**JS DOM**

DOM, HTML ve XML yapıdaki belgelerin hiyerarşik yapısını gösteren bir kılavuzdur. “**D**ocument **O**bject **M**odel” kelimelerinin baş harfleriyle oluşan DOM, Türkçe olarak “Belge Nesne Modeli” olarak ifade edilir. Bu model, yazılım dillerinin belgelere erişmesini ve değişiklik yapmasını sağlayan bir arabirimdir.

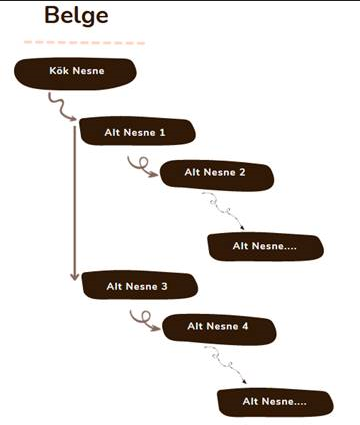
## DOM Nedir?

**DOM, belgenin içeriğini belirli bir düzen içerisinde nesne olarak temsil edilmesini sağlayan bir modeldir.** Bu modelde belge yapısında bulunan her bir nesne için bir düğüm bulunur. Alt düğümler kök bir düğüme bağlanır. Belge ile iletişim kuran servis düğümü çözerek işlemlerini gerçekleştirir. Örneğin, bir WEB [API](https://www.hosting.com.tr/bilgi-bankasi/api/) servisi, XML dosyadan veri çekerken bu modele göre belgeye erişir.

İnternet tarayıcısı bir web sayfasını açarken sunucuya ilk isteği gönderir ve sayfaya ait belge yükler. Belgedeki nesneleri düğümlerden çözerek hiyerarşiye göre sayfa içerisine yerleştirir ve site açılmış olur. Web sayfasına dışardan erişim sağlayan tüm servisler bu şekilde kolaylıkla belgenin mantıksal yapısına ulaşır ve sayfada değişiklik yapabilir.

## DOM Yapısı Nedir?

DOM yapısı nesnelerden oluşan bir yapıdır. Bir çeşit alt-üst ilişkisi bulunan soyağacı gibidir. Ana nesne altında birbirinden farklı alt nesneler, alt nesnelerin altında da hiyerarşik yapıda nesnelerden oluşur.

[](https://cdn.hosting.com.tr/bilgi-bankasi/wp-content/uploads/2022/07/resim-1.png)

DOM yapısını bir de HTML kodu üzerinden inceleyelim.

<!doctype html>

<html>

<head>

<title>Başlık</title>

<metacharset="karakter seti">

</head>

<body>

<h1>Alt başlık</h1>

<p>Paragraf</p>

</body>

</html>

HTML’de her bir açılan ve kapanan bölüm arasında kalan bölüm DOM yapısı içerisinde bir nesne oluşturur. Bu durumu görselleştirmek istersek aşağıdaki gibi bir yapı ortaya çıkar.

[d
C[O8Å
أ»
لملe9
IAIH
C[Y3H
إ).
١) ](https://cdn.hosting.com.tr/bilgi-bankasi/wp-content/uploads/2022/07/image.png)

DOM yapısı, HTML ve XML sayfaların nesne olarak modellenmesi diye ifade edilebilir. Javascript, [Python](https://www.hosting.com.tr/blog/python-ile-web-sitesi-nasil-yapilir/)gibi yazılım dillerinin bu model sayesinde HTML ve XML sayfalarına erişim sağlarlar ve gerektiğinde sayfalarda değişiklik yapabilirler.

## DOM ile Javascript İlişkisi

DOM Javascript dışında farklı programlama dilleri ile de uyumludur. Bununla birlikte dinamik HTML sayfa kullanımında Javascript yaygın olarak karşımıza çıkar. Özellikle internet tarayıcılarında dinamik HTML oluşturmak için tercih edilir. DOM ile Javascript ilişkisi sayesinde aşağıdaki işlemler daha kolay ve etkili bir şekilde yapılabilmektedir:

* HTML ögelerini ve değerlerini değiştirmek
* CSS stillerini değiştirmek
* HTML ögelerini ve değerlerini silmek
* Olmayan HTML ögeler ve değerler eklemek
* Sayfada HTML olarak yanıt vermek
* Yeni HTML bölümü eklemek

## DOM Kaynak Kod mu?

DOM yapısı incelendiğinde sanki websitesine ait kodları içeriyor gibi algılanabilir. Öncelikle DOM kaynak kod değildir. **DOM nesne modelleme ile kodların daha anlaşılır hale gelmesini sağlayan bir arabirim standardıdır.** Bu standart, W3C (**W**orld **W**ide **W**eb **C**onsoritum) ve WHATWG (The **W**eb **H**ypertext **A**pplication **T**echnology **W**orking **G**roup) tarafından belirlenmektedir. İnternet tarayıcıları bu standartları kullanır.

## DOM Bölümleri Nelerdir?

DOM yapısını belirleyen W3C (**W**orld **W**ide **W**eb **C**onsortium) otoritesi belirlediği standarda göre DOM üç bölümden oluşmaktadır.

* **DOM Core:** DOM yapısının çekirdeğini oluşturur ve standart kullanımda olan bir modeldir.
* **DOM XML:** Sadece XML belgelerinde kullanılan bir modeldir.
* **DOM HMTL:** Sadece HTML belgelerinde kullanılan bir modeldir.

DOM standardı belirli seviyelerden geçerek günümüze kadar gelişim göstermiştir.

### DOM Level 1

W3C otoritesi DOM standardını ilk defa 1998 yılında belirlemiştir. Bu kapsamda DOM Core ve DOM HTML modellerini ortaya koymuştur. Ek olarak bazı sınırlamalar getirmiştir:

* İç alt küme ve dış alt küme için bir nesne modeli
* Doğrulama
* Stil dosyaları ile belge oluşturma kontrolü
* Giriş kontrolü
* Akış güvenliği

### DOM Level 2

2000 yılında standartlaşan bu kısımda arabirim yönetimi üzerinden nesne modellemesi sağlandı. getElementById() gibi metodlar eklendi. Bu sayede CSS gibi stil yapılarının öncesini ve sonrasını görme imkanı sunuldu. Ayrıca olay yönetimi, belge ağacında gezinme gibi modüller eklendi.

### DOM Level 3

Bu seviyede belgeleri tek türde yükleme standardı getirilirken doğrulama yöntemleri modernize edildi. Xpath ve klavye olay işleme desteği eklendi.

## DOM ile İlgili Temel Kavramlar Nelerdir?

DOM ile ilgili araştırma yapanların sıkça karşısına çıkan temel kavramlar şu şekildedir:

* **Belge:** HTML veya [XML](https://www.hosting.com.tr/bilgi-bankasi/xml-nedir/) ile hazırlanan sayfanın tamamı belge adıyla tanımlanır.
* **Nesne:** HTML veya XML ile hazırlanan sayfanın içeriğini oluşturan ögeler nesne adıyla tanımlanır.
* **Çağrılmak:** Bir belge içerisindeki kod dizisinin DOM yapısına göre hiyerarşik bir biçimde istenmesi olayına verilen bir tanımdır.
* **Düğüm Ağacı (Node Tree):** Belge içerisindeki nesnelerin hiyerarşik yapı içerisinde birbirlerine bağlandığı yapıdır.
* **Düğüm (Node):** Her bir nesnenin düğüm ağacına bağlanmak için oluşturduğu noktadır.
* **Element:** Düğüm içerisinde yer alana kod dizeleridir.
* **Öznitelik (Attribute):** DOM yapısında erişimi sağlayan nesnelerin özniteliklerini tanımlayan kısımlardır.
* **Öge Dizin (Named Node Map):** Nesnelerin özniteliklerine göre belirli bir  hiyerarşik sıralaması olmadan listelendiği dizine verilen addır. Bu dizinde yapılan değişiklik sayfada otomatik olarak yansır.

Jquery Dom

jQuery kütüphanesi yardımı ile istediğimiz bir elementin child’larına erişebiliriz. Aşağıdaki kod ile parent id’li elementin bütün childrenları &kids isimli değişkene atanmıştır.

const $kids = $('#parent').children();

Şimdi bu kids isimli değişkene ait olan childrenlar için click event oluşturalım ve eğer bu elementlere tıklanırsa CSS özelliklerini aşağıdaki gibi değiştirebiliriz.

$kids.on('click', event => {   
$(event.currentTarget).css('border', '1px solid black');   
});

Örneğin aşağıdaki kodta parent isimli bir class’ın atandığı div elementi var. Bu elementin altında ise liste tanımlanmış(1,2,3,4,5). parent li elementi yani listedeki elementlere tıklandığında aşağıdaki ifadeler teker teker gerçekleşecek.

1- İlk önce tıklanan liste elementine (örneğin 1) color-1 isimli sınıf eklenecek. Yani rengi pembeye dönecek.

2- Ardından $(event.currentTarget).**siblings**().removeClass(‘color-1’) kodu ile tıklanan liste elementinin kardeşlerinden yani tıklanmayan liste elementlerinden eğer color-1 sınıfı atandıysa silinecek.

3- Son adım olarak parent isimli sınıfın atandığı div elementinin çocuklarından herhangi birinde color-2 sınıfı atandıysa; bu sınıf o element/elementlerden silinecek.

$('.parent li').on('click', event => {  
 $(event.currentTarget).addClass('color-1');  
   
 $(event.currentTarget).siblings().removeClass('color-1')  
   
 $('.details').children().removeClass('color-2')   
   
 });----HTML----<div class='parent'>  
 <ul class='number'>  
 <li>1</li>  
 <li class='color-1'>2</li>  
 <li>3</li>  
 <li>4</li>  
 <li>5</li>  
 </ul>  
   
 <div class='button color-2'>  
 <h3>EKLE/h3>  
 </div>  
   
 </div>  
----CSS---.color-1: {  
pink }.color-2: {  
green}

Şimdi de **closest()** methodundan bahsedelim.

Yukarıdanki kodta ilk iki adım aynı. Son adımı ise farklı bir kod yapısı ile gerçekleştirebiliriz. Şuanki hedefte yani div blokunun listesinde, parent sınıfına ait div blokuna en yakın olan childrenlardan color-2 sınıfına ait olan elementten bu sınıfı sil şeklinde de tanımlayabilirdik.

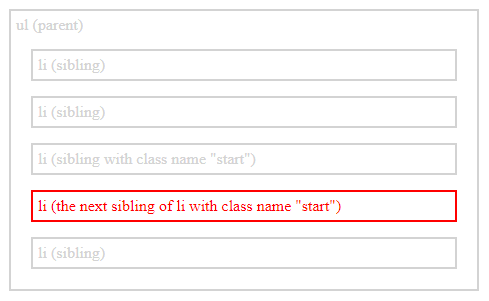
Bu metotu daha iyi anlamak için [w3schools](https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_traversing_closest)’ta güzel bir örnek var, inceleyebilirsiniz.

$('.parent li').on('click', event => {  
 $(event.currentTarget).addClass('color-1');  
   
 $(event.currentTarget).siblings().removeClass('color-1')  
   
 **$(event.currentTarget).closest('.parent').children().removeClass('color-2')**    
   
 });----HTML---<div class="parent"><ul class='number'>  
 <li>1</li>  
 <li class='color-1'>2</li>  
 <li>3</li>  
 <li>4</li>  
 <li>5</li>  
 </ul>  
   
 <div class='button **color-2**'>  
 <h3>EKLE/h3>  
 </div></div>

Sırada **next()** metodu var. Bu metot ise seçilen elementin en yakın kardeşine (sibling) işaret eder. Kardeş elementler (siblings) aynı ebeveyne sahip (parent) elementlerdir.

Aşağıdaki kodta start sınıfına sahip liste elementine en yakın olan kardeş elementin CSS bilgileri değiştirilsin istenmiş. Örnek [w3school](https://www.w3schools.com/jquery/traversing_next.asp)’dan alınmıştır.

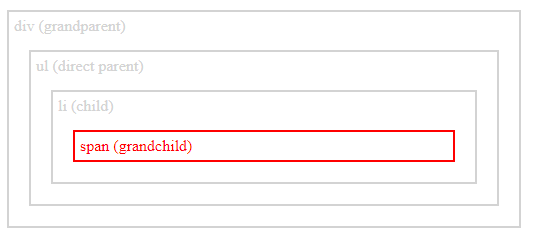
$(document).ready(function(){ $("li.start").next().css({"color": "red", "border": "2px solid red"});  
   
 });----HTML---<div style="width:500px;" class="siblings">  
 <ul>ul (parent)   
 <li>li (sibling)</li>  
 <li>li (sibling)</li>  
 <li class="start">li (sibling with class name "start")</li>  
 <li>li (the next sibling of li with class name "start")</li>  
 <li>li (sibling)</li>  
 </ul>   
</div>



**find()** metoduna gelelim. Bu metot ile belirtilen elementin, çocuğu (child), torunu (grand child) hatta torununun torununa kadar bulunabilir.

Yine [w3school](https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_traversing_find)’dan aldığımız bir örnekle açıklayalım. Aşağıda ul elementinin çocukları ve torunları mevcuttur. Çocuğu li yani liste elementi iken, torunu span elementidir. find() metodu ile torunu olan span elementlerinin CSS özellikleri değiştirilmiştir.

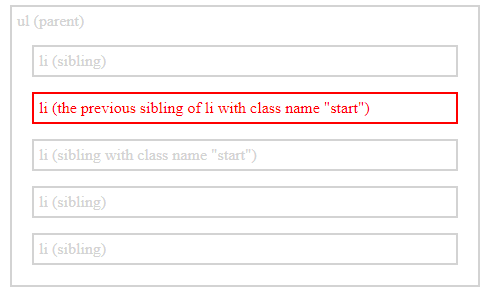
$(document).ready(function(){$("ul").find("span").css({"color": "red", "border": "2px solid red"});  
   
 });----HTML---<div style="width:500px;">div (grandparent)  
 <ul>ul (direct parent)   
 <li>li (child)  
 <span>span (grandchild)</span>  
 </li>  
 </ul>   
 </div>



**prev()** metodu ise bulunan elementin bir önceki kardeş(sibling) elementine işaret eder. Detaylı açıklamalara [w3school](https://www.w3schools.com/jquery/traversing_prev.asp)üzerinden ulaşabilirsiniz.

Aşağıda start sınıfının tanımlandığı liste elementinden bir önce kardeş(sibling) liste elementinin css kodları değiştirilmiştir.

$(document).ready(function(){  
 $("li.start").prev().css({"color": "red", "border": "2px solid red"});  
   
 });----HTML---<div style="width:500px;" class="siblings">  
 <ul>ul (parent)   
 <li>li (sibling)</li>  
 <li>li (the previous sibling of li with class name "start")</li>  
 <li class="start">li (sibling with class name "start")</li>  
 <li>li (sibling)</li>  
 <li>li (sibling)</li>  
 </ul>   
</div>



Yukarıda bahsettiğimiz DOM yapısı iyi algılanırsa, yapılmak istenen basit işlemlerde hızlıca gerçekleştirilebilir ve daha sistemli kod mimarileri yazılabilir.

