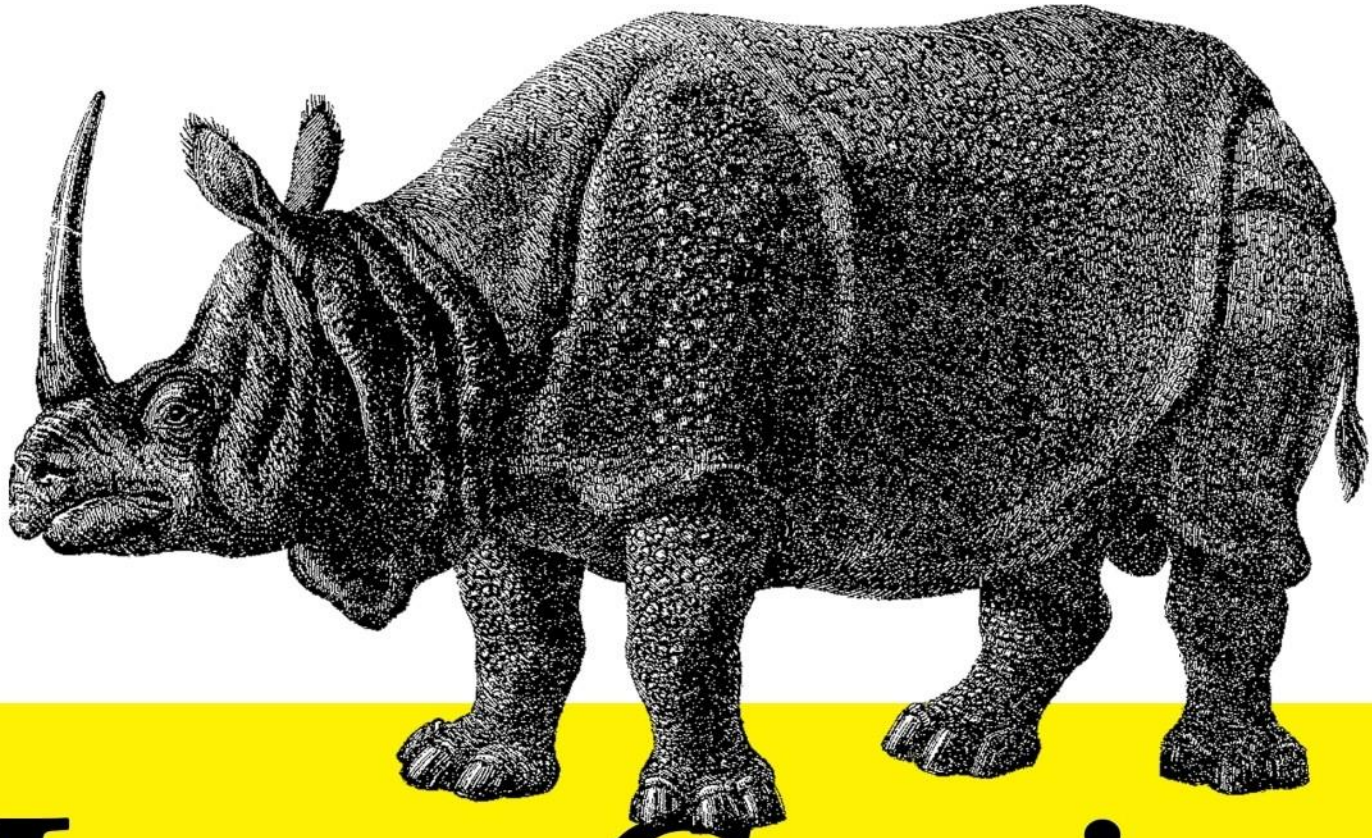


JavaScript

geniş izah,
ətraflı baxış,
əsas anlayışlar

Devid Flanaqan

JavaScript

Ətraflı izah

I hissə

1

JavaScript-ə giriş

JavaScript – obyekt yönümlü proqramlaşdırma imkanları olan interprerativ proqramlaşdırma dilidir. Sintaksis nöqteyi-nəzərindən JavaScript-in baza dili C, C++ və Java PD-lərinin proqram konstruksiyalarına bənzəyir (məs.: if, while və && operatort). Ancaq bu oxşarlıq yalnız *sintaktis oxşarlıq* ilə məhdudlaşır.

JavaScript – tipləşdirilməmiş dildir , yəni bu dildə dəyişənlərin tiplərini müəyyən etmək tələb olunur. JavaScript-də obyektlər, ad xüsusiyyətlərində sərbəst qiymətləri əks etdirir. Bu xüsusiyyətinə görə Perl PD-sinin assosiativ massivlərini, C strukturlarını və ya C++ və ya Java obyektlərini xatırladırlar. JavaScript dilinin nüvəsi sadə məlumat tipləri, ədədlər, sətirlər və Bull qiymətlərini dəstəkləyir.

Bundan başqa bu dil massivlər, tarixlərin və müntəzəm ifadə obyektlərinin inteqrasiya edilmiş

dəstinə malikdir. Adətən JavaScript veb-brauzerlərdə tətbiq edilir. Obyektlərin tətbiqi nəticəsində bu dilin imkanları daha genişdir. JavaScript, istifadəçi ilə qarşılıqlı təsiri təşkil etməyə, veb-brauzeri idarə etməyə və veb-brauzerin pəncərəsinin daxilində əks etdirilən sənədin tərkibini dəyişdirməyə imkan verir. JavaScript veb səhifələrin HTML-koduna tətbiq edilmiş ssenarilər vasitəsilə inteqrasiya edir. Bir qayda olaraq, bu versiya JavaScript-in kliyent dili adlanır. Qeyd etmək lazımdır ki, ssenari kliyenti veb-serverdə deyil, kompüterinizdə (oflayn rejimdə) icra edilir. JavaScript və bu dilin dəstəklədiyi məlumat tipləri dili beynəlxalq standartlara əsaslanır. Bunun sayəsində reallaşdırmalar arasında çox gözəl uyğunluq yaranır.
Bunun sayəsində reallaşdırmalar arasında çox gözəl uyğunluq yaranır. JavaScript-kliyentinin bəzi hissələri rəsmi standart, bəzi hissələri hissələr de-fakto standartdır, amma kliyentin elə hissələri də var ki, brauzerin konkret versiyasına uyğun spesifik genişlənmədir. Müxtəlif brauzerlərdə JavaScript reallaşdırmalarının uyğunluğu JavaScript-in kliyent dilindən istifadə edən proqramçıları tez-tez düşündürür və narahat edir. Bu fəsildə dilin imkanlarının faktiki öyrənməsinə keçməzdən əvvəl JavaScript-in qısa icmalı və giriş informasiyası verilir.
Bundan başqa, praktik veb-proqramlaşdırma JavaScript-in kliyent dilində bir neçə kod fraqmenti bu fəsildə nümayiş etdirilir.

1.1. JavaScript nədir

JavaScript-in haqqında çoxlu dezinformasiyalar və qarışıq məlumatlar mövcuddur. JavaScript-in öyrənilməsindən əvvəl, bu dillə bağlı yayılmış bəzi mifik məlumatları aydınlaşdıraq.

1.1.1. JavaScript – Java deyil

JavaScript haqqında ən geniş yayılmış yanlış fikirlərdən biri ondan ibarətdir ki, guya bu dil Sun Microsystems şirkəti tərəfindən hazırlanmış Java proqramlaşdırma dilinin sadələşdirilmiş versiyasıdır. Bəzi sintaktis oxşarlıq və veb-brauzerə icra edilə bilən tərkib mənimsətmək

qabiliyyətindən savayı, bu iki dil arasında heç bir bağlılıq yoxdur. Adlarının oxşarlığına gəlincə isə, bu sadəcə marketinq siyasətinin nəticəsidir (dilin ilkin adı LiveScript olmuşdur, sonralar isə bu dil JavaScript adlandırdırıldı). Ancaq JavaScript və Java bir-birinə qarşılıqlı təsir edə bilər.

1.1.2. JavaScript sadə dil deyil

Bir halda ki, JavaScript izah edilən dildir və bu dil adətən proqramlaşdırma dili kimi deyil ssenarilər şəklində yerləşdirilir, bu halda nəzərə almaq lazımdır ki, ssenari dilləri təkcə, çox mürəkkəb şəraitdə işləməyi bacaran proqramçıdan tutmuş, elementar proqramlaşdırma bilikləri olan proqramçı üçün yönəlməmişdir, hətta bu dil adi istifadəçi üçün də nəzərdə tutulmuşdur. Əslində, JavaScript tipləşdirilməmiş xüsusiyyəti sayəsində, təcrübəsiz proqramçılar tərəfindən tiplərin təyini zamanı buraxılan səhvləri güzəştə gedir. Buna görə də əksər veb-dizaynerlər dəqiq reseptlər üzrə yerinə yetirilən məsələlərin məhdudlaşmış əhatədə həlli üçün JavaScript-dən istifadə edir. Ancaq JavaScript-in zahiri sadəliyi baxmayaraq, daxilində bir o qədər də mürəkkəb və dəyərli proqramlaşdırma dili gizlənir. JavaScript imkanlarını tam anlamadan bu dilin köməyilə qeyri-adi məsələləri həll etməyə çalışan proqramçılar hazırlanma prosesində tez-tez məyus olurlar. Bu kitabda, JavaScript-in hərtərəfli təsviri verilmişdir. Əgər bu kitabdan əvvəl hazır məzmunu ehtiva edən JavaScript məlumat kitabçalarından istifadə etmisinizsə, bu kitabdakı fəsillərinin dərinliyi və ifadə təfərrüatı yəqin ki, sizi təəccübləndirəcək.

1.3. JavaScript kliyenti

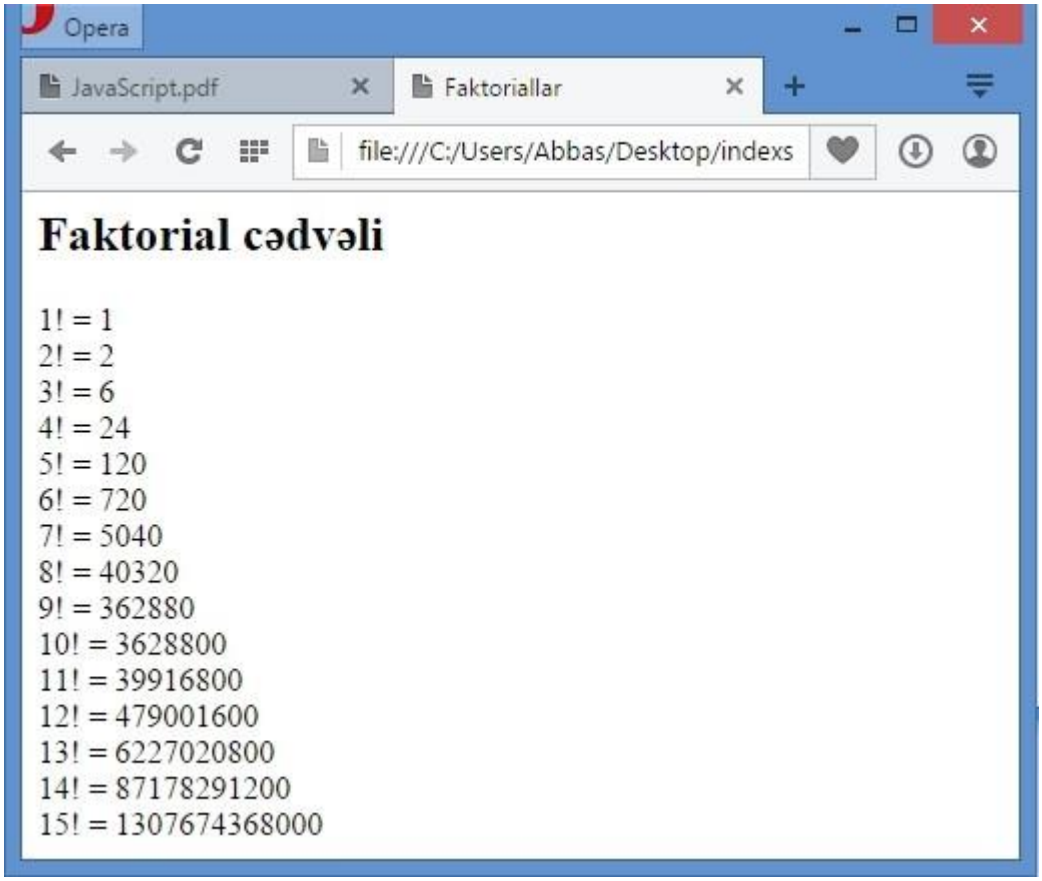
JavaScript interpretatoru veb-brauzerə JavaScript kliyenti vasitəsilə inteqrasiya edir. Elə buna görə də, JavaScript dedikdə, insanların ağlına ilk öncə JavaScript kliyenti gəlir. Bu kitabda JavaScript-in kliyent dili bu dilin alt çoxluğunu təşkil edən JavaScript-bazası ilə birlikdə təsvir edilir.
JavaScript kliyentinin daxilində JavaScript interpretatorunu və obyektiveb-brauzerlə müəyyən edilən sənədin obyekt modeli (Document Object Model, DOM) yerləşir.
Sənədlər JavaScript-ssenariləri özündə saxlaya bilər. Bu ssenarilər, öz növbəsində DOM modelindən istifadə edərək, sənədin və ya idarə etmənin üsulunun modifikasiyası üçün istifadə edilir. Başqa sözlə demək olar ki, JavaScript kliyenti veb səhifələrin statik tərkibin davranışını müəyyən etməyə imkan verir. JavaScript kliyenti veb-proqramların hazırlamasının texnologiyalarının əsasıdır, DHTML (16-cı fəsil), AJAX (20-ci fəsil) arxitekturaların). ECMA-262 spesifikasiyası JavaScript-in baza dilinin standart versiyasını müəyyən edir və World Wide Web Consortium (W3C) təşkilatı tərəfindən standartlaşdıran DOM spesifikasiyasını hazırlamışdır hansı ki, brauzer bu standartı öz obyekt modelində dəstəkləməlidir. W3C DOM standartı ən məşhur brauzerlərlə tam dəstəklənir.
Yalnız – Microsoft Internet Explorer tərəfindən dəstəklənmir; bu brauzerdə hadisələrin emalı mexanizminin dəstəyi yoxdur.

1.3.1. JavaScript kliyentindən istifadə nümunələri

JavaScript interpretatoru ilə təchiz edilmiş veb-brauzer İnternet vasitəsilə JavaScript-ssenarilər şəklində icra edilən tərkibi göstərə bilir. Nümunə 1.1-də JavaScript dilində veb-səhidəyə inteqrasiya edilmiş sadə proqram göstərilmişdir.
Nümunə 1.1. JavaScript dilində sadə proqram

```
<html>
<head>
```

```
<title>Faktoriallar</title>
</head>
<body>
<h2>Faktorial  cədvəli</h2>
<script>
var fact = 1;
for  (i  =  1;  i  =< 15;  i++)
     {
     fact = fact*i;
     document.write (i  + "! = "+ fact + "<br>");
     }
</script>
</body>
</html>
```



Şəkil 1.1 JavaScript-də faktorial siyahısı alqoritmi

Sözügedən proqram JavaScript-i dəstəkləyən brauzer vasitəsilə icra edildikdə, şəkildəki nəticəni verəcək.

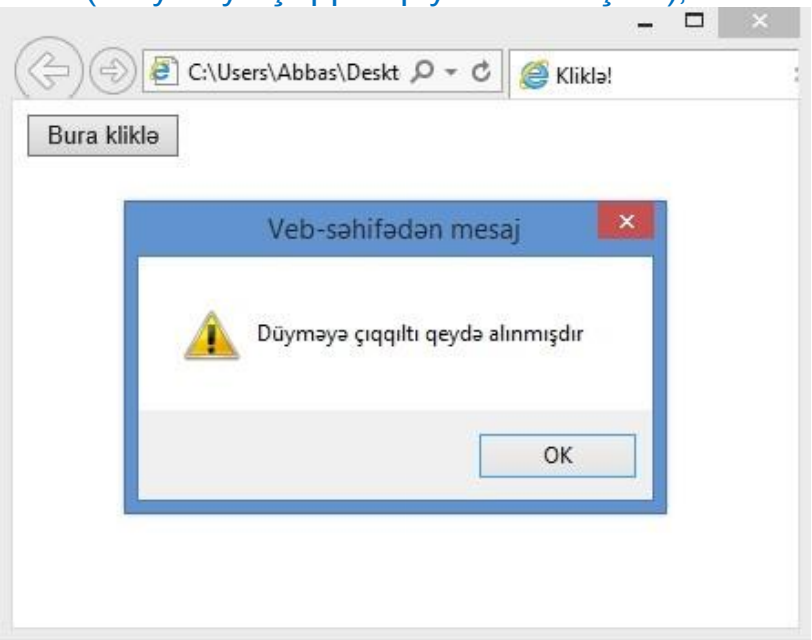
Bu nümunədən göründüyü kimi, JavaScript-kodunun HTML-faylına yerləşdirilməsi üçün <script> və </script> teqlərindən istifadə edilmişdir. <script> teqi haqqında 13-cü fəsildə daha aydın bəhs edilir. Burada əsas – document.write() metodundan istifadə edilir. Bu metod, sənədin veb-brauzerlə yükləməsi anında HTML-sənədin daxilində dinamik HTML-mətn istehsal etməyə imkan verir.

JavaScript yalnız HTML-sənədin tərkibini deyil, həm də onların davranışının idarəetməsini təmin edir. Başqa sözlə, JavaScript-proqram istifadəçisinin hərəkətlərinə reaksiya verə bilər (məs.: mətn sahəsinə qiymətin daxil edilməsi və ya sənəddə təsvir sahəsində siçandan icra olunan çıqqıltıya cavab). Bu sənəd üçün hadisələrin emalçılarının təyini yolu ilə əldə edilir. Məsələn, müəyyən

hadisənin yaranması anında icra edilən JavaScript-kodlarını nümunə göstərmək olar. Nümunə 1.2-də HTML-kodun sadə fraqmenti göstərilmişdir. Burada emalçı çıqqıltıya cavab olaraq müəyyən hadisəni (burada xəbərdarlıq) icra edir.

Nümunə 1.2. JavaScript dilində hadisə emalçısı vasitəsilə HTML-düymə

```
<button onclick="alert ('Düyməyə çıqqıltı qeydə alınmışdır');">Bura kliklə</button>
```



Şəkil 1.2 JavaScript-də hadisə emalçısı

Şəkildə. 1.2 Düyməyə çıqqıltının nəticəsi göstərilmişdir.Nümunə 1.2-dəki onclick atributu – icra edilən JavaScript-kodunun sətridir. Göründüyü kimi bu atribut klikləməyə cavab verir. Bu nümunədə onclick hadisəsinin emalçısı alert() funksiyasına səbəb olur. Şəkildən görün-düyü kimi. 1.2, alert() funksiyası göstərilən mesajla birlikdə dialoq pəncərəsini vasitəsilə təsvir edilir. 1.1 və 1.2 nümunələri JavaScript kliyentinin ən sadə imkanlarıdır. Bu dilin real gücü ondan ibarətdir ki, ssenarilər HTML-sənədin tərkibində yerləşir.

1.4. JavaScript-in başqa sahələrdə istifadəsi

JavaScript – ümumi təyinatlı proqramlaşdırma dilidir və onun istifadəsi tək veb-brauzerlər ilə məhdudlaşmır. Əvvəllər JavaScript, istənilən proqrama ssenarilər ilə yerləşdirilir və icra edilir. İlk günlərdən Netscape şirkətinin veb-serverləri JavaScript ssenarilərini icra etməyi bacaran JavaScript interpretatoru dəstəkləyirdi. Oxşar üsulla Microsoft korporasiyası Internet Explorer-ə əlavə olaraq öz veb serverlərində IIS və Windows Scripting Host məhsulu üçün JScript interpretatoru istifadə edir. Adobe şirkəti öz Flash- fayllarının oxuyucusunun idarə etməsi üçün JavaScript-dən törəyən dili cəlb etdi. Həmçinin Sun şirkəti Java 6.0 distributivinə JavaScript interpretatoru quraşdırdı və bunun sayəsində istənilən Java-proqramına ssenarilərin yerləşdirilməsi imkanı əhəmiyyətli dərəcədə yüngülləşdi.

Netscape və Microsoft öz JavaScript interpretatorlarının reallaşdırmalarını, öz proqramlarını əlavə etmək istəyən şirkətlər və proqramçılar üçün əlçatan etdi. Netscape şirkəti tərəfindən yaradılmış interpretator açıq mənbəli və azad yayılan PT olub, hal-hazırda Mozilla (http://www.mozilla.org/js/) təşkilatı vasitəsilə yayılır. Mozilla faktiki olaraq JavaScript 1.5 interpretatorunin iki müxtəlif versiyasını yayır: biri C dilində yazılmışdır və SpiderMonkey adlanır, o biri isə Java dilində yazılmışdır və kitabın müəllifinin fikrincə çox təkmil şəkildə

hazırlanmış və Rhino *(kərgədan)* adı verilmiş interpretatordur.

1.5. JavaScript öyrənilməsi

İstənilən yeni proqramlaşdırma dilinin metodikası zamanı praktikadan istifadə etmək lazımdır. Bu kitabı oxuduğunuzda, sizə JavaScript imkanlarını yoxlamağı məsləhət görürəm. Məsələn, sadə funksiyalardan ibarət kodlardan başlaya bilərsiniz. Proqram kodunu oxumağa və anlamağa çalışın. JavaScript öyrənilməsinə ən aşkar yanaşma – sadə ssenarilərin yazılışıdır. JavaScript kliyentinin üstünlüklərindən biri ondan ibarətdir ki, işləmə mühiti hər hansı, veb-brauzer və ən sadə mətn redaktoru ilə qurulmuşdur. JavaScript-də proqramlar yazmaq üçün, xüsusi proqram təminatının yüklənməsinə ehtiyac yoxdur. Məsələn, fatorialların yerinə Fibonaççi ədədlərinin ardıcıllığını göstərmək üçün, aşağıdakı nümunə 1.1 kopyalamaq olar:

```
<script>
document.write ("<h2>Fibonaççi ədədləri</h2>);
for (i=0, j=1, k=0, fib =0; i<50; i++, fib=j+k, j=k, k=fib)
{
   document.write ("Fibonacci ("+i+") ="+ fib);
   document.write ("</br>);
}
</script>
```

Bu kod sizə qəliz görünə bilər (əgər kodu anlaya bilmədizsə, narahat olmayın), amma belə qısa proqramlarla təcrübə etmək üçün, kodu olduğu kimi kopyalamaq və lokal URL-ünvanlı fayl kimi veb- brauzerdə onu icra etmək kifayətdir. Nəzərə alın ki, hesablamaların nəticəsini göstərmək üçün document.write() metodundan istifadə olunur. Bu metoddan istifadə, JavaScript ilə tanışlıq zamanı faydalıdır. Alternativ olaraq dialoq pəncərəsində sadə mətn nəticəsini göstərmək üçün alert() metodunu tətbiq etmək olar:

```
alert ("Fibonacci (" + i  + ") = " + fib);
```

Qeyd edək ki, JavaScript ilə belə sadə sınaq kodlarını HTML faylın daxilində <html>, <head> və <body> teqlərinin içərisində yerləşdirmək olar.

JavaScript ilə sınaqların daha da sadələşdirməsi üçün URL-ünvana mənimsədilə bilən javascript: psevdoprotokol spesifikatoru yaradılmışdır. Bu üsul ayəsində JavaScript-də ifadənin və qiymətin nəticənin hesablamaq olar. Belə URL-ünvan psevdoprotokol spesifikatorundan (javascript:) ibarətdir, hansı ki, burada sərbəst olaraq JavaScript-kodu (təlimatlar biri-birindən nöqtəli vergül ilə ayrılır) göstərilir. URL-ünvanı psevdoprotokol vasitəsilə yükləndikdə, brauzer sadəcə JavaScript-kodunu icra edir. Belə URL- ünvanda ifadənin son qiyməti sətir tipinə dəyişdiriləcək və bu sətir yeni sənəd kimi veb-brauzer vasitəsilə göstəriləcək. Məsələn, bəzi operatorların və JavaScript dilinin təlimatlarını yoxlamaq üçün, veb-brauzerin ünvan sahəsində aşağıdakı URL-ünvanları yığmaq olar:

```
javascript:5%2
javascript:x = 3; (x<5)? "x qiyməti kiçikdir": "x qiyməti daha böyükdür"
javascript:d = new  Date(); typeof d;
javascript:for (i=0, j=1, k=0, fib=1; i<5; i++, fib=j +k, k=j, j=fib) alert (fib);
javascript:s=""; for (i in navigator) s+=i+":" +navigator [i] +" \n"; alert (s);
```

Firefox təksətirli veb- brauzerində ssenarilər JavaScript-konsollarını ehtiva edir. Bu konsolu Alətlər menyusundan işə salmaq olar. Burada sadəcə yoxlamaq istədiyiniz ifadəni və ya təlimatı daxil etmək lazımdır. JavaScript- konsoldan istifadə zamanı psevdoprotokol spesifikatorunu (javascript:) işə salmaq olar.

JavaScript- kodun öyrənilməsinin baza metodikası, başqa dillərin metodikası ilə uyğun gəlir.
Əgər siz JavaScript- ssenarilərində tez-tez xətalarla rastlaşırsınızsa, ehtimal ki, JavaScript-in cari sazlayıcısı ilə maraqlanacaqsınız. Internet Explorer-də Microsoft Script Debugger, Firefox-da Venkman adlı məlum genişlənmənin modul sazlayıcısından istifadə etmək olar. Bu alətlərin təsviri kitabın əhatə mövzundan kənar olmasına baxmayaraq, siz asanlıqla İnternetdə, hər hansı bir axtarış sistemindən istifadə edib bu barədə məlumat toplaya bilərsiniz. Daha bir alət jslint-dir. Ciddi desək bu sazlayıcı deyil; Bu alət JavaScript-proqram kodunda olan xətaları axtarmağa yönəlmişdir (*http:// jslint.com*).

2

Leksik struktur

Proqramlaşdırma dilinin leksik strukturu – proqramların yazılma qaydalarını müəyyən elementarların dəstidir. Dilin aşağı səviyyəli sintaksisi; dəyişənlərin adları, şərhlər üçün istifadə edilən simvollar, bir təlimatı digərindən ayırmaq və s-dir. Bu qısa fəsil JavaScript-in leksik strukturunu sənədlərlə əsaslandırır.

2.1. Simvol yığımı

JavaScript-də proqramların yazılışı zamanı Unicode simvol yığımından istifadə olunur. Yalnız ingilis dili üçün ASCII-yə və ISO Latin-1-ə yaxın kodlaşdırmadan və əsas qərbi avropa dillərinin 8-dərəcəli kodlaşdırmasının 7- dərəcəli kodlaşdırmasından fərqli olaraq, Unicode-un 16-dərəcəli kodlaşdırması praktik olaraq istənilən yazı dilinin tətbiqini təmin edir. Bu imkan beynəlmiləlləşdirmə üçün və xüsusilə "ingilis olmayan" proqramçılar üçün əhəmiyyətlidir.
Amerikanlar və digər ingilis dilində danışan proqramçılar proqramları, adətən yalnız ASCII və ya Latin-1 kodlaşdırmasını dəstəkləyən mətn redaktorunun köməyi ilə yazırlar və buna görə onlarda Unicode tam simvolların dəsti əlçatan deyil. Ancaq əks tərəfin bu barədə heç bir çətinliyi, bir halda ki, çünki ASCII və Latin-1 kodlaşdırmaları Unicode kodlaşdırmasının alt çoxluqlarını təşkil edir və hər hansı bir simvol dəstinin köməyi ilə yazılmış JavaScript-proqramı tamamilə düzgündür.
ECMAScript v3 standartı JavaScript-proqramlarında istənilən yerdə Unicode-simvolların mövcudluğuna imkan edir.

2.2. Registrə həssaslıq

JavaScript – *registrə həssas* dildir. Bu isə o deməkdir ki, əsas sözlər, dəyişənlər, funksiya adları və dilin ixtiyari bir identifikatoru həmişə böyük və kiçik hərflərin eyni cür yığımını ehtiva etməlidir. Məsələn, while açar sözü "While" və ya "WHILE" kimi deyil "while" kimi yazılmalıdır. Online, Online, OnLine və ONLINE – bu dörd müxtəlifin dəyişən adları da analojidir. Qeyd edək, HTML dili, JavaScript-dən fərqli olaraq, registrə həssas deyil. HTML-in və JavaScript-kliyentinin yaxın əlaqəsini nəzərə alaraq bu fərq qarışıqlığa gətirib çıxara bilər. Nəzərə alın ki, JavaScript- obyektləri və onların xüsusiyyətləri ilə HTML dilinin teq və atributları eynidir. Əgər HTML-dilində bu teqlər və atributlar istənilən registrdə yazıla bildiyi halda, JavaScript-də adətən kiçik hərflərlə yazılmalıdır. Məsələn, HTML-də onclick hadisəsinin emalçısının atributu əksər hallarda onClick kimi, ancaq JavaScript-

kodunda (və ya XHTML-sənədində) onclick kimi göstərilməlidir.

2.3. Simvol-ayırıcılar və sətir keçidləri

JavaScript – proqramın daxilində leksemləri arasında boşluqlara, tabulyasiyalara və sətir keçidlərinə *məhəl qoymur*. Buna görə də boşluq, tabulyasiya və yeni sətir simvolları proqram mətnində xüsusi simvolların köməyilə məhdudiyyətsiz istifadə edilə bilər. Ancaq bu barədə, növbəti bölmədə danışacağıq.

2.4. Vacib olmayan nöqtəli vergüllər

Sadə JavaScript-də təlimatları C, C++, və Java PD-ləri kimi adətən nöqtəli vergül (;) simvolları ilə qurtarır. Nöqtəli vergül təlimatlar bir-birindən ayırmasına xidmət edir. Ancaq JavaScript-də əgər hər təlimat ayrı sətirdə yerləşirsə, nöqtəli vergülü qoymamaq olar. Məsələn, aşağıdakı fraqment nöqtəli vergüllərsiz ola bilər:

```
a = 3;
b = 4;
```

Ancaq əgər hər iki təlimat bir sətirdə yerləşdirilmişsə, onda məcburi olaraq nöqtəli vergül qoyulmalıdır:

```
a = 3; b = 4;
```

Proqramlaşdırmada nöqtəli vergüllərin qoyulma kriteriyalarını düzgün anlamaq çətindir və buna görə istənilən məqamda nöqtəli vergüllərdən istifadə etməyi özünüzə vərdiş edin. Nəzəri alın ki, JavaScript, istənilən iki leksem arasında sətir boşluğu güman edir, amma JavaScript-in sintaktis analizatorunun avtomatik nöqtəli vergülləri qoyma vərdişinin bəzi istisnaları mövcuddur. Əgər proqram kodunun sətirində bitirilmiş təlimatdan sonar yeni sətirə keçilibsə, JavaScript-in sintaktis analizatoru nöqtəli vergülü avtomatik qoyur.
Məsələn, aşağıdakı fraqmentə baxaq:

```
return
true;
```

JavaScript-in sintaktis analizatoru güman edir ki, proqramçının yazdığı kod aşağıdakı kimidir:

```
return;
true;
```

Hərçənd ki əslində proqramçı, return true; yazmaq istəyirdi. Gördüyünüz kimi, diqqətli olmaq lazımdır – bu kod sintaktis cəhətdən səhv deyil, amma kodda aşkar olmayan nasazlığı mövcuddur. Oxşar xoşagəlməz hadisə: break outerloop; yazanda da yarana bilər.
JavaScript break açar sözündən sonra nöqtəli vergülü avtomatik qoyur və növbəti sətiri icra edildikdə xəta ilə üzləşir. Analoji səbəblərə görə postfiks operatorlarını (++ və --) (5-ci fəsilə baxmaq), aid olduğu ifadənin sətirində yerləşdirmək məcburidir.

2.5. Şərhlər

JavaScript, həmçinin Java və C++, C stilində olan şərhləri dəstəkləyir. Sətir sonunda // simvolları arasında daxil edilən istənilən mətnə, şərh kimi baxılır. Şərhlərin icraedici funksiyası yoxdur. Həmçinin / * və */ simvolları arasında daxil edilən istənilən mətnə şərh kimi baxılır. C stilində sözügedən şərhlər bir neçə sətirdən ibarət ola bilər və qoyulmuş ola bilmir. Aşağıdakı kod sətirlərində düzgün JavaScript-şərhləri göstərilmişdir:

```
// Bu təksətirli şərhdir.
/* Bu da həmçinin şərhdir */ // və bu başqa cür şərh formasıdır.

/*
* Bu da daha bir şərh formasıdır.
* Bu formada olan şərhlər bir neçə sətirdə yerləşdirilə bilər.
*/
```

2.6. Literallar

Literal – proqramın mətnində bilavasitə göstərilmiş qiymətdir. Aşağıda bəzi literal nümunələri göstərilmişdir:

```
12              // On iki ədədi
1.2             // Bir tam onda iki onluq ədədi
"hello world"   // Mətn sətiri
'Hi'            // Başqa cür sətir
true            // Məntiqi qiymət
false           // Digər bir məntiqi qiymət
/javascript/gi // Müntəzəm ifadə (şablon üzrə axtarış üçün)
null            // Obyektin yoxluğu
```

ECMAScript v3-də ifadələrdə həmçinin massiv literalları və obyekt literalları dəstəklənir. Məsələn:

```
{ x:1, y:2}     // Obyektin inisializatoru
[1,2,3,4,5]     // Massivin inisializatoru
```

Literallar – istənilən proqramlaşdırma dilinin mühüm hissəsidir, çünki, literalsız proqram yazmaq mümkün deyil. JavaScript-də olan müxtəlif literallar 3-cü fəsildə təsvir edilmişdir.

2.7. İdentifikatorlar

İdentifikator – sadəcə verilən addır. JavaScript-də identifikatorlar, dəyişənlərin və funksiyaların adları kimi, həmçinin bəzi dövrlərin nişanları kimi çıxış edir. Mümkün identifikatorların formalaşması qaydaları Java və bir çox başqa proqramlaşdırma dillərinin qaydaları ilə eynidir. Birinci simvol hərf, sətir xətti (_) simvolu və ya dollar ($) işarəsi[1] olmalıdır. Birinci simvoldan sonra

[1] ECMAScript v3 həmçinin $ işarəsini də dəstəkləyir, lakin JavaScript 1.1 versiyasına qədər olan versiyalarda, bu

istənilən hərf, rəqəm, sətir xətti simvolu (_) və ya dollar işarəsi ola bilər. (Birinci simvol heç bir halda rəqəm ola bilməz, çünki bu halda şərhçi ədədləri identifikatorlardan ayırmağa çətinlik çəkir.) Mümkün identifikator nümunələri:

```
i
my_variable_name
v13
_dummy
$str
```

ECMAScript v3-də identifikatorlar Unicode simvol dəstində olan bütün hərfləri və rəqəmləri ehtiva edə bilər. Standartın bu versiyasına qədər JavaScript- identifikatorları ASCII dəsti ilə məhdudlaşmışdı. Bu işarə yalnız kodun generasiyası vasitələri üçün nəzərdə tutulmuşdur, buna görə də identifikatorlarda bu işarənin istifadəsindən çəkinmək lazımdır. 2.8. Ehtiyata saxlanılan sözlər Unicode escape- ardıcıllıqları – identifaktorlarda 16 dərəcəli 4 onaltılıq rəqəmdən ibarət simvol kodu olan \u simvol birləşməsi vasitəsilə icra edilir. Məsələn, π identifikatoru üçün \u03c0 kimi yazmaq olar. Bu sintaksis bu cür narahat görünə bilər, amma mətnlər ilə iş zamanı tam Unicode simvol dəstini dəstəkləməyən redaktorlarda və ya digər vasitələrdə bu Unicode-simvollarının JavaScript- proqramların transliterasiyasını təmin edir.
Nəhayət, identifikatorlar ilə JavaScript-də digər məqsədlər üçün istifadə edilən açar sözləri bir-birilə uyğun gəlmir. Aşağıdakı bölmədə JavaScript-in xüsusi ehtiyacları üçün ehtiyata saxlanmış açar sözlər göstərilmişdir.

2.8. Ehtiyata saxlanılan sözlər

JavaScript-də bir neçə ehtiyata saxlanılan söz mövcuddur. Bu sözlər JavaScript-proqramlarında identifikator kimi (dəyişən, funksiya və dövrlərin nişan adları kimi) çıxış edə bilməz. Cədvəl 2.1-də ECMAScript v3-də standartlaşdırılmış açar sözləri göstərilmişdir. JavaScript interpretatoruna görə bu sözlər xüsusi qiymətə malikdir və dilin sintaksis hissəsini təşkil edir.
Cədvəl 2.1. JavaScript-in ehtiyata saxlanılan açar sözləri

break	do	if	switch	typeof
case	else	in	this	var
catch	false	instanceof	throw	void
continue	finally	new	true	while
default	for	null	try	with
delete	function	return		

Cədvəl. 2.2 başqa açar sözləri göstərilmişdir. Hal-hazırda bu sözlər JavaScript-də istifadə olunmur, amma dilin gələcək inkişaf etdirmələri üçün ECMAScript v3-də ehtiyatda saxlanmışdır.
Cədvəl 2.2. ECMA genişlənilmələri üçün ehtiyata saxlanmış sözlər

Abstract	double	goto	native	static
Boolean	enum	implements	package	super
byte	export	import	private	synchronized
char	extends	int	protected	throws
class	final	interface	public	transient
const	float	long	short	volatile

```
debugger
```

ECMAScript v4 standartının cari layihələri as, is, namespace və use açar sözlərinin tətbiqinə imkan verir. Hərçənd ki, JavaScript cari interpretatorları bu dörd açar sözündən identifikator kimi istifadə edilməsini qadağan etmir, ancaq identifikatorlarda bu sözlərin istifadəsindən çəkinmək lazımdır.

Bundan başqa, JavaScript dilində qabaqcadan müəyyən edilmiş qlobal dəyişən və funksiya identifikatorlarının istifadədən çəkinmək lazımdır. Cədvəl. 2.3-də ECMAScript v 3 standartı ilə müəyyən edilən qlobal dəyişənlər və funksiyalar göstərilmişdir. Konkret reallaşdırmalar öz qlobal görünmə sahəsi vasitəsilə qabaqcadan müəyyən edilmiş elementləri ehtiva edə bilər. Bundan başqa, JavaScript-in konkret platforması üzrə (müştəri, server və başqalar) bu siyahını genişləndirə bilərik.

Cədvəl 2.3. Çəkinməli olduğumuz digər identifikatorlar

argument	encodeURI	Infinity	Object	String
Array	Error	isFinite	parseFloat	SyntaxError
Boolean	escape	isNaN	parseInt	TypeError
Date	eval	Math	RangeError	undefined
decodeURI	EvalError	NaN	ReferenceError	unescapedecode
URIcomponent	Function	Number	RegExp	URIError

3

Məlumat tipləri və qiymətlər

Kompüter proqramları qiymətlərin manipulyasiya nəticəsində işləyir. Proqramlaşdırma dilində təqdim edilmiş və emal edilə bilən, qiymətlər məlumatlar tipindən *(data types)* və proqramlaşdırma dilinin ən fundamental xarakteristikalarından biri olan bu tipi dəstəkləyən məlumat tiplərinin dəstindən ibarətdir. JavaScript üç elementar məlumat tipi ilə işləməyə imkan verir: ədədlər, mətn sətirləri (və ya sadəcə sətirlər) və məntiqi doğruluq qiymətləri (və ya sadəcə məntiqi qiymətlər). JavaScript-də həmçinin iki bayağı məlumat tipi təyin edilir: null və undefined. Bu qiymətlərin hər biri yalnız bir qiymətə malikdir.
JavaScript bu elementar məlumat tiplərindən savayı obyekt *(object)* kimi bilinən tərkib məlumat tipini də dəstəkləyir. Obyekt (yəni, obyekt məlumat tipinin üzvü) qiymətlərin (ədədlər sətirlər və ya elementlər kimi və ya daha mürəkkəb, məsələn başqa obyektlər) kolleksiyasını təşkil edir. Obyektlər JavaScript-də ikili təbiətə malikdir: obyekt adlandırılmış qiymətlərin qaydaya salınmamış kolleksiyası kimi və ya nömrələnmiş qiymətlərin qaydaya salınmış kolleksiyası kimi. Obyektin sonuncu vəziyyətində obyekt massiv *(array)* adlanır. Hərçənd ki, JavaScript-də obyektlər və massivlər ayrı-ayrı məlumat tipidir və bu kitabda ayrı tiplər kimi baxılır.
JavaScript-də obyektin daha bir xüsusi tipi müəyyən edən funksiyadır *(function)*. Funksiya – icra edilən kodun bağlandığı obyektdir. Funksiya *(invoked),* müəyyən əməliyyatın icra edilməsi üçün çağırıla bilər. Massivlər kimi, funksiyalar da, digər obyekt növlərindən fərqlər və JavaScript-də funksiyalar ilə işləmək üçün xüsusi sintaksis müəyyən edilmişdir. Buna görə biz obyektlərdən və massivlərdən asılı olmayaraq funksiyalarla tanış olacağıq.
JavaScript-in baza dilində funksiyalardan və massivlərdən başqa obyektlərin bir qədər xüsusi növləri də vardır. Bu obyektlər yeni olmayan məlumat tiplərini və yalnız obyektlərin yeni siniflərini (classes) dəstəkləyir.
Bu fəsilin qalan hissəsində elementar tiplərin hər biri təfərrüatı ilə təsvir edilmişdir.
Fəsil başlanğıc səviyyə üçün bir qədər dar ixtisaslaşdırılmış təfərrüatları özündə ehtiva edir.

3.1. Ədədlər

Ədədlər – xüsusi izah tələb etməyən əsas məlumat tipidir. JavaScript C və Java kimi PD-lərindən fərqli olaraq, o qədər ki, etmir tam və natural ədədlər arasında fərq qoymur. JavaScript-də bütün ədədlər 64-dərəcəli natural maddi qiymətlər ilə təqdim edilir. Bu format IEEE 754.1 standartı ilə təyin edilmişdir.[2] Bu format $\pm 1{,}7976931348623157 \times 10^{308}$ dən $\pm 5 \times 10^{-324}$ qədər olan intervalda olan ədədləri təyin edə bilir.
Bilavasitə JavaScript- proqramının kodunda olan ədəd, ədəd literalı adlanır. JavaScript bir neçə

[2] Bu format double tipinin formatına bənzədiyi üçün kimi Java-proqramçılarına tanış olmaslıdır. Java-dan başqa double formatı, C və C++ və demək olar ki, bütün müasir proqramlaşdırma dillərində mövcuddur.

ədəd literalını dəstəkləyir. Bu haqda sonrakı bölmələrdə bəhs olunur. Diqqətli olun: istənilən mənfi ədəd literalı əvvəlində "mənfi" işarəsi qoyulur. Ancaq faktiki olaraq mənfi (minus) işarəsinin dəyişməsi ədəd literallarının sintaksis hissəsi olmayan unar operatorunu təşkil edir (5-ci fəsilə baxmaq) .

3.1.1. Tam literallar

JavaScript-də tam onluq ədədlər rəqəm ardıcıllığı ilə yazılır. Məsələn:

```
0
3
10000000
```

JavaScript-in ədəd formatı 9007199254740992 (-2^{53}) dən 9007199254740992 (2^{53}) qədər olan ədəd diapazonunda bütün tam ədədləri dəqiq təqdim etməyə imkan verir. Bu diapazondan kənar tam qiymətlərin kiçik dərəcədə dəqiqliyi itə bilər. Qeyd etmək lazımdır ki, JavaScript-də (xüsusən bit operatorları, 5-ci fəsildə təsvir edilmişdir) bəzi tam ədəd əməliyyatları 32-dərəcəli tam ədədlər ilə icra olunur və 2147483648 (-2^{31}) dən 2147483647 ($2^{31}-1$) -ə qədər olan qiymətləri alır.

3.1.2. Onaltılıq və səkkizlik literallar

JavaScript-in onluq tam literallarında başqa onaltılıq say sistemində (əsası 16 olan) olan qiymətləri də tanıyır. Onaltılıq literal "0x" və ya "0X" simvolların ardıcıllığı ilə başlayır. Bu halda, sətir onaltılıq ədəddən təşkil olunur.
Onaltılıq rəqəm – 0- dan 9-a qədər və ya a-dan (və ya A) f-ə qədər (və ya F) qədər hərflərdən ibarət rəqəmdir. Onaltılıq sistemin tam ardıcıllığı (0-F) onluq say sisteminin 0-15 ədədlərinə uyğundur.
Aşağıda onaltılıq tam ədəd literallarının nümunələri göstərilib:

```
0xff // 15*16 + 15 = 255 (10 əsaslı)
0xCAFE911
```

Hərçənd ECMAScript standartı səkkizlik say sistemində tam ədəd literallarını dəstəkləmir, yalnız JavaScript-in bəzi reallaşdırmaları bu tip məsələləri həll etməyə icazə verir. Səkkizlik literallar 0-7 aralığında olur və digər ədədlər bu rəqəmlərin vasitəsilə düzəlir.
Məsələn:

```
0377 // 3*64 + 7*8 + 7 = 255 (10)
```

Bəzi reallaşdırmalar səkkizlik say sisteminin literallarını dəstəklədiyi halda, bəziləri dəstəkləmir.

3.1.3. Həqiqi ədəd literalları

Həqiqi ədəd literalları onluq say sisteminə malik olmalıdır; bu literallar həqiqi ədədlərin ənənəvi

sintaksisindən ibarətdir. Tam qiymət tam ədədin, onluq kəsrini və ya qalıqlı ədədin tam hissəsi müəyyən edir.
Həqiqi ədədlərin literalları həmçinin səciyyəvi nəsihətdə təqdim edilə bilər: həqiqi ədəd, e ədədini (və ya E), plyus və ya minus və tam eksponenti dəstəkləyir. Bu nəsihət 10-da qiymətlə (mənayla) müəyyən edilən dərəcələrə vurulmuş (artırılmış) həqiqi ədədi ifadə edir eksponentlər. Belə sintaksisin daha yığcam təyini:

```
[rəqəmlər] [.rəqəmlər] [(E|e) [(+|-)] rəqəmlər]
```

Məsələn:

```
3.14
2345.789
.333333333333333333
6.02e23 // 6.02 X 10^23
1.4738223E-32 // 1.4738223 X 10^-32
```

Diqqətli olun: həqiqi ədədlər sonsuz qiymətə malikdir, lakin JavaScript-də həqiqi ədədlər məhdudlaşdırılmışdır (daha dəqiq 18437736874454810627) və yalnız məhdudlaşdırılmış həqiqi ədədlər dəqiq ifadə edilir. Deməli, JavaScript-də həqiqi ədədlərlə işləyərkən ədədin təqdim edilməsi zamanı həqiqi ədədləri yuvarlaqlaşdırılması ola bilər. Yuvarlaqlaşdırma dəqiqliyi, kifayət qədərdir və təcrübələrdə nadir hallarda xətalara gətirib çıxarır.

3.1.4. Ədədlərlə iş

JavaScript-dilində proqramlarda ədədlərlə işləmək üçün hesab operatorlarından istifadə edilir. Əsas hesab operatorları, toplama, çıxma, vurma, bölmə operatorlarından ibarətdir. Bu və digər hesab operatorlarının ətraflı təsviri 5-ci fəsildə verilib.
JavaScript-in göstərilmiş əsas hesab operatorlarından başqa daha mürəkkəb riyazi əməliyyatların icra edilməsinə kömək edən və dilin baza hissəsinə aid olan böyük miqdarda riyazi funksiya aiddir. Rahatlıq üçün bu funksiyalar Math obyektinin xüsusiyyətləri şəklində saxlanılır və funksiyalara giriş üçün həmişə Math hərfi adından istifadə olunur.
Məsələn, sinus x dəyişəninin ədədi qiymətini aşağıdakı qaydada hesablamaq olar:

```
sin_of_x = Math.sin (x);
```

Ədədin kvadrat kökü isə belə hesablanır:

```
hypot = Math.sqrt (x*x + y*y);
```

JavaScript-də dəstəklənən bütün riyazi funksiyalar, həmçinin Math obyekti haqqında kitabın üçüncü hissəsində geniş məlumat verilir

3.1.5. Ədəd dəyişiklikləri

JavaScript dilində ədədləri sətirlər şəklində təqdim etmək və ədədləri sətirlərə dəyişdirmək olar. Bu dəyişikliklərin sırası 3.2 paraqrafında təsvir edilir.

3.1.6. Xüsusi ədəd qiymətləri

JavaScript-də bir neçə xüsusi ədəd qiyməti müəyyən edilmişdir. Məsələn, təsəvvür edilən ən böyük həqiqi ədəd, qiymət diapazonunu aşır və nəticəyə JavaScript-də Infinity kimi bilinən xüsusi sonsuzluq qiyməti mənimsənilir. Bu bir qayda olaraq, müsbət və mənfi sonsuzluğa müvafiq olaraq Infinity və -Infinity kimi təqdim edilir.
JavaScriptin aldığı daha bir xüsusi ədəd qiyməti NaN qiymətidir. Bu qiymət riyazi əməliyyat qeyri-müəyyən nəticə aldıqda və xətalı olduqda (məsələn, sıfırın sıfıra bölünməsi, tangens 90 dərəcənin qiyməti) nəticəyə mənimsədilir. NaN qiyməti "ədəd-deyil" xüsusi qiymətinin nəticədir. "Ədəd-deyil" (Not-a-Number) qeyri-adi xüsusiyyətlərə malikdir: o heç bir ədədə hətta özünə belə bərabər deyil! Bu qiymətın yoxlanılması xüsusi isNaN() funksiyası mövcuddur. Oxşar funksiya olan, isFinite(), ədədi, NaN və ya müsbət/mənfi sonsuzluq bərabərsizliyində yoxlamağa imkan verir.
Cədvəl. 3.1 JavaScript-də xüsusi ədəd qiymətləri üçün müəyyən edilmiş bir neçə işarə göstərilmişdir.

Cədvəl 3.1. Xüsusi ədəd sabitləri

Sabit	Qiyməti
Infinity	Sonsuzluğu ifadə edən XÜSUSİ qiymət
NaN	"Ədəd-deyil" XÜSUSİ qiyməti
Number.MAX_VALUE	TƏSƏVVÜR edilən maksimal qiymət
Number.MIN_VALUE	TƏSƏVVÜR edilən minimal qiymət
Number.NaN	"Ədəd-deyil" XÜSUSİ qiyməti
Number. POSITIVE_ INFINITY	MÜSBƏT sonsuzluğu ifadə edən XÜSUSİ qiymət
Number.NEGATIVE_ INFINITY	Mənfi sonsuzluğu ifadə edən XÜSUSİ qiymət

ECMAScript v1-də müəyyən edilmiş JavaScript 1.3 versiyasına qədər Infinity və NaN sabitləri yoxdur. Ancaq Number tipli müxtəlif sabitlər JavaScript 1.1-dən başlayaraq mövcuddur.

3.2. Sətirlər

Sətir, JavaScript-də hərf, rəqəm, punktuasiya nişanları və digər Unicode-simvollar ardıcıllıqlarını və həmçinin mətnin daxiledilməsinə imkan verən məlumat tipidir. Siz növbəti dərslərdə görəcəksiniz ki, sətir literalları cüt və ya tək dırnaq (apostrof) simvolları arasında olur.
Diqqətli olun: C, C++ və Java-da simvolik məlumat tipi kimi bilinən char tipi JavaScript-də yoxdur. Tək simvol tək uzunluğa malik sətir ilə təqdim edilə bilər.

3.2.1. Sətir literalları

Sətir literalı – tək və ya ikiqat dırnaqlarla (' və ya ") əhatə olunmuş Unicode- simvolları ardıcıllığıdır. Əgər mətndə ikiqat dırnaqlardan istifadə etmək lazımdırsa literalı tək dırnaqlarla (') əhatələmək (içərisində yazmaq) lazımdır. Eyni qaydanın əksi olaraq, əgər mətndə tək dırnaqlardan (apostrof) istifadə etmək lazımdırsa literalı cüt dırnaqlarla (") əhatələmək (içərisində yazmaq) lazımdır. Sətir literalı proqramın bir sətirində yazılmalıdır və ikinci sətirə keçirilmir. Sətir literalında yeni sətir daxil etmək üçün \n simvol ardıcıllığından istifadə etmək lazımdır.
Sətir literallarına aid nümunələr:

```
"" // Boş sətir: burada simvol sayı sıfırdır və sadəcə 'testing' edirik
"3.14"
```

```
'name=" myform" ▪
"Siz O'Reilly nəşriyyatının kitablarına üstünlük verirsiniz, elə deyilmi?"
"Bu sətir literalı\niki sətirə malikdir"
"π – dairənin onun diametrinə olan nisbəti"
```

Sonuncu sətir nümunəsində göründüyü kimi ECMAScript v1 standartı sətir literallarında Unicode-simvollarından istifadə etməyə imkan verir. Ancaq bu JavaScript 1.3-dən erkən versiyalarda dəstəklənmir və bu versiyalarda sətirlər adətən yalnız ASCII və ya Latin-1 dəstində olan simvolları dəstəkləyir.

Nəzərə alın ki, tək dırnaq ilə məhdudlaşdırmiş sətir ilə ehtiyatlı davranmaq lazımdır. Kiçik bir yiyəlik halı şəkilçisi (can't, O'Reilly kimi) olan apostrofun düzgün daxil edilməməsi nəticəsində xəta baş verə bilər. Çünki apostrof ilə tək dırnaq işarəsi eyni bir simvoldur və istisna halı metodu ilə əks sleş simvolu vasitəsilə istifadə edilməlidir.

JavaScript kliyentində proqramlar adətən HTML-koddan ibarət ola bilər və öz növbəsində HTML-kodda, JavaScript-kodunun sətirlərini ehtiva edə bilər. Buna görə də JavaScript-i HTML teqlərə və hadisələrə tətbiq edərkən JavaScript kodu dırnaq içərisində yazılır. Aşağıdakı nümunədə JavaScript-ifadəsi kimi "Təşəkkür" sətiri tək dırnaqlarla əhatələnmişdir. Kodun özü isə HTML atributunun hadisə emalçısına mənimsədildiyinə görə cüt dırnaqlar ilə əhatələnmişdir:

```
<a href="" onclick=" alert ('Təşəkkürlər')">Məni klikləyin!</a>
```

3.2.2. Sətir literallarında idarəedici ardıcıllıqlar

Əks sleş (\) simvolu JavaScript-sətirlərində xüsusi təyinata malikdir. Onu yanında gələn simvollarla birlikdə, müvafiq simvol ifadə edir. Adətən sətirin daxilində yerləşdirilməsi mümkün olmayan simvolların sətirə mənimsədilməsi üçün istifadə edilir. Məsələn, \n – *yeni sətirə keçmək üçün istifadə olunur (escape sequence)*.[3]

Əvvəlki bölmədə qeyd olunmuş başqa nümunə – tək dırnaq simvolunu ifadə edən \' simvol birləşməsidir. Bu idarəedici sətir ardıcıllığı tək dırnaq ilə əhatələnmiş sətir sahəsinə tək dırnağın simvolunun əlavə edilməsi üçündür. İndi bizə aydın olur ki, niyə bu ardıcıllıqları idarə edici adlandırırıq. Burada əks sleş simvolu tək dırnaq simvolunun interpretasiyasını idarə etməyə imkan verir. Aşağıdakı nümunədə biz tək dırnaq işarəsinin əks sləş simvolu ilə tətbiqi nəticəsində biz bu işarəni sətir sonluğu kimi deyil, apostrof kimi veririk:

```
'You\'re right, it can\'t be a quote'
```

Cədvəl. 3.2-də idarəedici ardıcıllıqlar və onların tərəfindən ifadə edilən simvollar göstərilmişdir. İki idarə edici ardıcıllıq ümumiləşdirilmişdir; onlar əks sləşin yanında onaltılıq say sistemində olan kodu göstərməklə Latin-1 və ya Unicode simvol dəstinin tərkibində olan istənilən simvolun göstərilməsi üçün tətbiq edilə bilər. Məsələn, \xA9 ardıcıllığı müəllif hüquqları simvolunu ifadə edir. Bu simvol, Latin-1 kodlaşdırmasında A9 onaltılıq koduna malikdir. \u simvolları da analoji xüsusiyyətə malikdir və dörd onaltılıq rəqəmlə göstərilmiş sərbəst Unicode simvolunu ifadə edir. Məsələn, \u03c0 simvolu π ifadə edir. Qeyd etmək lazımdır ki, idarəedici ardıcıllıqlarında ECMAScript v1 standartı üzrə Unicode-simvol dəsti tələb olunur. JavaScript 1.3-dən daha əvvəl çıxmış versiyalarda bu xüsusiyyət dəstəklənmirdi.

Cədvəl 3.2. JavaScript-in idarə edici ardıcıllıqları

Sabit	Qiyməti
\0	NUL simvolu (\u0000)

[3] 1 C, C++ və Java-da proqramlaşdırmağı bacaran şəxslərə, JavaScript-in bu və başqa idarəedici ardıcıllıqları artıq tanış olmalıdır.

\b	"Quyu dibi" simvolu (\u0008)
\t	ÜFÜQİ tabulyasiya (\u0009)
\n	Yeni sətirə keçid (\u000A)
\v	Şaquli tabulyasiya (\u000B)
\f	Yeni səhifəyə keçid (\u000C)
\r	Karetin qaytarılması (\u000D)
\"	İkiqat dırnaq (\u0022)
\'	Tək dırnaq (\u0027)
\\	Əks sleş (\u005C)
\xXX	Latin-1 simvol dəstinə aid edilən, onaltılıq say sistemində verilən iki XX ədədi
\uXXXX	Unicode simvol dəstinə aid edilən, onaltılıq say sistemində verilən dörd XXXX ədədi
\XXX	Latin-1 simvol dəstinə aid edilən, səkkizlik say sistemində verilən ÜÇ XXX ədədi. 1-dən 377-ə qədər kod diapozununa malikdir. ECMAScript v3 dəstəklənmir; burada belə yazı ÜSULU istifadə olunmamalıdır.

3.2.3. Sətirlərlə iş

JavaScript-in inteqrasiya edilmiş imkanlarından biri də sətirləri birləşdirməkdir. Əgər + operatoru ədədlərə tətbiq edilirsə, bu ədədlər toplanır, əgər sətirlərə tətbiq edilirsə, bu sətirlər bitişdirilir, bu halda ikinci birinci sətirin sonuna əlavə edilir. Məsələn:

```
msg = "Hello, "+ "world"; // Alınır: "Hello, world"
greeting = "Hörmətli, "+name+". Mənim ana səhifəmə xoş gəlmisiniz";
```

Sətir uzunluğunun (sətirdə olan simvolların miqdarı) təyini üçün length xüsusiyyətindən istifadə olunur. Məsələn, əgər s dəyişəni sətiri tipindədirsə, bu sətirin uzunluğunu aşağıdakı qaydada göstərmək olar:

```
s.length
```

Sətirlərlə işləmək üçün bir neçə metod mövcuddur. s sətirinin son simvolu belə göstərmək olar:

```
last_char = s.charAt(s.length – 1)
```

İkinci simvolu çıxartmaq üçün, s sətirinin üçüncü və dördüncü simvolları, təlimata tətbiq edilir:

```
sub = s.substring (1,4);
```

S sətirində birinci simvol olan "a" hərfinin mövqeyini aşağıdakı qaydada müəyyən etmək olar:

```
i = s.indexOf ('a');
```

Sətirlərlə işləyən digər metodlar da vardır. Bu metodların təffərrüatı String obyektinin təsvirində və kitabın üçüncü hissəsinin listinqlərində sənədlərlə əsaslandırılmışdır. Əvvəlki nümunələrdən anlamaq olar ki, JavaScript-sətirləri (və gələcəkdə tanış olacağımız JavaScript massivləri) 0-dan başlayaraq indeksləşdirilir. Başqa sözlə, sətirin birinci simvolunun sıra nömrəsi sıfıra bərabərdir. C, C++ və Java-da işləmiş proqramçılara bu metod tanışdır. Bir məqam da var ki, bu dillərdə sətirlərin və massivlərin numerasiyası vahiddən başlanır. JavaScript-in bəzi (bir qədər) reallaşdırmalarında ayrı simvollar götürülə bilər sətirlər (amma sətirlərə yazılmamaq) sətirlərə müraciət (rəftar) vaxtı (yanında) necə massivlərə, daha əvvəl nəticədə göstərilən charAt() metodunun çağırışı aşağıdakı

qaydada yazılmış ola bilər:

```
last_char = s [s.length – 1];
```

Ancaq bu sintaksis ECMAScript v3-də standartlaşdırılmamışdır, daşınan deyil və ondan çəkinmək lazımdır.
Biz obyekt məlumat tipi müzakirə edəndə, xüsusiyyətin əvvəlki nümunələri və sətirlər metodları obyektlər metodları kimi istifadə olunur. Bu o demək deyil ki, sətirlər – obyektlərin tipidir. Əslində sətirlər JavaScript-in ayrı bir məlumat tipidir. Onların xüsusiyyətlərinə və metodlarına giriş üçün obyekt sintaksisi istifadə olunur, amma sətir özü heç vaxt obyekt olmur! Bunun niyə belə olduğunu, biz bu fəsilin sonunda biləcəyik.

3.2.5. Ədəd-sətir dəyişikliyi

Sətir ədəd kontekstində istifadə olunduqda, avtomatik olaraq ədədə dəyişdiriləcək. Məsələn, aşağıdakı ifadə tamamilə mümkündür:

```
var product = "21" * "2"; // nəticədə 42 ədədi alınacaq.
```

Bu şəraitin əksi zəruri olduqda ədədi sətiri dəyişdirmək olar; bunun üçün kifayət qədər sadə metod, sətirdən 0 qiymətini çıxmaq lazımdır:

```
var number = string_value - 0;
```

(Diqqətli olun: bu vəziyyətdə toplama əməliyyatı sətirlərin bitişdirilməsi əməliyyatı kimi yerinə yetiriləcək.)
Ədəd sətir dəyişikliyinin daha az inkişaf etmiş və daha düzxətli üsulu Number() funksiya konstruktoruna müraciət ilə icra edilir:

```
var number = Number (string_value);
```

Ədəd-sətir dəyişikliyinin belə üsulunun çatışmazlığı ondan ibarətdir ki, bu qədər sadə əməliyyatı həddən artıq ciddi üsulla yerinə yetirir. Bu üsul yalnız onluq say sistemində oluna bilər və bu üsul yalnız rəqəm mövcudluğunu güman edərək, boşluq simvolları, sətirdə ədəddən sonra gələn başqa qeyri-rəqəmsal simvolların yaranmasına icazə verir. Dəyişikliyin daha elastik üsulu parseInt() və parseFloat()funksiyalarının köməyilə təmin olunur. Bu funksiyalar istənilən qeyri-rəqəmsal simvollara məhəl qoymadan sətirin başlanğıcında duran sərbəst ədədləri dəyişdirəcək və ədədin ardınca yerləşdirilmiş simvollar qaytarılacaq. parseInt() funksiyası yalnız tam ədəd dəyişikliyini yerinə yetirir. parseFloat()funksiyası isə həm tam, həm də həqiqi ədədlər ilə ədəd dəyişikliyini yerinə yetirə bilər. Əgər sətir "0x" və ya "0x" simvollarından başlayırsa, parseInt() funksiyası sətiri onaltılıq ədəd kimi göstərir. Məsələn:

```
parseInt ("3 dovşan"); // 3 qiymətini alır
parseFloat ("3.14 metr") ; // 3.14 qiymətini alır
parseInt (" 12.34"); // 12 qiymətini alır
parseInt ("0xFF"); // 255 qiymətini alır
```

İkinci arqument kimi parseInt() funksiyası hesablama sistemlərini də qəbul edə bilər. 2-dən 36-a qədər olan ədəd diapozunda düzgün alınır[4], məsələn:

```
parseInt ("11", 2); // 3 (1*2 + 1) qiymətini alır
parseInt ("ff", 16); // 255 qiymətini alır (15*16 + 15)
parseInt ("zz", 36); // 1295 qiymətini alır (35*36 + 35)
parseInt (" 077", 8); // 63 (7*8 + 7) Qaytaracaq
parseInt (" 077", 10); // 77 (7*10 + 7) Qaytaracaq
```

Əgər parseInt() və parseFloat() metodlarında dəyişliyi yerinə yetirmək mümkün deyilsə, onlar NaN qiymətini alır:

```
parseInt ("eleven"); // NaN qiymətini
parseFloat (" $72.47"); // NaN qiymətini alacaq.
```

3.3. Məntiqi qiymətlər

Ədəd və sətir məlumat tipləri böyük və ya sonsuz mümkün qiymətlər miqdarına malikdir. Məntiqi məlumat tipi isə, əksinə, yalnız iki true və false literalları ilə təqdim edilmiş mümkün məntiqi qiymətlərdən ibarətdir.

Məntiqi qiymət, doğruluq ətrafında icra edilir. JavaScript-proqramlarda yerinə yetirilən müqayisələrin nəticələri adətən məntiqi qiymətlərdir. Məsələn:

```
a == 4
```

Bu ifadə a dəyişəninin 4 ədədinə bərabər olmasını yoxluyur. Əgər a dəyişən həqiqətən də dördə bərabərdirsə, true məntiqi qiymətinin şərti ödənilir. Əgər a dəyişəni dördə bərabər deyilsə, müqayisənin nəticəsi false olacaq.

JavaScript-də məntiqi qiymətlər adətən idarəedici konstruksiyalarda istifadə olunur. Məsələn, JavaScript-də if/else təlimatı hər hansı bir arqumenti yerinə yetirir, əgər məntiqi qiymət true-a bərabər olarsa, əməliyyat yerinə yetirilir, əks halda false qiyməti alınır. Adətən məntiqi qiyməti yaradan müqayisə bilavasitə istifadə olunan təlimatla birləşir. Bu sözü kodla ifadə edək:

```
if (a == 4)
b = b + 1;
else
  a = a + 1;
```

Burada yoxlama yerinə yetirilir, əgər a dəyişəni 4 ədədinə bərabərdirsə b dəyişəninin qiymətinə bir vahid əlavə edilir; əks təqdirdə a dəyişəninin qiymətinə bir vahid əlavə edilir.

İki mümkün true və false məntiqi qiymətlərinə, bəzən "düzdür" (true) və ya "səhvdir" (false) və ya "bəli" (true) və "xeyr" (false) kimi baxılır.

3.3.1. Məntiqi qiymətlərin dəyişikliyi

Məntiqi qiymətlər başqa tiplərin qiymətlərinə asan dəyişdirilir, həm də bu tip dəyişikliklər əksər hallarda avtomatik yerinə yetirilir[5]. Əgər məntiq ədəd kontekstində istifadə olunur. true qiyməti 1-

[5] C-də proqramlaşdırmağı bacaran şəxs diqqətli olmalıdır ki, JavaScript-də C-dən fərqli olaraq ayrı bir məntiqi məlumat tipi mövcuddur, hansına ki, məntiqi qiymətlərin tam ədədlərə təqlidinə xidmət edir. Java-proqramçıları isə nəzərə almalıdırlar ki, JavaScript-də məntiqi tip, Java-dakı kimi boolean məlumat tipi kimi "təmiz deyil və JavaScript-dəki məntiqi qiymətlər digər məlumat tiplərində asan dəyişdirilir və buna görə də söyləmək olar ki, JavaScript-dəki məntiqi qiymətlər ilə iş praktiki olaraq Java-ya nisbətən C dilinə yaxındır.

ədədinə, false qiyməti isə 0 ədədinə dəyişdiriləcək. Əgər məntiqi qiymət sətir kontekstində istifadə olunursa, onda true qiyməti "true" sətirinə , false qiyməti isə "false" sətirinə dəyişdiriləcək.
Məntiqi qiymət birdən yuxarı ədəd kimi istifadə olunduqda, true qiymətinə dəyişdiriləcək. Əgər qiymətlər 0-a və ya NaN-a bərabərdirsə, false məntiqi qiymətinə dəyişdiriləcək. Məntiqi qiymətlərdə sətirlərdə istifadə edildikdə, əgər sətir boş sətir deyilsə, true qiymətinə dəyişdiriləcək, əks təqdirdə dəyişiklik nəticəsində false qiyməti alınacaq. Null və undefined xüsusi qiymətləri false qiymətinə dəyişdiriləcək və istənilən funksiya, obyekt və ya massivin qiymətləri null-dan böyükdür və true qiymətinə dəyişdiriləcəklər.
Əgər siz dəyişikliyi açıq- aydın yerinə yetirmək istəyirsinizsə, Boolean() funksiyasından istifadə etmək olar:

```
var x_as_boolean = Boolean (x);
```

Açıq dəyişikliyin başqa üsulu ikiqat məntiqi inkarın operatorunun istifadəsi ilə mümkündür:

```
var x_as_boolean =!! x; 3.4.
```

3.4. Funksiyalar

Funksiya – icra edilən kodun fraqmentidir, hansı ki, JavaScript-proqramında və ya JavaScript reallaşdırmasında qabaqcadan müəyyən edilmişdir. Funksiya JavaScript- proqramında yalnız bir dəfə təyin edilir, lakin istənilən icra edilə və ya səbəb çağırıla bilər Funksiyalar arqumentləri, qiyməti və ya qiymətləri müəyyən edən və hesablamaları yerinə yetirməli olan parametrləri ötürə bilir; həmçinin funksiya bu hesablamaların nəticəsini təşkil edən qiyməti qaytara bilər. JavaScript reallaşdırmaları bir çox qabaqcadan müəyyən edilmiş funksiya təklif edir. Bunlara misal olaraq, bucağın sinusunu hesablayan Math.sin() funksiyasını göstərmək olar.
JavaScript, proqramların ehtiva etdiyi şəxsi funksiyaları da müəyyən edə bilər məsələn, belə bir kod:

```
function square (x) //  Funksiya square adlanır. O bir x arqumentini qəbul edir.
{                   //  Burada funksiyanın əsası başlanır.
return x*x;         //  Funksiya öz arqumentini kvadrata yüksəldir və alınmış
                        qiymətə qayıdır
}                   //Burada funksiya bitir.
```

Bir çox dillərdə, həmçinin Java-da funksiyalar – dilin məlumat tipi kimi deyil, yalnız sintaktis elementləridir: funksiyaları müəyyən etmək və çağırmaq olar. O halda ki, JavaScript-də funksiyalar əsl qiymətlər (təşkil edir, dilə böyüyü verir elastiklik. Bu bildirir ki, funksiyalar dəyişənlərdə saxlanıla bilər, massivlər və obyektlər, həmçinin arqumentlər kimi başqa funksiyalara ötürülmək. Çox tez-tez bu çox rahat olur. Funksiyaların çağırışı və onlardan istifadə etmək üsullarında 8-ci danışılır.
Bir halda ki, funksiyalar ədəd və sətirlər kimi qiymətlərdən təşkil olunub, onlara obyekt xüsusiyyətlərinə mənimsədilə bilər. Funksiyaya obyekt xüsusiyyətinə mənimsədildikdə (obyekt məlumat tipi və obyektin xüsusiyyətləri 3.5 saylı paraqrafda təsvir edilmişdir), bu obyekt metodu adlanır. Metodlar – *obyekt yönümlü proqramlaşdırmanın* mühüm hissəsidir. Məhz, 7-ci fəsil *OYP-yə* həsr edilmişdir.

3.4.1. Funksional literallar

Əvvəlki bölmədə biz square() funksiyasının təyinini gördük. Adətən JavaScript- proqramlarında funksiyaların əksəriyyəti bu sintaksis ilə təsvir edilir. Ancaq ECMAScript v3 standartı funksional literalların təyini üçün sintaksisə (JavaScript 1.2-də və daha gec versiyalarda reallaşdırılmış) malikdir. Funksional literal function açar sözünün köməyi ilə, funksiyanın adı ilə birlikdə verilir. Funksiyanın adının qarşısında duran mötərizələrin içərisində bu funksiyanın malik olduğu arqumentlərin siyahısı göstərilir. Funksiya fiqurlu mötərizələr ilə başlayıb, fiqurlu mötərizələr ilə bitir. Funksional literal ilə təyin olun funksiyalara ad verilməyə bilər. Ən böyük fərq ondan ibarətdir ki, funksional literallar digər JavaScript- ifadələri kimi yazıla bilər. Yəni, square() bu şəkildə funksiyasını təyin etmək vacib deyil:

```
function square (x)
{
  return x*x;
}
```

Bunu literalların köməyi ilə aşağıdakı kimi də etmək olar:

```
var square = function (x)
{
  return x*x;
}
```

Belə, müəyyən edilmiş funksiyalar adətən lyambda-funksiya adlandırılır. Bu, üsul ilk dəfə LISP proqramlaşdırma dilində istifadə edilmişdir. Elementar səviyyədə siz bu literalların xeyirini görməsənizdə, mürəkkəb ssenarilərdə siz görəcəksiniz ki, onlar kifayət qədər rahat və faydalıdır.
Funksiyanın təyininin daha bir üsulu mövcuddur: arqumentlərin siyahısını və funksiyanın əsasını Function()-konstrukturunda sətirlər şəklində vermək olar. Məsələn:

```
var square = new Function (" x", "return x*x;");
```

Funksiyaların bu cür təyindən nadir hallarda istifadə olunur. Adətən burada əsas hissəsini sətir şəklində vermək narahatdır və oxşar tərzdə müəyyən edilmiş funksiyalar yuxarı sadalanan iki üsul ilə daha azeffektivdir.