berkay\_akar1@saintjim.me

JavaScript

İçindekiler

[Javascript Kodlarını Projeye Dahil Etmek 2](#_Toc98534367)

[Temel JS Komutları 3](#_Toc98534368)

[Veri Tipleri 4](#_Toc98534369)

[Veri Tiplerinin Birbirlerine Dönüşümleri 8](#_Toc98534370)

[String e Dönüşüm: 8](#_Toc98534371)

[String bir ifadeden sayısal bi ifadeye Dönüşüm: 8](#_Toc98534372)

* 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

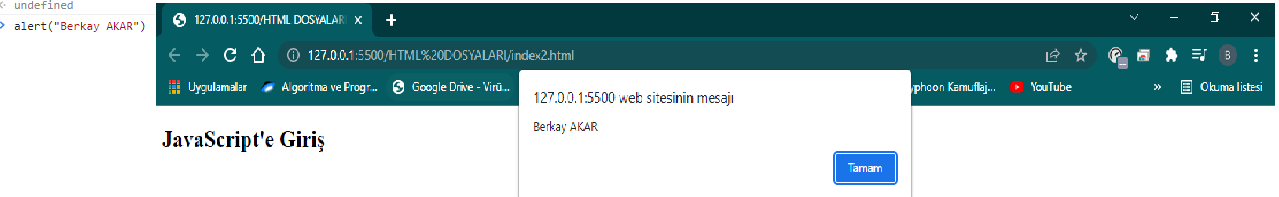
Tavsiye edilen tanımlama tipi body tagının bitim yerinin bir üstüdür. (bu tanımlama sadece uzak sunucuda iken parallel indirmede başlatılması işin kullanılır).

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.

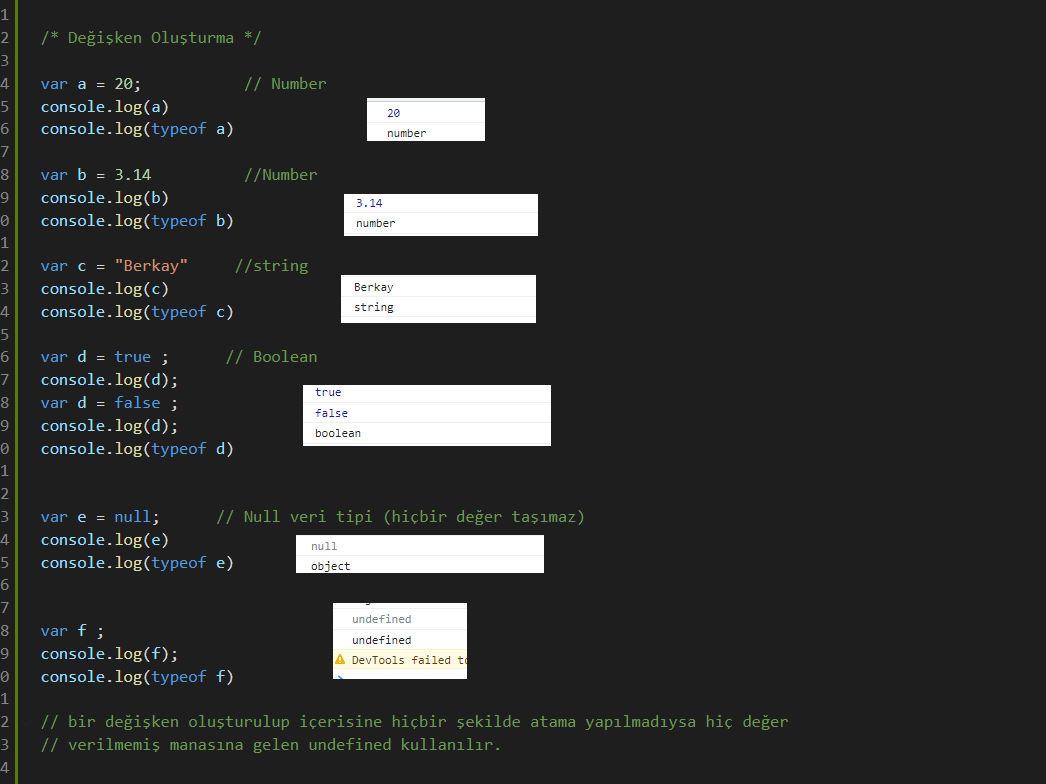
**metin, ekran, ekran görüntüsü içeren bir resim

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

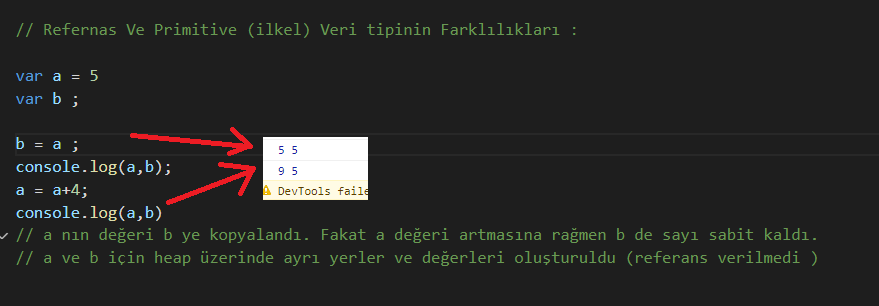
Açıklama otomatik olarak oluşturuldu

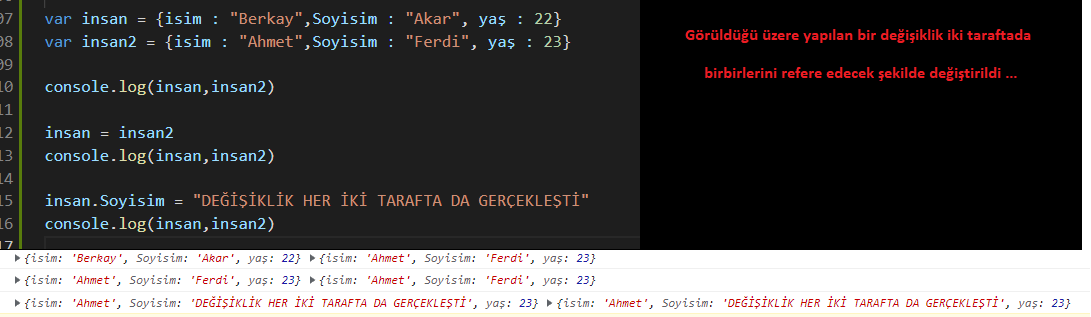
metin içeren bir resim

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.