Mündəricat

1. JavaScript nədir?

2. JavaScript-də datatypes

3. JavaScript-də let, const, var fərqi

4. JavaScript-də operatorlar:

Arithmetic, Assignment, Comparsion, Logical və s.

5. JavaScript-də adlandırma qaydaları

6. JavaScript-də if else; switch case (ternary)

7. JavaScript-də do while; while; break continue

8. JavaScript-də array, length, push və s.

9. JavaScript-də object

10. JavaScript-də funksiyalar (arrow, decleration, expression)

11. String, templete, literal nümunələr

12. String metodları

13. Hoisting nədir?

**JavaScript nədir?**

**JavaScript** – web səhifələri, tətbiqləri və oyunları inkişaf etdirərkən daha dinamik qarşılıqlı əlaqə yaratmaq üçün istifadə edilən obyektyönümlü **(OOP-**Object-Oriented Programming) proqramlaşdırma dilidir. O, 1995-ci ildə Netscape işçisi Brendan Eich tərəfindən yaradılıb. O ilk olaraq Mocha daha sonra Mona, LiveScript kimi müxtəlif cür adlandırılmışdır. JavaScript yarandığı gündən bu günə kimi inkişaf etməyə davam edir. JavaScript vasitəsilə formların düzgünlüyünü yoxlamaq; animasiya və effektlər əlavə etmək; oyunlar, interaktiv xəritələr,kalkulyatorlar və digər funksional elementlər yaratmağa kömək edir.

*Daha sadə izah*:Əgər *HTML* bir web səhifənin skeleti, *CSS* isə onun dizaynıdırsa, *JavaScript* bu skeletə “can” verən ruhdur.

**JavaScript-in əsas xüsusiyyətləri.**

**Asinxron Proqramlaşdırma:** JavaScript bir **single-threaded** proqramlaşdırma dilidir. Yəni sətir-sətir işləyir, bir sətir yerinə yetirilməmiş digərinə keçmir. Buna baxmayaraq **Asinxron əməliyyatlar** sayəsində o bir neçə əməliyyatı paralel olaraq icra edə bilər.

**Çarpaz Platformalı Dəstək:**

JavaScript brauzerlərlə işləməkdən əlavə *Node.js* vasitəsilə serverdə, *Electron.js* vasitəsilə masaüstü proqramlarda və *React Native* ilə mobil tətbiqlərdə də istifadə olunur.

**JavaScript ilə nələr etmək olar?**

**1.İnteraktiv Web Səhifələr Yaratmaq**

* Açılan menyular
* Qalereyalar
* Form yoxlamaları

*Form yoxlamaları dedikdə nə nəzərdə tutulur?*

Onlayn müraciət formasında bizdən istənilən məlumatların düzgün daxil edildiyinin, e-poçtun doğru formatda daxil edilib-edilmədiyinin,şifrə gücünün və s. yoxlanılması

**2. Mobil Tətbiqlər Hazırlamaq**

**React Native** kimi texnologiyalar sayəsində həm **Android** həm də **iOS** üçün tətbiqlər hazırlamaq

olar.

**3. Oyunlar Yaratmaq**

* Canvas API ilə 2D oyunlar
* Three.js ilə 3D qrafikalı oyunlar

**4.Masaüstü Tətbiqlər**

**Electron.js** vasitəsilə həm **Windows**, həm **macOS**, həm də **Linux** üçün proqramlar hazırlamaq olar. Nümunə: **Visual Studio Code.**

**5.Server-Tərəfli Proqramlaşdırma (Node.js)**

* API-lər yaratmaq
* Məlumat bazası ilə işləmək
* Real-time (canlı) tətbiqlər (məsələn, chat proqramları) qurmaq mümkündür.

Sürətli və əlverişli proqramlaşdırma dili olan JavaScriptin

müsbət cəhətləri olduğu kimi mənfi cəhətləri də var. Təhlükəsizlik problemləri, tiplərin dinamik olması, böyük layihələrdə idarəetmənin çətinləşməsi və s.

**JavaScript-də datatypelar**

Javascript-də data tiplər **Primitive** və **Non-primitive** olmaqla 2 yerə bölünür**. Primitive data tipləri** sadə və dəyişməzdir (immutable). **Non-primitive data tipləri (Reference Types)** obyektlərə əsaslanır və mürəkkəb verilənləri ifadə edir. Obyektlər dəyişkəndir və onlara istinadla (reference) müraciət edilir.

**Primitive datatypes**

1. **Number**

Bütün növ rəqəmləri ifadə edir.

let num = 3

1. **String**

Mətni ifadə etmək üçün istifadə edilir. Tək, ikiqat (“ ”), backtick (`) işarələri ilə yazılır.

let str = “salam”

1. **Boolean**

2 mümkün dəyərə sahib ola bilər:*true*(doğru) və *false*(yanlış)

1. **Undefined**

Dəyişən təyin edildikdə,lakin ona dəyər verilmədikdə istifadə olunur.

1. **Null**

Boş və ya mövcud olmayan dəyəri ifadə etmək üçün istifadə olunur. Undefined-dən fərqi odur ki bunu istifadəçi özü verir.

Let person = null;

Console.log(person); null qaytaracaq

1. **Symbol**

Unikal və dəyişdirilə bilməz dəyərdir. O, əsasən obyektlərin xassələrini müəyyən etmək üçün istifadə olunur.

1. **BigInt**

Çox böyük tam ədədlərlə işləmək üçün istifadə olunur.

**Non-primitive datatypes**

1. **Object**

Key-value cütlərini saxlamaq üçün istifadə olunur. Açarlar yalnız *string* və *symbol* ola bilər.

*let person = {name: ”Arzu”, age: 11};*

Burada *name* və *age* açarlardır (key). “Arzu” və 11 isə uyğun olaraq onlara verilmiş dəyərdir.

1. **Array**

Dəyərlərin sıralı bir kolleksiyasıdır. İndeks və dəyərlərdən ibarətdir.

*let arr = [ “blue” , “green” , “orange” , “red”];*

1. **Function**

Müəyyən tapşırığı yerinə yetirmək üçün təyin edilən kod blokudur.

*Functuion write () {*

*console.log(“Salam”);*

*}*

*Write();*

Funksiyanın adı feil, parametri isim olmalıdır.

1. **Date**

Tarix və vaxtla işləmək üçün istifadə olunur.

1. **Map**

Key-value dəyərlərini saxlamaq üçün istifadə edilən strukturdur. Obyektlərdən fərqli olaraq açarlar hər istənilən tipdə ola bilər.

1. **Set**

Təkrar olunmayan dəyərlər kolleksiyasını saxlamaq üçün istifadə olunur.

*Data tipini yoxlamaq üçün* **typeof** *operatorundan istifadə olunur.*

**JavaScript-də let,const, var fərqi**

JavaScriptdə keywordlar: let, const, var. Bunlar dəyişən (variable) elan etmək üçün istifadə olunan açar sözlərdir. Keywordlar və onların fərqlərindən danışmazdan öncə scopelardan danışaq.

**Scope nədir?**

*Scope*{} ingiliscədən tərcümədə ərazi, sahə mənasını verir, proqramlamada isə yazdığımız kodda dəyişənin görülə və ya ulaşıla bilməsini təmin edir. 2 növ scope var: *Global* və *Local*. *Global scope* funksiyalar xaricində olan sahədir. Burada elan edilmiş bütün dəyişənlər istənilən yerdə access oluna bilər. *Local scope* funksiyalar daxilində olan ərazidir. Hər funksiyanın öz scope-u var. Funksiya daxilində çağırılan dəyişən ancaq həmin əraziydə keçərlidir*. Local scope* 2 cür olur: *Block* və *Function*. Var function scope-dur, let və const isə block scope-dur.

İndi gəlin var,let və const-un fərqlərinə baxaq.

*Var*

*Funksiya* daxilində yazılmış *var* dəyişəni yalnız funksiya daxilində keçərlidir. Hoist olunduqda undefined yəni dəyərsiz olur.

*Let*

*Blok* daxilində yazılmış *let* sadəcə həmin blokun daxilində işləyəcək. Xaricdə yazsaq işləməz. Hoist olunduqda errorverəcək. Let dəyişəni yenilənə bilər, lakin yenidən təyin oluna bilməz.

*Const*

*Blok* daxilində yazılmış const *let* kimi yalnız blok daxilində loga versək işləyər, xaricdə yazsaq işləməyəcək. Hoist olunduqda da eynən let kimi error verəcək. *Let* ilə fərqi odur ki, *const* nə yenilənə, nə də yenidən təyin oluna bilər. Adından da göründüyü kimi sabitdir(constant).

Sonda onu qeyd edim ki, *var*-dan istifadə etməməyimizin səbəbi error-u çıxarmamasıdır.

**Operatorlar**

**Asignment operators**

Ad Operator Mənası

Assignment x = f() x = f()

Addition assignment x += f() x = x + f()

Substraction assignment x -= f() x = x - f()

Multiplication assignment x \*= f() x = x \* f()

Division assignment x /= f() x = x / f()

Remainder assignment x %= f() x = x % f()

Exponentiation assignment x \*\*= f() x = x % f()

**Comparsion operators**

Müqayisə operatorlarının növləri bunlardır: Equal (==) yalnız dəyərin bərabər olduğunu göstərir; Strict equal (===) həm dəyərin, həm də tipininbərabər olduğunu göstərir; Not equal (!=) dəyərin bərabər olmadığını göstərir; Strict not equal (!==) dəyərin və tipinin bərabər olmadığını göstərir; Greater than(>); Greater than or equal (>=); Less than (<); Less than or equal (<=)

**Arithmetic operators**

Addition (+) toplama; Subtraction (-) çıxma; Multiplication (\*) vurma; Division (/) bölmə; Remainder (%) qalığı hesablayır; Increment (++) birini əlavə edir; Decrement (--) birini çıxır; Unary negation (-) məs. x 3dürsə -x -3 qaytaracaq; Unary plus (+) məs. +”3” 3 qaytaracaq yəni rəqəmə çevirir. Əgər +true olarsa 1 false olarsa 0 qaytarar; Exponentiation operator (\*\*) qüvvətə yüksəldir;

**Logical operators**

Logical and (&&) verilmiş şərtin hər ikisi doğrudursa true əks halda false göstərir; Logical or (||) iki şərtdən heç olmasa biri doğru olarsa true əks halda false ; Nullish coalescing operator (??) undefined və ya null olmayanı qaytarır; Logical not (!) nəticəni əksinə çevirir.

**JavaScript-də adlandırma qaydaları**

JavaScriptdə dəyişənlərin adları *rəqəmlə* başlaya bilməz. Onlar *$* , *\_* və *hərflərlə* başlayır. Dəyişən və funksiya adlarında rəqəm də daxil olmaqla (başlanğıcda gəlməmək şərti ilə) yalnız əvvəldə sadaladıqlarım ola bilər. Bəzi sözlər var ki, onlar dəyişən və ya funksiya adı kimi istifadə oluna bilməz. Belə sözlər əvvəlcədən təyin edilmiş açar sözlərdir. Nümunə: *break*, *class*, *for*, *return*, *try*, *let* və s.

**Adlandırma stilləri**

**Camel Case** - ən çox istifadə olunan adlandırma qaydasıdır.

Birinci sözün ilk hərfi kiçik, digər sözlərinki isə böyük olur.

**Pascal Case** – bütün sözlər böyük hərflə başlayır.

**Snake Case** – adlar arasında \_ istifadə olunur və hərflər kiçik olur.

*Adlar mənalı olmalıdır.*

**Şərt blokları**

*İf else* strukturunu belə izah edə bilərik: Əgər a>5 isə onda ekrana a-nı çıxart, *else* hissədə isə əks halda nə olsun onu yazacıyıq məsələn ekrana 5 çıxsın. Nümunədə də göründüyü kimi if hissədə şərt olur, else hissədə yox. Əgər kodumuzda çoxlu sayda bir-birindən asılı şərtlər varsa *if-else-if* strukturundan istifadə olunur. *If-in* içində istənilən sayda *elseif* yazmaq olar. *If-else-ifin* alternativi *switchdir*. Şərtlər çox olduqda *switch* *case*-dən istifadə etmək daha məqsədəuyğundur. Onu da qeyd edim ki, *switch* case bir neçə mümkün dəyəri eyni anda müqayisə etməyə imkan verdiyi üçün kodu daha asan oxunur. *Switch* blokunun daxilində case qeyd olunur məsələn *case* *1*: yaşıl (promptun içində 1 yazdıqda yaşıl gələcək) bu əməliyyatdan sonra digərinə keçməmiş bunu bitirmək yəni *break* yazmaq lazımdır. *Default* da əlavə edə bilərik bu isə şərtlərə uyğun olmayan nə isə daxil edildikdə görünəcək.

**Ternary operator** (üçlü operator) şərti ifadələrin daha qısa və tək sətirdə yazılmasına imkan verir.

Nümunə: condition ? expr1 : expr2

Burada *condition* şərtdir. Şərt true olduqda ekrana 1-ci əks halda 2-ci yazı çıxacaq.

**İf else nümunə**

for(*let* i=1; i<=10;i++){

    if( i===5)

    continue;

    if(i===8)

    break;

console.log(i);

}

Nəticə belə olacaq:

1

2

3

4

6

7

**Switch case nümunə**

*let* month = *Number*(prompt("reqem yaz"))

switch (month){

    case 1:

        console.log("yanvar");

    break;

    case 2:

        console.log("fevral");

    break;

    case 3:

        console.log("mart");

    break;

    case 4:

        console.log("aprel");

    break;

    case 5:

        console.log("may");

    break;

    case 6:

        console.log("iyun");

    break;

    case 7:

        console.log("iyul");

    break;

    case 8:

        console.log("avqust");

    break;

    case 9:

        console.log("sentyabr");

    break;

    case 10:

        console.log("oktyabr");

    break;

    case 11:

        console.log("noyabr");

    break;

    case 12:

        console.log("dekabr");

    break;

}

**Dövrlər**

Bir işin dövrü olaraq icra edilməsini təmin edir. Bu bizim işimizi sürətləndirir və asanlaşdırır.

**For**

Burada başlanğıc , şərt, artma/azalma olur.

Məs: for(i = 0; i < 100; i++){

if(i % 2 === 0){

console.log(i)

}

}

For dövrü təyin edilmiş şərt doğru olduqca işləyir. Şərtin səhv olduğu anda kod işləmir, dayanır.

**While**

While dövrəsi şərtə əsaslanır şərt doğru olduqca işləyir.

let i = 1

while (i <= 10){

console.log(i);

i++;

}

**Do while**

Bu dövr şərt yoxlanılmadan ən az bir dəfə işləyir daha sonra şərt yoxlanılır.

let i=10;

do{

console.log(" Hello World");

}

while(i<5);

**Break**

Kod dayanır.

**Continue**

Dövr işlədiyi zaman continue-nin altında qalan hissəni pas keçir, onu göstərmir, amma kod işləməyə davam edir.

**JavaScript-də arrays**

Verilənlərin nömrələnmiş ardıcıllığına **massiv (array)** deyilir. Massivin ilk elementinin sıra nömrəsi 0-dır. Arraylar indekslərdən və valuelardan ibarət olur. Arrayin i-ci elementi belə göstərilir:arr[i]. Burada i indeksdir. Arrayin uzunluğu belə göstərilir:arr.length. Metodlara baxaq: arr.push(3) massivə yeni dəyişən əlavə edir(məsələn 3); arr.pop() massivin son elementini silir; arr.shift() ilk elementi silir; arr.reverse massivi əksinə ardıcıllıqla düzür; arr.unshift(9,0) massivin əvvəlinə bir və ya bir neçə element artırır (məsələn 9 və 0);indexOf elementin arraydəki ilk indeksini geri qaytarır; lastIndexOf isə əksinə; slice(begin , end) burada biz begin indeksindən başlayaraq end indeksinə qədər olan elementlərdən yeni array yaradırıq.

Nümunə:

let arr = [1,2,3,4]

console.log(arr)

console.log(arr[2])

Nəticə: (4) [1 , 2, 3, 4]

3

Burada 4 indeksdir.

**JavaScript-də object**

Key-value cütlərini saxlamaq üçün istifadə olunur. Burada dəyərlər property adlanır. Açarlar yalnız *string* və *symbol* ola bilər.

*let person = {name: ”Arzu”, age: 11};*

*console.log(person.name)*

*Nəticə: Arzu*

Burada *name* və *age* açarlardır (key). “Arzu” və 11 isə uyğun olaraq onlara verilmiş dəyərdir.

**JavaScript-də funksiyalar**

Müəyyən tapşırığı yerinə yetirmək üçün təyin edilən kod blokudur. Nümunə:

functuion write () {

console.log(“Salam”);

}

*Write();*

Funksiyanın adı feil, parametri isim olmalıdır. İndi isə parametrli funksiya yaradaq. Qeyd edim ki, parametrdən funksiyalara məlumat göndərmək üçün istifadə edirik. Parametrdən istifadə zamanı funksiya daha dinamik olur. Nümunə:

*function(a,b){*

*ceturn a + b;*

*}*

*console.log(sum(5,3))*

*Nəticə: 8*

Funksiyaların növləri: *Declaration*,*expression*,*arrow*. Yuxarıdakı nümunələrdə biz *declaration function*-dan istifadə etmişik. Funksiyaların digər növləri daha qısadır. İndi isə digərlərinə aid nümunələrə baxaq:

**Function Expression**

let write = function(a){

return 5 + a;

};

console.log(write(4));

Nəticə:9

**Arrow Function**

let write = (a) => 7 + a

console.log(write(8));

Bu funksiyalar hoisted deyil.

**Literals**

Dəyişənlərə mənimsədilə biləcək verilən növünə literal deyilir.

**String**

“İlahə”

**Template literal**

`yaşıl top ${value}`

**Number**

19

**Array**

[1,2,3,4,5]

**Object**

{name:İlahə}

**Boolean**

true

**String metodları**

Metodlar stringlərlə işləyərkən işimizi asanlaşdırır. İndi gəlin onlardan ən çox istifadə olunanlarına baxaq.

1. **length**

Simvolların sayını qaytarır.(1-dən başlayır saymağa)

1. **charAt()**

Bu metodla biz *position number* yazaraq istənilən pozisiyadakı simvolu ekrana çıxara bilərik. charAt(0). 0-cı simvolu qaytaracaq. Göründüyü kimi saymağa 0-dan başlayır. Əgər *position number* yazmasaydım birinci stringin birinci simvolunu göstərəcəkdi. Əgər olmayan *position number* versəydim,boş string göstərərdi.

1. **indexOf()**

Bu metodla 2-ci metodun gördüyü işin əksini edirik deyə bilərik. Belə ki biz burada *string* veririk və həmin *stringin* hansı pozisiyada olduğunu görürük*. indexOf(“i”)*. *lastIndexOf(“i”)* sondan başlayıb axtarmağını istəyiriksə belə yazırıq. *lastIndexOf(“i” , 12)* belə yazdıqda isə sonuncunu 12 olaraq qəbul edir. Yəni adi halda 20 simvol var və o 20dən başlayır, amma mən 12 qeyd etmişəmsə 12-cidən başlayacaq.

1. **slice()**

Lazım olan hissəni kəsib qaytarır. Nümunə:

let str = “Ulduz”

console.log(str.slice(2,4))

Nəticə: duz

1. **trim()**

Boşluqları silir. Nümunə:

let str = “ Ulduz”

console.log(str.trim())

Nəticə: Ulduz

1. **toLowerCase**

Bütün hərfləri kiçildəcək. Məsələn console.log(str.toLowerCase()) yazsam nəticə ulduz olacaq.

1. **toUpperCase**

Bütün hərfləri böyüdəcək. Məsələn console.log(str.toUppercase()) yazsam nəticə ULDUZ olacaq.

1. **include()**

Axtarılan varsa true, yoxdursa false qaytarır. Məsələn console.log(str.include(l)) yazsam nəticə true olacaq çünki ulduz sözündə l var.

**Hoisting nədir?**

**Hoisting** Javascriptdə dəyişənlərin və declaration function-ın yuxarı qaldırılmasıdır. Belə ki *var*,*let* və *const* **hoisted-dir**. Yəni biz onları əvvəlcədən çağıra bilərik. Biz *var*-a dəyər verməzdən öncə onu çağırsaq bu dəyişən hoist olunacaq lakin onun dəyəri yox özü **hoist** olunduğundan undefined göstəriləcək. **Hoist** olunan zaman *var* dəyişəni yaddaşa yazılır sadəcə olaraq hələ dəyər qəbul etmədiyindən undefined göstərir. Digər dəyişənlərdə yəni *let* və *const*-da isə yenə dəyişənlər **hoist** olunur dəyərləri isə yox. Bu zaman var-dan fərqli olaraq error görünür. Bildiyimiz kimi 3 cür funksiya var. Bunlardır: *declaration, expression, arrow*. *Function declaration* **hoisted**-dir digərləri isə belə deyil. Digər iki funksiyanı təyin etməzdən öncə çağırmaq olmaz. Error çıxacaq. Çünki funksiyanın özü **hoisted** deyil buna görə də yaddaşa yazılmır. Qısaca belə deyə bilərik ki, dəyişənlərin və *function declaration*-un hoist edilməsi o deməkdir ki, onlar mövcuddur lakin təyin edilməyib.

*Nümunələr*

1. console.log(x); //undefined

var x = 3

console.log(x);

Nəticə: 3

1. console.log(x); error

let x = 5

1. write(); error

let write = (a) => 7 + a

Düzgün hal: let write = (a) => 7 + a

console.log(write(8));

Nəticə: 15