# JAVASCRIPT

# Dili

Yazar : Milad Mərəndi

2015-ci ildə

# Başlıq

# Giriş

Javascript(cava iskiript oxuyaq) bilgisayarın dinamik proqrəmləşdirmə dillərindəndi ki 1995-ci ildə BRENDAN EICH tərəfindən C dilinin vasitəsi ilə yazılıb. bu dil C dilinə çox bənzəyir və sintaksi C/C++/Java dilləriynən birdir. geçəcaqda bu dil təkcə müştəri tərəfində (Client-Side) işlənirdi amma indi server tərəfindədə (Server-Side) Node.js adlı tanınan texnologiyanın vasitəsi ilə işlənir. Javascript indi internet saytlarının yüzə doxsanında işlənir.
AJAX-ınan JQuery(ceykueri oxuyaq) texnologiyaları javascript-dən törəniblər və indi geniş şəkildə işlənillər. bu kitabda Jquery texnologiyasının barəsində danışmayacayıq və təkcə xalis javascript (pure js) barəsində və bir qədər AJAX barəsində danışacağıq. ixtisarca javascript-i JS formasında yazallar və bu dilin fayllari (.js) formətində yazılar.

JS-i yazmaqa çoxlu Birləşmiş İnkişaf Mühiti(Integrated development environment) vardır, ixtisarca IDE və ya BİM adlanır.

Bu bağlantılardan endirə bilərsiz:

www.jetbrains.com/webstorm/

www.aptana.com

bu kitabda biz sadəcə Notepad redaktorundan istifadə edəcəyik.

ilkdə bu adresi izləyin və Notepad yazılımını (software) açın

start menu>run>Notepad

sonradan bu kodi yazın

```
<html>
<head>
</head>
<body>
   <script  language="javascript">
      document.write("Hello World");
   </script>
</body>
</html>
```

İndi bu yazını bu adıynan( *first.html* ) xilas edin,sonradan bu faylın üstündə tikləyin(click)  taki açılsın, səhifədə *Hello World*  yazısın görə bilərsiz, bu ilk proqrəmidi.

Tam yazılar ki bu <> əlamətlərinin arasında yazılıllar təq(TAG) adlanıllar, yazan proqrəmdə 8 təq var, təqlər bəzən cüt olur və bəzən yalqız,məsəl üçün *<script>* təqi açılandan sonra gərək *</script>*  təqiynən bağlana ta ki bəllənəki kod yazmaq qutulub.

<*br*> təqi bir yalqız təqdi, yani bağlanmağa ehtiyaci yoxdur.

Javascript kodların üç yolda brauzerə tanıddırmaq olar ki aşağıda yazılıb

## Birinci yol:

```
<html>
<head>
   <script  language="javascript" src="code.js"></script>
</head>
<body>
</body>
</html>
```

Bu modeldə yazanda biz gərək proqrəmi bir başqa faylda(*code.js*) yazıb və HTML şablonuna (template) idxal edaq Kodi idxal edmaq təkcə bu iki təqin *<head></head>* arasinda mümkündür. bu yol səbəb olur ki bizim HTML şablonumuz çox uzun və qarışıq olmasın.

code.js faylının içində yazılan proqrəm sadəcə budur:

```
document.write('Hello World');
```

## İkinci yol:

```
<html>
<head>
   <script  language="javascript">
      document.write("Hello World");
   </script>
</head>
<body>
```

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

Çünki HTML şablonunun ilkdə *<head></head>* təqi emal olunur və sonradan *<body></body>* təqi, o proqrəm ki *<head></head>* təqində yazılır ilkdə yüklənir və o proqrəm ki *<body></body>* təqinin arasında yazılır sonradan yüklənir.

## Üçüncü yol:

```
<html>
<head>
</head>
<body>
   <script  language="javascript">
      document.write("Hello World");
   </script>
</body>
</html>
```

veb səhifəsi tam halda yüklənməmişdən öncə proqrəm emal olunur və ola bilər ki proqrəm tam şəkildə emal olunmuya.

Javascript  kodların başqa yollarınanda HTML şablonuna idxal edmaq olar ki diyən üç yolun altyığımından(subset) sayılıllar,aşağıda bir neçə yolu yazılıb:

## Birinci başqa yol:

```
<html>
<head>
   <script language="javascript" >
        var script = document.createElement('script');
        script.src = 'code.js';
        document.write(script.outerHTML);
  </script>
</head>
<body>
</body>
</html>
```

Bu yolunan javascript proqrəmini bir element formasında HTML şablonunun *<head></head>* bölümünə əlavə olunur, bu idxal yolunun üstünlüki (Superiority) budur ki veb səhifəsin hər bir yerində və hər zamanda yeni skript(script) əlavə edmək olar, ilkdə diyən üç yol təkcə səhifə yüklənəndə skriptlər yüklənillər ama bu başqa yollarınan idxal olan skriptlər dinamik formasında olullar.

## İkinci başqa yol:

```
<html>
<head>
  <script language="javascript" >
        document.write("<script  language='javascript'  src='code.js'><\/script>");
 </script>
```

```
</head>
<body>
</body>
</html>
```

Bu da skripti dinamik şəkildə veb səhifəsinə idxal edən başqa yoludur. bu proqrəmdə *document* bir şeydi(object) və *write()* bir üsul (method), gələcəkdə bunlara görə danışacağıq.

## Üçüncü başqa yol:

```
<html>
<head>
   <script  language="javascript"  src="http://localhost/y/code.php">
   </script>
</head>
<body>
</body>
</html>
```

Bu idxal yolu öncəki yollarınan bir qədər fərq edir, fərqi budur ki biz *.js* faylının əvəzinə bir *.php* faylına server üstündə bağlanırıq(connect) və bizim js proqrəmimiz *.php* faylının içində yazılır. deməliyik ki *.php* formətində olan faylılar, *PHP* dilinin faylılarıdılar və server tərəfində işlənillər.

*code.php* içində olan proqrəm bu şəkildədi:

```
<?php
Header("content-type: application/x-javascript");
echo "document.write('Hello World');";
?>
```

## Dördüncü başqa yol:

```
<html>
<head>
<script src="data:text/javascript;base64,ZG9jdW1lbnQud3JpdGUoJ0hlbGxvIFdvcmxkJyk7">
</script>
</head>
<body>
</body>
</html>
```

Bu idxal yolunda biz proqrəmi *base64* alqoritminən kodlaşdırıq(encode) və *src* partısında(part) yazırıq, *base64* kodlaşdırmaq yolunun barəsində sonradan danışacağıq.

```
ZG9jdW1lbnQud3JpdGUoJ0hlbGxvIFdvcmxkJyk7  =  document.write("Hello World")
```

Soldakı kod həmən sağdakı kodun şifrlənmiş(encoded) formasıdı.

## Dəyişilən (variable) :

Dəyişilənlər javascript dilində sadəcə *var* sözüynən müəyyən olunullar (diqqət edin kiçik hərflərinən yazılar) və veriləri(Data) onların içinə qoymaq olar, aşağıdaki məsələ baxın:

```
var  varAdi= 2;
```

indi *varAdi* dəyişilənin içində *2* oturub.

bu örnəkdə *var* sözün yazmaq icbaridi və *varAdi* sözü ixtiyari addır ki proqrəmçi onu yazıb, ilk karakter(character) bir hərfinən və ya _ və ya *$* əlamətiynən başlana bilər və başqa karakterlər hərf və say(number) ola bilər. diqqət edin *var* və *varAdi* sözlərinin arasında azı bir boşluq (space) gərəklidir.

Başqa düz örnəklər(example):

```
var  _test=10;
var   $test= 2.3;
var  test_63= false;
var  test_new= function(){return true};
var  _test20=document;
var  test60= "string";
```

səhv örnəklər :

```
vartest= 20 ;
var  6test =30;
var  %test= 60;
```

dəyişilənlər təkcə bir kəz(dəfə) müəyyən olunullar və yenidən *var* sözüynən müəyyən edmək lazim dəyil, bu örnəkə diqqət edin :

```
var  test1= 6;
var  test2= test1;
test1= 10;
```

bu proqrəmdə birinci sətrdə, *test1* dəyişilənin dəyəri(value) ilkdə *6* olub və ikinci sətrdə *test1* dəyişilənin dəyəri *test2* dəyişilənin içində oturur və üçüncü sətrdə yenidən *test1* dəyişilənin dəyəri *10* olur, proqrəmin sonunda *test1* dəyəri *10* və *test2* dəyəri *6* olunur.

bir neçə söz ki dəyişilənləri və funksiyaları (function) onlarınan adlandıranmarıq (We can not named) çünki  js dilində onlar rezerv olunub və ya js dilində bir şeyi bilindirillər, örnək üçün *var* sözi.

aşağıda yazılanlar bu sözlərdəndilər :

| | | | | |
|---|---|---|---|---|
| case | else | new | var | function |
| catch | finally | return | void | do |
| continue | for | switch | while | instanceof |
| default | if | this | with | try |
| delete | in | throw | break | typeof |
| abstract | enum | int | short | public |
| boolean | export | interface | static | import |
| byte | extends | long | super | double |
| char | final | native | synchronized | |
| class | float | package | throws | |
| const | goto | private | transient | |
| debugger | implements | protected | volatile | |

bir neçə səhv örnək :

```
var   var =200;
var   break = "string ";
var    int=2.4 ;
```

diqqət edin əgər rezerv olmuş sözlərin ilk hərfini böyük hərfinən yazsaz problemi yoxdur, məsəl üçün  bu örnəklər :

```
var   Var =200;
var   Break = "string ";
var    Int=2.4 ;
var   var_   = 200;
```

js dilində 7 cür/növ veri var ki dəyişilənlər onlari tanıyıllar və qəbul edəllər , dəyişilənlərin növünü anlamaq üçün *typeof()* funksiyasından istifadə edərik. bu örnəkinən dəyişilənlərin növünü (kind of) anlamaq olar.

örnək :

```
<html>
<head>
</head>
<body>
   <script  language="javascript">
      var  say_a = 365 ;
      var  nov =  typeof(say_a);
      document.write(nov);
   </script>
</body>
</html>
```

Js-in 7 növ tanıyan verilər bunlardılar :

| | | | |
|---|---|---|---|
| undefined | boolean | string | function |
| null | number | object | |

## Dəyişilənin qeyri müəyyən (undefined) veri növü:

əgər dəyişiləni adlandıraq və ona miqdar vermiyaq(verməyək) onda o dəyişilənin miqdari qeyr müəyyən olar, bir örnək aşağada :

```
<html>
<head>
</head>
<body>
   <script  language="javascript">
       var  say_b ;
       var  nov =  typeof(say_b);
       document.write(nov);
   </script>
</body>
</html>
```

## Dəyişilənin boş (null-hükümsüz) veri növü:

əgər dəyişilənin növü şey (object) ola və o şey çağırmamışdan öncə müəyyən olunmuya, dəyişilənin miqdari *null* olar. buni açıqlama üçün bu örnəkə diqqət edin :

```
<html>
<head>
</head>
<body>
   <script  language="javascript">
       var a=null;
       document.write ("The value of a is " + a);
```

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

js-in görüşünnən *null* və *undefined* ikisidə bir miqdardılar və onların ikisində bərabər tanıyır, bir örnək bu sözi açıqlama üçün :

```
<html>
<head>
</head>
<body>
    <script  language="javascript">
        var  a1 =null;
        var a2;
        document.write(a1+"<br/>");
        document.write(a2+"<br/>");
        if(a1==a2){
           document.write("a1=a2");
        }
    </script>
</body>
</html>
```

## Dəyişilənin bul (boolean) veri növü :

bu növda dəyişilənlərin miqdari iki cür olar ki proqrəmlərdə çox işlənəllər bu iki növ *true / false* -dılar. bir örnək bu növ miqdara :

```
<html>
<head>
</head>
<body>
   <script  language="javascript">
       var a1=false
       var  a2=true;
       if(a1==a2){
           document.write("a1=a2=true");
       }
      else  {
          document.write("The value of a1 is "+a1+"  and  "+" The value of a2 is "+a2);
       }
    </script>
</body>
</html>
```

## Dəyişilənin say (number-nömrə) veri növü :

əgər dəyişilənin içində tam ədəd(Integer) və ya üzən ədəd(Float) olsa onda dəyişilənin miqdari *number* sayılar.

örnək:

```
<script  language="javascript">
       var  num1=1023;
       var num2=30.44;
       document.write(num1+"<br/>");
```

```
        document.write(num2+"<br/>");
        document.write(num2+ num1);
    </script>
```

## Dəyişilənin sim (string) veri növü :

əgər bir ya neçə karakteri birlikdə bir dəyişilənin içində yazsaq onda string növündən istifadə edirik.string bu " əlamətin və ya bu ' əlamətin arasında yazılar.

örnək :

```
<script  language="javascript">
        var  str1="bu bir string-di";
        var str2='bu bir başqa string-di';
        document.write(str1+"<br/>"+str2);
</script>
```

## Dəyişilənin şey (object) veri növü :

bundan öncə örnəklərdə şeydən istifadə edmişik, biz object-dən gerçək dünyada çox istifadə edirik məsəl üçün hər vaxt bir kəs deyir Ağac, biz bilirik ki ağac bir şeydi ki yapraqı və köki var və....

aşağadakı örnəkdə *document* şeyi bizim veb səhifəmizdə görünən object-lərə işarə edir və *write* bir üsuldu.

örnək :

```
<script  language="javascript">
        var  bir_object=document ;
```

```
    bir_object.write("men bir object-em");
</script>
```

## Dəyişilənin funksiya (function) veri növü :

bu cür veri növü çox işlənməz və təkcə adsız funksiya (nameless) və ya sinif (class) düzəldməkə işlənər. adsız funksiyasi üçün bir örnək :

```
<script  language="javascript">
    var  bir_fun=function(){return true;} ;
    document.write(bir_fun());
</script>
```

## Dəyişilənlərin məlumat növların dəyişmək :

ayrı proqrəmləşdirmə dillər kimin js-in bir neçə funksiyasi və üsulu vardi ki onlarınan eləmək olar dəyişilənlərin növünü çevirək. 7 növ məlumatdan təkcə 3 növün bir birinə (to each other) çevirmək olar. Bu üç növ *String / Number / Boolean*-dilər

### 1. Number və Boolean növünü string növünə çevirmək :

iki yol var ki ondan nümrə bə bul miqdarın string miqdarına çevirmək olar, birinci yolda dəyişilən özü şey(object) sayılır və *toString()* üsuluynan (method) dəyişilənin miqdari çevinir. diqqət edin ki *S* böyük hərfinən yazılar  və *t* kiçik hərfinən.

Birmci örnək :

```
<script  language="javascript">
    var  bul_miqdar=true ;
    var  numre_miqdar=100 ;
    var  yeni_string1=bul_miqdar.toString();
    var  yeni_string2=numre_miqdar.toString();
    document.write(yeni_string1+"<br/>"+yeni_string2);
</script>
```

Say və bul miqdarını sim növünə çevirməkin ikinci yolu *String()* funksiyasından istifadə edməkdir, bu yolda hərnə String funksiyasının içinə yazsaq bu funksiya bizə sim növünü qaytaracaq.

ikinci örnək :

```
<script  language="javascript">
    var  bul_miqdar =false ;
    var  numre_miqdar= 123 ;
    var  yeni_string1=String(bul_miqdar);
    var  yeni_string2=String(numre_miqdar);
    document.write(yeni_string1+"<br/>"+yeni_string2);
</script>
```

## 2. String və Boolean növünü Number növünə çevirmək :

Js dilində iki cür say var, biri tam ədəd(integer) və biri üzən ədəd (float), iki yol var ki ondan başqa növləri sim növünə çevirmək olar.

birinci yolda *parseInt()* və *parseFloat()* funksiyalarından istifadə olunur

və təkcə string növünü say növünə çevirək olar ama bul növünü say növünə çevirək olmaz. diqqət edin ki *I* və *F* hərfləri bu funksiyaların adlarında böyük hərfinən yazılar. əgər o simi ki istəyirik miqdarı say növünə çevirlənə içində nümrə olmuya onda JS bizə *NaN* (not a number) miqdarın qaytaracaq.

birinci örnək :

```
<script language="javascript">
    var str ="100.33abc" ;
    var tam_say=parseInt(str);
    var uzen_say=parseFloat(str);
    document.write(tam_say +"<br/>"+ uzen_say);
</script>
```

İkinci yolda string növünü, say və bul növlərinə çevirmək olar. bu yolun birinci yolunan fərqi bundadi ki bul növünü sim növünə çevirir. bu növdə *Number()* funksiyasından istifadə olunar. diqqət edin ki *N* hərfi böyük hərfinən yazılar. əgər sim növündan olan dəyişilənin içində nümrədən başqa karakter ola JS bizə *NaN* qaytaracaq.

ikinci örnək :

```
<script language="javascript">
    var str1_sehv ="100.33abc" ;
    var str2_ duz="100.33" ;
    var bul1 = false ;
    var bul2= true ;
    var numre1=Number(str1_sehv);
```

```
    var  numre2=Number(str2_ duz);
    var  numre3=Number(bul1);
    var  numre4=Number(bul2);
    document.write(numre1+"<br/>"+ numre2 +"<br/>"+ numre3+"<br/>"+  numre4);
</script>
```

## 3. String və Number növünü Boolean növünə çevirmək :

*Boolean()* funksiyası nümrə və sim növlərin bul növünə çevirər. diqqət edin *B* böyük hərfinən yazılar. əgər *""* (boş string) və ya *null* növün və ya *0* miqdarın bu funksiya veraq, onda bu funksiya bizə *false* miqdarın qaytaracaq və  əgər *1* miqdarın versaq bizə *true* qaytaracaq.

örnək :

```
<script  language="javascript">
    var  str1 ="" ;
    var  str2 ="100str" ;
    var  numre1 = 100 ;
    var  numre2= 0 ;
    var  bul1=Boolean(str1);
    var  bul2= Boolean (str2);
    var  bul3= Boolean (numre1);
    var  bul4= Boolean (numre2);
    document.write(bul1+"<br/>"+ bul2 +"<br/>"+ bul3+"<br/>"+  bul4);
</script>
```

## Xususi karakterlər (Special Characters) :

Bir neçə karakter js dilində var ki onları görsədmək üçün \ (backslash) əlamətindən istifadə edərik, bu karakterlər aşağada yazılıb.

\n yeni xətt (göstərici bir yeni xəttə gedib və bundan sonra yazılan yazılar yeni xəttə yazılar)

\t yeni tab (bir yeni tab əlavə olunar,keybord üstündə olan tab kimi)

\ " cüt sitat (double quote)

\ ' tək sitat (single quote)

\r yeni xətt (bu xuxusi karakter *\n* fərqi yoxdur. *\n unix* əməliyyat sistemində işlənər və *\r\n windows* əməliyyat sistemində)

## Şərt (Condition) :

js-də üç növ şərt var ki onların vasitəsi ilə müxtəlif vəziyyətlərdə müxtəlif işlər görmək olar. ilk şərt *if* şərtidir, bu şərt bu *if(){}* formada yazılar, əgər parantezdə yazılan şərt düz ola onda braketin içində yazılan kodlar icra olar, iki örnək bu sözü açıqlama üçün.

birinci örnək :

```
<script  language="javascript">
  var a=122;
  if(a%2==0){
      document.write(a+" bir cut ededdir")
  }
</script>
```

Üstdəki örnəkdə əgər *122* sayının ikiyə bölünməsinin artıqlıqı *0* olmuya, o say cüt ədəd dəyir. diqqət edin *%* əlaməti artıqlıqın əlamətidir. əgər

şərtin kodu bir xətdə yazılsa *{ }* əlaməti lazim dəyir. ikinci örnəkdə braket yazılmıyıb.

ikinci örnək :

```
<script  language="javascript">
  var a=122;
  if(a%2==0)
      document.write(a+" bir cut ededdir")
</script>
```

*If* şərtinin başqa forması *if..else…* formasıdır, əgər istəsək şərtin düz olmamaqına bir başqa iş görək və ya kod yazaq bu formadan istifadə edərik. bir örnək :

```
<script  language="javascript">
  var a=123;
  if(a%2==0){
      document.write(a+" bir cut ededdir")
  }
  else{
      document.write(a+" bir cut eded deyir")
  }
</script>
```

If şərtinin bir başqa şəkili *if...if else...* formasıdır. əgər neçə şərt olsa bu yoldan istifadə edərik, bu yolda müxtəlif şərtlərin vasitəsilə müxtəlif kodlar icra olar.

örnək :

```
<script  language="javascript">
  var a=123;
  if(a%2==0){
       document.write(a+" bir cut ededdir")
  }
  if else(a%2>0){
       document.write(a+" bir cut eded deyir")
  }
</script>
```

İkinci şərt növü *switch...case* şərtidir, bu şərt *if...if else...* şərtinən fərqi bundadı ki hər kəz if else yazmaq istəmir və *( )* içində yazılan şərti *case* -də yazılan şərtlərinən müqayisə edər və əgər bərabər olala *case* altında yazılan kodlari icra edər. diqqət edin ki *break* kodu şərtin sonunda yazılar ta ki şərt qutulsun və əgər break yazılmasa iki şərti bir birinən *AND* edər. *default* kodunu yazmaq ixtiyarıdır və əgər yazılan şərtlərin heç biri bərabər olmasa *default* altında yazılan kodlar icra olunar.

örnək :

```
<script  language="javascript">
  var a=123;
  switch(a%2){
       case 0 :
            document.write(a+" bir cut ededdir");
            break;
```

```
        default :
                document.write(a+" bir cut eded deyir")
  }
</script>
```

Üçüncü şərt növü *( )?:* şərt operatorudur, bu operator *if()else{}* operatoruna taydır və kiçik və sadə şərtləri ondan yazmaq çox rahatdır. Şərt parantez içində yazılar və əgər qaytaran miqdarı düz ola *?* və *:* arasında yazılan kod icra olar və əgər bərabər olmasa *:* əlamətindən sonra yazılan kod icra olar.

örnək :

```
<script  language="javascript">
  var num=prompt("bir numre yazin");
  (num%2==0)?alert(num+" bir cut ededdir"): alert(num+" bir cut eded deyir");
</script>
```

JS dilində istifadəçidən bir miqdar almaq üçün *prompt()* üsulundan istifadə edərik və *alert()* üsulu bir yazını istifadəçiyə göstərər, gələcəkdə bu üsullara barə danışacağıq.

## Döngü (Loop) :

Js dilinin 4 növ döngüsü var, döngülər neçə işi təkrar edmək üçün istifadə olunar.

birinci döngü *for(){}* döngüsüdi, bu döngüdə başlanış (start point) və qutuluş (finish)  və addım (step) parantez içində yazılar və *;* əlamətinən ayrılarlar, tekrar olunan kodlar braket içində yazılar.

birinci örnək :

```
<script  language="javascript">
   for(i=0;i<10;i++){
       document.write(i+"</br>");
   }
</script>
```

Addım bu örnəkdə *i++*-dır və hər kəz tekrar olanda *1* nömrə i dəyişilənə artılar və əgər *i--* olsa bir nömrə çıxılar.

ikinci örnək :

```
<script  language="javascript">
   for(i=10;i>0;i--){
       document.write(i+"</br>");
   }
</script>
```

ikinci döngü növü *for in* döngüsüdür, bu döngü silsiləıərin ünsürlərinə əl tapmaq üçün (access to)  istifadə olunar. aşağadakı örnəkdə item bir dəyişiləndir ki silsilənin tək-tək ünsürlərinə işarə edir.

örnək :

```
<script  language="javascript">
    var arrx={ad:"Milad",sonad:"Merendi",yash:24}
    for(item in arrx){
       document.write(item+"</br>");
    }
</script>
```

Üçüncü döngü növü *while(){}* döngüsüdür, bu döngü o çağacan ki parantezin içində yazılan şərt qutulmuyub icra olunar. bir örnək bu sözi açıqlama üçün :

```
<script  language="javascript">
    while(1){
        var  inp=prompt("nomre yazin :");
         if(inp)
            document.write(inp+"</br>");
         if(!inp)
            break;
   }
</script>
```

Bu örnəkdə *while(1)* yanı heç çağ bu döngü qutulmaz və sonsuzdur, bu döngüdən çıxmaq üçün *break* operatorundan istifadə edərik, *if(inp)* bir şərt bilindirir və o şərt budur ki əgər *inp* dəyişilənin miqdarı *null* olmuya (*null* o zaman olur ki istifadəçi *cancel* düyməsini tikliyə və ya bir miqdar yazmıya) bir kod icra ola. *if(!inp)* bir başqa şərt bilindirir və oda budur ki əgər istifadəçi *cancel* düyməsin tikliyə və ya bir miqdar yazmıya və *ok* eliyə bu şərt icra olar və döngü sona çatar, *!* NOT operatoru adlanar və hər miqdarı tərzinə çevirər (true-->false ; false-->true ; 0-->1 ; 1-->0 ; null-->not null;not null-->null).

dördüncü döngü növü *do{}while()* döngüsüdür, bu döngüdə şərt parantez içində yazılar və kodlar braket içində, fərqi *while()* üsülüynan

bundadı ki ilkdə kod icra olar və sonra şərtə baxılar (ikinci örnək fərqin bilindirir). diqqət edin ki gərək döngünün addımı braketin içində yazıla.

birinci örnək :

```
<script  language="javascript">
    var x=0;
    do{
       document.write(x++);
    }
    while(x<10)
</script>
```

ikinci örnək :

```
<script  language="javascript">
      x=10;
      do{
         document.write(x);
         x++;
       }
       while(x<10)
</script>
```

İkinci örnəkdə şərt düz dəyir ama kod icra olur və *while* döngüsünən fərqi budur.

## Funksiya (function) :

Funksiyalar javascript dilində iki formada yazılallar, birinci formada ilkdə function sözü və bir boşluqdan (space) sonra funksiyanın adı (bu ad ixtiyarıdı)  yazılar və sonra ( ){ } əlamətləri yazılar. əgər funksiyanın bir ya neçə parametri (parametrlər o verilərdilər ki biz funksiya göndərik ta ki funksiyada onlardan istifadə olunsun) olsa onlar () əlamətin içindəyazılar və , əlamətiynən ayrılallar, kodlar hamısı { } əlamətin içində yazılar.

birinci yolda funksiyanın son forması bucur olar :

function  ad(birinci_para , ikinci_para , ...){ kodlar }

bir örnək bu məsələni açıqlama üçün :

```
<script  language="javascript">
   function  birinci (){
      document.write(" birinci funksiya <br/>");
   }
   function ikinci(str){
      document.write(str);
   }

   birinci();
   ikinci(" ikinci funksiya1<br/>");
   var z=" ikinci funksiya2";
   ikinci(z) ;
</script>
```

Funksiyalar səbəb olar ki biz bir ya neçə kodi bir kəz yazıb və hər zaman ehtiyacımız olsa təkcə o funksiyanın adın çağırmağınan bizim kodlarımız icra ola. funksiyani çağırmağa təkcə adın yazıb və () əlamətin yazacaq, örnək üçün *ad()* .

```
<script  language="javascript">
   function  ad (){
      document.write("funksiya");
   }
   ad() ;
</script>
```

Funksiyalar iki növdə olarlar, birinci növdə bizə bir miqdar qaytarmaz, öncəki örnəkdə bu növ funksiyadan istifadə olunub, ikinci növdə funksiya bizə bir miqdar qaytarar, bu miqdar *string/number/boolean* və ya *null* ola bilər. miqdari qaytarmaq üçün *return* sözündən istifadə edərik

bir örnək funksiyaların ikinci növünə :

```
<script  language="javascript">
   function  ad (){
      var df= "funksiya1" ;
      return df ;
   }
   var  abc=ad() ;
   document.write(abc) ;
</script>
```

funksiyaların ikinci növünə başqa örnək :

```
<script  language="javascript">
   function  ad (){
      return  "funksiya2" ;
   }
   document.write(ad()) ;
</script>
```

Funksiyalar hamısı ki buracan onların haqqında danışmışıq birinci formadanıdılar .  ikinci formada ki onun adı adsız funksiyadı (nameless function), biz funksiyanı bir dəyişilənin içinə qoyuruq. bu forma bizə bu imkani verir ki bir sinif düzəldaq, siniflərin haqqında sonradan danışacağıq.

adsız funksiya üçün  bir örnək :

```
<script  language="javascript">
   var ad=function  (){
      return  "adsiz funksiya" ;
   }
   document.write(ad()) ;
</script>
```

Öncəki örnəkdə görən kimin bu funksiyanın adı yoxdur və biz o dəyişilən ki funksiya onun içində oturub çağırırıq. bu formada olan funksiyalarda *return* sözünü yazmq icbarıdı.

## Silsilə (Array) :

Javascript dilində bir Array adlı sinif var ki onun vasitəsilən bir şey düzəldmək olar ki bir silsiləyə işarə edsin.əgər biz istəsək neçə dəyişilənin yerinə bir dəyişiləndən istifadə edək onda array-dən istifadə edərik.

Birinci yolda silsilədən istifadə edmək üçün ilkdə bir dəyişilən adlandırlıq və bir yeni *new Array()* şeyi düzəldərik, əgər verilərin sayı bəlli ola onda şeyi *new Array(x)* formasında yazacağıq, *x* haman verilərin sayıdı, sonradan veriləri silsilənin içinə yazacağıq.

İkinci yolda silsiləni adlandıranda, veriləri silsilənin içinə yazacağıq.

Üçüncü yolda *new Array()* şeyindən istifadə edmərik və sadəcə *[ ]* əlamətindən istifadə edərik. üç örnək silsiləri düzəldmək üçün :

birinci örnək :

```
<script  language="javascript">
    var arr1=new Array();
    var arr2=new Array(3);
    arr1[0]= "Milad";
    arr1[1]= " Merendi" ;
    document.write(arr1[0]+arr1[1]+ "<br/>") ;
    arr2[0]= "bu";
    arr2[1]= " bir" ;
    arr2[2]= " silsiledi" ;
    document.write(arr2[0]+arr2[1]+ arr2[2]) ;
</script>
```

İkinci örnək :

```
<script  language="javascript">
    var arrx=new Array("bu"," bashqa"," silsiledi");
    document.write(arrx[0]+arrx[1]+ arrx[2]+ "<br/>") ;
    document.write(arrx[0]+ arrx[2]) ;
</script>
```

Üçüncü örnək :

```
<script  language="javascript">
    var arrx=["bu"," bashqa"," silsiledi"];
    document.write(arrx[0]+arrx[1]+ arrx[2]+ "<br/>") ;
    document.write(arrx[0]+ arrx[2]) ;
</script>
```

Silsilənin uzunluqun tapmaq üçün *length* xassəsindən (property) istifadə edərik, bu xassə bizə silsilənin içində olan ünsürlərin sayların qaytarar. bir örnək bu xassə üçün :

```
<script  language="javascript">
    var arrx=["bu"," bashqa"," silsiledi"];
    document.write(arrx.length) ;
</script>
```

Silsiləni ayrı yolunanda yazmaq olar, bu yolda silsilə bir şey sayılır və Indekslər (indexes) nömrə dəyillər və biz indeksləri müəyyən edirik. diqqət edin bu yolda *[ ]* əlamətinin yerinə *{ }* əlamətinnən istifadə edərik

örnək :

```
<script  language="javascript">
    var arrx={Ad : "Milad", Soyad:" Merendi", yash : 25};
    document.write(arrx[Ad]) ;
</script>
```

## Silsilənin ünsürlərin silmək:

Bu işi görmək üçün üç yol var, birinci yolda *pop()* üsulu ilə silsilənin sağından bir ünsür silinər. ikinci yolda *shift()* üsulu ilə silsilənin solundan bir ünsür silinər. hər kəz ki bu iki üsulu çağırılıl bir ünsür sağdan və ya soldan silinir.

əgər istəsək bir/neçə ünsürü ki silsilənin arasındadı silək onda *splice(x,y)* üsulundan istifadə edərik. bu üsulun çoxlu parametri var ki burda iki parametrindən istifadə olunar, birinci parametr *(x,)* o silinən ünsürün mövqeyin müəyyən edir və ikimci parametr *(,y)* silinən ünsürlərin sayın müəyyən edir.

əgər istəsək silsilənin ünsürlərinin hamısını görək təkcə silsilənin adın yazacağıq, məsəl üçün *arrx* aşağadakı örnəkdə.

birinci yola bir örnək :

```
<script  language="javascript">
    var arrx=["Milad", " Merendi",  25];
    arrx.pop();
    document.write(arrx) ;
</script>
```

ikinci yola bir örnək :

```
<script  language="javascript">
    var arrx=["Milad", " Merendi",  25];
    arrx.shift();
    document.write(arrx) ;
</script>
```

üçüncü yola bir örnək :

```
<script  language="javascript">
    var arrx=["Milad", " Merendi",  25];
    arrx.splice(1,1);
    document.write(arrx) ;
</script>
```

## Silsiləyə ünsür əlavə edmək:

Birinci yolda *push(x)* üsulu ilə bir veri silisilənin sağına əlavə edmək olar , bu üsulun *x* parametri həmən o veridi ki istəyirik silsiliyə əlavə ola.

İkinci yolda *unshift(x)* üsulundan istifadə edərik, bu üsulun *x* parametri həmən o veridi ki istəyirik silsiliyə  soluna əlavə ola.

Üçüncü yolda *push()* üsulunun yerinə *arrAdi[arrAdi.length]* yoldan istifadə edərik, *arrAdi* həmən silsilənin adıdı ki burda *arrAdi* fərz olunub və *arrAdi.length* silsilənin sonuna işarə edir.

Dördüncü yolda *splice()* üsulu ilə bir ya neçə veri silsiliyə əlavə olunar, bu üsul bu *splice(x,y,z,...)* formada yazılar, birinci parametr *(x)* əlavə olunan ünsürün mövqeyin müəyyən edir və ikimci parametr *(y)* silinən

ünsürlərin sayın müəyyən edir  ki burda *0* olacaq, başqa parametrlər əlavə olunan verilərdilər.

birinci yola bir örnək :

```
<script  language="javascript">
    var arrx=["Milad", " Merendi"];
    arrx.push(25);
    document.write(arrx) ;
</script>
```

ikinci yola bir örnək :

```
<script  language="javascript">
    var arrx=[" Merendi",  25];
    arrx.unshift("Milad");
    document.write(arrx) ;
</script>
```

üçüncü yola bir örnək :

```
<script  language="javascript">
    var arrx=["Milad", " Merendi"];
    arrx[arrx.length]=25 ;
    document.write(arrx) ;
</script>
```

dördüncü yola bir örnək :

```
<script  language="javascript">
    var arrx=["Milad",25];
    arrx.splice(1,0, " Merendi") ;
    document.write(arrx) ;
</script>
```

## Silsilənin ünsürlərin çeşidləmək(sort) :

Bu işi görmək için *sort()* üsulundan istifadə edərik, *sort()* üsulu silsilənin içində olan veriləri əlifba sırası ilə nizamlandırar(regularize).

örnək :

```
<script  language="javascript">
    var arr=["Alma", " Portaghal",  "Banana", "chiyelek"];
    arr.sort();
    document.write(arr) ;
</script>
```

əgər verilər *string* növündən olmaya və say növündən ola onda bu yolunan silsiləni nizamlandırarıq.bu yolda nömrələr kiçikdən böyükə nizamlanıllar.

örnək :

```
<script  language="javascript">
    var arr=[35 ,120 ,40, 10 ,6];
    arr.sort(function(a,b){return a-b});
    document.write(arr) ;
</script>
```

əgər istəsək nömrələr silsilədə böyükdən kiçikə nizamlansın onda *a-b* əvəzinə *b-a* kodundan istifadə edərik, örnək :

```
<script  language="javascript">
    var arr=[35 ,120 ,40, 10 ,6];
    arr.sort(function(a,b){return b-a});
    document.write(arr) ;
</script>
```

əgər diqqət edsəz nömrələri nizamlandırmaq üçün *sort* üsulunun içində bir adsız funksiyadan istifadə edmişik, bu funksiya tutuşdurma funksiya (comparison function) adlanır, bu funksiya iki miqdar alır və bir miqdar sort üsuluna qaytarır, bu miqdar ya sıfırdan azdır və ya çoxdur və ya sıfırdır, məsəl üçün *a=5* və *b=3* olsa onda *a-b* miqdarı *2* olar, cünki *2* sıfırdan böyükdür ona görə kiçik nömrə *a* dəyişilənin içində oturub və böyük nömrə *b* dəyişilənin içində otüracaq.

əgər *a=3* və *b=5* ola onda *a-b= -2* olar, çünki *-2* sıfırdan kiçikdir ona görə böyük nömrə *b* dəyişilənin içində oturacaq və kiçik nömrə *a* dəyişilənin içində.

əgər *a=3* və *b=3* ola onda *a-b= 0* olar, dəyişilənlərin miqdari dəyişilməz.

## Silsilənin ünsürlərin dəyişmək:

Silsiləni ünsurlərin dəyişmək üçün onun indeks sayından istifadə edərik bu örnəkə diqqət edin :

```
<script  language="javascript">
    var arr=[1,2 ,3, 5 ,8];
    arr[2]=6 ;
    document.write(arr) ;
</script>
```

## Silsilənin ünsürlərin silməkə bir başqa yol:

Bu yolda silsilənin ünsürləri şey sayılar və onların yerinə *undifined* miqdarı oturar. bu yolda *delete* operatorundan ünsürləri silmək üçün istifadə edərik.

örnək :

```
<script  language="javascript">
    var arr=[1,2 ,3, 5 ,8];
    delete arr[4];
    document.write(arr) ;
</script>
```

## Silsiləni tərsinə çevirmək:

Silsiləni tərinə çevirmək üçün *reverse()* üsulundan istifadə edərik, məsəl üçün silsilənin sonunda olan ünsür ilk ünsürünan yürin dəyişər.

örnək :

```
<script  language="javascript">
    var arr=[0,1 ,2, 3 ,4];
    arr.reverse() ;
    document.write(arr) ;
</script>
```

## silsilələri yapışdırmaq:

iki silsiləni yapışdırmaq üçün *concat()* üsulundan istifadə edərik, bu üsulun adı *Concatenates* sözündən törənib ki yapışdirmaq anlamı verir. bir örnək bu mövzunu açıqlama üçün (bu örnəkdə ikinci silsilə birinci silsiliyə yapışır ) :

```
<script  language="javascript">
    var birinci_arr=[0,1 ,2, 3 ];
    var ikinci_arr=[4,5 ,6, 7];
    var birleshmish=birinci_arr.concat(ikinci_arr) ;
    document.write(birleshmish) ;
</script>
```

Neçə silsiləni  bir birinə yapışdırmaq üçün parantezin içində bir silsilənin adının yrinə neçə ad yazıb və [ , ] əlamətinən bir birindən ayıracağıq.

örnək :

```
<script  language="javascript">
    var birinci_arr=[ "computer " ," programming " ,"language. "];
    var ikinci_arr=[ "is " ,"a " ,"dynamic " ];
    var ucuncu_arr=[ "JavaScript ", "(JS) "];
```

```
    var birleshmish=ucuncu_arr.concat(ikinci_arr,birinci_arr) ;
    document.write(birleshmish) ;
</script>
```

## Silsilənin neçə ünsürün seçmək:

Bir ya neçə ünsüri silsilənin içindən seçmək üçün *slice()* üsulundan istifadə edərik, bu üsul iki parametr qəbul edər, birinci parametr seçimin başlanışını və ikinci parametr bitişini bilindirir. əgər ikinci parametr yazılmasa javascript belə sanar ki ikinci miqdar silsilənin ölçüsüynən bərabərdi.

birinci örnək *slice* üsuluna :

```
<script  language="javascript">
    var arr1=[0,1 ,2, 3,4,5 ];
    var arr= arr1.slice(2,5) ;
    document.write(arr) ;
</script>
```

ikinci örnək *slice* üsuluna :

```
<script  language="javascript">
    var arr2=[0,1 ,2, 3,4,5 ];
    var arr= arr2.slice(3) ;
    document.write(arr) ;
</script>
```

## Silsiləni string növünə çevirmək:

Javascript dilində dörd üsul var ki silsiləni sim növünə çevirillər , *toString()* üsulu bunların biridi, ikinci üsul *valueOf()*-di və üçüncü *toLocaleString()* üsuludur.

birinci örnək :

```
<script  language="javascript">
    var arr=[1,2 ,3, 5 ,8];
    document.write(arr.toString()) ;
</script>
```

ikinci örnək :

```
<script  language="javascript">
    var arr=[1,2 ,3, 5 ,8];
    document.write(arr.valueOf()) ;
</script>
```

üçüncü örnək :

```
<script  language="javascript">
    var arr=[1,2 ,3, 5 ,8];
    document.write(arr.toLocaleString()) ;
</script>
```

Dördüncü üsul *join()* adlı bir üsuldur və bir az öncəki yapışdıran üsullarınan fərq edir, bu üsul silsilənin ünsürlərin bir ayırıcının (the separator) vasitəsilə bir birinə yapışdırır, bu ayırıcı bir boşluq (space) və ya bir karakter və ya neçə karakter ola bilər.

birinci örnək :

```
<script  language="javascript">
    var arrx=[1,2 ,3, 5 ,8];
    document.write(arrx.join(" ")) ;
</script>
```

ikinci örnək :

```
<script  language="javascript">
    var arrx=[1,2,3, 5 ,8];
    document.write(arrx.join("-")) ;
</script>
```

üçüncü örnək :

```
<script  language="javascript">
    var arrx=[1,2 ,3, 5 ,8];
    document.write(arrx.join("***")) ;
</script>
```

## String növünü silsilə növünə çevirmək:

*Split()* üsulunun vasitəsilə sim növünü silsilə növünə çevirmək olar, bu üsul bir ya neçə karakter qəbul edər, bu karakterlər olan yerlərdən silsilənin ünsürləri ayrılar.

birinci örnək :

```
<script  language="javascript">
    var str="M-i-l-a-d";
    var arr=str.split("-") ;
    document.write(arr[0]+ arr[1]+ arr[2]+ arr[3]+ arr[4]) ;
</script>
```

ikinci örnək :

```
<script  language="javascript">
    var str="java script";
    var arr=str.split(" ") ;
    document.write(arr[0]) ;
</script>
```

üçüncü örnək :

```
<script  language="javascript">
    var str="Milad";
    var arr=str.split("") ;
    document.write(arr[0]) ;
</script>
```

## Silsilənin indeksinin olmasını müəyyən edmək:

indeksin olmasını müəyyən edmək üçün *in* operatorundan istifadə edərik , əgər çağıran indeks nömrəsi silisilənin içində olsa bu operator bizə *true* miqdarın qaytaracaq və əgər olmasa *false* miqdarın qaytaracaq.

diqqət edin indekslər 0-dən başlanar.

örnək :

```
<script  language="javascript">
    var a = [1, 2, 5, 4]
    document.write(1 in a);
</script>
```

başqa örnək :

```
<script  language="javascript">
    var a = [1, 2, 5, 4] ;
    var b=4 in a;
    document.write(b);
</script>
```

## Silsilənin içində bir ünsürün olmasını müəyyən edmək :

bunu müəyyən edmək üçün *indexOf()* üsulundan istifadə edərik.biz bu üsula bir ünsür göndərik və bu üsul ünsürün indeksinin nömrəsin qaytarar, əgər ünsür silsilənin içində olmasa *-1* miqdarı qaytaracaq.

örnək :

```
<script  language="javascript">
    var arr = ["m", "i","l", "a", "d"] ;
    var str=arr.indexOf("a");
    document.write(str);
</script>
```

## Silsilənin ünsürlərinin miqdarının olmaqını yoxlama:

əgər silsilənin içində olan ünsürlərin birinin və ya neçəsinin miqdari *null/undefined/NaN* olsa onda *every()* üsulunun vasitəsiynən təyin etmək olar. *every()* üsulu bizə false və ya true miqdarın qaytarar.bu üsulun içində bir adsız funksiya yazılar ki birbəbir ünsürlərin miqdarın *every()* üsuluna qaytarar və bu üsul *null* ya *undefined* və ya *NaN* olan miqdarları false növünə çevirər və qaytarar.

birinci örnək :

```
<script  language="javascript">
    var arr= [1, 6, 3,9, 5 ,6];
    var c=arr.every(function(item, index, array){return item;});
    document.write(c);
</script>
```

ikinci örnək :

```
<script  language="javascript">
    var arr= [1, 6, 3, undefined, 5 ,6];
    var c=arr.every(function(item, index, array){return item;});
    document.write(c);
</script>
```

## Silsilənin ünsürlərin süzmək(to filter):

hərdən *null* ya *undefined* və ya *NaN* miqdarları silsilənin içində oturar, bu növ veriləri silsilənin içindən silmək üçün *filter()* üsulundan istifadə edərik. bir adsız funksiya bu üsulun içində yazılar, bu adsız funksiya ünsürlərin miqdarın *filter()* üsuluna qaytarar və bu üsul *null undefined NaN* olan miqdarları silər və başqa miqdarlarin özünü qaytarar.

örnək :

```
<script  language="javascript">
    var arrx = ['a',’b’,null,'c'];
    cx=arrx.filter(function(item, index, array){return item;});
    document.write(cx);
</script>
```

## Silsilənin ünsürlərinin miqdarının olmamaqını yoxlama:

bu üsulun *every()* üsuluynan fərqi bundadi ki every üsulu əgər silsilənin ünsürlərinin birinin miqdari *null* ya *undefined* və ya *NaN* ola onda *false* miqdarın bizə qaytaracaq isə (but) *some()* üsulu əgər silsilənin tam ünsürləri *null* ya *undefined* və ya *NaN* ola onda false miqdarın qaytaracaq əks halda *true* miqdarın qaytaracaq.

birinci örnək :

```
<script  language="javascript">
    var ary = [null,null,1,undefined];
    c=ary.some(function(item, index, array){return item;});
    document.write(c);
</script>
```

ikinci örnək :

```
<script  language="javascript">
    var ary = [null,null,null,undefined];
    c=ary.some(function(item, index, array){return item;});
    document.write(c);
</script>
```

## Silsilənin tam ünsürlərinə bir miqdar əlavə edmək:

*map()* üsulunun vasitəsilə bir miqdar silsilənin tam ünsürlərinə əlavə edmək olar, təkcə *undefined* və *NaN* miqdarlarına bir miqdar əlavə edmək olmaz. bu üsulunda içində bir adsız funksiya yazılar taki ünsürlərin miqdarın *map* üsuluna qaytarsın.

birinci örnək :

```
<script  language="javascript">
    var arr = [null,null,10,6];
    c2=arr.map(function(key, value, array){return key+5;});
    document.write(c2);
</script>
```

ikinci örnək :

```
<script  language="javascript">
    var arr = [null,null,10,undefined];
    c2=arr.map(function(key, value, array){return key+5;});
    document.write(c2);
</script>
```

üçüncü örnək :

```
<script  language="javascript">
    var arr = [10,null, "bir",undefined];
    c2=arr.map(function(key, value, array){return key+"yazi";});
    document.write(c2);
</script>
```

## Sim (String) :

Bir neçə karakter birlikdə string adlanar.sim veri növünü düzəldmaq üçün js dilində *String()* adlı bir sinif var. bu sinifin vasitəsiynən bir şey düzəldmaq olar ki bir sim növünə işarə edir. başqa yolda sim növün düzədmaq üçün *String()* sinifindən istifadə edmək lazim dəyir və sadəcə *" "* və ya *' '* əlamətindən istifadə edərik. js dilində karakter və sim növü fərq edməz.

*String()* sinifindən istifadə edmək yoluna bir örnək :

```
<script  language="javascript">
    var str= new String("bu bir simdir");
    document.write(str);
</script>
```

*String()* sinifindən istifadə edməmək yoluna bir örnək :

```
<script  language="javascript">
    var str="bu bir simdir";
    document.write(str);
</script>
```

*diqqət edin bir sinifdən istifadə edmək üçün gərək *new* sözündən sinifin adından öncə istifadə edək. biz siniflərin vasitəsiynən şey (object) düzəldə bilərik, birinci örnəkdə str veri növü şeydir və ikinci örnəkdə simdir. bunu açıqlama üçün bir örnək :

```
<script  language="javascript">
      var str_obj=new String("bu bir simdir");
      var str_str="bu bir simdir";
      document.write("str_obj : "+typeof(str_obj));
      document.write( "<br/>"+"str_str : " +typeof(str_str));
</script>
```

## Simləri birbirlərinə əlavə edmək :

iki yolunan simləri birbirlərinə əlavə edmək olar, birinci yolda *concat()* üsulundan istifadə olunar və ikinci yolda + operatorundan.

ilk yolda birinci verinin adın yazıb və *.concat()* üsulun sonradan yazacağıq, əlavə olunan veriləri və ya dəyişilənləri *()* əlamətinin içində yazacağıq, əgər dəyişilənlərin və ya verilərilərin sayı bir-dən çox olsa onlari ayırmaq üçün ( , ) əlamətindən istifadə edərik.

örnək birinci yola :

```
<script  language="javascript">
    var s1="Mi";
    var s2="la";
    var c1=s1.concat(s2,"d");
    document.write(c1);
</script>
```

örnək ikinci yola :

```
<script  language="javascript">
    var s1="Mi";
    var s2="la";
    var c2=s1+s2+"d";
    document.write(c2);
</script>
```

əgər sim və say növünü bir birinə əlavə edsək onda say növü sim növünə çevirlənər və ondan sonra əlavə olunar.

örnək :

```
<script  language="javascript">
    var s1="Mi";
    var s2=5;
    var c=s1+s2 ;
    document.write( c );
</script>
```

## Simdən bir karakter seçmək :

İki yolunan karakteri seçmək olar, birinci yolda *charAt()* üsulundan istifadə olunar, bu üsul bir say qəbul edər və simdə olan karakterin eyni sayında olan karakteri qaytarar. məsəl üçün *"Milad"* siminin karakterləri tərtib ilə *M=0 i=1 l=2 a=3 d=4* olur, əgər *3* nümrəsin *charAt()* üsulunun içində yazsaq bu üsul *a* karakterin qaytaracaq. (bu yolda sim bir şey sayılır )

örnək :

```
<script  language="javascript">
    var s1="Milad";
    var c=s1.charAt(3);
    document.write(c);
</script>
```

Simdən karakter seçmək üçün ikinci yol indeksdən istifadə edməkdir, bu yolda sim bir silsilə sayılır və karakterin indeksin çağırıb və karakterin alarıq.

örnək :

```
<script  language="javascript">
    var s1="Milad";
    var c=s1[3];
    document.write(c);
</script>
```

## Simdə karakterin yerini tapmaq :

Bir simdə karakterin indeksin tapmaq üçün *indexOf()* üsulundan istifadə edərik, bu üsul bir ya neçə karakteri qəbul edər və onun yerini tapandan sonra indeksinin nömrəsin qaytarar və əgər karakter tapılmasa *-1* miqdarın qaytarar.

örnək :

```
<script  language="javascript">
    var c="Milad ";
    var s1=c.indexOf("a");
    var s2=c.indexOf("la");
    document.write(s1+"<br/>"+s2);
</script>
```

*indexOf()* üsulu karakteri simin sol tərəfindən axtarır, əgər istəsək sağ tərəfdən axtarsın *lastIndexOf()* üsulundan istifadə edərik.

diqqət edin ilkdəki hərf *L* hərfinin kiçik şəkilidi və ikinci *i* hərfinin böyük şəkilidi.

örnək :

```
<script  language="javascript">
    var str="Milad Merendi";
    var chr1=str.lastIndexOf("d");
    var chr2=str.lastIndexOf("M");
    document.write(chr1+"<br/>"+chr2);
</script>
```

Əgər istəsək bir simdə karakterin axtarışı *0* yerinə biz istəyən nömrədən başlansın onda bir başqa arqument *indexOf()* üsulunun içində yazarıq, karakterin axtarışı bu nömrədən olacaq və nömrədən öncəki karakterlər sayılmaz.

örnək :

```
<script  language="javascript">
    var strx="Milad Merendi";
    var ch1=strx.indexOf("e",8);
    var ch2=strx.indexOf("M",3);
    document.write(ch1+"<br/>"+ch2);
</script>
```

Başqa yol bir və ya neçə karakteri simin içində tapmaq üçün *search()* üsulundan istifadə edməkdir, bu üsul sol tərəfdən axtarışı başlar.

örnək :

```
<script  language="javascript">
       var str= "Milad Merendi";
       var pos = str.search("e");
      document.write(pos);
      document.write("<br/>");
      chr= /m/i;
      pos = str.search(chr);
      document.write(pos);
</script>
```

Üstdəki məsəldə */m/i* yazısı bir karakteri kiçik və böyük şəklinə qeyr həssas edir,  bu yazıda *m* bir karakterdir.

## Simdən karakteri silmək:

beş yolunan simdən bir ya neçə karakteri silmək olar, birinci yolda *trim()* üsulundan istifadə edərik. bu üsul təkcə boş karakterləri və yeni sətirləri simin ilkindən və sonundan silər. diqqət edin arada olan boşluqları silməz.

örnək:

```
<script  language="javascript">
   var message = "    Milad Merendi    \n  ";
   document.write(message.trim());
</script>
```

*\n* sözi bir yeni sətir düzəldər.

İkinci yolda *replace()* üsulundan istifadə edərik, bu üsul bir ya neçə karakteri bir başqa karakter və ya sözünən və ya boşluqunan dəyişər.

Birinci örnək :

```
<script  language="javascript">
      var s1 = "sen ged";
      document.write(s1 + "<br>");
      s2 =  s1.replace("d","l");
      document.write(s2+ "<br>");
</script>
```

İkinci ornək :

```
<script  language="javascript">
      var s1 = "b..u? bir ya!!!zid$ir";
      document.write(s1 + "<br>");
      s2 =  s1.replace(/[^a-zA-Z 0-9]/g,"");
      document.write(s2+ "<br>");
</script>
```

Bu örnəkdə *[^a-z A-Z 0-9]* tam karakterlər ki *abcd.... wxyz* və *ABC...XYZ* və *012…89* dəyillər seçilillər. əgər ^ karakteri yazılmasa *abcd.... wxyz* və *ABC...XYZ* və *012…89* karakterləri seçiləllər və başqa karakterlər seçilməzlər. *g* sözi *global* sözünün ilk karakteridi ki müəyyən edir simin başabaş karakterləri gərək axtarıla, əgər bu sözi yazmasaq ilk karakter ki tapıldı axtarış qutular və başqa karakterlər dəyişilməz.

Üçüncü ornək :

```
<script  language="javascript">
    var s1 = "b..u? bir ya!!!zid$$$$ir";
    document.write(s1 + "<br>");
    s2 =  s1.replace(/[^a-zA-Z 0-9 ]{3,5}/g,"");
    document.write(s2+ "<br>");
</script>
```

Üçüncü ornəkdə *{3,5}* yazı seçilən karakterlərin neçə kəz ardıcıl tekrar olunmasın müəyyən edir, məsəl üçün *!* karakteri 3 kəz ardıcıl tekrar olunub və *$* karakteri 4 kəz amma *?* karakteri 1 kəz , *$* və *!* karakterləri seçilər amma *?* karakteri seçilməz.

Üçüncü yolda *substr()* üsulundan istifadə edərik, bu üsul bir ya neçə karakteri seçir və başqa karakterlərə etina edmir. bu üsul iki parametr qəbul edər birinci parametr başlanışı müəyyən edər və ikinci parametr seçilən karakterlərin sayın müəyyən edir. əgər ikinci parametr yazılmasa js simin axirinci miqdarın fərz edər. simdə karakterlərin miqdarları sıfırdan başlanır, y=7 .

örnək :

```
<script  language="javascript">
    var str1 = "bu bir yazidir";
    document.write(str1 + "<br>");
    str1 =  str1.substr(7,4);
    document.write(str1+ "<br>");
</script>
```

Dördüncü yolda *slice()* üsulundan istifadə edərik, bu üsul iki parametr qəbul edər birinci parametr miqdarı seçilməkin başlanışın müəyyən edir və ikinci miqdar seçilməkin qutuluşun. əgər ikinci parametr yazılmasa js simin axirinci miqdarın fərz edər.

örnək :

```
<script  language="javascript">
    var str2 = "bu bir yazidir";
    document.write(str2 + "<br>");
    str2 =  str2.slice(7,11);
    document.write(str2+ "<br>");
</script>
```

Beşinci yolda *substring()* üsulundan istifadə edərik, bu üsul *substr()* üsulu kimin bir ya neçə karakteri seçir və başqa karakterlərə etina edmir. bu üsul iki parametr qəbul edər birinci parametr başlanışı müəyyən edər və ikinci parametr qutarışını (nəhayətin) müəyyən edir. əgər ikinci parametr yazılmasa js simin axirinci miqdarın fərz edər.

örnək :

```
<script  language="javascript">
    var str2 = "bu bir yazidir";
    document.write(str2 + "<br>");
    str2 =  str2.substring(7,11);
    document.write(str2+ "<br>");
</script>
```

## Simin karakterlərini kiçik və böyük formaya çevirmək :

*toUpperCase()* üsulu simdə olan kiçik karakterləri böyük karakterə çevirər və *toLowerCase()* üsulu böyük karakterləri kiçik formaya çevirər.

birinci örnək :

```
<script  language="javascript">
    var sim1= "Bu bir simdir";
    var sim2=sim1.toUpperCase();
    document.write(sim2);
</script>
```

birinci örnək :

```
<script  language="javascript">
    var sim1= "Bu BiR SimDir";
    var sim2=sim1.toLowerCase();
    document.write(sim2);
</script>
```

## Simi uyğunlaşdırmaq (matches) :

bir ya neçə karakterin olmasini bir simin içində bilmək üçün *match()* üsulundan istifadə edərik. bu üsul parametrdə yazılan karakterləri simin içində axtarar və əgər onun tayın tapsa silsilə formasında qaytarar.

örnək :

```
<script  language="javascript">
      var sim= "azdcafAJkADSC";
     var arrx= sim.match(/[c-f]/g);
     document.write(arrx);
</script>
```

Üstdəki örnəkdə *match()* üsulu *c/d/f* karakterlərin simin içində axtarar və əgər tapsa *arrx* silsiləsinə yazar.

başqa örnək :

```
<script  language="javascript">
      var sim= "azdcafAJkADSC";
     var arrx= sim.match(/[c-f]/g);
     document.write(arrx[0]);
</script>
```

Üstdəki örnəkdə çünki *arrx* bir silsilədir biz ilk karakteri seçmişik ki bu örnəkdə *d* olacaq.

əgər istəsək *match()* üsulunun bir ya neçə karakterdən başqa karakterlərini seçək ^ əlamətindən parametrlərin ilkində istifadə edərik bu sözü açıqlama üçün bir örnək :

```
<script  language="javascript">
      var sim= "azcafAJkADSC";
     var arrx= sim.match(/[^c-f]/g);
     document.write(arrx);
</script>
```

## İki simi müqayisə edmək (compare) :

Bu işi qörmaq üçün *localeCompare()* üsulundan istifadə edərik, bu üsul iki simin karakterlərini sol tərəfdən bir bə bir (one by one) müqayisə edər karakterlərin sırası bu üsulun baxışından aşağadakı kimindir :

0123456789aAbBcCdDeEfF..... yYzZ

əslində bu üsul *str1.localeCompare(str2)* formada yazılar, *str2* ikinci simdir ki *str1* (birinci siminən) müqayisə edilir. bu üsul üç miqdar qaytarar *0/1/-1* ,əgər ikinci simin karakterləri birinci simin karakterlərindən sırada qabaq olsa *1* miqdarı və əgər bərabər olsa *0* miqdarı və əgər ikinci simin karakterləri (sol tərəf üstünlükdür) birinci simin karakterlərindən sonra olsa *-1* miqdarı qayıtar. bir neçə örnək bu sözləri açıqlama üçün.

birinci örnək:

```
<script  language="javascript">
      var sim1= "abda";
      var  sim2= "abcf";
      var str= sim1.localeCompare(sim2);
      document.write(str);
</script>
```

Üstdəki örnəkdə ikisidə *a b* karakterləri bərabər dilər ama *d* və *c* fərq edillər və çünki *c* sirada *d* karakterindən qabaqdır *sim2* kiçik sayılır və *1* miqdarı qayıtır.

ikinci örnək:

```
<script  language="javascript">
      var sim1= "abda";
      var  sim2= "abef";
      var str= sim1.localeCompare(sim2);
      document.write(str);
</script>
```

Üstdəki örnəkdə *e* karakteri ikinci simidə *d* karakteriynən müqayisə olunur və çünkü ikinci karakter kiçikdir *-1* miqdarı qayıtır.

üçüncü örnək:

```
<script  language="javascript">
      var sim1= "abcd";
      var  sim2= "abcd";
      var str= sim1.localeCompare(sim2);
      document.write(str);
</script>
```

Üstdəki örnəkdə iki simin miqdarı bərabərdir və *0* qayıtar.

## Karakter kodunu karakterə çevirmək :

Hər karakter js dilində bir xas kodu var ki onluq rəqəmdier (decimal) , bu rəqəmlər *ascii* kodlaşdırma sistemində 0-dən başlanır və 255-də qutulur və başqa kodlaşdırma sistemi *utf-8* sistemidir.

js dilində string sinifində *fromCharCode()* adlı bir üsul karakter kodların karakterə çevirə bilər, bu üsul bir ya neçəkarakterin kodların çevirə bilər

birinci örnək :

```
<script  language="javascript">
       aa=String.fromCharCode(77);
       document.write(aa);
</script>
```

ikinci örnək :

```
<script  language="javascript">
       aa=String.fromCharCode(77,105,108,97,100);
       document.write(aa);
</script>
```

## Karakteri karakter koduna çevirmək :

əgər istəsək karakterləri js dilində onluq rəqəmə çevirək *charCodeAt()* üsulundan istifadə edərik, bu yolda biz simi bir şey sanırıq.

birinci örnək :

```
<script  language="javascript">
    var str="M";
    en=str.charCodeAt();
    document.write(en);
</script>
```

əgər karakterlərin sayı bir-dən çox olsa bir parametr parantezin içində yazarıq ki biz seçən karakterın indeksin müəyyən edər.

ikinci örnək :

```
<script  language="javascript">
     var strs="Milad";
     en=strs.charCodeAt(3);
     document.write(en);
</script>
```

## Tarih (Date) :

Js dilində tarihə nail olmaq üçün *Date()* adlı bir şey var ki onun vasitəsi ilə ay,gün, saat, dəqiqə və saniyə bilmək olar.

örnək :

```
<script  language="javascript">
    var tari = new Date();
    document.write(tari);
</script>
```

diqqət edin ki new sözi şeyləri çağırmaqında yazılar və görsədər (göstərər) ki bir yeni şey işlənir.

Tarih şeyinin bir neçə üsulu var ki onların vasitəsi ilə ay,gün, saat ,dəqiqə və saniyə təkcə tapa bilərik.

Birinci örnəkdə *getDay()* üsulundan istifadə olunub və həftənin günün müəyyən edir məsəl üçün bazar ertəsi (Monday) 2 olar, 0-dən başlanır və 6-da qutulur.

örnək :

```
<script  language="javascript">
   var date = new Date();
   var day=date.getDay()
   document.write(day);
</script>
```

ikinci örnəkdə ayın günü müəyyən olur, əgər *getDay()* üsulunun yerinə *getDate()* üsulun yazaq, ayın neçənci günü müəyyən olar.

örnək :

```
<script  language="javascript">
    var date = new Date();
    var day=date.getDate()
    document.write(day);
</script>
```

Üçüncü üsul *getFullYear()* üsuludur və ili bizə qaytarar, məsəl üçün 2015

örnək :

```
<script  language="javascript">
    var date = new Date();
    var day=date.getFullYear();
    document.write(day);
</script>
```

Başqa üsullar tarih şeyində var ki adları aşağada yazılıb

| | |
|---|---|
| getHours() | Saatı qaytarır (0-23) |
| getMilliseconds() | Milisaniyəni qaytarır (0-999) |
| getMinutes() | Dəqiqəni qaytarır (0-59) |
| getMonth() | Ayı qaytarır (0-11) |
| getSeconds() | Saniyəni qaytarır (0-59) |
| getTime() | milisaniyə qaytarir (Jan 1, 1970 ildən indiyəcan) |
| date.getUTCHours() | Orta Qrinviç saatın qaytarır (0-23) |

## Riyaziyyat (Math) :

*Math()* adlı şey yaradıcılara bu imkani verir ki riyazi məsələri yerinə yetirələ. bu şeydə çoxlu üsullar və xüsusiyyətlər var ki onları tanıdırırıq. *Math()* şeyi tanıdırmaq üçün *new* sözü lazım dəyir.

## Math şeyinin xüsusiyyətləri :

Xüsusiyyətlər tanımaq üçün onların sonunda baxarıq çünki parantez əlaməti təksə üsulların sonunda olar və xüsusiyyətlərin sonunda heç nə olmaz,aşağadakı örnəkə diqqət edin :

var a=Math.PI; (PI bir xüsusiyyətdir)

var a=Math.log(); (log() bir üsuldur)

ilk örnək Eulnerin konstantını qaytarır ki təxminən 2.718 olar

örnək :

```
<script  language="javascript">
    var m1 =Math.E;
    document.write(m1);
</script>
```

Başqa xüsusiyyətlər aşağada yazılıb :

| | |
|---|---|
| Ln2 | 2-nin doğal loqaritmini qaytarar(təxminən 0.693) |
| Ln10 | 10-nun doğal loqaritmini qaytarar(təxminən 2.302) |
| LOG2E | E-nin 2 əsasında olan loqaritmi qaytarar(təxminən 1.442) |
| LOG10E | E-nin 10 əsasında olan loqaritmi qaytarar(təxminən 0.434) |
| PI | Pi konstant sayını qaytarar(təxminən 3.1415) |

SQRT1_2 1/2 yada 0.5-mın 2 köklü inteqralın qaytarar

SQRT2 2-nin 2 köklü inteqralın qaytarar(təxminən 1.414)

## Math şeyinin üsulları :

ilk üsul ki geniş istifadəsi var *random()* üsuludur bu üsul 0 və 1 sayının arasında bir rastgələ say qaytarar.

örnək :

```
<script  language="javascript">
   var ra =Math.random();
   document.write(ra);
</script>
```

əgər istəsək bu rasgələ say biz istiyən iki rəqəmin arasında ola onda aşağadakı örnək kimi davranarıq, ilk say nihayət sayı bilindirir və ikinci say başlanış sayı məsəl üçün aşağadakı örnəkdə 5 və 2 arasında olan rəqəmlər qaytarar.

örnək :

```
<script  language="javascript">
   var  ran =Math.random()*5+2;
   document.write(ran);
</script>
```

əgər istəsək üzən ədəd olmasın və tam ədəd olsun *floor()* üsulundan istifadə edərik. bu üsul üzən hissəni silər və tam hissə qalar.

örnək :

```
<script  language="javascript">
    var tes =Math.floor(Math.random()*5+2);
    document.write(tes);
</script>
```

Başqa üsul riyaziyyar şeyində *min()* və *max()* üsuludur, bu üsullar kiçik və böyük rəqəmi bir neçə rəqəmin arasından qaytarar.

örnək :

```
<script  language="javascript">
        var max=Math.max(5,2,9);
        document.write(max);
</script>
```

Bu örnəki bu formada yazmaq olar :

```
<script  language="javascript">
        var   ara=[5,2,9]
        var   y=ara[0];
        for(i=0;i<ara.length;i++){
            if(ara[i]>y){
                y=ara[i];
            }
        }
        document.write(y);
</script>
```

Başqa üsullar *Math* şeyində aşağadakı üsullardır.

abs(x) bu üsul mütləq miqdarı qaytarar(məsəl üçün *Math.abs(-2)* 2 olar)

acos(x) bir sayın arkkosinusun qaytarar

asin(x) bir sayın arksinusun qaytarar

atan(x) bir sayın arktangentin qaytarar

atan2(y,x) iki sayın arasında olan açını,radian miqdarında qaytarar

ceil(x) üzən sayı yuxarı yaxın miqdarına yumrular

cos(x) bir sayın kosinusun qaytarar

exp(x) $e^x$ miqdarın qaytarar

floor(x) üzən sayı aşağı yaxın miqdarına yumrular

log(x) E əsasında olan bir rəqəmin miqdarın qaytarar

pow(x,y) x-i y gücündə olan miqdarı qaytarar

round(x) bir üzən sayı ən yaxın saya yumrular

sin(x) bir sayın sinusun qaytarar

sqrt(x) bir sayın 2 köklü inteqralın qaytarar

## Operatorlar :

Js dilində neçə operator var ki onlarn vasitəsi ilə bir para işlər verilərin üstündə görmək olar, bir neçə operator aşağada yazılıb :

\+ iki sim və ya iki say miqdarı bir birinə artar(örnək üçün 2+2=4)

\- iki say miqdarı bir birindən çıxar

/ bir sayı başqa saya bölər(örnək üçün 6/2=3 ,6-nı 2 yerə bölür)

* iki sayı bir birinə çarpar(örnək üçün 2*4=8)

% bir sayı başqa saya bölünmə artıqlıqını qaytarar(7%2=1)

= sağ tərəfində yazılan miqdarı solundakı tərəfə təyin edər

+= sağda yazılan miqdarı soldakına artar(əslində bucur a=a+b)

-= sağda yazılan miqdarı soldakına çıxar(əslində bucur a=a-b)

> soldakı miqdar sağdakından çoxdur(7>3)

< sağdakı miqdar soldakından çoxdur(3<7)

>= soldakı miqdar sağdakından çoxdur və ya birboydur(7>=3 ya 3>=3)

<= soldakı miqdar sağdakından azdır və ya birboydur (3<=7 ya 7<=7)

++ bir nümrə sayın miqdarına artar(x=3 ; x++ ; indi x-in miqdarı 4 olub)

-- bir nümrə sayın miqdarınadan çıxar(x=3;x--;indi x-in miqdarı 2 olub)

== iki miqdar birboydur(x=3;y=3;x==y ikisi bərabər)

!= iki miqdar birboy dəyir(x=4;y=5;x!=y ikisi birboy dəyir)

<< biti sola dəyişiklik edir,bayneri kodi(örnək üçün 6 sayının bayneri kodi 110 olar və əgər 1 bit sola dəyişiklik edək soldan bir bit silinər və sağına bir 0 bit izafə olunar məsəla 6<<1 onda 4==100 olar)

>> biti sağa dəyişiklik edir,bayneri kodi(örnək üçün 5 sayının bayneri kodi 101 olar və əgər 1 bit sağa dəyişiklik edək sağdan bir bit silinər və soluna bir 0 bit izafə olunar məsəla 1>>5 onda 2==010 olar)

& iki miqdarı AND edər,bayneri kodi(örnək üçün 5==101 və 6==110 olsa onda 5&6 miqdarı 100 olar ki onluq rəqəmdə bərabəri 4 olar.diqqət edin ki hər bit iki bayneridə təkcə qarşında olan bitinən AND olar $\begin{cases} 5 & 101 \\ 6 & 110 \end{cases}$)

| iki miqdarı OR edər,bayneri kodi(örnək üçün 5|6 ki 7 olar)

^ iki miqdarı XOR edər,bayneri kodi(örnək üçün 5|6 ki 3 olar)

~ bir miqdarı NOT edər,bayneri kodi(x=1;~x; x miqdarı -2 olar)

! bir miqdarı NOT edər(x=true;!x; x miqdarı false olar)

&& iki miqdarı AND edər(x=1;y=0;z=x&&y; miqdarı 0 olar)

|| iki miqdarı OR edər(x=1;y=0;z=x&&y; miqdarı 1 olar)

in bu operator silsilədə indeksin olmasını bilindirər(örnək üçün a=[3,5,1] ;b=3 in a ; b miqdarı false olur çünki təkcə 0,1 və 2 indeksi var)

=== iki miqdar bərabər ola,yeni brovzerlərdə tanınıb

!== iki miqdar bərabər olmuya,yeni brovzerlərdə tanınıb

typeof bu operator bir dəyişilənin novünü bilindirər

eval() bu operator bir js koduni icra edər

## HTML sənədinin ünsürlərinə nail olmaq :

HTML sənədinin tam ünsürlərinə js dilində nail olmaq olar,örnək üçün document şeyi html sənədin içində olan ünsürlərə işarə edir və window brauzer sayfasına işarə edir və history tarıhə bağlanan üsullara işadə edir.

## HTML sənədinin window şeyi :

Bu şeyin çoxlu üsullari və metodları var ki document şeyi özü window şeyini bir xassəsi sayılır. Başqa xassələrin biri *innerHeight* xassəsidir ki sayfanın boyun bizə qaytarır və ikinci xassə *innerWidth* xassəsidir ki sayfanın enin bizə qaytarır. bu sözi açıqlama üçün bir örnək :

```
<script  language="javascript">
   var  boy=window.innerHeight;
   var  en=window.innerWidth ;
   document.write("boyu: "+boy +"</br>"+"eni: "+en);
</script>
```

Başqa xassələr ki istifadəçinin sayfasının ölçüsün bizə qaytarar aşağa örnəkdə yazılıb,bu xassədən faydalanmaq üçün *window* sözünü yazmaq lazim dəyir. diqqət edin ki *document*,*history*,*location*,*navigator* və *screen* şeyləri özləri *window* şeyinin üsulları sayılıllar isə *window* sözünü onların önündə yazmaq lazim dəyil örnək üçün *window.screen.height;* və *screen.height;* fərq edməz.

örnək :

```
<script  language="javascript">
   var  boy=screen.height;
   var  en=screen.width;
   document.write("boyu: "+boy +"</br>"+"eni: "+en);
</script>
```

Window-nun sayfasının xassələri və üsulları :

| | |
|---|---|
| screen.availWidth | istifadəçinin sayfasının mümkün olan eninin ölçüsüni qaytarar |
| screen.availHeight | istifadəçinin sayfasının mümkün olan boyunun ölçüsüni qaytarar |
| screen.colorDepth | bilgisayarın hər bir rəngi göstərən bitləri qaytarır |

screen.pixelDepth bilgisayarın piksel dərinliyi qaytarır

Window-nun adresinin xassələri və üsulları :

location.href cari sayfanın adresin qaytarar

location.hostname veb hastını domenin qaytarar

location.protocol istifadə olunan veb protokolun qaytarar (*http://* ya *https://*)

location.hash bir veb adresinin # olan əlamətinin parçasın göstərər və ya bu parçanı veb adresinə əlavə edər(*location.hash = "parça";alert(location.hash);* bu kodi icra edəndən sonra brauzerin adresinə diqqət edin)

location.host bir hastın adın, adresin portun qaytarar

location.origin protokol,hastın adı,veb adresin portunu qaytarar

location.pathname veb adresinin yoladını(path name) qaytarar

location.port bir veb adresinin port nümrəsin qaytarar

location.search veb adresinin simsoruşunu(querystring) qaytarar və ya simsoruş əlavə edər adresə(*location.search="m=10";* bu kod m=10 simin adresə əlavə edər)

location.assign(x) bu üsul yeni bir sənəd açar(x sözünün yerinə bir veb sayfasının adresin yazarıq örnək üçün *window.location.assign("http://www.google.com")* )

location.reload(x) bu üsul cari sənədi yenidən yüklər(x bir bul

| | |
|---|---|
| | növündəndir və əgər false yazılsa gizliyerdən(catche) yüklənər və əgər true yazılsa server üstündən yüklənər) |
| location.replace(x) | bu üsul cari veb sayfasının adresini yeni veb adresinən dəyişər və *assign()* üsulununan fərqi bundadır ki qayıt(back) düyməsi işləməz(x-in yerinə adres yazarıq örnək üçün *location.replace("http://www.google.com");* ) |

Window-nun brauzerinin xassələri və üsulları :

| | |
|---|---|
| navigator.appCodeName | brauzerin kod adın qaytarar |
| navigator.appName | brauzerin adın qaytarar |
| navigator.appVersion | brauzerin versiyasının bilgisin qaytarar |
| navigator.cookieEnabled | brauzerin kukisinin aktiv olmasını bilindirər |
| navigator.geolocation | istifadəçini en və boy dayrəsinin mövqeyyətin qaytarar |
| navigator.language | brauzerin dilin qaytarar |
| navigator.onLine | brauzerin xət üstündə olmasını bilindirər |
| navigator.platform | brauzerin platformasını qaytarar |
| navigator.product | brauzerin mühərrikini qaytarar |
| navigator.userAgent | istifadəçinin acentinin başlıqın brauzerdən serverə sarı yollar |
| navigator.javaEnabled() | java-nı aktiv olmasını bilindirər |
| navigator.taintEnabled() | veri ləkkələndirməki aktiv olmasini bilindirər |

Window-nun tarix(history) xassələri və üsulları :

history.length; bu xassə historidə olan adreslərin sayın qaytarar

history.back() bu üsul öncəki sayfanı historidən yüklər

history.forward() bu üsul sonradakı sayfanı historidən yüklər

history.go(x) bu üsul sayfan historidən yüklər(örnək üçün *history.go(-3)* üç sayfa cari sayfanı açmamışdan öncəni yüklər və *history.go(2)* iki sayfa cari sayfadan sonranı historidən yüklər)

Window-nun sənəd(document) xassələri və üsulları :

document.baseURI bu xassə sənədin tam adresin qaytarar

document.body bu xassə sənədin body adlı elementin qaytarar

document.close() *document.open()* üsulunun təvəsutuynan açılan axını(stream) bağlar

document.doctype sənədin növünü qaytarar(html)

document.documentMode cari sənəddə brauzerin oxumaq modunu qaytarar(fəqət IE)

document.documentURI cari sənədin adresin qaytarar

document.domain yüklənən serverin domenin qaytarar

document.implementation sənədin şey modelasının(DOM) dəstəklənməsini müəyyən edər

document.inputEncoding sənədin karakter şfrləşmə növünü qaytarar

document.lastModified sənədin ən son dəyişmək çağın qaytarar

document.open() bir yeni axın açar taki bir yazı yazılsın

| | |
|---|---|
| document.readyState | sənədin yüklənmə vəziyətin qaytarar |
| document.referrer | cari sənədin yüklənən adresin qaytarar |
| document.title | sənədin adın qaytarar |
| document.URL | sayfanın bütün veb adresin qaytarar |
| document.write() | HTML ifadələrin yazar və ya js kodların sənəddə icra edər |
| document.writeln() | *document.write()* kimidir isə bir yeni xəttdə yazar |
| document.cookie; | kukilərin hamısın qaytarar(kuki bir yaddaşdır ki brauzer hər veb sayfasına ixtisas verir və bir yazı onun içində yazmaq olar taki vebsayt hər çağ yüklənəndə bu yaddaşlardan istifadə edilsin, bu yaddaşda istifadəçinin adı və tarix və ya şfrəsini saxlamaq olar) |
| document.createAttribute() | bir əlamət düyünü yaradar |
| document.createComment() | bir müəyyən yorum yazısını düyündə yaradar |
| document.createDocumentFragment() | bir boş parçasənəd duyunu yaradar |
| document.adoptNode() | bir başqa sənəddən bir duyunu qəbul edər |
| document.anchors | *<a>* ünsürlərin hamısına ki *name* əlaməti varlarıdı işarə edir(örnək üçün *<a name="ad1">yazi</a>* ünsürü sənəddə, əgər *alert(document.anchors.length;)* skiriptini icra edsək çıxışda *1* görə bilərik) |

| | |
|---|---|
| document.applets | *<applet>* ünsürlərinə işarə edir( *document.applets.length;* bu təqlərin sayların qaytarar) |
| document.documentElement | bir sənədin sənəd ünsürün qaytarar(örnək üçün *document.documentElement.nodeName;* kodi *<html>* təqin qaytarar) |
| document.domConfig | sənədin DOM quruluşun qaytarar |
| document.embeds | sənədin tüm *<embed>* təqlərini qaytarar |
| document.forms | sənədin tüm *<form>* ünsürlərin qaytarar |
| document.getElementsByClassName() | bir düyün siyahısında sinifinən müəyyən olunan tüm ünsürlər qaytarar |
| document.getElementsByName() | bir düyün siyahısında adınan müəyyən olunan tüm ünsürlər qaytarar |
| document.getElementsByTagName() | bir düyün siyahısında təqinən müəyyən olunan tüm ünsürlər qaytarar |
| document.head | sənədin *<head>* ünsürün qaytarar |
| document.images | sənəddə tüm *<img>* ünsürlərin yığımın qaytarar |
| document.links | sənəddə *<a>* və *<area>* href əlaməti olan tüm ünsürlərinin yığımın qaytarar |
| document.scripts | sənəddə tüm *<script>* ünsürlərin yığımın qaytarar |
| document.importNode() | ayrı sənəddən bir düyün idxal edmək |

| | |
|---|---|
| document.querySelector() | sənəddə ilk ünsür ki müəyyən olan CSS seçicisinən uyğunlaşır qaytarır |
| document.querySelectorAll() | sənəddə statik duyun siyahısında tüm ünsürləri ki müəyyən olan CSS seçicisinən uyğunlaşır qaytarır |
| document.normalize() | boş mətn düyünlərin silib və tüm qarışıq düyünləri yapışdirar(birləşdirər) |
| document.normalizeDocument() | boş mətn düyünlərin silib və tüm qarışıq düyünləri yapışdirar |
| document.removeEventListener() | sənəddə bir hadisə baş verən gözləyən üsulu silər( *addEventListener()* üsuluynan birlikdə gələr) |
| document.renameNode() | müəyyən olunan duyunun adın dəyişər |
| document.strictErrorChecking | səhv yoxlama başverməsin və ya başverməməsin qaytarar |

sənədin başqa üsullarının biri *getElementById()* üsuludur, bu üsul sənəddə bir təyinləməli(id) ünsürə işarə edər. bu üsulun vasitəsi ilə ünsürlərin xüsusiyyətlərinə çatmaq olar.

örnək:

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

```
    </head>
    <body>
        <p id="abc" style="color:red "  onclick="fun() "  >bunu tikle</p>
        <script>
            function fun() {
                var  a=document.getElementById("abc");
                a.style.color="blue";
            }
        </script>
    </body>
</html>
```

Üstədəki örnəkdə *a.style* paraqrafın style xüsusiyyətinə işarə edir və *.color* bu xüsusiyyətin rənginə işarə edir və başqa rənq ona ixtisas verir. sənədin başqa üsullarının biri *addEventListener()* üsuludur, bu üsul sənəddə baş verən bir hadisəni gözlər, məsəl üçün bir tikləməni və ya mosun(mouse) hərəkətini, bu hadisə baş verən an bir funksiyani çağırar ta ki icra olsun.

örnək üçün açılan sayfada tikləyin:

```
<script  language="javascript">
        function  funksiya() {
                alert("tiklenme");
        }
        document.addEventListener("click", funksiya);
</script>
```

Bu üsul sənədin bir parçasınada baş verən hadisəni gözləyə bilər, üstdəki örnəkdə tüm sayfada olan tiklənməni hiss edir isə başqa şəkildə

sayfanın bir parçasında baş verən tikləməyi hiss edər.

örnək :

```
<html>
   <head>
   </head>
   <body>
      <p id="idx">bunu tikle</p>
      <script>
         function funk() {
           alert("mesaj");
         }
         var  a=document.getElementById("idx");
         a.addEventListener("click", funk);
      </script>
   </body>
</html>
```

Üstdəki örnəkdə *<p>* təqi bir paraqrafa işarə edir ki onun *id*-si *idx* sözüdi(*id* ya təyinləmə, bir şeyə sənəddə işarə edmək üçün lazimdir) a adlı dəyişilən sənəddə olan idx təyinləməsinə işarə edir.

Başqa üsul sənəd şeyində *document.createElement()* üsuludur, bu üsul bir yeni ünsür sənəddə yaradır və *document.createTextNode()* üsulu bir yazı düyünü yaradır və *document.body.appendChild(x)* yaradılan düyünü vasitəsi ilə *document.createElement()* düzələn ünsürə sənədin *body* parçasına izafə edər.

örnək:

```
<html>
```

```
<body>
    <script>
        var duyme = document.createElement("BUTTON");
            var yazi = document.createTextNode("duyme yazi");
            duyme.appendChild(yazi);
            document.body.appendChild(duyme);
        </script>
      </body>
</html>
```

Windows-un çoxlu üsullari var ki onların bir azı üstdə yazılıb və bir azı aşağıda yazılıb:

| | |
|---|---|
| alert() | bir avariyal mesaj və OK düyməsi göstərər |
| atob() | base-64 növündə olan verini sim növünə çevirər |
| blur() | fukusu cari pəncərədən silər |
| btoa() | simi base-64 növünə kodlaşdırar |
| clearInterval() | *setInterval()* üsulunun vasitəsi ilə yaranan taymei silər |
| clearTimeout() | *setTimeout()* üsulunun vasitəsi ilə yaranan taymei silər |
| close() | cari pəncərəni bağlar |
| confirm() | dialoq pəncərəsində bir mesaj və OK düyməsi və Cancel düyməsi göstərər |
| createPopup() | pop-up pəncərəsi yaradar |
| focus() | cari pəncərədə fukus edər |
| moveBy() | pəncərəni cari yerinə nisbət tərpəştirmə(hərəkət vermək) |
| moveTo() | pəncərəni müəyyən edən yerinə tərpəştirmə |

| | |
|---|---|
| open() | yeni brauzer pəncərəsi açmaq |
| print() | cari pəncərənin məzmunun çap edmək |
| prompt() | istifadəçidən bir giriş yazı almaq üçün dialoq pəncərəsi |
| resizeBy() | müəyyən olan pikselə pəncərənin ölçüsün dəyişmək |
| resizeTo() | müəyyən olan en və boya pəncərənin ölçüsün dəyişmək |
| scroll() | bu üsul yerinin *scrollTo()* üsuluna verib |
| scrollBy() | müəyyən piksel sayıla sənədi skurul edmək |
| scrollTo() | müəyyən kordinata sənədi skurul edmək |
| setInterval() | bir funksiya və ya kodi müəyyən milisaniyə aralıkının əsasında icra edər(icra tekrar olar) |
| setTimeout() | bir funksiya və ya kodi müəyyən milisaniyə taymından sonra yalnız bir kəz icra edər |
| stop() | pəncərənin yüklənməsini dayandırmaq |

## Bir şəkil sənədə izafə edmək :

ilk yol bir şəkili sənədə izafə edmək yolu *innerHTML* xüsusiyyətidir, aşağada olan örnəkdə *sh.jpg* adlı bir şəkil sənədə izafə olunar(bir şəkil HTML faylızın yanına köçürün və onun adın yazın).

örnək:

```
<html>
  <head>
    <script>
      function fun(id) {
        var  a=document.getElementById(id);
```

```
            a.innerHTML="<img src='sh.jpg'>";
        }
    </script>
</head>
<body>
    <p id="t1" onclick="fun(this.id)" >bunu tikle</p>
</body>
</html>
```

Üstdəki örnəkdə əgər paraqırafın üstündə tikləsək *fun()* adlı funksiya icra olar, *this.id* cari şeyın təyinləməsinə işarə edir ki burda bir paraqırafdır, *a.innerHTML* paraqırafa bir şey izafə edir ki burda bir şəkildir, deməliyik ki bir yazı və ya iskiriptdə bu xüsusiyyətinən izafə oluna bilər, *src* bu şəkilin adresin müəyyən edir ki iki formada ola bilər ki burda yerli(local) formasından istifadə olunub yanı şəkil gərək *HTML* faylının yanında ola və ya bir qovluqun(folder) içində ola ki onda bu *folder adi/sh.jpg* formada yazılar, başqa yol adres vermaq yolu veb adresindən istifadə edməkdir ki məsəl üçün üstdəki örnəkdə *sh.jpg* adının yerinə *http://www.google.com/intl/en_com/images/logo_plain.png* veb adresin yazaq, bu yolda *http://* porotokolun yazmaq gərəklidir.

ikinci örnək:

```
<html>
```

```
    <head>
        <script>
            function fun(id) {
            var  a=document.getElementById(id);
            a.innerHTML=a.innerHTML+"<img src='sh.jpg'>";
            }
        </script>
    </head>
    <body>
        <p id="t1" onclick="fun(this.id)" >bunu tikle</p>
    </body>
</html>
```

üstdəki örnək öncəki örnəkinən fərqi bundadi ki *a.innerHTML=a.innerHTML* kod bir yazı və ya şəkili cari şeyə izafə edəndə öncəki yazı və ya şəkillər silmir, bu kodi *a.innerHTML+=* formada yazmaq olar və fərq edməz.

Şəkili idxal edməkin ikinci yolu *document.write()* kodudur, bu kod bir şəkili sənədin body partısına izafə edər.

örnək:

```
<html>
    <head>
        <script>
            function fun() {
              document.write("<img src='sh.jpg'>");
            }
```

```
        </script>
    </head>
    <body>
        <p id="t1" onclick="fun()" >bunu tikle</p>
    </body>
</html>
```

Üçüncü yolda *appendChild()* üsulunda istifadə edərik, bu üsuldan istifadə edmək üçün gərək ilkdə *new Image()* şeyinən bir şəkil yaradaq və sonradan ona adresinin əlamətin(src attribute) əlavə edək.

örnək:

```
<html>
    <head>
        <script>
            function fun(id) {
              var img = new Image();
              img.src="sh.jpg";
              var  a=document.getElementById(id);
              a.appendChild(img);
            }
        </script>
    </head>
    <body>
        <p id="t1" onclick="fun(this.id)" >bunu tikle</p>
    </body>
</html>
```

ikinci örnək:

```
<html>
   <head>
      <script>
         function fun(id) {
           var img = new Image();
           img.setAttribute("src","sh.jpg");
           var  a=document.getElementById(id);
           a.appendChild(img);
         }
      </script>
   </head>
   <body>
      <p id="t1" onclick="fun(this.id)" >bunu tikle</p>
   </body>
</html>
```

Üstdəki örnək ilk örnəkinən fərqi yoxdur və sadəcə *setAttribute()* üsulundan istifadə edilib ki bir əlaməti şəkilə əlavə edə bilər, bu əlamətlər şəkilin adresi(src) və ya onun en və boyu ola bilər məsəl üçün bu örnək:

```
<html>
   <head>
      <script>
         function fun(id) {
```

```
            var img = new Image();
            img.setAttribute("src","sh.jpg");
            img.setAttribute("height", "400");
           img.setAttribute("width", "500");
           var  a=document.getElementById(id);
           a.appendChild(img);
          }
      </script>
   </head>
   <body>
      <p id="t1" onclick="fun(this.id)" >bunu tikle</p>
   </body>
</html>
```

dördüncü yolda *createElement* üsulundan istifadə edərik, bu üsul bir şəkil sənəddə yaradar.

örnək:

```
<html>
   <head>
      <script>
         function fun(id) {
            var img =document.createElement("img");
            img.setAttribute("src","sh.jpg");
            var  a=document.getElementById(id);
            a.appendChild(img);
         }
      </script>
```

```
    </head>
    <body>
        <p id="t1" onclick="fun(this.id)" >bunu tikle</p>
    </body>
</html>
```

beşinci yolda *input* düyməsindən istifadə edərik. diqqət edin HTML sənədizin yeri şəkillərin yanında olmalıdır.

örnək:

```
<html>
    <head>
        <script>
            function fun(id,bd) {
                var  a=document.getElementById(id).value;
                var  b=document.getElementById(bd);
                b.innerHTML=b.innerHTML+"<img src='"+a+"'>";
            }
        </script>
    </head>
    <body id="bd">
            <input type="file" id="a1"/>
            <p onclick="fun('a1','bd')" >bunu tikle</p>
    </body>
</html>
```

bu təq *<input  type="file"  id="a1" />* bir fayl seçim təqidir, istifadəçi bu

təqinən bir fayl seçmə bilər, *type="file"* bu təqin növün bilindirir məsəl üçün *type="text"* bir yazı növünü bilindirir və başqa növlər ...

*id="a1"* bu təqin təyinləməsin bilindirir. */>* təqin qutuluşun bilindirir, diqqət edin *input* təqi bir yalnız təqdir.

bu təq * bunu tikle * bir paraqraf bilindirir, diqqət edin *p* bir cüt təqdir və qutuluşu * * təqiynən bilinər.

*onclick* elanı paraqrafın üstündə tikləyən anda bir funksiya çağırar(icra edər), bu funksiyanın iki parametri var ki ilk parametr fayl seçən təqinin təyinləməsidir və ikinci parametr *body* təqinin təyinləməsidir.

*var a=document.getElementById(id).value* fayl seçimin seçib və *value* xüsusiyyətinin vasitəsi ilə bu seçimdə olan miqdarına(burda şəkilin adıdır) yetişmək olar, bu miqdar *a* dəyişilənin içində oturar.

*var b=document.getElementById(bd)* sənədin *body* adlı təqini seçib və *b* dəyişilənə göndərər(refer)

*b.innerHTML* bir yazı və ya kod və ya şəkil sənədin işarə olunan ünsürə izafə edər(burda body ünsürü və ya təqi).

*b.innerHTML=b.innerHTML+* yanı cari yazılar və şeylər silinməsin və sadəcə bir yeni miqdar öncəki yazılara və şeylərə izafə olunsun(*b.innerHTML+=* formada yazıla bilər).

*"<img src='"+a+"'>"* bir şəkil yaraldar ki onun adı a dəyişiləninin içində oturub, *"* əlamətinin arasında yazılan yazılar sim sayılıllar çünkü a bir sim dəyir və dəyişiləndir ona görə *"<img src=' "* formada bu simi qutarıb və *+* əlaməti ilə a dəyişilən izafə edib və sonradan *+* əlaməti ilə bir yeni sim

" '>" onun sonuna yapışdıracağıq. ' əlamətləri buna görədir ki *src* xassəsinə bir miqdar vemaq üçün o miqdarı bu əlamətlərin və ya " əlamətlərinin içində yazmaq gərəklidir. çünku burda " əlaməti öncədən istifa olunub biz ' əlamətindən *src* xassəsində istifadə edərik əlbət bu *"<img src=\" "+a+"\">"* formada yazmaq olar bu yolda \ əlaməti " əlamətinin sim növünda olmasını bilindirir.

əslində bu kod *<img src='adi.jpg'>* formasına sənədə izafə olunur və əgər ikinci formada yazılsa bu *<img src="adi.jpg">* formasına sənədə izafə olunar.

diqqət edin ki *src* və *type* və *id* hamısı xüsusiyyət tanınıllar.

## Bir yazı sənədə izafə edmək:

yazı izafə edmək şəkilin idxal yoluna taydır və *img* təqinin yerinə yazı yazarıq.

örnək:

```
<html>
   <head>
      <script>
         function funk(){
            var  c=document.getElementsByTagName("body");
            c[0].innerHTML+="yazi";
         }
      </script>
   </head>
   <body>
```

```
        <p onclick="funk()" >bunu tikle</p>
    </body>
</html>
```

üstdəki örnəkdə document.getElementsByTagName("body") kodi body təqin seçib və *c* dəyişilənə göndərər(*img* təqin seçmək üçün sadəcə *body* yerinə *img* sözündən istifadə edərik və *div* seçmək üçün *div* sözündən) diqqət edin *c* dəyişiləni bir silsilədir və təqlər *0* indeksindən başlayıb və... indeksə çata bilər.

*c[0].innerHTML+="yazi"* bir yazını *c* dəyişilənin 0-ci ünsürünə izafə edər.

## Sənədin ünsürlərinin xassələrin dəyişmək və redak edmək :

id xassəni dəyişmək üçün bir təq seçib və onun id miqdarın dəyişərik.

örnək:

```
<html>
    <head>
        <style>
        #a1{
            color:blue;
          }
        #a2{
            color:red;
            font-size:20px;
          }
      </style>
      <script>
```

```
        function funk(st){
          var  paraq=document.getElementsByTagName("p");
          paraq[0].id=st;
        }
    </script>
  </head>
  <body>
      <p id="a1" onmouseover="funk('a2')" onmouseout="funk('a1')" >bunu tikle</p>
  </body>
</html>
```

Diqqət edin ki css(cascade style sheet) kodları style təqlərinin arasında (outline) yazılallar və ya təqin özündə(inline) *style=" "* formasında yazılallar, yazılar və ya görsəl(visual) şeyləri sayfada redact edmək olar və ya yeni görsəl yaradmaq olar ki yazıların rəngi və ya ölçüsü bu kodlardandılar.

*#a1* bir *a1* adlı təyinləmənin stilin bilindirir və *#a2* bir *a2* adlı təyinləmənin stilin.

*onmouseover="funk('a2')"* mos paraqrafın üstünə gedən anda *funk* adlı funksiyanı çağırır və *a2* miqdarın parametr unvanında bu funksiyaya yollayır.

*onmouseout="funk('a1')"* mos paraqrafın üstünə çəkilən anda *funk* adlı funksiyanı çağırır və *a1* miqdarın parametr unvanında bu funksiyaya yollayır.

*var  paraq=document.getElementsByTagName("p")* sənəddə olan paraqrafları

seçib və *paraq* dəyişilənə göndərir.

*paraq[0].id=st* ilk paraqrafı seçib(*paraq[0]*) və onun *id* xassəsinın miqdarın (*id="a1"*) parametrdə olan miqdar qərar verir.

bir başqa örnək:

```
<html>
   <head>
      <script>
         function funk(){
            var  paraq=document.getElementsByTagName("p");
            paraq[0].style.color="red";
            paraq[0].style.fontSize="20px";
         }
      </script>
   </head>
   <body>
         <p style="color:blue;font-size:14px" onclick="funk()">bunu tikle</p>
   </body>
</html>
```

*paraq[0].style.color="red"* bu kod paraqrafın ilk ünsürünün *style* xassəsinin *color* xüsusiyyətinə yetişmək üçün istifadə olunur və *paraq[0].style.fontSize="20px"* paraqrafın *font-size* xüsusiyyətinə yetişmək üçün, diqqət edin css-in - əlaməti js dilində yazılmaz və yerinə ikinci sözun(burda size) ilk hərfi(*Size*) böyük formada yazılar, *px* burda *pixel* sözünün ixtisarıdır.

## Karakterləri kodlaşdıran funksiyalar(fəqət ASCII karakterlər) :

Bir neçə tanınmamış karakterləri(örnək üçün ? $ @ və ...) kodlaşdırmaq üşün JS dilində neçə funksiya var, bu tanınmamış karakterlər bir % əlaməti ilə və sonradan HEX(16-lıq) koduynan tanınar.

ilk funksiya *encodeURIComponent()* funkisiyasıdır, bu funksiya bu karakterləri kodlaşdırar: *, / ? : @ & = + $ #*

örnək:

```
<html>
   <head>
        <style>
          .ids{
            color:blue;
            font-size:18px;
            heght:20px;
            width:300px;
            border-radius:8px;
            background-color:rgba(125, 223, 227, 0.33);
          }
        </style>
   </head>
   <body>
     <input class="ids" id="alan" type="text" /></br>
     <input class="ids" id="gosteren" type="text"/>
     <script>
        var sayan=0;
        function funk2(){
                sayan2=sayan+1;
```

```
                var  paraq=document.getElementById("alan").value;
                var  bd=document.getElementById("gosteren");
                var zp=paraq.substring(sayan,sayan2);
                bd.value=bd.value+""+encodeURIComponent(zp);
                sayan++;
        }
        var  duyme2=document.getElementById("alan");
        duyme2.onkeyup=funk2;
    </script>
  </body>
</html>
```

üstdəki örnəkdə *style* təqində *.ids* adlı xassə tüm *ids* adlı sinifləri(class) qabsayır(contain), diqqət edin əgər *.ids* yerinə *#ids* yazaq onda bir sinif yerinə bir təyinləməyə işarə edər.

bu CSS(şəlalə stil vərəqi) kodi *border-radius:8px* bir ünsüsün qıraq və ya sərhəddin istəyən miqdara əyər.

*background-color:rgba(125, 223, 227, 0.33)* -kodi ünsürün fon rəngini üç əsasi *red/green/blue* (qırmız/yaşıl/göy) rənglərinin qarışmasından düzələn rəngə dəyişir, dördüncü rəqəm 0 və 1 arasında olar və tutqunluqu (opacity) bilindirər.

*sayan* dəyişiləni *funk2* funksiyasi hər kəz icra olan çağ bir miqdar artar.

*duyme2.onkeyup=funk2* bu kod hər bir klaviaturanın açarının boşlanmasını gözlər və hadisə baş verən an *funk2* adlı funksiyanı çağırar(icra edər).

*sayan2=sayan+1* bu dəyişilən *sayan* adlı dəyişilənin miqdarindan bir say

çoxlüqun seçər.

*var paraq=document.getElementById("alan").value* bu kod sənəddə *alan* adlı təyinləmənin miqdarın alar və *paraq* dəyişilənin içində oturdar.

*var bd=document.getElementById("gosteren")* bu kod *gosteren* adlı təyinləmənin özün *bd* dəyişilənin içində oturdar(istəyirik bir miqdar bu ünsürə əlavə yazaq).

*var zp=paraq.substring(sayan,sayan2)* bu kod paraq adlı dəyişilənin sim növündə olan miqdarının bir karakterin seçər,*funk2* hər kəz icra olanda bir say bu dəyişilənin miqdarına izafə olar və yeni yazılan karakterin yerinin nümrəsini sol tərəfdən müəyyət edər. birinci parametr başlanış say və ikinci qutuluş saydır(örnək üçün: *var paraq="abcdef";*

*var zp=paraq.substring(5,6);* indi *zp* dəyişilənin miqdarı *f*-dir).

*bd.value=bd.value+""+encodeURIComponent(zp)* bu kod *bd* dəyişilənin miqdarına(dəyər-value) *zp* dəyişilənin *encodeURIComponent()* vasitəsi ilə kodlaşılan miqdarın izafə edər.

funksiyanın son kodi *sayan++* bir say *sayan* dəyişilənə izafə edər.

İkinci funksiya tanınmamış karakterləri kodlaşdırmaq üçün *escape()* funksiyasıdır, bu *?<>:'"{}[]~!#$%^&()=|* karakterlər 16-lıq miqdarınan kodlaşdırar.diqqət edin bu kodlaşdırma funksiya ASCII və UTF sisteminə dəstəkdir.

örnək:

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

```
            .ids{
            color:blue;
            font-size:18px;
            heght:20px;
            width:300px;
            border-radius:8px;
            background-color:rgba(125, 223, 227, 0.33);
            }
            </style>
    </head>
    <body>
      <input class="ids" id="alan" type="text" /></br>
      <input class="ids" id="gosteren" type="text"/>
      <script>
        function funk2(){
                    var  paraq=document.getElementById("alan").value;
                    var  bd=document.getElementById("gosteren");
                    bd.value=escape(paraq);
        }
                   var  duyme2=document.getElementById("alan");
                   duyme2.onkeyup=funk2;
      </script>
   </body>
</html>
```

## Karakterləri kodlaşdıran funksiyalar(ASCII və Unicode) :

Unicode kodlaşdırma sistemi 16 parçadan(bit-dən) istifa edir,diqqət edin bu kodlaşdırma sistemi karakterlər üçün istifadə olunar ki iki növü var. UTF-8 bir kodlaşdırma sistemidir ki bir yazı ya verini ikilik növünə(binary) şifrləşdirər.

Unicode özü iki növü var,birinci növ böyük qütülüş(Big Endian/BE) və ikinci növ kiçik qütülüş(Little Endian/LE).

örnək üçün ‘Azərbaycan’ sözi beş növdə:

HEX kodi: 41-7a-259-72-62-61-79-63-61-6e

Unicode(BE): 0041-007a-0259-0072-0062-0061-0079-0063-0061-006e

Unicode(LE) : 4100-7a00-5902-7200-6200-6100-7900-6300-6100-6e00

Javascript-də(BE): %u0041%u007a%u0259%u0072%u0062%u0061%u0079%u0063%u0061%u006e

HTML-də: Azərbaycan

*diqqət edin HTML-də yazılan kodlar 10-luq rəqəmdilər və başqalar 16-lıq rəqəm{65(DEC)-->41(HEX)}.

örnək:

```
<html>
   <head>
        <meta http-equiv="Content-Type" content="text/html;charset=UTF-16">
       <style>
       .ids{
```

```
        color:blue;
        font-size:18px;
        width:300px;
        border-radius:8px;
        background-color:rgba(125, 123, 227, 0.33);
        }
        </style>
</head>
<body>
 bir yazi yaz :</br>
<input class="ids" id="alan" type="text" /><p id='elan' onclick="funk2()">TIKLE</p>
<textarea class="ids" cols="40" rows="6"  id='goster1'></textarea></br>
<textarea class="ids" cols="40" rows="6"  id='goster2'></textarea>
<script>
    function funk2(){
        karakter="";
        var  paraq=document.getElementById("alan").value;
                    for(i=0;i<paraq.length;i++){
                        kar=paraq.charCodeAt(i).toString(16);
                            while(kar.length<4){
                             kar="0"+kar;
                            }
                            karakter+="%u"+kar;
                    }
                    var  bd=document.getElementById("goster1");
                    bd.innerHTML="Unicode code:\n"+karakter;
                    var  bdh=document.getElementById("goster2");
```

```
                bdh.innerHTML="Unicode char(s):\n"+unescape(karakter);
        }
    </script>
  </body>
</html>
```

bu *kar=paraq.charCodeAt(i).toString(16)* kod bir simin *i* indeksli karakterin seçər və onu 16-lıq rəqəmə çevirib və sim növünən *kar* dəyişilənin içinə qoyar.

Bu 16-lıq kodlar *0*-dən başlanıb və *FFFF* miqdarına çatar,isə Unicode sistemi *4* rəqəm istəyir və əgər bir kod sayı örnək üçün *05D* ola çünkü bu say *3* rəqəmdir xəta baş verər,buna görə *4* dən az olan rəqəmlərin soluna *0* izafə edərik,bu döngü *while(kar.length<4){kar="0"+kar;}* bu işi görər.

bu *karakter+="%u"+kar* kodi 16-lıq rəqəmlərin soluna *%u* əlamətin ki Unicode karakterlərin vebdə əlamətidir izafə edər və hamısın *karakter* dəyişilənin içinə qoyar.

bu *unescape()* funksiya 16-lıq əsasında olan rəqəmləri karakterə çevirər. hər 16-lıq rəqəm *0*-dən başlanıb və *F* rəqəmdə qutular.

| **10-luq (DEC)** | **16-lıq (HEX)** | **2-lik (BIN)** | **8-lik (OCT)** | **//////// ////////** | **10-luq (DEC)** | **16-lıq (HEX)** | **2-lik (BIN)** | **8-lik (OCT)** |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0000 | 0 | ///////// | 8 | 8 | 1000 | **10** |
| **1** | 1 | 0001 | 1 | ///////// | 9 | 9 | 1001 | **11** |
| **2** | 2 | 0010 | 2 | ///////// | 10 | A | 1010 | **12** |
| **3** | 3 | 0011 | 3 | ///////// | 11 | B | 1011 | **13** |
| **4** | 4 | 0100 | 4 | ///////// | 12 | C | 1100 | **14** |
| **5** | 5 | 0101 | 5 | ///////// | 13 | D | 1101 | **15** |
| **6** | 6 | 0110 | 6 | ///////// | 14 | E | 1110 | **16** |
| **7** | 7 | 0111 | 7 | ///////// | 15 | F | 1111 | 17 |

Diqqət edin ki Unicode 4 rəqəm(0000~FFFF) 16-lıqdır və onun 2-lik kodu bu olar *0000 0000 0000 0000* və ya son rəqəmi *FFFF FFFF FFFF FFFF* olar,yanı birlikdə 16 bit.

örnək:

```
<html>
   <head>
         <meta http-equiv="Content-Type" content="text/html;charset=UTF-16">
        <style>
        .ids{
        color:blue;
        font-size:18px;
        width:300px;
        border-radius:8px;
        background-color:rgba(125, 123, 227, 0.33);
        }
        </style>
   </head>
   <body>
 bir yazi yaz :</br>
<input class="ids" id="alan" type="text" /><p id='elan' onclick="funk2()">TIKLE</p>
<textarea class="ids" cols="40" rows="6"  id='goster1'></textarea></br>
<textarea class="ids" cols="40" rows="6"  id='goster2'></textarea>
      <script>
       function funk2(){
          karakter=karakter2="";
          var  paraq=document.getElementById("alan").value;
                    for(i=0;i<paraq.length;i++){
```

```
                kar=paraq.charCodeAt(i).toString(10);
                    karakter+="&#"+kar+";";
                    karakter2+="&# "+kar+";";
                }
                var  bd=document.getElementById("goster1");
                bd.innerHTML="HTML Code:\n"+karakter2;
                var  bdh=document.getElementById("goster2");
                bdh.innerHTML="HTML Char:\n"+karakter;
        }

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

## AJAX (eycəks) :

Bir yeni texnologiya cava skript dilində eycəksidir, bu texnologiya ilə bir veri müştəri tərəfindən server tərəfinə yollaya bilərik. eycəks kodları təhlükə(danger) səbəblərə görə yalnız server üstündə olan veb sayfalarında icra olar yani bilgisayarın üstündə icra olmaz.

Örnək:

```
<html>
<head>
</head>
<body>
```

```
<script language="javascript"  type="text/javascript">
   function  eyceks01(veri){
       ad ="www.google.com/index.php?"+ "veriler="+ veri;
       s1 = new XMLHttpRequest();
       s1.onreadystatechange = cevab1;
       s1.open("GET",ad, true);
       s1.send(null);
  {

   function  cevab1(){
       if (s1.readyState == 4)
       {
              alert(s1.responseText);
       }
   }
</script>
<input   id="yolla01"  type="text" /><p onclick=" eyceks01(yolla01.value)">TIKLE</p>
</body>
</html>
```

Üstdəki örnəkdə *input* təqi bir veri istifadəçidən alır və *TIKLE* düyməsi bu verini *eyceks01* funksiyasına yollur və bu funksiyanı icra edir, *eyceks01* funksiyasında *ad* adlı dəyişilən bu verini bir başqa dəyişilənin içinə(*veriler*) qoyar. diqqət edin soruş əlaməti(*?*) bir *GET* yollayış halətin bilindirir, başqa yol *POST* yollayış halətidir ki onda soruş əlaməti yazmaq lazim dəyir. *index.php* o server tərəfində olan veb sayfasıdır ki bir veri ona sarı yolluruq.

*s1 = new XMLHttpRequest();* kodu bir eycəks şeyin *s1* dəyişilənin içində yazar ki sonradan ondan istifadə edəcəyik.

*s1.onreadystatechange = cevab1;* servərə yolladığımız verinin cəvabı hazır olanda bir funksiya icra edər.

*s1.open("GET",ad, true);* bir serverə bağlayış(connection) kodunu icra edər.

*s1.send(null);* yollayış icra olarş.

*if (s1.readyState == 4)* serverdən qayıtan cəvabın hazir olmasını bilindirər, miqdar gərək *4* ola.

*s1.responseText* serverdən qayıtan cəvab.

## əsas sözlər :

əl tapmaq.....access

açıqlama.....explain

adlandıranmarıq.....We can not named

adsız funksiya.....nameless

altyığım.....subset

axtarış.....search

ayırıcı.....separator

boş.....null-hükümsüz

boşluq.....space

bilgisayar.....computer

bölüm.....part

bul.....boolean

bağlanmaq.....connect

bağlantı.....link

bağlayış.....connection

çeşidləmək.....sort

çevirmək.....convert

dəyər.....value

dəyişilən.....variable

endir.....download

forma.....shape

funksiya.....function

gizliyer.....catche

idxal.....input

ikilik.....binary

karakter.....character

kodlaşdırmaq.....encode

müqayisə.....compare

növ.....kind

nizamlandırmaq.....regularize

onluq rəqəm.....decimal

örnək.....example

parça.....bit

qeyri müəyyən.....undefined

qutarış.....end

qabsayır.....contain

qayıt.....back

redaktor.....Editor

riyaziyyat.....Math

rastgələ.....random

səhifə.....page

sənəd.....document

say.....number-nömrə

sim.....string

simsoruş.....querystring

sinif.....class

sitat.....quote

süzmək.....filter

şərt.....Condition

şəlalə stil vərəqi.....cascade style sheet

şey.....object

şablonuna.....template

şifrlənmiş.....encoded

tiklə.....click

təhlükə.....danger

təq.....TAG

tərs.....reverse

təyinləmə.....id

tarih.....Date

tarix.....history

tam ədəd.....Integer

tutqunluq.....opacity

tutuşdurma.....comparison

uyğunlaşdırmaq.....matches

üstünlüki.....Superiority

üsul.....method

üzən ədəd.....Float

veri.....Data

xassə.....property

yazılım.....software

yapışdırmaq.....Concatenates

yüklənir.....Loading

yüklə.....upload

yoladı.....path name

yollayış.....send

<---Son--->

www.fb/DeEp.CrAcK

maxmilad@gmail.com