuncha beradi.

Operator nima?

JavaScriptda operatorlar o’zgaruvchilar va qiymatlar ustida amallar bajarish uchun ishlatiladi. Operatorlar arifmetik, mantiqiy, taqqoslash va boshqa turlarga bo’linadi.

Arifmetik operatorlar (Arithmetic Operators)

Arifmetik operatorlar raqamlar ustida matematik amallarni bajaradi.

Misollar:

- + (qo’shish): Ikkita raqamni qo’shadi.

```
let a = 5;
let b = 3;
let natija = a + b; // natija 8 ga teng bo'ladi
```

- - (ayirish): Ikkita raqamni ayiradi.

```
let natija = a - b; // natija 2 ga teng bo'ladi
```

- * (ko’paytirish): Ikkita raqamni ko’paytiradi.

```
let natija = a * b; // natija 15 ga teng bo'ladi
```

- / (bo’lish): Bir raqamni boshqa raqamga bo’ladi.

```
let natija = a / b; // natija 1.6667 ga teng bo'ladi
```

- % (qoldiq): Bo’lishdan qolgan qoldiqni qaytaradi.

```
let natija = a % b; // natija 2 ga teng bo'ladi
```

- ** (daraja): Bir raqamni ikkinchi raqam darajasiga ko’taradi.

```
let natija = a ** b; // natija 125 ga teng bo'ladi
```

Taqqoslash operatorlari (Comparison Operators)

Taqqoslash operatorlari ikki qiymatni taqqoslaydi va mantiqiy natija (true yoki false) qaytaradi.

Misollar:

- == (teng): Ikkita qiymat tengligini tekshiradi (tipni hisobga olmaydi).

```
let natija = (a == b); // natija false bo'ladi
```

- === (qattiq teng): Ikkita qiymat va ularning tipini tekshiradi.

```
let natija = (a === b); // natija false bo'ladi
```

- != (teng emas): Ikkita qiymat teng emasligini tekshiradi (tipni hisobga olmaydi).

```
let natija = (a != b); // natija true bo'ladi
```

- !== (qattiq teng emas): Ikkita qiymat va ularning tipi teng emasligini tekshiradi.

```
let natija = (a !== b); // natija true bo'ladi
```

- > (katta): Birinchi qiymat ikkinchi qiymatdan katta ekanligini tekshiradi.

```
let natija = (a > b); // natija true bo'ladi
```

- < (kichik): Birinchi qiymat ikkinchi qiymatdan kichik ekanligini tekshiradi.

```
let natija = (a < b); // natija false bo'ladi
```

- >= (katta yoki teng): Birinchi qiymat ikkinchi qiymatdan katta yoki teng ekanligini tekshiradi.

```
let natija = (a >= b); // natija true bo'ladi
```

- <= (kichik yoki teng): Birinchi qiymat ikkinchi qiymatdan kichik yoki teng ekanligini tekshiradi.

```
let natija = (a <= b); // natija false bo'ladi
```

Mantiqiy operatorlar (Logical Operators)

Mantiqiy operatorlar mantiqiy qiymatlar ustida amallar bajaradi.

Misollar:

- && (va): Ikkala ifoda ham true bo’lsa, true qaytaradi.

```
let natija = (a > 0 && b > 0); // natija true bo'ladi
```

- || (yoki): Ikkala ifodadan biri true bo’lsa, true qaytaradi.

```
let natija = (a > 0 || b < 0); // natija true bo'ladi
```

- ! (emas): Ifodaning mantiqiy qiymatini teskarisiga o’zgartiradi.

```
let natija = !(a > 0); // natija false bo'ladi
```

O’zlashtirish operatorlari (Assignment Operators)

O’zlashtirish operatorlari o’zgaruvchiga qiymat berish uchun ishlatiladi.

Misollar:

- = (oddiy o’zlashtirish): O’zgaruvchiga qiymat beradi.

```
let c = 10;
```

- += (qo’shish va o’zlashtirish): O’zgaruvchiga qiymat qo’shadi va natijani saqlaydi.

```
c += 5; // c endi 15 ga teng bo'ladi
```

- -= (ayirish va o’zlashtirish): O’zgaruvchidan qiymat ayiradi va natijani saqlaydi.

```
c -= 3; // c endi 12 ga teng bo'ladi
```

- *= (ko’paytirish va o’zlashtirish): O’zgaruvchini qiymatga ko’paytiradi va natijani saqlaydi.

```
c *= 2; // c endi 24 ga teng bo'ladi
```

- /= (bo’lish va o’zlashtirish): O’zgaruvchini qiymatga bo’ladi va natijani saqlaydi.

```
c /= 4; // c endi 6 ga teng bo'ladi
```

- %= (qoldiq va o’zlashtirish): O’zgaruvchini qiymatga bo’lgandan qolgan qoldiqni saqlaydi.

```
c %= 5; // c endi 1 ga teng bo'ladi
```

Boshqa operatorlar

- `++` (inkrement): O’zgaruvchi qiymatini 1 ga oshiradi.

```
let d = 5;
d++; // d endi 6 ga teng bo'ladi
```

- `--` (dekrement): O’zgaruvchi qiymatini 1 ga kamaytiradi.

```
d--; // d endi 5 ga teng bo'ladi
```

- `? :` (ternar operator): Shart ifodasini tekshiradi va shart rost bo’lsa birinchi qiymatni, yolg’on bo’lsa ikkinchi qiymatni qaytaradi.

```
let yosh = 18;
let ruxsat = (yosh >= 18) ? "Kirish ruxsat etilgan" :
        "Kirish ruxsat etilmagan";
console.log(ruxsat); // Konsolga "Kirish ruxsat etilgan"
        yozadi
```

Funksiyalar (Functions)

> Bu bo’lim o’quvchilarga JavaScriptda funksiyalarni qanday yaratish, chaqirish va ulardan qanday foydalanish haqida tushuncha beradi.

Funksiya nima?

Funksiya - bu qayta ishlatish mumkin bo’lgan kod blokidir, u ma’lum bir vazifani bajaradi. Funksiyalarni yaratish va ulardan foydalanish dasturiy ta’minotni tashkil etishda juda muhimdir.

Funksiyani yaratish

Funksiya yaratish uchun function kalit so’zidan foydalaniladi, so’ngra funksiya nomi va qavslar ichida parametrlar kiritiladi. Funksiya tanasi `{}` (jingalak qavslar) ichida yoziladi.

Misol:

```
function salomBer() {
  console.log("Salom, dunyo!");
}
```

Bu misolda `salomBer` nomli funksiya yaratildi va u chaqirilganda konsolga `"Salom, dunyo!"` deb yozadi.

Funksiyani chaqirish

Yaratilgan funksiyani chaqirish uchun **uning nomi va qavslar** yoziladi.
Misol:

```
salomBer(); // Konsolga "Salom, dunyo!" yozadi
```

Parametrlar va argumentlar

Funksiyalar parametrlarni qabul qilishi mumkin. **Parametrlar** - bu funksiya chaqirilganda unga uzatiladigan qiymatlar.

Misol:

```
function yigindi(a, b) {
  return a + b;
}
```

```
let natija = yigindi(5, 7); // natija 12 ga teng bo'ladi
console.log(natija); // Konsolga 12 yozadi
```

Bu misolda `yigindi` funksiyasi ikkita parametr - `a` va `b` qabul qiladi va ularning yig’indisini qaytaradi.

Funksiya ifodalari (Function Expressions)

Funksiya ifodasi - bu funksiyani o’zgaruvchiga tayinlashdir. Bunday funksiyalar nomlanmagan bo’lishi mumkin (anonim funksiyalar).

Misol:

```
let kvadrat = function(x) {
  return x * x;
};

console.log(kvadrat(4)); // Konsolga 16 yozadi
```

Arrow funksiyalar (Arrow Functions)

ES6 (ECMAScript 2015) dan boshlab, JavaScriptda yangi turdagi funksiyalar - `arrow funksiyalar` joriy etildi. Ular qisqaroq sintaksisga ega.

Misol:

```
let kub = (x) => {
  return x * x * x;
};

console.log(kub(3)); // Konsolga 27 yozadi
```

> Arrow funksiyalarni yanada qisqartirish mumkin, agar funksiya tanasi bitta ifodadan iborat bo’lsa: ### Misol:

```
let ikkiBaravar = x => x * 2;
console.log(ikkiBaravar(5)); // Konsolga 10 yozadi
```

Funksiya ichida funksiya (Nested Functions)

Funksiya ichida boshqa funksiyalar yaratish va chaqirish mumkin. ### Misol:

```
function tashqiFunksiya(x) {
  function ichkiFunksiya(y) {
    return x + y;
  }
```

```
  return ichkiFunksiya;
}

let natijaFunksiya = tashqiFunksiya(5);
console.log(natijaFunksiya(3)); // Konsolga 8 yozadi
```

Rekursiv funksiyalar (Recursive Functions)

Funksiya o’z-o’zini chaqirishi mumkin. Bunday funksiyalar rekursiv funksiyalar deb ataladi.

Misol:

```
function faktorial(n) {
  if (n === 0) {
    return 1;
  } else {
    return n * faktorial(n - 1);
  }
}

console.log(faktorial(5)); // Konsolga 120 yozadi
```

> Bu misolda faktorial funksiyasi rekursiv ravishda faktorialni hisoblaydi.

Funksiya doirasi (Function Scope)

Funksiyalar o’zgaruvchilarni o’z ichida yaratishi va ulardan foydalanishi mumkin. Funksiya ichida yaratilgan o’zgaruvchilar funksiyadan tashqarida mavjud bo’lmaydi.

Misol:

```
function test() {
  let lokal = "Bu lokal o'zgaruvchi";
  console.log(lokal); // Konsolga "Bu lokal o'zgaruvchi" yozadi
}

test();
// console.log(lokal); // Xato beradi: lokal is not defined
```

Boshqaruv Tuzilmalari

Shart Operatorlari

JavaScriptda shart operatorlari yordamida kodning qaysi qismi bajarilishini aniqlash mumkin. Eng ko’p qo’llaniladigan shart operatorlari `if`, `else if`, `else` va `switch` hisoblanadi.

`if` Operator

`if` operatori shartni tekshiradi va agar shart to’g’ri bo’lsa, kod blokini bajaradi.

```
let yosh = 20;

if (yosh > 18) {
    console.log("Siz kattasiz.");
}
```

`else if` va `else` Operatorlari

`else if` va `else` operatorlari `if` operatori bilan birga qo’llaniladi. `else if` qo’shimcha shartlarni tekshirish uchun ishlatiladi, `else` esa barcha shartlar noto’g’ri bo’lsa, bajariladi.

```
let baho = 85;

if (baho >= 90) {
    console.log("A'lo baho");
} else if (baho >= 75) {
    console.log("Yaxshi baho");
} else {
    console.log("Yana ko'proq harakat qilish kerak");
}
```

`switch` Operator

`switch` operatori bir nechta shartlarni tekshirish uchun ishlatiladi. U if-else if-else zanjiriga muqobil hisoblanadi.

```
let meva = 'olma';

switch (meva) {
    case 'olma':
```

```
        console.log("Bu olma");
        break;
    case 'banan':
        console.log("Bu banan");
        break;
    default:
        console.log("Bu meva noma'lum");
}
```

Sikllar

JavaScriptda sikllar bir xil kodni bir necha marta bajarish imkonini beradi. Eng ko’p qo’llaniladigan sikllar `for`, `while` va `do...while` hisoblanadi.

`for` Sikli

`for` sikli aniq miqdordagi takrorlanishlar uchun ishlatiladi.

```
for (let i = 0; i < 5; i++) {
    console.log("Raqam: " + i);
}
```

`while` Sikli

`while` sikli shart to’g’ri bo’lganida takrorlanadi.

```
let i = 0;

while (i < 5) {
    console.log("Raqam: " + i);
    i++;
}
```

`do...while` Sikli

`do...while` sikli kamida bir marta bajariladi va keyin shart tekshiriladi.

```
let i = 0;

do {
    console.log("Raqam: " + i);
    i++;
} while (i < 5);
```

break va continue Operatorlari

break va continue operatorlari sikllarda maxsus vazifalarni bajarish uchun ishlatiladi.

break Operator

break operatori siklni to’xtatadi va undan chiqadi.

```
for (let i = 0; i < 10; i++) {
    if (i === 5) {
        break;
    }
    console.log("Raqam: " + i);
}
```

continue Operator

continue operatori joriy iteratsiyani o’tkazib yuboradi va siklning keyingi iteratsiyasini boshlaydi.

```
for (let i = 0; i < 10; i++) {
    if (i === 5) {
        continue;
    }
    console.log("Raqam: " + i);
}
```

Xulosa

Ushbu bo’limda biz JavaScriptning boshqaruv tuzilmalari bilan tanishdik. Shart operatorlari va sikllar yordamida kodning qaysi qismi bajarilishini nazorat qilishimiz mumkin. Bu tushunchalar dasturlashning asosiy qismi hisoblanadi va sizning JavaScript dasturlaringizda keng qo’llaniladi. Keyingi bo’limda biz funksiyalarni ko’rib chiqamiz.

Ob’ektlar va Massivlarda Iteratsiya

Massivlarda Iteratsiya

JavaScriptda massivlar (arrays) o’z ichiga bir nechta elementlarni oladi va ularda iteratsiya qilishning bir necha usullari mavjud.

`for` Tsikli

Eng oddiy usul — `for` tsiklidan foydalanish.

```
let fruits = ["olma", "banan", "uzum"];

for (let i = 0; i < fruits.length; i++) {
    console.log(fruits[i]);
}
```

`for...of` Tsikli

ES6da joriy qilingan `for...of` tsikli, massiv elementlarida iteratsiya qilish uchun qulay usul hisoblanadi.

```
for (let fruit of fruits) {
    console.log(fruit);
}
```

`forEach` Metodi

`forEach` metodi massivdagi har bir element uchun berilgan funksiyani chaqiradi.

```
fruits.forEach(function(fruit, index) {
    console.log(index + ": " + fruit);
});
```

`map` Metodi

`map` metodi massivning har bir elementi ustida berilgan funksiyani bajaradi va yangi massivni qaytaradi.

```
let upperFruits = fruits.map(function(fruit) {
    return fruit.toUpperCase();
});
```

```
console.log(upperFruits);
```

Ob’ektlarda Iteratsiya

Ob’ektlar (objects) kalit-qiymat juftliklarini o’z ichiga oladi. Ob’ektlarda iteratsiya qilish uchun bir nechta usullar mavjud.

`for...in` Tsikli

`for...in` tsikli ob’ekt kalitlarida iteratsiya qilish uchun ishlatiladi.

```
let person = {
    name: "Ali",
    age: 30,
    city: "Toshkent"
};

for (let key in person) {
    console.log(key + ": " + person[key]);
}
```

`Object.keys` Metodi

`Object.keys` metodi ob’ektning barcha kalitlarini massiv ko’rinishida qaytaradi, bu esa `for...of` yoki `forEach` bilan iteratsiya qilish imkonini beradi.

```
let keys = Object.keys(person);

keys.forEach(function(key) {
    console.log(key + ": " + person[key]);
});
```

`Object.values` Metodi

`Object.values` metodi ob’ektning barcha qiymatlarini massiv ko’rinishida qaytaradi.

```
let values = Object.values(person);

values.forEach(function(value) {
    console.log(value);
});
```

`Object.entries` Metodi

`Object.entries` metodi ob’ektning kalit-qiymat juftliklarini massiv ko’rinishida qaytaradi.

```
let entries = Object.entries(person);

entries.forEach(function([key, value]) {
    console.log(key + ": " + value);
});
```

Aralash Misollar

Quyida massivlar va ob’ektlarda iteratsiya qilish bo’yicha aralash misollar keltirilgan.

Massiv ichidagi Ob’ektlar

```
let people = [
    {name: "Ali", age: 30},
    {name: "Vali", age: 25},
    {name: "Guli", age: 27}
];

people.forEach(function(person) {
    console.log(person.name + " " + person.age);
});
```

Ob’ekt ichidagi Massivlar

```
let department = {
    name: "Marketing",
    employees: ["Ali", "Vali", "Guli"]
};

console.log(department.name + " bo'limidagi xodimlar:");

department.employees.forEach(function(employee) {
    console.log(employee);
});
```

Xulosa

Ushbu bo’limda biz JavaScriptda massivlar va ob’ektlarda iteratsiya qilish usullarini ko’rib chiqdik. `for`, `for...of`, `forEach`, `map` kabi tsikl va metodlar yordamida massivlarda iteratsiya qilish, `for...in`, `Object.keys`,

`Object.values`, `Object.entries` metodlari yordamida ob’ektlarda iteratsiya qilishni o’rgandik. Keyingi bo’limda biz JavaScriptda asinxron dasturlash va AJAX bilan ishlashni o’rganamiz.

Date Obyekti (Date Object)

> Bu bo’lim o’quvchilarga JavaScriptda Date obyekti bilan qanday ishlash, joriy sana va vaqtni olish, sanalarni o’rnatish va formatlash, sanalarni taqqoslash va vaqt farqlarini hisoblash haqida tushuncha beradi.

Date obyekti nima?

Date obyekti JavaScriptda sana va vaqtni ifodalash uchun ishlatiladi. Bu obyekt yordamida joriy vaqtni olish, sanalarni hisoblash, vaqt farqlarini aniqlash va boshqa ko’plab amallarni bajarish mumkin.

Date obyektini yaratish

Date obyektini yaratish uchun `new Date()` konstruktoridan foydalaniladi.

Joriy sana va vaqt

```
let hozirgiVaqt = new Date();
console.log(hozirgiVaqt);
```

Belgilangan sana va vaqt

```
let belgilanganSana = new Date('2024-05-29T14:00:00');
console.log(belgilanganSana);
```

Yil, oy, kun va boshqa komponentlar bilan sana yaratish

```
let maxsusSana = new Date(2024, 4, 29, 14, 0, 0); // Yil, oy
        (0-11), kun, soat, daqiqa, soniya
console.log(maxsusSana);
```

Date metodlari

Date obyekti ko’plab metodlarni o’z ichiga oladi.

Sana va vaqt olish

```
let hozirgiSana = new Date();

let yil = hozirgiSana.getFullYear(); // Yilni olish
let oy = hozirgiSana.getMonth(); // Oyini olish (0-11)
let kun = hozirgiSana.getDate(); // Kunini olish (1-31)
```

```
let haftaKuni = hozirgiSana.getDay(); // Haftaning kunini olish
        (0-6)
let soat = hozirgiSana.getHours(); // Soatini olish (0-23)
let daqiqa = hozirgiSana.getMinutes(); // Daqiqani olish (0-59)
let soniya = hozirgiSana.getSeconds(); // Soniyani olish (0-59)
let millisoniya =
        hozirgiSana.getMilliseconds(); // Millisoniyani olish
        (0-999)
```

UTC vaqtini olish

```
let yilUTC = hozirgiSana.getUTCFullYear(); // Yilni UTC bo'yicha
        olish
let oyUTC = hozirgiSana.getUTCMonth(); // Oyini UTC bo'yicha
        olish
let kunUTC = hozirgiSana.getUTCDate(); // Kunini UTC bo'yicha
        olish
let haftaKuniUTC = hozirgiSana.getUTCDay(); // Haftaning kunini
        UTC bo'yicha olish
let soatUTC = hozirgiSana.getUTCHours(); // Soatini UTC bo'yicha
        olish
let daqiqaUTC = hozirgiSana.getUTCMinutes(); // Daqiqani UTC
        bo'yicha olish
let soniyaUTC = hozirgiSana.getUTCSeconds(); // Soniyani UTC
        bo'yicha olish
let millisoniyaUTC = hozirgiSana.getUTCMilliseconds(); //
        Millisoniyani UTC bo'yicha olish
```

Sana va vaqtni o’rnatish

```
let sana = new Date();

sana.setFullYear(2025); // Yilni o'rnatish
sana.setMonth(11); // Oyini o'rnatish (0-11)
sana.setDate(25); // Kunini o'rnatish (1-31)
sana.setHours(10); // Soatini o'rnatish (0-23)
sana.setMinutes(30); // Daqiqani o'rnatish (0-59)
sana.setSeconds(45); // Soniyani o'rnatish (0-59)
sana.setMilliseconds(500); // Millisoniyani o'rnatish (0-999)
```

Sana va vaqtni string sifatida olish

```
let sanaString = hozirgiSana.toString(); // String formatida
        olish
let sanaUTCString = hozirgiSana.toUTCString(); // UTC formatida
        string olish
```

```
let sanaISOString = hozirgiSana.toISOString(); // ISO formatida
        string olish
let sanaLocaleString = hozirgiSana.toLocaleString(); // Lokal
        formatda string olish
```

Sanalarni taqqoslash

Date obyektlari sanalarni taqqoslash uchun ishlatilishi mumkin.

Sanalarni solishtirish

```
let sana1 = new Date('2024-05-29');
let sana2 = new Date('2023-05-29');

let katta = sana1 > sana2; // true
let teng = sana1.getTime() === sana2.getTime(); // false
```

Vaqt farqini hisoblash

Sanalar orasidagi vaqt farqini hisoblash uchun `getTime()` metodidan foydalaniladi. Bu metod millisoniyalarda vaqtni qaytaradi.

```
let sana1 = new Date('2024-05-29');
let sana2 = new Date('2023-05-29');

let farqMillisoniya = sana1.getTime() - sana2.getTime(); //
        Millisoniyalarda farq
let farqKun = farqMillisoniya / (1000 * 60 * 60 * 24); //
        Kunlarda farq
console.log(farqKun); // 365
```

Date obyekti bilan ishlashning foydali funksiyalari

Joriy sanani formatlash

```
function formatlashSana(sana) {
  let yil = sana.getFullYear();
  let oy = (sana.getMonth() + 1).toString().padStart(2, '0'); //
        Oyini 2 raqamli qilish
  let kun = sana.getDate().toString().padStart(2,
        '0'); // Kunini 2 raqamli qilish
  return `${yil}-${oy}-${kun}`;
}

let hozirgiSana = new Date();
console.log(formatlashSana(hozirgiSana)); // "2024-05-29"
```

Hodisalar (Events)

Hodisalar Nima?

Hodisalar (events) — bu foydalanuvchi yoki brauzer tomonidan sodir bo’ladigan harakatlar. JavaScript yordamida ushbu hodisalarga javob berishimiz va veb-sahifani interaktiv qilishimiz mumkin. Hodisalarga misollar kiritma (input) maydonini to’ldirish, tugmani bosish, sahifani yuklash va hokazo.

Hodisa Turlari

JavaScriptda ko’plab hodisa turlari mavjud. Eng ko’p qo’llaniladigan hodisalardan ba’zilari:

- **`click`**: Tugma yoki boshqa element bosilganda.
- **`dblclick`**: Element ikki marta bosilganda.
- **`mouseover`**: Kursor element ustiga kelganda.
- **`mouseout`**: Kursor elementdan chiqib ketganda.
- **`keydown`**: Klaviatura tugmasi bosilganda.
- **`keyup`**: Klaviatura tugmasi qo’yib yuborilganda.
- **`load`**: Sahifa yoki tasvir yuklanganda.
- **`submit`**: Shakl yuborilganda.
- **`input`**: Kiritma maydoni qiymati o’zgartirilganda.

Hodisalarni Qo’shish

JavaScriptda hodisalarni qo’shish uchun bir nechta usul mavjud. Eng ko’p ishlatiladigan usul bu `addEventListener` metodidan foydalanishdir.

`addEventListener` Metodi

`addEventListener` metodi yordamida elementga hodisani qo’shish.

```
let button = document.getElementById("myButton");

button.addEventListener("click", function() {
    alert("Tugma bosildi!");
});
```

Inline Hodisa Qo’shish

HTML teglari ichida hodisa qo’shish usuli. Biroq, bu usul kamroq tavsiya etiladi, chunki bu usul kodni tartibsiz va qiyin boshqariladigan qiladi.

```
<button id="myButton" onclick="alert('Tugma bosildi!')">Bos</
        button>
```

Hodisa Ob’ekti

Hodisa sodir bo’lganda, brauzer hodisa ob’ektini yaratadi, unda hodisa haqida ma’lumotlar saqlanadi. Hodisa ob’ektiga hodisa qo’shilganda funksiya orqali kirish mumkin.

```
button.addEventListener("click", function(event) {
    console.log("Hodisa turi: " + event.type);
    console.log("Bosilgan element: " + event.target);
});
```

Ko’p Martalik Hodisa Qo’shish

Bir elementga bir nechta hodisalarni qo’shish mumkin.

```
button.addEventListener("click", function() {
    console.log("Birinchi hodisa");
});

button.addEventListener("click", function() {
    console.log("Ikkinchi hodisa");
});
```

Hodisani Bekor Qilish

Ba’zi hodisalarni bekor qilish uchun `preventDefault` metodidan foydalanish mumkin. Masalan, shaklni yuborish hodisasini bekor qilish.

```
let form = document.getElementById("myForm");

form.addEventListener("submit", function(event) {
    event.preventDefault(); // Shakl yuborilishini bekor qiladi
    console.log("Shakl yuborilishi bekor qilindi");
});
```

Hodisa Tarqalishini To’xtatish

Hodisa tarqalishini to’xtatish uchun `stopPropagation` metodidan foydalanish mumkin.

```
let outerDiv = document.getElementById("outerDiv");
let innerDiv = document.getElementById("innerDiv");

outerDiv.addEventListener("click", function() {
```

```
    console.log("Outer div bosildi");
});

innerDiv.addEventListener("click", function(event) {
    event.stopPropagation(); // Hodisaning tarqalishini
        to'xtatadi
    console.log("Inner div bosildi");
});
```

Delegatsiya (Delegation)

Hodisalarni boshqarishda delegatsiya usulidan foydalanish mumkin, bu esa xuddi shu hodisani bir nechta bolalar elementlariga qo’llashni osonlashtiradi.

```
let list = document.getElementById("myList");

list.addEventListener("click", function(event) {
    if (event.target.tagName === "LI") {
        alert("Ro'yxat elementi bosildi: " +
        event.target.textContent);
    }
});
```

Xulosa

Ushbu bo’limda biz JavaScriptdagi hodisalar bilan tanishdik. Hodisalar yordamida veb-sahifalaringizni interaktiv qilish, foydalanuvchi harakatlariga javob berish va sahifa bilan o’zaro aloqani yaxshilash mumkin. Keyingi bo’limda biz JavaScriptda shakllar bilan ishlashni o’rganamiz.

DOM Manipulyatsiyasi

DOM Nima?

DOM (Document Object Model) — bu HTML va XML hujjatlarning tuzilishini ifodalovchi dasturiy interfeys. DOM hujjatning elementlarini daraxt tuzilmasida tashkil etadi, bunda har bir tugun (node) hujjatdagi elementni ifodalaydi. JavaScript yordamida DOM orqali veb-sahifalarning tarkibini, tuzilishini va uslubini o’zgartirish mumkin.

DOM Elementlarini Tanlash

DOM elementlarini tanlash uchun JavaScriptda bir nechta metodlar mavjud.

`getElementById`

Elementni uning `id` atributi orqali tanlash.

```
let element = document.getElementById("myElement");
```

`getElementsByClassName`

Elementlarni ularning `class` atributi orqali tanlash. Bu metod HTMLCollection (elementlar to’plami) qaytaradi.

```
let elements = document.getElementsByClassName("myClass");
```

`getElementsByTagName`

Elementlarni teg nomi orqali tanlash. Bu metod ham HTMLCollection qaytaradi.

```
let elements = document.getElementsByTagName("div");
```

`querySelector`

Elementni CSS selektor orqali tanlash. Birinchi mos keladigan elementni qaytaradi.

```
let element = document.querySelector(".myClass");
```

`querySelectorAll`

Elementlarni CSS selektor orqali tanlash. Barcha mos keladigan elementlarni qaytaradi.

```
let elements = document.querySelectorAll(".myClass");
```

DOM Elementlarini O’zgartirish

DOM elementlarini tanlaganingizdan so’ng, ularga turli o’zgarishlar kiritishingiz mumkin.

Matnni O’zgartirish

Elementning ichki matnini `textContent` yoki `innerHTML` yordamida o’zgartirish.

```
let element = document.getElementById("myElement");
element.textContent = "Yangi matn";
```

HTML Kontentni O’zgartirish

Elementning ichki HTML kodini `innerHTML` yordamida o’zgartirish.

```
let element = document.getElementById("myElement");
element.innerHTML = "<strong>Yangi HTML</strong>";
```

CSS Stilini O’zgartirish

Elementning uslubini `style` xususiyati orqali o’zgartirish.

```
let element = document.getElementById("myElement");
element.style.color = "red";
element.style.fontSize = "20px";
```

Klasslarni Qo’shish va O’chirish

Elementning klasslarini `classList` yordamida boshqarish.

```
let element = document.getElementById("myElement");
element.classList.add("newClass"); // Klass qo'shish
element.classList.remove("oldClass"); // Klass o'chirish
element.classList.toggle("toggleClass"); // Klassni almashtirish
```

DOMga Yangi Element Qo’shish

DOMga yangi element qo’shish uchun yangi element yaratib, uni tegishli joyga qo’shish kerak.

Yangi Element Yaratish

`createElement` metodi yordamida yangi element yaratish.

```
let newElement = document.createElement("div");
newElement.textContent = "Yangi element";
```

Yangi Elementni Qo’shish

appendChild metodi yordamida yangi elementni ota elementga qo’shish.

```
let container = document.getElementById("container");
container.appendChild(newElement);
```

DOMdan Elementni O’chirish

DOMdan elementni o’chirish uchun ota elementdan removeChild metodini ishlatish kerak.

```
let element = document.getElementById("myElement");
element.parentNode.removeChild(element);
```

Hodisalarni Boshqarish (Event Handling)

JavaScript yordamida hodisalarni (events) boshqarish va ularga javob berish mumkin. Eng ko’p ishlatiladigan hodisalarga tugmalarni bosish, shakllarni yuborish, kiritmalar va boshqalar kiradi.

Hodisani Qo’shish

addEventListener metodi yordamida hodisalarni qo’shish.

```
let button = document.getElementById("myButton");

button.addEventListener("click", function() {
    alert("Tugma bosildi!");
});
```

Hodisa Turlari

JavaScriptda ko’plab hodisa turlari mavjud, jumladan:

- click — element bosilganda
- mouseover — kursor element ustiga kelganda
- mouseout — kursor elementdan chiqib ketganda
- submit — forma yuborilganda
- input — foydalanuvchi input qiymatini o’zgartirganda

Xulosa

Ushbu bo’limda biz DOM manipulyatsiyasi bilan tanishdik. DOM elementlarini tanlash, o’zgartirish, yangi elementlar qo’shish va hodisalarni

boshqarish haqida o’rgandik. Bu tushunchalar JavaScript yordamida interaktiv veb-sahifalar yaratishda juda muhimdir. Keyingi bo’limda biz JavaScriptda shakllar bilan ishlashni o’rganamiz.

HTML DOM Metodlari

Kirish

HTML Document Object Model (DOM) — bu veb-sahifaning tuzilishini va mazmunini JavaScript orqali boshqarish uchun ishlatiladigan API. DOM yordamida siz HTML elementlarini yaratish, o’zgartirish, o’chirish va ularning xususiyatlariga murojaat qilish imkoniga ega bo’lasiz.

DOM Elementlarini Tanlash

getElementById

Bu metod HTML hujjatidagi ID atributiga ega bo’lgan elementni tanlash uchun ishlatiladi:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>getElementById Misol</title>
</head>
<body>
    <h1 id="sarlavha">Salom, Dunyo!</h1>
    <script>
        let element = document.getElementById("sarlavha");
        console.log(element.innerText); // "Salom, Dunyo!"
    </script>
</body>
</html>
```

getElementsByClassName

Bu metod ma’lum bir klassga ega bo’lgan barcha elementlarni tanlash uchun ishlatiladi:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>getElementsByClassName Misol</title>
```

```
</head>
<body>
    <p class="matn">Biror matn</p>
    <p class="matn">Yana bir matn</p>
    <script>
        let elementlar = document.getElementsByClassName("matn");
        console.log(elementlar.length); // 2
        console.log(elementlar[0].innerText); // "Biror matn"
    </script>
</body>
</html>
```

getElementsByTagName

Bu metod ma’lum bir teglarga ega bo’lgan barcha elementlarni tanlash uchun ishlatiladi:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>getElementsByTagName Misol</title>
</head>
<body>
    <div>Div elementi</div>
    <div>Yana bir div elementi</div>
    <script>
        let divlar = document.getElementsByTagName("div");
        console.log(divlar.length); // 2
        console.log(divlar[0].innerText); // "Div elementi"
    </script>
</body>
</html>
```

querySelector va querySelectorAll

querySelector metod birinchi mos keluvchi elementni, querySelectorAll esa barcha mos keluvchi elementlarni qaytaradi:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>querySelector Misol</title>
</head>
<body>
    <p class="matn">Biror matn</p>
    <p class="matn">Yana bir matn</p>
    <script>
        let birinchiElement = document.querySelector(".matn");
        console.log(birinchiElement.innerText); // "Biror matn"

        let barchaElementlar =
        document.querySelectorAll(".matn");
        console.log(barchaElementlar.length); // 2
        console.log(barchaElementlar[1].innerText); // "Yana bir
        matn"
    </script>
</body>
</html>
```

DOM Elementlarini O’zgartirish

innerHTML va textContent

Bu xususiyatlar yordamida elementning ichki HTML yoki matn mazmunini o’zgartirish mumkin:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>innerHTML va textContent Misol</title>
</head>
<body>
    <div id="quti">Eski mazmun</div>
    <script>
        let quti = document.getElementById("quti");

        // innerHTML orqali o'zgartirish
        quti.innerHTML = "<strong>Yangi mazmun</strong>";

        // textContent orqali o'zgartirish
        quti.textContent = "Faqat matn";
```

```
    </script>
</body>
</html>
```

style Xususiyati

Elementning uslublarini o’zgartirish uchun `style` xususiyatidan foydalanish mumkin:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Style Misol</title>
</head>
<body>
    <p id="matn">Bu matn qizil rangda bo'ladi.</p>
    <script>
        let matn = document.getElementById("matn");
        matn.style.color = "red";
        matn.style.fontSize = "20px";
    </script>
</body>
</html>
```

classList Xususiyati

Element klasslarini qo’shish, o’chirish va almashtirish uchun `classList` xususiyatidan foydalanish mumkin:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>classList Misol</title>
    <style>
        .qizil { color: red; }
        .katta { font-size: 20px; }
    </style>
</head>
<body>
```

```
    <p id="matn">Klasslar bilan ishlash.</p>
    <script>
        let matn = document.getElementById("matn");

        // Klass qo'shish
        matn.classList.add("qizil");

        // Klass olib tashlash
        matn.classList.remove("katta");

        // Klassni almashtirish
        matn.classList.toggle("katta");
    </script>
</body>
</html>
```

DOMga Yangi Element Qo’shish

createElement va appendChild

Yangi element yaratish va uni DOMga qo’shish:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>createElement va appendChild Misol</title>
</head>
<body>
    <div id="quti"></div>
    <script>
        let quti = document.getElementById("quti");

        // Yangi paragraf yaratish
        let yangiParagraf = document.createElement("p");
        yangiParagraf.textContent = "Yangi paragraf";

        // Yangi paragrafni qo'shish
        quti.appendChild(yangiParagraf);
    </script>
</body>
</html>
```

DOMdan Elementni O’chirish

`removeChild`

Elementni DOMdan o’chirish uchun `removeChild` metodidan foydalanish mumkin:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>removeChild Misol</title>
</head>
<body>
    <div id="quti">
        <p id="eskiParagraf">Eski paragraf</p>
    </div>
    <script>
        let quti = document.getElementById("quti");
        let eskiParagraf =
        document.getElementById("eskiParagraf");

        // Elementni o'chirish
        quti.removeChild(eskiParagraf);
    </script>
</body>
</html>
```

Xulosa

Ushbu bo’limda siz HTML DOM metodlari yordamida elementlarni tanlash, o’zgartirish, qo’shish va o’chirishni o’rgandingiz. Bu usullar veb-sahifalarni dinamik va interaktiv qilish uchun juda muhimdir.

DOM Voqealari va Voqea Tinglovchilari

Kirish

Veb-sahifalarda interaktivlikni ta’minlash uchun voqealar va voqea tinglovchilari muhim rol o’ynaydi. Voqealar foydalanuvchi harakatlari (masalan, tugmachani bosish, kiritish, sahifani yuklash) natijasida yuzaga keladi. JavaScript yordamida biz ushbu voqealarni tinglash va ularga javob berishimiz mumkin.

Voqealarni Ko’rish

`onclick` Voqeasi

Eng sodda va keng tarqalgan voqea bu `onclick` voqeasi bo’lib, elementga bosish harakatini aniqlaydi.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>onclick Voqea Misoli</title>
</head>
<body>
    <button onclick="salom()">Bosing</button>
    <script>
        function salom() {
            alert("Salom, Dunyo!");
        }
    </script>
</body>
</html>
```

Voqea Tinglovchilari (Event Listeners)

`addEventListener` Metodi

Ko’proq moslashuvchanlik uchun `addEventListener` metodidan foydalanish tavsiya etiladi. Bu metod bir elementga bir nechta voqea tinglovchilarini qo’shishga imkon beradi.

Bosish Voqeasi

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>addEventListener Misoli</title>
</head>
<body>
    <button id="tugma">Bosing</button>
    <script>
        document.getElementById("tugma").addEventListener("click",
        function() {
            alert("Salom, Dunyo!");
        });
    </script>
</body>
</html>
```

Bir Nechta Voqea Tinglovchilarini Qo'shish

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Bir Nechta Voqea Tinglovchilari</title>
</head>
<body>
    <button id="tugma">Bosing</button>
    <script>
        let tugma = document.getElementById("tugma");
        tugma.addEventListener("click", function() {
            alert("Bir marta bosildi!");
        });
        tugma.addEventListener("click", function() {
            console.log("Tugma bosildi!");
        });
    </script>
</body>
</html>
```

Tez-tez Ishlatiladigan Voqealar

onmouseover va onmouseout

Bu voqealar element ustiga sichqoncha olib borilganda va undan olib chiqilganda yuz beradi.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>onmouseover va onmouseout Misoli</title>
</head>
<body>
    <p id="matn">Sichqoncha bu matn ustiga olib borilganda rangi
        o'zgaradi.</p>
    <script>
        let matn = document.getElementById("matn");
        matn.addEventListener("mouseover", function() {
            matn.style.color = "red";
        });
        matn.addEventListener("mouseout", function() {
            matn.style.color = "black";
        });
    </script>
</body>
</html>
```

onkeydown va onkeyup

Bu voqealar klaviaturadan tugma bosilganda va qo’yib yuborilganda yuz beradi.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>onkeydown va onkeyup Misoli</title>
</head>
<body>
    <input type="text" id="kiritish" placeholder="Bu yerga
        yozing...">
```

```
    <script>
        let kiritish = document.getElementById("kiritish");
        kiritish.addEventListener("keydown", function(event) {
            console.log("Tugma bosildi: " + event.key);
        });
        kiritish.addEventListener("keyup", function(event) {
            console.log("Tugma qo'yib yuborildi: " + event.key);
        });
    </script>
</body>
</html>
```

Voqeani Bekor Qilish

preventDefault

Ba’zi voqealar standart xatti-harakatga ega. Masalan, formani yuborish voqeasi sahifani qayta yuklaydi. Bu xatti-harakatni `preventDefault` metodi yordamida bekor qilish mumkin.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>preventDefault Misoli</title>
</head>
<body>
    <form id="forma">
        <input type="text" placeholder="Ism">
        <button type="submit">Yuborish</button>
    </form>
    <script>
        let forma = document.getElementById("forma");
        forma.addEventListener("submit", function(event) {
            event.preventDefault();
            alert("Forma yuborilmadi!");
        });
    </script>
</body>
</html>
```

Voqeani Tashlash

stopPropagation

Voqealar zanjirli ravishda yuqori darajadagi elementlarga o'tadi. Bu xatti-harakatni `stopPropagation` metodi yordamida to'xtatish mumkin.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>stopPropagation Misoli</title>
</head>
<body>
    <div id="ota">
        <button id="tugma">Bosing</button>
    </div>
    <script>
        let ota = document.getElementById("ota");
        let tugma = document.getElementById("tugma");

        ota.addEventListener("click", function() {
            alert("Ota element bosildi");
        });

        tugma.addEventListener("click", function(event) {
            event.stopPropagation();
            alert("Tugma bosildi");
        });
    </script>
</body>
</html>
```

Xulosa

Ushbu bo'limda siz HTML DOM voqealari va voqea tinglovchilari haqida batafsil ma'lumot oldingiz. Siz `onclick`, `addEventListener`, `preventDefault`, va `stopPropagation` kabi voqea metodlarini o'rganib, sahifani qanday qilib interaktiv qilishni bilib oldingiz.

DOM orqali CSSni boshqarish

Kirish

JavaScript yordamida HTML elementlarning CSS xususiyatlarini boshqarish mumkin. Bu sahifa ko’rinishini dinamik ravishda o’zgartirishga imkon beradi. JavaScript orqali CSS uslublarini boshqarishning bir necha usullari mavjud.

Elementning `style` Xususiyati

JavaScriptda elementning uslubini o’zgartirishning eng sodda usuli — bu `style` xususiyatidan foydalanishdir.

Uslubni O’zgartirish

Elementning CSS uslubini o’zgartirish uchun `style` xususiyatidan foydalanish mumkin. Misol:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Style Xususiyati Misol</title>
</head>
<body>
    <p id="matn">Bu matn rangini o'zgartiradi.</p>
    <button onclick="rangOzgartirish()">Rangni O'zgartirish</
        button>

    <script>
        function rangOzgartirish() {
            let matn = document.getElementById("matn");
            matn.style.color = "blue";
        }
    </script>
</body>
</html>
```

Bir Nechta Uslubni O’zgartirish

Bir nechta uslubni o’zgartirish uchun quyidagi usuldan foydalanish mumkin:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Bir Nechta Uslubni O'zgartirish Misol</title>
</head>
<body>
    <p id="matn">Bu matnning ko'rinishi o'zgaradi.</p>
    <button onclick="uslubOzgartirish()">Uslubni O'zgartirish</
        button>

    <script>
        function uslubOzgartirish() {
            let matn = document.getElementById("matn");
            matn.style.color = "green";
            matn.style.fontSize = "24px";
            matn.style.fontWeight = "bold";
        }
    </script>
</body>
</html>
```

Klasslarni Qo’shish va O’chirish

Elementlarga klasslar qo’shish va olib tashlash CSS uslublarini boshqarishning yanada kuchli usuli hisoblanadi. Klasslar yordamida ko’plab elementlarning uslublarini bir vaqtning o’zida boshqarish mumkin.

`classList` Xususiyati

`classList` xususiyati yordamida klasslarni qo’shish, olib tashlash va almashtirish mumkin.

Klass Qo’shish

Klass qo’shish uchun `classList.add` metodidan foydalaning:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>classList.add Misol</title>
```

```
    <style>
        .yangi-klass {
            color: red;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <p id="matn">Bu matnga yangi klass qo'shiladi.</p>
    <button onclick="klassQosh()">Klass Qo'shish</button>

    <script>
        function klassQosh() {
            let matn = document.getElementById("matn");
            matn.classList.add("yangi-klass");
        }
    </script>
</body>
</html>
```

Klass Olib Tashlash

Klass olib tashlash uchun `classList.remove` metodidan foydalaning:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>classList.remove Misol</title>
    <style>
        .klass {
            color: blue;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p id="matn" class="klass">Bu matndan klass olib
        tashlanadi.</p>
    <button onclick="klassOlibTashlash()">Klass Olib Tashlash</
        button>

    <script>
```

```
        function klassOlibTashlash() {
            let matn = document.getElementById("matn");
            matn.classList.remove("klass");
        }
    </script>
</body>
</html>
```

Klassni Almashtirish

Klassni almashtirish uchun `classList.toggle` metodidan foydalaning:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>classList.toggle Misol</title>
    <style>
        .faol {
            color: green;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <p id="matn">Bu matnga klass almashtiriladi.</p>
    <button onclick="klassAlmashtirish()">Klass Almashtirish</
        button>

    <script>
        function klassAlmashtirish() {
            let matn = document.getElementById("matn");
            matn.classList.toggle("faol");
        }
    </script>
</body>
</html>
```

CSSni `setAttribute` Metodi orqali Boshqarish

Elementning `style` atributini `setAttribute` metodi yordamida o’zgartirish mumkin:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>setAttribute Misol</title>
</head>
<body>
    <p id="matn">Bu matnning uslubi o'zgaradi.</p>
    <button onclick="uslubOzgartirish()">Uslubni O'zgartirish</
        button>

    <script>
        function uslubOzgartirish() {
            let matn = document.getElementById("matn");
            matn.setAttribute("style",
        "color: purple; font-size: 22px;");
        }
    </script>
</body>
</html>
```

Xulosa

Ushbu bo’limda siz JavaScript yordamida HTML elementlarining CSS uslublarini boshqarishni o’rgandingiz. `style` xususiyati orqali uslublarni o’zgartirish, `classList` yordamida klasslarni boshqarish va `setAttribute` metodidan foydalanishni o’rgandingiz. Bu usullar veb-sahifalarni dinamik va interaktiv qilish uchun juda muhimdir.

Shakllar bilan Ishlash (Working with Forms)

Shakllar va Ularning Elementlari

Shakllar (forms) veb-sahifalarda foydalanuvchi ma’lumotlarini yig’ish uchun ishlatiladi. HTML shakli `<form>` tegi yordamida yaratiladi va ichida turli kiritma (input) elementlarini o’z ichiga oladi, masalan, matn kiritma maydonlari, radio tugmalar, katakchalar (checkbox), va boshqalar.

```
<form id="myForm">
    <label for="name">Ismingiz:</label>
    <input type="text" id="name" name="name">

    <label for="email">Email:</label>
    <input type="email" id="email" name="email">

    <button type="submit">Yuborish</button>
</form>
```

Shakl Elementlarini Tanlash

JavaScript yordamida shakl elementlarini tanlash va ularga kirish mumkin.

`getElementById`

Elementni uning `id` atributi orqali tanlash.

```
let nameInput = document.getElementById("name");
let emailInput = document.getElementById("email");
```

`querySelector`

Elementni CSS selektor orqali tanlash.

```
let nameInput = document.querySelector("#name");
let emailInput = document.querySelector("#email");
```

Shakl Ma’lumotlarini Olish va O’rnatish

Shakl elementlarining qiymatlarini olish va o’rnatish uchun `value` xususiyatidan foydalaniladi.

Qiymatni Olish

```
let name = nameInput.value;
let email = emailInput.value;

console.log("Ism: " + name);
console.log("Email: " + email);
```

Qiymatni O’rnatish

```
nameInput.value = "Ali";
emailInput.value = "ali@example.com";
```

Shaklni Yuborish Hodisasi

Shakl yuborilganda hodisani ushlab qolish va JavaScript yordamida qo’shimcha ishlov berish mumkin. Buning uchun `submit` hodisasidan foydalaniladi.

Shaklni Yuborish Hodisasini Qo’shish

```
let form = document.getElementById("myForm");

form.addEventListener("submit", function(event) {
    event.preventDefault(); // Shakl yuborilishini bekor qiladi
    console.log("Shakl yuborilishi bekor qilindi");

    let name = nameInput.value;
    let email = emailInput.value;

    console.log("Ism: " + name);
    console.log("Email: " + email);
});
```

Shaklni Tekshirish (Validation)

Shakl yuborilishidan oldin uning ma’lumotlarini tekshirish uchun JavaScriptni ishlatish mumkin.

Oddiy Tekshirish Misoli

```
form.addEventListener("submit", function(event) {
    event.preventDefault(); // Shakl yuborilishini bekor qiladi

    let name = nameInput.value;
    let email = emailInput.value;
```

```
    if (name === "" || email === "") {
        alert("Barcha maydonlarni to'ldiring!");
    } else {
        console.log("Ism: " + name);
        console.log("Email: " + email);
        // Bu yerda shaklni serverga yuborish kodini
        qo'shishingiz mumkin
    }
});
```

Qo’shimcha Elementlar bilan Ishlash

Shakllar turli elementlarni o’z ichiga oladi, masalan, radio tugmalar, katakchalar, tanlov qutilari (select box), va boshqalar.

Radio Tugmalar

```
<label>
    <input type="radio" name="gender" value="male"> Erkak
</label>
<label>
    <input type="radio" name="gender" value="female"> Ayol
</label>
```

Qiymatni olish:

```
let gender =
        document.querySelector('input[name="gender"]:checked').value;
console.log("Jinsi: " + gender);
```

Katakchalar (Checkbox)

```
<label>
    <input type="checkbox" id="subscribe" name="subscribe">
        Yangiliklarga obuna bo'lish
</label>
```

Qiymatni olish:

```
let isSubscribed = document.getElementById("subscribe").checked;
console.log("Obuna bo'ldingizmi: " + isSubscribed);
```

Tanlov Qutilari (Select Box)

```
<select id="country" name="country">
    <option value="uz">O'zbekiston</option>
```

```
    <option value="us">AQSh</option>
    <option value="uk">Birlashgan Qirollik</option>
</select>
```

Qiymatni olish:

```
let country = document.getElementById("country").value;
console.log("Davlat: " + country);
```

Xulosa

Ushbu bo’limda biz JavaScriptda shakllar bilan ishlashni ko’rib chiqdik. Shakl elementlarini tanlash, qiymatlarni olish va o’rnatish, shakl yuborilish hodisasini boshqarish va shakl ma’lumotlarini tekshirish haqida o’rgandik. Keyingi bo’limda biz AJAX va asinxron so’rovlar bilan ishlashni o’rganamiz.

JSON (JavaScript Object Notation)

JSON — bu ma’lumotlarni almashtirish uchun yengil format. U JavaScript ob’ektlariga asoslangan bo’lib, ma’lumotlarni inson o’qiy oladigan va mashina tushuna oladigan tarzda ifodalash uchun ishlatiladi.

JSONning asosiy xususiyatlari

- **Yengil**: JSON sodda va ixcham formatda yoziladi.
- **Inson o’qiy oladigan**: JSON ma’lumotlari oson tushunarli va o’qilishi qulay.
- **Mashina tushuna oladigan**: JSON ma’lumotlarini mashinalar oson parsilashadi va generatsiya qilishadi.

JSON Sintaksisi

JSON quyidagi ma’lumot turlarini qo’llab-quvvatlaydi:

- **Ob’ektlar**: `{ }` qavslar ichida yoziladi.
- **Massivlar**: `[ ]` qavslar ichida yoziladi.
- **Qatorlar**: Ikkilik qo’shtirnoq ichida yoziladi.
- **Raqamlar**: Butun sonlar va o’nli kasrlar.
- **Boolean**: `true` yoki `false` qiymatlar.
- **Null**: `null` qiymat.

JSON Misollari

JSON Ob’ekt

```
{
  "ism": "Ali",
  "yosh": 25,
  "talaba": true,
  "manzil": {
    "shahar": "Toshkent",
    "pochta_kodi": 100000
  },
  "telefonlar": ["998901234567", "998912345678"]
}
```

Yuqoridagi misolda bizda `ism`, `yosh`, `talaba`, `manzil` va `telefonlar` kabi kalit-qiymat juftliklari mavjud.

JSON Massiv

```
[
  {
    "ism": "Ali",
    "yosh": 25
  },
  {
    "ism": "Vali",
    "yosh": 30
  }
]
```

Yuqoridagi misolda ikki ob’ektni o’z ichiga olgan massiv mavjud, har bir ob’ekt `ism` va `yosh` kalitlariga ega.

JavaScriptda JSON bilan ishlash

JavaScriptda JSON bilan ishlash uchun `JSON.parse()` va `JSON.stringify()` metodlaridan foydalaniladi.

JSON.parse()

`JSON.parse()` metodi JSON qatorini JavaScript ob’ektiga aylantiradi.

```
let jsonString = '{"ism": "Ali", "yosh": 25, "talaba": true}';
let obj = JSON.parse(jsonString);

console.log(obj.ism); // Natija: Ali
console.log(obj.yosh); // Natija: 25
console.log(obj.talaba); // Natija: true
```

JSON.stringify()

`JSON.stringify()` metodi JavaScript ob’ektini JSON qatoriga aylantiradi.

```
let obj = {
  ism: "Ali",
  yosh: 25,
  talaba: true
};

let jsonString = JSON.stringify(obj);

console.log(jsonString);
// Natija: '{"ism":"Ali","yosh":25,"talaba":true}'
```

JSON bilan ishlashga oid mashqlar

1. **Mashq**: Quyidagi JSON qatorini JavaScript ob’ektiga aylantiring va `ism` va `telefon` qiymatlarini konsolga chiqaring.

   ```
   '{"ism": "Zafar", "telefon": "998901234567"}'
   ```

   Yechim:

   ```
   let jsonString = '{"ism": "Zafar", "telefon":
           "998901234567"}';
   let obj = JSON.parse(jsonString);

   console.log(obj.ism); // Natija: Zafar
   console.log(obj.telefon); // Natija: 998901234567
   ```

2. **Mashq**: Quyidagi JavaScript ob’ektini JSON qatoriga aylantiring.

   ```
   let talaba = {
     ism: "Olim",
     yosh: 22,
     kurs: 3
   };
   ```

   Yechim:

   ```
   let talaba = {
     ism: "Olim",
     yosh: 22,
     kurs: 3
   };

   let jsonString = JSON.stringify(talaba);

   console.log(jsonString);
   // Natija: '{"ism":"Olim","yosh":22,"kurs":3}'
   ```

Xulosa

Bu darsd JSON haqida boshlang’ich tushunchalar va JavaScriptda JSON bilan ishlash usullarinio’rgandik. Kitobning bu qismi boshlang’ich darajadagi dasturchilar uchun yaxshi boshlanish nuqtasi bo’ladi.

Tarmoq So’rovlari (XMLHttpRequest va Fetch Api)

XMLHttpRequest

XMLHttpRequest ob’ekti JavaScript-da serverga so’rov yuborish va javob olish uchun ishlatiladi. Ushbu metod eski bo’lsa-da, hali ham ko’plab loyihalarda foydalaniladi.

XMLHttpRequest misoli

```
let xhr = new XMLHttpRequest();
xhr.open("GET", "https://jsonplaceholder.typicode.com/posts",
        true);

xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200) {
    let response = JSON.parse(xhr.responseText);
    console.log(response);
  }
};

xhr.send();
```

fetch API

fetch API yangi va qulayroq usulda tarmoq so’rovlarini amalga oshirish imkonini beradi. U promises asosida ishlaydi va so’rovlar bilan ishlashni osonlashtiradi.

fetch API misoli

```
fetch("https://jsonplaceholder.typicode.com/posts")
  .then(response => {
    if (!response.ok) {
      throw new Error("Tarmoqda xato: " + response.statusText);
    }
    return response.json();
  })
  .then(data => {
    console.log(data);
  })
```

```
  .catch(error => {
    console.error("Xato sodir bo'ldi:", error);
  });
```

async/await

async/await sintaksisi promises bilan ishlashni yanada sodda va o'qilishi oson qiladi.

async/await misoli

```
async function fetchData() {
  try {
    let response = await fetch("https://
        jsonplaceholder.typicode.com/posts");
    if (!response.ok) {
      throw new Error("Tarmoqda xato: " + response.statusText);
    }
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.error("Xato sodir bo'ldi:", error);
  }
}

fetchData();
```

Tarmoq So'rovlarini Amalga Oshirish

GET So'rovi

GET so'rovi serverdan ma'lumot olish uchun ishlatiladi.

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error("Xato sodir bo'ldi:", error));
```

POST So'rovi

POST so'rovi serverga yangi ma'lumot jo'natish uchun ishlatiladi.

```
fetch("https://jsonplaceholder.typicode.com/posts", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
```

```
  },
  body: JSON.stringify({
    title: "Yangi post",
    body: "Bu yangi postning mazmuni.",
    userId: 1
  })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error("Xato sodir bo'ldi:", error));
```

PUT So’rovi

PUT so’rovi serverdagi mavjud ma’lumotni yangilash uchun ishlatiladi.

```
fetch("https://jsonplaceholder.typicode.com/posts/1", {
  method: "PUT",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify({
    id: 1,
    title: "Yangilangan post",
    body: "Bu yangilangan postning mazmuni.",
    userId: 1
  })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error("Xato sodir bo'ldi:", error));
```

DELETE So’rovi

DELETE so’rovi serverdagi ma’lumotni o’chirish uchun ishlatiladi.

```
fetch("https://jsonplaceholder.typicode.com/posts/1", {
  method: "DELETE"
})
  .then(() => console.log("Post o'chirildi"))
  .catch(error => console.error("Xato sodir bo'ldi:", error));
```

Mashqlar

1. **Mashq**: Quyidagi URL-dan foydalangan holda GET so’rovi yuboring va konsolda natijani chiqaring.

`https://jsonplaceholder.typicode.com/users`

Yechim:

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error("Xato sodir bo'ldi:",
        error));
```

2. **Mashq**: Quyidagi URL-ga yangi post yuboring va javobni konsolda chiqaring.

   `https://jsonplaceholder.typicode.com/posts`

   Post ma’lumotlari:

   ```
   {
     "title": "Yangi post",
     "body": "Bu yangi postning mazmuni.",
     "userId": 1
   }
   ```

   Yechim:

   ```
   fetch("https://jsonplaceholder.typicode.com/posts", {
     method: "POST",
     headers: {
       "Content-Type": "application/json"
     },
     body: JSON.stringify({
       title: "Yangi post",
       body: "Bu yangi postning mazmuni.",
       userId: 1
     })
   })
     .then(response => response.json())
     .then(data => console.log(data))
     .catch(error => console.error("Xato sodir bo'ldi:",
           error));
   ```

3. **Mashq**: Quyidagi URL-dagi postni yangilang va javobni konsolda chiqaring.

   `https://jsonplaceholder.typicode.com/posts/1`

   Yangilangan post ma’lumotlari:

```
{
  "id": 1,
  "title": "Yangilangan post",
  "body": "Bu yangilangan postning mazmuni.",
  "userId": 1
}
```

Yechim:

```
fetch("https://jsonplaceholder.typicode.com/posts/1", {
  method: "PUT",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify({
    id: 1,
    title: "Yangilangan post",
    body: "Bu yangilangan postning mazmuni.",
    userId: 1
  })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error("Xato sodir bo'ldi:",
        error));
```

Xulosa

Bu darsda JavaScript-da tarmoq so’rovlarini amalga oshirish usullari haqida batafsil ma’lumot berildi.

Yuqori darajadagi mavzular (Advanced Topics)

ES6+ Xususiyatlari

Template Literals

Template literals (shablon qatorlari) qatorlarni yozishda va interpolatsiya qilishda yangi va qulay usuldir.

```
let ism = "Ali";
let yosh = 25;

let salomlashish = `Salom, mening ismim ${ism} va yoshim $
        {yosh}.`;
console.log(salomlashish);
// Natija: Salom, mening ismim Ali va yoshim 25.
```

Destructuring (Tuzilmadan ajratish)

Destructuring massiv va ob’ektlardan qiymatlarni osongina chiqarib olish imkonini beradi.

Ob’ekt Destructuring

```
let talaba = {
  ism: "Vali",
  yosh: 20,
  fakultet: "Informatika"
};

let { ism, yosh, fakultet } = talaba;
console.log(ism); // Natija: Vali
console.log(yosh); // Natija: 20
console.log(fakultet); // Natija: Informatika
```

Massiv Destructuring

```
let raqamlar = [1, 2, 3, 4, 5];

let [birinchi, ikkinchi, uchinchi] = raqamlar;
console.log(birinchi); // Natija: 1
console.log(ikkinchi); // Natija: 2
console.log(uchinchi); // Natija: 3
```

Spread va Rest Operatorlari

Spread Operator (...)

Spread operatori massiv yoki ob'ekt elementlarini yangi massiv yoki ob'ektga yoyish uchun ishlatiladi.

```
let array1 = [1, 2, 3];
let array2 = [...array1, 4, 5, 6];

console.log(array2); // Natija: [1, 2, 3, 4, 5, 6]

let objekt1 = { a: 1, b: 2 };
let objekt2 = { ...objekt1, c: 3 };

console.log(objekt2); // Natija: { a: 1, b: 2, c: 3 }
```

Rest Operator (...)

Rest operatori funksiya parametrlaridan yoki massiv elementlaridan qolgan barcha qiymatlarni bir joyga to'plash uchun ishlatiladi.

```
function yigish(...raqamlar) {
  return raqamlar.reduce((sum, current) => sum + current, 0);
}

console.log(yigish(1, 2, 3, 4)); // Natija: 10
```

Promises va Async/Await

Promises

Promise JavaScriptda asinxron operatsiyalarni boshqarish uchun ishlatiladi.

```
let vada = new Promise((resolve, reject) => {
  let muvaffaqiyat = true;

  if (muvaffaqiyat) {
    resolve("Ish muvaffaqiyatli yakunlandi.");
  } else {
    reject("Ish muvaffaqiyatsiz yakunlandi.");
  }
});

vada
  .then((natija) => {
    console.log(natija);
```

```
  })
  .catch((xato) => {
    console.log(xato);
  });
```

Async/Await

Async/await sintaksisi promises bilan ishlashni yanada sodda qiladi.

```
function vaqtOladi(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}

async function misol() {
  console.log("Boshlanish...");
  await vaqtOladi(2000);
  console.log("2 soniya o'tdi.");
}

misol();
```

Modullar

JavaScriptda modullar kodni turli bo’laklarga bo’lish va qayta ishlatish imkonini beradi. `export` va `import` kalit so’zlari yordamida modullarni yaratish va ulardan foydalanish mumkin.

```
// fayl: math.js
export function qo'shish(a, b) {
  return a + b;
}

export function ayirish(a, b) {
  return a - b;
}

// fayl: app.js
import { qo'shish, ayirish } from './math.js';

console.log(qo'shish(5, 3)); // Natija: 8
console.log(ayirish(5, 3)); // Natija: 2
```

Xatolarni Boshqarish

JavaScriptda xatolarni boshqarish uchun `try`, `catch`, va `finally` bloklari ishlatiladi.

```
try {
  // Xato sodir bo'lishi mumkin bo'lgan kod
  let natija = x / y;
  console.log(natija);
} catch (xato) {
  // Xato sodir bo'lganda bajariladigan kod
  console.log("Xato sodir bo'ldi: " + xato.message);
} finally {
  // Har doim bajariladigan kod
  console.log("Bu har doim bajariladi.");
}
```

Bu yerda JavaScriptning ba’zi ilg’or mavzulari haqida qisqacha ma’lumot berildi. Bu bo’lim talabalarga murakkabroq xususiyatlar va dasturlash texnikalari bilan tanishish imkonini beradi.

JavaScript Window Navigator API

Kirish

Window Navigator API brauzer va foydalanuvchi qurilmasi haqida ma’lumot olish uchun ishlatiladi. Bu API veb-saytlarga brauzer haqida ma’lumot beradi, masalan, brauzer turi, versiyasi, foydalanuvchi operatsion tizimi va boshqa ko’plab foydali ma’lumotlar.

Navigator Ob’ekti

Navigator API `window.navigator` ob’ekti orqali kiriladi. Bu ob’ekt quyidagi xususiyatlar va metodlarni o’z ichiga oladi:

`navigator.appName`

Brauzer nomini qaytaradi:

```
console.log(navigator.appName);
```

`navigator.appVersion`

Brauzer versiyasi haqida ma’lumot qaytaradi:

```
console.log(navigator.appVersion);
```

`navigator.userAgent`

Brauzer foydalanuvchi agenti satrini qaytaradi. Bu satr brauzer va operatsion tizim haqida batafsil ma’lumot beradi:

```
console.log(navigator.userAgent);
```

`navigator.platform`

Foydalanuvchi qurilmasi operatsion tizimini qaytaradi:

```
console.log(navigator.platform);
```

`navigator.language`

Foydalanuvchi brauzeri tilini qaytaradi:

```
console.log(navigator.language);
```

navigator.onLine

Foydalanuvchining onlayn yoki oflayn holatini qaytaradi:

```
if (navigator.onLine) {
    console.log("Foydalanuvchi onlayn.");
} else {
    console.log("Foydalanuvchi oflayn.");
}
```

Geolocation API

Navigator API tarkibiga geolokatsiya xizmati ham kiradi. Bu foydalanuvchining joylashuv ma’lumotlarini olish uchun ishlatiladi.

navigator.geolocation.getCurrentPosition

Bu metod foydalanuvchining hozirgi joylashuvini olish uchun ishlatiladi:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Geolocation Misol</title>
</head>
<body>
    <button onclick="getLocation()">Joylashuvni olish</button>
    <p id="joylashuv"></p>

    <script>
        function getLocation() {
            if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(showPosition,
        showError);
            } else {
                document.getElementById("joylashuv").innerHTML =
        "Geolokatsiya xizmati qo'llab-quvvatlanmaydi.";
            }
        }

        function showPosition(position) {
            document.getElementById("joylashuv").innerHTML =
                "Kenglik: " + position.coords.latitude + "<br>" +
                "Uzunlik: " + position.coords.longitude;
```

```
        }

        function showError(error) {
            switch(error.code) {
                case error.PERMISSION_DENIED:
                    document.getElementById("joylashuv").innerHTML
        = "Foydalanuvchi joylashuv so'rovini rad etdi.";
                    break;
                case error.POSITION_UNAVAILABLE:
                    document.getElementById("joylashuv").innerHTML
        = "Joylashuv ma'lumotlari mavjud emas.";
                    break;
                case error.TIMEOUT:
                    document.getElementById("joylashuv").innerHTML
        = "Joylashuv so'rovi vaqtni tugatdi.";
                    break;
                case error.UNKNOWN_ERROR:
                    document.getElementById("joylashuv").innerHTML
        = "Noma'lum xato yuz berdi.";
                    break;
            }
        }
    </script>
</body>
</html>
```

Battery Status API

Battery Status API yordamida foydalanuvchi qurilmasining batareya holatini bilish mumkin. Bu xususiyat ko’p brauzerlarda hali qo’llab-quvvatlanmaydi, lekin kelajakda keng qo’llanilishi mumkin.

navigator.getBattery

Bu metod batareya haqida ma’lumot olish uchun ishlatiladi:

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Battery Status Misol</title>
</head>
<body>
```

```
    <button onclick="getBatteryStatus()">Batareya holatini
        olish</button>
    <p id="batareya"></p>

    <script>
        function getBatteryStatus() {
            navigator.getBattery().then(function(battery) {
                let batareyaInfo = "Batareya darajasi: " +
        battery.level * 100 + "%<br>" +
                                    "Zaryad olayapti: " +
        (battery.charging ? "Ha" : "Yo'q") + "<br>";
                document.getElementById("batareya").innerHTML =
        batareyaInfo;
            });
        }
    </script>
</body>
</html>
```

Xulosa

Ushbu bo’limda siz JavaScript Window Navigator API va uning asosiy xususiyatlari haqida ma’lumot oldingiz. Navigator API yordamida brauzer va foydalanuvchi qurilmasi haqida ko’p foydali ma’lumotlarga ega bo’lish mumkin. Navigator API tarkibiga kiruvchi Geolocation API va Battery Status API yordamida foydalanuvchi joylashuvi va batareya holatini bilish mumkin.

Ikkilik Ma’lumotlar va Fayllar bilan Ishlash (Binary Data and Files)

ArrayBuffer va Typed Arrays

ArrayBuffer JavaScript-da ikkilik ma’lumotlarni saqlash uchun ishlatiladi. U bir nechta typed arrays yordamida manipulyatsiya qilinadi, masalan, Uint8Array, Int16Array, Float32Array va boshqalar.

ArrayBuffer yaratish va Typed Arrays bilan ishlash

```
// 8 baytli ArrayBuffer yaratish
let buffer = new ArrayBuffer(8);

// Uint32Array bilan ishlash (4 baytli elementlar)
let view = new Uint32Array(buffer);
view[0] = 123456;
view[1] = 654321;

console.log(view); // Natija: Uint32Array(2) [123456, 654321]
```

Blob

Blob (Binary Large Object) katta hajmdagi ikkilik ma’lumotlarni saqlash uchun ishlatiladi. U odatda fayllarni yoki fayl qismlarini saqlashda qo’llaniladi.

Blob yaratish

```
// Matndan Blob yaratish
let matn = "Bu matn Blob ob'ektida saqlanadi.";
let blob = new Blob([matn], { type: "text/plain" });

console.log(blob); // Natija: Blob { size: 31, type: "text/
        plain" }
```

Blob’dan URL yaratish

```
let blobURL = URL.createObjectURL(blob);
console.log(blobURL); // Natija: blob: URL manzili

// URL dan foydalanib faylni yuklab olish
```

```
let a = document.createElement("a");
a.href = blobURL;
a.download = "example.txt";
document.body.appendChild(a);
a.click();
document.body.removeChild(a);
```

FileReader

FileReader ob’ekti foydalanuvchi tanlagan fayllarni o’qish uchun ishlatiladi. U fayl ma’lumotlarini o’qish uchun bir necha metodlarni taqdim etadi, masalan, `readAsText`, `readAsArrayBuffer`, `readAsDataURL`.

FileReader bilan faylni o’qish

```
<input type="file" id="fileInput" />
<script>
  let fileInput = document.getElementById('fileInput');

  fileInput.addEventListener('change', function(event) {
    let file = event.target.files[0];
    let reader = new FileReader();

    reader.onload = function(event) {
      console.log(event.target.result); // Fayl mazmuni
    };

    reader.readAsText(file);
  });
</script>
```

FileReader bilan ikkilik faylni o’qish

```
fileInput.addEventListener('change', function(event) {
  let file = event.target.files[0];
  let reader = new FileReader();

  reader.onload = function(event) {
    let arrayBuffer = event.target.result;
    let uint8Array = new Uint8Array(arrayBuffer);

    console.log(uint8Array); // Faylning ikkilik ma'lumotlari
  };
```

```
  reader.readAsArrayBuffer(file);
});
```

Fayllarni yuklash va yuklab olish

Faylni yuklash

```
fetch('https://example.com/somefile')
  .then(response => response.blob())
  .then(blob => {
    let url = URL.createObjectURL(blob);
    let a = document.createElement('a');
    a.href = url;
    a.download = 'downloadedfile';
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
  })
  .catch(error => console.error('Xato sodir bo\'ldi:', error));
```

Faylni serverga yuklash

```
<input type="file" id="uploadInput" />
<script>
  let uploadInput = document.getElementById('uploadInput');

  uploadInput.addEventListener('change', function(event) {
    let file = event.target.files[0];
    let formData = new FormData();
    formData.append('file', file);

    fetch('https://example.com/upload', {
      method: 'POST',
      body: formData
    })
    .then(response => response.json())
    .then(data => {
      console.log('Yuklash muvaffaqiyatli:', data);
    })
    .catch(error => {
      console.error('Xato sodir bo\'ldi:', error);
    });
  });
</script>
```

Mashqlar

1. **Mashq**: Ikkilamchi ma’lumotlarni `ArrayBuffer` va `Uint8Array` yordamida yozish va o’qish.

   - 8 baytli `ArrayBuffer` yaratish.
   - `Uint8Array` yordamida bu bufferga qiymatlar yozish.
   - Qiymatlarni konsolga chiqarish.

   Yechim:

   ```
   let buffer = new ArrayBuffer(8);
   let view = new Uint8Array(buffer);

   view[0] = 1;
   view[1] = 2;
   view[2] = 3;
   view[3] = 4;

   console.log(view); // Natija: Uint8Array(8) [1, 2, 3, 4, 0,
           0, 0, 0]
   ```

2. **Mashq**: `FileReader` yordamida foydalanuvchi tanlagan faylni o’qish va konsolga chiqarish.

   - Fayl input elementini yaratish.
   - Fayl tanlanganda `FileReader` yordamida mazmunni o’qish.
   - Fayl mazmunini konsolga chiqarish.

   Yechim:

   ```
   let input = document.createElement("input");
   input.type = "file";

   input.onchange = (event) => {
     let file = event.target.files[0];
     let reader = new FileReader();

     reader.onload = (e) => {
       console.log("Fayl mazmuni: " + e.target.result);
     };

     reader.readAsText(file);
   };

   document.body.appendChild(input);
   ```

3. **Mashq**: `fetch` API yordamida serverdan binary ma’lumot olish va `Uint8Array` ga aylantirish.

    - Berilgan URL dan `fetch` yordamida binary ma’lumot olish.
    - Ma’lumotni `Uint8Array` ga aylantirish.
    - `Uint8Array` ni konsolga chiqarish.

    Yechim:

```
fetch("https://example.com/data.bin")
  .then(response => response.arrayBuffer())
  .then(buffer => {
    let uint8Array = new Uint8Array(buffer);
    console.log(uint8Array);
  })
  .catch(error => console.error("Xato sodir bo'ldi:",
        error));
```

Xulosa

Bu darsda JavaScript-da ikkilamchi ma’lumotlar va fayllar bilan ishlash haqida batafsil ma’lumot berildi.

Clipboard API

Clipboard API foydalanuvchiga clipboard-ga ma’lumot yozish yoki clipboard-dan ma’lumot o’qish imkonini beradi. Bu API asosan `navigator.clipboard` ob’ekti orqali amalga oshiriladi.

Matnni Clipboard-ga Nusxalash

Matnni clipboard-ga nusxalash uchun `navigator.clipboard.writeText()` metodidan foydalaniladi.

Misol: Matnni Clipboard-ga Nusxalash

```
<button id="copyButton">Matnni nusxalash</button>
<input type="text" id="textInput" value="Salom, dunyo!" />

<script>
document.getElementById("copyButton").addEventListener("click",
        () => {
  let text = document.getElementById("textInput").value;
  navigator.clipboard.writeText(text)
    .then(() => {
      console.log("Matn clipboard-ga nusxalandi.");
    })
    .catch(err => {
      console.error("Nusxalashda xato yuz berdi:", err);
    });
});
</script>
```

Clipboard-dan Matnni Olish

Clipboard-dan matnni olish uchun `navigator.clipboard.readText()` metodidan foydalaniladi.

Misol: Clipboard-dan Matnni Olish

```
<button id="pasteButton">Matnni qo'shish</button>
<input type="text" id="output" placeholder="Matn bu yerda
        ko'rsatiladi" />

<script>
document.getElementById("pasteButton").addEventListener("click",
        () => {
```

```
  navigator.clipboard.readText()
    .then(text => {
      document.getElementById("output").value = text;
      console.log("Matn clipboard-dan o'qildi.");
    })
    .catch(err => {
      console.error("O'qishda xato yuz berdi:", err);
    });
});
</script>
```

Clipboard API-ni qo’llab-quvvatlashni tekshirish

Clipboard API ishlashini tekshirish uchun quyidagi kodni ishlatishingiz mumkin:

```
if (navigator.clipboard) {
  console.log("Clipboard API qo'llab-quvvatlanadi.");
} else {
  console.log("Clipboard API qo'llab-quvvatlanmaydi.");
}
```

Amaliy Misollar

Matnni Avtomatik Nusxalash

Ushbu misolda, matn maydoniga e’tibor berilganda matn avtomatik ravishda clipboard-ga nusxalanadi.

```
<input type="text" id="autoCopyInput" value="Bu avtomatik
        nusxalanadigan matn." />

<script>
document.getElementById("autoCopyInput").addEventListener("focus",
        (event) => {
  navigator.clipboard.writeText(event.target.value)
    .then(() => {
      console.log("Matn avtomatik ravishda clipboard-ga
        nusxalandi.");
    })
    .catch(err => {
      console.error("Nusxalashda xato yuz berdi:", err);
    });
});
</script>
```

Clipboard-dan Matnni Avtomatik Olish

Ushbu misolda, tugmani bosganda clipboard-dan matn olinadi va matn maydoniga joylashtiriladi.

```
<button id="autoPasteButton">Matnni avtomatik qo'shish</button>
<input type="text" id="autoPasteInput"
        placeholder="Matn bu yerda ko'rsatiladi" />

<script>
document.getElementById("autoPasteButton").addEventListener("click",
        () => {
  navigator.clipboard.readText()
    .then(text => {
      document.getElementById("autoPasteInput").value = text;
      console.log("Matn clipboard-dan o'qildi va maydonga
        qo'shildi.");
    })
    .catch(err => {
      console.error("O'qishda xato yuz berdi:", err);
    });
});
</script>
```

Mashqlar

1. **Mashq**: Foydalanuvchi kiritgan matnni clipboard-ga nusxalash uchun input va tugma yarating.

    - Matnni kiritish uchun input elementi yarating.
    - Matnni clipboard-ga nusxalash uchun tugma yarating.
    - Tugma bosilganda matn clipboard-ga nusxalansin.

    Yechim:

    ```
    <input type="text" id="userText" placeholder="Matnni
            kiriting" />
    <button id="copyUserText">Matnni nusxalash</button>

    <script>
    document.getElementById("copyUserText").addEventListener("click",
            () => {
      let text = document.getElementById("userText").value;
      navigator.clipboard.writeText(text)
        .then(() => {
          console.log("Matn clipboard-ga nusxalandi.");
        })
    ```

```
    .catch(err => {
      console.error("Nusxalashda xato yuz berdi:", err);
    });
});
</script>
```

2. **Mashq**: Clipboard-dan matnni olish va matn maydoniga joylashtirish uchun tugma yarating.

   - Matnni olish uchun tugma yarating.
   - Clipboard-dan matn oling va boshqa matn maydoniga joylashtiring.

   Yechim:

```
<button id="pasteText">Clipboard-dan matnni olish</button>
<input type="text" id="pastedText"
       placeholder="Matn bu yerda ko'rsatiladi" />

<script>
document.getElementById("pasteText").addEventListener("click",
       () => {
  navigator.clipboard.readText()
    .then(text => {
      document.getElementById("pastedText").value = text;
      console.log("Matn clipboard-dan o'qildi.");
    })
    .catch(err => {
      console.error("O'qishda xato yuz berdi:", err);
    });
});
</script>
```

Xulosa

Bu darsda Clipboard API yordamida JavaScript-da clipboard bilan qanday ishlashni o’rgandik.

Notification API

Kirish

Notification API yordamida veb-sahifalar foydalanuvchiga brauzer orqali bildirishnomalar yuborishi mumkin. Bu API foydalanuvchining e'tiborini jalb qilish uchun foydalidir, masalan, yangi xabar kelganida yoki muhim hodisalar yuz berganida.

Bildirishnomalarni Ruxsat So'rash

Bildirishnomalar yuborishdan oldin foydalanuvchidan ruxsat so'rash kerak. Bu `Notification.requestPermission` metodi yordamida amalga oshiriladi.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Notification Ruxsatini So'rash</title>
</head>
<body>
    <button onclick="requestNotificationPermission()">Bildirishnoma
        Ruxsati</button>

    <script>
        function requestNotificationPermission() {
            Notification.requestPermission().then(function(permission)
        {
                if (permission === "granted") {
                    alert("Bildirishnoma ruxsati berildi.");
                } else {
                    alert("Bildirishnoma ruxsati rad etildi.");
                }
            });
        }
    </script>
</body>
</html>
```

Bildirishnoma Yaratish

Ruxsat olingandan so'ng, `new Notification` konstruktoridan foydalanib, bildirishnoma yaratish mumkin.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Bildirishnoma Yaratish</title>
</head>
<body>
    <button onclick="showNotification()">Bildirishnoma Yaratish</
        button>

    <script>
        function showNotification() {
            if (Notification.permission === "granted") {
                new Notification("Salom!", {
                    body: "Bu bir test bildirishnoma.",
                    icon: "https://via.placeholder.com/48"
                });
            } else {
                alert("Iltimos, avval bildirishnoma ruxsatini
        bering.");
            }
        }
    </script>
</body>
</html>
```

Bildirishnoma Variantlari

Bildirishnomalar uchun turli variantlar mavjud. Quyida ba’zi asosiy variantlar keltirilgan:

- `body`: Bildirishnoma mazmuni.
- `icon`: Bildirishnoma ikonkasi.
- `image`: Bildirishnoma ichida ko’rsatiladigan rasm.
- `badge`: Kichik ikona (mobil qurilmalar uchun).
- `vibrate`: Vibratsiya naqshi (mobil qurilmalar uchun).
- `actions`: Bildirishnoma tugmalari.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
```

```
    <title>Bildirishnoma Variantlari</title>
</head>
<body>
    <button onclick="showDetailedNotification()">To'liq
        Bildirishnoma</button>

    <script>
        function showDetailedNotification() {
            if (Notification.permission === "granted") {
                new Notification("Salom!", {
                    body: "Bu to'liq bildirishnoma misoli.",
                    icon: "https://via.placeholder.com/48",
                    image: "https://via.placeholder.com/150",
                    badge: "https://via.placeholder.com/48",
                    vibrate: [200, 100, 200],
                    actions: [
                        { action: "yes", title: "Ha" },
                        { action: "no", title: "Yo'q" }
                    ]
                });
            } else {
                alert("Iltimos, avval bildirishnoma ruxsatini
        bering.");
            }
        }
    </script>
</body>
</html>
```

Bildirishnoma Hodisalari

Bildirishnomalar bilan ishlashda ba’zi hodisalar mavjud. Bu hodisalar bildirishnoma ochilganda, yopilganda yoki foydalanuvchi tugmachani bosganda yuz beradi.

```
<!DOCTYPE html>
<html lang="uz">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Bildirishnoma Hodisalari</title>
</head>
<body>
    <button onclick="showEventNotification()">Bildirishnoma
        Hodisalari</button>
```

```
    <script>
        function showEventNotification() {
            if (Notification.permission === "granted") {
                let notification = new Notification("Salom!", {
                    body: "Bu bildirishnoma hodisalari misoli."
                });

                notification.onclick = function() {
                    alert("Bildirishnoma bosildi.");
                };

                notification.onclose = function() {
                    console.log("Bildirishnoma yopildi.");
                };
            } else {
                alert("Iltimos, avval bildirishnoma ruxsatini
        bering.");
            }
        }
    </script>
</body>
</html>
```

Xulosa

Ushbu bo’limda siz JavaScript Notification API haqida o’rgandingiz. Bildirishnomalarni yaratish, ruxsat so’rash va turli variantlarni qo’llash orqali foydalanuvchilarga ma’lumot yetkazish usullarini ko’rdik. Bu API foydalanuvchilarning e’tiborini jalb qilish va muhim ma’lumotlarni yetkazish uchun juda qulaydir.

Share API

Share API foydalanuvchilarga veb-sahifadagi kontentni boshqa dastur yoki qurilmalarga ulashish imkonini beradi. Bu API asosan mobil qurilmalarda ishlaydi va oddiy interfeys orqali kontentni ulashish imkoniyatini taqdim etadi.

Share API-ni Qo’llab-quvvatlashni Tekshirish

Birinchi navbatda, Share API qo’llab-quvvatlanishini tekshirish kerak. Bu quyidagicha amalga oshiriladi:

```
if (navigator.share) {
  console.log("Share API qo'llab-quvvatlanadi.");
} else {
  console.log("Share API qo'llab-quvvatlanmaydi.");
}
```

Kontentni Ulashish

Share API orqali kontentni ulashish uchun `navigator.share` metodidan foydalaniladi. Bu metod promises asosida ishlaydi.

Ulashish Misoli

Quyidagi misolda foydalanuvchi tugmani bosganda, sahifadagi kontent ulashiladi:

```
document.getElementById("shareButton").addEventListener("click",
        () => {
  if (navigator.share) {
    navigator.share({
      title: 'Men bilan ulashilgan veb-sahifa',
      text: 'Bu yerda ulashilgan kontent bor.',
      url: window.location.href
    })
    .then(() => {
      console.log("Kontent muvaffaqiyatli ulashildi.");
    })
    .catch(err => {
      console.error("Ulashishda xato yuz berdi:", err);
    });
  } else {
    console.error("Share API qo'llab-quvvatlanmaydi.");
```

```
  }
});
```

HTML:

```
<button id="shareButton">Kontentni ulashish</button>
```

Qo’shimcha Ma’lumotlarni Ulashish

Share API yordamida faqat matn va URL emas, balki boshqa ma’lumotlarni ham ulashish mumkin. Misol uchun, fayllarni ulashish ham mumkin.

Fayllarni Ulashish Misoli

```
document.getElementById("shareButton").addEventListener("click",
        () => {
  if (navigator.share) {
    const file = new File(["Hello, world!"], "hello.txt", {
      type: "text/plain",
    });

    navigator.share({
      files: [file],
      title: 'Men bilan ulashilgan fayl',
      text: 'Bu yerda ulashilgan fayl bor.',
    })
    .then(() => {
      console.log("Fayl muvaffaqiyatli ulashildi.");
    })
    .catch(err => {
      console.error("Ulashishda xato yuz berdi:", err);
    });
  } else {
    console.error("Share API qo'llab-quvvatlanmaydi.");
  }
});
```

HTML:

```
<button id="shareButton">Faylni ulashish</button>
```

Amaliy Misollar

Sahifadagi Ma’lumotlarni Ulashish

Misol uchun, foydalanuvchi sahifadagi biror bir matnni ulashishi mumkin.

```
document.getElementById("shareTextButton").addEventListener("click",
        () => {
  let textToShare =
        document.getElementById("shareText").innerText;

  if (navigator.share) {
    navigator.share({
      title: 'Men bilan ulashilgan matn',
      text: textToShare,
      url: window.location.href
    })
    .then(() => {
      console.log("Matn muvaffaqiyatli ulashildi.");
    })
    .catch(err => {
      console.error("Ulashishda xato yuz berdi:", err);
    });
  } else {
    console.error("Share API qo'llab-quvvatlanmaydi.");
  }
});
```

HTML:

```
<div id="shareText">Bu ulashiladigan matn.</div>
<button id="shareTextButton">Matnni ulashish</button>
```

Rasmlarni Ulashish

Foydalanuvchi sahifadagi rasmlarni ulashishi ham mumkin.

```
document.getElementById("shareImageButton").addEventListener("click",
        () => {
  fetch('image-url.jpg')
    .then(response => response.blob())
    .then(blob => {
      const file = new File([blob], 'image.jpg', { type:
        blob.type });

      if (navigator.share) {
        navigator.share({
          files: [file],
          title: 'Men bilan ulashilgan rasm',
          text: 'Bu yerda ulashilgan rasm bor.',
        })
        .then(() => {
```

```
        console.log("Rasm muvaffaqiyatli ulashildi.");
      })
      .catch(err => {
        console.error("Ulashishda xato yuz berdi:", err);
      });
    } else {
      console.error("Share API qo'llab-quvvatlanmaydi.");
    }
  });
});
```

HTML:

```
<button id="shareImageButton">Rasmni ulashish</button>
```

Mashqlar

1. **Mashq**: Foydalanuvchi sahifadagi URL ni ulashishi uchun tugma yarating.
   - Ulashish tugmasi yarating.
   - Tugma bosilganda, sahifadagi URL ni ulashing.

   Yechim:

   ```
   <button id="shareUrlButton">URL ni ulashish</button>

   <script>
   document.getElementById("shareUrlButton").addEventListener("click",
           () => {
     if (navigator.share) {
       navigator.share({
         title: 'Men bilan ulashilgan URL',
         text: 'Bu yerda ulashilgan URL bor.',
         url: window.location.href
       })
       .then(() => {
         console.log("URL muvaffaqiyatli ulashildi.");
       })
       .catch(err => {
         console.error("Ulashishda xato yuz berdi:", err);
       });
     } else {
       console.error("Share API qo'llab-quvvatlanmaydi.");
     }
   });
   </script>
   ```

2. **Mashq**: Foydalanuvchi sahifadagi biror bir matnni ulashishi uchun tugma yarating.
    - Ulashish tugmasi yarating.
    - Tugma bosilganda, sahifadagi matnni ulashing.

    Yechim:

```
<div id="shareText">Bu ulashiladigan matn.</div>
<button id="shareTextButton">Matnni ulashish</button>

<script>
document.getElementById("shareTextButton").addEventListener("click",
        () => {
  let textToShare =
        document.getElementById("shareText").innerText;

  if (navigator.share) {
    navigator.share({
      title: 'Men bilan ulashilgan matn',
      text: textToShare,
      url: window.location.href
    })
    .then(() => {
      console.log("Matn muvaffaqiyatli ulashildi.");
    })
    .catch(err => {
      console.error("Ulashishda xato yuz berdi:", err);
    });
  } else {
    console.error("Share API qo'llab-quvvatlanmaydi.");
  }
});
</script>
```

Bu darsda Share API yordamida JavaScript-da kontentni qanday ulashish mumkinligi haqida batafsil ma’lumot berildi.

LocalStorage va Sessiya Storage

Kirish

Mahalliy (localStorage) va sessiya saqlash (sessionStorage) veb-brauzerning saqlash API’laridir. Ular yordamida foydalanuvchi ma’lumotlarini brauzerda saqlash va keyinchalik foydalanish uchun qulay imkoniyat yaratiladi.

Mahalliy Saqlash (localStorage)

Mahalliy saqlash — bu foydalanuvchi brauzerida ma’lumotlarni saqlash usuli bo’lib, ma’lumotlar brauzer yopilganida ham saqlanib qoladi.

Sessiya Saqlash (sessionStorage)

Sessiya saqlash — bu foydalanuvchi brauzer sessiyasi davomida ma’lumotlarni saqlash usuli bo’lib, ma’lumotlar faqat bir sessiya davomida saqlanadi va brauzer yopilganda o’chiriladi.

Mahalliy Saqlashdan Foydalanish

Ma’lumotni Saqlash

Mahalliy saqlashga ma’lumot qo’shish uchun `localStorage.setItem` metodidan foydalanamiz:

```
localStorage.setItem('kalit', 'qiymat');
```

Misol:

```
localStorage.setItem('ism', 'Ali');
```

Ma’lumotni O’qish

Mahalliy saqlashdan ma’lumot olish uchun `localStorage.getItem` metodidan foydalanamiz:

```
let ism = localStorage.getItem('ism');
console.log(ism); // "Ali"
```

Ma’lumotni O’chirish

Mahalliy saqlashdagi ma’lumotni o’chirish uchun `localStorage.removeItem` metodidan foydalanamiz:

```
localStorage.removeItem('ism');
```

Barcha Ma’lumotlarni Tozalash

Mahalliy saqlashdagi barcha ma’lumotlarni tozalash uchun `localStorage.clear` metodidan foydalanamiz:

```
localStorage.clear();
```

Sessiya Saqlashdan Foydalanish

Ma’lumotni Saqlash

Sessiya saqlashga ma’lumot qo’shish uchun `sessionStorage.setItem` metodidan foydalanamiz:

```
sessionStorage.setItem('kalit', 'qiymat');
```

Misol:

```
sessionStorage.setItem('ism', 'Ali');
```

Ma’lumotni O’qish

Sessiya saqlashdan ma’lumot olish uchun `sessionStorage.getItem` metodidan foydalanamiz:

```
let ism = sessionStorage.getItem('ism');
console.log(ism); // "Ali"
```

Ma’lumotni O’chirish

Sessiya saqlashdagi ma’lumotni o’chirish uchun `sessionStorage.removeItem` metodidan foydalanamiz:

```
sessionStorage.removeItem('ism');
```

Barcha Ma’lumotlarni Tozalash

Sessiya saqlashdagi barcha ma’lumotlarni tozalash uchun `sessionStorage.clear` metodidan foydalanamiz:

```
sessionStorage.clear();
```

Amaliy Misol

Keling, oddiy misolni ko’rib chiqamiz, bunda foydalanuvchi ismini mahalliy saqlashga yozamiz va uni qayta o’qib chiqamiz.

```
<!DOCTYPE html>
<html lang="uz">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Mahalliy Saqlash Misoli</title>
</head>
<body>
    <h1>Mahalliy Saqlash Misoli</h1>
    <input type="text" id="ism" placeholder="Ismingizni
        kiriting">
    <button onclick="saqlash()">Saqlash</button>
    <button onclick="o'qish()">O'qish</button>
    <p id="natija"></p>

    <script>
        function saqlash() {
            let ism = document.getElementById('ism').value;
            localStorage.setItem('ism', ism);
            alert('Ismingiz saqlandi!');
        }

        function o'qish() {
            let ism = localStorage.getItem('ism');
            if (ism) {
                document.getElementById('natija').innerText =
        'Saqlangan ism: ' + ism;
            } else {
                document.getElementById('natija').innerText =
        'Ism topilmadi';
            }
        }
    </script>
</body>
</html>
```

Xulosa

Ushbu bo’limda siz JavaScriptda mahalliy va sessiya saqlash bilan tanishdingiz. Bu usullar foydalanuvchi ma’lumotlarini brauzerda saqlash va qayta foydalanish uchun qulay imkoniyatlar yaratadi. Mahalliy saqlash ma’lumotlarni uzoq muddatga saqlash uchun, sessiya saqlash esa qisqa muddatli saqlash uchun ishlatiladi.

MediaStream API

MediaStream API foydalanuvchining kamera va mikrofon kabi media qurilmalaridan audio va video ma’lumotlarni olish imkonini beradi. Bu API asosan real vaqt rejimida media ma’lumotlarini olish uchun ishlatiladi.

`getUserMedia` Metodi

`getUserMedia` metodi foydalanuvchi qurilmasidan audio va video ma’lumotlarini olish uchun ishlatiladi. Bu metod promises asosida ishlaydi va muvaffaqiyatli bo’lsa `MediaStream` ob’ektini qaytaradi.

Kamera va Mikrofondan Media Olish

```
navigator.mediaDevices.getUserMedia({ video: true, audio: true })
  .then(stream => {
    let videoElement = document.getElementById("video");
    videoElement.srcObject = stream;
    videoElement.play();
  })
  .catch(err => {
    console.error("Media qurilmalarga ulanishda xato yuz berdi:",
        err);
  });
```

HTML:

```
<video id="video" width="600" controls></video>
```

Faqat Videoni Olish

Agar faqat videoni olish kerak bo’lsa, `getUserMedia` metodiga faqat video parametrini jo’natish kifoya.

Faqat Videoni Olish Misoli

```
navigator.mediaDevices.getUserMedia({ video: true })
  .then(stream => {
    let videoElement = document.getElementById("video");
    videoElement.srcObject = stream;
    videoElement.play();
  })
  .catch(err => {
    console.error("Videoni olishda xato yuz berdi:", err);
  });
```

HTML:

```
<video id="video" width="600" controls></video>
```

Faqat Audioni Olish

Agar faqat audioni olish kerak bo'lsa, `getUserMedia` metodiga faqat audio parametrini jo'natish kifoya.

Faqat Audioni Olish Misoli

```
navigator.mediaDevices.getUserMedia({ audio: true })
  .then(stream => {
    let audioElement = document.getElementById("audio");
    audioElement.srcObject = stream;
    audioElement.play();
  })
  .catch(err => {
    console.error("Audioni olishda xato yuz berdi:", err);
  });
```

HTML:

```
<audio id="audio" controls></audio>
```

MediaStream-ni To'xtatish

MediaStream-ni to'xtatish uchun `MediaStream` ob'ektidagi `getTracks` metodidan foydalanib, barcha treklarni to'xtatish kerak.

MediaStream-ni To'xtatish Misoli

```
navigator.mediaDevices.getUserMedia({ video: true, audio: true })
  .then(stream => {
    let videoElement = document.getElementById("video");
    videoElement.srcObject = stream;
    videoElement.play();

    // To'xtatish tugmasi
    document.getElementById("stopButton").addEventListener("click",
        () => {
      stream.getTracks().forEach(track => track.stop());
      videoElement.srcObject = null;
    });
  })
  .catch(err => {
```

```
    console.error("Media qurilmalarga ulanishda xato yuz berdi:",
        err);
  });
```

HTML:

```
<video id="video" width="600" controls></video>
<button id="stopButton">To'xtatish</button>
```

Amaliy Misollar

Kameradan Surat Olish

Kameradan surat olish uchun video stream-dan frame ni canvas elementiga chizib olish mumkin.

```
navigator.mediaDevices.getUserMedia({ video: true })
  .then(stream => {
    let videoElement = document.getElementById("video");
    let canvasElement = document.getElementById("canvas");
    let context = canvasElement.getContext("2d");

    videoElement.srcObject = stream;
    videoElement.play();

    document.getElementById("captureButton").addEventListener("click",
        () => {
      context.drawImage(videoElement, 0, 0, canvasElement.width,
        canvasElement.height);
    });
  })
  .catch(err => {
    console.error("Videoni olishda xato yuz berdi:", err);
  });
```

HTML:

```
<video id="video" width="600" controls></video>
<button id="captureButton">Surat olish</button>
<canvas id="canvas" width="600" height="400"></canvas>
```

Mikrofondan Audio Yozib Olish

Mikrofondan audio yozib olish uchun `MediaRecorder` API dan foydalanish mumkin.

```
let mediaRecorder;
let audioChunks = [];

navigator.mediaDevices.getUserMedia({ audio: true })
  .then(stream => {
    mediaRecorder = new MediaRecorder(stream);

    mediaRecorder.ondataavailable = event => {
      audioChunks.push(event.data);
    };

    mediaRecorder.onstop = () => {
      let audioBlob = new Blob(audioChunks, { type: 'audio/
        wav' });
      let audioUrl = URL.createObjectURL(audioBlob);
      let audio = new Audio(audioUrl);
      audio.play();
    };

    document.getElementById("startRecording").addEventListener("click",
        () => {
      audioChunks = [];
      mediaRecorder.start();
    });

    document.getElementById("stopRecording").addEventListener("click",
        () => {
      mediaRecorder.stop();
    });
  })
  .catch(err => {
    console.error("Audioni olishda xato yuz berdi:", err);
  });
```

HTML:

```
<button id="startRecording">Yozishni boshlash</button>
<button id="stopRecording">Yozishni to'xtatish</button>
```

Mashqlar

1. **Mashq**: Kamera va mikrofon ma’lumotlarini olish uchun HTML va JavaScript kodini yozing.
    - Video va audio olish uchun input tugmalari yarating.
    - Video va audio ma’lumotlarini oynada ko’rsating.

Yechim:

```
<video id="video" width="600" controls></video>
<button id="startMedia">Media olish</button>
<button id="stopMedia">Media to'xtatish</button>

<script>
let mediaStream;

document.getElementById("startMedia").addEventListener("click",
        () => {
  navigator.mediaDevices.getUserMedia({ video: true, audio:
        true })
    .then(stream => {
      mediaStream = stream;
      let videoElement = document.getElementById("video");
      videoElement.srcObject = stream;
      videoElement.play();
    })
    .catch(err => {
      console.error("Media qurilmalarga ulanishda xato yuz
        berdi:", err);
    });
});

document.getElementById("stopMedia").addEventListener("click",
        () => {
  if (mediaStream) {
    mediaStream.getTracks().forEach(track => track.stop());
    document.getElementById("video").srcObject = null;
  }
});
</script>
```

2. **Mashq**: Kameradan surat olish uchun tugma va canvas yarating.
    - Kameradan video stream olib, canvas ga surat chizing.

Yechim:

```
<video id="video" width="600" controls></video>
<button id="captureButton">Surat olish</button>
<canvas id="canvas" width="600" height="400"></canvas>

<script>
navigator.mediaDevices.getUserMedia({ video: true })
  .then(stream => {
```

```
        let videoElement = document.getElementById("video");
        let canvasElement = document.getElementById("canvas");
        let context = canvasElement.getContext("2d");

        videoElement.srcObject = stream;
        videoElement.play();

        document.getElementById("captureButton").addEventListener("click",
            () => {
          context.drawImage(videoElement, 0, 0,
            canvasElement.width, canvasElement.height);
        });
      })
      .catch(err => {
        console.error("Videoni olishda xato yuz berdi:", err);
      });
    </script>
```

Bu darsda MediaStream API yordamida JavaScript-da kamera va mikrofon kabi media qurilmalaridan qanday foydalanish mumkinligi haqida batafsil ma’lumot berildi.

Web Speech API

Web Speech API foydalanuvchiga nutqni tanib olish va sintez qilish imkonini beradi. Bu API asosan ikki qismdan iborat: `SpeechRecognition` (nutqni tanib olish) va `SpeechSynthesis` (nutqni sintez qilish).

Nutqni Tanib Olish (`SpeechRecognition`)

`SpeechRecognition` interfeysi foydalanuvchining nutqini matnga aylantirish imkonini beradi. Bu interfeys promises asosida ishlaydi va `result` tadbirida tan olingan nutqni qaytaradi.

Nutqni Tanib Olish Misoli

```
// SpeechRecognition interfeysini yaratish
let recognition = new (window.SpeechRecognition ||
        window.webkitSpeechRecognition)();

// Tilni o'rnatish
recognition.lang = 'uz-UZ';

// Uzoq nutqlarni tanib olish
recognition.continuous = true;

// Tanib olish jarayonida ishlash
recognition.onresult = (event) => {
  let result = event.results[event.results.length - 1]
        [0].transcript;
  document.getElementById("recognizedText").innerText = result;
};

// Xatolarni ushlash
recognition.onerror = (event) => {
  console.error("Nutqni tanib olishda xato yuz berdi:",
        event.error);
};
```

HTML:

```
<div id="recognizedText">Bu yerda tan olingan matn ko'rsatiladi</
        div>
<button id="startRecognition">Boshlash</button>
<button id="stopRecognition">To'xtatish</button>
```

```
<script>
document.getElementById("startRecognition").addEventListener("click",
        () => {
  recognition.start();
});

document.getElementById("stopRecognition").addEventListener("click",
        () => {
  recognition.stop();
});
</script>
```

Nutqni Sintez Qilish (SpeechSynthesis)

SpeechSynthesis interfeysi matnni ovozli nutqqa aylantirish imkonini beradi. Bu interfeys yordamida turli tillarda va ovozlarda matnni ovoz chiqarish mumkin.

Nutqni Sintez Qilish Misoli

```
// SpeechSynthesis interfeysini yaratish
let synth = window.speechSynthesis;

// Matnni ovozli nutqqa aylantirish funksiyasi
function speak(text) {
  if (synth.speaking) {
    console.error("Hozirda sintez qilinmoqda...");
    return;
  }
  let utterance = new SpeechSynthesisUtterance(text);
  utterance.lang = 'uz-UZ';
  synth.speak(utterance);
}

// Tugmani bosganda matnni ovozli nutqqa aylantirish
document.getElementById("speakButton").addEventListener("click",
        () => {
  let text = document.getElementById("textToSpeak").value;
  speak(text);
});
```

HTML:

```
<textarea id="textToSpeak" placeholder="Bu yerga matn
        kiriting"></textarea>
<button id="speakButton">Ovoz chiqarish</button>
```

Amaliy Misollar

Nutqni Tanib Olish va Sintez Qilish

Foydalanuvchi nutqini tanib olish va keyin uni qaytarib ovoz chiqarish misoli.

```
// Nutqni tanib olish interfeysini yaratish
let recognition = new (window.SpeechRecognition ||
        window.webkitSpeechRecognition)();
recognition.lang = 'uz-UZ';
recognition.continuous = true;

recognition.onresult = (event) => {
  let result = event.results[event.results.length - 1]
        [0].transcript;
  document.getElementById("recognizedText").innerText = result;
  speak(result);
};

recognition.onerror = (event) => {
  console.error("Nutqni tanib olishda xato yuz berdi:",
        event.error);
};

document.getElementById("startRecognition").addEventListener("click",
        () => {
  recognition.start();
});

document.getElementById("stopRecognition").addEventListener("click",
        () => {
  recognition.stop();
});

// Nutqni sintez qilish interfeysini yaratish
let synth = window.speechSynthesis;

function speak(text) {
  if (synth.speaking) {
    console.error("Hozirda sintez qilinmoqda...");
    return;
  }
  let utterance = new SpeechSynthesisUtterance(text);
  utterance.lang = 'uz-UZ';
```

```
  synth.speak(utterance);
}
```

HTML:

```
<div id="recognizedText">Bu yerda tan olingan matn ko'rsatiladi</
        div>
<button id="startRecognition">Boshlash</button>
<button id="stopRecognition">To'xtatish</button>
```

Mashqlar

1. **Mashq**: Foydalanuvchi nutqini tanib olish va tan olingan matnni sahifaga chiqarish.

    - Nutqni tanib olishni boshlash va to’xtatish tugmalari yarating.
    - Tan olingan matnni sahifada ko’rsating.

    Yechim:

    ```
    <div id="recognizedText">Bu yerda tan olingan matn
            ko'rsatiladi</div>
    <button id="startRecognition">Nutqni tanib olishni boshlash</
            button>
    <button id="stopRecognition">Nutqni tanib olishni
            to'xtatish</button>

    <script>
    let recognition = new (window.SpeechRecognition ||
            window.webkitSpeechRecognition)();
    recognition.lang = 'uz-UZ';
    recognition.continuous = true;

    recognition.onresult = (event) => {
      let result = event.results[event.results.length - 1]
            [0].transcript;
      document.getElementById("recognizedText").innerText =
            result;
    };

    recognition.onerror = (event) => {
      console.error("Nutqni tanib olishda xato yuz berdi:",
            event.error);
    };

    document.getElementById("startRecognition").addEventListener("click",
            () => {
      recognition.start();
    ```

```
});

document.getElementById("stopRecognition").addEventListener("click",
        () => {
  recognition.stop();
});
</script>
```

2. **Mashq**: Foydalanuvchi kiritgan matnni ovozli nutqqa aylantirish.
   - Matn kiritish uchun textarea yarating.
   - Ovoz chiqarish tugmasi yarating.
   - Tugma bosilganda matnni ovozli nutqqa aylantiring.

   Yechim:

```
<textarea id="textToSpeak" placeholder="Bu yerga matn
        kiriting"></textarea>
<button id="speakButton">Ovoz chiqarish</button>

<script>
let synth = window.speechSynthesis;

function speak(text) {
  if (synth.speaking) {
    console.error("Hozirda sintez qilinmoqda...");
    return;
  }
  let utterance = new SpeechSynthesisUtterance(text);
  utterance.lang = 'en-US';
  synth.speak(utterance);
}

document.getElementById("speakButton").addEventListener("click",
        () => {
  let text = document.getElementById("textToSpeak").value;
  speak(text);
});
</script>
```

Bu darsda Web Speech API yordamida nutqni tanib olish va sintez qilish haqida batafsil ma’lumot berildi.