**1. JavaScript nədir?**

JavaScript (JS) veb səhifələri interaktiv hala gətirmək ücün istifadə olunan proqramlaşdırma dilidir. HTML səhifənin qurulması, CSS dizaynı təmin etdiyi kimi, JavaScript funksionallıq və dinamika qatır. Bu dildə yazılan kodlar brauzerdə birbaşa işləyir, istifadəçi əməllərinə reaksiya verir, məzmunu dəyişdirməyə imkan yaradir.

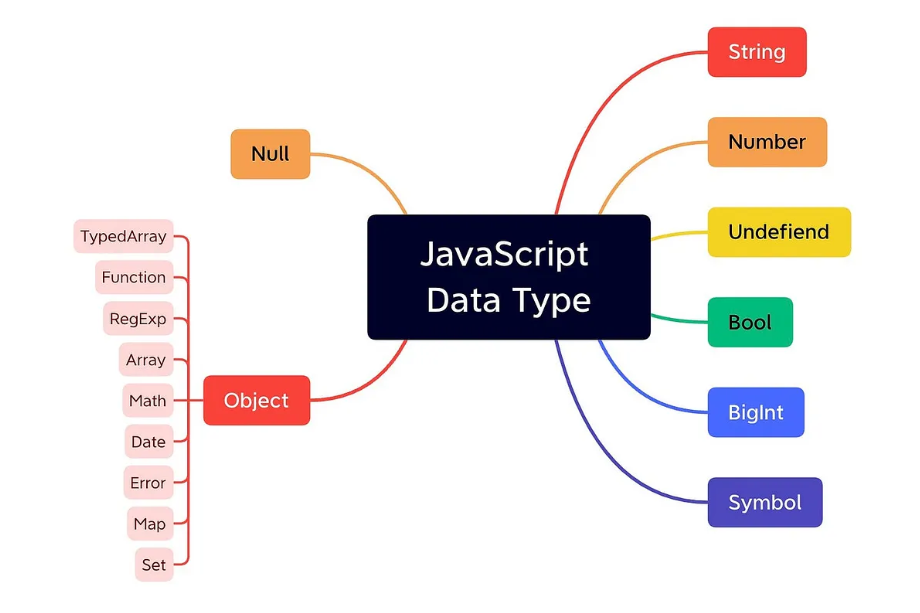
1995-ci ildə Brendan Eich tərəfindən yaradılan JavaScript zamanla inkiaf edərək, sadə skript dilindən veb proqramlaşdırmanın əsas texnologiyasına çevrildi. Əvvəllər tək brauzer üçün yaradılsa da, bu gün Node.js vasitəsiylə server tərəfdə də istifadə olunur.

**JavaScript-in əsas xüsusiyyətləri:**

* **Dinamik tiplənmə**: Dəyişkənlərin tipi kod yazıldıqca avtomatik təyin olunur.
* **Obyekt yönülmüş yanaşma**: Obyektlər vasitəsilə verilərin saxlanması və istifadəsi rahat təşkiledilir.
* **Hadisəyə reaksiya**: Klik, form göndərmə və ya klaviatura daxil etmə kimi əməllərə cavab verir.
* **Asinxron proqramlaşdırma**: Promise və async/await kimi strukturla eyni anda bir neçə əməliyyat işlənə bilir.
* **Çox-tərəfli istifadə**: Hem brauzer, hem də server tərəfdə istifadə edilə bilir.

JavaScript bu gün sadə bir skript dili deyil, tam funksional proqramlaşdırma imkanları təklif edir. Web tərəqqisinin ayrılmaz hissəsidir.

### ****2. JavaScript-də Data Typelar****



JavaScript dinamik tiplənməli dildir, yəni dəyişkən elan edərkən onun tipini açıq şəkildə yazmaq vacib deyil. Dəyərə görə avtomatik olaraq tip təyin olunur.

**ƏSAS DATA TYPELAR:**

1. **String** – Mətn tipli dəyərlər.

let ad = "Leyla";

1. **Number** – Rəqəmlər (tam və ya onluq).

let yas = 25;

let maas = 1250.50;

1. **Boolean** – Doğru və ya yalan (true / false).

let aktiv = true;

1. **Undefined** – Təyin olunmamış dəyər.

let test;

console.log(test); // undefined

1. **Null** –"boş" dəyər.

let deyer = null;

1. **Object** – Mürəkkəb tiplər (array, funksiyalar daxil olmaqla).

let user = {

ad: "Leyla",

yas: 25

};

1. **Symbol** – Unikal identifikator (müqayisə olunmayan dəyər).
2. **BigInt** – Çox böyük rəqəmləri saxlamaq ücün.

let boyukEded = 123456789012345678901234567890n;

**typeof Operatoru**

Bir dəyişkənin tipini yoxlamaq ücün typeof operatorundan istifadə olunur:

let x = "Salam";

console.log(typeof x); // "string"

Bu hissə, JavaScript-in təməl anlayışı olan tiplərin qavranılması ücün vacibdir. Proqramda dəyərlə işlərərkən hansı tipdə olduqlarını anlamaq əsas bacarıqlardandır.

**3. JavaScript-də let, const, var fərqləri**

JavaScript-də dəyişkən yaratmaq ücün let, const və var açar sözlərindən istifadə olunur. Bunlar arasında istifadə qaydası və davranış baxımından fərqlər var.

**let**

let dəyişkənlər blok scope-ludur. Yəni, ancaq təyin olunduğu blok daxilində əlçatan olur.

let ad = "Leyla";

ad = "Aysel"; // dəyər dəyişir

* Yenidən təyin oluna bilməz (eyni adlı let eyni blokda).
* Amma dəyəri sonradan dəyişə bilir.

**const**

const sabit dəyər yaratmaq ücün istifadə olunur. Təyin olunduqdan sonra dəyərini dəyişmək olmaz.

const PI = 3.14;

// PI = 3.1415; // xəta verir

* Blok scope-ludur.
* Referens dəyərlər (obyekt, array) təyin olunarsa, daxilindəki dəyərlər dəyişə bilər, amma obyektin özü yox:
* const arr = [1, 2];
* arr.push(3); // OK

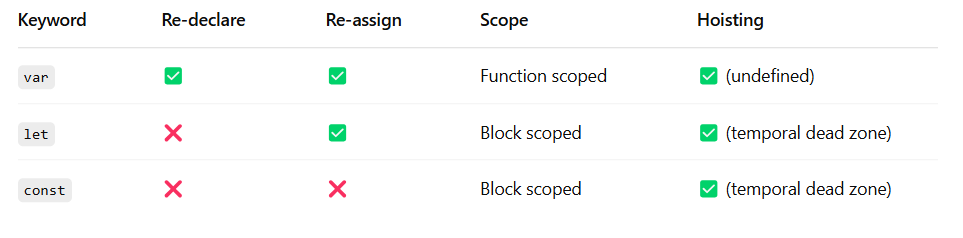
arr = [4, 5]; // Xəta

**var**

var daha əski yanaşmadır. Global scope və ya function scope-da əlçatandır, blok scope-u tanımaz.

var ad = "Leyla";

* Funksiya daxilində elan olunubsa, orda qalır.
* Blok daxilində elan olunsa da, kənarda da istifadə oluna bilir.
* Hoistingə məruz qalır (sonra elan olunub əvvəl istifadə edilsə də xəta vermir).

**Nə vaxt hansından istifadə etmək olar?**

* Əksər hallarda let istifadə edilir.
* Dəyərin sabit qalması şərtlidirsə, const.
* var daha çox köhnə kodlarda rast gəlinir, tərəfdarı çox deyil.

Bu hissə JavaScript-də dəyişkənlərlə işləyərkən hansı açar sözdən istifadə etməyin daha düzgün olduğunu anlamağı əsas verir.

### ****4. JavaScript-də Operatorlar****

JavaScript operatorlar vasitəsilə hesablamalar aparır, dəyərləri mri m\u00fqcayisə edir və ya mantıqı əməllərlə işlər yerinə yetirilir. Operatorlar kodda çox vaxt istifadə olunan əsas hissədir.

#### ****1. Arithmetic (Riyazi) Operatorlar****

Bu operatorlar riyazi əməliyyatlar aparmaq ücün istifadə olunur:

+ toplama

- çıxma

\* vurma

/ bölmə

% qaliq

\*\* quvvet (2 \*\* 3 = 8)

++ 1 artırma

-- 1 azaltma

#### ****2. Assignment (Təyin etmə) Operatorları****

= dəyər təyin edir

+= üstəgəl təyin

-= çıx təyin

\*= vur təyin

/= böl təyin

%= qaliq təyin

#### ****3. Comparison (Müqayisə) Operatorları****

== bərabərdir (tipi yoxlamadan)

=== bərabərdir (tipi də yoxlayır)

!= bərabər deyil

!== tipi daxil bərabər deyil

> böyük

< kiçik

>= böyük bərabər

<= kiçik bərabər

#### ****4. Logical (Mantıqı) Operatorlar****

&& və (and)

|| və ya (or)

! deyil (not)

#### ****5. String Concatenation (Birleşdirmə)****

let ad = "Leyla";

let soyad = "Rüstəmova ";

let tamAd = ad + " " + soyad;

#### ****6. Ternary Operator****

Qısa if-else yazmaq ücün istifadə olunur:

let yas = 18;

let mesaj = (yas >= 18) ? "Yetkin" : "Uşaq";

Bu operatorlar sayəsində proqram daxilində riyazi, müqayisə, məntıqı hesablamalar yerinə yetirilir. Onlar proqramın əsas işləyən mexanizmini təşkil edir.

### ****5. JavaScript-də Adlandırma Qaydaları (Naming Conventions)****

Dəyişkən, funksiya, obyekt və ya sinif (class) yaradarkən adların düzgün, oxunaqlı və məna verici olması proqramı daha anlaşılan və idarə oluna bilən edir. JavaScript-də bu adları təyin edərkən konkret adlandırma stilləri izlənir.

#### ****1. camelCase****

* Kiçik hərf və sonraki sözə başlayan hərflər böyük.
* Funksiya adları, dəyişkənlər və obyekt adları üçün standartdır.

let userName = "Leyla";

function getUserData() {}

#### ****2. PascalCase****

* Hər söz böyük hərfə başlayır.
* Adətən sinif (class) və konstruktor adlarında istifadə olunur.

class UserProfile {}

function CreateUser() {}

#### ****3. snake\_case****

* Sözlər alt xətlə ayrılır, adətən JavaScript-də istifadə olunmur.
* Python və ya backend dillərdə daha çox rast gəlinir.

let user\_name = "Leyla";

#### ****4. kebab-case****

* Hər söz tire (-) ilə ayrılır.
* JavaScript-də istifadə olunmur, HTML/CSS tərəfdə class adlarında görülür.

<div class="user-profile"></div>

#### ****Yaxşı Praktikalar****

* Adlar məna verici olsun: let x əvəzinə let userAge
* Funksiya adları feil formada yazılsın:
* function calculateSalary() {}

function sendMessage() {}

* Sinif adları büyük hərflə başlamalıdır:

class ProductList {}

* Qısa, açıqlayıcı və nəqli adlar verin.

Adlandırma proqram kodunun oxunaqlı olmasına birbaşa təsir edir. Təmiz, sistemli kod yazmaq istəyən proqramçı bu qaydalara diqqət etməlidir.

### ****6. JavaScript-də İf/Else, Switch Case, Ternary Operatoru****

Proqram daxilində şərtlərə görə işlər aparmaq ücün JavaScript şərt operatorlarından istifadə olunur.

#### ****if / else****

let yas = 20;

if (yas >= 18) {

console.log("Yetkin yaşdasın");

} else {

console.log("Hələ uşaq sayılırsan");

}

#### ****else if****

Bir neçə şərt olduqda:

let bal = 85;

if (bal >= 90) {

console.log("A");

} else if (bal >= 80) {

console.log("B");

} else {

console.log("Digər");

}

#### ****switch / case****

Bir dəyər bir neçə haldan birinə uyğundursa:

let gun = "Cümə";

switch (gun) {

case "Bazarertəsi":

console.log("Həftənin ilk günü");

break;

case "Cümə":

console.log("Demək istirahət yaxındadı!");

break;

default:

console.log("Adi bir güdür");

}

#### ****Ternary Operatoru****

Qısa şərt yazmaq ücün:

let daxilOlub = true;

let mesaj = daxilOlub ? "Xoş gəldin!" : "Zəhmət olmasa daxil olun";

Bu operatorlar kodun axışını idarə etmək ücün əsas vasitələrdir. Şərtlər sayəsində proqram istifadəçinin davranışına reaksiya verir, dinamik hala gəlir.

### ****7. JavaScript-də Dövrlər: for, while, do while, break, continue****

Kodun bir hissəsini təkrar işlətmək istədikdə JavaScript bizə dövrlər (loops) təklif edir. Bu dövrlər kodu bir əməliyyatdan daha çox təkrar işləməyə imkan verir.

#### ****1. for dövrü****

Əvvəlcə dəyər təyin edilir, sonra şərt yoxlanır və hər iterasiyadan sonra sayğac artırılır:

for (let i = 0; i < 5; i++) {

console.log("Salam " + i);

}

Nəticə: Salam 0, Salam 1, ..., Salam 4

#### ****2. while dövrü****

Şərt düzgün olduqca kod icra olunur:

let i = 0;

while (i < 3) {

console.log(i);

i++;

}

#### ****3. do...while dövrü****

Kod əvvəlcə bir dəfə icra olunur, sonra şərt yoxlanır:

let i = 0;

do {

console.log("Say: " + i);

i++;

} while (i < 3);

#### ****4. break operatoru****

Dövrü vaxtından əvvəl dayandırmaq ücün:

for (let i = 0; i < 10; i++) {

if (i === 5) break;

console.log(i);

}

#### ****5. continue operatoru****

Dövrü dayandırmadan sadəcə cari iterasiyanı atlır:

for (let i = 0; i < 5; i++) {

if (i === 2) continue;

console.log(i);

}

Bu kodda 2 yazılmayacaq.

#### ****Əlavə Misal: Array elementlərinin toplu göstərilməsi****

let adlar = ["Leyla", "Rəvan", "Elvin"];

for (let i = 0; i < adlar.length; i++) {

console.log("Salam, " + adlar[i]);

}

### ****8. JavaScript-də Array-lar (Massivlər)****

Array – verilənlərin siyahı şəklində saxlanması ücün istifadə olunur. Birdən çox dəyərin eyni struktur daxilində toplanması üçün idealdır.

let meyveler = ["alma", "banan", "nar"];

#### ****Array-a daxil olmaq****

Array-da dəyərlər indekslərlə idarə olunur. Sayım 0-dan başlayır:

console.log(meyveler[0]); // alma

#### ****length – Uzunluq****

Array-da neçə element olduğunu bildirir:

console.log(meyveler.length); // 3

#### ****push – Yeni element əlavə etmək****

meyveler.push("armud");

console.log(meyveler); // ["alma", "banan", "nar", "armud"]

#### ****pop – Son elementi silmək****

meyveler.pop();

console.log(meyveler); // ["alma", "banan", "nar"]

#### ****shift – ilk elementi silmək****

meyveler.shift();

console.log(meyveler); // ["banan", "nar"]

#### ****unshift – ilkə yeni element əlavə etmək****

meyveler.unshift("kivi");

console.log(meyveler); // ["kivi", "banan", "nar"]

#### ****forEach – Hər bir elementi işləmək****

meyveler.forEach(function(meyve) {

console.log("Meyvə: " + meyve);

});

#### ****includes – Dəyər axtarmaq****

console.log(meyveler.includes("banan")); // true

#### ****indexOf – Dəyərin indeksini tapmaq****

console.log(meyveler.indexOf("nar")); // 2

#### ****splice – Element silmək/əlavə etmək****

meyveler.splice(1, 1); // 1-ci indeksdən 1 elementi sil

console.log(meyveler);

#### ****join – Array-ı stringə çevirmək****

let cumle = meyveler.join(", ");

console.log(cumle); // "kivi, nar"

Massivlər veb tətbiqlərdə verilərin siyahı şəklində saxlanması, işlənməsi və göstərilməsi ücün ən güclü vasitələrdən biridir.

### ****9. JavaScript-də Obyektlər (Objects)****

JavaScript-də obyektlər, verilənləri qruplaşdırmaq və təşkil etmək üçün istifadə olunur. Obyektlər həmçinin, real dünyadakı obyektləri təmsil etməyə kömək edir. Məsələn, bir insanın adı, yaşı və peşəsi kimi məlumatları obyekt daxilində saxlayaraq bir arada qruplaşdıra bilərik.

#### ****Obyekt yazmaq****

Obyektlər {} ilə yaradılır və daxilində açar-dəyər cütləri olur. Açar və dəyərlər arasında : işarəsi istifadə olunur, cütlər arasında isə vergül , ilə ayrılır.

js

CopyEdit

let insan = {

ad: "Leyla",

yas: 25,

meslek: "frontend developer"

};

Bu misalda, insan adlı obyektin 3 açarı var: ad, yas, və meslek. Bu açarların hər biri müəyyən dəyərlərə malikdir.

#### ****Obyektə daxil olmaq****

Obyektin dəyərlərinə daxil olmaq üçün açar adı istifadə olunur:

js

CopyEdit

console.log(insan.ad); // Leyla

console.log(insan["yas"]); // 25

#### ****Obyektə yeni dəyər əlavə etmək****

Obyektə yeni dəyər əlavə etmək üçün birbaşa açar adı ilə təyin etmə edilir:

js

CopyEdit

insan.telefon = "0551234567";

console.log(insan);

// {ad: "Leyla", yas: 25, meslek: "frontend developer", telefon: "0551234567"}

#### ****Obyektin metodları****

Obyektlər həm də metodları (funksiyaları) saxlaya bilir. Məsələn:

js

CopyEdit

let adam = {

ad: "Rəvan",

yas: 30,

salamlama: function() {

console.log("Salam, mənim adım " + this.ad);

}

};

adam.salamlama(); // Salam, mənim adım Rəvan

Obyektlər verilənləri və funksiyaları birləşdirmək üçün istifadə olunur və veb tərtibatında çox geniş şəkildə istifadə edilir.

### ****10. JavaScript-də Funksiyalar (Functions)****

JavaScript-də funksiyalar müəyyən bir əməliyyatı yerinə yetirmək üçün yazılmış kod bloklarıdır. Funksiyalar həm sadə, həm də mürəkkəb əməliyyatları həyata keçirmək üçün istifadə edilə bilər.

#### ****Funksiya bəyanatı (Function Declaration)****

Funksiya bəyanatı funksiyanı əvvəldən bəyan edib sonra çağırmağa imkan verir. Misal:

js

CopyEdit

function topla(a, b) {

return a + b;

}

console.log(topla(3, 5)); // 8

#### ****Funksiya ifadəsi (Function Expression)****

Funksiya ifadəsi anonim (adı olmayan) funksiyaları yaratmağa imkan verir. Bu funksiyalar bir dəyişkənə təyin edilir:

js

CopyEdit

let vur = function(a, b) {

return a \* b;

};

console.log(vur(3, 5)); // 15

#### ****Arrow Funksiyaları (Arrow Functions)****

Arrow funksiyaları qısa yazılış formasıdır. => işarəsi ilə yazılır və funksiyanın bədənini daha sadə edir:

js

CopyEdit

let bol = (a, b) => a / b;

console.log(bol(10, 2)); // 5

### ****11. Template Literals (Şablon Sətirləri)****

Template literals (şablon sətirləri) JavaScript-də string-ləri daha oxunaqlı yazmaq üçün istifadə olunur. Bu zaman dəyişkənlər və ifadələr birbaşa string-lər daxilində yazıla bilir.

js

CopyEdit

let ad = "Leyla";

let yas = 25;

let mesaj = `Mənim adım ${ad} və mən ${yas} yaşındayam.`;

console.log(mesaj); // Mənim adım Leyla və mən 25 yaşındayam.

Şablon sətirləri, həmçinin çoxsətirli stringlər üçün də faydalıdır:

js

CopyEdit

let uzunMətn = `Bu,

bir neçə sətirdən ibarət

olan bir mətnin nümunəsidir.`;

console.log(uzunMətn);

### ****12. JavaScript-də Hoisting nədir?****

Hoisting, JavaScript-də dəyişkənlər və funksiyaların təriflərinin, istifadə olunduğu yerdən əvvəl yerinə yetirilməsi hadisəsidir. Yəni, dəyişkənlərin və funksiyaların bəyanatları avtomatik olaraq "yuxarıya çəkilir" (hoisted) və beləliklə, onları istifadə etməzdən əvvəl, yəni tərifdən əvvəl belə istifadə edə bilərsiniz.

#### ****Dəyişkənlərdə Hoisting****

JavaScript-də var ilə yaradılan dəyişkənlər hoisting edilir, amma onların qiymətləri yalnız tərif olunduğu yerdə müəyyənləşir. Bu, kodu qarışdıran vəziyyətlərə gətirib çıxara bilər:

js

CopyEdit

console.log(a); // undefined

var a = 5;

console.log(a); // 5

Amma let və const ilə yaradılan dəyişkənlər hoisting edilmir, onlara təyin olunmadan istifadə etmək səhv verir:

js

CopyEdit

console.log(b); // ReferenceError: Cannot access 'b' before initialization

let b = 10;

#### ****Funksiya Hoisting****

Funksiyalar da hoisted edilir, amma yalnız funksiya bəyanatı ilə yazıldıqda:

js

CopyEdit

topla(2, 3); // 5

function topla(a, b) {

return a + b;

}

Funksiya ifadələri isə hoisting edilmir:

js

CopyEdit

topla(2, 3); // TypeError: topla is not a function

var topla = function(a, b) {

return a + b;

};