jQuery

Emre Can ÖZTAŞ

www.emrecanoztas.com

Kapak Tasarımı: Emre Can ÖZTAŞ

jQuery

Book Version: 1.0
jQuery Version: 3.1.1

Yazan: Emre Can ÖZTAŞ

Kitap Hakkında

Bu kitap; temel ve ileri düzey jQuery anlatmak için yazılmıştır. Kitabı takip edebilmeniz için: HTML + CSS ve az da olsa JavaScript bilginizin olması beklenmektedir. Konuların anlatımı basit düzeyde tutulmuştur. Yeni başlayanlar ve ileri düzey geliştiriciler kitabı rahatlıkla takip edebilir.

Memnun kalmanız dileğiyle.

Yazar Hakkında

Emre Can ÖZTAŞ, yazılım geliştirici, web girişimcisi, eğitmen ve yazardır. Çeşitli dillerde yazılım geliştirmektedir. Kendisine ait olan web projeleri vardır ve web girişimcilik alanında çalışmaları sürmektedir. Çeşitli alanlarda birebir veya grup olarak özel eğitimler vermektedir. Genellikle öğrencileri ondan memnundur. Bunun yarısıra; bilgisi ve tecrübesi dahilinde Türkçe kaynağa destek ve öğrenmeye istekli olan herkesin yararlanabilmesi için e-kitap'lar yazmaktadır. Yazdıklarından herhangi bir ücret talep etmemiştir ve kazanmamıştır.

Yazar hakkında daha fazlasını merak ederseniz; kendisi www.emrecanoztas.com adresinde çeşitli konularda yazılar yazmakta ve paylaşımlarda bulunmaktadır. Aynı zamanda kitap hakkındaki; görüş, öneri ve yanlışları da bu adrese iletebilirsiniz.

İçindekiler

BÖLÜM 1: jQuery Temelleri

Bu bölümde jQuery nedir, neler yapılabilir ve nasıl kullanılır gibi çeşitli soruların üzerinde durmaya çalışacağız. jQuery oldukça kullanışlı bir yapıya sahip. Bunu sizde zamanla göreceksiniz. jQuery'i anlatırken oldukça basit bir dil kullanmaya dikkat edeceğim lakin herşeyi de temelden anlatmayacağım. En azından temel JavaScript bilginizin olduğunu varsayarak konularımızı ele alacağız. Eğer JavaScript hakkında en ufak bir bilginiz yok ise daha önce yazmış olduğum `Simple JavaScript` kitabına bir göz atabilirsiniz.

Hazırsanız, o halde "Bismillah" diyerek jQuery öğrenmeye başlayabiliriz.

1.1 jQuery Nedir?

jQuery bir JavaScript library (kütüphane)'sidir diyebiliriz. Kullanımı ve öğrenimi basit olduğu kadar oldukça etkili bir kütüphane. Bildiğiniz gibi JavaScript yazarken deyim yerindeyse kılı kırk yararak ya da bin dereden su getirerek bir şeyler yapmaya çalışırız. Tabi bu da ekstra emek ve efor sarfetmek demektir. Ayrıca kod karmaşıklığı da bir hayli artmaktadır. Lakin jQuery ile belli başlı işlemler daha kolay ve efektif bir yolla çözülebilinmektedir.

Günümüzde bir çok JavaScript framework'ü ve kütüphanesi bulunmaktadır. Bunlara örnek verecek olursak; AngularJS, BackboneJS, Dojo, NodeJS, React, Ember ve daha fazlası. Tabiki bunlara jQuery'i de dahildir. JavaScript kütüphaneleri hakkında detaylı bilgi almak isterseniz aşağıdaki adrese bakmanızı tavsiye ederim.

https://en.wikipedia.org/wiki/List_of_JavaScript_libraries

jQuery, 2006 yılında John Resig tarafından geliştirilmeye başlanmıştır. Günümüzde de gelişimini sürdürmektedir. Open – Source (Açık Kaynak) olduğu için kullanımından kaynaklı herhangi bir ücret ödenmesi söz konusu değildir. Zira, MIT lisansına sahiptir. Yani bu da demektir ki; bu ürünü kullanabilirsiniz, dağıtabilirsiniz, yazılım geliştirebilirsiniz ve bu yazılımdan para kazanabilirsiniz. Çok güzel değil mi?

jQuery, web yazılımcıları açısından en çok tercih edilen kütüphanelerden birisidir. Bunun temel nedeni; basitlik ve çok yönlülük olarak nitelendirebiliriz. Bu cümlemizi biraz daha açmak istersek: jQuery, CSS ve temel JavaScript üzerine kurulmuştur. Yani biraz CSS ve JavaScript bilginiz varsa jQuery'i kısa bir sürede öğrenebilirsiniz. Temel JavaScript bilgisi, jQuery için şart değildir. İlk olarak jQuery'i de öğrenmeye başlayabilirsiniz lakin benim size tavsiyem ilk olarak temel JavaScript bilginizin olması gerekir. Çünkü daha kolay anladığınızı ve daha rahat hareket ettiğinizi anlayacaksınız. jQuery ile bir çok işlem (DOM, CSS, Ajax, Efekler, Animasyon v.s) daha kolay ve kısa bir şekilde gerçekleştirilebilinmektedir. Bunları da bir sonraki başlıkta inceleyelim.

1.2 Neden jQuery?

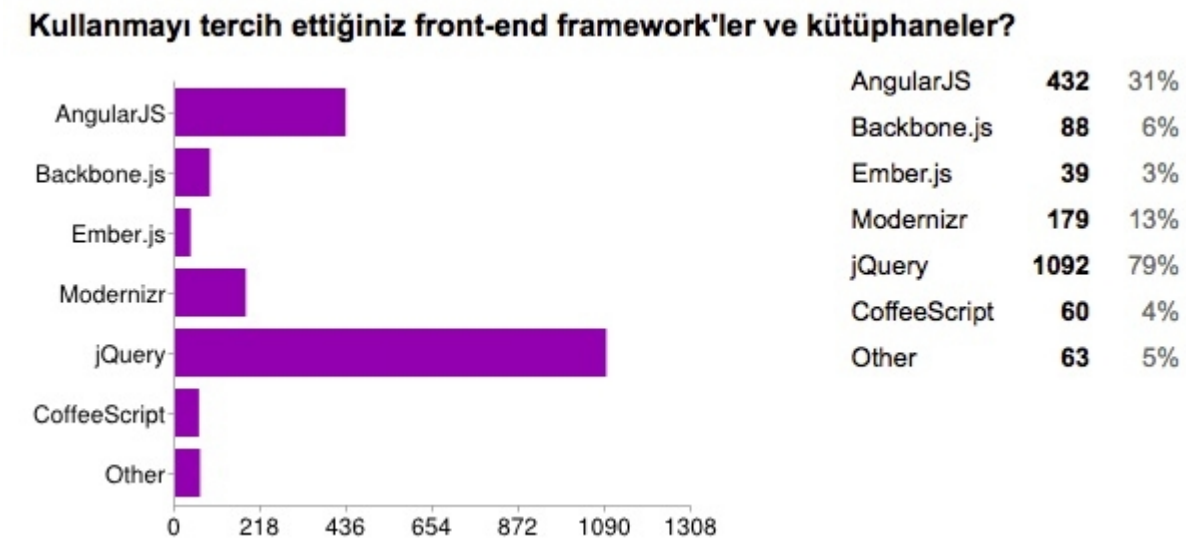
Neden jQuery? Aslında neden jQuery değil diye sormamız daha yerinde olacaktır, bana göre. Neden böyle dediğimi gelin birlikte inceleyelim.

Daha önce de belirtmiştik; bazı işlemler jQuery ile daha kolay ve daha hızlı bir şekilde halledilebiliyor. JavaScript'te olduğu gibi çok fazla satır kod yazmamıza da gerek kalmıyor. Zaten jQuery'nin sloganı "Write less, do more!" Türkçe'ye çevirmek istersek: "Az kod, çok iş!" gibi bir anlam çıkarabiliriz. Hakikaten bunun böyle olduğunu siz de ilerleyen bölümlerde göreceksiniz.

jQuery'nin en çok tercih edilen JavaScript kütüphanelerinden birisi olmasında elbette bir takım sebepler var. Bunları sırasıyla verecek olursak:

- Open Sources (Açık Kaynak)'tir.
- Dokümantasyonu çok iyidir ve yeni başlayanlar için çok sayıda materyale sahiptir.
- DOM (Document Object Model) işlemleri basittir.
- AJAX işlemleri basittir.
- CSS değişiklikleri ve işlemleri basittir.
- Güçlü bir Event (Olay) metotlarına sahiptir.
- Efekt ve anismasyon olayları oldukça güçlü, çeşitli ve basittir.
- Ve akla gelmeyen daha fazlası...

Türkiye'de yapılan bir anketi de sizinle paylaşmak isterim. Bu anket webrazzi tarafından 2014 yılında yapılmıştır. Ankete göre en çok kullanılan JavaScript kütüphanesi jQuery olmuştur.



Bu anket aslında Türkiye'de olduğu gibi dünyadaki kullanım için geçerlidir. Çünkü jQuery çok fazla tercih edilen bir kütüphanedir.

Yukarıdaki ankete (yapılan başka anketlerde var, ilginizi çekebilir) aşağıdaki adresten ulaşabilirsiniz.

http://webrazzi.com/2015/01/19/turkiye-yazilim-gelistiricileri-anketi-2014-sonuclari/

1.3 jQuery Kullanımı

jQuery kullanımı, daha doğrusu JavaScript kütüphanelerinin kullanımı oldukça kolaydır. Tek yapmanız gereken şey; sayfanızda kullanacağınız çatıyı belirtmektir. Bu belirtme işlemi de tabiki <script></script> etiketinde belirtilir.

Öncelikle jQuery 3.x sürümünü kullanacağımızı belirteyim. Zaten şuan; 1.x | 2.x | 3.x sürümleri mevcut. 2.x ve 3.x sürümlerinde; IE: 6 | 7 | 8 desteği bulunmamaktadır. Dolayısıyla bu tarayıcılardan herhangi birini kullanan (Evet, maalesef hala kullanan var) bir kullanıcıya destek veremeyeceğiz. Yani yazacağımız Script (Betik)'lerin büyük bir kısmı (veya tamamı) bu IE sürümlerinde çalışmayacaktır. Şayet bu tarayıcılara da destek vermek isterseniz; 1.x sürümünü kullanmanızı tavsiye ederim.

jQuery kütüphanesini, sayfanıza iki farklı şekilde ekleyebilirsiniz. Bunlardan birincisi: CDN (Content Delivery Network). Yani uzak bir sunucuda bulunan jQuery kütüphanesini doğrudan sayfanıza ekleyebilirsiniz. Burada dikkat etmeniz gereken şey ise; geliştirme yaparken internete bağlı olup olmadığınızdır. Eğer internete bağlı değilseniz, herhangi bir çalıştırma veya deneme şansınız yok.

jQuery-3.1.1 için Google ve Windows CDN adresleri:

Google: https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js
Microsoft: https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.1.1.min.js

Bir diğer yöntem ise: jQuery'i indirip, herhangi bir path (yol)'e yerleştirmeniz ve daha sonra kullanacağınız sayfanıza eklemenizdir. Benim genellikle tercih ettiğim yöntem de budur.

Öncelikle aşağıdaki adrese girip, jQuery'i indirmeliyiz.

http://jQuery.com/download/

Yukarıdaki adrese girdiğinizde 3.x sürümünü seçelim. Şuan güncel olarak (2017): jQuery 3.1.1 sürümü mevcuttur. Burada da karşınıza iki farklı tür çıkacaktır. Bunlar:

- `jQuery-3.x.x.js`
- `jQuery-3.x.x.min.js`

Bunlardan hangisini kullanmamız gerekir? Öncelikle .min'ler hakkında herhangi bir bilginiz yoksa kısaca bahsedelim. Bir web sayfasının çabuk yüklenen ve hızlı işlem yapanı makbuldür. Dolayısıyla sayfada kullanılacak kaynakların da yükünün olabildiğince az olması gerekir. Örneğin; resim dosyaları küçültülmelidir, CSS ve JavaScript dosyaları sıkıştırılmalıdır. Bu sıkıştırılma işlemi çeşitli araçlarla

gerçekleştirilebilir. İşte sıkıştırılan bu dosyaların isimlerinin sonlarına belli olması için .min konulur. Buradaki min (minimum)'i belirtmektedir. Yukarıda belirttiğimiz dosyalardan herhangi birini indirebilirsiniz. Lakin web sayfasınızda; .min mahlaslı dosyayı kullanmaya dikkat edin. Gelin isterseniz biz de öğrenimimiz boyunca normal olanı yani jQuery-3.x.x.js sürümünü kullanalım. Böylelikle karışıklığa engel olmuş oluruz hemde jQuery'nin kodları rahatça inceleyebiliriz.

Herneyse, jQuery-3.x.x.js dosyamızı indirdik. Şimdi bunu sayfamızın hemen yanına konumlandırdığımızı varsayarak sayfamıza ekleyelim. Bu işlemi yine; <script></script> etiketlerinde yapacağız. Aşağıda olduğu gibi.

```
<script src="jQuery-3.1.1.js"></script>
```

Bildiğiniz gibi JavaScript, <script></script> arasında yazılır. Eğer herhangi ekstra bir dosya veya çatı eklenecekse bu src özelliğinde belirtilir. Yani: `<script src="jsFile"></script>` HTML 5 ile çalışıyorsanız script etiketinde herhangi başka bir şey belirtmeye gerek yoktur lakin HTML 5 değilde örneğin; HTML 4 ile çalışıyorsanız, `type` özelliğini de eklemelisiniz. Yani:

```
<script type="text/javascript" src="jsFile"></script>
```

BÖLÜM 2: Syntax

Bu bölümde genel olarak jQuery'nin sahip olduğu Syntax (Söz Dizimi)'tan bahsedeceğiz. Aslında jQuery'nin söz dizimi size ilk başlarda biraz garip gelebilir o yüzden hemen endişelenip öğrenmeyi yarıda bırakmayın. Zamanla bu söz dizimine alıştığınızda ne kadar da güzel, kullanışlı ve mantıklı bir söz dizimine sahip olduğunu anlayacaksınız.

jQuery'nin söz dizimine bakmadan önce indirmiş olduğumuz jQuery dosyasını, yeni bir .html uzantılı sayfa oluşturup bu sayfamıza ekleyelim. Ben basit olması açısından bir dosya açtım ve içerisine index.html ve jQuery dosyasını yerleştirdim.

Yani:

```
jQueryExample/
|--- jQuery-3.0.0.js
|--- index.html
```

şeklinde bir yapı kurdum. Burada; jQueryExample benim çalışma dosyamın ismi yani workspace'm. Şimdi index.html isimli dosyamıza temel HTML yapımızı kuralım ve jQuery-3.1.0.js isimli dosyamızı ekleyelim. Aşağıda olduğu gibi.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>jQuery Example</title>
<script type="text/javascript src="jQuery-3.1.1.js"></script>
</head>
<body>

</body>
</html>
```

Dilerseniz jQuery dosyamızın adını basit olması için değiştirebilirsiniz. Yani aynı ismi kullanmak zorunda değilsiniz. Tabiki değiştirdiğiniz dosyanın adıyla sayfanıza eklemek koşuluyla!

Her şey tamam. Şimdi jQuery'nin söz dizimine bakabiliriz. Öncelikle bilmemiz gereken jQuery'i kullanırken aşağıdaki yapıyı ilk olarak mutlaka kurmalıyız.

```
$(document).ready(function($) {
     // yapilacaklar
});
```

Bu yapıyı mutlaka kurmalıyız dedim ama kurmak zorunda da değiliz. Sıradan JavaScript yazar gibi de yazabiliriz, burada herhangi bir sıkıntı yok. Lakin bu yapıyı kurmamızın amacı: jQuery'i kullandığımız sayfa, yüklendiği anda jQuery'inin

çalışmaya başlamasıdır. Yani bir nevi Scope (Etki) alanı oluşturuyoruz. Bu etki alanı jQuery'e ait. Sayfa tamamen yüklendiği zaman bu oluşturduğumuz alan çalışmaya başlayacaktır. Sayfa tam yüklenmeden bu alan çalışmaz. Bu yüzden güvenli bir yapı o yüzden böyle kullanmak zorundayız (güvenlik için).

Sayfamız tam anlamıyla yüklendiği anda bu etki alanını tanıyacak ve içerisinde ne işlem tanımlamış isek; browser (tarayıcı) bu işlemleri yerine getirmeye çalışacaktır. Bildiğiniz gibi web tarayıcıları üç şeyden anlarlar. Bunlar: HTML, CSS ve JavaScript. Bunlar dışında herhangi bir şeyden anlamazlar.

Küçük bir örnek yapalım. Mesela alert( ) ile bir mesaj verdirelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>jQuery Example</title>
<script type="text/javascript" src="jQuery-3.1.1.js"></script>
</head>
<body>

<script type="text/javascript">
    $(document).ready(function() {
        alert("Hello World!");
    });
</script>

</body>
</html>
```

Yukarıdaki örneğimizi incelediğimizde; indirmiş olduğumuz jQuery dosyamızı, sayfamıza ekledik. Daha sonra temel jQuery yapısını kurduk. Dikkat ederseniz bu yapıyı ayrı bir script etiketinde kurduk. jQuery dosyasını eklediğimiz script etiketinde kurmaya kalkarsak; yanlış olur ve yazdığımız betikler çalışmaz. Buna çok dikkat edelim.

Sayfamızı tarayıcımızda görüntülediğimiz anda: "Hello World" yazan bir mesaj gelecektir. Daha öncede dediğim gibi; bu etki alanında ne yazarsak o işlenecektir. Neden? Çünkü jQuery'i web sayfasıyla aynı anda çalışmaya başlayacaktır. Burada herhangi bir koşul belirtmediğimiz için direk işlemlere geçilecektir. Örneğin; document.write( ) kullandığımız zaman da yine aynı şekilde ekrana ne istemiş isek onu yazacaktır. Buna da dikkat edelim.

Yukarıdaki kullanım stilinin bir diğer versiyonu da aşağıdaki gibidir.

```
jQuery(document).ready(function() {
    // yapilacaklar
```

```
});
```

Aslında temelde başlığımız: jQuery idir. $ işareti burada bu kelime yerine kullanılan bir alias (takma ad – lakap. Biz mahlas diyelim)'dir. Yani kısaltılmış şeklidir. Hangisi size kolay geliyorsa o başlığı kullanabilirsiniz. Burada bu başlığın olması demek; "Hey burada jQuery betiği var, tanıdım!" gibi bir anlam taşımaktadır. İlerleyen bölümde hep bu başlık ile işlem yaptığımızı göreceksiniz. Daha da akılda kalıcı olması için; bu başlık jQuery'i başlığıdır. Yani eklediğimiz kütüphane bu şekilde çağırılıp, çalıştırılmaktadır.

Yukarıdaki her iki yapıda size garip ve zor geldiyse; aşağıdaki yapıyı da kullanabilirsiniz.

```
$(function(){
    // yapilacaklar
});
```

Seçim size kalmış hangisini kullanmak isterseniz. Lakin ilk verdiğim kullanım daha genel ve en çok kullanılan yapıdır. Bunu tercih etmenizi öneririm.

2.1 $.noConflict( )

$ işareti, jQuery için bir mahlastır dedik. Peki biz bu mahlası kendimiz verebilir miyiz ya da değiştirebilir miyiz? Veyahut bu mahlası başka bir kütüphane kullanıyorsa? O zaman ne yapacağız? Bu konu daha önce jQuery ekibi tarafından düşünülmüş ve bu mahlasın değiştirilebiliniyor olması sağlanmış. Bu işlem için $.noConflict( ) metodu kullanılmaktadır. Bu metoda daha yakından bakmak için aşağıdaki örneğinimizi inceleyelim.

```
var jayJay = $.noConflict();
jayJay(document).ready(function(){
    //yapilacaklar
});
```

Yukarıdaki örneğimizde bir değişken tanımladık ve bu değişkenimizi yeni jQuery mahlası olarak belirledik. Artık $ veya jQuery mahlaslarını kullanmamıza gerek yok. jayJay mahlasını kullanarak jQuery kullanmaya devam edebiliriz.

BÖLÜM 3: Selectors

Bu bölümde genel olarak Selector (Seçiciler)'den bahsetmeye çalışacağız. Önceki bölümlerde jQuery'nin çok güçlü bir seçici desteğine sahip olduğundan bahsetmiştik. Peki selector nedir? Selector'ler herhangi bir elementin seçilmesidir ya da işaret edilmesidir. Peki bu bizim ne işimize yarayacak derseniz de bir web sayfasında belli elementlerin seçilip value (değer)'lerinin değiştirilmesi gerekir. Bazen de varolan değerlerinin alınıp üzerinde belli işlemlerin gerçekleştirilmesi gerekir. Örneğin kullanıcı herhangi bir input (giriş) alanına bir değer girdi diyelim. Bu değer alınıp üzerinde işlemler yapılması, incelenmesi ve değişiklikler yapılması veya herhangi bir sayfaya gönderilmesi gerekebilir. İşte böyle durumlarda selector'ler kullanılır.

jQuery temel olarak CSS selector'lerini kullanır. jQuery selector'lerinden bahsetmeden önce bu işlem klasik JavaScript'te nasıl yapılır buna bakalım. En azından karşılaştırmalı olarak gidersek kafanıza daha çok oturacağından eminim.

Aşağıdaki örneğimizi inceleyelim.

```
<!DOCTYPE html>
<html>
<head>
<title>Example 3.1</title>
<meta charset="utf-8">
</head>
<body>

<script type="text/javascript">
function changeVal(){
    var value = document.getElementById("value").value;
    if(value != ""){
        var newValue = "Hello! " + value;
        document.getElementById("newValue").value = newValue;
    } else {
        document.getElementById("newValue").value = "";
    }
}
</script>

<input type="text" id="value" placeholder="input text here"
onkeyup="changeVal()">
<input type="text" id="newValue" placeholder="new value" readonly>

</body>
</html>
```

Yukarıdaki örneğimizde; basit olarak bir input alanından girilen değeri diğer input alanına, Hello! kelimesini başa ekleyerek yazdırdık. Gördüğünüz gibi klasik JavaScript'te herhangi bir element alanından değer alırken

document.getElementById(“elementName”) yapısını kullanıyoruz. Bu kullanım elementin id değerine göre alınıyor. Şayet bir input alanı ise .value, diğer herhangi bir tag (etiket) ise .innerHTML yapısını eklememiz gerekiyor.
Velev ki herhangi bir elementin id değerine göre değil de Tag Name (Etiket Adı) göre almak istersek:

```
document.getElementByTagName("tagName")
```

Elementin class (sınıf) ismine göre almasak istersekte:

```
document.getElementByClassName("className")
```

İfadelerini kullanırız. Klasik JavaScript'te bu işlemler oldukça uzun kod yazmalı ve sıkıntılı işlemlerdir. Lakin jQuery'de bu işlemler daha basit ve daha az kodla çözülebilinmektedir. Şimdi de jQuery'nin selector'larına bir bakalım.

jQuery'nin bir çok selector yöntemi mevcut. O yüzden bu bölüm biraz uzun olacak. O yüzden dikkatli okumanızı ve her uygulamayı yapmanızı öneririm.

jQuery konusunda henüz bahsetmediğimiz iki konu var. Bunlar; CSS ve Event (Olay). Selector konusunda anlatacaklarımızı daha iyi görmek için bu iki konudan iki küçük bilgi verelim. Zaten daha sonra bu konulardan detaylı olarak bahsedeceğiz.

jQuery ile CSS işlemleri aşağıdaki gibi yapılır.

```
$("selector").css("property", "value");
```

Yukarıdaki yapımızı incelersek; öncelikle bir elementi seçeceğiz (selector) daha .css ( ) metodu ile bu elementin CSS değerlerini değiştireceğiz. CSS metodundaki; property parametresi; elementin CSS kodu ve value parametresi ise o CSS kodunun değeri olacak. Yani herşey çok basit.

jQuery ile herhangi bir elemente (örneğin button) tıklanması (click) durumu da aşağıdaki gibi kullanılır.

```
$("selector").click(function () {
    // yapilacaklar
});
```

Yukarıdaki yapımızı inceleyecek olursak; ilk olarak bir element seçiyoruz (selector) daha sonra ise .click ( ) metodunu yazarak bu metodun parametresinde bir anonymous (anonim) fonksiyon tanımlayarak bu fonksiyon içerisinde yapılacak olanları yapabiliyoruz. Örneğin; .click ( ) ve .css ( ) metotlarını birleştirirsek;

```
$("selector").click(function () {
```

```
    $("selector").css("property", "value");
});
```

şeklinde bir yapı elde ederiz. Yani herşey çok basit. Şimdilik bu kavramlara çok fazla aşina değiliz. Zaten biz de en basit şekliyle anlattık. Paniğe gerek yok ilerleyen bölümlerde daha detaylı olarak bu iki kavramdan da bahsedeceğiz. Lakin siz şimdilik böyle kullanıldığını bilmelisiniz.
Herneyse uzunca bir selector konusuna başlayalım.

3.1 jQuery Selectors

3.1.1 Basic

jQuery'nin temel olarak CSS selector'larını kullandığından bahsetmiştik. Dolayısıyla bu bölümde CSS selector'larından bahsedeceğiz. Bu başlık altında anlatacaklarımız aslında tüm jQuery selector'larının temeli olacaktır.

3.1.1.1 element

jQuery element (veya tag - etiket) isimlerine göre seçebilir. Bu işlemi gerçekleştirmek için elementin adını vermek yeterlidir. Element seçme işlemi de aşağıdaki gibi gerçekleştirilir.

```
$("elementName")
```

Örneğin aşağıdaki yapımızı inceleyelim.

```
<div>Merhaba Dünya</div>
<label>Merhaba Dünya</label>
<span>Merhaba Dünya</span>
```

Yukarıdaki yapımızda; üç tane elementimiz var elimizde. Bu elementlerin doğrudan element isimlerini yazarak seçebiliriz. Şöyleki;

```
$("div")
$("label")
$("span")
```

Şimdi bu seçtiğimiz elementlerin CSS değerlerini değiştirebiliriz.

```
$("div").css("color", "red");
$("label").css("color", "red");
$("span").css("color", "red");
```

Görüldüğü gibi üç elementi de seçtik ve color değerlerini red yani kırmızı yaptık.

Peki elimizde; seçeceğimiz elementlerden birden fazla varsa ve bizde bu elementlerin hepsini seçmek istemiyorsak? Yukarıdaki yapımızda; aynı olan elementlerin hepsini seçer. Buna dikkat edelim. İlerleyen bölümlerde bu konuyu daha detaylı inceleyeceğiz. Bekleme de kalın.

3.1.1.2 #id

Elementlere verilecek olan id değerleri ile seçme işlemi yapılabilir. Bildiğiniz gibi CSS'de herhangi bir elementin id değerine göre CSS yazılacağı zaman # işareti kullanılır. jQuery'de de durum aynen böyledir. id değerine göre seçme işlemi aşağıdaki gibi gerçekleştirilir.

```
$("#elementIdName")
```

Örneğin elimizde aşağıdaki gibi bir yapımız olsun.

```
<div id="divID">Merhaba Dünya</div>
<label id="labelID">Merhaba Dünya</label>
<span id="spanID">Merhaba Dünya</span>
```

Yukarıdaki yapımızda; üç tane elementimiz var elimizde ve bu elementlerin de belirli bir id değerleri var. Bu elementlerin id değerlerini yazarak seçebiliriz. Şöyleki;

```
$("#divID")
$("#labelID")
$("#spanID")
```

Şimdi bu seçtiğimiz elementlerin CSS değerlerini değiştirebiliriz.

```
$("#divID").css("color", "yellow");
$("#labelID").css("color", "yellow");
$("#spanID").css("color", "yellow");
```

Görüldüğü gibi üç elementi de id değerlerine göre seçtik ve "color" değerlerini "yellow" yani "sarı" yaptık.

Aynı elementleri seçmede id değeri bir kriter olabilir. Her elemente farklı bir id değeri verirsek; Aynı elementleri birbirlerinden ayırarak seçebiliriz. Lakin bu yol da oldukça uzun olacaktır. Zira her bir elemente ayrı bir id değerini nerden bulacağız? Beklemede kalın, daha basit yollarla karşılaşacağız.

3.1.1.3 .class

Elementlerin class değerlerine göre de seçme işlemi gerçekleştirilir. Bildiğiniz gibi CSS'te bir class yazılırken başına nokta (.) işareti konulur. jQuery'de de aynen devam ediyoruz. class değerine göre seçme işlemi aşağıdaki gibi gerçekleştirilir.

```
$(".className")
```

Örneğin elimizde aşağıdaki gibi bir yapımız olsun.

```
<div class="divClass">Merhaba Dünya</div>
<label class="labelClass">Merhaba Dünya</label>
<span class="spanClass">Merhaba Dünya</span>
```

Şimdi bu seçtiğimiz elementlerin CSS değerlerini değiştirebiliriz.

```
$(".divClass").css("color", "red");
$(".labelClass").css("color", "red");
$(".spanClass").css("color", "red");
```

Görüldüğü gibi üç elementi de class değerlerine göre seçtik ve "color" değerlerini "red" yani "kırmızı" yaptık.

3.1.1.4 (*)

Bazı durumlarda tüm elementlerin seçilmesi gerekebilir. Böyle durumlarda asterix (*) operatörü ile sayfadaki tüm elementler seçilebilir. Asterix operatörünün kullanımı aşağıdaki gibidir.

```
$("*")
```

Asterix kullanılması durumunda elementlerde kullanılan; .class ve #id değerleri dikkate alınmaz. Tüm elementler seçilir. Örneğin sayfamızdaki tüm elementleri seçip, arkaplan renklerini değiştirmek istersek;

```
$("*").css("background-color", "yellow");
```

yazmamız yeterli olacaktır.

3.1.1.5 Multi Selection

Aynı anda birden fazla elementte seçilebilir. Bu element seçimleri; aynı #id, aynı .class veya aynı elementler olabileceği gibi herhangi bir kritere göre de olabilir. Multi selection işlemi ise oldukça basittir. Nasıl mı? Tıpkı aşağıdaki gibi.

```
$("selector1, selector2, selector3, selectorN")
```

Mesela aşağıdaki gibi bir yapıya sahip olduğumuzu varsayalım.

```
<div>Merhaba Dünya</div>
<label id="labelID">Merhaba Dünya</label>
<span class="spanClass">Merhaba Dünya</span>
```

Yukarıdaki yapımızdaki elementleri belirli kriterlere göre ayrı ayrı seçmek yerine tek hamlede alabiliriz. Yukarıdaki yapımızın background-color özelliğini yellow yapalım.

```
$("div, #labelID, .spanClass").css("background-color", "yellow");
```

Görüldüğü gibi herşey çok kolay. jQuery selector'larının temeli bu kadar. Bundan sonra göreceklerimiz aslında biraz da teferruat. Yani temelde yine bu 4 maddeyi takip edeceğiz. Buraya kadar herhangi bir sıkıntı yoksa; kendinizi tebrik edebilirsiniz.

3.1.2 Hierarchy

jQuery hiyerarşik olarak element seçimine olanak tanır. Şimdi hiyerarşik olarak element seçimine bir gözatalım.

3.1.2.1 ancestor descendant

ancestor (dede veya ata) ve descendant (torun) şeklinde olan yapıları jQuery ile kolayca seçebiliriz. Yani burada iç içe bulunan elementler söz konusudur. Örneğin aşağıdaki yapımızı inceleyelim.

```
<div>
    <label>Hello World!</label>
    <span>Hello Master!</span>
</div>
```

Yukarıdaki yapımızda; ancestor olan; div elementidir. Bu div elementinin descendant'ları ise; label ve span elementleri. Örneğin başka elementlerde bu ana div elementinin içerisinde olabilirdi. Aklınıza gelebilecek her türlü senaryo yazılabilir. Şimdi bu div elementinin içerisindeki üye (torun) elementleri nasıl seçeceğimize bir bakalım.

Aşağıdaki yapımızı dikkatli inceleyelim.

```
$("div label")
```

Yukarıdaki yapımızda; önce div elementini yazdık daha sonra bir boşluk bıraktık ve bu div elementi içerisinde bulunan label elementini yazdık. Şayet span elementini de almak isteseydik;

```
$("div label,span")
```

yazmamız yeterlidir. Şayet başka elementlerde olsaydı onları da yukarıdaki yapımızda olduğu gibi "multi selection" yöntemiyle alabilirdik.

Burada dikkat etmeniz gereken nokta; element adı, #id, class v.s herhangi birisini yazabilirsiniz. Örneğin; ancestor elemetimizin adı yerine bir .class veya #id değerini yazabilirdik. Yani buradaki seçimde özgürsünüz. Sadece iç içe bulunan elementlerde önce; ancestor (dede) elementi seçip bir boşluk bırakıp daha sonra bu ancestor elementin sahip olduğu descendant (torun) elementi veya elementleri seçmeliyiz.

Mesela yukarıdaki örneğimizde seçtiğimiz elementlerin arkaplanını "yellow" yapalım.

```
$("div label,span").css("background-color", "yellow");
```

Yukarıdaki yapımıza ek olarak asterix işaretini de kullanarak; ancestor içerisindeki tüm elementleri seçebiliriz.

```
$("div *").css("background-color", "yellow");
```

3.1.2.2 parent > child

parent (ata) ve child (çocuk) seçimi de ancestor descendant'a benzer yapıdadır. Burada da yine iç içe geçmiş elementler büyüktür (>) işareti yardımıyla seçilebilir. Öncelikle parent elementin belirleyici özelliği yazılır ki belirleyici özellik; element adı, #id veya .class değeri olabilir. Keza child içinde aynı şeyler geçerlidir. Daha sonra büyüktür işareti konulur ve child elementin belirleyici özelliği yazılır.

Elimizde aşağıdaki gibi bir yapımız olduğunu farzedelim.

```
<form>
    <input type="text" name="username">
    <input type="password" name="password">
</form>
```

Yukarıdaki yapımızda; bir form elementimiz var ve bu form elementimiz altında iki tane input alanımız var. Bu iki input alanı aslında form alanının child'ı yani çocukları veya bir nevi elementleri. Şimdi bu input alanını seçmek istersek;

```
$("form > input")
```

yazmamız yeterlidir. Her zaman yaptığımız gibi seçtiğimiz bu elementlerin arkaplanlarını kırmızı yapalım.

```
$("form > input").css("background-color", "red");
```

Buraya kadar herşey güzel, normal tamam ama parent > child yönteminin, ancestor descendant yönteminden farkı ne? İkisi de aynı kapıya çıkıyor diyebilirsiniz. Şimdiye kadar anlattıklarımızla bu kanıya varmakta tamamen haklısınız. Hemen durumu açıklamaya çalışalım. Yukarıdaki örneğimize bir div alanı ekleyelim ve bu div alanına da bir input alanı daha ekleyelim. Yani yapımız aşağıdaki gibi olsun.

```
<form>
    <input type="text" name="username">
    <input type="password" name="password">
    <div>
        <input type="text" name="nickname">
    </div>
```

```
</form>
```

Peki şimdi bu sonradan eklediğimiz div'in içinde elementi parent > child yöntemiyle alabilir miyiz? Yani $("form > input") dediğimiz zaman div içindeki input alanını da seçebiliyor muyuz? Cevap: Hayır! Peki neden? Çünkü div elementi form alanının doğrudan child'ı ama div içerisindeki input alanı doğrudan form alanının child'ı değil! Buraya dikkat! div içerisindeki input alanı div'in child'ı. Yani form alanının torunu oluyor bir nevi. Dolayısıyla burada div içerisindeki değer ancestor descendant yöntemine giriyor. Yani;

```
$("form  input")
```

dersek; form elementinin altındaki tüm elementler, ancestor descendant ile seçilebilir. Peki parent > child yöntemi burada işe yaramaz mı? Tabi ki yarar lakin kod uzunluğu artar. Bu işlemi de aşağıdaki şekilde gerçekleştirebiliriz.

```
$("form > div > input")
// veya
$("form > input, form > div > input")
```

Şeklinde kullanabiliriz. Hangisi işimize yararsa!

3.1.2.3 prev + next

prev + next seçicisi, bahsetmiş olduğumuz diğer iki seçiciye göre biraz farklı davranır. Burada komşuluk ilişkisi önemlidir. Örneğin elimizde aşağıdaki gibi bir yapımız olsun.

```
<label>Name:</label>
<input type="text" name="name">
```

Yukarıdaki yapımız bir div içerisinde de olabilir. Yani;

```
<div>
    <label>Name:</label>
    <input type="text" name="name">
</div>
```

Ya da:

```
<div>Merhaba!</div>
<label>Emre Bey</label>
```

şeklinde de olabilir. Velhasıl; bu elementler arasında bir komşuluk ilişkisi mevcut. İşte böyle komşuluk durumlarında; prev + next seçici yöntemi uygulanmalıdır. Burada dikkat edilecek nokta; prev + next ile seçilen elementlerde; next elementi dikkate alınır ve seçilir. Yani son yapımıza dönecek olursak; label elementi seçilecektir.

O halde label elementini seçelim.

```
$("div + label")
```

Yukarıdaki seçmiş olduğumuz label elementimizin CSS değerleriyle oynayalım.

```
$("div + label").css("font-family", "helvetica");
```

Burada dikkat edilecek bir diğer nokta ise şudur; prev + next yönteminde sadece next seçilir. next için belirtilen kriterlere uyan varsa bu alınmaz. Sadece ilk next alınır. Buna dikkat edelim.

3.1.2.4 prev ~ siblings

prev + next konusunda; sadece bir tane next alınır demiştik. İşte prev ~ siblings yöntemine göre bütün next'ler alınır. Yani elimizde aşağıdaki gibi bir yapımız olduğunu farz edelim.

```
<div>Kardeşler</div>
<label>Birinci kardeş</label>
<label>İkinci kardeş</label>
<span>Aranıza girdim, k. bakmayın!</span>
<label>Üçüncü kardeş</label>
```

Yukarıdaki yapımızda div ve sonrasında gelen iki label kardeşler. Çünkü yanyana bulunuyorlar. span elementi ise bir üçüncü olan label ile diğer label'lerin arasında girdi. Ama mühim değil. Onlar yine kardeşler. Çünkü hepsi yanyana bulunuyorlar. Araya giren bir span elementi bu üçlünün arasını bozamaz. Üçüncü label sadece gurbete düştü.

Herneyse, yukarıdaki yapımızı; prev ~ siblings'e göre seçelim.

```
$("div ~ label")
```

Yukarıdaki jQuery selector ile div ve devamında bulunan (yan yana) bütün label'leri seçecektir.
Bu label'ler seçilecek lakin prev olan div'e dokunulmayacaktır. Şimdi bu seçtiğimiz div'lerin CSS değerlerini değiştirelim.

```
$("div ~ label").css("color", "red");
```

Görüldüğü gibi herşey çok basit

3.1.3 Basic Filter

Temel Filtreleme'de işin derinlerine daha çok iniyoruz. Buraya kadar olan bölüm bizim için yeterli değil. Dolayısıyla jQuery ekibi oturmuş ve her türlü durum için selector'lar hazırlamışlar, sağ olsunlar.

3.1.3.1 :first

:first ile belirlenen kriterdeki elementlerden ilki seçilir. Yani elimizde aşağıdaki gibi bir yapı varsa;

```
<div>div 1</div>
<div>div 2</div>
<div>div 3</div>
<div>div 4</div>
<div>div 5</div>
```

Bu yapıdaki ilk div elementi seçilmek isteniyorsa :first kullanılmalıdır. Peki nasıl kullanılmalıdır? Cevabı basit ilk olarak seçilmek istenilen elementin adı yazılmalı ve daha sonra :first yazılmalıdır. Yani aşağıdaki gibi kullanılmalıdır.

```
$("div:first")
```

Görüldüğü gibi ilk div elementini seçtik. Bu elementin CSS değerleriyle oynayalım.

```
$("div:first").css("background-color", "red");
```

Peki ekstra bir bilgi olması açısından; birden fazla CSS değerini oynayabilir miyiz? Cevap: Evet!

```
$("div:first").css("background-color", "red").css("color", "white");
```

Yukarıdaki kodlarımıza dikkatli bakarsanız ard arda css ( ) metodunu kullandık ve böylece belirlediğimiz elementin hem arkaplanını hemde rengini değiştirdik. Tabi ki CSS için uzunca bir satır. Emin olun CSS bölümüne geçtiğimizde bu fazlalıklardan kurtulacağız.

3.1.3.2 :last

:last, :first'un aksine en sondaki elemanı seçer. Yapımız yine aynı olsun, aşağıdaki gibi.

```
<div>div 1</div>
<div>div 2</div>
<div>div 3</div>
<div>div 4</div>
<div>div 5</div>
```

Bu yapımızdaki son div elementini seçmek için aşağıdaki yöntemi kullanabiliriz.

```
$("div:last")
```

Görüldüğü gibi son div elementini seçtik. Şimdi CSS değerleriyle oynayabiliriz.

```
$("div:last").css("background-color", "red").css("color", "white");
```

Aynı anda hem :first hemde :last yöntemlerini de kullanabiliriz.

```
$("div:first, div:last").css("background-color", "red").css("color",
"white");
```

Bunun dışında buraya kadar olan bölümdeki anlattıklarımızdan herhangi bir senaryo içerisinde de bu durumları gerçekleştirebilirsiniz.

3.1.3.3 :not ( )

:not ( ) bir metot gibi çalışır. Yani belirli değerleri kontrol eder lakin olumsuzluğunu. Örneğin hep kullandığımız yapımızı tekrar ele alalım.

```
<div>div 1</div>
<div>div 2</div>
<div>div 3</div>
<div>div 4</div>
<div>div 5</div>
```

Bu yapımızda :not( )'ı kullanmak istersek; örneğin; ilk element olmayan veya son element olmayan şeklinde bir kriter belirtebiliriz. Bir örnek yapalım. Şimdi aşağıdaki örneğimizi inceleyelim.

```
$("div:not(:first)")
```

Yukarıdaki örneğimizde; ilk div elementi dışındaki tüm div elementleri seçilecektir. Peki ilk ve son div elementleri desek? O zaman da aşağıdaki gibi bir yapı kurmamız gerekir.

```
$("div:not(:first, :last)")
```

Görüldüğü gibi ilk ve son element dışındaki tüm elementleri aldık. Yani aldığımız elementler; div 2, div 3 ve div 4. Yine bu elementler için CSS değerleri değiştirilebilir.

3.1.3.4 :even

:even ile çift sayıdaki elementler seçilebilir. Ki zaten even Türkçe'de çift anlamına gelmektedir. Burada dikkat etmeniz gereken konu ise; jQuery'de element sayılarının indisi 0'dan başlar. Yani tıpkı bir dizi mantığında.

Örneğin elimizde bir table elementi olsun.

```
<table>
    <tr><td>0</td></tr>
    <tr><td>1</td></tr>
    <tr><td>2</td></tr>
```

```
    <tr><td>3</td></tr>
    <tr><td>4</td></tr>
</table>
```

Bu table elementindeki çift sıradaki row (satır) seçelim.

```
$("tr:even")
```

Görüldüğü gibi çok basit bir şekilde çift sıradaki satırları seçtik. Bunun dışında yukarıdaki gibi kullanılabileceği gibi aşağıdaki gibi de kullanılabilir.

```
$("table tr:even")
```

table elementinin altındaki satırlar için bu işlemi gerçekleştir dedik.

Bazı web sayfalarında table'ların çok şekilli olduğunu farketmişsinizdir. Bu işlemi CSS ile gerçekleştirebileceğiniz gibi çoğunlukla jQuery ile gerçekleştirilir. Örneğin biz de kendi table'ımızı basit bir biçimde şekillendirelim.

```
$("tr:even").css("background-color", "#F3F3F3");
```

Bu işlemi sadece table elementinde yapmak zorunda değilsiniz. Aklınıza gelebilecek herhangi bir elemente (mümkünse çift sayıda olsun, 0'da dahil) uygulayabilirsiniz.

3.1.3.5 :odd

:event'ın aksine :odd da tek sayıdaki elementleri seçer. :event için kullandığımız yapıyı tekrar ele alırsak;

```
$("tr:odd")
```

şeklinde kullanabiliriz.
CSS değerlerini de değiştirelim.

```
$("tr:odd").css("background-color", "#000000");
```

3.1.3.5 :eq(index)

:eq(index) yönteminde ise; nokta atışı yöntemiyle element seçilir. Burada index elementin sıra numarasıdır. eq ise equal yani eşittir anlamına gelmektedir. Yani belirlediğimiz elementin sıra numarasını girerek elementi seçebiliriz.

Elimizde yine bir table olsun ve table'ın 3 numaralı tr elemanını seçelim.

```
$("tr:eq(3)")
```

Görüldüğü gibi hemen seçtik. Peki elimizde başka elementler de varsa? Farketmez her türlü elementi bu yöntem ile seçebilmekteyiz.

Örneğin aşağıdaki gibi bir yapımız olsun.

```
<label>label 0</label>
<label>label 1</label>
<label>label 2</label>
<label>label 3</label>
<label>label 4</label>
```

Bu label elementlerden üçüncü sıradakini seçmek istersek;

```
$("label:eq(3)")
```

dememiz yeterli olacaktır. Peki 0, 2 ve 4 numaralı label'leri seçmek istersek? Yine Multi selection'ı uygulamamız gerekir.

```
$("label:eq(0), label:eq(2), label:eq(4)")
```

3.1.3.6 :gt(index)

:gt(index), belirlenen kriterin üzerini alacaktır. Yani 3'üçüncü tr'nin üstündekileri al veya 5'inci sıradaki span elementinin üzerindeki span elementleri al dediğimizde devreye girecektir. Hadd-i zatında gt, greater then manasına gelip -den büyük anlamı taşımaktadır. Buradaki index ise bizim şu numaranın üzerindekileri al'daki numaramız olacaktır.

Yapımızı yine kullanalım.

```
<label>label 0</label>
<label>label 1</label>
<label>label 2</label>
<label>label 3</label>
<label>label 4</label>
```

Örneğin; 2 numaralı label elementinin üzerindeki yani 3 ve 4 numaralı label'leri seçelim.

```
$("label:gt(2)")
```

Şeklinde kullanmamız yeterlidir. Keza bu işlemi diğer elementler içinde belirlenen kriterlere göre yapabiliriz.

3.1.3.7 :lt(index)

:lt(index), :gt(index) yönteminin tam tersini yani belirlenen kriterin altındakileri alacaktır. lt, less then anlamına gelir ve -den küçük(az) manalarını taşımaktadır.

Yapımız yine aşağıdaki gibi olsun.

```
<label>label 0</label>
<label>label 1</label>
<label>label 2</label>
<label>label 3</label>
<label>label 4</label>
```

Örneğin; 2 numaralı label elementinin altında yani 0 ve 1 numaralı label'leri seçelim.

```
$("label:lt(2)")
```

şeklinde kullanmamız yeterlidir.

3.1.3.8 :header

:header yöntemi h1, h2, h3, h4, h5 ve h6 header yani başlıklarını seçer.

Elimizde aşağıdaki gibi header elementleri olduğunu varsayalım.

```
<h1>h1</h1>
<h2>h2</h2>
<h3>h3</h3>
<h4>h4</h4>
<h5>h5</h5>
<h6>h6</h6>
```

:header yöntemi ile hepsini seçebiliriz. Şöyleki:

```
$(":header")
```

Görüldüğü gibi oldukça basit. Buraya kadar incelediğimiz diğer selector'ları da işin içine katabiliriz.

3.1.3.9 :animated

:animated yöntemi, jQuery'nin sahip olduğu animation (animasyon) özelliğinin kullanıldığı elementleri seçer. Henüz animate kavramına değinmedik ama yine de aşağıdaki örneğimizi inceleyelim.

```
<div>Hello World!</div>

function animatedDiv() {
  $("div").slideToggle("slow", animatedDiv);
}
animatedDiv();

$("div:animated").css("color", "red");
```

Yukarıdaki örneğimizde; herhangi bir div elementi; slideToggle ( ) metodu ile animasyon özelliği kazandırılmıştır. Daha sonra bu metot bir fonksiyona bağlanmış ve bu fonksiyon devamlı çağrılmaktadır. Yani animasyon sürekli olacaktır. :animated ile bu animasyon özelliği kazandırılmış olan div'i seçebiliyoruz.

3.1.4 Filter Contents

"İçeriğe Göre Filtreleme"de ise; belirlenen herhangi bir text değerine göre veyahut diğer herhangi selector değerine göre, seçme işlemi gerçekleştirilebilir. Çeşitli yöntemler mevcut. Şimdi bu yöntemlere sırasıyla değinelim.

3.1.4.1 :contains (text)

:contains (text) yönteminde; belirlenen bir elementin text içeriğine göre seçme işlemi gerçekleştirilebilir. text olarak bir parametre göndermeliyiz.

Örneğin aşağıdaki gibi bir yapımız olsun.

```
<div>emrecan-oztas</div>
```

Yukarıdaki div elementimize adımı yazdım. Şimdi belirlediğim bir text değerine göre seçme işlemini gerçekleştirelim.

```
$("div:contains('emre')")
```

Yukarıdaki yazdığımız kodlarımızda; div elementi içerisinde geçen emre kelimesini arıyoruz. Şayet bu aradığımız div içerisinde "emre" kelimesi geçiyorsa (ki geçiyor) çeşitli işlemleri gerçekleştirebiliriz. Şuana kadar selector üzerinde durduk. Sadece click( ) ve css( ) metotlarını biliyoruz. Dolayısıyla biz de css metodunu kullanalım. Örneğimiz aşağıdaki gibi olacaktır.

```
$("div:contains('emre')").css("color", "red");
```

3.1.4.2 :empty

:empty yöntemi; belirlenen elementin içerisinde child (çocuk) bir element veya bir text değeri yoksa seçilecektir. Bu yöntem genellikle table elementinde göze hoş gelen bir yapı kurmak için kullanılır.

Aşağıdaki gibi bir yapımız olduğunu varsayalım.

```
<table>
    <tr>
        <td>c0</td>
        <td>c1</td>
    </tr>
    <tr>
        <td></td>
        <td>c3</td>
```

```
    </tr>
    <tr>
        <td>c4</td>
        <td></td>
    </tr>
</table>
```

Bu yapımızda belirli td alanları boş yani empty durumunda. Dolayısıysa bu empty olan td'leri seçebiliriz.

```
$("td:empty")
```

Görüldüğü gibi empty olan td değerlerini seçtik. İsterseniz boş olan td'lerin arkaplan renklerini kırmızı yapalım.

```
$("td:empty").css("background-color", "red");
```

3.1.4.3 :has (selector)

:has (selector) yönteminde; belirtilen elementin içerisinde parametre olarak verilen element seçilir. Şayet parametrede belirtilen element varsa.
Örneğin aşağıdaki gibi bir yapımız olduğu varsayalım.

```
<div>
    <label>Burası bir label alanı </label>
</div>
```

Yukarıdaki yapımızda; div elementi içerisinde bir label alan bulunmakta. Yani kodlarımız aşağıdaki gibi olacaktır.

```
$("div:has(label)")
```

Yukarıdaki kodlarımızda; div içerisinde; label elementi var ise seçilecektir. Yoksa seçilmeyecektir. Örneğin kodlarımız aşağıdaki gibi de olabilirdi.

```
<div>
    <p>
        <label>Burası bir label alanıdır!</label>
    </p>
</div>
```

Yukarıdaki kodlarımızda; div içerisinde bir p elementi ve bu p elementi içerisinde de bir label elementi var. Yani div alanı en az bir tane label içeriyor. Yukarıdaki kodlarımızı aynen tekrar bu durum içinde kullanabiliriz.

3.1.4.4 :parent

:parent yöntemi parent yani ata olan içerisinden child (çocuk) elementler veya text bulunan elementleri seçer.

Şöyleki elimizde aşağıdaki gibi bir yapı olduğunu farkedersek;

```
<table>
    <tr><td>col 1</td><td>e</td></tr>
    <tr><td>col 2</td><td>e</td></tr>
</table>
```

Bu yapıdaki td kolonlarından içerisinde text bulunan kolonlar parent olacaktır. Ayrıca bu kolonlar içerisinde herhangi element veya elementler bütünü de olabilirdi. Dolayısıyla bu td kolonlarını seçmek için aşağıdaki kodumuzu yazabiliriz.

```
$("td:parent")
```

3.1.5 Filter Aspect

"Görünürlük Filtreleme"de; elementlerin görünülürlüğüne göre seçme işlemi gerçekleştirilir. Örneğin herhangi bir elementin görünülürlüğü kapalı olabilir. Dolayısıyla böyle durumlarda bu filtreleme devreye girer.

3.1.5.1 :hidden

:hidden yönteminde; örneğin herhangi bir element hidden durumunda olabilir. İşte böyle durumlarda bu yöntem devreye girer.

Örneğin elimizde aşağıdaki gibi bir yapı olabilir.

```
<div hidden>Hello World!</div>
```

İşte yukarıdaki gibi bir element hidden durumunda ise aşağıdaki kodumuzla bu hidden durumundaki elementi seçebiliriz.

```
$("div:hidden")
```

Dilersek bu elementin görünülürlüğünü aşağıdaki kod ile kaldırabiliriz.

```
$("div:hidden").show();
```

.show ( ) metodundan henüz bahsetmedik. Lakin şimdilik bu metodun bir elementi görünür duruma getirdiğini bilmemiz yeterlidir.

3.1.5.2 :visible

:visible yöntemi, :hidden yönteminin aksine görünür olan elementleri seçer. Yani hidden durumunda olan elementleri seçmez.
Örneğin elimizde aşağıdaki gibi bir yapı olduğunu farz edelim.

```
<div hidden>Hello World!</div>
<div>Hello World!</div>
```

Yukarıdaki yapımızda; birinci div elementi hidden durumunda iken ikinci div elementi görünür durumda. Yani "hidden" değil. Dolayısıyla aşağıdaki kod ile bu görünür olan div elementini seçebiliriz.

```
$("div:visible")
```

3.1.6 Filter By Value

"Değere Göre Filtreleme"de; elementin attribute (özellik veya değer olarak çevirebiliriz)'ne göre seçme işlemi gerçekleştirilebilir. Örneğin elementlerin attribute'leri değişebilir. Mesela bir div elementi için konuşursak; id ve class değerleri en temel attribute'leri olacaktır. İşte bu ve buna benzer çeşitli yöntemlerle elementler seçilebilir. Şimdi detaylıca bu konuya değinelim.

3.1.6.1 [attribute]

Sanırım bir örnek üzerinden gidersek daha yerinde olacaktır. Elimizde aşağıdaki gibi bir yapımız olduğunu varsayalım.

```
<div hidden>Hello</div>
<div>Merhaba</div>
```

Yukarıdaki yapımızda; ilk div elementinin attribute'u hidden lakin diğer div'in herhangi bir attribute özelliği kullanılmamış. Atrribute kullanılan elementi seçmek istersek aşağıdaki gibi bir yapı kullanmamız gerekir.

```
$("div[hidden]")
```

Yukarıdaki kod parçasında; hidden attribute kullanılan div'i seçtik.

Henüz bir elementin değerini almayı görmek lakin aşağıdaki kodlarda; bir elementin değerini alıyoruz. Yani şimdilik bunu bilelim.

```
var value = $("div[hidden]").text();
```

Dilersek value değişkenini yazdırabiliriz. Artık value değişkeninde; "Hello" özelliği bulunmaktadır.

Konuyu daha iyi anlayabilmek için başka bir yapıyı ele alalım. Örneğin yapımız aşağıdaki gibi olsun.

```
<div id="divID">Hello</div>
```

Yukarıdaki div elementinde; id attribute'ine bir değer atanmış. Şimdilik bu değer bizi ilgilendirmiyor. Bu div elementinin id attribute kullandığı için biz bu elementi bu özelliğinden dolayı seçebiliriz. Nasıl mı? Aşağıdaki kodlarımıza bakalım.

```
$("div[id]")
```

3.1.6.2 [attribute=value]

[attribute=value] yönteminde ise; attribute'nin değerine göre seçme işlemi gerçekleştirilir. Örneğin elimizde bir input elementi varsa ve bu input elementinin type özelliği text ise biz buradan bu elementi seçebiliriz.

```
<input type="text">
```

Yukarıdaki input elementi text olduğu için bunu aşağıdaki kod parçası ile kolayca seçebiliriz.

```
$("input[type=text]")
```

input elementinin diğer özelliklerine göre de seçme işlemini gerçekleştirebiliriz.

```
<input type="text" name="name">
```

Yukarıdaki elementin name özelliğine seçme işlemini gerçekleştirelim.

```
$("input[name=name]")
```

Elementimize basit olması için name ismini verdik. Bu ismi "isim" olarakta değiştirebiliriz veyahut başka isim de verebiliriz. Sadece burda bilmemiz gereken name attribute'e hangi ismi vermiş isek o ismi yazmamız gereklidir, seçim işlemi yaparken.

3.1.6.3 [attribute!=value]

[attribute!=value] yöntemi, [attribute=value] yönteminin tam tersidir. Yani; elementin bir attribute'i bir değere eşit değilse durumunda seçil işlemi gerçekleştirilecek.

[attribute!=value] konusunda ele aldığımız örneğimizi yeniden ele alırsak;

```
<input type="text" name="name">
```

Yukarıdaki elementimizi name özelliğine göre seçelim.

```
$("input[name!=isim]")
```

Gördüğünüz gibi elementimizin name attribute'i "name" lakin biz bu yöntemle bu elementi seçerken farklı bir değer yazdık ("isim"). Çünkü burada ! işareti değilse anlamında olacaktır. Yani yine elementimizi bu yöntem ile seçebiliyoruz.

3.1.6.4 [attribute^=value]

[attribute=value] yönteminde; yine elementin herhangi bir attribute'ine göre seçim işlemi yapılır. Lakin burada vereceğimiz value değeri bu attribute değerinin başında bulunmalıdır. Yani value değeri elementin attribute değerinin başında geçmeli. Örneğin;

```
<label for="inputName">Name:</label>
<label for="inputSurname">Surname:</label>
<label for="inputAge">Age:</label>
<label for="inputMail">e-Mail Address:</label>
```

gibi bir yapı olsun. Bu yapımızda; label elementlerinin for attribute'inde değişik isimlendirmeler yapılmış. Fakat hepsi input ile başlıyor. Yani şuan içinde bulunduğumuz yöntem ile bunu kolayca gerçekleştirebiliriz. Aşağıdaki kodlarımıza bakalım.

```
$("label[for^=input]")
```

Görüldüğü gibi label elementlerini for attribute'ine göre kolayca seçtik. Burada 5 farklı karaktere göre de seçim yapabilirdik.

3.1.6.5 [attribute$=value]

[attribute$=value] yöntemi de [attribute=value] tam tersi; belirtilen value değeri elementin attribute değerinin sonunda aranır. Yani yapı tamamen aynı. Örneğimizi yeniden ele alalım.

```
<label for="inputName">Name:</label>
<label for="inputSurname">Surname:</label>
<label for="inputAge">Age:</label>
<label for="inputMail">e-Mail Address:</label>
```

Yapımızın for attribute değerinin son karakter setlerine bakalım. Bir ve ikinci for attribute'lerinde "name"ler ortak. Öyle değil mi? Cevabınız "evet" ise yanıldınız! Bommm! Bütün puanlar gitti. jQuery, case-sensitive'dir. Yani büyük / küçük harfe duyarlıdır. Dolayısıyla; name ve Name aynı şeyler değildir. Burada aynı olan karakterler; ame olacaktır. Bir, iki ve üç for attribute'lerinde "e"ler ortak. Sonuncu biraz asi olduğu için diğerlerinden farklı.

```
$("label[for$=ame]")
```

Bir ve ikinci label elementini seçtik.

```
$("label[for$=e]")
```

Bir, iki ve üçüncü label elementini seçtik.

İlerleyen kısımlarda kriterleri ayrı ayrı belirtmek yerine; Multi Selection ile belirleyeceğimiz zaman daha rahat edeceğiz. Şimdi böyle olduğunu bilmemiz kafî. Beklemede kalın!

3.1.6.6 [attribute*=value]

[attribute*=value] yöntemi attribute değerinin içerisinde geçen herhangi bir karakter veya karakter setlerine göre seçme işlemi gerçekleştirir. Yapımızı tekrar ele alalım.

```
<label for="inputName">Name:</label>
<label for="inputSurname">Surname:</label>
<label for="inputAge">Age:</label>
<label for="inputMail">e-Mail Address:</label>
```

Bu yapımızda; örneğn pu karakterlerine göre seçme işlemini gerçekleştirelim.

```
$("label[for*=pu]")
```

Seçtiğimiz bu elementlerin arkaplanları ve rengini boyayalım.

```
$("label[for*=pu]").css("background-color", "black").css("color",
"white");
```

3.1.6.7 Multi Attribute Selection

Daha önce değinmiştik. attribute değerlerine göre seçme işlemini birden fazla kriter belirleyerek gerçekleştirebiliriz. Şimdi bu işlem nasıl gerçekleştirilir. Bunun üzerinde duralım.

```
<label for="name">Name:</label>
<label for="surname">Surname:</label>
<label for="age">Age:</label>
<label id="about">About:</label>
```

Yukarıdaki yapımızı belirli kriterlere göre seçelim. Seçimimiz aşağıdaki gibi olacaktır.

```
$("label[for='name'][for!='aname'][for^='na'][for$='me']")
```

Yukarıdaki örnek kodlarımızda; birinci label alanını, değişik kriterler vererek seçtik. Eğer birbirine çok benzeyen yapılar varsa elinizde; bu yöntem oldukça yararlıdır.

3.1.7 Child Filters

Bu başlık altında bir elementin sahip olduğu childs (çocuklar)'ına göre filtreleme ya da seçme işlemi yapacağız.

3.1.7.1 :nth-child(index/even/odd/equation)

:nth-child ( ) yönteminde; belirlenen kriterlere göre child yani çocuk-üye elemanlar seçilebilir.

Aşağıdaki gibi bir yapımız olduğunu farz edelim.

```
<ul>
    <li>İstanbul</li>
    <li>Ankara</li>
    <li>İzmir</li>
    <li>Bursa</li>
    <li>Adana</li>
    <li>Antalya</li>
    <li>Konya</li>
</ul>
```

Bu yapımızda; ul elementinin tam yedi tane child-li elementi var. İşte bu benzeri durumlarda :nth-child ( ) yöntemi devreye girer. Burada belirtmek istediğim bir şey var. Bu yöntem sadece; listelerde kullanılmaz. Aklınıza gelebilecek her yerde kullanabilirniz. Yeterli şartlar elverişli olsun.

Herneyse, mesela yapımızdaki 3 numaralı child elemanı seçelim.

```
$("ul li:nth-child(3)")
```

Görüldüğü gibi çok basit. Sadece seçeceğimiz elementin sıra numarasını girmemiz yeterli.
Sıra numarasına göre seçebiliyoruz. Tamam burda sıkıntı yok. Bunun dışında even ve odd'a göre de seçim yapabiliyoruz.

Örneğin; even'a göre seçelim,

```
$("ul li:nth-child(even)")
```

odd'a göre seçelim,

```
$("ul li:nth-child(odd)")
```

Bu yöntemin bir diğer güzel olan yanına da değinelim. Örneğin; belirli katlardaki sayılar verebiliriz, eleman seçimi için. Örneğin aşağıdaki yapımıza bakalım.

```
$("ul li:nth-child(3n)")
$("ul li:nth-child(3n+1)")
$("ul li:nth-child(2n)")
$("ul li:nth-child(2n-1)")
```

Yukarıdaki örnek kodlarımızda; belirli katlardaki sayılar ile çalıştık. Bu katlardaki sayılarda bulunan elemanlar seçilecektir. Örneğin; birinci yapımızda 3n, ikinci yapımızda 3n+1, üçüncü yapımızda 2n ve son yapımızda 2n-1 değerlerini verdik. Burada dikkat edilmesi gereken nokta; herhangi bir sıralı liste verildiği zaman; indis değeri 1'den başlar.

3.1.7.2 :first-child

:first-child yönteminde ise ilk child eleman seçilir. Daha açıklayıcı olması açısından yapımızı tekrar ele alalım.

```
<ul>
    <li>İstanbul</li>
    <li>Ankara</li>
    <li>İzmir</li>
    <li>Bursa</li>
    <li>Adana</li>
    <li>Antalya</li>
    <li>Konya</li>
</ul>
```

Yukarıdaki yapımızda; örneğin ilk child eleman İstanbul dolayısıyla bu elemanı seçmek için aşağıdaki kodu yazmamız yeterli olacaktır.

```
$("ul li:first-child")
```

3.1.7.3 :last-child

:last-child yöntemi, :first-child yönteminin tam tersi, yani en sondaki child elemanı seçecektir.

Örneğimize devam edersek, yazacağımız kodlarımız aşağıdaki gibi olacaktır.

```
$("ul li:last-child")
```

3.1.7.4 :only-child

:only-child yöntemi; belirtilen parent elementinin yalnız ve yalnızca bir tane child elementi olması durumunda bu tek elemanı seçer. Aksi halde herhangi bir işlem gerçekleştirmez.

```
<div>
    <p>Hello there!</p>
</div>
```

Yukarıdaki yapımızda div (parent) elementinin sadece bir tane child elemanı var. O da p elementi. Bizim bu p elementini seçmek için aşağıdaki kodları yazmamız yeterli olacaktır.

```
$("div p:only-child")
```

3.1.8 Forms

Formlar tipik bildiğimiz, kullanıcı ile tasarladığımız sistemin iletişime geçmesini sağlayan elementler. İşte bu elementleri seçmek için belirli seçiciler var. Bu başlık altında bunlara değineceğiz.

3.1.8.1 :input

:input yönteminde; bir web sayfasında bulunan tüm input alanları seçilir. Bu input alanlarına textarea, select, button ve aklınıza gelen kullanıcıyla iletişime geçebilecek tüm input alanlarını seçebilir.

Burada size farklı bir şey göstermek istiyorum. Örneğin aşağıdaki gibi bir yapımız olduğunu farz edelim.

```
<input type="text">
<input type="text">
<input type="text">
```

Yukarıdaki yapımızdaki tüm input değerlerini aşağıdaki yöntem ile seçebiliriz.

```
$(":input")
```

Burada göstermek istediğim nokta diğer ayırıcı yöntemleri kullanarak, tüm input alanlarını seçmek yerine belirlediğimiz kriterlere göre seçebiliriz. Aşağıdaki yapımızı inceleyelim. Daha önce gördüğümüz yapılar olduğu için herhangi bir açıklama yapmak yerine yorumu sizlere bırakıyorum.

```
$(":input:even").css("background-color", "red");
$(":input:odd").css("background-color", "red");
$(":input:eq(0)").css("background-color", "red");
$(":input:first").css("background-color", "red");
$(":input:second").css("background-color", "red");
```

ve aklınıza gelebilecek dahası...

3.1.8.2 :text

:text, type="text" olan elementleri seçer. Kullanımı aşağıdaki gibi gidir.

```
$(":text")
```

Örneğin; seçilen type="text" elementinin CSS değerleri değiştirilebilinir.

```
$(":text").css("text-transform", "capitalize");
```

3.1.8.3 :password

:password, sayfa üzerindeki type="password" alanlarını seçer. Kullanımı aşağıdaki gibidir.

```
$(":password")
```

3.1.8.4 :radio

:radio, type="radio" olan elemetleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":radio")
```

3.1.8.5 :checkbox

:checkbox, type="checkbox" olan elementleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":checkbox")
```

3.1.8.6 :submit

:submit, type="submit" olan elementleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":submit")
```

3.1.8.7 :image

:image, type="image" olan elementleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":image")
```

3.1.8.8 :reset

:reset, type="reset" olan elementleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":reset")
```

3.1.8.9 :button

:button, type="button" olan tüm elementleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":button")
```

3.1.8.10 :file

:file, type="file" olan tüm elementleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":file")
```

3.1.8.11 :hidden

:hidden, type="hidden" olan tüm elementleri seçer. Kullanımı aşağıdaki gibidir.

```
$(":hidden")
```

3.1.9 Form Filters

Bu bölümde; bir formun elemanları olan elementler üzerinde duracağız. Bu elementleri çeşitli yönlerden (seçildi, devredışı bırakıldı gibi) incelemeye çalışacağız.

3.1.9.1 :enabled

Herhangi bir form elementinin disabled yapılması durumunda o form elementi deyim yerindeyse; çevrimdışı olur yani dışarıdan herhangi bir müdahale edilemez. :enabled ise bu disabled edilen element dışındaki elementleri seçer.

Daha iyi açıklayıcı olması açısından aşağıdaki örneğimize bakalım.

```
<input type="text" disabled>
<input type="text">
```

Yukarıdaki yapımızda; ilk input alanı disabled edilmiş durumda. Diğer input alanı ise açık, herhangi bir kısıtlaması yok. Dolayısıyla bu ikinci input alanını aşağıdaki yöntem ile kolayca seçebiliriz.

```
$("input:enabled")
```

3.1.9.2 :disabled

:disabled yöntemi de :enabled yönteminin tam tersi bir şekilde; disabled olan elementleri seçecektir.

Örneğimizi ele alalım.

```
<input type="text" disabled>
<input type="text">
```

Yukarıdaki yapımızda; ilk input alanı disabled olduğu için bunu aşağıdaki satır ile kolayca seçebiliriz.

```
$("input:disabled")
```

3.1.9.3 :checked

:checked, işaretlenebilir bir alanın işaretlendiği anda çalışmaya başlar. Örneğin herhangi bir form üzerinde kullanıcının çeşitli bilgilerini "checkbox"lar ile alabilirsiniz. Bu aldığınız değerler ile belirli işlemleri gerçekleştirebilirsiniz.

:checked kullanımı aşağıdaki gibidir.

```
$("input:checked")
```

Henüz farklı bir şey görmedik dolayısıyla örneğimizi oldukça basit tutalım ve aşağıdaki yapımıza bakalım.

```
<input type="checkbox" checked>
<input type="checkbox">
<input type="checkbox" checked>
<input type="checkbox">
<input type="checkbox">
```

Yukarıdaki yapımızda; bir ve üç numaralı checkbox'lar checked edilmiş durumda. İşte bu elementleri kolayca seçebiliriz. Nasıl mı? Aşağıdaki yöntem ile.

```
$("input:checked")
```

3.1.9.4 :selected

:selected, herhangi bir seçim alanında; seçilen bir eleman olması durumunda çalışmaya başlar.

Daha iyi anlayabilmek için aşağıdaki yapımıza bakalım.

```
<select name="city" multiple="multiple">
    <option>İstanbul</option>
    <option selected>Ankara</option>
    <option>İzmir</option>
    <option>Bursa</option>
    <option>Adana</option>
    <option>Antalya</option>
    <option selected>Konya</option>
</select>
```

Yukarıdaki yapımızda; Ankara ve Konya olan seçenekler seçilmiş. Ayrıca select alanımız da multiple özelliğinde. Yani kullanıcının birden fazla alanı seçebilmesi için tasarlanmış. Bu yapımızda selected olan satırları alabiliriz. Bu işlem aşağıdaki satır ile çok basit bir olay.

```
$("select option:checked")
// veya
```

```
$("select>option:checked")
```

Şeklinde kolayca "selected" edilmiş satırlar alınabilir ve her türlü değişikliğe maruz bırakılabilir.

BÖLÜM 4: Attributes

jQuery, HTML DOM (Document Object Model) konusunda oldukça güçlü bir yapıya sahiptir. jQuery'nin sahip olduğu bir çok metot ile istenilen işlem kolayca, deyim yerindeyse "Tereyağından kıl çeker gibi" halledilebilir. Burada önemli olan "jQuery Selector" yapısını iyi bilmeniz gerekli. Bir önceki bölümde detaylı olarak bu konu üzerinde durmuştuk. İncelemediyseniz, bence bakmanızda yarar var.

Pure JavaScript'le çok uğraştığımız bazı konuların size jQuery ile daha basit ve kolay olduğunu anlayacaksınız. Gelin beraber inceleyelim.

4.1 attr

attr yani attribute metodu ile bir elementin sahip olduğu attribute'ler üzerinde çeşitli işlemler gerçekleştirebiliyoruz. Bu başlık altında attr metodunun çeşitli kullanım stilleri üzerinde duracağız. Çübkü attr metodu oldukça ilginç bir metot. Siz de kullandıkça bunu anlayacaksınız

4.1.1 attr (name)

attr(name) metodu, belirtilen elementin, istenilen attribute'unu yani özelliğini öğrenmek için kullanılır. Bir anlamda bir elementin attribute'nde hangi değer varsa bu değeri alabiliriz.

attr(name) kullanımı aşağıdaki gibidir.

```
$("selector").attr(name);
```

name parametresi elementin attribute adıdır.

attr(name) elementi çok yönlü bir element. Aldığı parametrelere göre yapacağı işlemlerde değişebiliyor. attr(name) metodundan detaylı olarak bahsedeceğiz.

Bu başlık altında; bir elementin attribute özelliği nasıl öğrenilir bunun üzerinde duruyoruz. Dolayısıyla örneğimizde bu yönde olacaktır. Aşağıdaki gibi bir input alanımız olsun.

```
<input type="text" name="username" value="Input your username,
please!">
```

Bu input alanımızın; name attribute: "username" ve value attribute: "Input your username, please!" şeklinde. Dilersek bu attribute değerlerine ulaşabiliriz. Aşağıdaki örneğimizi inceleyelim.

```
$(document).ready(function () {
    var name = $(":input").attr("name");
```

```
    var val = $(":input").attr("value");
    alert(name + "\n" + val);
});
```

Yukarıdaki kodlarımızda; input alanına ait olan name ve value attribute'lerini aldık ve bunları alert ( ) ile uyarı şeklinde verdik. attr (name) metodu ile elementleri attribute değerlerini alırken bunları bir değişkene atamalıyız ya da doğrudan yazdırmalıyız. Aksi halde bu değerler uçar gider.

4.1.2 attr (key, value)

attr(key, value) metodu, bir elementin varolan herhangi bir attribute'ndeki değere yeni değer atamaya veyahut varolmayan bir attribute'u tanımlama işleminde de kullanılır. Burada key değeri attribute adı, value değeri ise bu attribute'e atanacak olan değerdir.

attr (key, value) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").attr(key, value);
```

Bir örnek üzerinden durumu açıklamaya çalışalım. Örneğin aşağıdaki gibi bir elementimiz olduğunu varsayalım.

```
<input type="text">
```

Yukarıdaki yapımızda input alanının sadece type="text" olduğu belli. Bunun dışında herhangi bir attribute'si yok. Gelin isterseniz bu elementin attribute'lerini biz tanımlayalım. Aşağıdaki örneğimizi inceleyelim.

```
$(document).ready(function () {
    $(":input").attr("name", "username");
});
```

Yukarıdaki örneğimizde; input elementinin "name" attribute'sine "username" değerini verdik. Peki bu hep böyle midir? Yani tek tek, satır satır attr ( ) metodunu kullanarak, attribute özelliklerini tamamlamak zorunda mıyız? Hayır, aşağıdaki yöntem ile birden fazla istediğimiz attribute'yi attr ( ) metodu ile kazandırabiliriz.

```
$(document).ready(function () {
    $(":input").attr("name", "username").attr("value", "Your
username");
});
```

Yukarıdaki yapımızda; uç uca ekleme yöntemi ile attribute tanımlama işlemini gerçekleştirdik. Burada istediğiniz kadar attr ( ) metodunu uç uca ekleyebilirsiniz. Bu aslında daha da uzun bir yol. Birazdan göreceğimiz bir yöntem bile bu yöntemi kısaltacağız. Bekleme de kalın!

4.1.3 attr (properties)

attr (properties) metodu, bir elemente birden fazla attribute verilmesini sağlar. Daha önce bir elemente birden fazla attribute verilecekse; uç uca atr ( ) ekleme yöntemiyle gerçekleştirilebileceğini söylemiştik. Lakin bu yöntem ile kolayca, birden fazla attribute ekleyebiliriz.
attr (properties) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").attr({
      key1: value1,
      key2: value2,
      key3: value3
});
```

Yukarıdaki yapımızda da görüldüğü üzere; küme parantezleri ({ }) arasında; JSON yapısına benzer bir yapıda bir elemente attribute değerleri kazandırılabilir.
Devamlı kullandığımız yapımızı yeniden ele alalım.

```
<input type="text">
```

Bu elementimize attribute değerleri atayalım. Örneğimiz aşağıdaki gibi olacaktır.

```
$(document).ready(function () {
    $(":input").attr({
      name: "username",
      value: "Your username",
      id: "user"
    });
});
```

Yukarıdaki örneğimizd elementimize; name, value ve id attribute'lerini kazandırdık.

4.1.4 attr(key, fn)

attr ( ) fonksiyonu içerisinde herhangi bir fonksiyon tanımlanabilir ve bu tanımlanan fonksiyon içerisinde çeşitli işlemler gerçekleştirilebilir. Kullanımı aşağıdaki gibidir.

```
$("selector").attr(key, function() {
    // yapilacaklar
});
```

Burada fonksiyon parametre de alabilir. Örneğin elimizde bir img alanı olduğunu varsayarsak;

```
<img src="">
```

Bu img elementinin src attribute'i atanmamış durumda. Bir betik yazarak bunu biz atayabiliriz. Aşağıdaki örneğimizi inceleyelim.

```
$("document").ready(function () {
    var src = "";
    $("img").attr("src", function () {
      src = $("img").attr("src");
      if(src == ""){
        $("img").attr("src", "image.png");
      }
    });
});
```

Yukarıdaki örneğimizi inceleyelim. Öncelikle bir "src" değişkeni tanımladık ve herhangi bir değer ataması yapmadık. Daha sonra "img" elementimiz için attr ( ) metodunu tanımladık ve "function" kısmında; şayet bu "img" elementinin "src" attribute'nin herhangi bir değeri yok ise (if ile kontrol ettik) bu "src" attribute'ine bizim belirlediğimiz bir image path (yol) atadık. Bu ve buna benzer aklınıza gelebilecek olan bir çok betik yazılabilir. Lakin temel haliyle kullanım böyledir.

4.1.5 removeAttr(name)

removeAttr(name) metodu, adından da anlaşılabileceği üzere; atanan veya atanmış olan herhangi bir attribute değerini kaldırır. Kullanımı aşağıdaki gibidir.

```
$("selector").removeAttr("name")
```

Burada name parametresi, seçilen elementin attribute değeridir.

Basit bir örnek ile anlattıklarımızı pekiştirelim. Yine elimizde bir "img" elementi olsun ve "src" attribute'sine bir değer atayalım.

```
<img src="image.png">
```

Bu elementin "src" attribute'i kaldıralım. Kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("img").removeAttr("src");
});
```

4.2 class

class, bildiğiniz gibi CSS'de bir yapıdır. Belirlenen class değerleri elementleri şekillendirmek için kullanılır. Örneğin CSS'de bir Class aşağıdaki gibi tanımlayabilir.

```
.colored{
    color: white;
    background-color: green;
}
```

Gördüğünüz gibi; class isminin başına nokta (.) işareti konur. Zaten jQuery, CSS Selector'lerini kullanır. Şimdi jQuery ile elementlerin class atamalarını veya bu atanan / atanmış class değerlerinin nasıl kaldırılacağını görelim.

Örneklerimizde bu tanımladığımız colored ismindeki class'ı kullanalım.

4.2.1 addClass(class)

addClass(class) metodu, tanımlanmış olan bir CSS class'ının herhangi bir elemente atanması işlemini gerçekleştirir. Kullanımı aşağıdaki gibidir.

```
$("selector").addClass("class")
```

Burada parametre olarak belirtilen class değeri class'ın ismidir.

Çok basit bir örnek yapalım. Örneğin elimizde bir div elementi var.

```
<div>Hello World!</div>
```

Elimizdeki bu div elementinin herhangi bir CSS değeri yok. "4.2 Class" başlığı altındaki "colored" isimli class değerini bu div elementine atayalım. Kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("div").addClass("colored");
});
```

Görüldüğü gibi oldukça basit bir şekilde div elementinin CSS şekillendirmesini addClass (class) metodu ile gerçekleştirdik.

4.2.2 removeClass(class)

removeClass(class) metodu da tahmin edebileceğiniz gibi; atanan veya atanmış olan elementeki class değerini kaldırır. Kullanımı aşağıdaki gibidir.

```
$("selector").removeClass(class)
```

Elimizde aşağıdaki gibi; class değeri atanmış olan div elementinin class değerini kaldıralım.

```
<div class="colored">Hello World!</div>
```

Kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("div").removeClass("colored");
});
```

4.2.3 toggleClass(class)

toggleClass(class) metodu, herhangi bir elemente; şayet bir class atanmışsa kaldırılmasını, atanmamışsa atanmasını sağlar. Bu yapıyı herhangi bir event'a bağlayarak veya otomatikleştirerek bu işlemi göze hoş gelen bir şekle dönüştürebiliriz. toggleClass(class) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").toggleClass(class)
```

Elimizde bir div elementi ve "colored" isimli CSS class'ı olduğunu varsayalım.

```
<div>Emre Can ÖZTAŞ</div>
```

Örneğimizin daha iyi anlaşılabilinmesi için bir de <button>Colored</button> elementi olduğu varsayalım. Kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("button").click(function () {
    $("div").toggleClass("colored");
    });
});
```

Yukarıdaki örneğimizde; bir butona tıklandığı anda; div elementine atanmış bir class varsa bu kaldırılıyor eğer atanmış bir class değeri yok ise class değeri atanıyor.

4.3 HTML

HTML, JavaScript'te bildiğimiz ve kullandığımz .innerHTML fonksiyonuna karşılık geliyor desek yeridir herhalde. Çünkü yapılan iş aynı yani bir anlamda DOM (Document Object Model) işlemini gerçekleştiriyoruz. Şimdi beraber "HTML" metodunu inceleyelim.

4.3.1 html ( )

html ( ) metodu, belirtilen html elementinin value değerini almak için kullanılır. Burada dikkat etmeniz gereken şey; html ( ) metodunun kullanılacağı element bir input alanı olmamalı. Biraz JavaScript bilginiz var ise; ".innerHTML" metodu ile yaptıklarımızın aynısını gerçekleştireceğiz. Yani HTML'nin temel yapılarında html ( ) elementini kullanmalıyız. Yani bu elementler; div, body, title v.s olabilir ama bir input alanı olmamalıdır.

html ( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").html()
```

Yukarıdaki kullanımımızda aldığımız value değerini herhangi bir değişkene bağlamaz isek veya doğrudan yazdırmaz isek; bu aldığımız değer uçup gider. Dolayısıyla bir değişkene bağlayıp kullanmamız yerinde olacaktır.

```
var elementVal = $("selector").html();
```

Basit bir örnek gerçekleştirelim. Aşağıdaki gibi yapılarımız olsun elimizde.

```
<div>Hello World!</div>
<span id="spanID">Hello World!</span>
<label class="labelClass">Hello World!</label>
```

Bu yapımızdaki elementlerin value'lerini alalım. Örneğimiz aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    var divVal = $("div").html();
    var spanVal = $("#spanID").html();
    var labelVal = $(".labelClass").html();

    console.log(divVal + " " + spanVal + " " + labelVal);
});
```

Örneğimizde; html ( ) metodu ile aldığımız value'leri bir değişkene bağladık ve console.log () ile bu değerleri yazdırdık.

4.3.2 html (val)

html (val) metodu ise belirtilen elemente bir değer ataması yapılmak için kullanılır. Burada dikkat edilmesi gereken nokta; html (val) metodu ile herhangi bir value ataması da yapılabilir. HTML element veya elementlerinin ataması da yapılabilir.

html (val) elementinin kullanımı aşağıdaki gibidir.

```
$("selector").html(val)
```

Elimizde div elementi olduğunu farzedelim.

```
<div></div>
```

div elementimizin value değeri yok. Bu div elementine bir value değeri atayalım.

```
$("document").ready(function () {
    $("div").html("Hello World!");
});
```

Bu div elementine HTML elementleri de atayabiliriz. O halde atayalım.

```
$("div").html("<h3>Hello World!</h3>");
```

4.4 Text

Text, HTML'den farklı olarak sadece "value" atanmasını sağlar. Yani HTML'de olduğu gibi herhangi bir HTML elementi ataması yapılamaz. HTML için ne konuşmuş isek aynı şeyler Text için de geçerlidir.

4.4.1 text ( )

text ( ) metodu, belirtilen herhangi bir elementin value değerini almak için kullanılır. Burada dikkat edilmesi gereken nokta; text ( ) metodu sadece .innerHTML metodunun geçerli olduğu durumlarda kullanılabilir. Yani herhangi bir input elementinin value değerini alamaz.
text ( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").text()
```

Yukarıdaki kullanımı bir değişkene bağlamamız daha doğru olacaktır.

```
var val = $("selector").text();
```

Elimizde bir label elementi olduğunu varsayarsak;

```
<label>Hello World!</label>
```

Bu label elementinin value değerini almak için aşağıdaki kodları kullanabiliriz.

```
$("document").ready(function () {
    var val = $("label").text();
    console.log(val);
});
```

4.4.2 text (val)

text (val) metodu, belirtilen herhangi bir elemente value atamasını gerçekleştirir. Burada dikkat edilmesi gereken nokta ise; value ataması yapılacak olan elementin bir input alan olmamasıdır. Yani JavaScript'teki klasik .innerHTML ataması gibi. text (val) metodunun, html (val) metodundan farkı; text (val) metodunda; belirtilen elemente herhangi bir HTML element ataması yapılamaz. Eğer HTML element ataması yapılacaksa; html (val) metodu kullanılmalıdır.

text (val) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").text(val)
```

Value değeri atanmış olan bir div elementimiz olduğunu varsayarsak;

```
<div>Hello World!</div>
```

Bu div elementinin value değerini kendimiz değiştirebiliriz. Kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("div").text("Merhaba Dünya!");
});
```

4.5 Value

HTML ve Text konularının sahip oldukları metotlar ile "input" alanları dışındaki elementlere value ataması veya value değerlerinin alınması işlemlerini gerçekleştirebiliriz. "input" alanlarına değer ataması yapmak için veya sahip oldukları value değerlerini almak için "Value" konusunun sahip olduğu metotlar kullanılmalıdır. Şimdi beraber bu metotlara değinelim.

4.5.1 val ( )

val ( ) metodu, belirtilen input alanın value'lerini almak için kullanılır. Örneğin kullanıcının doldurduğu input alanlarındaki değerleri almak ve kontrol etmek için bu metodu sıklıkla kullanırız.

val ( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").val( )
```

Aldığımız değerlerin kaybolmaması için alınan değerleri bir değişkene bağlasak fena olmaz.

```
var val = $("selector").val();
```

Basit bir örnek yapalım. Kullanıcın mail adresini oldurduğu alandaki mail adresini alalım ve içerisinde @ geçiyor mu diye kontrol edelim. Örneğimiz çok basit. Normal hayatta mail kontrol işlemleri böyle olmaz. Buna da dikkat edelim.

Öncelikle elimizde bir input alanı ve bir de label alanı olduğunu varsayalım. input alanı kullanıcıdan değer almak için ve label alanı da bu alıp incelediğimiz değerin doğru / yanlış sonuçlarını göstermek için kullanacağız.

```
<input type="mail" name="mail">
<label id="mailControl"></label>
```

Şimdi kodlarımızı yazalım.

```
$("document").ready(function () {
```

```
    var mailAddress = $("input[type='mail'][name='mail']").val();
    if(mailAddress.indexOf('@') < 0){
      $("#mailControl").text("This is not mail!");
    } else {
      $("#mailControl").text("Hey, This is a mail. Thanks!");
    }
});
```

Yukarıdaki örneğimiz; kullanıcının mail adresine bir değer girip button'a tıklaması durumunda daha güzel bir işleyiş içerisinde olacak. Yani HTML kodlarımıza bir button ekleyelim ve kodlarımızı aşağıdaki gibi yeniden düzenleyelim.

```
<input type="mail" name="mail">
<label id="mailControl"></label>
<button>Ok</button>
```

Kodlarımızı yeniden düzenleyelim.

```
$("document").ready(function () {
    $("button").click(function () {
        var mailAddress = $("input[type='mail'][name='mail']").val();
        if(mailAddress.indexOf('@') < 0){
          $("#mailControl").text("This is not mail!");
        } else {
          $("#mailControl").text("Hey, This is a mail. Thanks!");
        }
    });
});
```

Web sayfalarında genel işleyiş böyledir. Lakin daha öncede dediğim gibi mail kontrol işlemleri böyle gerçekleştirilmez. Temel yapı veya kullanım bu şekildedir.

4.5.2 val (val)

val (val) metodu, belirtilen bir input alanına value atamasını gerçekleştirir. Burada dikkat edilmesi gereken; val (val) metodu kullanılırken; kullanılacak elementin bir input alan olmasıdır.

val (val) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").val(val);
```

Basit bir örnekle bu bölümü bitirelim. Elimizde iki tane "button" ve bir de input alanı olduğunu varsayalım. Yani;

```
<button>One</button>
<button>Two</button>
<input type="text">
```

Şimdi örneğimizin senaryosu şu şekilde olacak; kullanıcı birinci butona bastığı zaman; input alanına "One" yazılacak ve ikinci button'a tıklaması durumunda input alanına "Two" yazılacak. Burada yazılacak olan değerler de button'ların value'leri olacak. Örnek kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    // button 1
    $("button:first").click(function () {
      var btn1 = $("button:first").text();
      $("input[type='text']").val(btn1);
    });
    // button 2
    $("button:last").click(function () {
      var btn2 = $("button:last").text();
      $("input[type='text']").val(btn2);
    });
});
```

Görüldüğü gibi input alanına atamalarımızı gerçekleştirdik.

BÖLÜM 5: Traversing

Traversing, JavaScript ortamında adlandırdığımız DOM (Document Object Model) olaylarını kolaylaştırmak için geliştirilmiş metotlar bütünü desek yanlış olmaz herhalde. Çeşitli bakımlardan bir element DOM edilebilinir. jQuery sahip olduğu zengin içerik ile bize DOM olaylarını kolay bir şekilde sunmaktadır.

Hazırsanız, o halde Traversing veyahut "Traversal Metot"lara birlikte bakalım.

5.1 Filter

Bir element çeşitli yönlen bakımından filtrelenebilir ve bu filtrelenmiş yapı üzerinde belirli işlemler uygulanabilir.

5.1.1 eq (index)

eq yani equal ile herhangi bir elementin belirli bir indis değerine göre filtreleme ve üzerinde çeşitli işlemler gerçekleştirilebilir. Burada konuşacağımız eq (index) metodu aslında biraz da selector konusunda bahsetmiş olduğumuz :eq (index) konusuna benzemektedir. Lakin burada biraz farklılıklar var. Bu farklılıklara beraber değineceğiz, inşAllah!

eq (index) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").eq(index)
```

Yukarıdaki örnek kullanımda da gördüğünüz gibi eq (index) metodu selector parantezleri içerisinde değil de deyim yerindeyse selector'un ucuna eklenen bir metot. Hemen basit bir örnek ile durumu kavramaya çalışalım.

Elimizde bir div elementlerinden oluşan bir demet olsun.

```
<div>div 1</div>
<div>div 2</div>
<div>div 3</div>
<div>div 4</div>
<div>div 5</div>
```

Bu div demetinde istediğimiz herhangi bir div elementini (örneğin; 2) seçelim. Kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("div").eq(2);
});
```

Yukarıdaki örneğimizde; div element demetinden 2 numaralı elementi seçtik. Tamam. Buraya kadar herhangi bir sıkıntı yok. Lakin bu metodumuzun kullanımı havada kaldı.

Çünkü herhangi bir şey yapmadık. O halde bu filtrelediğimiz elementin CSS değerleriyle oynayalım.

```
$("document").ready(function () {
    $("div").eq(2).css("background-color", "yellow");
});
```

Yukarıdaki yapımızda da görüldüğü gibi metotlarımızı uç uca ekleyerek yazdık. Örneğin; bu filtrelediğimiz elementinde value değerini de alabiliriz. Mesela bunu da kodlarımıza ekleyelim.

```
$("document").ready(function () {
    $("div").eq(2).css("background-color", "yellow");
    var val = $("div").eq(2).text();
    console.log(val);
});
```

Kodlarımız ekledik. eq (index) kullanılırken; indis değeri 0'dan başlar.

eq (index) metodunu aşağıdaki gibi selector parantezleri içerisinde de kullanabilirdik.

```
$("div:eq(2)")
```

5.1.2 hasClass (class)

hasClass (class) metodu, herhangi bir elementin belirtilen bir CSS class'ı alıp almadığını kontrol eder. Şayet bu element belirtilen class'ı almışsa doğal olarak true değerini döndürecek ve bunun üzerinden bu elemente çeşitli işlemler gerçekleştirilebilecektir. Burada dikkat edilmesi gereken nokta; hasClass(class) metodu geriye bir değer döndürür. Bu değer true veya false olacaktır. Bu geri dönen değerlere göre işlemlerimizi gerçekleştirmeliyiz. Yani işleyişi biraz farklı

hasClass (class) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").hasClass (class)
```

Yukarıdaki örnek kullanımda; class parametresi CSS class'ını belirtmektedir. Elimizde aşağıdaki gibi bir class yapısını olduğunu varsayalım.

```
.colored {
    color:white;
    background-color: green;
}
```

Ayrıca elimizde bir div elementi olduğunu farz edelim.

```
<div class="colored">div 3</div>
```

Yukarıdaki yapımızda da görüldüğü üzere div elementimize class değeri olarak colored atanmış.

Şimdi kodlarımızı yazalım.

```
$("document").ready(function () {
    if($("div").hasClass("colored")){
      console.log("Yes, there is a class here!");
    } else {
      console.log("I'm sorry but i didn't found any class here!")
    }
});
```

Yukarıdaki yapımızda; hasClass (class) ile div elementinin bir class değeri olup olmadığını if içerisinde kontrol ettik. Şayet varsa bir mesaj yoksa yine bir mesaj verecektir.

5.1.3 filter (expr)

filter (expr) metodu herhangi bir bakımdan, belirtilen bir element üzerinde inceleme işleminde bulunabilir. Bu inceleme işlemi bir belirtime göre olabileceği gibi birden fazla belirtime göre de olabilir. Bu belirtimler; herhangi bir jQuery filtreleme yöntemi olabilir. Örneğin; id, class, :first ve aklınıza gelebilecek herhangi bir yol.

filter (expr) kullanımı aşağıdaki gibidir.

```
$("selector").filter (expr)
```

Basit bir örnekle anlattıklarımızı pekiştirelim. Elimizde aşağıdaki gibi bir div element demeti olduğunu farz edelim.

```
<div>div 1</div>
<div>div 2</div>
<div class="colored">div 3</div>
<div>div 4</div>
<div>div 5</div>
```

Bu div elementi demeti içerisinde; sadece bir div elementinin class attribute'i var. Şimdi bunu filter (expr) metoduyla filtreleyebiliriz.

```
$("div").filter(".colored")
```

Filtrelediğimiz bu sonucun text değerini alalım. Yani örneğimiz aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    var val = $("div").filter(".colored").text();
    console.log(val);
```

```
});
```

filter (expr) metodu birden fazla filtreleme belirtimi alabilir. Kullanımı aşağıdaki gibidir.

```
$("selector").filter("exp1 ,exp2, exp3...)
```

5.1.4 filter (fn)

filter (fn) metodunda herhangi bir fonksiyon tanımlanabilir ve bu tanımlanan fonksiyon üzerinden belirli işlemler gerçekleştirilebilir.

```
$("selector").filter(function () {
    // yapilacaklar
})
```

Bu fonksiyona dilenirse; parametre ataması da yapılabilinir.

Çok basit bir örnek yapalım ve örnek kodlarımız aşağıdaki gibi olsun.

```
$("document").ready(function () {
  $("div").filter(function () {
    var val = $("div").eq(2).text();
    console.log(val);
  });
});
```

5.1.5 is (expr)

is (expr) metodu, belirtilen herhangi bir kriterin, belirtilen elementte olup olmadığını kontrol eder. Şayet belirtilen kriter elementte var ise true yoksa false değerini döndürür.

is (expr) kullanımı aşağıdaki gibidir.

```
$("selector").is(expr)
```

is (expr) metodunda, expr parametresine herhangi bir parametre verilebilir. Örneğin; id, class, :first, :last veya aklına gelebilecek diğer jQuery filtreleme yolları.

Elimizde aşağıdaki gibi bir yapımız olduğunu farz edelim.

```
<div>div 1</div>
<div>div 2</div>
<div class="colored">div 3</div>
<div>div 4</div>
<div>div 5</div>
```

Bu yapımızda; üçüncü sıradaki div elementinin class attribute'i bulunmakta. Şimdi bunu is (expr) metoduyla kontrol edelim. Örnek kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    if ($("div").is(".colored")){
        $("div[class='colored']").removeClass("colored");
    }
});
```

Yukarıdaki kodlarımızda temel olarak yaptığımız şu; div elementlerinden herhangi birisi "colored" class'ısını içeriyorsa bu colored class'ını içeren class'ın class attribute'ini yok et dedik.

5.1.6 not (expr)

not (expr) meotodu adından da anlaşılabileceği üzere; kendisine parametre olarak verilen belirtimin olmadığı ya da içermeyen element veya elementleri seçer. Burada selector'da belirtilen element önemlidir.

not (expr) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").not(expr);
```

expr olarak verilen parametre herhangi bir belirtim veya birden fazla belirtim olabilir. Bir örnekle konuştuklarımızı pekiştirelim. Aşağıdaki gibi CSS class yapılarımızın olduğunu farz edelim.

```
.colored {
    color: white;
    background-color: green;
}

.bordered {
    border: 1px dotted;
    border-color: black;
}
```

Yukarıdaki class yapımızı uygulayan span yapılarımız da olsun.

```
<span class="colored bordered">span1</span>
<span>span2</span>
<span class="bordered">span3</span>
```

Şimdi kodlarımızı yazabiliriz. Örnek kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("span").not(".colored").addClass("colored");
});
```

Yukarıdaki kodlarımızda neler yaptık? Kodlarımızda; colored ve class'ını uygulamayan span'lara colored class'ının atamasını yaptık.

not(expr) metodu çoklu belirtim alabilir. Yukarıdaki örneğimiz için konuşursak; $("span").not(".colored, .bordered").addClass("colored"); şeklinde de yazabilirdik. Burada belirtim sayısı tabiki arttırılabilir. Seçim size kalmış.

5.1.7 slice(start, [end])

slice(start, [end]) metodu, belirtilen elementten belirtilen aralık kadarını filtrelenmesini sağlar. Örneğin elimizde 5 tane input değeri varsa; şayet biz de ilk elementi almak istiyorsak; o halde (0, 1) aralığını belirtmeliyiz. Burada aralık arttırılıp azaltılarak alınacak element sayısı belirlenebilir.

slice(start, [end]) metodunda indis değeri 0'dan başlar.

slice(start, [end]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").slice(start, [end])
```

Bu verdiğimiz bilgiler ışığında basit bir örnek yapalım. Örneğin elimizde aşağıdaki gibi bir yapı olduğunu farz edelim.

```
<input type="text">
<input type="text">
<input type="text">
<input type="text">
<input type="text">
```

Yukarıdaki input demetinden 2 ve 4 numaralı input alanlarını alıp değer atamasını gerçekleştirelim. Yazacağımız kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("input").slice(2, 4).val("Hello!");
});
```

Yukarıdaki örneğimizde; (2, 4) arasındaki input elementlerine "Hello!" kelimesini atadık.

5.2 Finding

Bu başlık altında filtrelemeye yakın olarak çeşitli elemanların bulunması eklenmesi gibi değişik konular içeriyor. Biz de bu konulara değinmeye çalışacağız.

5.2.1 add(expr)

add(expr) metodu, herhangi bir elemente, varolan başka bir elementi ekleme işlemini gerçekleştirir. Eklenecek element yok ise add (expr) metodunun parantezleri arasında yeni bir element tanımlaması gerçekleştirilebilir.

add(expr) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").add(expr)
```

Her zaman olduğu gibi anlattıklarımızı bir örnek ile taçlandıralım. Aşağıdaki gibi yapılarımız olduğunu varsayalım.

```
<div>Hello</div>
<p>Merhaba</p>
```

div elementine p elementini ekleyelim. Daha sonra da her ikisinin text değerlerini değiştirelim. Örnek kodlarımız aşağıdaki gibi olacaktır.

```
$("document").ready(function () {
    $("div").add("p").text("Bonjour");
});
```

add(expr) metodun da varolmayan elementlerin de ataması yapılabilir dedik. Bu işlemi de aşağıdaki gibi gerçekleştirebiliriz.

```
$("div").add("<span>Merhaba</span>").text("Bonjour");
```

5.2.2 children([expr])

children([expr]) metodu, belirtilen elementin tüm children (çocuklar) elementlerini seçer. Yani bir elementin ne kadar alt elementi varsa bunların hepsi bu metot yardımıyla seçilebilir. children([expr]) metodu parametre belirtilerek kullabileceği gibi parametre belirtilmeden de kullanılabilir.

children([expr]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").children([expr])
```

Yukarıdaki metot herhangi bir değişkene bağlı olarak kullanılabileceği gibi uç uca ekleme yoluyla diğer metotlar ile de çalışabilir.

Örneğin herhangi bir elementin altındaki diğer elementlerin değerlerini almak için aşağıdaki şekilde kullanılabilir.

```
var val = $("selector")children().text();
```

Herneyse basit bir örnek ile anlattıklarımızı taçlandıralım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style type="text/css">
  .colored {
    color: white;
    background-color: green;
  }
</style>
<div>
  <p>Merhaba
    <span>Hello
      <i>Bonjour</i>
    </span>
  </p>
</div>

<script type="text/javascript">
    $("document").ready(function () {
      $("div").children().addClass("colored");
    });
</script>
```

Yukarıdaki örneğimizde de görüldüğü gibi; div elementinin altındaki tüm children elementleri seçtik ve bu seçtiğimiz children elementlere "colored" class'ını atadık.

5.2.3 find (expr)

find (expr) metodu, belirtilen elementin altında belirtilen elementi bulur. Hadd-i zatında find, bulmak anlamına gelmektedir. Bu metot yardımıyla descendant (torun) elementleri bulmak için oldukça elverişli bir yapıdır.

find (expr) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").find(expr)
```

Aşağıdaki yapımızı inceleyelim.

```
<style type="text/css">
    .colored {
        color: white;
        background-color: green;
    }
</style>

<div>
    <p>
        Where is my mind?
    </p>
</div>
```

```
<script type="text/javascript">
    $("document").ready(function () {
      $("div").find("p").addClass("colored");
    });
</script>
```

Yukarıdaki yapımızda bir CSS class'ı tanımladık. Daha sonra da div ve içinde bir de p elementi tanımladık. Kodlarımızda ise yaptığımız 'şu; div elementinin altında varsa p elementini bul ve bu p elementine colored class'ını ata.

find (expr) metoduna parametre olarak istediğiniz herhangi bir selector'u verebilirsiniz. Hepsini desteklemektedir.

5.2.4 next(expr)

next(expr) metodu, belirtilen elementin sibling (kardeş) olan elementi seçer. Burada dikkat edilmesi gereken nokta; next(expr) metodu sadece bir kardeş elementi verir. Tüm kardeş elementleri vermez. Buna dikkat edelim.

next(expr) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").next(expr)
```

next(expr) metodunda belli parametreler verilebilir de verilmeyebilir de. Verilmezse; herhangi bir kriter olmaksızın, belirtilen elementin kendisine en yakın olan kardeş elementi verir. Şayet parametre olarak belirli kriterler verilirse; seçme işlemini bu kriterlere göre yapar.

Basit bir örnekle konumuza devam edelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<style type="text/css">
    .colored {
        color: white;
        background-color: green;
    }
</style>

<div>
    <p>
        Where is my mind?
    </p>
    <p>
        It's lost!
    </p>
    <p>
        :'(
    </p>
</div>
```

```
<script type="text/javascript">
    $("document").ready(function () {
      $("div").find("p").next().addClass("colored");
    });
</script>
```

Yukarıdaki örneğimizde neler yaptık? Örneğimizde; div elementi altındaki p elementini find (expr) metodu ile bulduk. Daha sonra bulduğumuz bu elementin kardeşini next (expr) metodu yardımıyla bulduk ve tanımlamış olduğumuz CSS class'ını bu kardeş elemente atadık.

Örneğin HTML yapımız aşağıdaki gibi olsaydı:

```
<div>
    <p>
        Where is my mind?
    </p>
    <p id="n">
        It's lost!
    </p>
</div>
```

yazacağımız kodlarımız da aşağıdaki gibi olacaktı.

```
$("document").ready(function () {
        $("div").find("p").next("#n").addClass("colored");
});
```

5.2.5 nextAll(expr)

nextAll(expr) metodu, next (expr) metodunun aksine tüm sibling (kardeş) elementleri seçer.

Aşağıdaki örneğimizi inceleyelin.

```
<style type="text/css">
    .colored {
        color: white;
        background-color: green;
    }
</style>

<div>
    <p>
        Where is my mind?
    </p>
    <p>
        It's lost!
```

```
    </p>
    <p>
        :'(
    </p>
</div>

<script type="text/javascript">
    $("document").ready(function () {
      $("div>p").nextAll().addClass("colored");
    });
</script>
```

Yukarıdaki örneğimizde; div elementinin altındaki, ilk p elementinden sonraki tüm p elementlerini seçtik ve class değerlerine colored isimli CSS class'ını atadık. Burada dikkat edilmesi gereken nokta; ilk p elementi seçilmeyecek olup bu ilk p elementinden sonraki kardeş elementler seçilecektir.

nextAll (expr) metoduna; kardeş elementleri seçmek için istediğiniz herhangi bir belirtimi de verebilirsiniz. Böylelikle seçilecek kardeş element sayısını azaltabilirsiniz.

5.2.6 parent (*expr)

parent (*expr) metodu, belirtilen child (çocuk) elementin hemen üzerindeki parent (ata) elementi verir.

parent (*expr) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").parent (*expr)
```

parent (*expr) metodunda belirtimler kullanılabilir, bu belirtimlere sahip olan parent element seçilir.

Aşağıdaki örneğimizi inceleyelim.

```
<div>
    <p>Hello World!</p>
</div>

<script type="text/javascript">
    $(document).ready(function () {
        $("p").parent().css("color", "blue");
    });
</script>
```

Yukarıdaki örneğimizde; p elementinin parent elementi div. Dolayısıyla jQuery satırımızda bu p elementinin parent parent() metodu ile seçtik. Daha önce de belirttiğimiz gibi çeşitli belirtimler kullanılarak parent element seçilebilir.

5.2.7 parents (*expr)

parent (*expr) metodu sadece bir parent'i seçer lakin parents (*expr) metodu ne kadar parent element varsa hepsini seçer. Buna çok dikkat edelim.

parents (*expr) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").parents (*expr)
```

parents (*expr) metodunda birden fazla belirtim kullanılabilir.

Aşağıdaki örneğimizi inceleyelim.

```
<div>
    <span>
        <p>Hello World!</p>
    </span>
</div>

<script type="text/javascript">
    $(document).ready(function () {
        $("p").parents().css("color", "blue");
    });
</script>
```

Yukarıdaki örneğimizde; p elementinin iki tane parent elementi var. Dolayısıyla bu parent elementleri parents() metodu ile kolayca seçtik.

5.2.8 prev([expr])

prev([expr]) metodu, belirtilen elementten önceki sibling (kardeş) elementi seçer. Burada dikkat edilmesi gereken nokta; prev([expr]) metodu ile sadece bir kardeş element seçilir.
prev([expr]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").prev([expr])
```

Şimdi aşağıdaki örneğimizi inceleyelim.

```
<div>Sawyer</div>
<div>Jack</div>
<div>Kate</div>

<script type="text/javascript">
    $(document).ready(function () {
        $("div:last").prev().css("font-weight", "bold");
    });
</script>
```

Yukarıdaki örneğimizde; birbirleriyle kardeş olan üç tane div elementi var. Biz ortadaki div elementini seçmek için jQuery kodlarımızı yazdık.

prev([expr]) metoduna belirtimler de gönderilebilir. Böylelikle istenilen element kolaylıklar seçilebilir.

5.2.9 prevAll([expr])

prevAll([expr]) metodu, belirtilen elementten önceki tüm kardeş elementleri seçer. prev([expr]) metodu ise sadece bir tane element seçer. İki metot arasındaki fark budur.

prevAll([expr]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").prevAll([expr]);
```

Aşağıdaki örneğimizi inceleyelim.

```
<div>Sawyer</div>
<div>Jack</div>
<div>Kate</div>

<script type="text/javascript">
    $(document).ready(function () {
        $("div:last").prevAll().css("font-weight", "bold");
    });
</script>
```

Yukarıdaki örneğimizde; son div elementinin üzerindeki bütün div elementleri seçilecektir.

5.2.10 siblings([expr])

siblings([expr]) metodu, belirtilen elementin dışındaki, o elementin tüm kardeş elementlerini seçer. Yani burada elementlerin sırası ve yeri önemli değildir.

siblings([expr]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").siblings([expr])
```

Aşağıdaki örneğimizi inceleyelim.

```
<div>Sawyer</div>
<div id="val">Jack</div>
<div>Kate</div>

<script type="text/javascript">
    $(document).ready(function () {
        $("#val").siblings().css("font-weight", "bold");
    });
```

```
</script>
```

Yukarıdaki örneğimizde; ortadaki div elementine bir id değeri verdir. Bu üç div elementi de birbirleriyle gördüğünüz gibi kardeş. jQuery kodlarımızda ise; ortadaki div elementi (id="val") dışındaki kardeş elementleri (div) seçtik.

siblings([expr]) metoduna belirtimler gönderilerek farklı seçme işlemleri yapılabilir.

BÖLÜM 6: Manipulation

Bu bölümde temel olarak; çeşitli metotlar yardımıyla belirlediğimiz elementlere, örneğin; önlerine veya arkalarına yeni elementler ekleyeceğiz, başka elementlerle değiştireceğiz ve buna benzer pek çok manipülasyon tekniklerini kullanağız.

O halde, hazırsanız başlayalım.

6.1 Inserting Inside

Inserting Inside ile herhangi bir elementin sınırları içerisine çeşitli elementler veya text'ler ekleyebiliriz. Burada elementin sınırları dediğimiz yer, örneğin; elementinin içerisi olabilir.

6.1.1 append(content)

append(content) ile belirlenen herhangi bir elementin sonuna, yeni bir element veya text değeri ekleyebilmekteyiz.

append(content) kullanımı aşağıdaki gibidir.

```
$("selector").append(content)
```

Burada content değeri daha öncede belirttiğimiz gibi; text veya yeni bir element olabilir. Lakin sayfa üzerinde daha önce oluşturulmuş olan herhangi bir elementi ekleyemez. Buna çok dikkat edelim.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<p>Hello!</p>

<script>
    $(document).ready(function (){
            $("p").append('<b> Merhaba!</b>');
    });
</script>
```

Yukarıdaki örneğimizde; p elementinin içerisinde bulunan text'in sonuna b elementi içerisindeki text'i ekledik. Burada b elementini kullanmak zorunda değiliz. Doğrudan text'imizi de yazabiliriz.
append(content) metodu, appendChild metodu ile benzer bir yapıdadır.

6.1.2 appendTo(content)

appendTo(content) metodu, append(content) metodunun tam tersi; belirtilen herhangi bir text veya elementi, belirtilen elementin başına ekler.

appendTo(content) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").appendTo(content)
```

appendTo(content) metodunu daha iyi anlamak için basit bir örnek yapalım örneğimiz aşağıdaki gibi olacaktır.

```
<p>Hello!</p>
<div>Merhaba</div>

<script>
    $(document).ready(function (){
            $("p").appendTo('div');
    });
</script>
```

Yukarıdaki örneğimizde; p elementinin başına div elementini ekledik. Burada örneğimiz çok basit olduğu için doğrudan element isimlerini kullanarak işlemlerimizi gerkeçleştirdik. Lakin siz elementlerin daha belirleyici özelliklerini kullanmaya özen göstermelisiniz. Aksi halde tüm p ve div elementlerini birleştirebilirsiniz.

appentTo(content) metodu ile sayfada daha önce oluşturulmuş olan elementler birleştirilebilir. Lakin append(content) metodunda böyle bir şey söz konusu değildir.

6.1.3 prepend(content)

prepend(content) metodu, append(content) metoduna benzer bir yapıdadır. Lakin prepend(content) metodundan farklı olarak bir elementin başına istenilen text veya elementi ekleyebilir. Eğer bir elementin başına eklemek istediğiniz değerler var ise bu iş için prepend(content) metodu biçilmiş kaftandır.

prepend(content) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").prepend(content)
```

prepend(content) metodunu daha iyi anlayabilmek için aşağıdaki örneğimizi inceleyelim.

```
<span>This is first!</span>

<script>
    $(document).ready(function (){
        $("span").prepend("<div>This is second!</div>");
    });
</script>
```

Yukarıdaki örneğimizde; span elementinin başına This is second! elemtini ve bu element içerisindeki ifadeyi yerleştirdik.

6.1.4 prependTo(content)

prependTo(content) metodu, appendTo(content) metoduna benzer bir yapıdadır. Belirtilen herhangi bir element, text veya sayfa üzerindeki daha önce tanımlanmış olan herhangi bir elementi, belirtilen bir elementin sonuna ekleyebilir.

prependTo(content) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").prependTo(content)
```

Aşağıdaki örneğimizi inceleyelim.

```
<span>This is first!</span>
<div>This is second!</div>

<script>
    $(document).ready(function (){
        $("span").prependTo("div");
    });
</script>
```

Yukarıdaki örneğimizde; span elementinin sonuna; div elementini ekledik.

6.2 Inserting Outside

Inserting Outside ile herhangi bir elementin sınırları dışında, yani o elementin etiketleri dışına istenilen değerler eklenebilmektedir. Bu eklenecek olan değerler; element veya text olabilir.

6.2.1 after (content)

after (content) metodu ile istenilen herhangi bir elementen sonra content yani değerler eklenebilmektedir.

after (content) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").after(content)
```

Bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>This is first!</span>

<script>
    $(document).ready(function (){
        $("span").after("<div>This is second!</div>");
    });
</script>
```

Yukarıdaki örneğimizde; span elementinden hemen sonra; This is second! elementini ve içerisindeki text değerini ekledik. Burada sadece text değeri de kullanabilirdik. Yani seçim bize kalmış durumda.

6.2.2 before (content)

before (content) metodu, after (content) metodundan farklı olarak; belirtilen herhangi bir elementen önce text veya herhangi bir element eklenebilir.

before (content) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").before (content)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>This is first!</span>

<script>
    $(document).ready(function (){
        $("span").after("<div>This is second!</div>");
    });
</script>
```

Yukarıdaki örneğimizde; span elementinden önce This is second! ifadesini ekledik.

6.2.3 insertAfter(content)

insertAfter(content) metodu, belirtilen herhangi bir elementten önce, sayfa üzerinde tanımlanmış olan bir başka elementi ekler.

insertAfter(content) elementinin kullanımı aşağıdaki gibidir.

```
$("selector").insertAfter(content)
```

Basit bir örnek gerçekleştirelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>This is first!</span>
<div>This is second!</div>

<script>
    $(document).ready(function (){
        $("span").insertAfter("div");
    });
</script>
```

Yukarıdaki örneğimizde; span elementinin öncesine, daha önce tanımlamış olduğumuz div elementini ekledik.

6.2.4 insertBefore(content)

insertBefore(content) metodu, insertAfter(content) metodundan farklı olarak; belirtilen elementten sonra, sayfa üzerinde daha önce tanımlanmış olan bir başka elementi ekler. insertBefore(content) kullanımı aşağıdaki gibidir.

```
$("selector").insertBefore(content)
```

Konuyu daha iyi anlamak için basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>This is first!</span>
<div>This is second!</div>

<script>
    $(document).ready(function (){
        $("span").insertAfter('div');
    });
</script>
```

Yukarıdaki örneğimizde; span elementinden hemen sonra div elementini yerleştirdik. Sayfa dizaynımıza baktığımızda; herhangi bir değişiklik yapmadığımızı görüyoruz.

6.3 Inserting Around

Inserting Around ile herhangi bir elementi, başka bir element ile kaplayabiliriz. Yani bir elementi, belirlediğimiz başka bir elementin içerisine alabiliriz.

6.3.1 wrap(elem)

wrap( ) metodu ile istenilen herhangi bir element, belirlenen başka bir element içerisine alınabilir.

wrap( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").wrap(elem);
```

Anlattıklarımızın ışında basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style type="text/css">
    .extra-wrapper {
        background-color: yellow;
    }
</style>

<span>Hello World!</span>

<script>
    $(document).ready(function (){
```

```
        $("span").wrap('<div class="extra-wrapper"></div>');
    });
</script>
```

Yukarıdaki örneğimizde; span elementini, div 'elementi içerisine aldık. Lakin burada şöyle bir durum var: div elementinin class'ı extra-wrapper ve bu class için yazılan bir css kodu var. wrap ( ) fonksiyonunu kullandıktan sonra; artık span elementi de div elementi içerisinde olacağı için, span elementi de css kodundan etkilecektir.

6.3.2 wrapAll(elem)

wrapAll(elem) metodu ile istenilen elementlerin hepsi, başka bir element içerisine alınabilirler.

Peki burada aklınıza; "wrap ( ) ile wrapAll ( ) metotları arasında ne farklar var?" diyebilirsiniz. Haklısınız. wrap ( ) metodu her elementi ayrı ayrı olarak alır ve belirtilen elementin içerisine yerleştirir. Bunu bir queue (kuyruk) gibi düşünebilirsiniz. wrapAll ( ) metodu ile belirtilen tüm elementi alır ve belirtilen elementin içerisine yerleştirir. Yani yanyana ve topluca.

wrapAll( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").wrapAll(elem)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style type="text/css">
    .extra-wrapper {
        background-color: yellow;
    }
</style>

<span>Hello!</span>
<span>Hola!</span>
<span>Bonjour!</span>
<span>Moikka!</span>

<script>
    $(document).ready(function (){
        $("span").wrapAll('<div class="extra-wrapper"></div>');
    });
</script>
```

Yukarıdaki örneğimizde; tüm span elementlerini div elementi içerisine aldık.

6.3.3 wrapInner(elem)

wrapInner(elem) metodu diğer wrap (elem) metotlarından farklı olarak doğrudan elementlerin içlerindeki içerikleri hedef alır. Yani bir div elementi içerisinde bir element veya text varsa doğrudan bu element veya text'i etkiler

wrapInner(elem) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").wrapInner(elem)
```

Basit bir örnek yapalım ki bu metodu daha yakından tanıyabilelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>Hello!</span>
<span>Hola!</span>
<span>Bonjour!</span>
<span>Moikka!</span>

<script>
    $(document).ready(function (){
        $("span").wrapInner('<b></b>');
    });
</script>
```

Yukarıdaki örneğimizde; span elementlerinin içlerindeki text değerlerini b elementi ile bold yani kalın yaptık.

6.4 Replacing

Replacing yardımıyla, varolan elementleri yeni tanımlayacağımız elementler ile değiştirebiliriz veya varolan diğer elementler ile değiştirme işlemlerimizi gerçekleştirebilmekteyiz.

6.4.1 replaceWith(content)

replaceWith(content) metodu, belirlenen elementi başka bir elementle değiştirme işlemini gerçekleştirir. Hadd-i zatında replace with kelime anlamı olarak; ile değiştir anlamına gelmektedir. İşte biz o "ile" kısmını belirleyeceğiz.

replaceWith(content) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").replaceWith(content)
```

Burada content değeri element veya text değeri olabilmektedir. Burada dikkat etmeniz gereken nokta; sayfa üzerinde olan bir elementi değiştirmek için yeni değer olarak kullanamazsınız. Yani yeni element veya text değerleri oluşturmanız gerekmektedir.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>Hello!</span>
<span>Hola!</span>
<span>Bonjour!</span>
<span>Moikka!</span>

<script>
    $(document).ready(function (){
        $("span").replaceWith('<b>Merhaba!</b>');
    });
</script>
```

Yukarıdaki örneğimizde; span elementlerini, Merhaba! ifadesi ile değiştirdik.

6.4.2 replaceAll(selector)

replaceAll(selector) metodu için aslında; replaceWith (content) fonksiyonun farklı bir versiyonu diyebiliriz. replaceAll(selector) metodunda değiştirilecek olan elementin selector'unun yazılması gerekir. Örneğin; replaceWith (content) metodunda ise yeni gelecek olan içeriği yazmalıyız.

replaceAll(selector) metodunun kullanımı aşağıdaki gibidir.

```
$("content").replaceAll(selector)
```

Her zaman olduğu gibi basit bir örnek yaparak durumu kavramaya çalışalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>Hello!</span>
<span>Hola!</span>
<span>Bonjour!</span>
<span>Moikka!</span>

<script>
    $(document).ready(function (){
        $("<b>Merhaba</b>").replaceAll('span');
    });
</script>
```

Yukarıdaki örneğimizde; tüm span elementlerini, Merhaba ifadesi ile değiştirdik.

6.5 Removing

Removing ile sayfa üzerinde bulunan ve artık bulunmasını istemediğimiz çeşitli elementleri kaldırabilirz veyahut bir runtime (çalışma zamanı) içerisinde silebiliriz. Yani sayfa yenilendiği zaman tekrar o elementler geri gelecektir.

6.5.1 empty( )

empty( ) metodu, belirtilen elementi bir anlamda boşaltır. Yani içerisinde olan herneyse hepsini siler. Burada dikkat edilmesi gereken nokta; empty( ) metodu ile elementin içerisinde var olan: text veyahut elementler de silinir. Yani belirtilen elementin altında bulunan her şey silinecektir.

empty( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").empty( )
```

Çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<div>
    Merhaba
    <span>Hello!</span>
    <span>Hola!</span>
    <span>Bonjour!</span>
    <span>Moikka!</span>
    Nasılsınız?
</div>

<script>
    $(document).ready(function (){
        $("div").empty();
    });
</script>
```

Yukarıdaki örneğimizde; div elementi içerisinde ne varsa herşey silindi. Lakin div elementimiz silinmedi. Bunu anlamak için kodlarımızı aşağıdaki düzenlememiz yeterli.

```
$(document).ready(function (){
    $("div").empty();
    $("div").append("<b>div is here!</b>");
});
```

Yukarıdaki örneğimizde, sadece span elementlerini silmek için selector'umuzu düzenlememiz yeterli.

```
$("div>span").empty();
```

6.5.2 remove ( )

remove ( ) metodu, empty ( ) metodunun aksine varolan elementi de siler. empty ( ) metodu sadece belirtilen elementin altında bulunan text veya elementi siliyor id. Lakin remove ( ) metodu karşısına çıkan ne varsa buna selector ile belirtilen elementte dahil herşeyi siler.

remove ( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").remove ( )
```

Basit bir örnek gerçekleştirelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<div>
    Merhaba
    <span>Hello!</span>
    <span>Hola!</span>
    <span>Bonjour!</span>
    <span>Moikka!</span>
    Nasılsınız?
</div>

<script>
    $(document).ready(function (){
        $("div").remove();
        // $("div").append("<b>div is here!</b>");
    });
</script>
```

Yukarıdaki örneğimizde; div elementi altında ne varsa (div elementi de dahil) herşey silindi. Burada açıklama satırları ile kapatılan alan çalışmayacaktır. İsterseniz deneyebilirsiniz. Çünkü div elementi silinecektir. Yani ortada div elementi kalmayacaktır.

6.6 Copying

Copying metotları ile istediğimiz herhangi bir elementi çoğaltabiliriz. Örneğin elimizde bir div elementi varsa bunun sayını arttırabiliriz.

6.6.1 clone ( )

clone ( ) metodu ile belirtilen element çoğaltılır. Yani bir anlamda koyun dolly gibi aynı özelliklere sahip yeni element sahibi olabilirsiniz.

clone ( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").clone ( )
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<input type="button" value="Dolly :)"/>

<script>
    $(document).ready(function (){
        $("input[type='button']").click(function () {
            $(this).clone().insertAfter(this);
```

```
        });
    });
</script>
```

Yukarıdaki örneğimizde ilginç işler yaptık. Öncelikle yaptığımız; oluşturduğumuz button elementine tıklandığı anda, kendisinin bir kopyasını oluşturduk. Burada this keyword (anahtar kelime)'leri, button'u işaret etmektedir. Yani button'a tıklayınca bu button'u clone'la ve bu button'dan sonrasına kopyayı ekle dedik.

6.6.2 clone (true)

clone (true) metodu ise, clone ( ) metodundan farklı olarak son oluşturulan elemente de ilk kopyalanan elementin özelliklerini verir. Örneğin; bir button'u oluşturduğumuzda, oluşturulan bu button'a tıkladığımızda herhangi bir olay gerçekleştirilmez. Çünkü böyle bir özelliği yoktur. Yeni bir clone oluşturmak için ilk button'a tıklamamız gerekir. clone (true) metoduna gönderdiğimiz true parametresi ile son oluşturulan elemente de ilk elementin özellikleri aktarılır. Yani bunu bir kalıtım olarak düşünebilirsiniz.

clone (true) metodunun kullanımının clone ( ) metodundan herhangi bir farkı yok. Lakin biz yine de gösterelim.

```
$("selector").clone (true)
```

Şimdi örneğimize geçelim.

```
<input type="button" value="Dolly :)"/>

<script>
    $(document).ready(function (){
        $("input[type='button']").click(function () {
            $(this).clone().insertAfter(this);
        });
    });
</script>
```

Yukarıdaki örneğimizde; son oluşturulan button'a tekrar tıkladığımızda bile yeni button oluşturmaya devam edecektir.

BÖLÜM 7: CSS

CSS başlığı altında, sayfa üzerindeki çeşitli elementlerin CSS değerlerinin doğrudan değiştirilmesine değineceğiz. Yani klasik olarak bildiğimiz CSS işlemlerini gerçekleştireceğiz.

7.1 CSS

jQuery ortamında, CSS kullanımı; .css ( ) metodu ile sağlanmaktadır. Şimdi CSS işlemlerine geçelim.

7.1.1 css (name, value)

css (name, value) metodunda, name CSS özelliği ve value değeri ise bu CSS özelliğinin değeridir. Basit olarak bu metot; tek işlemde, bir elementin yalnızca bir CSS özelliğini değiştirebilir.

css (name, value) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").css (name, value)
```

CSS özelliğini daha iyi anlayabilmek için çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>Hello there! What is the weather like?</span>

<script>
    $(document).ready(function (){
        $("span").css("background-color", "yellow");
        });
</script>
```

Yukarıdaki örneğimizde; span elementinin arkaplan rengini "yellow" yani "sarı" yaptık. Şayet bu elementin diğer CSS değerlerini değiştirmek istersek iki farklı yolu seçebiliriz. Bu yollardan birincisi;

```
$("span").css("background-color", "yellow").css("color", "red");
```

şeklinde uç uca ekleyerek CSS kodlarını yazmak, ikinci yol ise;

```
$("span").css("background-color", "yellow")
$("span").css("color", "red");
```

şeklinde alt alta css ( ) metodunu kullanmaktır.

7.1.2 css (properties)

css (name, value) yöntemiyle, bir elementin CSS değerlerini şekillendirmek için oldukça fazla sayıda kod yazıyoruz, bildiğiniz gibi. Lakin css (properties) yöntemiyle daha az sayıda kod yazacağız.

css (properties) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").css (properties)
```

Burada properties kısmına, değiştirmek ya da eklemek istediğimiz CSS değerlerini yazmamız gerekir. Buraya yazım şekli tıpkı bir JSON yapısını andırmaktadır. Yani şöyleki;

```
{
    backgroundColor: "yellow",
    color: "red",
    fontSize: "20px";
}
```

Şeklinde olacaktır. JSON yapısında da değerler key: "value" şeklinde bulunur. Burada da durum aynen böyledir. Bir diğer konuysa; CSS özellikleri yazılırken aradaki - işareti kaldırılır ve ikinci kelimenin baş harfi büyük yazılır. Eğer özellik ismi tek kelime ise herhangi bir şey yazılmaz.

Şimdi anlattıklarımızın ışında basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>Hello there! What is the weather like?</span>

<script>
    $(document).ready(function (){
        $("span").css({
            backgroundColor: "yellow",
            color: "red",
            fontSize: "20px",
            fontWeight: "bold"
        });
    });
</script>
```

Yukarıdaki örneğimizde; span elementinin CSS değerlerini yazdık.

7.1.3 css(name)

css(name) metodu ile herhangi bir elementin, CSS değerleri alınabilir daha doğrusu öğrenilebilir.

css(name) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").css(name)
```

Burada; name CSS özelliğinin ismi olacaktır.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style type="text/css">
    span {
        background-color: green;
    }
</style>

<span>Hello there! What is the weather like?</span>

<script>
    $(document).ready(function () {
        var color = $("span").css("background-color");
        console.log(color);
    });
</script>
```

Yukarıdaki örneğimizde; color değişkenine, span elementinin background-color değerini atadık ve bunu console'de yazdırdık.

css(name) metodu ile alınan color (renk) değerleri rgb şeklinde gelecektir. Dipnot olarak; rgb açılımı; red-green-blue idir. Yani temel renkler.

7.2 Positioning

Positioning ile bir elementin offset değerini öğrenebilmekteyiz. Yani pozisyonunu.

7.2.1 offset ( )

offset ( ) metodu ile herhangi bir elementin sayfa üzerindeki offset pozisyonunu öğrenebilmekteyiz. Yani; left ve top değerleri.

offset ( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").offset( )
```

offset ( ) metodunu daha iyi anlamak için basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>Hello there!</span>

<script>
    $(document).ready(function () {
        var position = $("span").offset();
        console.log(position.left); // left-side
```

```
        console.log(position.top); // top-side
    });
</script>
```

Yukarıdaki örneğimizde; span elementinin offset değerlerini elde ettik. Kullanımına dikkat ederseniz; .left ve .top ile offset değerlerine erişebilmekteyiz.

7.3 Height and Width

Height and Width ile elementlerin height (yükseklik) ve width (genişlik) değerlerini alabilmekteyiz ve aynı zamanda bu değerleri kendimiz belirleyebilmekteyiz.

7.3.1 height( )

height( ) metodu ile herhangi bir elementin height yani yükseklik değerini alabilmekteyiz.

height( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").height( )
```

Basit bir örnek ile anlama katsayımızı arttıralım.

```
<img src="test_image.png">

<script>
    $(document).ready(function () {
        var imgHeight = $("img").height();
        console.log(imgHeight);
    });
</script>
```

Yukarıdaki örneğimizde; img elementinin yükseklik değerini öğrendik ve console çıktı verdik.

7.3.2 height(val)

height(val) metodu ile istenilen elementin height değeri değiştirilebilir.

height(val) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").height(val)
```

Burada; val değeri, elementin yeni value'si yani değeri olacaktır.

Konumuzu daha iyi anlamak için bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
```

```
<script>
    $(document).ready(function () {
        $("img").height(50);
    });
</script>
```

Yukarıdaki örneğimizde; img elementinin yükseklik değerini 50 olarak değiştirdik.

7.3.3 width( )

width( ) metodu ile bir elementin width yani genişlik değeri öğrenilebilinmektedir.

width( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").width( )
```

width( ) metodu ile ilgili basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">

<script>
    $(document).ready(function () {
        var imgWidth = $("img").width();
        console.log(imgWidth);
    });
</script>
```

Yukarıdaki örneğimizde; img elementinin width yani genişlik değerini aldık ve bu değeri console'de yazdırdık.

7.3.4 width(val)

width(val) metodu ile bir elementin genişlik değeri değiştirilebilinmektedir.

width(val) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").width(val)
```

Her zaman olduğu gibi basit bir örnek ile konumuzu taçlandıralım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">

<script>
    $(document).ready(function () {
        $("img").width(50);
    });
</script>
```

Yukarıdaki örneğimizde; img elementinin width değerini 50 olarak belirledik.

BÖLÜM 8: Events

Events bir web sayfası üzerinde meydana gelebilecek her türlü Olay'dır. Örneğin; herhangi bir button veya elemente tıklanması, sayfanın açılması, kapatılması, herhangi bir text'in seçilmesi ve daha aklınıza gelebilecek herhangi bir durum event yani olay olarak nitelendirilir.

jQuery, events konusunda oldukça yetenekli. Klasik olarak JavaScript ortamında önce fonksiyon tanımlayıp daha sonra bu fonksiyonu kullanılacak elementin event'ine yazmak yerine doğrudan jQuery kodları ile halledilebilir. JavaScript konusunda basit bir örnek yaparak olayları hatırlayalım.

Örneğimiz aşağıdaki gibi olacaktır.

```
<script type="text/javascript">
    function buttonClick () {
        alert("Hello World!");
    }
</script>

<input type="button" value="Click" onclick="buttonClick()">
```

Yukarıdaki örneğimizde; bir fonksiyon tanımlayıp, daha sonra kullanılacak olan elementin (button) event'ine yazmak oldukça mantıksız bir durum. Çünkü event'lerin ucunun kaçtığı durumlarla bazen karşılaşabiliyoruz. Oysa elementin event kısmına herhangi bir şey yazmadan jQuery ile bu işlemi daha basit ve daha sonra tekrar kullanılabilir bir yolla çözebilmekteyiz. O halde aynı işlemi yerine getiren jQuery kodlarımız da aşağıdaki gibi olacaktır.

```
$("input[type='button']").click( function () {
    alert("Hello");
});
```

jQuery ile bir nevi JavaScript ve HTML kodlarını birbirlerinden ayırmış oluyoruz. Burada tabiki en büyük görev; selector'lara düşüyor.

O halde, hazırsanız vakit kaybetmeden events konusuna başlayalım.

8.1 Page Load

Page Load bir web sayfasının yüklenme durumuyla ilgilenir. jQuery'de bir web sayfası tam yüklenmeden ya da tam yüklendikten sonra çeşitli kullanım durumları olabilir. Bunlara değineceğiz.

8.1.1 ready(fn)

ready(fn) metodu, belirli durumlarda; o belirtilen durumun hazır olması durumunda işlem yapmaya başlar. ready(fn) metodu aslında her jQuery kodunda kullanıyoruz. Şöyleki:

```
$(document).ready(function () {

});
```

kullandığımız bu yapı; sayfanın tam yüklenmesiyle çalışacaktır. Sayfa tam yüklenmeden herhangi bir işlemi yerine getirmez. Bu aslında oldukça sağlıklı bir yöntemdir. Çünkü sayfa tam yüklenmeden yapılacak olan işlemler sayfadaki karışıklığın artmasına ve yerli yersiz çeşitli işlemlerin gerçekleştirilmesine neden olacaktır. Dolayısıyla yukarıdaki kullandığımız yapıyı, her jQuery kodunda kullanmamız oldukça iyi bir davranış olacaktır.

Ayrıca; bir sayfada birden fazla ready(fn) metodunu kullanabiliriz. Şöyleki:

```
$(document).ready(function () {

});

$(document).ready(function () {

});

$(document).ready(function () {

});
```

Yukarıdaki gibi birden fazla ready(fn) metodunun kullanılması herhangi bir probleme neden olmaz. Bu metotlarda sahip oldukları sıraya göre işlemlerini yerine getirirler. JavaScript, top-down çalışan bir yapıda olduğu için, yukarıdan aşağıya doğru bu metotlar işletilecektir.

8.2 Event Handling

Event Handling, çeşitli metotlar yardımıyla event'larin kontrolünü sağlamayı ve çeşitli spesifik işlemleri yerine getirmeyi sağlar. Bu konuda çeşitli fonskiyonlar var şimdi bunların üzerinde duralım.

8.2.1 bind(type, [data], fn)

bind(type, [data], fn) metodu, hem event tanımlamaya hem de event'in object (nesne)'sini tanımlamaya izin verir. Tanımlanan object ile çeşitli işlemler gerçekleşirilebilir. Dolayısıyla tek adımda birden fazla işlemi gerçekleştirmeye olanak sağlar. Oldukça yararlı bir metottur.

bind(type, [data], fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").bind("eventName", function (obj) {
    // yapicalacaklar
})
```

bind () metodunu daha iyi anlamak için bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<label id="pageX"></label>,<label id="pageY"></label>

<script>
    $(document).ready(function () {
        $("img").bind("click", function(obj) {
            $("#pageX").text(obj.pageX);
            $("#pageY").text(obj.pageY);
        });
    });
</script>
```

Yukarıdaki örneğimizde; event olarak click'i seçtik. Kullanıcı resmin herhangi bir yerine tıkladığında, tıkladığı alanın x ve y cinsinden koordinatları; label'lere yazdırılacaktır. Burada tanımladığımız obj parametresinin (object) çeşitli özellikleri var. Bu özelliklerinden; pageX | pageY, obj parametresinin birer metodu. Daha pek çok metoda sahip bu parametre. Bunlara değineceğiz. Ayrıca eventName olarak diğer tanımlı olan herhangi bir event'ı seçebiliriz. İlerleyen zamanlarda tüm event'lere değineceğiz. Şimdi sadece click event'ini bildiğimiz için bu event'i yazdık.

bind ( ) metodunun parametre olarak aldığı object'in çeşitli parametrelerinin olduğunu söylemiştik. Bu parametreler aşağıdadır.

```
altKey, ctrlKey, data, keyCode, metaKey, pageX, pageY, relatedTarget,
screenX, screenY, shiftKey, target, timestamp, type, which
```

8.2.2 one(type, [data], fn)

one(type, [data], fn) metodu da bind(type, [data], fn) metoduyla aynı mantığa ve yapıya sahiptir. Lakin one ( ) metodunu bind ( ) metodundan ayıran en önemli özellik; one(type, [data], fn) metodu bir kere çalışır. Yani one(type, [data], fn) metodu tanımlanan işlemi yapar ve çalışmayı bırakır. Lakin bind(type, [data], fn) metodu çalışmayı bırakmaz ve işlem yapmaya devam eder.

one(type, [data], fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").one("eventName", function (obj) {
    // yapicalacaklar
})
```

bind(type, [data], fn) metodu konusunda yaptığımız örneği one(type, [data], fn) metodu ile yapalım. örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<label id="pageX"></label>,<label id="pageY"></label>

<script>
    $(document).ready(function () {
        $("img").one("click", function(obj) {
            $("#pageX").text(obj.pageX);
            $("#pageY").text(obj.pageY);
        });
    });
</script>
```

Daha öncede dediğimiz gibi; one ( ) metodu belirtilen işi bir kere yapar ve çalışmayı bırakır. Örneğimiz için konuşacak olursak; kullanıcı resme tıkladığı zaman bir sefere mahsus resmin x ve y koordinatları belirtilecektir. Kullanıcı tekrar tıkladığı zaman herhangi bir işlem yapmayacaktır.

8.2.3 trigger(type, [data])

trigger(type, [data]) metodu, bir olay gerçekleştiği zaman diğer bir işleminde tetiklenmesini sağlar. Örneğin bir butona tıklandığı zaman yapılması gereken sıralı işlemler olabilir. Bu işlemleri trigger(type, [data]) metodu ile tetiklenebilir.

trigger(type, [data]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").trigger(type, [data])
```

Çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<button>Click</button>

<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("button").trigger(alert("Button is clicked!"));
        });
    });
</script>
```

Yukarıdaki örneğimizde; button'a tıklayınca, alert( ) metodu çalışacaktır.

Daha spesifik bir örnek olarak; jQuery ekinin hazırladığı örnek verilebilir. Bu örnekte aşağıdaki gibidir.

```
<button>Button #1</button>
```

```
<button>Button #2</button>
<div><span>0</span> button #1 clicks.</div>
<div><span>0</span> button #2 clicks.</div>

<script>
    $(document).ready(function(){
        $("button:first").click(function () {
            update($("span:first"));
        });

        $("button:last").click(function () {
            $("button:first").trigger('click');
                update($("span:last"));
        });

        function update(j) {
          var n = parseInt(j.text(), 0);
          j.text(n + 1);
        }
  });
```

Yukarıdaki örneğimizde; birinci button'a tıklayınca hem birinci hem de ikinci button tetiklenecek ve tıklama sayıları artacaktır. Aynı şey ikinci button için de geçerlidir.

8.2.4 triggerHandler(type, [data])

triggerHandler(type, [data]) metodu, trigger(type, [data]) metoduna benzer bir yapıdadır. Daha doğrusu ikisi de aynı işleri yerine getirirler. İki metodun birbirinden ayıran kısım ise:

- trigger(type, [data]) metodu çalıştığı zaman yaptığı işlemlerden browser (tarayıcı)'ı haberdar eder.
- triggerHandler(type, [data]) metodu çalıştığı zaman ise yaptığı işlemlerden browser'ı haberdar etmez.

triggerHandler(type, [data]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").triggerHandler(type, [data])
```

Daha akılda kalıcı olması açısından basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<button>Click</button>
<input type="text" value="0">

<script>
    $(document).ready(function () {
        $("button:last").click(function(){
            $("input").triggerHandler("click");
        });
```

```
        $("input").click(function(){
            var counter = parseInt($("input").val());
            counter++;
            $("input").val(counter);
        });
    });
</script>
```

Yukarıdaki örneğimizi; triger ( ) metodu ile de gerçekleştirebilirdik. triger fonksiyonları, bir elemente atanmış olan event'leri de çağırma yetkisine sahiptir. Örneğimizde; input'a tıklandığı zaman, input içerisindeki değer artarken bu işlemi button'a tıkladığımız zamanda gerçekleştirebilmekteyiz.

8.2.5 unbind([type], [data])

unbind([type], [data]) metodu, bind(type, [data], fn) metodunun zıttıdır. Yani bind(type, [data], fn) metodu ile yapılan işlemler, unbind([type], [data]) metodu ile geri alınabilir. unbind([type], [data]) kullanımı aşağıdaki gibidir.

```
$("selector").unbind([type], [data])
```

unbind([type], [data]) metodu ile çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<button>bind ()</button>
<button>unbind ()</button>

<script>
    $(document).ready(function () {
        $("button:first").bind("click", function () {
            $("button:first").attr("disabled", "disabled");
        });

        $("button:last").click(function () {
            $("button:first").unbind();
        });
    });
</script>
```

Yukarıdaki örneğimizde; birinci button'a tıklandığı zaman bu button disabled olacaktır. Lakin birinci button'a tıklamadan ikinci button'a tıklandığı zaman birinci button'a artık bind ( ) metodu eklenemez. Yani bir anlamda; unbind ( ) metodu ile birinci button'u bind ( ) metoduna karşı korumaya aldık.

8.3 Interaction Helpers

Interaction Helpers, genellikle çeşitli görsel event'ler üzerinde durur. Birazdan bahsettiğimizde aslında ne demek istediğimi anlayacaksınız. O yüzden fazla laf sarfetmeden metotlarımızı anlatmaya geçelim.

8.3.1 hover(over, out)

hover(over, out) metodu, belirtilen elementin mouse ile üzerine gelince belirli tepkiler verilmesini sağlar. Dikkat ederseniz; metodumuz, over, out şeklinde parametreleri var. Yani belirtilen elementin üzerine mouse ile gelince ve mouse ayrılınca verilen tepkileri belirleyebiliriz.
hover(over, out) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").hover(over, out)
```

Burada; over ve out kelimeleri yerine anonymous (anonim) fonksiyonlar tanımlayıp bu fonksiyonlar aracılığı ile işlemlerimizi yerine getireceğiz.

Basit bir örnek yaparak, anlattıklarımızı taçlandıralım. Örneğimiz aşağıdaki gibi olacaktır.

```
<span>Hello there! How do you do?</span>

<script>
    $(document).ready(function () {
        $("span").hover(
            function (){
                $(this).css("background-color", "green").css("color",
"white");
            },
            function (){
                $(this).css("background-color", "white").css("color",
"black");
            }
        );
    });
</script>
```

Yukarıdaki örneğimizde; span elementi üzerine mouse gelindiğinde, arkaplan rengi: yeşil ve yazı rengi beyaz olacaktır. mouse span elementi üzerinden ayrılınca tekrar eski halini alacaktır. Tabiki bu işlemleri biz css ( ) metodunu kullanarak gerçekleştirdik.

Burada göstermek istediğimi başka bir şey ise; şayet kullanıcı belirtilen elementin üzerine geldiğinde bir efekt verilmek isteniyorsa ve mouse ayrıldıktan sonra da bu efektin kalınması isteniyorsa, sadece over'ın metodu yazılabilir. Şöyleki;

```
$("span").hover(
     function (){
         $(this).css("background-color", "green").css("color",
"white");
});
```

şeklinde yazılabilinir.

8.4 Event Helpers

Event Helpers, klasik JavaScript'ten bildiğimiz event'lar. jQuery ile event'ler biraz farklılık göstermiş ve daha işlevsel hale gelmiştir. Şimdi birlikte bu event'ları inceleyelim.

8.4.1 blur(fn)

blur(fn) metodu, kullanılan elementten focus özelliği gittiği zaman çalışmaya başlar. Örneğin bir input="text" alanınız var. Kullanıcı herhangi bir şeyler yazdı, elementten çıktığı zaman blur(fn)`metodu çalışmaya başlar.

blur(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").blur(fn)
```

blur(fn) metodundaki fn fonksiyonun kısaltmasıdır. Yani blur ( ) metodunun parantezleri içerisine bir fonksiyon yazılarak yapılacak olan işlemler tanımlanır.

Çok basit bir örnek yapalım ve blur(fn) metodunu tanımaya çalışalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<input type="text" />

<script>
    $(document).ready(function () {
        $("input[type='text']").blur(function () {
            var text = $(this).val();
            console.log(text);
        });
    });
</script>
```

Yukarıdaki örneğimizde; kullanıcı input alanından çıktıktan sonra, giriş yapmış olduğu karakterleri console.log ile yazdırdık.

8.4.2 change(fn)

change(fn) metodu, belirtilen elementin değerinin değiştirilmesi durumunda çalışmaya başlar. Herhangi bir element için change(fn) metodunu kullanabilirsiniz.

change(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").change(fn)
```

Bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<select>
```

```
    <option selected="selected">Turkey</option>
    <option>Finland</option>
    <option>Canada</option>
    <option>USA</option>
    <option>Australia</option>
</select>

<script>
    $(document).ready(function () {
        $("select").change(function () {
            var text = $(this).val();
            console.log(text);
        });
    });
</script>
```

Yukarıdaki örneğimizde; select alanından herhangi bir değer seçildiği zaman change(fn)'çalışmaya başlayacaktır.

8.4.3 click(fn)

click(fn) metodu, bildiğimiz gibi belirtilen elemente tıkladığımızda çalışamaya başlar.

click(fn)metodunun kullanımı aşağıdaki gibidir.

```
$("selector").click(fn)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<button>Click</button>

<script>
    $(document).ready(function () {
        var counter = 0;
        $("button").click(function () {
            counter++;
            $(this).text(counter);
        });
    });
</script>
```

Yukarıdaki örneğimizde; kullanıcı button'a tıkladıkça counter değeri artacak ve bu değer button'un text'i olarak değiştirilecektir.

8.4.4 dblclick(fn)

dblclick(fn) metodu, belirlenen elementi çift tıklama ile çalışır. Bildiğiniz gibi click(fn) metodu da tek tıklamayla çalışır.

dblclick(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").dblclick(fn)
```

Bir de örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<button>Click</button>

<script>
    $(document).ready(function () {
        var counter = 0;
        $("button").dblclick(function () {
            counter++;
            $(this).text(counter);
        });
    });
</script>
```

click(fn) metodu ile yaptığımız örneğimizi, dblclick(fn) metoduna uyarladık. Artık button'a çift tıklamayla counter bir artacak ve button'a text değeri olarak eklenecek.

8.4.5 focus(fn)

focus(fn) metodu, herhangi bir elemente focus ya da Türkçe'siyle odak'lanınca çalışmaya başlar. Örneğin; blur ( ) metodu, odak kaybolunca çalışmaya başlardı. İşte blur ( ) metodunun zıttı da focus ( ) metodudur.
focus(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").focus(fn)
```

Çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<input type="text" />

<script>
    $(document).ready(function (){
        $("input[type='text']").focus(function (){
            console.log("focused!");
        });
        $("input[type='text']").blur(function (){
            console.log("blured!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; input="text" alanına tıklanınca yani odaklanınca; focus ( ) metodu çalışmaya başlayak, odaklanma bitince yani input="text" alanından çıkınca da blur ( ) metodu çalışmaya başlayacaktır.

8.4.6 keydown(fn)

keydown(fn) metodu, klavye üzerindeki herhangi bir tuşa basıldığı anda çalışmaya başlar.

keydown(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").keydown(fn)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olsun.

```
<script>
    $(document).ready(function (){
        $("input").keydown(function() {
            console.log("keydown");
        });
    });
</script>
```

Yukarıdaki örneğimizde; kullanıcı input="text" alanına herhangi bir karakter yazdığı zaman keydown ifadesini console'e yazdırdık.

8.4.7 keypress(fn)

keypress(fn) metodu, keydown(fn) metodu ile aynı yapıdadır. Yani kullanıcı klavye üzerinde herhangi bir tuşa bastığı anda bu metot çalışmaya başlar.

keypress(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").keypress(fn)
```

Örneğimiz aşağıdaki gibi olacaktır.

```
<input type="text" />

<script>
    $(document).ready(function (){
        $("input").keypress(function() {
            console.log($(this).val());
        });
    });
</script>
```

Yukarıdaki örneğimizde; kullanıcı input="text" alanına yazdığı karakterlerin tümü, hangi karaktere basmışsa sırasıyla tüm karakterleri console'e yazacaktır.

Daha genel bir örnek vermek istersek; örneğin kullanıcının enter tuşuna bastığını anlamak için bir key listener yazabiliriz.

```
<script>
    $(document).ready(function (){
        $(document).keypress(function(e) {
            if(e.which == 13) {
                console.log("pressed enter!");
            }
        });
    });
</script>
```

Yukarıdaki örneğimizde; kullanıcı sayfa üzerindeyken enter tuşuna bastığı zaman console'e "pressed enter!" yazdıracaktır. enter tuşunun sayısal kodu: 13'tür. Bir dipnot daha vereyim. esc tuşunun sayısal kodu da 27'dir.

8.4.8 keyup(fn)

keyup(fn) metodu, keydown(fn) ve keypress(fn) metodu ile aynı görevdedir. Yani herhangi bir tuşa basıldığı anda çalışmaya başlar. Benim size tavsiyem genelde bu metodu kullanmanızdır. Diğer iki metotta da bazen gecikmeler olabiliyor lakin bu metotta böyle bir durumla henüz karşılaşmadım.

keyup(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").keyup(fn)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<input type="text" />
<span></span>

<script>
    $(document).ready(function (){
        var ch = "";
        $("input").keyup(function() {
            ch = $(this).val();
            $("span").text(ch);
            ch = "";
        });
    });
</script>
```

Yukarıdaki örneğimizde; input="text" alanına girdiğiniz tüm karakterler span alanına yazdırılacaktır. Yazdığınız karakterleri sildiğiniz zaman bu karakterler de span elementinden silinecektir.

8.4.9 mousedown(fn)

mousedown(fn) metodu, belirtilen elementin üzerine mouse ile gelip ve de tıklayınca çalışmaya başlar.

mousedown(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").mousedown(fn)
```

Çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png" />

<script>
    $(document).ready(function (){
        $("img").mousedown(function (){
            console.log("mousedown on img");
        });
    });
</script>
```

Yukarıdaki örneğimizde; mouse ile img elementinin üzerine gelip tıklayınca mousedown(fn) metodu çalışmaya başlayacaktır.

8.4.10 mousemove(fn)

mousemove(fn) metodu, belirtilen elementin üzerinde mouse ile gezinmesi sonucu veya mouse gelindiği zaman çalışmaya başlacaktır.

mousemove(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").mousemove(fn)
```

Yapmış olduğumuz örneğimize bir bakalım.

```
<img src="test_image.png" />
<span>0</span>
<span>0</span>

<script>
    $(document).ready(function (){
        $("img").mousemove(function (e){
            $("span:first").text(e.pageX);
            $("span:last").text(e.pageY);
        });
    });
</script>
```

Yukarıdaki örneğimizde; mouse ile img elementinin üzerinde dolaştığımız zaman, gezdiğimiz yerlerin x ve y koordinatlarını span elementlerine atadık.

Örneğin; sayfa üzerinde gezinen mouse'nin x ve y değerlerini almak için aşağıdaki kodu yazmamız yeterlidir.

```
$(document).ready(function (){
    $(document).mousemove(function (e){
        $("span:first").text(e.pageX);
        $("span:last").text(e.pageY);
    });
});
```

8.4.11 mouseout(fn)

mouseout(fn) metodu, belirlenen elementtin sınırları dışına mouse'nin çıkması durumunda çalışır.

mouseout(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").mouseout(fn)
```

mouseout(fn) metodu ile ilgili basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png" />

<script>
    $(document).ready(function (){
        $("img").mouseout(function (){
            alert("img out!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; mouse, img elementi sınırlarının dışına çıktığı zaman uyarı verecektir.

8.4.12 mouseover(fn)

mouseover(fn) metodu, belirtilen elementin üzerine mouse ile gelindiğinde çalışmaya başlar.

mouseover(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").mouseover(fn)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png" />

<script>
    $(document).ready(function (){
        $("img").mouseover(function (){
            alert("img over!");
```

```
        });
    });
</script>
```

Yukarıdaki örneğimizde; mouse, img elementinin üzerine geldiğinde uyarı verecektir.

8.4.12 mouseover(fn)

mouseover(fn) metodu, belirtilen elementin üzerine gelip tıklandığı zaman çalışır.

mouseover(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").mouseup(fn)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png" />

<script>
    $(document).ready(function (){
        $("img").mouseup(function (){
            alert("img over!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; img elementinin üzerine gelip, tıklanınca uyarı verecektir.

8.4.13 resize(fn)

resize(fn) metodu, belirtilen elementin boyutunun değiştirilmesiyle çalışır. Bu metodunun en iyi kullanım yeri; jQuery UI'dir. Çünkü jQuery UI'nin resizable metodu ile elementlerin boyutları değiştirilebilinmektedir. Dolayısıyla herhangi bir elementin boyutunun değiştirilmesiyle bu metot devreye girecektir.

resize(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").resize(fn)
```

Bir örnek yapalım. Mesela sayfanın boyutunun değiştirilmesini kontrol edelim ve uyarı mesajı verelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<script>
    $(document).ready(function (){
        $(window).resize(function (){
            alert("page is resized!");
        });
    });
</script>
```

8.4.14 scroll(fn)

scroll(fn) metodu, sayfanın kaydırma çubuğu etkin olduğu zaman çalışmaya başlar. Örneğin kullanıcı kaydırma çubuğunu hareket ettirdiği zaman bu metot çalışır ve tanımlanan işlemleri yerine getirir.
scroll(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").scroll(fn)
```

Örneğin, kullanıcı kaydırma çubuğunu kullanmaya başladığı zaman uyarı verelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<script>
    $(document).ready(function (){
        $(window).scroll(function (){
            alert("scroll is used!");
        });
    });
</script>
```

8.4.15 select(fn)

select(fn) metodu, herhangi bir input alanındaki text'in seçilmesi durumunda çalışır. Diğer herhangi bir elemente çalışmaz. Yalnızca input alanı olmalıdır.

select(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").select(fn)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<input type="text" value="Hello there!" />

<script>
    $(document).ready(function (){
        $("input").select(function (){
            console.log("text in input is selected!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; input="text" alanındaki value değeri mouse ile seçildiği zaman console'a uyarı verecektir.

8.4.16 submit(fn)

submit(fn) metodu, herhangi bir form submit yapıldığı zaman çalışmaya başlayacaktır.

submit(fn) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").submit(fn)
```

Basit bir örnek ile konumuzu tamamlayalım. Örneğimiz aşağıdaki gibidir.

```
<form action="#" method="post">
    <input type="text" />
    <input type="text" />
    <input type="submit" />
</form>

<script>
    $(document).ready(function (){
        $("form").submit(function (){
            alert("Form is submitted!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; form, submit yapıldığı zaman uyarı verecektir.

BÖLÜM 9: Effects

Effects yani Efektler jQuery'in sahip olduğu en güzel özelliklerden bir tanesidir, bana göre. Görsel açıdan bir çok özellik kolaylıkla tanımlanabilmekte ve kullanılabilinmektedir.

Bu konumuz boyunca effects'ler üzerinde duracağız. Hazırsanız, o halde hemen başlayalım.

9.1 Basic

Basic efektler, göster ve gizle gibi temel işlemlerden oluşmaktadır. Tanımlı olan fonksiyonlara birlikte bakalım.

9.1.1 show( )

show( ) metodu, gizli olan herhangi bir elementin gösterilmesini sağlar.

show( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").show( )
```

Basit bir örnek verelim. Örneğimiz aşağıdaki gibi olsun.

```
<img src="test_image.png" hidden="hidden">
<button>Show</button>

<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").show();
        });
    });
</script>
```

Yukarıdaki örneğimizde; hidden özelliğine sahip olan img elementini, show( ) metodu ile gösterdik.

9.1.2 show(speed, [callback])

show(speed, [callback]) metodu için zaman skalası belirtilebilir. Ayrıca zamanın yanında bir de fonksiyon tanımlanarak başka işlemlerde gerçekleştirilebilir.

show(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").show(speed, [callback])
```

Zaman olarak: slow | normal | fast ibareleri kullanılabilir. Örneğin:

```
$("selector").show("slow");
$("selector").show("normal");
$("selector").show("fast");
```

Bu ifadelerin yanı sıra sayısal türden bir değer de girilebilir. Burada girilecek olan değer milliseconds yani milisaniye cinsinden olmalıdır. Örneğin;

```
$("selector").show(2000);
```

Basit bir örnek ile anlattıklarımızı taçlandıralım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png" hidden="hidden">
<button>Show</button>

<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").show(5000);
        });
    });
</script>
```

9.1.3 hide( )

hide( ) metodu, show( ) metodunun tam tersi yani belirtilen elementi gizleyecektir.

hide( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").hide( )
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>Show</button>
<button>Hide</button>

<script>
    $(document).ready(function (){
        $("button:first").click(function() {
            $("img").show();
        });
        $("button:last").click(function() {
            $("img").hide();
        });
    });
</script>
```

Yukarıdaki örneğimizde; Hide button'una tıklayınca; hide ( ) metodu ve Show button'una tıklayınca; show ( ) metodu çalışacaktır.

9.1.4 hide(speed, [callback])

hide(speed, [callback]) metodu için zaman skalası belirtilebilir. Ayrıca zamanın yanında bir de fonksiyon tanımlanarak başka işlemler de gerçekleştirilebilir.

hide(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").hide(speed, [callback])
```

Zaman olarak; slow, normal ve fast ibareleri kullanılabilir. Örneğin;

```
$("selector").hide("slow");
$("selector").hide("normal");
$("selector").hide("fast");
```

Bu ifadelerin yanı sıra sayısal türden bir değer de girilebilir. Burada girilecek olan değer milliseconds yani milisaniye cinsinden olmalıdır. Örneğin;

```
$("selector").hide(2000);
```

Basit bir örnek ile anlattıklarımızı taçlandıralım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>Show</button>
<button>Hide</button>

<script>
    $(document).ready(function (){
        $("button:first").click(function() {
            $("img").show("fast");
        });
        $("button:last").click(function() {
            $("img").hide("slow");
        });
    });
</script>
```

Yukarıdaki örneğimizde; hide ("fast") ve hide ("slow") kullandık. Yani gizlenirken hızlı, açılırken "nazlı" pardon yavaş!

9.1.5 toggle( )

toggle( ) metodu, hide ( ) ve show ( )'larını birleştirir. Yani kullanılan element, gizliyse gösterir, gösteriliyorsa gizlenir.
toggle( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").toggle( )
```

Basit bir örnek gerçekleştirelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>Show / Hide</button>

<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").toggle();
        });
    });
</script>
```

Yukarıdaki örneğimizde; tek bir button'a hem hide ( ) hem de show ( ) metotlarını ekledik.

9.2 Sliding

Sliding kısmında, çeşitli slayt efektleri göreceğiz.

9.2.1 slideDown(speed, [callback])

slideDown(speed, [callback]) metodu, belirtilen elementin slayt şeklinde yukarıdan aşağı doğru açılmasını sağlar. Burada önemli olan elementin hidden özelliğinde olması gerekliliğidir.

slideDown(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").slideDown(speed, [callback])
```

slideDown(speed, [callback]) metoduna, hız değeri verilebilir. Bu hız değeri örneğin;

```
$("selector").slideDown("slow");
$("selector").slideDown("normal");
$("selector").slideDown("fast");

// ya da
$("selector").slideDown(2000);
```

olabilir.

Ayrıca slideDown(speed, [callback]) elementine bir fonksiyon tanımlanarak değişik işlemler de yaptırılabilinir.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png" hidden="hidden">
<button>slideDown</button>
```

```
<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").slideDown("slow");
        });
    });
</script>
```

Yukarıdaki örneğimizde; hidden özelliğine sahip img elementi, "slow" bir şekilde açılacaktır.

9.2.2 slideUp(speed, [callback])

slideUp(speed, [callback]) metodu, slideDown(speed, [callback]) elemetinin tam tersi olarak; belirtilen elementi slayt şeklinde kapatır.

slideUp(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").slideUp(speed, [callback])
```

slideUp(speed, [callback]) metoduna, hız değeri verilebilir. Bu hız değeri örneğin;

```
$("selector").slideUp("slow");
$("selector").slideUp("normal");
$("selector").slideUp("fast");

// ya da
$("selector").slideUp(2000);
```

olabilir.

Ayrıca slideUp(speed, [callback]) elementine bir fonksiyon tanımlanarak değişik işlemler de yaptırılabilinir.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>slideDown</button>
<button>slideUp</button>

<script>
    $(document).ready(function (){
        $("button:first").click(function() {
            $("img").slideDown("slow");
        });
        $("button:last").click(function() {
            $("img").slideUp("slow");
        });
    });
```

```
</script>
```

Yukarıdaki örneğimizde; img elementi, slideDown isimli button'a tıklayınca; slayt şeklinde açılmakta ve slideUp isimli button'a tıklayınca; slayt şeklinde kapanmaktadır.

9.2.3 slideToggle(speed, [callback])

slideToggle(speed, [callback]) metodu, slideDown(speed, [callback]) ve slideUp(speed, [callback]) metotlarını birleşimidir. Aynı belirtilen element kapalıysa açar ve açıksa kapatır.

slideToggle(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").slideToggle(speed, [callback])
```

slideToggle(speed, [callback]) metoduna, hız değeri verilebilir. Bu hız değeri örneğin;

```
$("selector").slideToggle("slow");
$("selector").slideToggle("normal");
$("selector").slideToggle("fast");
```

şeklinde olabileceği gibi sayısal bir değer de;

```
$("selector").slideToggle(2000);
```

olabilir.

Ayrıca slideToggle(speed, [callback]) elementine bir fonksiyon tanımlanarak değişik işlemler de yaptırılabilinir.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>slideToggle</button>

<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").slideToggle("slow");
        });
    });
</script>
```

9.3 Fading

Fading, elementlerin opacity ve firing değerlerini kullanarak görünüp kaybolmalarını sağlar. Daha açıklayıcı olması açısından; belirtilen elementi soluklaştırarak kaybeder ve kapalı olan elementi soluktan başlayarak gösterir.

9.3.1 fadeIn(speed, [callback])

fadeIn(speed, [callback]) metodu, belirtilen elementi soluktan başlayarak gösterir. Kullanımı itibari ile Sliding metotlarına benzer. Burada dikkat edilmesi gereken nokta; elementin hidden olması gerekir.

fadeIn(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").fadeIn(speed, [callback])
```

fadeIn(speed, [callback]) metoduna, hız değeri verilebilir. Bu hız değeri örneğin;

```
$("selector").fadeIn("slow");
$("selector").fadeIn("normal");
$("selector").fadeIn("fast");
```

şeklinde olabileceği gibi sayısal bir değer de;

```
$("selector").fadeIn(2000);
```

olabilir.

Ayrıca fadeIn(speed, [callback]) elementine bir fonksiyon tanımlanarak değişik işlemler de yaptırılabilinir.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png" hidden="hidden">
<button>fadeIn</button>

<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").fadeIn("slow");
        });
    });
</script>
```

Yukarıdaki örneğimizde; hidden olan img elementi, fadeIn ( ) metodu ile slow bir şekilde ekranda belirecektir.

9.3.2 fadeOut(speed, [callback])

fadeOut(speed, [callback]) metodu, fadeIn(speed, [callback]) metodunun aksine, belirtilen elementi soluklaştırarak kaybeder.

fadeOut(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").fadeOut(speed, [callback])
```

fadeOut(speed, [callback]) metoduna, hız değeri verilebilir. Bu hız değeri örneğin;

```
$("selector").fadeOut("slow");
$("selector").fadeOut("normal");
$("selector").fadeOut("fast");
```

şeklinde olabileceği gibi sayısal bir değer de;

```
$("selector").fadeOut(2000);
```

olabilir.

Ayrıca fadeOut(speed, [callback]) elementine bir fonksiyon tanımlanarak değişik işlemler de yaptırılabilinir.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>fadeIn</button>
<button>fadeOut</button>

<script>
    $(document).ready(function (){
        $("button:first").click(function() {
            $("img").fadeIn("slow");
        });
        $("button:last").click(function() {
            $("img").fadeOut("slow");
        });
    });
</script>
```

Yukarıdaki örneğimizde; fadeIn button'una tıklayınca resim gösterilecek ve fadeOut button'una tıklayınca resim kaybolaşacaktır.

9.3.3 fadeToggle(speed, [callback])

fadeToggle(speed, [callback]) metodu, fadeIn(speed, [callback]) ve fadeOut(speed, [callback]) metotlarının yaptığı işleri tek başına yapar. Yani; belirtilen element gösteriliyorsa gizler, gizleniyorsa gösterilir.

fadeToggle(speed, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").fadeToggle(speed, [callback])
```

fadeToggle(speed, [callback]) metoduna, hız değeri verilebilir. Bu hız değeri örneğin;

```
$("selector").fadeToggle("slow");
$("selector").fadeToggle("normal");
$("selector").fadeToggle("fast");
```

şeklinde olabileceği gibi sayısal bir değer de;

```
$("selector").fadeToggle(2000);
```

olabilir.

Ayrıca fadeToggle(speed, [callback]) elementine bir fonksiyon tanımlanarak değişik işlemler de yaptırılabilinir.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>fadeToggle</button>

<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").fadeToggle("fast");
        });
    });
</script>
```

Yukarıdaki örneğimizde; fadeToggle button'una tıklayınca; img elementi gösteriliyorsa gizlenecek ve gizleniyorsa gösterilecektir.

9.3.4 fadeTo(speed, opacity, [callback])

fadeTo(speed, opacity, [callback]) metodu, ilginç bir metot. Belirtilen elementin belirtilen opacity kadar soluklaşmasını sağlar. İşin aslı; bazı durumlarda oldukça yararlı olabiliyor.

fadeTo(speed, opacity, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").fadeTo(speed, opacity, [callback])
```

fadeTo(speed, opacity, [callback]) metoduna, hız değeri verilebilir. Bu hız değeri örneğin;

```
$("selector").fadeTo("slow", opacity);
$("selector").fadeTo("normal", opacity);
$("selector").fadeTo("fast", opacity);
```

şeklinde olabileceği gibi sayısal bir değer de;

```
$("selector").fadeTo(2000, opacity);
```

olabilir.

Ayrıca fadeTo(speed, opacity, [callback]) elementine bir fonksiyon tanımlanarak değişik işlemler de yaptırılabilinir.

fadeTo(speed, opacity, [callback]) metoduna, opacity değeri girilmeden herhangi bir işlem yapılamaz. O yüzden buraya [0.0 - 1.0] arasında bir değer girmelisiniz.

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<button>fadeTo</button>

<script>
    $(document).ready(function (){
        $("button").click(function() {
            $("img").fadeTo("fast", 0.42);
        });
    });
</script>
```

Yukarıdaki örneğimizde; fadeTo button'una tıklanınca, img elementini % 42 değerinde soluklaştıracaktır.

9.4 Custom

Custom kullanıcının kendisinin tabir-i caizse kafasına göre animasyonlar tanımlamasıyla ilgilenir. Sahip olduğu çeşitli fonksiyonlar ile oldukça güzel ve etkili fonksiyonlar elde etmek mümkündür.

9.4.1 animate(params, [duration], [easing], [callback])

Eskiden Macromedia Flash vardı. Tabi daha sonra Adobe Flash oldu. Sonra Flash'ın ismi falan değişti. Herneyse, flash ile güzel animasyonlar hatta web siteleri falan oluşturabiliyorduk. animate(params, [duration], [easing], [callback]) metodu da aslında bir nevi Flash'ı andırıyor. Geliştiriye, sahip olduğu bir çok özellikle iyi bir animasyon deneyimi yaşatıyor.

animate(params, [duration], [easing], [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").animate(params, [duration], [easing], [callback])
```

animate ( ) metodunu daha iyi anlamak için üzerinde biraz konuşalım. Örneğin yukarıdaki kullanımda, paranrez içlerinde olan kavramdan detaylıca bahsedelim.

animate ( ) metodunu aşağıdaki gibi basit bir şekilde gösterebiliriz.

```
$("selector").animate({styles}, speed, easing, callback)
```

{styles} parametresi, CSS özelliklerini simgeler. Peki nedir bu CSS özellikleri?

```
backgroundPositionX, backgroundPositionY, borderWidth,
borderBottomWidth, borderLeftWidth, borderRightWidth, borderTopWidth,
borderSpacing, margin, marginBottom, marginLeft, marginRight,
marginTop, outlineWidth, padding, paddingBottom, paddingLeft,
paddingRight, paddingTop, height, width, maxHeight, maxWidth,
minHeight, minWidth, fontSize, bottom, left, right, top,
letterSpacing, wordSpacing, lineHeight, textIndent
```

speed animasyonun hızını belirler. Bu hız değeri; `slow | normal | fast gibi bir ifade olabileceği gibi sayısal bir değer de (milisaniye) olabilir.

easing, jQuery tarafında tanımlı animasyon çeşitleridir. Varsayılan olarak; Swing'tir. jQuery'nin web sayfasında tanımlı bir sürü animasyon çeşidi var. Buradan yararlanabilirsiniz. Bunun dışında ekstra olarak plugin (eklenti)'ler kullanabilirsiniz. Örneğin: [http://easings.net/#] sitesini ziyaret edilebilirsiniz.

callback ise function'u simgeler. Yani herhangi bir fonksiyon tanımlayıp başka işlerde yaptırabilirsiniz.

animate önemli bir kavram o yüzden bir çok örnek yaparak bu konuyu çok iyi anlamaya çalışalım. İlk örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png"><br>
<button>Animate</button>

<script>
    $(document).ready(function (){
        $("button").click(function () {
            $("img").animate(
            {
                height: "100px",
                width: "100px",
                opacity: "0.33"
            }, "fast", "linear");
        });
    });
</script>
```

Yukarıdaki örneğimizde; img elementi, Animate isimli button'a tıklandığı zaman belirtilen özelliklerde küçültülecek ve 0.33 değerinde soluklaştırılacaktır. Animasyon, slow ve linear şeklinde oluşacaktır.

Bir başka örneğimiz ise şöyle olacaktır.

```
<style>
    div{
        background-color: green;
        color: white;
        border: 2px black dotted;
    }
</style>

<div>Hello World</div>
<button>Animate</button>

<script>
    $(document).ready(function (){
        $("button").click(function () {
            $("div").animate(
            {
                width: "100px",
                padding: "50px"
            }, "slow", "linear");
        });
    });
</script>
```

Görüldüğü gibi animate kullanımı oldukça basit.

Şimdi size başka bir konudan bahsetmek istiyorum. Yukarıdaki örnekler de olduğu gibi basit animasyonlar oluşturabiliriz. Bunun yanısıra hareketli, deyim yerindeyse; uçan, kaçan animasyonlar da oluşturabilmek mümkün.
Basit bir örnek yapalım. Örneğimiz üzerinden konuşalım.

```
<style>
    div{
        position: absolute;
        background-color: red;
        height: 50px;
        width:  50px;
    }
</style>

<button id="right">Right</button>
<button id="left">Left</button>
<button id="top">Top</button>
<button id="bottom">Bottom</button>
<br/><br/>
<div></div>

<script>
    $(document).ready(function (){
        setTimeout(zipZip(), 1000);
        $("#right").click(function(){
            $("div").animate({"left": "+=50px"}, "slow");
```

```
        });
        $("#left").click(function(){
            $("div").animate({"left": "-=50px"}, "slow");
        });
        $("#top").click(function(){
            $("div").animate({"top": "-=50px"}, "slow");
        });
        $("#bottom").click(function(){
            $("div").animate({"top": "+=50px"}, "slow");
        });
    });
</script>
```

Yukarıdaki örneğimizde; 1 div elementi ve 4 tane de button elementi oluşturduk. Bu div elementinin CSS değerleriyle de oynadık. Şimdi örneğimizde yaptığımız şu:

Kulanıcı, Right isimli butona tıkladığında, div elementi sağa doğru hareket edecek ("left": "+=50px").

Kullanıcı, Left isimli butona tıkladığında, div elementi sola doğru hareket edecek ("left": "-=50px").

Aynı şekilde Top isimli butona tıkladığında, div elementi yukarı doğru hareket edecek ("top": "-=50px")

Ve Bottom isimli butona tıkladığında, div elementi aşağı doğru hareket edecek ("top": "+=50px").

Bu hareket işlemlerini; left | right ve tanımlara verdiğimiz; -+=number_px değeri ile sağlıyoruz. Bu ve buna benzer bir çok özellik ile daha etkili animasyonlar elde etmek oldukça mümkün.

Son olarak; bonus bir örnek yapalım. Bu örneğimizde; zıp zıp zıplayan kare kutu yapalım (top'lu olanı da siz yapın, ödev). Örneğimiz aşağıdaki gibi olacaktır.

```
<style>
    div{
        position: absolute;
        background-color: red;
        height: 50px;
        width:  50px;
    }
</style>

<div></div>

<script>
    $(document).ready(function (){
        setTimeout(zipZip(), 1000);
```

```
    });

    function zipZip(){
            $("div").animate({"top": "+=50px"}, "slow");
            $("div").animate({"top": "-=50px"}, "slow");
            zipZip();
    }
</script>
```

9.4.2 animate(params, options)

animate(params, options) metodu, kullanıcıyı özgür bırakır ve istediği animasyonu oluşturmasını sağlar. 9.4.1 animate(params, [duration], [easing], [callback]) no'lu başlıkta anlattıklarımızdan herhangi bir farkı yok. Yalnızca küçük bir farklılık var. O da istediğiniz animate () metodunu uç uca ekleyebilirsiniz.

animate(params, options) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").animate(params, options)
```

Daha önce de dediğimiz gibi uç uca ekleme yaparak yeni animasyonlar elde etmeniz mümkün. Yani:

```
$("selector").animate(params, options).animate(params,
options).animate(params, options)
```

şeklinde olacaktır.

Herneyse, basit bir örnek yaparak konumuzu tamamlayalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<img src="test_image.png">
<br/><br/><br/>
<button>Click Me!</button>

<script>
    $(document).ready(function (){
        $("button").click(function () {
            $("img")
            .animate({"height": "400px"}, "slow")
            .animate({"width": "500px"}, "fast")
            .animate({"opacity": 0.42})
            .animate({"borderRadius": "50px"});
        });
    });
</script>
```

9.4.3 stop( )

stop( ) metodu, tahmin edebileceğiniz gibi animasyonun durmasını sağlar.

stop( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").stop( )
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style>
    div{
        position: absolute;
        background-color: red;
        height: 50px;
        width:  50px;
    }
</style>

<div></div>
<br/><br/><br/>

<button>Start</button>
<button>Stop</button>

<script>
    $(document).ready(function (){
        $("button:first").click(function (){
            $("div").animate({"left": "+=100px"}, 5000);
        });
        $("button:last").click(function (){
            $("div").stop();
        });
    });
</script>
```

Yukarıdaki örneğimizde; Start butonuna tıklayınca animasyon başlayacak ve Stop butonuna tıklayınca animasyon duracaktır.

9.4.4 queue( )

queue( ) metodu, belirtilen elemente atanan animasyonun kaç kez tekrar edeceğini belirtir. Şöyle ki; örneğin bir buton var ve butona tıklayınca belirli bir elemente bir takım animasyonlar yaptırılıyor. Şayet ben bu butona birden fazla tıklarsam? İşte bu kuyruk olur. Yani animasyon bitince, aynı animasyon tekrar çalışmaya başlayacak. İşte bu kuyrukta bekleyen animasyonların sayısını öğrenmek için bu metot kullanılmaktadır.

queue( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").queue( )
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style>
    div{
        position: absolute;
        background-color: red;
        height: 50px;
        width:  50px;
    }
</style>

<div></div>
<br/><br/><br/>

<button>Start</button>
<button>Stop</button>
<br /><br /><br>
<span></span>

<script>
    $(document).ready(function (){
        $("button:first").click(function (){
            $("div").animate({"left": "+=100px"}, 1000);
            var n = $("div").queue();
            $("span").text(n.length);
        });
        $("button:last").click(function (){
            $("div").stop();
        });
    });
</script>
```

Yukarıdaki örneğimizde; div elementi, butona tıklandığında sağa doğru hareket edecektir. İşte bu butona ardı ardına tıkladığımızda; animasyon sayısı kuyruğa eklenecek ve span elementinde gösterilecektir.

9.4.5 queue(callback)

queue(callback) metodunun, queue( ) metodundan tek farkı; fonksiyon içerebiliyor olmasıdır. Yani parametre olarak tanımlanan bir fonksiyon ile değişik işlemler yaptırılabilir.

queue(callback) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").queue(callback)
```

Çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style>
    div{
        position: absolute;
        background-color: red;
```

```
        height: 50px;
        width:  50px;
    }
    .colored {
        background-color: green;
        color: white;
    }
</style>

<div></div>
<br/><br/><br/>

<button>Start</button>
<br /><br /><br>
<span></span>

<script>
    $(document).ready(function (){
        $("button:first").click(function (){
            $("div").animate({"left": "+=100px"}, 1000);
            $("div").queue(function(){
                $(this).addClass("colored");
            });
    });
</script>
```

Yukarıdaki örneğimizde; ilk animasyon bittiken sonra colored isimli CSS class'ı div elementine eklenecektir.

9.4.6 dequeue( )

dequeue( ) metodu, kuyruğu bir azaltır. Örneğin; kuyrukta bekleyen 5 animasyon varsa bunu 4'e düşürür. Dilenirse daha fazla da animasyon azaltabilir.

dequeue( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").dequeue( )
```

Basit bir örnek ile bu bu bölümü kapatalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<style>
    div{
        position: absolute;
        background-color: red;
        height: 50px;
        width:  50px;
    }
</style>

<div></div>
<br/><br/><br/>
```

```
<button>Start</button>
<button>Dequeue</button>
<br /><br /><br>
<span></span>

<script>
    $(document).ready(function (){
        var n = 0;
        $("button:first").click(function (){
            $("div").animate({"left": "+=100px"}, 1000);
            n = $("div").queue();
            $("span").text(n.length);
        });
        $("button:last").click(function (){
            $("div").dequeue();
            $("span").text(n.length);
        });
    });
</script>
```

Yukarıdaki örneğimizde; Dequeue isimli butona tıklayınca kuyrukta bekleyen animasyon sayısı bir azalacaktır.

BÖLÜM 10: AJAX

Bu bölümde AJAX konusuna derin bir giriş yapacağız. Öncelikle AJAX nedir? Bilmeyenler için bu konudan bahsedelim.

AJAX kelimesi, Asynchronous JavaScript and XML kelimelerinden oluşturulmuş bir kısaltmadır. Türkçe'ye çevirmek isterse; Eşzamansız JavaScript ve XML şeklindeki bir çeviri doğru olacaktır. AJAX, ile etkileşimli web sayfaları tasarlanması mümkün olmaktadır. Peki bu nedir? Örneğin elimizde bir web sayfası olduğunu düşünelim. Bu web sayfası bir form aracılığıyla yeni bir kullanıcı eklesin ya da bir sorgulama yapsın. Bu gibi durumlarda AJAX kullanılmadığında; geliştirilen uygulamanın mimarisine göre ya sayfa refresh (yenile) edilir veyahut yeni bir sayfada gösterim sağlanır. AJAX kullanıldığı durumda ise yeni bir sayfaya gerek kalmadan, web sayfasının bir kısmı veya tamamı yenilenmeye veya ekstra bir sayfalamaya gerek kalmadan değiştirilebilir. Yani bir anlamda sayfa; update (güncelleme) edilebilir. Ama sayfanın belli bir kısmı.

AJAX'ın nasıl çalıştığı veya avantajlar / dezavantajlar kısmından bahsetmeyeceğiz. Zira daha önce yazmış olduğum; “Classic AJAX” kitabında bu konuları detaylı olarak anlatmıştım. Bu kitap free (ücretsiz) olduğu için indirip inceleyebilirsiniz.

Herneyse, jQuery AJAX konusunda oldukça iyi. Sahip olduğu bir çok metot yardımıyla AJAX kullanımını çok basitleştiriyor. Bizde jQuery'nin nimetlerinden yararlanalım ve bu konudan bahsetmeye başlayalım.

10.1 AJAX Requests

Bu bölümde; bir AJAX request (istek) nasıl yapılır ve gelen değerler nasıl işlenir bunun üzerinde detaylı olarak durmaya çalışacağız.

10.1.1 jQuery.ajax(options)

jQuery.ajax(options) veya $.ajax(options) metodu, low level (düşük seviye veya alt seviye) AJAX işlemlerini yapmak için kullanılan metottur.

jQuery.ajax(options) meotodunun kullanımı aşağıdaki gibidir.

```
jQuery.ajax(options)
// veya
$.ajax(options)
```

options kısmında bir takım parametrelerin tanımlanması gerekir. Örneğin: url. Şimdi bu parametrelerden detaylı olarak bahsedelim.

options:
`async`
→ Type: Boolean
→ Default: true

→ Açıklama: Asynchronous / Synchronous işlemlerini belirtir. Varsayılan olarak true idir. Yani asynchronous işlem yapar.

`beforeSend`
→ Type: function
→ Açıklama: Requet (İstek) göndermeden önce, XMLHttpRequest nesnesini düzenlemek için kullanılır.

`cache`
→ Type: Boolean
→ Default: true
→ Açıklama: Browser (Tarayıcı) tarafından cache'leme yapılmasını engeller. Varsayılan olarak true idir. Yani cache'leme yaptırmaz.

`complete`
→ Type: function
→ Açıklama: İstek bittikten sonra, istekten dönen değerlerin işlenmesidir. Bir fonksiyondur. Fonksiyon içerisinde bu data'lar işlenebilir.

`contentType`
→ Type: String
→ Default: "application/x-www-form-urlencoded"
→ Açıklama: Server'a istek gönderilirken kullanılan içerik tipidir. Bir çok durum için oldukça kullanışlıdır.

`data`
→ Type: Object / String
→ Açıklama: Server'a gönderilecek olan data'lar veya parametrelerdir.

`dataType`
→ Type: String
→ Default: Intelligent Guess (xml or html)
→ Açıklama: Server'dan beklenen verilerin tipidir. Varsayılan olarak; xml ve html idir. Lakin bunlar dışında diğer data tiplerine de destek verilmektedir. Bu data tipleri şunlardır: xml, html, script, json, jsonp (json bloğu için kullanılır) ve text.

`error`
→ Type: function
→ Açıklama: İstek başarısızlığa uğradığı zaman, oluşan hatayı göstermek için kullanılan fonksiyondur.

`global`
→ Type: Boolean
→ Default: false
→ Açıklama: Bir istek için Global AJAX parametreleri çalıştırmak için kullanılır. Default olarak false idir. Bu parametreler: ajaxStart, ajaxStop v.s idir.

`ifModified`
→ Type: Boolean
→ Default: false
→ Açıklama: Eğer son yapılan istekten sonra bir değişiklik olduysa isteğe izin verir. Default olarak false değerindedir. Yani bu işleme izin verilmez.

`jsonp`
→ Type: String
→ Açıklama: Klasik olarak; callback=? yerine; {jsonp:'onJsonPLoad'} olarak kullanılmasını sağlar. Bu da server'a onJsonPLoad=? olarak gider.

`processData`
→ Type: Boolean
→ Default: true
→ Açıklama: Data bir data object olarak geçer ve bir string olarak işlenir. Şayet DOMDocuments veya işlenmemiş data kullanılmak istenirse; true olarak işaretlenmesi gerekir.

`success`
→ Type: function
→ Açıklama: Yapılan istek başarılıysa; yapılacak olan işlemler tanımlanır. Örneğin gelen data'lar işlenebilir veya herhangi bir mesaj verdirilebilir.

`timeout`
→ Type: number
→ Açıklama: Her bir istek için bir zaman belirlenebilir. Bu zaman içerisinde istek başarılı olmamışsa bir başarılı olmamış kabul edilir ve istek sonlandırılır.

`type`
→ Type: String
→ Açıklama: HTTP istek tipini belirtir. En sık kullanılanlar: GET ve POST'tur. Lakin bunlar dışındaki: PUT ve DELETE gibi istek tipleri de kullanılabilir. Fakat bazı tarayıcılar bu istek tiplerine destek vermemektedir.

`url`
→ Type: String
→ Açıklama: İstek yapılacak olan url belirtilir.

options konusundan bahsettik. Şimdi yapacağımız örnekler ile bu anlattıklarımızı daha iyi anlamaya çalışalım ve nasıl kullanıldıklarını görelim.

Örneklere başlamadan önce şunu belirtmek isterim. Bazı tarayıcılar herhangi bir server'da olmayan dosyaların AJAX ile kullanımına izin vermemektedir. Firefox'ta böyle bir engel yoktur. Lakin siz de benim gibi Chrome kullanıyorsanız bir server'a ihtiyaç duyacaksınız. O yüzden bir server olmalı ve localhost'ta çalışabilmelisiniz. Ben PHP üzerinde çalıştığım için: Apache üzerinde çalışacağım. Bunu baştan belirteyim.

İlk örneğimize bakalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<div></div>

<script>
    $(document).ready(function (){
        $.ajax({
            url: "test.txt",
            type: "GET",
            cache: false,
            success: function (data){
                $("div").text(data);
            }
        });
    });
</script>
```

Yukarıdaki örneğimizde neler yaptık? Örneğimizde; test.txt dosyasında olan içeriği almak için bir script (betik) yazdık. GET metodunu kullandık ve cache'leme olmasın dedik. success parametresinde ise; gelen data değerini, div elementine ver dedik. Oldukça basit değil mi?

Başka bir örneğimize bakalım.

```
<script>
    $(document).ready(function (){
        $.ajax({
            url: "reserve.js",
            type: "GET",
            dataType: "script"
        });
    });
</script>
```

Yukarıdaki örneğimizde; reserve.js isimli JavaScript dosyasını yükledik. Yani sayfa tam yüklendiği anda bu .js uzantılı dosya da çağrılacak ve çalıştırılacaktır.

Aşağıdaki örneğimize bakalım.

```
<div></div>

<script>
    $(document).ready(function (){
        $.ajax({
            url: "record.php",
            type: "POST",
            data: "name=emrecan&surname=oztas",
            success: function (data){
                $("div").text(data);
```

```
            }
        });
    });
</script>
```

Yukarıdaki örneğimizde; record.php isimli PHP dosyasına, POST ile name ve surname parametrelerini gönderdik. Gelen değeri de div elementine yazdıracağız. Peki bu örnekte amacımız neydi? O halde bir de record.php dosyasına bakalım.

```
<?php

$name = $_POST['name'];
$surname = $_POST['surname'];

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "example";

$conn = new mysqli($servername, $username, $password, $dbname);
mysqli_set_charset($conn, "utf8");

if ($conn->connect_error) {
    echo("Connection error!");
}

$sql = "INSERT INTO user (user_name, user_surname) VALUES('$name',
'$surname')";

if ($conn->query($sql) === TRUE) {
    echo("Success!");
} else {
    echo("Failed!");
}

        $conn->close();

?>
```

Yukarıdaki PHP dosyamızdan da anlaşılacağı üzere; name | surname değerlerini veri tabanına doğrudan kaydettik.

Son olarak aşağıdaki örneğimize bakalım.

```
<div></div>

<script>
    $(document).ready(function (){
        $.ajax({
            url: "header.html",
```

```
            cache: false,
            success: function (data){
                $("div").append(data);
            }
        });
    });
</script>
```

Yukarıdaki örneğimizde; header.html HTML dosyasını, içinde bulunduğumuz sayfaya ekledik. Yani bunu örneğin; PHP'deki include fonksiyonu gibi düşünebilirsiniz.

10.1.2 load(url, [data], [callback])

load(url, [data], [callback]) metodu, herhangi bir HTML dosyasında bulunan herhangi bir alanı almak için kullanılır. Örneğin; herhangi bir HTML dosyasındaki link, header, footer veyahut istediğiniz herhangi bir alanı alıp kendi sayfanızda kullanabilirsiniz.

load(url, [data], [callback]) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").load(url, [data], [callback])
```

Parametre olarak belirtilen: *url: * Uzaktaki HTML dosyasının adresi. *[data]: * HTML dosyasından alınacak kısım. *[callback]: * Fonksiyon. özellikleridir.

Çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<div></div>

<script>
    $(document).ready(function (){
        $("div").load("load.html div>span");
    });
</script>
```

load.html isimli HTML dosyasında; div>span altındaki içerik getirilip, kendi sayfamızdaki div elementine yazdırılacaktır.

10.1.3 jQuery.get(url, [data], [callback])

jQuery.get(url, [data], [callback]) metodu, temel olarak herhangi bir dosyaya; GET metodu ile request (istek) yapılmasını sağlar.

jQuery.get(url, [data], [callback]) metodu`nun kullanımı aşağıdaki gibidir.

```
jQuery.get(url, [data], [callback])
// veya
$.get(url, [data], [callback])
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<script>
    $(document).ready(function (){
        $.get("get.php"){
        });
    });
</script>
```

Yukarıdaki örneğimizde; get.php isimli dosyaya, GET ile basit bir istek yaptık. get.php isimli dosyada örneğin; bir sayaç koyabiliriz ve her istekte bu sayaç değeri artabilir. Bu ve buna benzer bir çok senaryo geliştirilebilir.
Başka bir örnek yapalım.

```
<script>
    $(document).ready(function (){
        $.get("get.php", {name: "james", surname: "sawyer"});
    });
</script>
```

Yukarıdaki örneğimizde; get.php dosyasına, parametre olarak name | surname gönderdik. get.php dosyası içerisinde bu parametreleri alabilir ve değişik işlemler yapabiliriz. Örneğin: veri tabanına kaydetmek gibi.

Bir diğer örneğe bakalım.

```
<script>
    $(document).ready(function (){
        $.get("test.php", {name: "james", surname: "sawyer"},
function (data){
            alert(data);
        });
    });
</script>
```

Yukarıdaki örneğimizde; test.php isimli PHP dosyasına, name | surname parametrelerini gönderdik. Bir de fonksiyon yazdık. Bu fonksiyon ile dönen değeri alert metodu ile gösterdik. Aslında burada yaptığımız basit bir kayıt işlemi idi.

10.1.4 jQuery.getJSON(url, [data], [callback])

jQuery.getJSON(url, [data], [callback]) metodu, GET metodu kullanılarak herhangi bir JSON dosyasını istek yapar.

jQuery.getJSON(url, [data], [callback]) metodunun kullanımı aşağıdaki gibidir.

```
jQuery.getJSON(url, [data], [callback])
// veya
```

```
$.getJSON(url, [data], [callback])
```

Burada parametre olarak belirtilen; *url: * JSON dosyasının yolu. *data: * JSON dosyasına iletilecek herhangi data'lar. *callback: * Fonksiyon. özellikleridir.

Basit bir örnek yapalım.

```
$.getJSON("json.js", function(json){
  alert(json.user[1].name);
});
```

Yukarıdaki örneğimizde; json.js isimli bir dosyaya istek yaptık. JSON dosyasından gelen değerleri de alert( ) metodu ile yazdırdık. JSON dosyamızın yapısı aşağıdaki gibidir.

```
{
    "user":[
        {"name": "emrecan", "surname": "oztas"},
        {"name": "james", "surname": "sawyer"}
    ]
}
```

Yani ekran çıktımız da: james olacak.

10.1.5 jQuery.getScript(url, [callback])

jQuery.getScript(url, [callback]) metodu, herhangi bir JavaScript dosyasını yüklemek için daha doğrusu çağırmak için kullanılır.

jQuery.getScript(url, [callback]) metodunun kullanımı aşağıdaki gibidir.

```
jQuery.getScript(url, [callback])
// veya
$.getScript(url, [callback])
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<script>
    $(document).ready(function (){
        $.getScript("validation.js");
    });
</script>
```

Yukarıdaki örneğimizde; validation.js isimli JavaScript dosyasını sayfamıza yükledik. Aslında bu güzel bir yöntem. Örneğin; birden fazla yükleyeceğiniz JavaScript dosyası varsa bunların sayfa tam yüklendiği anda çalışmasını istiyorsanız bu yöntem oldukça kullanışlı olabiliyor.

Burada örneğin; callback özelliğini de kullanarak, belirtilen dosyanın yüklenip yüklenmediğini kontrol edebilirsiniz.

```
<script>
    $(document).ready(function (){
        $.getScript("validation.js", function(){
          alert("Script loaded and executed.");
        });
    });
</script>
```

10.1.6 jQuery.post(url, [data], [callback])

jQuery.post(url, [data], [callback]) metodu, POST metodunu kullanarak; belirtilen dosyaya istek yapılmasını sağlar. Yani bir anlamda GET metoduyla aynı mantıkta lakin kullandıkları metotları farklı.

jQuery.post(url, [data], [callback]) metodunun kullanımı aşağıdaki gibidir.

```
jQuery.post(url, [data], [callback])
// veya
$.post(url, [data], [callback])
```

Basit bir örnek ile konumuza devam edelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<script>
    $(document).ready(function (){
        $.post("post.php"){
        });
    });
</script>
```

Yukarıdaki örneğimizde; post.php dosyasına, POST metodu ile basit bir istek gönderdik. Bunun dışında; çeşitli parametreler tanımlanarak ve istenirse bir fonksiyon tanımlanarak çeşitli işlemler yapılabilir.

10.2 AJAX Events

Bu bölümde, AJAX Events (Olaylar) üzerinde durmaya çalışacağız. Örneğin; bir istek tamamlandığında veya herhangi bir hata olduğunda bunları yakalayıp gerekli işlemleri yapabileceğiz.

10.2.1 ajaxComplete(callback)

ajaxComplete(callback) metodu, şayet AJAX işlemi başarıyla tamamlanmışsa çalışmaya başlar.

ajaxComplete(callback) metodunun kullanımı aşağıdaki gibidir.

```
$(document).ajaxComplete(callback)
```

Burada; document yerine sayfa üzerinde bulunan herhangi bir elementi de kullanabilirsiniz.

Çok basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<script>
    $(document).ready(function (){
        $.get("get.php", function(){
            $(document).ajaxComplete(function(){
               $("div").text("Request Complete!");
             });
        });
    });
</script>
```

Yukarıdaki örneğimizde; get.php dosyasına bir istek yaptık. Bu dosya var ve yolu doğru. Dolayısıyla metodumuz çalışacak ve div elementine Request Complete! yazacaktır.

10.2.2 ajaxError(callback)

ajaxError(callback) metodu, istek yapılırken herhangi bir hata olması durumunda çalışmaya başlayacaktır. Örneği; istek yapılacak olan ùrl'i yanlış vermiş olabilirsiniz.

ajaxError(callback) metodunun kullanımı aşağıdaki gibidir.

```
$(document).ajaxError(callback)
```

Konumuzu daha iyi anlamak için bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<div></div>

<script>
    $(document).ready(function (){
        $("div").load("testt.html div>span");
            $(document).ajaxError(function(){
               $("div").append("Request Failed!");
             });
        });
</script>
```

Yukarıdaki örneğimizde; testt.html HTML dosyasının div>span özelliğini almak istedik. Lakin elimizde; testt.html isminde bir HTML dosyası yok. Dolayısıyla metodumuz çalışmaya başlayacak ve Request Failed! hatasını div elementine atayacaktır.

10.2.3 ajaxSend(callback)

ajaxSend(callback) metodu, istek başarıyla gönderildiğinde çalışmaya başlar.

ajaxSend(callback) metodunun kullanımı aşağıdaki gibidir.

```
$(document).ajaxSend(callback)
```

Örneğimiz de aşağıdaki gibi olsun.

```
<script>
    $(document).ready(function (){
        $.get("get.php");
        $(document).ajaxSend(function(){
            console.log("Yes!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; get.php isimli dosyaya istediğimiz iletildiği anda metodumuz da çalışmaya başlayacaktır.

10.2.4 ajaxStart(callback)

ajaxStart(callback) metodu, istek başladığı zaman çalışmaya başlar.
ajaxStart(callback) metodunun kullanımı aşağıdaki gibidir.

```
$(document).ajaxStart(callback)
```

Örneğimiz de aşağıdaki gibi olsun.

```
<div></div>
<span></span>

<script>
    $(document).ready(function (){
        $("div").load("test.html");
        $("span").ajaxStart(function(){
            $(this).text("Request Start!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; istek başladığı anda; span elementine, Request Start! mesajı atanacaktır.

10.2.5 ajaxStop(callback)

ajaxStop(callback) metodu, tüm AJAX istekleri bittiği zaman çalışır.

ajaxStop(callback) metodunun kullanımı aşağıdaki gibidir.

```
$(document).ajaxStop(callback)
```

Basit bir örnek yapalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<div></div>
<span></span>

<script>
    $(document).ready(function (){
        $("div").load("test.html");
        $("span").ajaxStop(function(){
            $(this).text("Request Stop!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; istek bittiği anda; Request Stop! değeri div elementine atanacaktır.

10.2.6 ajaxSuccess(callback)

ajaxSuccess(callback) metodu, yapılan istek başarıyla gerçekleştirilmiş ise çalışmaya başlar.

ajaxSuccess(callback) metodunun kullanımı aşağıdaki gibidir.

```
$(document).ajaxSuccess(callback)
```

Örneğimiz de aşağıdaki gibi olsun.

```
<div></div>
<span></span>

<script>
    $(document).ready(function (){
        $("div").load("test.html");
        $(document).ajaxSuccess(function(){
        $("span").text("Succesful!");
        });
    });
</script>
```

Yukarıdaki örneğimizde; istek başarılı bir şekilde sonuçlandığında; span elementine, Successful! ifadesi atanacaktır.

10.3 Misc

Bu bölümde, jQuery'nin gözden kaçan veya püf noktalar diyebileceğimiz çeşitli özellikleri üzerinde duracağız. Bu özellikler gerçekten işlerinizi kolaylaştıracak. Hazırsanız, o halde hemen başlayalım.

10.3.1 jQuery.ajaxSetup(options)

jQuery.ajaxSetup(options) metodu, AJAX işlemleri yapmadan önce belirli parametrelerin tanımlanmasına izin verir. Bir sayfada bu metot ile parametrelerin tanımlanmasından sonra tekrar tanımlanmasına gerek kalmaz. Yani bir anlamda AJAX ile yapacağımız işlemleri tanımlayacağız.

jQuery.ajaxSetup(options) metodunun kullanımı aşağıdaki gibidir.

```
jQuery.ajaxSetup({
    // options
})
// veya
$.ajaxSetup({
    // options
});
```

Bir örnek yapalım ve metodumuzun nasıl kullanıldığından bahsedelim. Örneğimiz aşağıdaki gibi olacaktır.

```
<script>
    $.ajaxSetup({
        url: "record.php",
        type: "POST",
        cache: false
    });
</script>

<script>
    $(document).ready(function (){
        $.ajax({
            data: "name=emrecan&surname=oztas"
        });
    });
</script>
```

Yukarıdaki örneğimizde; ilk olarak, yapacağımız AJAX işlemlerini jQuery.ajaxSetup(options) metodu ile tanımladık. Artık sayfamızda özgürüz. Daha sonra da record.php dosyasına data'mızı parametre olarak geçtik.

10.3.2 serialize( )

serialize( ) metodu, belirtilen elementleri serileştirir. Yani onları data şekline çevirir. Örneğin elimizde bir form elementi ve bu form elementinin altında bir takım alanların olduğu varsayalım. İşte bu metodumuz ile bunları data haline dönüştürebiliriz.

serialize( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").serialize( )
```

Bir örnek yapalım. Ne demek istediğimi o zaman daha iyi anlayacağınıza eminim. Örneğimiz aşağıdaki gibi olacaktır.

```
<form>
    <input type="text" name="name"/>
    <input type="text" name="surname"/>
</form>
<button>Serialize ()</button>

<script>
    $(document).ready(function (){
        $("button").click(function (){
            var str = $("form").serialize();
            console.log(str);
        });
    });
</script>
```

Yukarıdaki örneğimizde; button elementine tıkladığımızda, form elementi içerisinde bulunan input="text" alanları serileştirilecektir. Örneğin name alanına emrecan ve surname alanına oztas girdiğimizi ve butona tıkladığımızı varsayarsak, ekran çıktımız şöyle olacaktır:

```
name=emrecan&surname=oztas
```

Görüldüğü gibi süper bir metot.

10.3.3 serializeArray( )

serializeArray( ) metodu da serialize( ) metoduna benzer bir yapıdadır. Hatta kardeşidir de diyebiliriz. Lakin tek farkla! serializeArray( ) metodu form elementlerini JSON verisine çevirir.
serializeArray( ) metodunun kullanımı aşağıdaki gibidir.

```
$("selector").serializeArray( )
```

Bir örnek yapalım ve ekran çıktımıza bakalım. Örneğimiz aşağıdaki gibi olacaktır.

```
<form>
    <input type="text" name="name"/>
    <input type="text" name="surname"/>
</form>
<button>serializeArray ()</button>
<span></span>

<script>
    $(document).ready(function (){
        $("button").click(function (){
            var str = $("form").serializeArray();
```

```
            jQuery.each(str, function(i, str){
                $("span").append(str.value + " ");
            });
        });
    });
</script>
```

Yukarıdaki örneğimizde; form elementi altındaki, input="text" elementleri serializeArray() metodu ile JSON verisine çevirdik ve span elementinden bu değerleri yazdırdık. Örneğin name alanına emrecan ve surname alanına oztas girdiğimizi ve butona tıkladığımızı varsayarsak, ekran çıktımız şöyle olacaktır: emrecan oztas.

KAYNAKLAR

[1]. https://learn.jquery.com/
[2]. http://www.w3schools.com/jquery/
[3]. https://www.codecademy.com/learn/jquery
[4]. http://tutorialzine.com/tag/jquery/
[5]. http://www.tutorialspoint.com/jquery/
[6]. http://www.vogella.com/tutorials/JQuery/article.html