jQuery dünyasına adım atıyoruz

javaScript kütüphaneleri kendilerine çok sağlam bir yer edindiler. Gerek kod yazma sürecini azaltmaları, gerek tarayıcı farklılıklarından doğan pek çok sorunla boğuşmamızı engellemeleri ve nesne tabanlı yapıları itibariyle javaScript kütüphaneleri çabuk benimsendiler. Prototype.js'nin tetiklediği bu süreçte yüzlerce javaScript kütüphanesi duyuruldu. Dolayısıyla bunların arasından bir seçim yapmak gerekiyor. Bugüne kadar benim seçimim Prototype.js idi ama artık çok daha isabetli bir seçim olduğuna inandığım jQuery ile yoluma devam ediyorum.

jQuery nedir ve kimler tarafından kullanılıyor?

jQuery hem javaScript hem ajax hem de efekt kütüphanesi olarak kullanılabilen bir framework'dür. 2006'nın Ocak ayında bir javaScript gurusu olan John Resig tarafından duyurulmuş. Şu anda ise 15 kişilik bir ekip tarafından gelişimi sürdürülüyor. Lisans konusuna da kısaca değinmek gerekirse, MIT veya GPL lisansının şartlarına uyduğunuz sürece kendi uygulamalarınızda kullanabiliyorsunuz. Bu konuda jQuery'nin resmi web sitesi olan www.jquery.com'dan daha fazla bilgi alabilirsiniz. Hem bu web sitesinden jQuery ile ilgili dokümantasyona, eğitsellere, eklentilere ve sorularınızı yazabileceğiniz foruma da ulaşabilirsiniz.

jQuery çok geniş bir kullanıcı kitlesine sahiptir. Bunlardan da biraz örnek vermek istiyorum. Sosyal haber sitesi Digg, açık kaynak kodlu projelere ev sahipliği yapan SourceForge, blog sitelerini analiz eden Technorati, RSS kaynaklarımızı yönettiğimiz FeedBurner ve birçoğumuzun kullandığı blog yazılımı WordPress bu örneklerden yalnızca birkaçıdır. Daha fazlası için jQuery Kullanan Siteler sayfasına bakabilirsiniz.

jQuery'yi nasıl indireceğim ve kullanacağım?

jQuery, sıkıştırılmış ve sıkıştırılmamış olmak üzere iki farklı şekilde dağıtılmaktadır. Sıkıştırılmamış sürümü yaklaşık olarak 61 KB iken, sıkıştırılmış sürümü ise yaklaşık 21 KB. Gördüğünüz gibi dosya boyutları hem javaScript, hem ajax, hem de efekt kütüphanesi için yeterince hafif. Eğer ki jQuery'yi oluşturan kodlara bakmak ve kodlar üzerinde değişiklik yapmak gibi bir düşünceniz yoksa her zaman için sıkıştırılmış sürümünü indirmenizi tavsiye ediyorum.

Eğer indirdiyseniz artık sıra onu kullanmaya gelmiş demektir. jQuery tek bir dosyadan oluşuyor. Bu dosyayı da web sayfanıza aşağıdaki örnekte olduğu gibi dâhil etmelisiniz. Web sayfanıza dahil ettikten sonra artık jQuery'nin bizlere sunmuş olduğu özellikleri kullanmaya başlayabilirsiniz. Özellik demişken şimdi de jQuery'nin özelliklerine kısaca göz atalım 😉

```
<script type="text/javascript" src="jquery-latest.js"></script>
```

jQuery'nin birkaç özelliği

Basit kullanım: Gerçekten jQuery'nin son derece basit bir kullanımı var. Yapmak istediğiniz pek çok işlemi çoğu zaman "tek bir satırda" halledebiliyorsunuz. Örneğin web sayfanızdaki

bütün `div` etiketlerini `$('div')` kodu ile yakalayabiliyorsunuz. Yok ben sadece `class` özniteliği "baslik" olan `div` etiketlerini yakalayacağım derseniz `$('div.baslik')` kodunu kullanmanız yetiyor. Yani jQuery'nin sloganında da yazdığı gibi: "daha az yazın, daha fazlasını yapın" 😉

Zincirlenebilirlik: jQuery'nin Sihri *(The Magic of jQuery)* olarak isimlendirilmiş bu özellik sayesinde çok kısa kodlar yazabilirsiniz. jQuery içerisindeki methodları birbirine zincirleyerek tek satırda birden fazla işlemi yapabilirsiniz. Örneğin şöyle bir kod ile web sayfanızdaki tüm linkleri önce yakalarsınız, sonra bir `class` ataması yaparsınız ve son olarak da `onclick` olayına bir fonksiyon eklersiniz:

```
$('a').addClass('deneme').click(fonksiyon);
```

Eklentiler: jQuery'nin bir başka mükemmel özelliği de eklentileridir. Eklentileri, belli görevleri yapan ve jQuery kütüphanesi üzerinden geliştirilmiş kod parçacıkları olarak tanımlayabiliriz. Örneğin web sitenizde "sekme (tab) menüler" kullanmak isterseniz Tabs eklentisi, web sayfanıza bir mp3 player yerleştirmek isterseniz jMP3 eklentisi ya da bir dosya yükleme uygulaması sunmak isterseniz jqUploader eklentisi ihtiyacınızı karşılamaya hazırdır. Buna benzer yüzlerce jQuery eklentisi vardır ve vakti geldikçe de bu konuda tanıtımlar yapacağım 😉

Uyumluluk: jQuery, şu anda en çok kullanılan web tarayıcılarında çalışabilmektedir. Internet Explorer 6.0+, Firefox 1.5+, Safari 2.0+ ve Opera 9.0+ ile sorunsuzca jQuery kullanabilirsiniz. Bunun haricinde Prototype.js gibi başka kütüphaneler ile birlikte de uyum içerisinde kullanabilirsiniz. Bu konuda jQuery'yi Diğer Kütüphaneler İle Birlikte Kullanmak sayfası size yardımcı olacaktır.

Efektler: jQuery, script.aculo.us gibi tamamen bir efekt kütüphanesi olmamasına rağmen yine de sık kullanılan efektleri sunuyor. Örneğin fadeOut efekti sayesinde bir nesnenin görünürlüğünü yavaş yavaş azaltırken, show efekti sayesinde daha önceden gizlenmiş bir nesneyi görünür kılabiliyorsunuz. En önemlisi de bu efektleri kullanmak çok basittir. Alttaki örnek kodda, "hide" efektini kullanarak "myDiv" isimli nesneyi yavaşça gizlemiş oluyoruz.

```
$("#myDiv").hide("slow");
```

AJAX: jQuery diğer konularda olduğu gibi AJAX konusunda da kullanıcılarına kolaylıklar sağlıyor. Böylece web sayfanızın tamamı yüklenmeden, başka bir web sayfası ile etkileşime geçebiliyorsunuz. AJAX işlemi başlamadan önce veya bittikten sonra bir olayı tetiklemek, sunucudan gelen veri türüne göre (xml, json) işlem yapmak, belli zaman aralıklarında tekrar tekrar istekler yollamak v.s. jQuery ile yapabileceklerinizin küçük bir bölümü. Örneğin alttaki örnek kod ile AJAX kullanarak *form.php* dosyasına *Erhan* ve *23* verilerini *post* etmiş oluyoruz.

```
$.ajax({
type: "POST",
url: "form.php",
data: "isim=Erhan&yas=23"
});
```

Bu da basitleştirilmiş ve kolaylaştırılmış versiyonu:

```
$.post( "form.php", {isim:"Erhan", yas:"23"} );
```

Sonuç…

Evet, basite indirgeyerek jQuery kütüphanesinden bahsetmeye çalıştım. Bundan böyle jQuery ile ilgili haberleri, kaynakları, örnek kodları ve kullanışlı eklentileri paylaşmaya gayret göstereceğim. Bu sayede Türkçe içeriğe de bir katkım olacağını düşünüyorum. Zira jQuery ile ilgili maalesef kendi dilimizde doğru düzgün bir kaynak bulunmuyor. Kendi web sitem haricinde bir de javam.org sitesinde de jQuery ile ilgili birkaç yazı bulabilirsiniz. Eğer sizin de bildiğiniz başka Türkçe jQuery yazıları varsa lütfen yorum kısmında belirtiniz

jQuery ve Seçiciler

Daha önce jQuery dünyasına adım atmıştık biliyorsunuz. Şimdi de bu dünyayı yakından tanımaya başlayalım. Öncelikle **seçiciler** konusuyla başlamak istiyorum. Bence jQuery için en önemli konu budur. Neden mi önemli? Çünkü web sayfalamızdaki elemanlara seçiciler sayesinde erişebiliyoruz. Örneğin bir yazının rengini değiştireceksiniz. Peki ama bu yazı web sayfanızın neresinde? Bu yazıyı jQuery'ye nasıl göstereceksiniz ve işleme almasını sağlayacaksınız? İşte **seçiciler** konusu bu yüzden önemli

Seçiciler

jQuery, web sayfamızdaki elemanlara erişmek için birkaç değişik yöntem kullanmakta. Ben bu yöntemleri 5 başlık altında grupladım.

1. Etiket seçicileri
2. Öznitelik seçicileri
3. Css seçicileri
4. Form seçicileri
5. Özel seçiciler

Bu seçicilerinin hepsini `$('seçici')` ifadesinde olduğu gibi kullanmamız gerekiyor. Şimdi bu seçicileri ayrı ayrı başlıklar altında inceleyelim.

1. Etiket seçicileri

Web sayfamızda yer alan elemanları HTML etiketlerini kullanarak oluştururuz. Parağraf elemanları oluşturmak için `P` etiketini, resim elemanları oluşturmak içinse `IMG` etiketini kullanırız. İşte bu elemanlara jQuery ile erişmek için, oluştururken kullandığımız etiket isimlerini **aynen** kullanıyoruz.

Örneğin web sayfamızdaki bütün parağraf elemanlarını seçmek isteyelim.

```
$('p')
```

Yukarıda yalnızca bir tek etiket seçimi yaptık o da P etiketi idi. Ama biz aynı anda hem P etiketlerini hem de IMG etiketlerini seçmek isteyebiliriz. Birden fazla etiket seçimi yapacaksak etiketler arasına virgül (,) koymamız gerekiyor. Aynen alttaki gibi:

```
$('p,img')
```

Tekli ve Çoklu etiketlerin nasıl seçildiklerini öğrendikten sonra sıra geldi **içiçe** olan etiketleri seçmeye. Örneğin web sayfamızdaki bütün P etiketlerinin içerisinde ne kadar IMG etiketi varsa seçelim.

```
$('p img')
```

Bu son kodda dikkat edeceğimiz şey, etiketler arasında boşluk kullandığımızdır.

2. Öznitelik seçicileri

Her HTML etiketi **attribute** adı verilen özniteliğe sahiptir. Örneğin bir link oluşturmak için **A** etiketinin **href** özniteliğini kullanırız. Hatırlamak için hemen bir örnek görelim:

```
<a href="www.eburhan.com"> link </a>
```

Bu örnekte "href" ifadesi bir özniteliktir. İşte jQuery ile web sayfamızdaki elemanlara özniteliklerini kullanarak da erişebiliyoruz. Bize en çok gerekli olabilecek öznitelik seçicilerine kısaca gözatalım.

- **a[href]** ile href özniteliği olan a etiketlerini seçiyoruz.
- **a[href=www.eburhan.com]** ile href özniteği "www.eburhan.com" ile tam olarak eşleşen a etiketlerini seçiyoruz.
- **a[href^=www]** ile href özniteliği "www" ile başlayan a etiketlerini seçiyoruz.
- **a[href$=com]** ile href özniteliği "com" ile biten a etiketlerini seçiyoruz.
- **a[href*=eburhan]** ile href özniteliği içerisinde "eburhan" geçen a etiketlerini seçiyoruz.

Öznitelik seçicilerini tek başına kullanabildiğimiz gibi birleştirerek de kullanabiliyoruz. Alttaki örnekte href özniteliği "www" ile başlayan VE yine href özniteliği "com" ile biten a etiketlerini seçmiş oluyoruz:

```
$('a[href^=www][href$=com]‘)
```

3. Css seçicileri

Css kodlarken kullandığımız yöntemleri, jQuery ile birlikte kullanabiliyoruz. Örneğin bir elemanın **id** özniteliği **metin** olsun. Bu elemanı seçmek için şöyle bir kod kullanıyoruz:

```
$('#metin')
```

Şimdi de **class** özniteliği **maviMetin** olan elemanları seçelim.

```
$('.maviMetin')
```

Bir de etiket seçicileri ile css seçicilerini birarada kullanalım. Örneğin web sayfamızdaki bütün DIV etiketlerini seçelim fakat bu DIV etiketlerinin **class** özniteliği **maviMetin** olmak zorunda olsun. İşte bu işi yapan kod:

```
$('div.maviMetin')
```

Elemanların class özniteliğine geri dönelim. Biliyorsunuz ki bir elemana birden fazla class değeri atanabiliyor. Örneğin `<div class="siyah koyu buyuk">` gibi… Böyle bir durumda class değerlerini biteşik yazarak seçim yapıyoruz.

```
$('div.siyah.koyu.buyuk')
```

Class özniteliğinde belli bir değeri içermeyen elemanları nasıl seçebiliriz ona bakalım. Örneğin class özniteliğinde **koyu** değeri bulunmayan elemanları seçmek istersek **not** ifadesini kullanmalıyız.

```
$('div:not(.koyu)')
```

Benim anlatacağım css seçileri şimdilik bu kadar (:

4. Form seçicileri

jQuery, form elemanları için birkaç özel seçici tanımlamıştır. Form seçicileri ile form elemanlarına çok kolay bir şekilde erişebiliyoruz.

- **:input**
  bütün form elemanları seçer
- **:text**
  sadece metin alanlarını seçer *(type="text")*
- **:file**
  sadece dosya alanlarını seçer *(type="file")*
- **:password**
  sadece parola alanlarını seçer *(type="password")*
- **:radio**
  sadece radyo düğmelerini seçer *(type="radio")*
- **:checkbox**
  sadece onay kutularını seçer *(type="checkbox")*
- **:submit**
  sadece gönder düğmelerini seçer *(type="submit")*
- **:image**
  sadece form resimlerini seçer *(type="image")*
- **:reset**
  sadece sıfırlama düğmelerini seçer *(type="reset")*
- **:button**
  sadece normal düğmeleri seçer *(type="button")*
- **:hidden**
  sadece gizlenmiş alanları seçer *(type="hidden")*

Form seçicilerinin nasıl kullanıldığına bakalım. Örneğin web sayfamızdaki **id** özniteliği **iletisim** olan bir formun bütün metin alanlarını seçelim. İşte kodumuz:

```
$("#iletisim :input")
```

Peki hem metin alanlarını hem de onay kutularını seçmek isteseydik?

```
$("#iletisim :input :checkbox")
```

Bunlara ek olarak jQuery'de 4 tane daha form seçicisi bulunuyor. Bu ek seçiciler özel durumlar için hazırlanmışlar. Nedir bu özel durumlar? Örneğin formunuzda 10 tane onay kutusu olsun. Ziyaretçinin bunlardan kaç tanesini fare ile işaretlediğini öğrenmek için alttaki *:checked* seçicisini kullanabiliriz.

- **:enabled**
  etkin olan form elemanlarını seçer
- **:disabled**
  devre dışı bırakılmış form elemanlarını seçer
- **:checked**
  işaretlenmiş olan onay kutularını seçer
- **:selected**
  bir açılır menünün (dropdown) seçilmiş değerini alır

5. Özel seçiciler

jQuery bazı elemanlara daha kolay erişebilmemiz için özel seçiciler sunmuştur. Nasıl ki form elemanlarına erişmek için özel form seçicileri kullanıyorsak, diğer elemanlar için de bazı özel seçiciler mevcut.

:even
seçilen elemanları 0′dan başlayarak numaralandırır, daha sonra bu numaralardan **çift** olanları seçer. Örneğin 5 tane TR etiketi olsun. Bu özel seçici sayesinde 0, 2 ve 4 nolu TR etiketlerini seçebiliriz.

:odd
seçilen elemanları 0′dan başlayarak numaralandırır, daha sonra bu numaralardan **tek** olanları seçer. Örneğin 5 tane TR etiketi olsun. Bu özel seçici sayesinde 1 ve 3 nolu TR etiketlerini seçebiliriz.

```
$('tr:even') // çift olanlar gelir
$('tr:odd') // tek olanlar gelir
```

:eq(2)
Kaçıncı sıradaki elemanı seçeceğimizi belirtir. Örneğin 5 tane DIV etiketi var ama biz 3. sıradakini seçmek istiyoruz. Öyleyse **:eq(2)** ifadesini kullanmalıyız.

:gt(3)
Greater Than demektir yani birşeyden daha büyük. 5 tane DIV etiketi var ama biz sırası 3′ten büyük olanları seçmek istiyoruz.

:lt(3)
Less Than demektir yani birşeyden daha küçük. 5 tane DIV etiketi var ama biz sırası 3′ten daha küçük olanları seçmek istiyoruz.

```
$('div:eq(2)') // 3'üncüsünü seç
$('div:gt(3)') // 3'den büyük olanları seç (4,5)
$('div:lt(3)') // 3'den küçük olanları seç (1,2)
```

:first
Seçmek istediğimiz eleman birden fazla ise "baştakini" seçmiş oluyoruz. Yani :eq(0) seçicisi gibi.

:last
Seçmek istediğimiz eleman birden fazla ise "sondakini" seçmiş oluyoruz.

:contains('X')
İçerisinde "X" ifadesi geçen elemanları seçer. Büyük-küçük harfe duyarlıdır. Örneğin içerisinde "erhan burhan" ifadesi geçen tüm paragrafları alttaki gibi seçebiliyoruz.

```
$(" p:contains('erhan burhan') ")
```

:visible
Görülebilir olan yani gizlenmemiş olan elemanları seçebilmemizi sağlar. Örneğin css'in bir özelliği olan `display:none` ile gizlenmiş elemanlara bu seçici ile ulaşamazsınız.

:hidden
Gizlenmiş olan elemanları seçer. Bir üstteki *:visible* seçicisi ile ters mantıktan çalışır.

```
$('div:hidden') // gizlenmiş div elemanlarını seç
```

Sonuç…

Burada anlattığım 5 seçiciden ziyade bir de XPath Seçicileri bulunuyor. Fakat XPath seçicileri, jQuery'nin 1.2 sürümüyle birlikte çekirdek kütüphaneden çıkarıldı ve ayrı bir eklenti olarak sunulmaya başlandı. XPath seçicilerini kullanmak isterseniz ilgili eklentiyi kurmanız gerekecektir.

Yazıyı bitirirken de şunu önemle vurgulamak istiyorum. Ben burada en çok kullanılan seçicilerden bahsettim. Oysa ki jQuery'nin daha fazla seçicisi var. O yüzden mutlaka bu sayfaya gözatmanızı öneririm. Bir sonraki yazımda buradaki seçicileri kullanarak nasıl işlemler yapılabileceğini anlatacağım 😉

jQuery ve Css işlemleri

Önceki JQuery ve Seçiciler konusunda web sayfamızdaki elemanlara nasıl erişebileceğimizi ve nasıl işleme tabi tutabileceğimizi görmüştük. Artık daha somut işlemler yapmaya geçebiliriz. İlk önce jQuery ile CSS işlemlerin nasıl yapıldığını inceleyelim. Bakalım jQuery'nin bu konuda bize sağladığı kolaylıklar nelermiş…

jQuery'nin CSS fonksiyonları

jQuery'nin css fonksiyonlarını 3 alt bölüme ayırabiliriz.

1. Css özellikleri *(property)*
2. Css sınıfları *(class)*
3. Css konumlandırmaları *(offset)*

Bu fonksiyonları kullanarak web sayfamızdaki yazıların rengini değiştirebilir, resimlere çerçeve ekleyebilir veya bir elemanın css özelliklerini başka bir elemana aktarabiliriz. Anlatımlara geçmeden önce de yazı boyunca kullanacağımız örneğe gözatalım:

```
<div class="lorem">
  Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore
  magna aliqua. Ut enim ad minim veniam, quis nostrud
  exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat. Duis aute irure dolor in reprehenderit in
  voluptate velit esse cillum dolore eu fugiat nulla pariatur.
</div>
```

Yukarıdaki kod, örneğimizin HTML kısmını oluşturuyor. HTML etiketi olarak `DIV` etiketini kullandık. İçerisinde de ünlü "lorem ipsum" ifadesi yer alıyor. Ayrıca CLASS özniteliğini de "lorem" olarak belirledik. Şimdi de Css ataması yaparak biçimlendirmesini yapalım:

```
.lorem {
    font: normal 12px/22px Verdana;
    padding: 10px;
    border: 2px solid #00CC66;
    background: #DFFFDF;
    color: #555555
}
```

Yukarıdaki css kodları yeterince anlaşılır.. Bu örneğimizin sonucunu görmek için Örnek isimli sayfaya bakabilirsiniz. Şimdi ise jQuery'ye geri dönelim

1. Css özellikleri (property)

jQuery ile herhangi bir elemanın css özelliğini almak veya değiştirmek amacıyla **css()** fonksiyonunu kullanıyoruz. İlk önce bir elemanın css özelliği nasıl alınıyor ona bakalım. Mesela yukarıdaki örneğimizin yazı rengi neymiş öğrenelim.

```
$('div.lorem').css('color');
```

`$('div.lorem')` ifadesini önceki yazıdan biliyoruz. Web sayfamızdaki hangi eleman üzerinde çalışacaksak onu seçmemizi sağlıyordu. Biz de burada class özniteliği "lorem" olan DIV elemanımızı seçmiş olduk. Daha sonra ise asıl önemli kısım başlıyor. Çünkü `css('color')` ile seçmiş olduğumuz elemanın yazı rengini alıyoruz. Bu işlemin sonucu bize **#555555** olarak geri dönecek. Test etmek için Örnek 1a sayfasına bakabilirsiniz.

Şimdi ise css özelliği nasıl değiştiriliyor ona bakalım. Örneğimizin yazı rengi **#555555** idi biliyorsunuz. Biz bunu kırmızı renk olan **#FF0000** ile değiştirelim.

```
$('div.lorem').css('color', '#FF0000');
```

Hangi css özelliği değiştirilecekse önce onu yazıyoruz, sonra yeni değerini belirliyoruz. Test etmek için Örnek 1b sayfasına bakabilirsiniz.

Ve şimdi birden fazla css özelliği nasıl değiştiriliyor ona bakalım. Örnek 1b'de elemanın yalnızca yazı rengini değiştirmiştik. Peki hem yazı rengini, hem arkaplan rengini, hem de çerçeve rengini değiştirmek isteseydik ne yapacaktık? İşte bunu:

```
$('div.lorem').css({
    color: '#D12F19',
    background: '#FBE3E4',
    borderColor: '#D12F19'
});
```

Burada dikkat etmeniz gereken en önemli nokta Css özelliklerini javaScript kurallarına göre yazmamız gerektiği… Yani çerçevenin rengini değiştirirken **border-color** yerine **borderColor** yazdığıma dikkat edin. Test etmek için Örnek 1c sayfasına bakabilirsiniz.

2. Css sınıfları (class)

jQuery ile elemanlara yeni css sınıfları ekleyebilir, sahip oldukları css sınıflarını silebilir ve bu ekleme-silme işlemini aynı anda yapabiliriz. Css sınıfı eklemek için `addClass()` fonksiyonunu, silmek için `removeClass()` fonksiyonunu ve ekleme-silme işlemini aynı anda yapmak içinse `toggleClass()` fonksiyonu kullanıyoruz. Şimdi bu üç fonksiyonu tek tek inceleyelim.

Örnek 1c'de elemanımızın renklerini değiştirmiştik. Fakat o örnekteki yeni renkleri tek tek jQuery'ye biz girmek zorundaydık. Oysaki bunun daha pratik şekli şudur: ilk önce css kodlarımız arasında yeni bir sınıf tanımlayacağız, daha sonrada bu sınıfı jQuery'nin `addClass()` fonksiyonu ile elemanımıza atayacağız.

```
.kirmizi {
    color: #D12F19;
    background: #FBE3E4;
```

```
    border-color: #D12F19
}
```

İlk önce css kodlarımız arasına **kirmizi** isimli yeni bir sınıf ekledik.

```
$('div.lorem').addClass('kirmizi');
```

Daha sonra da elemanımıza **kirmizi** isimli sınıfı jQuery ile eklemiş olduk. Test etmek için Örnek 2a sayfasına bakabilirsiniz.

Şimdi de eklediğimiz bu sınıfı silmeyi öğrenelim. Yapacağımız tek şey `removeClass()` fonksiyonu kullanarak silmek istediğimiz sınıf ismini belirtmek olacak o kadar.

```
$('div.lorem').removeClass('kirmizi');
```

Test etmek için Örnek 2b sayfasına bakabilirsiniz.

Son olarak da sınıf ekleme ve silme işlemlerini aynı anda nasıl yapıyoruz ona bakalım. Yani `toggleClass()` fonksiyonu nasıl kullanılıyor inceleyelim.

```
$('div.lorem').toggleClass('kirmizi');
```

Böylece sınıf ekleme ve silme işlemleri için ayrı ayrı kod yazmaktan kurtulmuş olduk. Test etmek için Örnek 2c sayfasına bakabilirsiniz.

3. Css konumlandırmaları (offset)

jQuery ile web sayfamızdaki bir elemanın yükseklik ve genişlik değerlerini almak, bu elemanın ekranın hangi noktasında bulunduğunu tespit etmek gerçekten çok kolaydır. Bir elemanın yükseklik değerini `height()` fonksiyonu ile genişlik değerini ise `width()` fonksiyonu ile alıyoruz. Bu elemanın konumunu almak içinse `offset()` fonksiyonunu kullanıyoruz.

İlk önce örneğimizin yükseklik değerini alalım.

```
$('div.lorem').height();
```

Bu işlem sonucunda örneğimizin yükseklik değeri piksel cinsinden bize geri dönecektir. Yine aynı fonksiyon ile yükseklik değerini de istediğimiz gibi değiştirebiliyoruz. Aynen alttaki gibi:

```
$('div.lorem').height(250);
```

Test etmek için Örnek 3a sayfasına bakabilirsiniz.

Şimdi de örneğimizin genişlik değerini alalım. Zaten aynen yükseklik değerini aldığımız gibi alıyoruz.

```
$('div.lorem').width();
```

Ve genişlik değerini değiştirmeyi görelim.

```
$('div.lorem').width(500);
```

Test etmek için Örnek 3b sayfasına bakabilirsiniz.

Ve son olarak da bir elemanın konumunu alalım. Konum ifadesini açarsak bir elemanın üstten kaç piksel uzaklıkta ve soldan kaç piksel uzaklıkta olduğundan bahsediyorum. Şimdi örneğimizin konumunu tespit etmek için kullanacağımız kodları görelim.

```
konum  = $('div.lorem').offset();
ustten = konum.top;
soldan = konum.left;
```

Gördüğünüz gibi `offset()` fonksiyonun kullanımı biraz değişik. İlk önce bu fonksiyonu bir değişkene atıyoruz. Daha sonra da Top ve Left değerlerine atadığımız bu değişken üzerinden ulaşıyoruz. Test etmek için Örnek 3c sayfasına bakabilirsiniz.

Ayrıca bu fonksiyon ile ilgili olarak şunu da eklemek istiyorum. Biliyorsunuz ki Top ve Left bir css özelliğidir. Bir elemanın Top ve Left özelliğini değiştirmek isterseniz Örnek 1c'dekine benzer bir atama yapmalısınız. Bu atamanın da çalışabilmesi için elemanınız mutlak konumlandırılmış olmasına da dikkat etmelisiniz.

jQuery ve Olaylar

Bir önceki jQuery ve Css işlemleri konusundan sonra şimdi çok daha önemli olan **olaylar (events)** konusuyla devam edelim. Olaylar, bir kullanıcının bir web sayfasında yapmış olduğu etkileşimlerdir. Örneğin kullanıcı bir linke tıkladığında click olayı, klavyeden bir tuşa bastığında ise keydown olayı gerçekleşmiş olur. Bu olayların ne olduğunu ve bu olaylar oluştuğunda nasıl işlemler yapılabileceğini bu yazıda anlatmaya çalışacağım.

Olaylar konusunu 5 başlık altında anlatmaya çalışacağım. Bu yüzden önce başlıklarına bir gözatalım:

1. Sayfa yüklendiği anda kod çalıştırmak
2. Anonim fonksiyonlardan faydalanmak
3. Olay Yardımcıları
4. Olay İşleme
5. Etkileşim yardımcıları

Yazı başlıklarını da öğrendikten sonra artık başlayabiliriz 😉

1. Sayfa yüklendiği anda kod çalıştırmak

Öncelikle bundan sonra sık sık kullanacağımız **ready** olayını tanıyalım. Bu olay ile bir elemanın kullanıma hazır olduğu anda işlemler yapabiliyoruz. Örneğin web sayfamızın yüklenip de kullanıma hazır olduğu anda bir fonksiyon çalıştırmasını sağlayalım:

```
$(document).ready( calisacakFonksion );
```

`$(document)` ifadesi ile üzerinde çalışacağımız belgeyi (dökümanı) seçtik. Daha sonra bu belgenin **ready** olayında **calisacakFonksiyon** isimli fonksiyonun işlem görmesini belirttik. Şimdi küçük bir örnekle bu durumu pekiştirelim:

```
$(document).ready( mesajGoster );

function mesajGoster()
{
```

```
    alert('merhaba dünya');
}
```

Bu kodlara sahip bir web sayfası yüklendiğinde "merhaba dünya" yazan mesaj kutusu gösterilecektir. Test etmek için Örnek 1 sayfasına bakabilirsiniz.

2. Anonim fonksiyonlardan faydalanmak

Dikkat ettiyseniz üstteki örnekte `mesajGoster()` fonksiyonu herhangi bir parametre almıyor. Fakat pekçok zaman bir fonksiyona parametreler göndermemiz gerekebiliyor. Örneğin `mesajGoster()` fonksiyonuna bir **isim** parametresi ekleyelim.

```
function mesajGoster( isim )
{
    alert('merhaba ' + isim);
}
```

Böyle bir durumda fonksiyonumuza parametre göndererek çalışmasını sağlamak için `$(document).ready(mesajGoster)` yerine şöyle bir kod kullanıyoruz:

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

    mesajGoster( 'türkiye' );

});
```

Bu kodların olduğu bir web sayfası yüklendiği zaman "merhaba türkiye" yazan mesajı gösterecektir. Test etmek için Örnek 2 sayfasına bakabilirsiniz.

3. Olay Yardımcıları

Yazının giriş kısmında da belirttiğim gibi kullanıcının bir linke tıklamasından, klavyedeki bir tuşa basmasına kadar pekçok türde olay bulunuyor. jQuery ise bu olay türlerinin hepsini **Olay Yardımcıları (Event Helpers)** başlığı altında toplamış. İşte biz de bu yardımcıları kullanarak istediğimiz işlemlerin istediğimiz bir olay gerçekleştiği anda icra edilmesini sağlayabiliyoruz. jQuery Events sayfasından olay yardımcılarının tamamına ulaşabilirsiniz. Ben burada en fazla kullanıldığını düşündüğüm olay yardımcılarından bahsedeceğim.

- **click():** bir elemana tıklanıldığında
- **dblclick():** bir elemana çift tıklanıldığında
- **mouseover():** fare imleci bir elemanın üzerine geldiğinde
- **mouseout():** fare imleci bir elemanın üzerinden başka bir yere götürüldüğünde
- **focus():** bir form elemanına odaklanıldığında
- **blur():** odaklanılmış bir form elemanından başka bir yere odaklanıldığında
- **change():** bir form elemanı değiştirildinde
- **submit():** bir form gönderildiğinde
- **keydown():** klavyeden bir tuşa basıldığında
- **keyup():** klavyeden bir tuşa basılıp bırakıldığı anda

Olay yardımcılarının kullanımı da aynen birinci bölümde bahsettiğim **ready** olayınınki gibidir. Fakat biz yine de bir örnek yapalım. Örneğimizde ID özniteliği "link" olan bir bağlantıya tıklandığında web sayfamızın arkaplan rengi yeşil olsun:

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

    $('a#link').click( function(){
```

```
        $('body').css('background-color', '#349934');

    })
});
```

Bu örnekte ilkönce web sayfamız kullanıma hazır mı değil mi `$('document').ready()` ile kontrol ettik. Daha sonra da `$('a#link')` ile bağlantımızı seçtik ve `click()` olayı gerçekleştiğinde web sayfamızın arkaplan rengini yeşil yapmış olduk. `css()` ifadesinin ise bir elemanın css özelliğini değiştirdiğini bir önceki yazı'dan hatırlıyor olmalısınız. Bu örneği Örnek 3 sayfasında test edebilirsiniz.

4. Olay İşleme

jQuery ile olayları daha kapsamlı bir biçimde işleyebilmek için **Olay İşleme (Event Handling)** fonksiyonlarını kullanıyoruz. Bu fonksiyonlar ile bir elemana yukarıda bahsettiğimiz olay yardımcılarından herhangi birisini atayabilir, önceden atadığımız bir olayı kaldırabilir veya bir eleman içerisinden başka bir elemana atanmış olan olayı tetikleyebiliriz. Benim anlatacağım olay işleme fonksiyonları `bind()`, `unbind()`, `one()` ve `trigger()` olacak.

İlkönce `bind()` fonksiyonuna bakalım. Bu olay işleme fonksiyonun amacı herhangi bir elemana olay yardımcılarından birisini atamaktır. Diğer önemli bir görevi ise yukarıda bahsettiğimiz olay yardımcılarının dışında kendimize özel olaylar da oluşturabilmektir. Alttaki örnek ile ID özniteliği "link" olan bir bağlantıya tıklandığı zaman web sayfamızın arkaplan rengi yeşil oluyor.

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

    $('a#link').bind('click', function(){
        $('body').css('background-color', '#349934');
    });

});
```

Aslında bu örnekte yapmış olduğumuz iş bir önceki Örnek 3 ile aynı sonucu verecektir. Fakat `bind()` ile atadığımız herhangi bir olayı, istediğimiz zaman `unbind()` fonksiyonu ile kaldırabilme avantajına sahibiz. Şimdi `unbind()` fonksiyonunu inceleyelim.

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

    $('a#link').unbind('click');

});
```

Bu şekilde daha önceden ID özniteliği "link" olan bağlantıya atadığımız "click" olayını kaldırmış oluyoruz. Böylece bu bağlantıya tıklandığında artık hiçbirşey olmayacaktır. Eğer click, mouseover, dblclick… gibi sahip olduğu bütün olayları tek seferde kaldırmak isteseydik o zaman `$('a#link').unbind();` ifadesini kullanmamız yeterli olacaktı. Bu son iki örneği bir arada test etmek isterseniz Örnek 4A isimli sayfaya bakabilirsiniz.

Şimdi de `one()` isimli olay işleme fonksiyonuna bakalım. Bazen bir olayın **yalnızca 1 kez** olmasını isteriz. Örneğin kullanıcı bir bağlantıya tıkladığında birkez mesaj kutusu görünsün, diğer tıklamalarında ise bu mesaj kutusu görünmesin gibi…

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

    $('a#link').one('click', function(){
```

```
        alert('bu mesaj kutusu yalnızca 1 kez görüntülenecek')
    })

})
```

İşte üstteki örnekte kullanıcı, ID özniteliği "link" olan bağlantıya ilk tıklamasında mesaj kutusunu görecek, diğer tıklamada ise görmeyecektir. Bu örneği test etmek için Örnek 4B isimli sayfaya bakabilirsiniz.

Son olarak ise `trigger()` fonksiyonundan bahsedelim. Kendisi bir tetikçidir Örneğin bir bağlantının "click" olayında bir mesaj kutusu gösteriliyor olsun. Normalde bu mesaj kutusu biz yalnızca bağlantıya tıkladığımızda görüntülenecektir. Fakat `trigger()` ile bir buton üzerine geldiğimizde, bağlantıya atanmış olan Click olayının tetiklenmesini ve mesaj kutusunun gösterilmesini sağlayabiliyoruz. Şimdi alttaki örneğe bakalım:

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

    $('a#link').bind('click', function(){
        alert('merhaba dünya')
    })

    $('button').bind('mouseover', function(){
                $('a#link').trigger('click')
    })

})
```

Gördüğünüz gibi sondaki `trigger('click')` ifadesi ile bir buton içerisinden dışarıdaki bir bağlantının "click" olayını tetikleyebiliyoruz. Örneği test etmek için Örnek 4C isimli sayfaya bakabilirsiniz.

5. Etkileşim yardımcıları

Yazının ilk paragrafında olayların, bir kullanıcının bir web sayfasında yapmış olduğu etkileşimler olduğunu söylemiştik. İşte bu etkileşimleri arttırmak için jQuery bizlere **Etkileşim Yardımcıları (Interaction Helpers)** isimli fonksiyonları sunmuştur. Bu fonksiyonlar `hover()` ve `toggle()` olmak üzere iki tanedir.

`hover()` fonksiyonu, fareyi bir elemanın üzerine getirdiğimizde bir işlem, üzerinden çektiğimizde ise başka bir işlem yapabilmemizi sağlıyor. Üçüncü bölümde bahsettiğimiz olay yardımcılarından `mouseover()` ve `mouseout()` ile de bunu yapabiliriz ama jQuery bize `hover()` ile önemli bir kolaylık sağlıyor. Şimdi örneğe geçelim. Örnekte fareyi bir bağlantının üzerine getirdiğimizde web sayfamızın arkaplan rengi yeşil renk, üzerinden çektiğimizde ise mavi renk olsun.

```
$(document).ready( function() {
    $('a#link').hover( yesilYap, maviYap );
})

function yesilYap(){
    $('body').css('background-color', '#339933')
}

function maviYap(){
    $('body').css('background-color', '#509dee')
}
```

Örnekte ilk önce `yesilYap()` ve `maviYap()` isimli fonksiyonların hazır olduğunu kabul ediyoruz. Daha sonra ise `hover()` ile sırasıyla bağlantının üstündeyken ve üstünde değilken bu fonksiyonların çalıştırılmasını belirtiyoruz. Örneği test etmek için Örnek 5A sayfasına bakabilirsiniz.

Şimdi ise diğer etkileşim yardımcısı olan `toggle()` fonksiyonuna bakalım. Bir önceki `hover()` fonksiyonuyla bir elemanın üzerindeyken ve üzerinde değilken olmak üzere 2 farklı işlem yapabiliyorduk. `toggle()` fonksiyonu ise bir elemana her tıkladığımızda farklı 2 işlemden birisinin yapılabilmesini sağlıyor. Yani bir bağlantıya ilk tıkladığımızda web sayfamızın arkaplanı yeşil oluyorken, diğer tıklayışımızda ise mavi oluyor. Şimdi bu son dediğimizi bir örneğe dönüştürelim:

```
$(document).ready( function() {
    $('a#link').toggle( yesilYap, maviYap )
})

function yesilYap(){
    $('body').css('background-color', '#339933')
}

function maviYap(){
    $('body').css('background-color', '#509dee')
}
```

Gördüğünüz gibi `toggle()` fonksiyonu da `hover()` fonksiyonu gibi çalışıyor. Bu örneği test etmek için Örnek 5B sayfasına bakabilirsiniz.

Sonuç…

Bu yazıda, jQuery ile olayları yönetebilmek için gerekli olan fonksiyonlardan bazılarını anlatmaya çalıştım. Olaylar konusunda anlatmak isteyip de anlatamadığım daha birçok konu kaldı maalesef. Bunların hepsini tek bir yazıya sıkıştırmak istemedim. İlerleyen zamanlarda bunlardan yeri geldikçe bahsedeceğim. Siz yine de jQuery web sitesinin Events sayfasına bakmayı unutmayın

jQuery ve DOM işlemleri

DOM ismi verilen bir standart sayesinde web sayfamızdaki her eleman *(resim, link, buton)* birer nesne olarak değerlendirilir. Biz de javaScript yardımıyla bu nesnelere erişebilir ve üzerinde işlemler yapabiliriz. Yani bir resmi yenisiyle değiştirebilir, bir linkin başına/sonuna yeni nesneler ekleyebilir, bir buton nesnesinin birebir kopyasını çıkartabiliriz. jQuery sayesinde ise bu işlemleri çok daha kolay bir şekilde yapabiliyoruz.

Konuya başlamadan önce başlıklara bir gözatalım:

1. Değiştirme
2. Ekleme
3. Silme
4. Kopyalama

1. Değiştirme

Bir elemanın tamamını veya sadece içeriğini değiştirmek için kullanabileceğimiz jQuery fonksiyonları `html()`, `text()`, `replaceWith()` ve `replaceAll()` isimli fonksiyonlar. `html()` ve `text()` fonksiyonları bir elemanın "içeriğini" değiştirirken, `replaceWith()` ve `replaceAll()` fonksiyonları ise bir elemanın "tamamını" değiştirir.

html(): Bu fonksiyon ile bir elemanın içeriğini HTML etiketleriyle birlikte alabilirsiniz ve değiştirebilirsiniz. innerHTML özelliğine benzer. Kullanımı ise şöyledir:

```
// içerik al
$('p#metin').html();

// içerik değiştir
$('p#metin').html('yeni içerik');
```

Bu fonksiyon ile ilgili bir örneği Örnek 1A sayfasında bulabilirsiniz.

text(): Bu fonksiyon sayesinde bir elemanın sahip olduğu içeriği "düz metin" olarak alabiliyoruz veya değiştirebiliyoruz. innerText özelliğine benzer. Kullanımı ise şöyledir:

```
// içerik al
$('p#metin').text();

// içerik değiştir
$('p#metin').text('yeni içerik');
```

Bu fonksiyon ile ilgili bir örneği Örnek 1B sayfasında bulabilirsiniz.

replaceWith(): Bu fonksiyon sayesinde bir elemanın kendisini başka bir eleman ile değiştirebiliriz. Örneğin bir butonu bir resim ile değiştirmek istiyorsanız bu fonksiyonu kullanmalısınız. İlk önce değiştirilecek elemanı belirtip, daha sonra da yeni elemanı belirtiyoruz. Kullanımı şöyledir:

```
// butonu resim ile değiştir
$('button').replaceWith('<img src="resim.jpg" />');
```

Bu fonksiyon ile ilgili bir örneği Örnek 1C sayfasında bulabilirsiniz.

replaceAll(): Bu fonksiyon biraz önce bahsettiğimiz replaceWith() fonksiyonuyla aynı işlemi yapmasına karşın ters şekilde çalışıyor. İlkönce yeni elemanı belirtip daha sonra değiştirilecek olan elemanı belirtiyoruz. Kullanımı şöyledir:

```
// bütün butonları resim ile değiştir
$('<img src="resim.jpg" />').replaceAll('button');
```

Bu fonksiyon ile ilgili bir örneği Örnek 1D sayfasında bulabilirsiniz.

2. Ekleme

jQuery'nin ekleme fonksiyonları sayesinde bir elemanın içerisine veya dışarısına yeni elemanlar ekleyebiliyoruz. Öncelikle bir elemanın içerisine yeni elemanlar eklerken kullanacağımız `append()` ve `prepend()` fonksiyonlarına bakalım.

append(): Bu fonksiyon ile bir elemanın sahip olduğu içeriğin "en sonuna" yeni bir eleman ekleyebiliriz. Alttaki örneğin uygulamasını Örnek 2A sayfasında bulabilirsiniz.

```
// örnek metin
<p id="metin">Merhaba</p>

// sonuna ekle
$('p#metin').append(' <strong>Erhan<strong>');

// sonuç
<p id="metin">Merhaba  <strong>Erhan<strong></p>
```

prepend(): Bu fonksiyon ile bir elemanın sahip olduğu içeriğin "en başına" yeni bir eleman ekleyebiliriz. Alttaki örneğin uygulamasını Örnek 2B sayfasında bulabilirsiniz.

```
// örnek metin
<p id="metin">Merhaba</p>

// sonuna ekle
$('p#metin').prepend('<strong>Erhan<strong> ');

// sonuç
<p id="metin"><strong>Erhan<strong> Merhaba</p>
```

Şimdi de bir elemanın dışına nasıl yeni elemanlar ekleyebiliriz ona bakalım. Bu iş için kullanacağımız fonksiyonlar `after()` ve `before()` olacak.

after(): Bu fonksiyon ile bir elemandan "sonra" yeni bir eleman ekleyebiliriz. Örneğin bir butondan sonra bir resim eklenebilir. Bu örneği Örnek 2C sayfasında bulabilirsiniz.

```
$('button').after('<img src="resim.jpg" />');
```

before(): Bu fonksiyon sayesinde bir elemandan "önce" yeni bir eleman ekleyebiliriz. Örneğin bir butondan önce bir resmin eklenmesini sağlayabiliriz. Bu örneği Örnek 2D sayfasında bulabilirsiniz.

```
$('button').before('<img src="resim.jpg" />');
```

jQuery'nin ekleme fonksiyonları aslında benim burada anlattığımla sınırla değil tabiki. Daha birçok ekleme fonksiyonu var. Bu yüzden lütfen jQuery'nin yardım belgelerine bakmayı ihmal etmeyin.

3. Silme

Pekçok durumda bir elemanı tamamen kaldırmak veya bir elemanın içerisindeki alt elemanları kaldırmak gerekebiliyor. Bu durumda jQuery'nin `empty()` veya `remove()` fonksiyonlarından birisini kullanmalıyız.

empty(): Bir elemanın içeriğini boşaltır. Örneğin ID özniteliği "alan" olan bir `DIV` elemanın içerisini boşaltmak/temizlemek için şöyle yapıyoruz:

```
$('div#alan').empty();
```

Bu örneği test etmek için Örnek 3A sayfasına bakabilirsiniz.

remove(): Bir elemanı tamamen ortadan kaldırır. Örneğin ID özniteliği "alan" olan bir `DIV` elemanı tamamen ortadan kaldırmak için şöyle yapıyoruz:

```
$('div#alan').remove();
```

Bu örneği test etmek için Örnek 3B sayfasına bakabilirsiniz.

4. Kopyalama

Eğer bir elemanın birebir kopyasını çıkartmak yani klonlamak istiyorsanız jQuery'nin `clone()` fonksiyonunu kullanmalısınız. Bu fonksiyon `clone()` ve `clone(true)` olmak üzere iki farklı şekilde kullanılıyor. Peki bir elemanın kopyasını neden çıkartma ihtiyacı duyarız? Bu sorunun cevabı için SpeedyShare sitesine girip *"Upload more files at once"* linkine tıklayın. Sadece bir örnek 

clone(): Bir elemanın kopyasını çıkartır fakat elemanın sahip olduğu olayları *(events)* kopyalamaz. Daha önceden fare ile üzerine geldiğinizde rengi değişen bir elemanın, kopyasının üzerine geldiğinizde ise rengi değişmeyecektir. Bu örneğin uygulamasını Örnek 4A sayfasında görebilirsiniz. Bu fonksiyonunun kullanımı ise şöyledir:

```
// bir metin kutusunun kopyasını çıkar
$('input').clone();
```

clone(true): Bir elemanın kopyasını sahip olduğu olaylar *(events)* ile birlikte çıkartır. Yani daha önceden fare ile üzerine geldiğinizde rengi değişen bir elemanın, kopyasının üzerine geldiğinizde yine rengi değişecektir. Bu örnekle ilgili bir uygulamayı Örnek 4B sayfasında görebilirsiniz. Bu fonksiyonunun kullanımı ise şöyledir

```
// bir metin kutusunun kopyasını çıkar (olayları ile birlikte)
$('input').clone(true);
```

Kopyalama işlemleri sırasında olaylar *(events)* konusunun da ismi geçti. Eğer olaylar konusunu daha önceden okumadıysanız ya da yeniden okumak isterseniz jQuery ve Olaylar başlıklı yazıma bakabilirsiniz. Bu yazıdaki örnekleri ise buradan indirebilirsiniz.

jQuery ve AJAX işlemleri

Bu yazıda jQuery kütüphanesi ile AJAX işlemlerinin nasıl yapıldığından bahsedeceğim. Prototype ve MooTools kütüphanelerini de kullanmış biri olarak size söyleyebilirim ki AJAX işlemlerinin en kolay ve esnek bir şekilde jQuery ile yapılabildiğini gördüm. Eminim ki yazının sonunda bu konuda bana hak vereceksiniz, çünkü gerçekten jQuery ile bir AJAX işlemi yapmak çok kolay 😉

Bu konuyu resmi jQuery dökümanında olduğu gibi 3 başlık altında anlatmak istiyorum:

1. Ajax istekleri
2. Ajax olayları
3. Ajax yardımcıları

1. Ajax istekleri

Bir AJAX isteği oluşturmak için `$.ajax()` fonksiyonundan faydalanıyoruz. Yapacağımız işlemin türüne göre bu fonksiyona farklı farklı seçenekler belirtmek durumundayız. Bu seçeneklerden en temel olanlarını kısaca inceleyelim:

- **type:** Bir web sayfasına yapılacak olan isteğin türünü belirler. *GET* veya *POST* olmak üzere iki farklı değerden birisini almalıdır.
- **url:** istek yapılacak sayfayı belirtmenizi sağlar. *google-backlink.php* gibi…
- **data:** istek yapılan sayfaya herhangi bir bilgi göndereceksenız bu bilginin ne olduğunu belirtmenizi sağlar. Örneğin *google-backlink.php* isimli bir sayfaya *url=www.eburhan.com* bilgisini göndermek için bu seçeneği kullanmalıyız.
- **success:** yapmış olduğumuz isteğin başarılı olup olmadığını kontrol etmemizi sağlar. Örneğin yapılan başarılı bir istek sonucunda geri dönen cevabı işlemek için bu seçeneği kullanabiliriz.

`$.ajax()` fonksiyonundan ve onun alabileceğini seçeneklerden de bahsettiğimize göre artık bunların nasıl kullanıldığına geçelim. Örnek uygulamamızda *"google-backlink.php"* sayfasına *"url=www.eburhan.com"* bilgisini *GET* yöntemi ile gönderip sonucu alacağız. Aldığımız sonucu ise ID özniteliği "sonuc" olan `DIV` etiketi içerisinde göstereceğiz.

```
$.ajax({
        type: 'GET',
        url: 'google-backlink.php',
        data: 'url=www.eburhan.com',
        success: function(ajaxCevap) {
                $('#sonuc').html(ajaxCevap);
        }
});
```

Bu örnek uygulamanın çalışır halini Örnek 1A sayfasında bulabilirsiniz.

jQuery ile AJAX isteği oluşturmanın çok daha kısa yolları da vardır. Bu yollar `$.get()` ve `$.post()` fonksiyonlarından geçiyor. Fonksiyon isimlerinden de anlaşılacağı üzere GET türünde bir istek için `$.get()` fonksiyonunu, POST türünde bir istek için `$.post()` fonksiyonunu kullanabiliriz. Bir önceki örnek uygulamanın `$.get()` fonksiyonu ile nasıl yapılabileceğini görelim:

```
$.get(
        'google-backlink.php',
        {url: 'www.eburhan.com'},
        function(ajaxCevap){
        $('#sonuc').html(ajaxCevap)
        }
);
```

`$.get()` fonksiyonunda öncelikle istek yapılacak sayfayı, daha sonra ise bu sayfaya gönderilecek bilgiyi belirtiyoruz. Sonra da geri dönen cevabı bir fonksiyon içerisinde işliyoruz. Çalışır halini Örnek 1B sayfasında görebilirsiniz.

Şimdi de herhangi bir web sayfasının içeriğini kendi web sayfamıza kolayca entegre edebilmemizi sağlayan `$.load()` fonksiyonunu inceleyelim. Örneğimizde "delicious-eburhan.php" diye bir dosyamız olsun. Bu dosyanın görevi, Del.icio.us isimli dünyaca ünlü sosyal imleme sitesinde saklamış olduğum son 10 bağlantıyı göstermek olsun. Biz bu gösterimi `$.load()` fonksiyonu ile kendi sayfamızda yer alan `DIV` etiketi içerisinde yapacağız.

```
$('#sonuc').load('delicious-eburhan.php');
```

İşte bu tek satırlık kod yardımıyla "delicious-eburhan.php" dosyasının tüm içeriğini web sayfamızdaki `DIV` etiketi içerisine yüklemiş oluyoruz. Gerçekten kolay değil mi? Bu örneğin çalışır halini Örnek 1C sayfasında bulabilirsiniz.

2. Ajax olayları

Bu yazının başındaki örnekte *(Örnek 1A)* **success** isimli bir seçenek kullanmıştık. Bu seçeneğin görevi, bir AJAX isteği başarılı olarak tamamlandıktan sonra bizim herhangi bir işlem yapabilmemizi sağlamaktı. Bu seçeneği her AJAX isteğinde kullanmak için her defasında belirtmemiz gerekiyor. Oysaki biz böyle tekrarlara düşmeyelim diye jQuery bize "ajax olayları" diye bir olanak sunmuş. Alttaki örneğe bakalım:

```
$("#sonuc").ajaxSuccess(function(){

        $(this).html('ajax isteği başarılı');

});
```

Bu örnekte, web sayfamızdaki **bütün** AJAX istekleri başarıyla sonuçlandığı zaman ID özniteliği "sonuc" olan bir elemanın içerisine "ajax isteği başarılı" mesajı yazılacaktır. Yani ayrıyeten herbir `$.ajax()` fonksiyonu içerisinde belirtmemize gerek kalmamıştır.
Örneği test etmek için Örnek 2A sayfasına bakabilirsiniz.

jQuery kütüphanesi içerisindeki ajax olayları şöyledir:

- **ajaxComplete:** bir ajax isteği tamamlandığında
- **ajaxError:** bir ajax isteği başarısız olduğunda

- **ajaxSend:** bir ajax isteği gönderilmeden önce
- **ajaxStart:** bir ajax isteği başladığında
- **ajaxStop:** bir ajax isteği durdurulduğunda
- **ajaxSuccess:** bir ajax isteği başarıyla tamamlandığında

Küçük bir örnekle de ajax olayları konusunu geçmek istiyorum. Hepiniz AJAX kullanan web sitelerinde bunlar gibi yükleniyor *(loading)* resimleri görmüşsünüzdür. İşte böyle birşey oluşturmak için **ajaxStart** isimli olayı kullanabiliriz. Böylece herhangi bir AJAX isteği başladığında web sayfanızın bir yerinde yükleniyor resmi gösterebilirsiniz

```
$("div#yukleniyor").ajaxStart(function(){

        $(this).html('<img src="yukleniyor.gif" />');

});
```

Bu örneğin uygulamasını da Örnek 2B sayfasında görebilirsiniz.

3. Ajax yardımcıları

Bu başlık altında `ajaxSetup()` ve `serialize()` isimli iki fonksiyonu inceleyeceğiz.

Yardımcı fonksiyonlar sayesinde AJAX işlemlerini daha da kolaylaştırabilirsiniz. Örneğin web sayfanızda birden fazla `$.ajax()` fonksiyonu kullanmışsınız fakat bu fonksiyonun bazı seçenekleri hep tekrar etmiş. Oysaki `ajaxSetup()` yardımcısı ile bu tekrar eden seçenekleri baştan belirleyebiliyoruz. Böyle olunca da her `$.ajax()` fonksiyonuna aynı seçenekleri tekrar tekrar girmekten kurtulmuş oluruz.

```
$.ajax({
        type: 'GET',
        url: 'google-backlink.php',
        data: 'url=www.eburhan.com',
        success: function(ajaxCevap) {
                $('div#sonuc').html(ajaxCevap);
        }
});
$.ajax({
        type: 'GET',
        url: 'google-backlink.php',
        data: 'url=www.yakuter.com',
        success: function(ajaxCevap) {
                $('div#sonuc').html(ajaxCevap);
        }
});
```

Şimdi yukarıdaki iki koda bakalım. Bu kodların ikisi de aynı web sayfasında bulunuyor. Dikkat ettiyseniz "type", "url" ve "success" seçenekleri her ikisinde de ortak. Oysaki `ajaxSetup()` yardımcısı ile bu ortak ifadelerin tek bir yerde toplanması sağlanabiliyor. Böylece kodlarımız daha temiz görünüyor.

```
$.ajaxSetup({
        type: 'GET',
        url: 'google-backlink.php',
        success: function(ajaxCevap) {
                $('div#sonuc').html(ajaxCevap);
```

```
        }
});
$.ajax({ data: 'url=www.eburhan.com' });
$.ajax({ data: 'url=www.yakuter.com' });
```

Böylesi daha iyi değil mi Test etmek isterseniz Örnek 3A sayfasına bakabilirsiniz.

Şimdi de form alanlarında sıklıkla kullanacağımız `serialize()` yardımcısına geçelim. Bu yardımcı sayesinde bir forma girilen bütün bilgiler **sorgu cümleciği** (query string) haline otomatik olarak dönüştürülürler. Bu sorgu cümleciğini herhangi bir sayfaya bilgi gönderirken kullanırız. Şimdi alttaki gibi basit bir formumuz olduğunu varsayalım:

```
<form method="post" action="form-isle.php">
  <p>İsim: <input type="text" name="isim" /></p>
  <p>Site: <input type="text" name="site" /></p>
  <p>Şehir:
     <select name="sehir">
     <option value="izmir">İzmir</option>
     <option value="istanbul">İstanbul</option>
     <option value="ankara">Ankara</option>
     </select>
  </p>
  <p><button type="submit">Formu Serileştir</button></p>
</form>
```

Girilen örnek bilgilerle beraber bu kodların ekran görüntüsü alttaki gibi olacaktır:

Şimdi de `serialize()` yardımcısını kullanarak bu formdaki bilgileri bir sorgu cümleciği haline dönüştüren kodu yazalım:

```
$('form).serialize();
```

Bu kodu çalıştırdığınızda altaki gibi bir sorgu cümleciği elde etmiş olacaksınız. Kendiniz test etmek isterseniz Örnek 3B sayfasına da bakabilirsiniz.

```
isim=Erhan&site=eburhan.com&sehir=izmir
```

Artık `serialize()` yardımcısı ile oluşturduğumuz bu sorgu cümleciğini istediğimiz gibi kullanabiliriz. Özellikle de bir formdaki bilgileri kullanarak bir AJAX isteği yaptığımızda bu yardımcı işimizi çok kolaylaştıracaktır. Öyleyse bir AJAX isteğinde bu yardımcı nasıl kullanıyor ona da bakalım ve bu yazıya da son noktayı koymuş olalım

```
$.ajax({
        type: 'POST',
        url: 'form-isle.php',
        data: $('form').serialize(),
        success: function(ajaxCevap) {
                $('#sonuc').html(ajaxCevap);
        }
```

```
});
```

NOT: yazıdaki örnek uygulamaları buradan indirebilirsiniz.

jQuery ve Efekt işlemleri

jQuery serisinin ilk yazısı olan jQuery dünyasına adım atıyoruz başlıklı yazıda jQuery kütüphanesinin hem javaScript, hem AJAX, hem de bir efekt kütüphanesi olduğunu belirtmiştim. Bu yazıda ise jQuery ile web sayfalarımıza zenginlik katan efekt işlemlerinin nasıl yapılabileceğinden bahsedeceğim. Hem bizlere sunulan hazır fonksiyonları, hem de kendi efektlerimizi nasıl yapabileceğimizi anlatmaya çalışacağım.

Öncelikle yazı başlıklarına bir gözatalım:

1. Gösterme ve gizleme efektleri
2. Slayt efektleri
3. Şeffaflık efektleri
4. Özel efektler

1. Gösterme ve gizleme efektleri

Gizli bir elemanı görünür kılmak, görünür bir elemanı gizlemek için kullanabileceğimiz efektlerdir. Bu efektleri kullanabilmemizi sağlayan fonksiyonlar `show()`, `hide()` ve `toggle()` fonksiyonlarıdır. `show()` fonksiyonu ile gizli bir elemanı gösterirken, `hide()` fonksiyonuyla da görünür bir elemanı gizliyoruz. `toggle()` fonksiyonu ile de gösterme ve gizleme efektlerinin aynı anda gerçekleşmesini sağlayabiliyoruz.

```
// gizle
$('div#metin').hide('normal');

// göster
$('div#metin').show('normal');

// gizle/göster
$('div#metin').toggle('normal');
```

Yukarıdaki fonksiyonlarla ID özniteliği "metin" olan bir `DIV` elemanı üzerinde gösterme ve gizleme efektlerini uygulamış oluyoruz. Fonksiyonlarda kullanılan **normal** ifadesi, efekt işleminin **hızını** belirtiyor. Eğerki "slow" ifadesi kullansaydık efekt yavaş, "fast" ifadesi kullansaydık hızlı bir şekilde gerçekleşecekti. Ayrıca bu efekt fonksiyonlarına "milisaniye" cinsinden değer de atayabilirsiniz. Örneğin `$('div#metin').hide(350);` gibi… Böylece bu efekt, 350 milisaniye hızında etkili olacaktır.

Gösterme ve gizleme efektleri ile ilgili örnekleri Örnek 1 sayfasında test edebilirsiniz.

2. Slayt efektleri

Slayt efektleri de bir elemanın görünürlüğünü değiştirirler. `slideDown()` fonksiyonu ile gizli bir elemanı, yükseklik *(height)* değerini yavaş yavaş arttırarak görünür kılıyoruz. Bunun tam tersi mantıkta çalışan `slideUp()` fonksiyonuyla ise görünür bir elemanı, yükseklik *(height)* değerini yavaş yavaş azaltarak gizlemiş oluyoruz. Bir de bu iki efekti aynı anda uygulamak için kullandığımız `slideToggle()` fonksiyonu var.

```
// gizle
$('div#metin').slideUp('normal');

// göster
$('div#metin').slideDown('normal');

// gizle/göster
$('div#metin').slideToggle('normal');
```

Slayt efektleri ile ilgili örnekleri Örnek 2 sayfasında test edebilirsiniz.

3. Şeffaflık efektleri

Şeffaflık efektleriyle bir elemanın şeffaflık *(opacity)* değerini kullanarak görünürlüğünü değiştiriyoruz. `fadeIn()` fonksiyonu ile gizli bir elemanı, şeffaflık değerini yavaş yavaş arttırarak görünür kılıyoruz. Bununla ters mantıkta çalışan `fadeOut()` fonksiyonuyla ise görünür bir elemanı, şeffaflık değerini yavaş yavaş azaltarak gizlemiş oluyoruz. Bunların yanında bir de `fadeTo()` isimli özel bir fonksiyon daha var. Bu fonksiyonun görevini alttaki paragrafta bulabilirsiniz.

Bir elemanın şeffaflık değeri **0** ile **1** arasında olabiliyor. Şeffaflık değeri 0′a yakın olan eleman gizleniyormuş gibi algılanırken, şeffaflık değeri 1′e yakın olan bir eleman ise görünür olarak algılanır. Bu iki değer arasında oynamalar yaparak çeşitli efektler oluşturabiliriz. İşte `fadeTo()` isimli fonksiyon da bir elemanın şeffaflık değeri üzerinde oynama yapabilmektedir. Bu fonksiyonu `$('div#metin').fadeTo('normal', 0.5);` şeklinde kullanabiliriz. İlk önce "normal" parametresi ile efektin gerçekleşme hızını, daha sonra da elemanın "yeni şeffaflık değeri" belirtmiş oluyoruz.

```
// gizle
$('div#metin').fadeOut('normal');

// göster
$('div#metin').fadeIn('normal');

// şeffaflığını 0.5 yap
$('div#metin').fadeTo('normal', 0.5);
```

Şeffaflık efektleri ile ilgili örnekleri Örnek 3 sayfasında test edebilirsiniz.

4. Özel efektler

Aslında buraya kadar anlattığımız bütün efektler, bir elemanın bazı CSS özellikleri üzerinde oynamalar yapılarak oluşturuluyorlar. Öyleyse biz de herhangi bir elemanın bazı CSS

özelliklerini değiştirerek yeni efektler/animasyonlar oluşturabiliriz. Peki, bunu nasıl yapacağız? Cevap `animate()` fonksiyonunda 😉

`animate()` fonksiyonuna öncelikle bir elemanın yeni CSS özelliklerinin ne olması gerektiğini belirtiyoruz. Daha sonra da bu değişikliğin hangi sürede gerçekleşeceğini giriyoruz. Gerisini `animate()` fonksiyonu hallediyor. Bu fonksiyonda dikkat etmeniz gereken tekşey, kullandığınız CSS özelliklerinin sayısal olarak arttırılabilir olmasıdır. Örneğin `height:15px` benzeri bir özellik sayısal olarak arttırılabilmesine karşın *(15px, 25px, 55px... gibi)*, `color:'red'` benzeri bir özellik sayısal olarak arttırılamaz.

```
$('div#metin').animate({
  width: '450px',
  padding: '25px',
  fontSize: '16px'
}, 3000 );
```

Bu örnekte ID özniteliği "metin" olan `DIV` elemanının `width`, `padding` ve `fontSize` özelliklerini değiştirmiş olduk. Ve bu değişimi 3000 milisaniyelik bir zaman dilimi içerisinde yaptık. Yani kendi efektimizi oluşturmuş olduk 😉Hazırladığımız bu efekti görmek için Örnek 4A sayfasına bakabilirsiniz.

Üstteki örnekte efektin gerçekleşme süresi 3000 milisaniye idi. Bazı zamanlar, efektin gerçekleşmesini durdurmak isteyebilir, yani 3000 milisayenin dolmasını beklememek isteyebiliriz. Böyle bir bir durumda `animate()` fonksiyonunun çalışmasını `stop()` isimli başka bir fonksiyon ile durdurabiliyorsunuz. Bu fonksiyonu `$('div#metin').stop();` şeklinde kullanabilirsiniz. Bir örnek için Örnek 4B sayfasına bakabilirsiniz.

Yazıda geçen örnekleri ise buradan indirebilirsiniz.

jQuery ve JSON işlemleri

AJAX teriminin sonundaki X harfi XML dilini temsil ediyor. XML ise farklı platformlar arasında veri alış-verişi yapılabilmesi için oluşturulmuş bir dil. Fakat AJAX işlemlerinde XML kullanmaya bir türlü ısınamamışımdır. Çünkü XML'in javaScript ile parse edilmesi uğraştırıcı olabiliyor. İşte tam bu noktada JSON imdadımıza yetişiyor. JSON olarak gelen verileri, herhangi bir parselleme işlemi yapmadan javaScript içerisinde kullanabiliyorsunuz. Bir de buna jQuery'nin getirdiği kolaylıkları eklersek değmeyin gitsin keyfimize 😃

JSON nedir?

XML dilinin çok farklı kullanım alanları olmasına karşın JSON ise yalnızca veri alış-verişi amacıyla oluşturulmuş bir veri biçimlendirme yöntemidir. Açılmış haliyle **J**ava**S**cript **O**bject **N**otation demektir. javaScript dilinin bir parçası olduğu için XML'den çok daha kolay ve hızlı bir şekilde işlenebilir. Ayrıca XML ile biçimlendirilmiş bir veri kümesini JSON ile biçimlendirdiğinizde daha az yer kapladığını görürsünüz.

Şimdi alttaki örneklere gözatalım. Elimizde 3 adet bilgisayar programının bilgileri var. Bu bilgiler XML ile ifade edildiğinde şöyle görünüyor olsun:

```xml
<programlar>
    <program>
        <isim>Zone Alarm</isim>
        <bilgi>bilgisayarın güvenliğini sağlar</bilgi>
        <adres>www.zonealarm.com</adres>
    </program>
    <program>
        <isim>Opera</isim>
        <bilgi>güvenli ve hızlı bir web tarayıcısıdır</bilgi>
            <adres>www.opera.com</adres>
        </program>
        <program>
            <isim>Photoshop</isim>
            <bilgi>güçlü bir imaj işleme yazılımıdır</bilgi>
            <adres>www.adobe.com</adres>
        </program>
    </programlar>
```

Şimdi de aynı bilgilerin JSON kullanılarak ifade edilmiş haline bakalım:

```json
{
   "programlar":[
      {
         "isim":"Zone Alarm",
         "bilgi":"bilgisayarın güvenliğini sağlar",
         "adres":"www.zonealarm.com"
      },
      {
         "isim":"Opera",
            "bilgi":"güvenli ve hızlı bir web tarayıcısıdır",
            "adres":"www.opera.com"
         },
         {
            "isim":"Photoshop",
            "bilgi":"güçlü bir imaj işleme yazılımıdır",
            "adres":"www.adobe.com"
         }
      ]
   }
```

Örneklere dikkatlice bakarsak JSON örneğinin daha okunabilir olduğunu söyleyebiliriz. Çünkü XML içinde bir sürü etiket açıp kapattığımız için veri kümemiz daha kalabalık ve karmaşık bir görüntü vermektedir. Ayrıca XML örneği yaklaşık 527 bayt büyüklüğünde iken JSON örneği 465 bayt büyüklüğündedir. Yani JSON örneği yaklaşık %12 daha az yer kaplamaktadır. %12'lik fark hemen size önemsiz gelmesin. Zira burada yalnızca 3 tane bilgisayar programın bilgilerini kullandık. Bu sayı 3 değil de 1000-5000 gibi büyük rakamlar olunca aradaki fark daha da büyüyecektir. Sonuçta büyük miktardaki verileri XML ile taşımaktan ziyade JSON ile taşıyarak hız kazancı elde etmeniz mümkündür.

jQuery ile JSON verilerini işlemek

jQuery kütüphanesi ile JSON tipindeki verileri iki yöntemle işleyebiliriz. Bunlardan ilki AJAX istekleri oluşturuyorken kullandığımız `$.ajax()` fonksiyonudur. Bu fonksiyonda **dataType** seçeneğini kullanıp değer olarak **json** ataması yaparsanız artık jQuery ile JSON verisi işleyebilirsiniz. Bir de JSON verilerini işlemek için özel olarak hazırlanmış `$.getJSON()` fonksiyonu vardır. Biz burada her iki yöntemi de ele alacağız.

Örneğimizde "program-bilgileri.php" isimli bir dosyamız mevcut. Bu dosyanın görevi, 3 adet bilgisayar programının bilgilerini JSON formatında vermek. Yani yukarıdaki JSON örneğinin aynısını kullanıyoruz 😉Sonra, bu dosyaya jQuery ile erişerek JSON tipinde gelen program bilgilerini işleyeceğiz.

```
1. $.ajax({
2.   url: 'program-bilgileri.php',
3.   dataType: 'json',
4.   success: function(JSON) {
5.        $('#sonuc').empty();
6.
7.        $.each(JSON.programlar, function(i, program){
8.             $('#sonuc')
9.             .append(JSON.program.isim +'<br />')
10.                 .append(JSON.program.bilgi+'<br />')
11.                 .append(JSON.program.adres+'<hr />');
12.            });
13.       }
14.    });
```

Bu örnekteki `$.ajax()` fonksiyonunu jQuery ve AJAX işlemleri yazısında ele almıştık. Buradaki ilk önemli nokta "dataType" seçeneği ile "program-bilgileri.php" dosyasından gelecek verinin JSON formatında olacağını bildirmektir. Sonraki önemli nokta ise alınan JSON formatındaki verilerin `$.each()` fonksiyonu kullanılarak tek tek işlenmesidir *(bu fonksiyona birazdan bakacağız)*. Örneğin çalışır halini Örnek 1 sayfasında bulabilirsiniz.

Şimdi de aynı işlemin `$.getJSON()` fonksiyonu ile nasıl yapıldığınına bakalım:

```
1. $.getJSON('program-bilgileri.php', function(JSON){
2.     $('#sonuc').empty();
3.
4.     $.each(JSON.programlar, function(i, program){
5.         $('#sonuc')
6.         .append(program.isim +'<br />')
7.         .append(program.bilgi+'<br />')
8.         .append(program.adres+'<hr />');
9.   });
10.    });
```

Eğer sık sık JSON tipindeki verilerle çalışacaksanız bu fonksiyon daha pratik gelecektir. Çünkü `$.ajax()` fonksiyonunda olduğu gibi her seferinde "dataType" ve "success" gibi seçenekleri belirtmek zorunda kalmazsınız. Örneği test etmek için Örnek 2 sayfasına bakabilirsiniz.

$.each() fonksiyonu ne işe yarar?

`$.each()` fonksiyonundan önceki yazılarda bahsetmediğim için şimdi bahsetmenin tam zamanı. Bu fonksiyonun görevi, javaScript'teki **object** veya **array** tipindeki verileri bir döngü

içerisinde işlemektir. `$.each()` fonksiyonu sayesinde ayrıca bir **for** döngüsü açmanıza gerek kalmaz. Şimdi elimizde şöyle bir object (nesne) olduğunu varsayalım.

```
1. var nesne={
2.     isim: "Erhan",
3.     soyisim: "BURHAN",
4.     yas: 24,
5.     web: "eburhan.com"
6. };
```

Bu nesnedeki özellikleri ve değerleri kullanabilmek için `$.each()` fonksiyonunu şöyle kullanıyoruz:

```
1. $.each(nesne, function(ozellik, deger) {
2.     $('#sonuc').append(ozellik+': '+deger+'<br />');
3. });
```

Sonuç olarak bu işlemin çıktısı alttaki gibi olacaktır. Sonucu test etmek için Örnek 3 sayfasına bakabilirsiniz. `$.each()` fonksiyonu ile ilgili daha ayrıntılı bilgiye ise şuradan ulaşabilirsiniz.

```
isim: Erhan
soyisim: BURHAN
yas: 24
web: eburhan.com
```

Sonuç...

JSON formatı artık kendisine geniş bir kullanım alanı bulmuştur. Bugün pekçok web servisini incelediğimizde API kaynaklarını JSON formatında da sunduklarını görüyoruz. Yahoo, Delicious, Flickr, YouTube, Getclicky... gibi büyük servisler buna sadece birkaç örnek. İşte bundan dolayı JSON formatı ile işlem yapmaya alışık olmalıyız.

Bu yazıdaki örnekleri buradan indirebileceğinizi hatırlatıp, JSON formatı hakkında daha fazla bilgi alabileceğiniz kaynaklardan bazılarına link vererek yazıyı noktalıyorum 😉

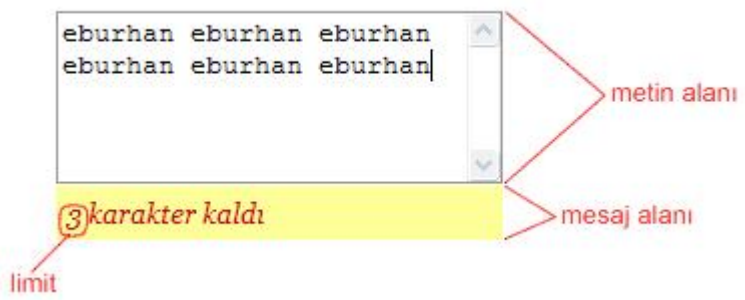
jQuery ile hazırladığım "Kalan Karakter" eklentisi

jQuery ile eklenti hazırlamanın önemli bir nedeni de değişik projelerde ihtiyaç duyulan aynı kodları, her seferinde kopyala-yapıştır yapmaktan kurtulmaktır. Ben de `textarea` gibi metin alanlarında kullandığım ve ziyaretçilerin kaç karakterlik yazı yazabileceklerini gösteren kodları bir eklenti haline getirdim. Eklentiye de **Kalan Karakter** ismini verdim 😉

Eklentinin özellikleri nelerdir?

1. Textarea ve Input gibi metin alanlarında kullanabilirsiniz
2. Karakter limitini istediğiniz gibi belirleyebilirsiniz

3. Ziyaretçiye gösterilecek mesajı istediğiniz gibi yazabilirsiniz
4. Ziyaretçiye gösterilecek mesaj için DIV gibi veya SPAN gibi istediğiniz bir HTML etiketi kullanabilirsiniz
5. Kullanımı çok kolaydır ve yalnızca 1 KB boyutundadır



Eklenti nasıl kullanılıyor?

Önce eBurhan Araçları sayfasından eklentiyi bilgisayarınıza indirin. İndirdiğiniz dosya içerisindeki "*jquery.kalanKarakter.js*" dosyasını jQuery kütüphanesi ile birlikte web sayfanıza alttaki gibi bağlamalısınız.

```
1. <script type="text/javascript" src="jquery.js"></script>
2. <script type="text/javascript"
   src="jquery.kalanKarakter.js"></script>
```

Bu işlemden sonra eklentiyi kullanmaya başlayabiliriz. Diyelim ki web sayfamızdaki bir iletişim formunda, ID özniteliği mesaj olan bir textarea elemanı bulunsun. Alttaki gibi:

```
1. <textarea id="mesaj" cols="25" rows="5"></textarea>
```

Bu metin alanına 50 karakterlik bir limit koymak için yazmamız gereken kodlar şöyle olmalıdır:

```
1. $('textarea#mesaj').kalanKarakter({
2.     limit: 50,
3.     mesaj: 'kalan karakter: #1',
4.     kapsa: '<div></div>',
5.     uyari: function(){
6.         alert('karakter limiti aşıldı !');
7.     }
8. })
```

İşte hepsi bu kadar Çalışır halini görmek için demo sayfasına bakabilirsiniz.

Eklentinin hangi seçenekleri var?

Yukarıdaki örnek uygulamada *limit, metin, kapsa, uyari* gibi seçenekler kullandık. Bunlar, eklentinin seçenekleridir ve 4 tanedir. Bu seçenekleri tanımlamak zorunda değilsiniz zira tanımlamadığınız zaman **varsayılan** ayarlar kullanılacaktır ve eklenti çalışmaya devam edecektir. Şimdi bu 4 adet seçeneğe kısaca gözatalım:

1. **limit:** metin alanına toplam kaç karakter yazı yazılabileceğini tanımlar. 500, 1000 gibi sayısal değerler verilmelidir. Varsayılan değeri 250 olarak tanımlıdır.

2. **mesaj:** metin alanının altında veya yanında ziyaretçiye gösterilecek olan mesajı tanımlar. #1 parametresi, limit seçeneğinin değerini işaret eder.
3. **kapsa:** mesajın gösterileceği alan için kullanılacak olan HTML etiketini tanımlar. DIV, SPAN veya EM gibi etiketler kullanabilirsiniz fakat etiketin başlangıçını ve bitişini mutlaka belirtmelisiniz. Örneğin etiket olarak EM kullanacaksanız bu seçeneğe `<em></em>` şeklinde bir değer girmelisiniz. Ayrıca bu etikete class ataması da yapılabilir. Örnek vermek gerekirse `<em class="kalanKarakter"></em>` formatında bir değer girebilirsiniz. Böylece CSS kullanarak mesaj alanını kolayca biçimlendirebilirsiniz.
4. **uyari:** karakter limiti aşıldığında bir fonksiyonun çalıştırılmasını sağlar. Örneğin bu seçenek ile "*alert()*" gibi javaScript fonksiyonlarını kullanarak ziyaretçiyi uyarabilirsiniz veyahut Css Message Boxes gibi uygulamaları tetikleyebilirsiniz.

Sonuç

Eklentinin kurulumu ve kullanımı çok kolay fakat yine bir **video sunumu** hazırladım. Alttaki ekran görüntüsüne tıklayarak videoyu online izleyebilirsiniz ve **Kalan Karakter** eklentisini daha yakından tanıyabilirsiniz. Eklentinin faydalı olması dileğiyle... 😉

jQuery ile “çek-bırak” özellikli nesneler oluşturmak

2. Yıl teması'nda küçük bir atraksiyon olsun diye, sol üst köşedeki rozet resmini jQuery kullanarak hareketlendirmiştim. İşleyişi gayet basit. Mouse *(fare)* ile nesneyi tutup çekiyorsunuz ve bıraktığınız zaman eski yerine geri dönüyor. Basit birşey olmasına rağmen nedense çok ilgi gördü ve bunun nasıl yapıldığını anlatmamı isteyenler oldu. Bu yazıyı da onlar için özel hazırladım 😉

Ben bu atraksiyonu ilk defa moo.fx sayfasında görmüştüm. Siz de bu sayfaya girerek sağ-üst köşedeki 3 KB yazan nesneyi inceleyebilirsiniz. Bu sayfadaki kodlara gözgezdirmiştim ve aynı şeyi jQuery kullanarak yapmayı denemiştim. Aslında 3 basit adımda siz de buna benzer atraksiyonlar yapabilirsiniz ve ziyaretçilerinizi ufak da olsa etkileyebilirsiniz. Şimdi aşağıdaki adımları takip ederek bu işlemi yapmaya başlayalım:

1. Html kodlaması
2. Css ile biçimlendirme ve konumlandırma
3. jQuery ile nesneye hareket kazandırma

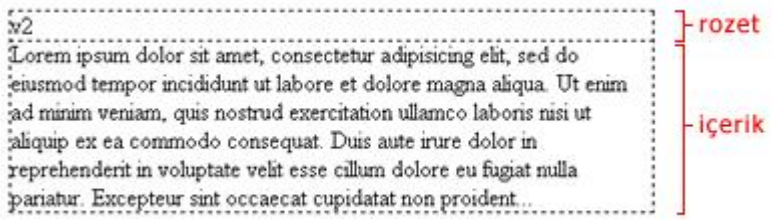
1. Html kodlaması

İlk adımda bir **içerik** nesnesi belirteceğiz. Daha sonra da bu içerik nesnesinin içerisine, hareketlendirmek istediğimiz **rozet** nesnesini yerleştireceğiz. Şimdilik "rozet" nesnesinin içerisinde yalnızca **v2** yazısı görünüyor ama daha sonra bunun arkaplanına css yardımıyla bir rozet resmi atayacağız.

```
1. <div id="icerik">
2.     <div id="rozet">v2</div>
3.     Lorem ipsum dolor sit amet, consectetur adipisicing
   elit,
```

```
4. sed do eiusmod tempor incididunt ut labore et dolore
   magna aliqua.
5. Ut enim ad minim veniam, quis nostrud exercitation
   ullamco laboris
6. nisi ut aliquip ex ea commodo consequat. Duis aute irure
   dolor in
7. reprehenderit in voluptate velit esse cillum dolore eu
   fugiat nulla
8. pariatur. Excepteur sint occaecat cupidatat non
   proident...
9. </div>
```

Bu adımda yaptığımız işlemlerin sonucunu Adım 1 sayfasında veya alttaki ekran görüntüsünde görebilirsiniz.



2. Css ile biçimlendirme ve konumlandırma

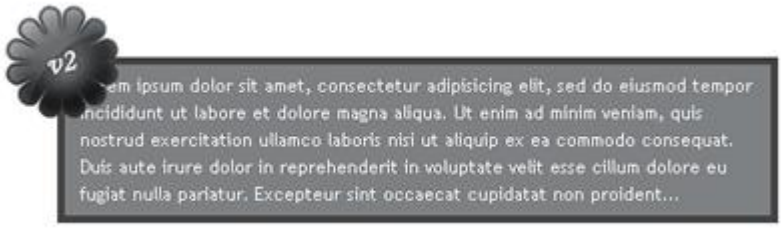
İlk adımda, hareketlendirmek istediğimiz "rozet" nesnesini ve bunu içerisinde tutan "içerik" nesnesini oluşturmuştuk. İkinci adımda bu nesneleri css kullanarak biçimlendireceğiz. Ayrıca "rozet" nesnesinin yerini css ile konumlandırma (positioning) konusundan faydalanarak "içerik" nesnesinin sol-üst köşesine taşıyacağız. Şimdi bu işlemleri yapan css kodlarına gözatalım:

```
1. <style type="text/css">
2.     div#icerik {
3.         position: relative;
4.         margin: 0 auto;
5.         width: 450px;
6.         padding: 5px 10px;
7.         border: 5px solid #444;
8.         background-color: #7f8183;
9.         color: #fff;
10.            font: normal 12px/18px "Trebuchet MS";
11.        }
12.
13.        div#rozet {
14.            position: absolute;
15.            width: 77px;
16.            height: 77px;
17.            top: -40px;
18.            left: -40px;
19.            text-indent: -9999px;
20.            background: url(rozet.png) 0 0 no-repeat;
21.            /* IE 6 için saydamlık probleminin çözümü */
22.            _background: transparent;
```

```
23.             _filter:
   progid:DXImageTransform.Microsoft.AlphaImageLoader(src='r
   ozet.png', sizingMethod='scale');
24.         }
25.     </style>
```

Bu kodlar yardımıyla "icerik" nesnesini **relative** olarak, "rozet" nesnesini ise **absolute** olarak konumlandırdık. Daha sonra "rozet" nesnesinin genişlik ve yükseklik değerlerini, arkaplan olarak olarak kullandığımız "rozet.png" dosyasının genişlik ve yükseklik değerlerine eşitledik *(77px)*. Hemen ardından top ve left değerleri ile "rozet" nesnesinin, "icerik" nesnesinin sol-üst köşesine yerleşmesini sağladık *(-40px)*. Son iki satırda ise IE 6'nın png dosyalarıyla yaşadığı saydamlık problemine çözüm getirmiş olduk. Bu problem ile ilgili ayrıntılı bilgi edinmek isterseniz Internet Explorer 6 için saydam PNG desteği sayfasını okumalısınız.

Bu adımda yaptığımız işlemlerin sonucunu Adım 2 sayfasında veya alttaki ekran görüntüsünde görebilirsiniz.



3. jQuery ile nesneye hareket kazandırma

Yapacak tek bir işlem kaldı: jQuery ve 2 adet eklenti yardımıyla "rozet" nesnesine hareket kazandırmak... Bu adımda bize EasyDrag ve Easing eklentileri de lâzım olacak. O yüzden bu eklentileri indirip jQuery ile birlikte web sayfamıza dahil etmeliyiz. Bu dahil etme işlemini şöyle yapabilirsiniz:

```
1. <script type="text/javascript" src="jquery.js"></script>
2. <script type="text/javascript"
   src="jquery.easydrag.js"></script>
3. <script type="text/javascript"
   src="jquery.easing.js"></script>
```

Artık herşey hazır olduğuna göre rozet nesnesini hareketlendirecek kodlarımıza bakalım:

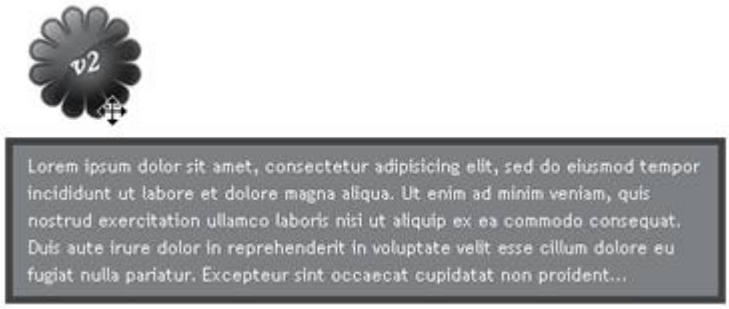
```
1. <script type="text/javascript">
2.     $(document).ready(function(){
3.
4.         $("div#rozet").easydrag();
5.
6.         $("div#rozet").ondrop(function(){
7.             $('div#rozet').animate(
8.                 { top:'-40px', left:'-40px'},
9.                 {duration: 1000, easing: 'easeOutElastic'}
10.                 );
11.             });
12.
13.         });
```

```
14.     </script>
```

Öncelikle `$("div#rozet").easydrag();` kodu ile rozet nesnesinin sürüklenebilir olmasını sağlıyoruz. Sonraysa `$("div#rozet").ondrop();` kodunu kullanarak sürüklenen nesne **bırakıldığında** herhangi bir işlem yapılmasını belirtiyoruz. Bu iki kod EasyDrag eklentisinin bize sağladığı özelliklerdir.

Bu işlemlerden sonra `ondrop()` fonksiyonu tetiklendiği anda rozet nesnesinin ilk **top** ve **left** değerlerine geri dönmesini jQuery'nin yerleşik fonksiyonu olan `animate()` ile sağlıyoruz. Bu işlem 1000 milisaniyelik bir sürede gerçekleşiyor. Ayrıca bu esnada rozet nesnemize Easing eklentisinin **easeOutElastic** isimli efektinin uygulanmasını belirtiyoruz. Tüm bu işlemleri yapmamızı sağlayan `animate()` fonksiyonu hakkında detaylı bilgiyi jQuery ve Efekt işlemleri isimli yazımda bulabilirsiniz.

Bu son adımda yaptığımız işlemlerin sonucunu Adım 3 sayfasında görebilirsiniz.



Sonuç

Gördüğünüz gibi son derece kolay bir işlem. Dikkat etmemiz gereken şey, ilk başta belirlemiş olduğumuz adımları tek tek ve sırayla uygulamaya geçirmiş olmamızdır. Aslında basit gibi görünen bu konuda pekçok bilgiyi de içiçe kullanmış olduk. O yüzden arada sırada böyle atraksiyonlar hazırlamak, öğrendiğimiz bilgileri somut olarak kullanma adına faydalı oluyor. Siz de mutlaka okuduğunuzla kalmayıp, buradaki işlemleri kendi kendinize uygulayınız. Bu yazıyla ilgili dosyaları ise bu bağlantıdan bilgisayarınıza indirebilirsiniz.

ÖRNEKLER

Kütüphanemizi Sayfamıza Ekleyelim

```
<script type="text/javascript" src="jquery.js"></script>
```

Element Seçimleri

```
$('div')  // tüm divleri seçmemizi sağlar...
$('div.baslik')  // baslik sınıfını seçmemizi sağlar...
$('div#baslik')  // baslik id'sini seçmemizi sağlar...
```

Javascript kodlarının çalıştırılması

```
$(document).ready(function() {
  // Sayfa yüklendiğinde otomatik olarak çalışmasını istediğimiz javascript
kodlarını buraya yazıyoruz...
```

```
});
```

Kodumuzu küçük bir örnekle açıklayalım

```
$(document).ready(function() {
  $("a").click(function() {
    alert("Hello world!");
  });
});
```

Sayfadaki herhangi bir bağlantıya tıklandığında alert verilmesini sağladık