berkay\_akar1@saintjim.me

JavaScript

İçindekiler

[Javascript Kodlarını Projeye Dahil Etmek 2](#_Toc98609148)

[Temel JS Komutları 3](#_Toc98609149)

[Veri Tipleri 4](#_Toc98609150)

[Veri Tiplerinin Birbirlerine Dönüşümleri 8](#_Toc98609151)

[String e Dönüşüm: 8](#_Toc98609152)

[String bir ifadeden sayısal bi ifadeye Dönüşüm: 8](#_Toc98609153)

[Operatörler Ve Matematik İşlemleri 10](#_Toc98609154)

[Random Sayı Üretme 11](#_Toc98609155)

[String Fonksiyonları 12](#_Toc98609156)

[DİZİLER 19](#_Toc98609157)

* Javascript dosyalarının uzantısı .js şeklindedir.

# Javascript Kodlarını Projeye Dahil Etmek

JavaScript kodlarını projeye dahil etmek için 3 farklı yöntemimiz vardır.

1. Body Tagı İçerisinde en alt kısma <script> </script> yazmak

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Body içerisine yazılır. Yapısı tıpkı aynı web sayfası içerisinde header kısmında <style> </style> şeklinde tanımlanmış olan css şeklinde düşünülebilir. () Body Tagının bitişinde konumlandırılır.

1. External dosya import işlemi

Body tagının bitişinden hemen önce tanımlanması gerekmektedir. External css dosyası gibi yorumlanabilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. Harici Dosya üzerinden tanımlama (Lokal)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

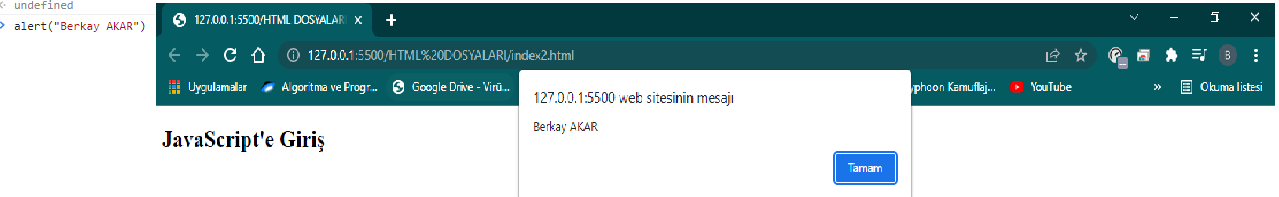
Tavsiye edilen tanımlama tipi body tagının bitim yerinin bir üstüdür. (bu tanımlama sadece uzak sunucuda iken parallel indirmede başlatılması işin kullanılır).

Kısaca üzerinden geçmek gerekirse genellikle body tagının bitiş yerinde tanımlanır

Script tagları içinde yazılan proje içinde src ile external dosyaya erişim sağlanabilir aynı şekilde uzaktaki bir dosya da çekilebilir.

# Temel JS Komutları

* **Alert(“ “) :** Ekrana popup şeklinde uyarı atmamızı sağlar çift tırnağın içi bir string gibi davranır.



* **Console.log(“Deneme”):** Ekran a Çıktı vermek için kullanılan bir komuttr.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Var a : değişken oluşturma için bir tiptir ve tip bağımsız iş yapar. Atılan tüm tipleri sistemde tutmasını sağlar C# taki gibi atanan ilk değer tipini referans alarak iş yapmamaktadır. C# taki gibi dynamic düşünülebilmektedir.

* **Console.clear():** Konsolu temizlemek için kullanılmaktadır.

* **Document.getElementById(“h3”).style.color =”green” :** id si h3 olan(tag değil h3 isimli bir tag (yani id = “h3”)) elementerlin renklerine green isminde bir string atayarak iş yaptık ve rengini değiştirdik. Ağırlıklı olarak sayfa üzerindelki yapılara müdahale amacı ile Document tagı kullanılmaktadır.

Javascript te kodlar noktalı virgül ile bitmektedir. (koymazsan hata vermemektedir. )

* **Var a = [1,2,3,4,5,6,7,8]**

Yukarıdaki örnekte bir dizi oluşturulmuştur. A[1],A[2] gibi index değerleri ile dizi içerisinde gezilebilmektedir. (C# taki gibi tip bagımlı değil içerisindeki item bazında bir tip bagımsız değişkendir.)

* **Typeof degisken\_ismi:** Sistemdeki değişkenin bizlere tip bilgisini çevirmektedir.
* **// , /\*\*/ :**  yorum satırı için işaretlerdir.

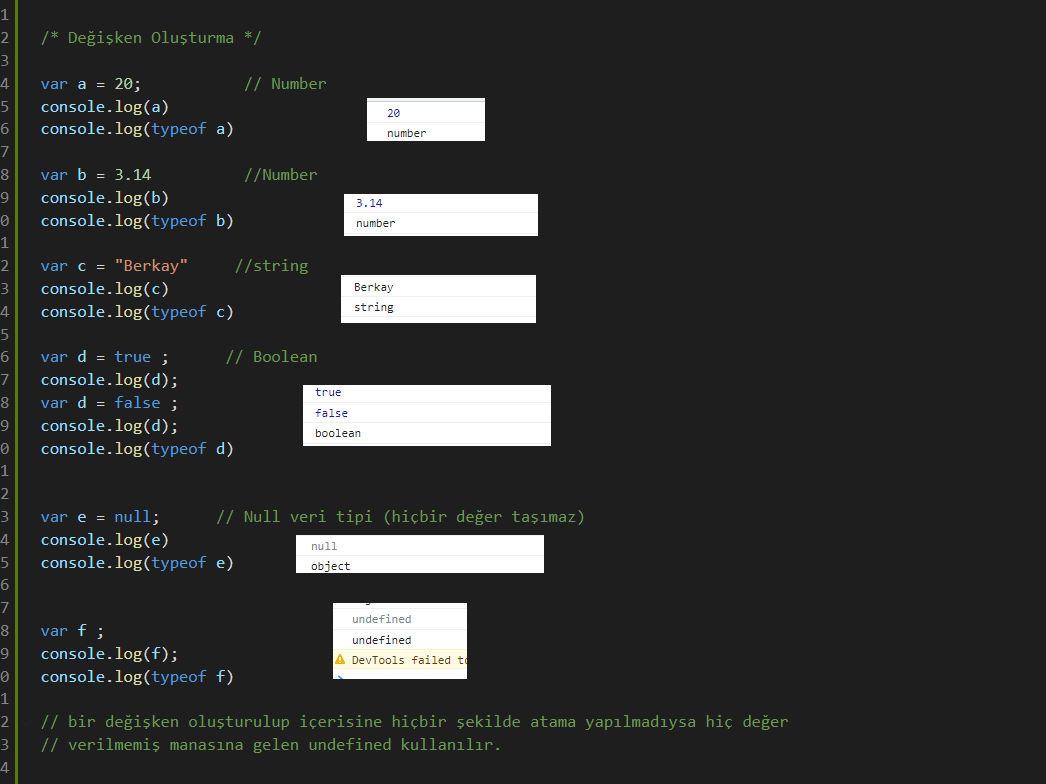
**metin, ekran, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**

# Veri Tipleri

Javascript te değişken tiplerimizi diğer değişkenler ile benzerlik göstermektedir (int float double decimal object …). Javascript te **iki farklı değişken** tipi bulunur. Bunlardan birisi **primitive**, diğeri ise **Referans**

* Primitive veri tipi ilkel veri tiplerine denmektedir. (C# taki int float double char string ….. gibi düşünülebilir.)



* REFERANS VERİ TİPLERİ :

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

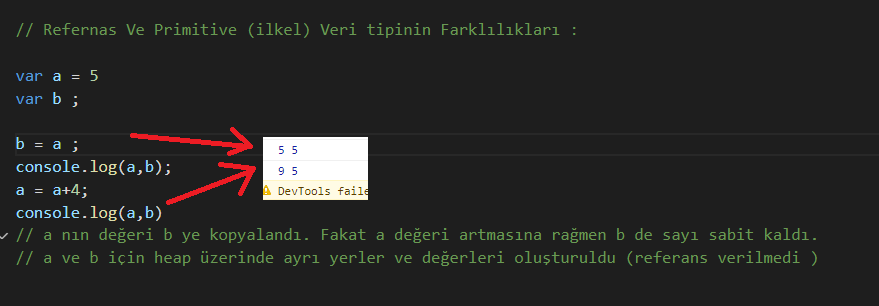
Açıklama otomatik olarak oluşturuldu

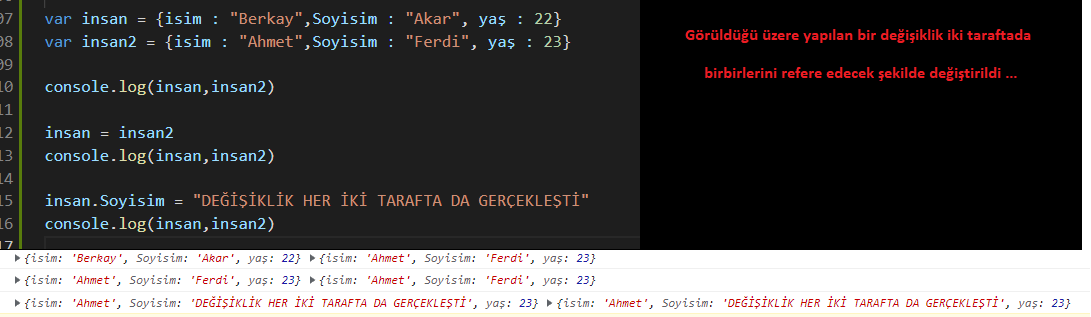
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Nasıl kullanıldıkları, tanımlandıkları gibi çeşitli bilgiler bir yukarıdaki menüde aktaılmış bulunmaktadır. Primitive ve referans (ilkel ve referans):

Primitive veri tipi sadece bir verinin değeridir. Referans ise bellek üzerinde bir yer tutar. Ve o değer üzerinden iş yapar. Birbirlerine eşitlenme durumunda bellek üzerinde aynı yer refere edileceğinden ötürü yapılan bir değişiklik tüm değişkenleri etkilemektedir.



Referans Veri tipinde : 

Fakat önemli bir esas ise değişken oluştururlken bu tip kelimeleri yerine 3 seçeneğimiz vardır.

**Var :** C# taki ile aynı olup içerisinde her veri tipini tutabilen ve değişiklik yapmamıza izin veren bir değişken gibidir. C# takine nazaran ilk tanımlandığındaki tip e bağlı kalmayıp (dynamic) çalışmasıdır. Aynı zamanda tanımlandığı yer ve oradan kalıtım alan diğer yerlerde geçerliliğini sürdürü. Bir fonkisyon iççinde tanımlanması koşulunda sadece o fonksiyon ve alt fonksiyonlarında geçerlidir. Lokal değişkendir Local scope olarakta geçer.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Üstte de gözüktüğü gibi C# ta izin vermese de JS kısmında bu değişiklige (hem değer hem tip değişikliğine) izin vermektedir.

Let: Değişkeni global olarak kullanmamıza yarayan bir değişken tipidir. Bir fonksiyon içerisinde kullanılması veya normal bir sayfada kulanılması ile birlikte onu (tanımlandığı yer farketmeksizin) global yapar. Global Scope olarak ta geçer

Const: C# taki const yapısı ile aynı özelliklier göstermektedir. Tanımlandıktan sonra değer değişikliğine izin vermeyen ve her yerde aynı değere sahip olan değişken tipidir.

# Veri Tiplerinin Birbirlerine Dönüşümleri

## String e Dönüşüm:

Let a ;

a = **String(576);**

a = **(123).ToString();**

Methodları ile dönüşümler gerçekleşmektedir.

let value;

value = 123;

// veri tipi bize number olarak dönmektedir.

value = String(value);

// bu şekilde veri tipi string e çevrildi.

value = String(function berkay() { console.log("ahmet")} )

// Bu dönüşümde bizlere hata vermeyecek ve stringe çevirecektir.

value = 187;

value = value.toString();

/\* Anlaşılacağı üzere ya

String() yada

(değişken\_ismi).ToStiring()

Yöntemleri ile bir castring Mevcuttur.

\*/

## String bir ifadeden sayısal bi ifadeye Dönüşüm:

Let a;

a = **Number(“123”)**;

a = **parseInt(“123”)**;

a = **parseFloat(“123”)**;

Yukarıdaki fonksiyonların 3 ü de string bir veriden sayıasl bir veriye dönüşümü sağlamaktadır. Parse ile int veya floata parsın çeşidine göre dönüştürülmektedir.

value = Number("325")

value = Number(null)

çevirim işlemini 0 olarak yapacak

value = Number(undefined)

// çevrimi yapar içi not a number tipinden  (nan) olur

// bu yüzden hatalı olarak düşünülebilir

value = Number("AKAR")

// hatalı bir dönüşüm yapacaktır not a number (nan) kısacası dönüşüm gerçekleşmeyecektir

value = Number("3.14")

// number olarak çevirecektir.

value = parseInt("3.14")

// number olarak çevirecektir çıktısı direk 3 olacak çünkü dönüşümü tamsayıya göre yapmaktadır.

value = parseFloat("3.14")

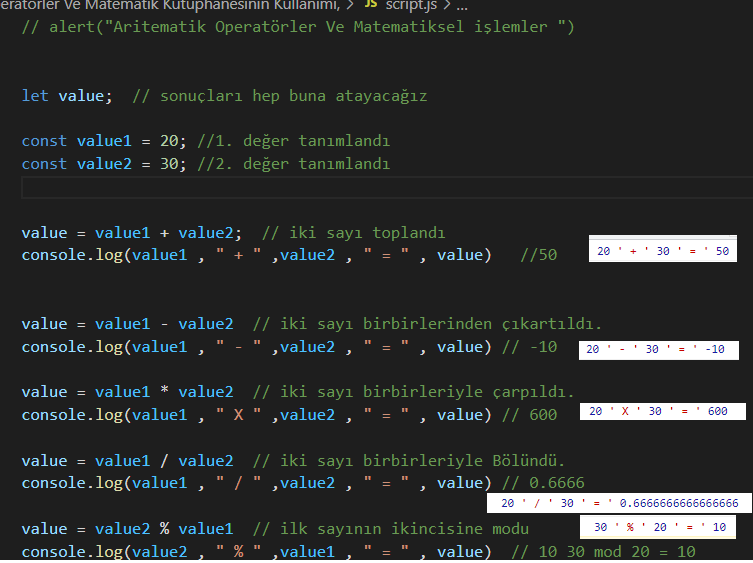
Anlaşılacağı üzere sadece null ve string bir ifadeden dönüşüm olmaktad diğer veri tiplerinin hepsinde undefied ve geri kalanlarında not a number not a number (nan) olarak vermektedir.

Let a ;

A = 5 + “Berkay”

Dönüşüm direk olarak stringe çevirilecektir. Bu işlemi bu şekilde gerçekleştirmektedir.

# Operatörler Ve Matematik İşlemleri



Görüldüğü üzere aritmatik işlemler bu şekilde sıralanmıştır. Esasında diğer programlama dillerindeki işlemler ile birlikte benzerlik göstermektedir.

                    /\*   MEATEMATİK KÜTÜPHANESİ    \*/

console.log("Pi Sayısı : " ,Math.PI) // value değerine pi sayısını verir. 3.141592653589793

console.log("E (Doğal Logaritma Katsayısı) Sayısı : " ,Math.E) // value değerine pi sayısını verir. 2.718281828459045

console.log("Yuvarlama işlemleri : ",Math.round(5.22))  //  5 . dan sonra sı 5 ten büyükse direk 6 yapar.

console.log("Yuvarlama işlemleri (Ceill): ",Math.ceil(5.01))  //  yukarı yuvarlama işlemidir. 5 ten sonra herhangi bir sayı varsa 0 dan büyük direk 6 yapar

console.log("Yuvarlama işlemleri (Ceill): ",Math.floor(5.99))  //  aşağı yuvarlama işlemidir. 5 ten sonra herhangi bir sayı varsa 0 dan büyük direk 5 yapar

console.log("Karekök alma işlemi : ", Math.sqrt(4))  // 2 yapar

console.log("üs alma işlemi : ", Math.pow(4,2))  // 16 yapar (eğer içeri tek parametre yollarsan not a number (NaN) cevabını döndürür.)

console.log("Mutlak değer işlemi -25 için ", Math.abs(-25)) // mutlak değere koyar.

MAXİMUM VE MİNİMUM OPERATÖRLERİ :

var dizi = [1,5, 12 , 8 , 22, 6 , -3 , 2 , 0]

console.log( dizi ," dizisi içerisindeki maximum değer :" ,Math.max(...dizi) , " Minimum Değeri : ", Math.min(...dizi));

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Üstteki kullanım orjinal kullanımından farklılık içermektedir. **Math.Max()** veya **Math.Min()** Fonksiyonuna direkt olarak bir dizi yollanamaz, içerisine sayılar tek tek parameter verilir. Örnek bir kullanımı aşağıda belirtilmiştir.

Math.Max(1,2,3,4,5,6)

Sonucunda sizlere en yüksek veya en düşük değeri döndürmektedir.

  console.log(Math.max(1,2,3,5,4) ," -----  " , Math.min(1,2,3,5,4))

# Random Sayı Üretme

Kütüphanesi **Math.Random()** olarak geçmektedir. 0 ile 1 arasında sürekli olarak sayılar üretmektedir. Her çalıştırılışta bu sayılar değişmektedir.

value = Math.random();

console.log(value);



Belirtilen aralıklarda sayı üretlmek istiyor isek :

// sıfır ile 20 arasında değerler üretmek istiyor isek

value = Math.random()\*20 ;

console.log(value);

// 1 ile 21 arasında değerler üretmek istiyor isek

value = Math.random()\*20+1 ;

console.log(value);

// 1 ile 20 arasında değerler üretmek istiyor isek

value = Math.floor( Math.random() \* 20 + 1 ) ;

console.log(value);

Matematiksel işlemler ile istenilen iki aralık arasında değerler üretilebilmektedir.

# String Fonksiyonları

let value;  // konsole ekranına basması için tanımlanmış bir değişken

var firstname = "Berkay";

var LastName = "Akar"

const Langs = "Java,Python,C++"

//                          String işlemleri :

value = firstname + LastName

console.log("İki Stringin Toplanmış Hali : ",value);

// += ile üstüne ekleme yapma :  (value = value + degisken == value += degisken)

value += " 2022 "

console.log("İki Stringin Toplanmış Hali + += operatörü ile  : ",value);



// bir stringin uzunlugunu bulmak lenght komutu ile gerçekleştir.

console.log("String uzunluğu : ",value.length);

String uzunlugu : 16 olarak çıktısını verecektir.

// += yada + ile iki stringi toplayabildiğimiz gibi diğer programlama dillerinden de hakim olduğumuz concat eklenebilir.

console.log(firstname.concat(LastName).concat(" 2022").concat(firstname.length));

// aynı zamanda istediğin kadar da concat içerisinde parametre verebilirisin

console.log(firstname.concat(LastName,value,LastName.length,"2022 ","Concat parametreli"))

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

// lower ve upper ile büyük küçük dönüşümü

console.log(firstname.toUpperCase() , " " , firstname.toLowerCase())

// BERKAY berkay

//string ifadeler aynı diğer dillerde  de oldugu gibi özünde bir dizi görevi göremektedir.

value= "Berkay"

for (var index = 0; index < value.length; index++)

{

    console.log(index , " . Harf : ",value[index])

}

// indexteki herhangi bir elemanın yerine yeni bir karakter ile değişltirme mevcut değildir.

value[2] = "A"

value[3] = "A"

console.log("Değiştirilmiş hali : " ,value)

// bu yüzden bu yöntemi kullanmamız gerekemetkedir.

String.prototype.replaceAt = function(index, replacement) {

    return this.substr(0, index) + replacement + this.substr(index + replacement.length);

}

value = value.replaceAt(2,"Qq");

console.log("Değiştirilmiş hali (ReplaceAtlıHali): " ,value)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu şekilde bir değiştirme işlemi yapmak mümkündür.

// bir string içerisinde bir karakter veya metin aramak istiyorsak index of methodlarını kullanabiliriz.

value = "Berkay akar ";

console.log(value.indexOf("a"));

// eğer içerisinde A geçen (Pascall Case dir büyük küçüğü farklı algılar.) bir değer var ise soldan başlayarak bulunduğu değeri bizlere iletir.  Bulamaz ise -1 sonucuu döndürmektedir.

console.log(value.lastIndexOf("a"));  // 9

// buda sondan bulduğu değeri getirir.

// ikiside birden fazla olup olmadığına bakmaz. birisi ilk buldugunu getirir soldan başlar diğeri soldan başlar ve ilk buldugunu getirir. İKiside sola göre konumlarını getirir.

/\*   CHAR - AT   \*/

value = "Berkay akar";

console.log(value.charAt(0) , " ---- " , value[0])   // B ---- B Döndü sonuç olarak

// hangi indexte hangi karakter oldugunu gösteren bir fonksiyondur.

// value[0] ile aynıdır.

// Split (bir string içerisinde belirli bir karaktere göre bölünmüş olan karakteri alabilmektedir.) Bölünen kısımların her birini bir array elemanı olarak yerleştirmetkedir.

var diller  = "Java,Python,C++"

console.log(diller.split(","))



Görüldüğü gibi bizlere 3 adet elemanlı ve içerisi string ifadeler ile dolu olan bir dizi getirdi.

// Split (bir string içerisinde belirli bir karaktere göre bölünmüş olan karakteri alabilmektedir.) Bölünen kısımların her birini bir array elemanı olarak yerleştirmetkedir.

var diller  = "Java,Python,C++"

console.log(diller.split(","))

var diller  = "Java,Python,C++"

dizi = diller.split(",");

for (let index = 0; index < dizi.length; index++) {

    console.log(index, " . dizininin elemanı " , dizi[index]);

}

Çıktısı :

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

// Replace fonksiyonu :

value = "Berkay akar "

value = value.replace("a","T")

console.log(value)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Gördüğüü ilk değerde değişiklik yaptıktan sonra devam etmemektedir.

**İNCLUDES**

// includes

value = "Berkay akar "

var sonuc = value.includes("akar")

console.log(sonuc)

gerek dizilerde , gerekse string içerisinde bir değerin olup olmadıgını aramatkadrır. Eğer bulur ise true , bulamaz ise bizlere false değer döndürmetkedir.



Genel C# tan hatırlanacağı üzere stringler formatlanabiliyordu. + ile toplama ${} ile içerisine değişkeni koyma, console.writeliene (“ {0} {1} {2}“ , 1 ,2 ,3) gibi çeşitli formatlanma yöntemleri mevcuttur. Beraberinde Ecma Script 6 ile birlikte gelen Literaller e bir bakış atalım .

// EcmaScript 6 ile birlikte gelen Literallere bakacağız.

const name = "Muhammed Berkay Akar"

const department = "Software Enginer"

const salary = 4000;

const a = "İsim: " + name + "\n" + "department : " + department + "\n" + "Maaş : " + salary;

console.log(a)

Eski yöntemi bu şekilde iken yeni yöntem :

// üstteki yazımın daha kullanışlı yöntemi ES6 ile birlikte gelmiş bulunmaktadır.

const b = ` İsim : ${name} \n Departman : ${department}  \n Maaş Bilgisi : ${salary}`

console.log(b)

// işaretin oluşturulması için Alt gr + noktalı virgül (noktalı virgüle 2 defa basılması)

const html = "<ul>" +

"<li> " +   name + "</li>" +

"<li> " +  department + "</li>" +

"<li> " +   salary + "</li>" + "</ul>";

document.body.innerHTML = html + "<br> <br>";

const html2 = `<ul>

                    <li>  İSMİM  (ES6): ${"Berkay AKAR"}  </li>

                    <li>  Departman Bilgisi :   (ES6): ${"Çaycı"}  </li>

                    <li>  Maaş  (ES6): ${"4250"}  </li>

                    `

document.body.innerHTML += html2;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

# DİZİLER

C# taki dizilerden farklılıklarına değinecek olursak :

* Başlangıçta boyut sayısı belirtmek zorunda değiliz.
* Içerisinde birden çok tip veri alınabilir.
* Const olarak tanımlanmış olsa dahi içerisindeki elemanlarda değişiklik yapılabilir, Eleman ekleme çıkratma yapılabilir.
* Atılan her elemanın tipi farklı olabilir ve bu tipleri kendi içeriğinde saklayabilmektedir.

/\*              Diziler                 \*/

let value;

value=[1,2,3,4,5,6]      // let value2= new Array(1,2,3,4,5,6)  // iki yöntem de kullanılıyor

const lang = ["Python","Java","C++","JavaScript"]

const karisik = ["Berkay","Akar",1,2,3,4,5,6,null,undefined,3.14,Date.now()]

console.log(value)

console.log(lang)

console.log(karisik)

// bir dizideki eleman uzunluguna bakma

console.log(value.length)

 // bir dizideki indexe göre eleman çağırma

console.log("Value dizisindeki 2. Eleman :",value[2]);  // string içerisinde char at çalışıyor fakat dizilerde CharAt Bulunmamamktadır.

// Herhangi Bir İndexteki Elemanı Değiştirme

console.log(value) // 1,2,3,4,5,6

value[3] = 99;

console.log(value) // 1,2,3,99,5,6

// index of (eleman varsa değerini döndürecek, yoksa -1 çevirecek)

console.log(value.indexOf(99)) // 3.bilgisini döndürdü. Çünkü Kendisi 3. İndex te

// dizilere eleman ekleme ve silme  (Pop - push - unshift - shift  )

value.push("Akar");

console.log(value)  // ekleme en sonuncu index e yapıldı.

// push en sona ekleme yaparken aradaEn başa   ekleme yapılacaksa = unshift methodu kullanılmalıdır

value.unshift(2,"Balıkesie")  // 2 tane eleman yolladım sırası ile ilk iki eleman bu şekilde

console.log(value)

value.unshift(5,"Gönen",187)  // 3 eleman yollandı.

console.log(value)

value.pop()

console.log(value) // en sonraki değeri silmiş oldu

value.shift();

console.log(value)  // en baştaki 5 çıkartılmış oldu

// belirlenen indexler arası değer atmak için

//  ['Gönen', 187, 2, 'Balıkesie', 1, 2, 3, 99, 5, 6]  son durumdaki dizi hali

value.splice(0,3)  // son elemanı almaz yani 0 1 2. elemanları atar

console.log(value)  // başlangıç ve bitiş indexleri

// dizilerdeki elemanları tam tersine çevirme

value.reverse()

console.log(value)  // tüm elemanları ters yaptı.

// dizilerde sıralama

value.sort()

console.log(value)

value.sort(function(x,y){

    // küçükten büyüğe

    return x-y

})

value.sort(function(x,y){

    // Büyükten Küçüğe s

    return y-x

})

console.log(value)

# DİZİ TANIMLAMA

const programmer = {

    firstname : "Berkay",

    Lastname : "Akar",

    Age : 22 ,

    Work\_on : ["Web" , "Desktop" , "Mobil"],

    Language : ["C#" , "Java" , ".NET" , "Core"],

    // iç içe objeler kullanılmaktadır.

    Adress : {

        City :"İstanbul",

        State : "Bayrampaşa",

        Full\_Adress : "Kartaltepe MAhallesi 22 / 2  "

    },

    selamlama\_fonkiyonu  : function()

    {

        console.log("Selamın Aleyküm");

    }

 }

 // üstte gözüktüğü gibi bazı alanlar tek değer , bazıları fonksiyon , bazıları iç içe objeler , bazıları ise diziler şeklinmde tanımlanmıştır.

 console.log(programmer.firstname , " -- ",  programmer.Lastname);

 console.log();

 console.log(programmer.Age , " Yaşında  ");

 console.log();

 console.log("Çalıştığı teknolojler : ");

 for (let index = 0 ; index < programmer.Work\_on.length ; index++ ){

     console.log(index + "  -  " +programmer.Work\_on[index])

 }

 console.log("  ");

 console.log("Diller : ");

 for (let index = 0 ; index < programmer.Language.length ; index++ ){

     console.log(index + "  -  " +programmer.Language[index])

 }

 console.log();

 console.log(" Adress : "+programmer.Adress.City + " " + programmer.Adress.State  + "  " + programmer.Adress.Full\_Adress)

console.log()

programmer.selamlama\_fonkiyonu();

// bu şekilde bir erişim olabileceği gibi bir başka erişim şeklide bir aşağıda belirtilmiştir.

console.log(programmer["Adress"].City);

// firs json app

const programmers = [

    {name :"Berkay" , age : 22  },

    {name :"Mahmut" , age : 22  }

]

// bu dizi objelerden oluşan bir arraydir.

for (let index = 0; index < programmers.length; index++) {

    console.log (programmers[index].name , "     " , programmers[index].age);

}