

JavaScript nədir?

JavaScript – internet səhifələrin yaradılmasında geniş istifadə olunan [proqramlaşdırma dilidir](#). JavaScript müxtəlif dillərdən ilhamlanaraq yaradılmışdır. 1995-ci ilə [Brendan Eyx](#) tərəfindən yazılmışdır. JavaScript [Obyekt Yönlü Proqramlaşdırma](#) (OYP) dilidir. Obyektə Yönlü Proqramlaşdırma mövzusu kifayət qədər mürəkkəb proqramlaşdırma deyil. JavaScript Netscape Navigator 2.0 ilə birlikdə Brendan Eich tərəfindən inkişaf etdirilən və əvvəllər Mocha sonralar LiveScript olaraq adlandırılan və sonda bu anki adını alan JavaScript dili başlanğıcda sadəcə müştəri tərəfindən (client-side) şərh edilən bir proqramlaşdırma dilidir.

Günümüzdə NodeJS texnologiyası ilə server tərəfli də (server-side) şərh edilən proqramlama dili halına gəldi. 1995-ci ildə [Netscape](#) şirkəti tərəfindən, Sun şirkətinin dəstəyi ilə hazırlanmış NN24-də istifadə olunan AD bundan sonra JavaScript adlandırılması ilə açıqlanma verdi. Buna qədər isə bu alqoritmik dil LiveScript adlandırılırdı. Bu gözlənilməz addım kifayət qədər problemlər yaratdı. Beləki, Sun şirkətinin JAVA AD-nə heç bir dəxli olmayan JavaScript-i JAVA-nın alt-çoxluğu kimi qəbul edənlərin sayı çoxaldı.

JavaScript - HTML səhifəyə integrasiya olunaraq, bu səhifə ilə istifadəçi arasındakı interfeysə əlavə funksional imkanlar verən alqoritmik dildir.

Node Js

Node.js asan sürətli, öncələmə bilən şəbəkə tətbiqləri yaratmaq üçün Chrome's JavaScript Runtime texnologiyası üzərində qurulmuş bir platformadır. Node.js dağıdılmış cihazlar üzərindən işləyə bilən gerçək zamanlı tətbiq etmələr üçün mükəmməl yüngül və səmərəli hala gətirən hadisə yönümlü, əngəllənməyən I/O modeli istifadə edir.

Nümunə

Əks olunduğu HTML sənədin **body** hissəsi:

```
<body>
  <form name="form" method="post" action="">
    <textarea name="latin" cols="59" rows="25" id="latin"></textarea>
    <textarea name="netice" cols="58" rows="25" id="netice"></textarea>
  </form>
  <input name="Submit" type="submit" value="Çevir" onkeyup =
"latinkiril()">
</body>
```

JavaScript in-Browser də nə edə bilər?

Brauzerdə JavaScript "təhlükəsiz" dildir, yəni müəyyən imkanları məhdudlaşdırmaqla istifadəçi təhlükəsizliyinə üstünlük verir. Onun gücü fəaliyyət göstərdiyi mühitdən asılıdır. In-browser JavaScript

əsasən veb-səhifə manipulyasiyası, istifadəçi qarşılıqlı əlaqəsi və veb-serverlərlə ünsiyyət üçün istifadə olunur.

```
DOM manipulyasiya
sənədi. getElementById('myButton'). addEventListener('click', function() {
  document.getElementById('output'). textContent = 'Button clicked!';
});

Şəbəkə sifarişi
fetch('https://api.example.com/data')
  . sonra(cavab => cavab.Json())
  . then(data => konsol.log(data));
```

İstifadəçi təhlükəsizliyini təmin etmək üçün brauzerdə JavaScript bəzi məhdudiyyətlərə malikdir. Sabit diskdə arbitr fayllara daxil ola bilmir, müxtəlif domenlərdən olan digər brauzer tablaları/windows ilə qarşılıqlı əlaqədə ola və ya uzaq serverlərlə sərbəst şəkildə ünsiyyət qura bilmir. Bu cür əməliyyatlar üçün istifadəçidən və ya konkret HTTP başlıqlarından icazə tələb olunur.

Note: JavaScript hələ də istifadəçilərin məlumatlarını və gizliliyini qoruyarkən bir çox güclü tapşırıqları yerinə yetirə bilər.

JavaScript-in özünəməxsus olması nədən xəbər verir?

JavaScript üç inandırıcı səbəbə görə fərqlənir:

1. **HTML/CSS ilə tam inteqrasiya.** JavaScript HTML və CSS ilə sıx şəkildə inteqrasiya edir, bu isə interaktiv veb interfeyslərin yaradılması üçün mükəmməl vasitə edir.

2. **Sadəlik:** JavaScript, developersin açıq və konkisüre kodu ilə böyük nəticələr əldə etməsinə imkan yaradır.
3. **Geniş yayılmış dəstək:** JavaScript bütün əsas brauzerlər tərəfindən dəstəklənir və default imkan verir, geniş auditoriyaya onun əlçatanlığını təmin edir.

VARIABLES

Dəyişənlər JavaScript və ya hər hansı proqramlaşdırma dili ilə proqramlaşdırmanın təməl aspektidir. Dəyişənin nə olduğunu və onlardan istifadə edərkən nə gözləməli olduğunu anlamaq mütləqdir. Ən uzun müddət JavaScript-də, ümumiyyətlə, dəyişənləri bəyan etmək üçün yalnız bir yol var idi, açar sözü ilə. ES6 (ES2015) gəldiyindən açar sözlər və dəyişən bəyannamə üçün mövcuddur. Bu xüsusiyyətləri dilə əlavə etməyin arxasında duran ideya dəyişənlərin necə bəyan edildiyinə nəzarət etmək, dəyərləri təyin etmək və miqyas anlayışı vasitəsilə JavaScript proqramının digər hissələrinə görünən yolları təqdim etmək idi.

var

Dəyişən deklarasiya yarandığı gündən javascript ilə istifadə olunur. Dəyişənlər (həmçinin və dəyişənlər) müxtəlif növ məlumatların saxlanması üçün istifadə olunur. Dəyişən, string, ədəd, booleans, objects, functions və daha çox kimi məlumatları saxlaya

bilər.

```
var myName = 'bob';
var myAge = 34;
var clubMember = true;
var schoolSupplies = ['pencil', 'notebook', 'pen', 'eraser'];
var schoolSchedule = {
  math: 'Advanced Algebra',
  science: 'Physics',
  literature: 'Modern Chinese Literature',
```

```
};
var sayHello = function () {
  console.log('hello');
};
```

Yuxarıdakı nümunədə dəyişənlərdə saxlanıla bilən müxtəlif növ məlumatlar göstərilmişdir. Yuxarıdakı bütün dəyişənlər açar sözü ilə bəyan edilir. Hər hansı digər kod blokları ilə və xaricində bir dəyişənin bəyan edilməsi onu **global miqyas** daxilində yer tutur. Bu o deməkdir ki, dəyişənlərə proqramın istənilən digər hissəsindən daxil olmaq olar və proqram daxilində global olaraq mövcuddur.

```
var myName = 'bob';function studentProfile() {
  console.log(`student name: ${myName}`);
}studentProfile(); // student name: bob - variable 'myName' is part
of the global scope
```

Funksiya kod blokları daxilində istifadə edərək bəyan edilən dəyişənlər - **funksiya miqyası** kimi tanınan - yalnız funksiya daxilində əlçatandır. İstisna hal dəyişənə qiymət təyin edildikdə, lakin aşağıda göstəriləndiyi kimi açar sözdən istifadə edilərək bəyan edilmədikdə. Belə bir qiymətin təyin edilməsi dəyişənə funksiyaadan kənarda daxil olmağa imkan verir.

```
var testResults
function getStudentGrade() {
  testResults = [75, 86, 93, 87];
  var testResultsAverage =
    testResults.reduce((a, b) => a + b) / testResults.length;
  return testResultsAverage;
}console.log(testResults); // 85.25 - variable 'testResults' is
part of the global scope
```

Məntiqi kod blokları daxilində istifadə edərək bəyan edilən dəyişənlər - **blok miqyası** (ex.) kimi tanınan - blokdan kənarda əlçatandır.

```
var if..else
```

```
if (true) {  
  var newName = 'ned';  
}console.log(newName); // ned - variable 'newName' is part of the  
global scope
```

qoy

Dəyişən deklarasiya bir çox cəhətdən oxşardır, lakin giriş və miqyas məsələsində bəzi əsas fərqləri daxil edin. Qlobal miqyas daxilində bəyan edilən dəyişənə proqramın istənilən yerinə daxil olmaq mümkündür.

Bu davranış dəyişən deklarasiya ilə baş verənlərə bənzəyir.

```
letvarletvar  
let studentName = 'melvin';function greetStudent() {  
  console.log(`Greetings, ${studentName}!`);  
}
```

Həmçinin, with ilə olduğu kimi , bir funksiya çərçivəsində bəyan edilən dəyişənlər də yalnız funksiya çərçivəsində əlçatandır və funksiya çərçivəsindən kənarda daxil olmağa cəhd edərkən xəta səbəb olacaq.

```
varlet
```

Tip Çevrilməsi:

Dönüşüm, dəyərləri çevirmədən əvvəl metod və ya funksiya adını qeyd edərək əllə və ya açıq şəkildə baş verir, Explicit Conversion kimi tanınır.

JavaScript-də **String**, **number**, **boolean**, **object** və **function** kimi verilənlərin beş müxtəlif növü mövcuddur.

Type dönüşümü yalnız **String**, **number** və **boolean** olan üç məlumat növündə istifadə olunur.

— To String Conversion :

Bu dönüşdə rəqəm bizə lazım olduqda bir stringə çevrilir.

JavaScript Explicit çevrilməsində istifadə olunan müxtəlif növ built-in üsulları və ya funksiyaları təqdim edir.

1.String()

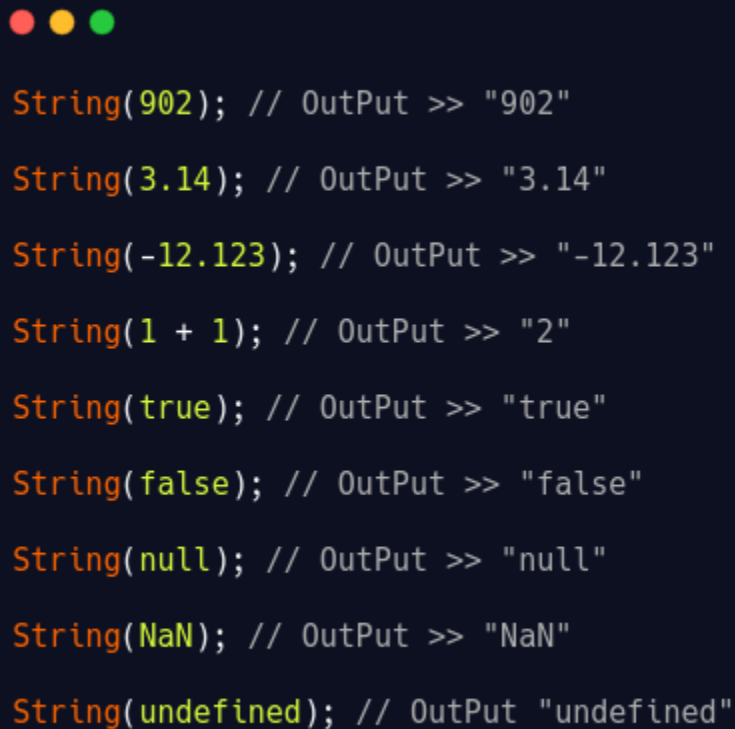
String() metodu Qlobal üsuldur və rəqəmlərin sətirlərə çevrilməsi üçün istifadə olunur.

Həmçinin bu sözdən hərfi və ifadələrin çevrilməsi üçün də istifadə etmək olar.

Sintaksis :

```
String(ValueToConvert)
```

Nümunə:



```
String(902); // OutPut >> "902"
String(3.14); // OutPut >> "3.14"
String(-12.123); // OutPut >> "-12.123"
String(1 + 1); // OutPut >> "2"
String(true); // OutPut >> "true"
String(false); // OutPut >> "false"
String(null); // OutPut >> "null"
String(NaN); // OutPut >> "NaN"
String(undefined); // OutPut "undefined"
```

Yuxarıdakı nümunədə olduğu kimi String() metodu da rəqəmləri gözlədiyimiz kimi bir stringə çevirir.

Biz null, NaN, və undefined kimi keçid edə bilərik ki, bu metoda dəyərlər sonra String() metodu onları string də çevirir, səhv atmadan.

Qeyd:

undefined — onun yaradılması mərhələsində heç bir dəyər təyin olunmayan dəyişəndir.

null - boş və ya başlanğıcda olmayan və ya müəyyən edilməyən və ya sıfır dəyərinə malik olan dəyərdir.

NaN, Not-a-Number olaraq adlandırılır, xətanı hökmlü bir nömrəyə daxil etmək üçün işarə etmək üçün istifadə edilirdi.

2. toString()

String() metodu ilə eyni işi görür.

Sintaksis:

```
number.toString()
```

Nümunə:

```
let num1 = 12345;
num1.toString(); // OutPut >> "12345"

(123).toString(); // OutPut >> "123"

let num2 = NaN;
num2.toString(); // OutPut >> NaN

let num3 = null;
num2.toString(); // OutPut >> TypeError: Cannot read property 'toString' of null

let num4 = undefined;
num3.toString(); // OutPut >> TypeError: Cannot read property 'toString' of undefined
```


İlk öncə, biz qiyməti bəyan edirik və dəyişənə təyin edirik, sonra isə dot(.) operatorundan istifadə edərək toString() metodu adlandırdığımız dəyişən adından istifadə edirik.

Əgər null və müəyyən edilməmiş dəyərləri keçməyə çalışsaq, onda JavaScript mühərriki TypeError-i göstərir.

3. toExponential()

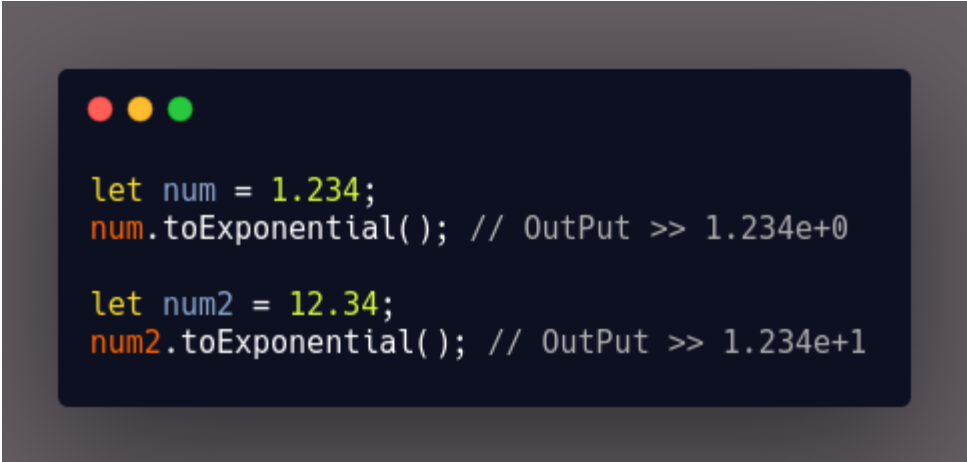
Bu üsuldan ədədi string exponential simvola (e+sayı) çevirmək üçün istifadə etmək olar.

Sintaksis:

```
number.toExponential(OptionalParameter)
```

Parametr onluq nöqtədən sonrakı ədədlərin sayını göstərən seçimlidir.

Nümunə:



```
let num = 1.234;  
num.toExponential(); // OutPut >> 1.234e+0  
  
let num2 = 12.34;  
num2.toExponential(); // OutPut >> 1.234e+1
```

4. toFixed()

JavaScript-də toFixed() üsulundan istifadə olunur ki, sayın onuncu rəqəmin ondan sonra sağa doğru təyin edilməsi ilə ədədi bir stringə çevirsin.

Sintaksis:

```
number.toFixed(optinalParameter)
```

Parametr onluq nöqtədən sonrakı ədədlərin sayını göstərən seçimlidir.

Nümunə:

```
let num = 123.456;  
num.toFixed(); // OutPut >> 123  
  
let num2 = 123.456;  
num2.toFixed(2); // OutPut >> 123.46
```

5. toPrecision()

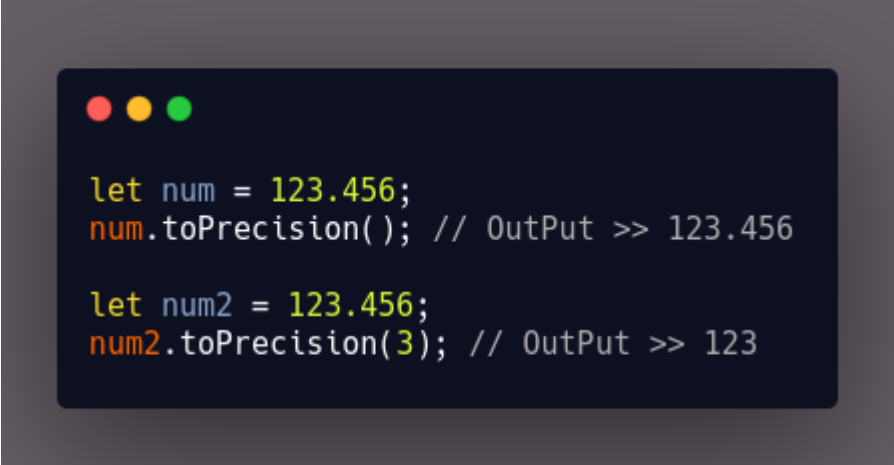
Bu üsulda rəqəm bir stringə çevrilir, ədədi sabit uzunluğa format edir.

Sintaksis:

```
number.toPrecision(optianlParamater)
```

Parametr ədədi rəqəmlərini təmsil edən seçimlidir.

Nümunə :



```
let num = 123.456;  
num.toPrecision(); // OutPut >> 123.456  
  
let num2 = 123.456;  
num2.toPrecision(3); // OutPut >> 123
```

— Sayı Konvertasiyasına :

String to Number JavaScript-də bəzi inşa edilmiş metodlardan istifadə etməklə çevrilir.

İnşa edilmiş metodlar aşağıdakılardır.

1.Nömrə()

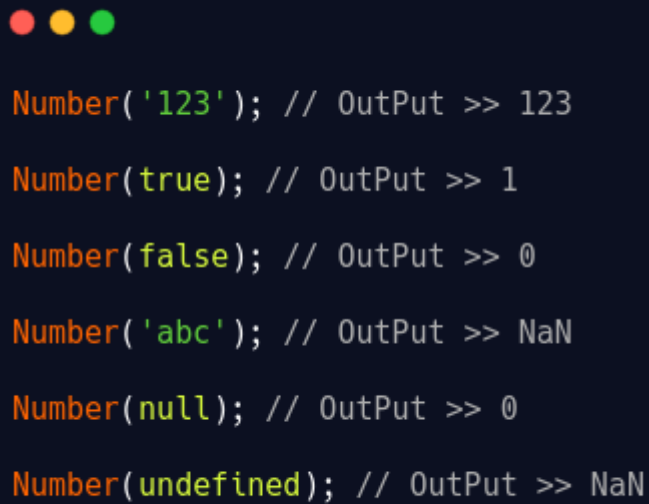
String qiyməti Global number() metodu ilə bir ədədə çevrilir.

Bu, ədədi mətnin, eləcə də boolean dəyərlərinin rəqəmlərə çevrilməsi üçün istifadə oluna bilər.

Sintaksis:

```
Number (valueToConvert)
```

Example:



```
Number('123'); // OutPut >> 123  
Number(true); // OutPut >> 1  
Number(false); // OutPut >> 0  
Number('abc'); // OutPut >> NaN  
Number(null); // OutPut >> 0  
Number(undefined); // OutPut >> NaN
```

Əgər biz qeyri-müəyyən string və undefined value keçirsək, onda nəticə həmişə NaN olaraq çap olunur. Null qiymətin default qiyməti 0-dır.

2. parseInt()

parseInt() metodu əvvəlcə stringi pars edir və ya təhlil edir, sonra isə integer yəni rəqəmə çevrilir.

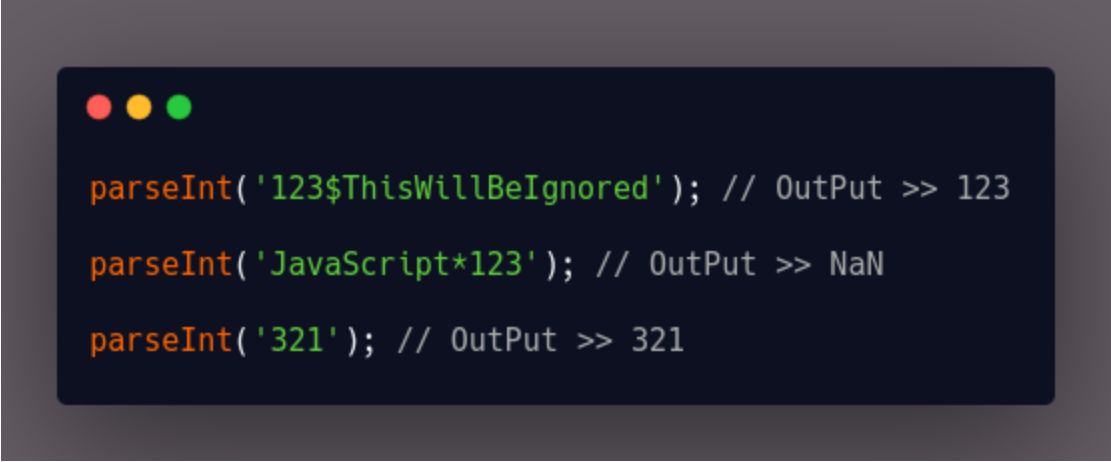
Sintaksis:

```
parseInt(String, Radix)
```

Bu metodda birinci string parametri mütləqdir.

Radiks ədədi say sisteminə, yəni hecalara bölünən rəqəmlərin onun dekadalı saya çevrilməsi üçün istifadə olunur. Bu parametr seçimlidir.

Nümunə:



```
parseInt('123$ThisWillBeIgnored'); // OutPut >> 123
parseInt('JavaScript*123'); // OutPut >> NaN
parseInt('321'); // OutPut >> 321
```

Birinci nümunədə nömrə '123' çap olunacaq, '\$' -dən qalan ip isə nəzərə alınmayacaq.

İkinci nümunədə string qiyməti nömrə ilə başlamalıdır, əks halda bütün string nəzərə alınmayacaq, çıxış isə NaN olaraq çap olunacaq.

3. parseFloat()

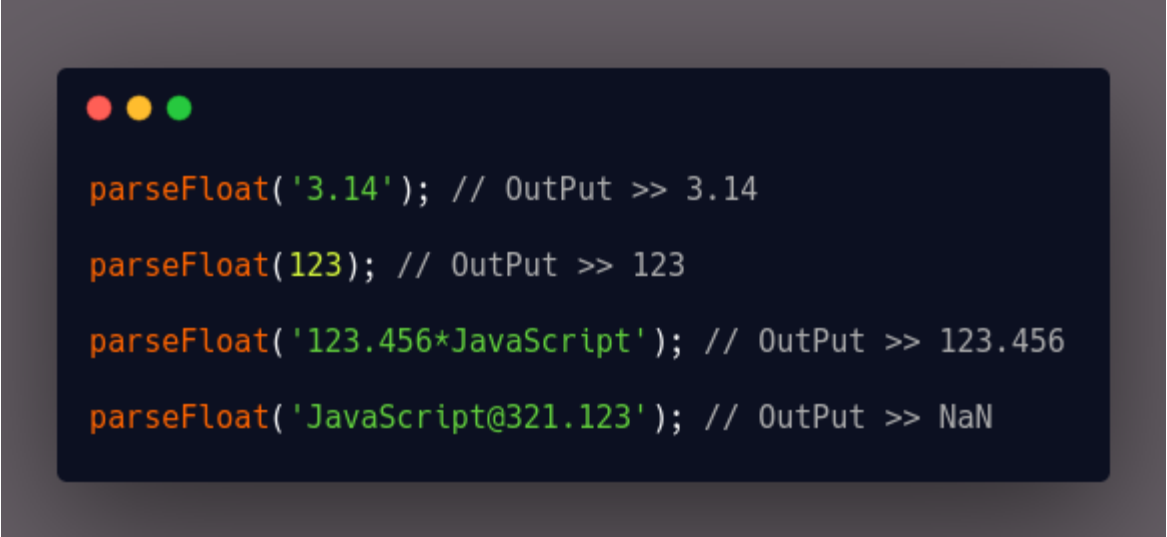
parseFloat() üsulu ilə sapı parza və ya analiz etmək üçün istifadə olunur, sonra isə üzən nöqtə nömrəsinə çevrilir, yəni 3.14.

Sintaksis:

```
parseFloat(String)
```

Vahid string type parametri yalnız metodda keçirilə bilər.

Nümunə:



```
parseFloat('3.14'); // OutPut >> 3.14  
parseFloat(123); // OutPut >> 123  
parseFloat('123.456*JavaScript'); // OutPut >> 123.456  
parseFloat('JavaScript@321.123'); // OutPut >> NaN
```

Yuxarıdakı nümunələrdə parseFloat() metodunun davranışı parseInt() metodu ilə eynidir.

— To Boolean Conversion:

Boolean çevrilməsində dəyərlər boolean dəyərlərə çevrilir.

Sintaksis:

```
Boolean(valuesToBoolean)
```

Nümunə:



```
Boolean('123'); // OutPut >> true
Boolean(1); // OutPut >> true
Boolean(0); // OutPut >> true
Boolean(''); // OutPut >> false
Boolean(' '); // OutPut >> true
Boolean(null); // OutPut >> false
Boolean(undefined); // OutPut >> false
Boolean(NaN); // OutPut >> false
```

Əgər boş string(""), null, undefined və NaN kimi dəyərləri keçsək nəticə həmişə yalnız olacaq.

Növ Məcburiyyət və ya İmmiqya Məcburiyyəti :

Type coercion tip çevrilməsinə bənzəyir, lakin yeganə əsas fərq JavaScript mühərriki tərəfindən avtomatik və ya implicit şəkildə yerinə yetirilən məcburiyyətdir.

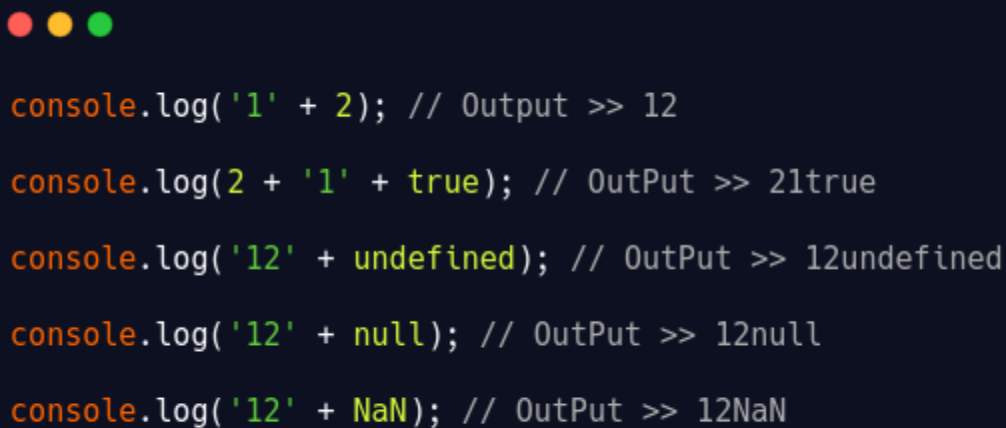
Məsələn, hər hansı bir dəyəri ötürərkən built-in alert() üsulunda JavaScript mühərriki avtomatik olaraq həmin qiymətləri bir stringə çevirəcək və nəticəni nümayiş etdirəcək.

Type coercion-da string, number və **boolean** kimi eyni data tipləri istifadə olunur, onların istədikləri tipə çevrilir.

— To String Coercion :

Plus(+) operatorundan istifadə edərək rəqəm və ya qeyri-string qiyməti ilə bir string əlavə edildikdə, ifadənin çıxışı həmişə bir string olur.

Nümunə:



```
console.log('1' + 2); // Output >> 12
console.log(2 + '1' + true); // OutPut >> 21true
console.log('12' + undefined); // OutPut >> 12undefined
console.log('12' + null); // OutPut >> 12null
console.log('12' + NaN); // OutPut >> 12NaN
```

Type coercion-da əgər birinci operand bir string, ikinci işlənir isə qeyri-stringdirsə, onda nəticə həmişə bir string kimi çap olunur.

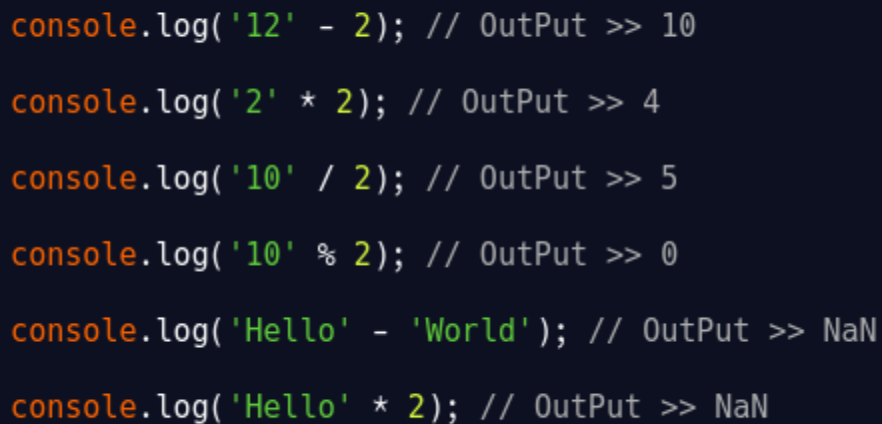
Burada '+' operatoru iki müxtəlif operandların birləşdirilməsi kimi çıxış edir.

— Məcbur Saymağa:

Subtraction(-), Multiplication(*), Division(/) və Modulus(%) kimi riyazi əməliyyatlar string ilə yerinə yetirilmiş sonra ifadənin çıxıntısı bir rəqəmə implicitly çevrilir.

Ədədi məcburiyyətdə plus(+) operatorundan istifadə olunmur.

Nümunə :



```
console.log('12' - 2); // OutPut >> 10
console.log('2' * 2); // OutPut >> 4
console.log('10' / 2); // OutPut >> 5
console.log('10' % 2); // OutPut >> 0
console.log('Hello' - 'World'); // OutPut >> NaN
console.log('Hello' * 2); // OutPut >> NaN
```

Əgər iplərdən hər hansı biri qeyri-numerikdirsə və ya hər ikisi qeyri-ədəddirsə, onda JavaScript nəticəni rəqəm deyil, NaN i.e. kimi göstərir.

— To Boolean Coercion:

Bulean məcburiyyətində həqiqi və yalançı kimi boolean dəyərləri bir ədədə çevrilir.



```
console.log(true + 1); // OutPut >> 2  
console.log(false - 1); // OutPut >> -1  
console.log(true * undefined); // OutPut >> NaN  
console.log(false / null); // OutPut >> NaN  
console.log(true % NaN); // OutPut >> NaN
```

Yuxarıdakı nümunələrdə JavaScript **həqiqi** dəyəri **1**, **saxta** dəyəri isə **0** hesab edir.