jQuery

Tercüme:

Merve KARADAYI ***Burcu SAYİN***

Editör:

Yiğit Ot

Kaynak:

http://www.wowebook.info/book/jquery-novice-to-ninja/

BÖLÜM 1: GİRİŞ

Not: Buradaki örneklerin kodlarına http://www.sitepoint.com/books/jquery1/code.php?&SID=fdce5013f27f547c53af2c7d538c7815 linkinden ulaşabilirsiniz.

Neden Jquery?

Kendi kütüphanemizi oluşturmak mı Javascript kütüphanelerinden birini kullanmak mı daha iyidir?

Çapraz Tarayıcı Uyumluluğu(Cross-browser Compatibility)

Jquery, öğrenmek zevkli olmasının yanısıra bir çok tarayıcıyla uyumlu çalışır.. Daha önce JavaScript ile ilgilenmiş olanlar çapraz tarayıcı tutarsızlıklarına şahit olmuşlardır. Örneğin, günlerinizi harcayarak geliştirdiğiniz bir arayüz bütün ünlü tarayıcılarda çalışıyorken Linux üzerinde çalışan Opera'da çalışmıyor ve alıcı sadece Opera kullanıyor. Bu tip problemlerin izini sürmek(bulmak) kolay olmamakla birlikte tamamen sorunu çözebilmek çok daha zor.

Jquery oluşabilecek bu tür problemlerin farkındadır ve neden bu tür sorunların yaşandığını anlamaktadır. Bu bilgiler kütüphaneye dahil edilmiştir ve bizim için uyarılar içermektedir. Yazdığımız kodların çoğu Internet Explorer 6 dahil yaygın kullanılan tüm tarayıcılarda aynı şekilde çalışacaktır.

Jquery'nin sağladığı bu faydalar zaman tasarrufu sağlar. Böylece, program yazarken fikirlerimizi gerçekleştirebilmek için daha fazla zamana sahip oluruz çünkü sunucu hatalarını çözme işini Jquery'e bırakırız.

CSS3 Seçiciler (CSS3 Selectors)

Jquery'nin bize sunduğu bir diğer olanak ise CSS3 Seçiciler'in özelliklerini destekliyor olması.
CSS3 Seçiciler ile ilgili önceden bilgi sahibi olmanız kod yazarken size çok büyük kolaylık sağlayacaktır, çalışmanız için size daha fazla araç verir.

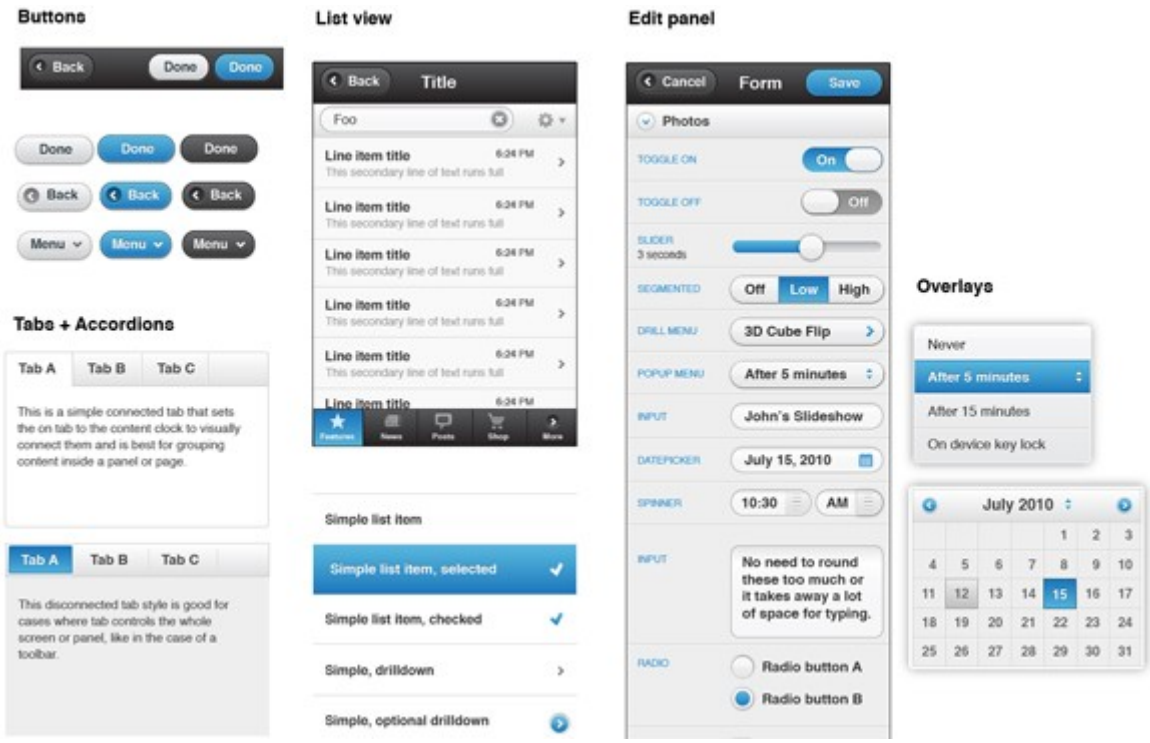
Faydalı Araçlar/Sunulan olanaklar

Jquery yazmak için faydalı olacak çok çeşitli fonksiyonlar var: dize kırpma, nesneleri(objeleri) kolayca genişletebilme ve daha fazlası... Bu fonksiyonları ayrı ayrı yeterince kullanışlı ayrıca bu fonksiyonlarla Jquery ve Javascript arasındaki birleştirme(entegrasyon) kolaylıkla sağlanabiliyor ve bu sayede kod yazılabilirliği ve sürdürülebilirliği kolaylaşıyor.

JQuery UI (JQuery Kullanıcı Arayüzü)

JQuery etkileyici görsel bileşenler ve etkiler yapmak amacıyla oluşturulmuştur. Bunların çoğu ana Jquery kütüphanesinde mevcuttur fakat Jquery grubu daha yüksek seviyeli yapıları ayırıp ayrı bir pakette toplayarak bunu muntazam bir kütüphane haline getirip Jquery yapısının en üst katmanına yerleştirmişlerdir. Bu katmana(kütüphaneye) JQuery UI (JQuery Kullanıcı Arayüzü) adı verilir.

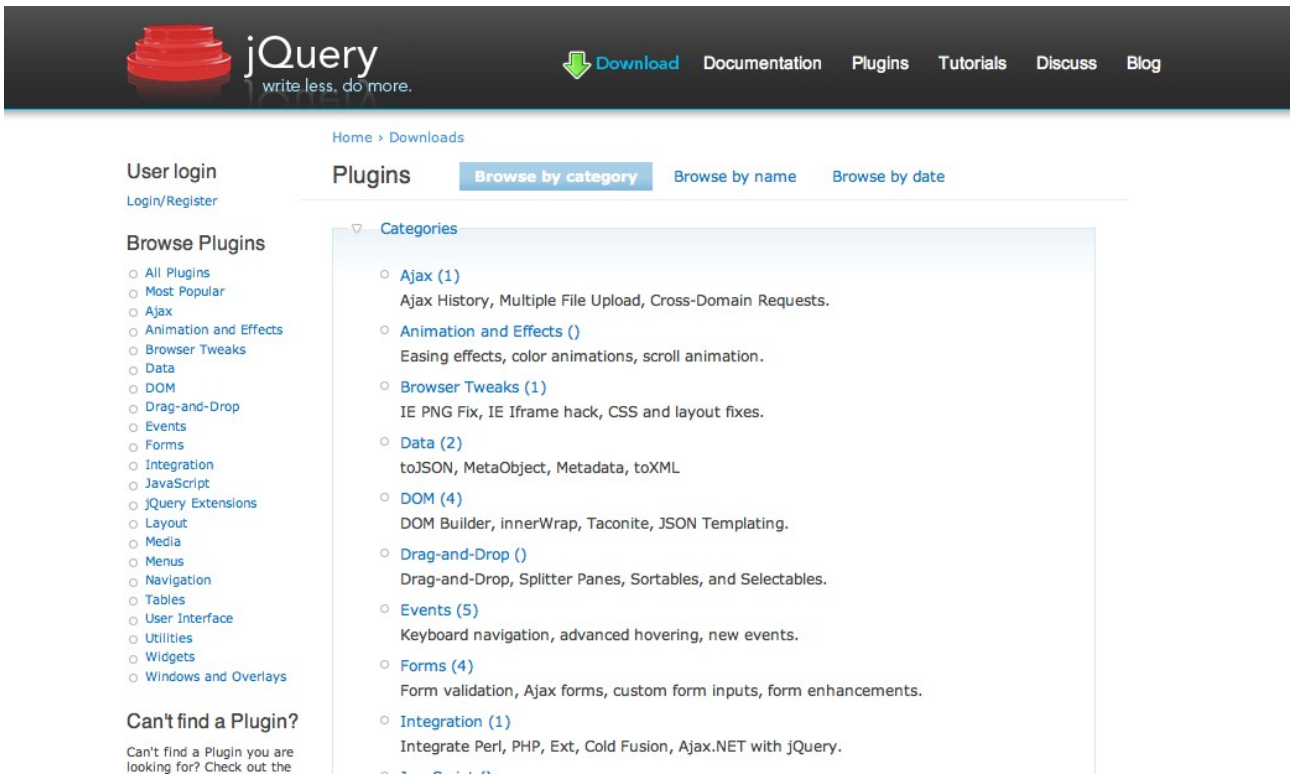
JQuery UI görsel bileşenlerini daha iyi anlayabilmek için bu resmi inceleyebilirsiniz:



Eklentiler(Plugins)

JQuery'de istediğimiz gibi eklentiler oluşturup bunları bütün JQuery projelerimizde tekrar tekrar kullanabilir ve diğer program geliştiren kişilerle rahatlıkla paylaşabiliriz.

Çoğu kullanıcı Jquery'nin genişletilebilirliğinin farkında ve bunu kullanabiliyor. Nasıl mı? JQuery deposunda şu an yüzlerce seçkin eklenti JQuery eklenti deposunda(The jQuery plugin repository) indirilebilir halde bekliyor ve sürekli yeni eklentiler de eklenmekte.



Biçimlendirmenin Temiz Tutulması(Keeping Markup Clean)

JQuery satır içi kod yazmaya izin vermiyor, bu ayrım bize daha temiz, daha düzenli kodlar yazma olanağı sunuyor. Böylece kod yazarken yanlış yapmamız neredeyse engellenmiş oluyor.

JQuery aynı zamanda, var olan HTML kodu karmaşasıyla kısıtlanamaz. Yani, kullanışlı fonksiyonlar aracılığıyla yeni sayfa elemanları veya doküman parçaları eklenebilir. Sayfanın herhangi bir yerine yeni HTML parçası eklemeleri yapılabilir. Ayrıca herhangi bir şeyi silme, çıkarma işlemleri de rahatlıkla yapılabilir.

Yaygın olarak kabul görmesi (Widespread Adoption)

IBM, Netflix, Google ve hatta Microsoft günümüzde JQuery teknolojisini kullanmakta.

Dezavantajları neler?

JQuery'nin dezavantajları konusunda insanlar pek çok görüşe sahip. Kimisi Javascript kütüphanesi kullanmanın hızı düşürdüğünü iddia ediyor çünkü sayfanın o kütüphaneyi yüklemesi için belli bir zaman geçmesi gerektiğini söylüyorlar.

İlk olarak, JavaScript kütüphanesi temel olarak sadece 19 KB yer kaplıyor. İkincisi, JQuery grubu hızı günden güne daha da arttırmaya çalışıyor, sürekli güncellemeler olmakta.

JavaScript kütüphanelerini karşılaştırdığımızda, DOM(Belge Nesnesi Modeli) idaresi, etkiler/uygulamalar eklenmesi ve Ajax istekleri yapma konusunda en iyisi Jquery.

JQuery İndirme ve Dahil Etme

JQuery ile çalışmaya başlamadan önce, kodun güncellenmiş son versiyonunu bulmalı ve internet sayfamıza eklemeliyiz. Bunu yapmanın birkaç yolu var. Hangisini seçersek seçelim HTML sayfamıza JQuery'i dahil etmeliyiz(zaten başka bir JavaScript kaynak dosyamız olsa bile).

Not: Her ne kadar farklı gibi gözükse de JQuery'nin de bir JavaScript kütüphanesi olduğunu unutmamalıyız. Bu sebeple, JavaScript'le yapamayacağımız bir işi JQuery ile de yapamayacağımızı bilelim.

JQuery indirme

Jquery'nin internet sitesinde her zaman güncellenmiş hali mevcut, oradan indir bağlantısına tıklayıp JavaScript dosyasını yeni bir çalışma dosyası oluşturup oraya kaydetmeliyiz. Bu dosyayı oluştururken HTML dosyalarımızın görebileceği bir yer olmasına dikkat etmeliyiz(genellikle sitenizin ana dökümanının altındaki scriptler ya da JavaScript dizininin içine). Çok basit bir örnekle açıklayalım:
JQuery içeren çok basit bir HTML dosyası şu şekilde hazırlanabilir:

```
<head>
<title>Merhaba JQuery Dünyası!</title>
<script type='text/javascript' src='jquery-1.4-min.js'></script>
<script type='text/javascript' src='script.js'></script>
</head>
```

Koddaki ilk script etiketi JQuery kütüphanesini yüklüyor, ikinci etiket ise script.js dosyasını işaret ediyor. Bu dosya bizim kendi JQuery kodlarımızı çalıştıracağımız yer. Şimdi JQuery kullanmaya hazırız!

Jquery'yi internet sayfamıza eklemenin birkaç yolu var demiştik, bunlardan en çok kullanılanı indirme işlemidir. Daha profesyonel çalışmak isteyenler diğer yollara da

başvurabilir: Google CDN (Google İçerik Dağıtım Ağı) gibi.

Hangi Jquery Paketini Kullanmalıyız?

Çoğumuz sıkıştırılmış paketi indirmeyi tercih eder çünkü daha az yer kaplar, bant genişliği maliyetlerini düşürür ve son kullanıcı için sayfa isteklerinin hızını arttırır. Dezavantajı ise okunulabilirliği azdır çünkü tüm boşluklar silinir, değişken isimleri kısaltılır vs.

Bir JQuery Senaryo anatomisi

Jquerynin yazım kuralı ilk bakışta biraz tuhaf gelebilir oysa tutarlı ve uygundur. İlk birkaç kod parçasından sonra eliniz alışacak ve daha fazla yazmak isteyeceksiniz...

JQuery Takma Adı (The jQuery Alias)

Kendi JavaScript kodumuzda veya diğer kütüphanelerle bir isim çakışması varsa, yani aynı fonksiyon adı farklı yerlerde farklı şekillerde geçiyorsa bu bir sorundur. Bu sorunu giderebilmek için Jquery, kütüphaneye ulaşmak için basit bir kısaltma kullanıyor: $ işareti. İlk bakışta bu işaret çok anlamsız gibi gözükse de bütün bir kodu düşündüğümüzde bir çok defa kütüphanenin çağrıldığını görebiliriz ve bu kısaltma sayesinde okunabilirliği arttırdığı açıktır.

Bir JQuery Komutunu İnceleme

JQuery komutları bir jQuery fonksiyonunu veya onun takma adını(alias) çağırmakla başlar. Şimdi JQuery komutlarının diğer bileşenlerini inceleyelim.

Selector	Action	Parameters
jQuery('p')	.css	('color' , 'blue');
$('p')	.css	('color' , 'blue');

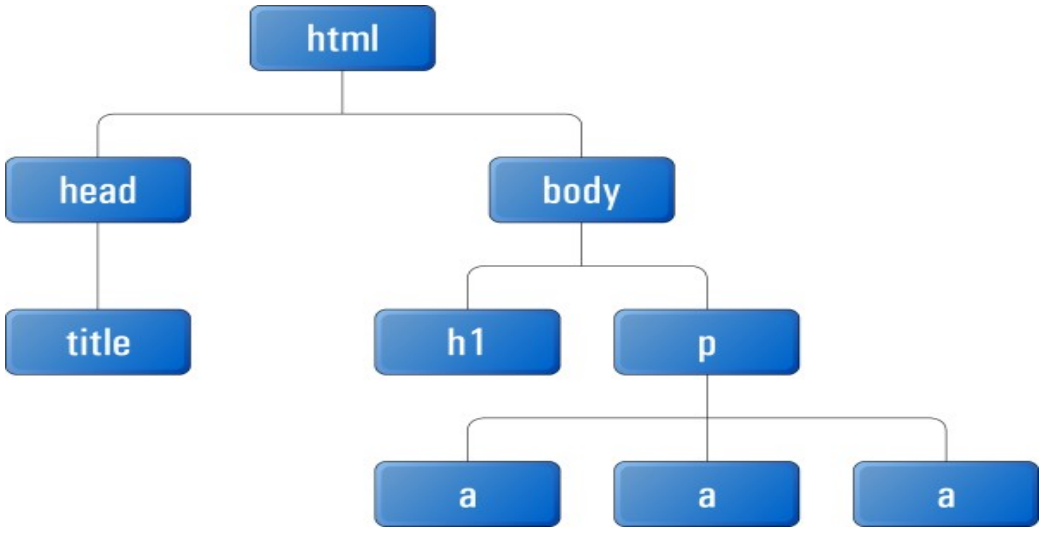
Her komutun 4 parçası vardır. JQuery fonksiyonu(ya da onun alias'ı), seçiciler(selectors), faaliyetler(action) ve parametreler. Şimdi bu kavramları açıklayalım. Fonksiyonun ne olduğunu hepimiz biliyoruz. Diğerlerini inceleyecek olursak:
İlk olarak, internet sayfamızda bir veya birden fazla eleman seçmek için seçicileri kullanırız. Sonra seçtiğimiz bu elemanlar için uygulayacağımız faaliyeti(action) seçeriz. Son olarak da bu seçilen faaliyetleri nasıl uygulamak istediğimize dair parametreler belirleyip JQuery'i bilgilendiririz.

JQuery kodu gördüğümüzde parçalara ayırıp incelersek daha kolay anlayabiliriz.

Yukarıda verilen örnekte, seçiciye bütün paragraf etiketlerini seçmesini söyledik(HTML'deki <p>). Sonra JQuery'nin css faaliyetini seçtik. Bu faaliyet, başlangıçta seçilen paragraf elemanlarının CSS özelliklerini değiştirmek için kullanılır. Son olarak da, CSS renk ('color') özelliğini mavi yapabilmek için bazı parametreler yazdık. Sonuç olarak bütün paragraflarımız mavi(blue) renk oldu! Bu örneğimizde biz sadece 2 tane parametre yazdık fakat bu her zaman böyle olmayabilir, bazı fonksiyonlar hiç parametre almazken bazıları 3, 4, hatta daha fazla parametre alabilir.

Belge Nesnesi Modeli(DOM)

DOM aslında özellikle JQuery ile ilgili bir kavram değil. DOM, bütün sunucu yapımcılarının kolaylıkla takip edebileceği, anlayabileceği, nesnelerin HTML içinde gösterilmesinin standart bir yoludur.
DOM aslında HTML biçimlendirmesinin hiyerarşik bir gösterimidir. Her elemanın bir ebeveyni(üst öğesi) vardır(container olarak adlandırılır). Aynı zamanda bir ya da birden fazla çocuk elemanları olabilir. Her elemanın bir numarası(id) vardır ve aynı zamanda bir ya da birden fazla sınıfın özelliğini taşıyabilir. Bu atamaları biz kendi HTML kaynak kodumuzda yaparız. Sunucu bir HTML sayfasını okuduğunda DOM'u oluşturur. Basit bir DOM yapısı örneğini şekilde görebilirsiniz:



Bu örnekte body'nin iki çocuğunun olduğunu, h1 ve p'nin kardeş olduğunu vs. net bir şekilde görebiliyoruz. Buradaki her bir elemana bir numara vererek her birinin tek(eşsiz) olmasını sağlayabiliriz. Bu sayede ulaşmak istediğimiz elemana herhangi bir anda rahatlıkla ulaşabiliriz.

BÖLÜM 2: SEÇMEK, DEKORASYON VE GELİŞTİRME

İlk bölümde bir JQuery komutunun anatomisini incelemiştik, şimdi ise JQuery'yi bir faaliyete koymak için gerekli olan adımları göreceğiz. Sayfa hazır olana kadar bekliyoruz, etiketimizi(tag) seçiyoruz ve onu değiştiriyoruz.

Sayfanın Hazır Olduğundan Emin Olmak

Sayfadaki HTML elemanlarıyla etkileşime geçmeden önce, onlar yüklenmiş olmalı. Biz ancak onlar sayfadayken üzerlerinde değişiklik yapabiliriz. Eskiden herhangi bir script'i çalıştırmadan önce bütün sayfanın yüklenmesini beklemek tek güvenilir yoldu. Ama şimdi, uygulamalarımızın daha hızlı kullanılabilir hale gelmesi için JQuery sabit bir olaya(event) sahip.

```
$(document).ready(function() {
alert('Sayfanız hazır!');
});
```

Burada bizim seçicimiz(selector) document, faaliyetimiz(action) ready, parametremiz ise bir fonksiyon (alert).

JQuery'nin Temeli: Seçme İşlemi

Elimizde bir tablo olduğunu düşünelim. Bu tabloda ünlü kişilerin isimlerinin olduğunu varsayalım. Müşterimizin bize ilettiği problem şu: tablo çok geniş olduğu için aradığımız ünlünün referans numarasına ulaşmakta zorluk çekiyoruz. Bizden isteği ise şu: aralardaki satırları açık gri yapalım ve böylece biz de referans numarasına kolaylıkla ulaşabilelim(bir satır açık gri, bir satır siyah).

Sayfamız bizim için hazır duruma geldi. Şimdi, etiket seçme işlemini yapmalıyız. Sayfada değiştirmek istediğimiz elemanı seçmek tamamen JQuery'nin işi! İyi bir iş çıkarabilmek için üzerinde işlem yapmak istediğimiz elemanları ustaca ve kolaylıkla seçebilmek çok önemli.

Bizim bütün seçicilerimizin JQuery fonksiyonunda ya da onun takma adı(alias) olan $'da paketlendiğini daha önce söylemiştik.

jQuery(<seçiciler burada>) ya da $(<seçiciler burada>)

Bundan sonra hep “$” işaretini kullanacağız.

Basit Seçme İşlemi

Seçme işlemini yaparken mümkün olduğu kadar açık olmalıyız, istediğimiz en öz kısma odaklanmalıyız ki bize istediğimiz şeyi döndürebilsin.

Tablonun bir kısmını yukarıdaki şekilden görebilirsiniz. Orjinal haline en başta verilen linkten ulaşabilirsiniz. Tabloyu incelediğimizde, seçeceğimiz kısmı satır olarak belirler ve $('tr') yazarsak hiçbir şey elde edemeyiz çünkü bu çok geniş bir kavram olur(tablodaki bütün satırları seçmiş oluruz). Benzer şekilde, eğer her paragrafı, div elemanını veya h1 başlığını seçmek istersek, şu şekilde yazmalıyız:

$('p')
$('div')
$('h1')

Ama biz tablodaki her satırı değiştirmek istemiyoruz, sadece ünlü bilgisine sahip olan satırı değiştirmek istiyoruz. Bu yüzden daha açık olmalıyız. İlk olarak ünlü listesini içeren elemanı seçmeliyiz. Şekle baktığımızda, tablonun kendisi veri sınıfına(class of data) sahipken ünlü tablomuzu içeren “div” ünlülerin numaralarına(id of celebs) sahip. Tabloyu seçerken bu ikisinden birini kullanabiliriz. Numara(id) ile seçmek için # işareti ve ardından elemanın numarasını yazarız ve bunu dize(string) olarak jQuery fonksiyonuna ekleriz:

$('#celebs')

Buradaki format CSS'in “id selector” formatıyla aynı, sadece bir elemanın geri döndürülmesini bekliyoruz çünkü numaraların(id) eşsiz olduğunu söylemiştik. Benzer şekilde, sınıf ile seçme işlemi için CSS'in selector sınıfını kullanabiliriz. Dizeyi “.” ve ardından elemanın ait olduğu sınıf ismiyle jQuery'e yazarız:

$(' .data ')

Bu iki komut da bizim için tabloyu seçecektir fakat daha önce de söylediğimiz gibi, DOM'dan bahsettiğimizde, bir sınıf bir çok eleman arasında paylaşılabilir ve jQuery bizim işaret ettiğimiz sınıftaki tüm elemanları bize döndürebilir. Bu sebeple numara ile seçim yapmak daha belirgin bir sonuç almamızı sağlayacaktır.

Seçimimizi Daraltmak

Tablomuzu başarıyla seçtik(tablonun tamamı bizi ilgilendirmemesine rağmen), tablonun içindeki her bir satırı istiyoruz. Bunu yapabilmek için tüm satırları içeren ana yapı ile onun içinde barındırdığı tüm satırlar anlamına gelen komutu arasında bir boşluk bırakarak şu şekilde yazıyoruz:

$('#celebs tr')

Bu yapıyı aradığınız alt seviyedeki verilere ulaşabilmek için kullanabilirsiniz ama daha duru bir anlatım için seçicileri mümkün olduğunca kısa ve öz tutmaya çalışmalısınız.

Bunu açıklayabilmek adına bir adım daha ileriye gidelim. “p” elemanının içindeki tüm “span” elemanlarını seçmek istediğimizi düşünelim. “p” elemanının kendisinin de “div” elemanının içinde olduğunu biliyoruz. Bu elemanlar da “fancy” sınıfına sahip. O halde seçiciyi şu şekilde kullanabiliriz:

$('div.fancy p span')

Eğer bunu takip ederseniz, herhangi bir şey seçmeye hazırsınız demektir!

Seçimimizi Test Etme

Seçimimizi test edebilmemizin en basit yolu “length” özelliğini kullanmak. “length” o anda seçiciyle eşleşen eleman sayısını döndürür. Elemanlarımızın seçildiğinden emin olabilmek için bir uyarı komutu ile “length”i şu şekilde birleştirebiliriz:

```
$(document).ready(function() {
   alert($( '#celebs tr' ).length + '  eleman! ' );
});
```

Bu uyarı bize ünlü tablosunun 7 elemanının olduğunu söyleyecek. Bu bizim beklediğimiz bir cevap olmayabilir çünkü tabloda 6 ünlü var! Ama HTML'e baktığımızda problemi çözeceğiz: tablo header'ı da bir satırdır(tr). Bu yüzden toplamda 7 satırımız var! Seçimimizi daraltmak adına “tbody” elemanının içindeki

satırları bulmak için şöyle bir kod parçası hazırlayabiliriz:

```
$(document).ready(function() {
   alert($( ' #celebs tbody tr ' ).length +  '  elements! ' );
});
```

Bu doğru satır sayısını gösterecektir(6).
Eğer uyarı 0 gösterirse, seçimimizde bir sorun var demektir. Böyle bir sorunu çözmenin en güzel yolu seçimimizi daha küçük ve basit hale getirmektir.

Bizim örneğimizde, basitçe $('#celebs ') yazabiliriz. Böylece sadece tablo elemanlarını ve 1 uzunluğundaki bir uyarıyı seçmiş oluruz.

Filtreler

Bütün tablo satırlarımızı başarıyla seçtiğimiz bilgisiyle, seçimimizi alttaki her bir satıra doğru daraltmak çok kolaydır çünkü jQuery bunu yapmak için filtreye sahip! Filtre, belirli parçaları siler ve sadece bizim istediğimiz kısımları bırakır.

```
$(document).ready(function() {
  alerts($( ' #celebs tbody tr:even' ).length  +  '  elements!' );
});
```

Filtreler, filtrelemek istediğimiz parçalara bağlanır(bu örnekte bu parçalar tablonun satırları) ve filtre ismiyle takip edilen bir sütun ile tanımlanır. Burada “:even” filtresi her çift indeksli elemanı tutup gerisini silmek için kullanılır. Şimdi seçim length'imizi uyarı olarak gösterdiğimizde 3 sayısını göreceğiz. Seçimimizde tüm tek sayılı satırlar filtrelenmiş yani silinmiş oldu. Bizim kullanımımıza açık olan geniş bir jQuery seçici dizisi vardır. :odd, :first, :last, :eq() (mesela üçüncü elemanı seçmek için) ve daha fazlası mevcuttur.

Çoklu Elemanlar Seçmek

Tek bir komut içerisinde birden fazla eleman seçebiliriz. Bu işlemi yapabilmek için, seçici dizelerini virgülle ayırarak yazmalıyız. Örneğin her bir paragrafı, “div” elemanını, h1 başlığını ve giriş kutusunu(input box) seçmek istersek, seçiciyi şu şekilde kullanmalıyız:

```
$( ' p, div,h1,input ' )
```

İyi Bir Seçici Olmak

Seçme işlemi çok basit gibi gözüküyor olabilir, şu ana kadar evet öyle. Ama biz

sadece yüzeysel olarak baktık işe. Bir çok durumda en temel bilgiler tüm ihtiyacımız olan şeydir, eğer basitçe bir elemanı veya birbiriyle ilgili elemanları etiketlemeye çalışıyorsanız; elemanın ismi, numarası ve sınıfı bizim için en çok gerekli olanlarıdır.

Verilen bir elemanın DOM'u üzerinden gidecek olursak, durum biraz daha zorlaşabilir. JQuery DOM'u dolaşırken(traversing) bize sayısız faaliyet ve seçici sunmakta. “Dolaşmak” kelimesinin buradaki anlamı; sayfa hiyerarşisini yukarıdan aşağıya ve aşağıdan yukarıya (çocuklar ve ebeveynler boyunca) dolaşmak. Bu işlemi yaparken eleman ekleyip çıkarabiliriz, her bir adımda farklı faaliyetler uygulayabiliriz.

Eğer CSS kullanımını çok iyi biliyorsanız, zaten çok fazla komuta aşinasınızdır. JQuery de bunların birçoğunu CSS'den almıştır. Ama CSS'de görmediğiniz bir kısım komut da olabilir, özellikle CSS3 seçicileriyle çok fazla ilgilenmediyseniz. Daha fazla örnek ve açıklama için şu adresi ziyaret edebilirsiniz: http://api.jquery.com/category/selectors/

Dekorasyon: JQuery ile CSS

Seçme işlemi JQuery'nin en zor kısmıdır. Bu aşamadan sonra her şey daha kolay ve eğlenceli. Etiketlerimizi(tag) seçtikten sonra, onları etkiler veya arayüzler yapmak için değiştirebiliriz. Bu bölümde, CSS ile ilgili JQuery fonksiyonlarını ele alacağız: ekleme-çıkarma stilleri, sınıflar ve daha fazlası... Gerçekleştireceğimiz işlemler, seçtiğimiz tüm elemanlara uygulanacak.

CSS Özelliklerini Okuma

CSS özelliklerini değiştirmeye çalışmadan önce, onlara basitçe nasıl ulaşabilceğimize bakalım. JQuery “css” fonksiyonuyla bunu yapmamıza izin veriyor. Bunu deneyin:

```
$(document).ready(function() {
    var fontSize = $( ' #celebs tbody tr:first ' ).css( 'font-size' );
    alert(fontSize);
});
```

Bu kod, seçici ile eşleşen ilk elemanın yazı tipi boyutunu bize gösterecek. CSS özelliklerini bu metodla almanın en güzel yanı , JQuery bize elemanın hesaplanan stilini verir. Bunun anlamı, siz CSS tanımındaki değer yerine kullanıcının sunucusunda yorumlanan değeri alacaksınız. Yani, eğer siz CSS dosyasında 200 piksel boyutunda bir div kullanırsanız ama içeriği 200 pikselin daha da üzerinde gönderilirse, JQuery, sizin tanımladığınız 200 piksel yerine elemanın gerçek boyutunu size sağlayacaktır. Bunun neden önemli olduğunu ilerideki konularda

göreceğiz.

CSS Özelliklerini Tespit Etme/Kurma

```
$(document).ready(function() {
  $( '#celebs tbody tr:even' ).css( 'background-color' , '#dddddd' );
});
```

Kodu incelediğimizde, öncekinden farklı olarak ekstra bir parametre yazıldığını görüyoruz, o özellik için tespit etmek istediğimiz değer. Faaliyeti, arkaplan renginin açık gri olması için kullandık. Kod arşivini açıp bunu deneyin ve doğru çalışıp çalışmadığını gözlemleyin. Sonucu aşağıda görebilirsiniz:

ID	Name	Occupation	Approx. Location	Price
203A	Johny Stardust (bio)	Front-man	Los Angeles	$39.95
141B	Beau Dandy (pic,bio)	Singer	New York	$39.95
2031	Mo' Fat (pic)	Producer	New York	$19.95
007F	Kellie Kelly (bio,press)	Singer	Omaha	$11.95
8A05	Darth Fader (pic)	DJ	London	$19.95
6636	Glendatronix (bio,press)	Keytarist	London	$39.95

Şimdi, açık gri yaptığımız satırların daha parlak görünmesini sağlayalım. Bunu yapabilmek için şu kodu deneyin:

```
$(document).ready(function(){
  $('#celebs tbody tr:even').css('background-color','#dddddd');
  $('#celebs tbody tr:even').css('color', '#666666');
});
```

Burada, önceki CSS komutumuza ek olarak ikinci bir satır ekledik. Fakat daha karmaşık kodlarda bu şekilde yapmak karışıklığa sebep olup okunabilirliği azaltabilir. Bunu önlemek için JQuery'nin bize sunduğu bir olanak var: abartısız/gerçek nesne(object literal). Aslında bu JavaScript'in içerdiği bir konu fakat burada çok fazla ayrıntıya girmeyeceğiz. Sadece bize anahtar/değer(key/value) çiftlerini gruplayabilmek için kolay bir yol sunduğunu bilmemiz yeterli. Bu sayede, bir önceki kodu şu şekilde değiştirebiliriz:

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

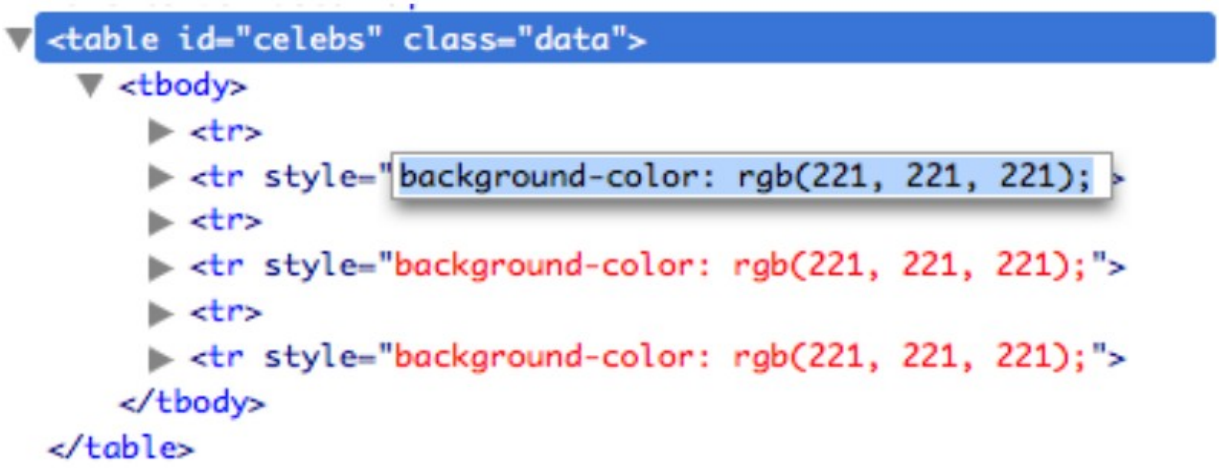
```
  $('#celebs tbody tr:even').css(
    {'background-color': '#dddddd', 'color': '#666666'}
  );
});
```

Görüldüğü gibi tüm değişiklikler tek bir css fonksiyonunun içinde süslü parantezler arasında ve aralarına virgül konarak yazılıyor. Tüm bu değişiklikleri kendinizin de uygulayıp görmesinde fayda var. Çok daha fazla değişikliği bir anda yapmak istediğinizde, okunulabilirliği arttırmak adına kodunuzu şu şekilde yazmanız sizin için daha faydalı olacaktır:

```
$(document).ready(function(){
  $('#celebs tbody tr:even').css({
    'background-color': '#dddddd',
    'color': '#666666',
    'font-size': '11pt',
    'line-height': '2.5em'
  });
});
```

Sınıflar

Şu ana kadar bizden istenilenleri yaptık gibi gözüküyor ama tablodaki çizgilerimizi Firebug gibi bir geliştirme aracında incelersek, CSS özelliklerinin paragrafa iç satır yazısı olarak şu şekilde eklendiğini görürürüz:



Not: Firebug nedir? Sunucumuzda CSS,HTML ve JavaScript'i gözlemlemek veya eklemeler yapabilmek gibi, DOM'u inceleyebilmek için faydalı bir araçtır. Mozilla Firefox'un bir uzantısı olarak kullanılabilir.
İç satır yazıları çok tercih edilmez. Daha temiz ve sürdürülebilir bir kod yazabilmek

için, stil bilgilerimizi toplu halde CSS dosyamızdan ayrı bir yerde tutmalıyız. Sonra, basitçe bu stilleri sınıf özelliklerini HTML etiketlerimize ekleyerek veya silerek değştirebiliriz.

Sınıfları Silme ve Ekleme

Eğer CSS'i iç satır yazısı şeklindeki kurallardan silmemiz gerekiyorsa, onu nereye koymalıyız? Ayrı bir stil sayfasına tabiki. Verilen bir sınıfa etiketlenen CSS'deki bir kurala istediğimiz bir stili koyabiliriz ve o sınıfı HTML'deki etiketlenen elemanlardan silmek veya eklemek için JQuery'i kullanırız. JQuery, DOM elemanlarının sınıf özelliklerini değitirebilmemiz için bize bazı fonksiyonlar sunuyor. Çizgili stilimizi CSS dosyasının içine taşıyabilmemiz için biz de bunlardan en yaygın olanını kullanacağız: addClass. Bu fonksiyon, sınıf adını içeren bir diziyi parametre olarak kabul eder. Aynı zamanda, birden fazla sınıfı aralarında boşluk bırakarak ekleyebiliriz. Daha iyi anlayabilmek için aşağıdaki yapıyı inceleyelim:

```
$( 'div' ).addClass( ' class_name ' );
$( 'div' ).addClass( 'class_name1 class_name2 class_name3');
```

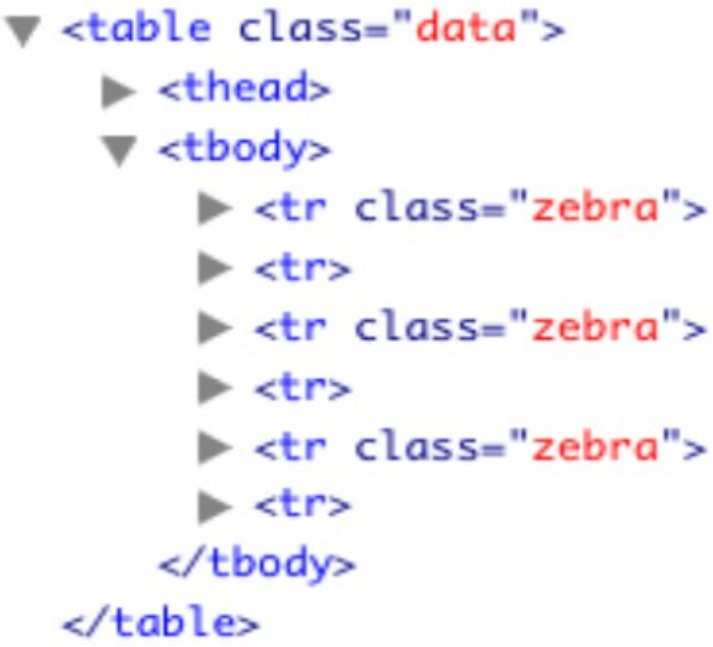
Tablomuzdaki çizgili yapıyı zebraya benzettiğimizi düşünelim. Biz sadece bir tane sınıf eklemek istiyoruz, adı da zebra olsun. İlk olarak, yeni bir CSS dosyası yaratıp içine kuralımızı yazıyoruz:

```
.zebra {
  background-color: #dddddd;
  color: #666666;
}
```

Sonra tekrar JavaScript dosyamıza dönüyoruz. Seçicimizi, css fonksiyonu yerine jQuery'nin addClass yapısını kullanarak şu şekilde değiştiriyoruz:

```
$(document).ready(function(){
  $('#celebs tbody tr:even').addClass('zebra');
});
```

Sonuç tamamen aynı olacak ama şimdi tablomuzu Firebug'da incelediğimizde, iç satır yazılarının gittiğini göreceğiz. Onların yerine bizim yeni sınıf tanımlamalarımız gelecek. Bunu aşağıdaki şekilden inceleyebilirsiniz:



Bu kullanım çok daha iyi. Şimdi, herhangi bir zamanda eğer zebra çizgilerimizin görüntüsünü değiştirmek istersek, basitçe CSS dosyamız üzerinde istediğimiz değişiklikleri yapabiliriz.
Benzer şekilde, elemanlardan sınıf isimlerini de silmek isteyebiliriz. Bunu yapabilmek için kullanacağımız komut: removeClass. Bu fonksiyonun kullanımı addClass'ın kullanımıyla aynı; sadece istenmeyen sınıfın ismini parametre olarak yazıyoruz:

$(' #celebs tr.zebra ').removeClass('zebra');

Aynı zamanda, numara ya da herhangi bir özelliği jQuery'nin “attr” metodunu kullanarak değiştirebiliriz. Bu metodun detaylarına daha sonra gireceğiz.

Genişletme: jQuery ile Efektler Ekleme

Şu ana kadar önemli bir aşama kaydettik. JQuery komutlarının bileşenlerini öğrendik: seçici, faaliyet ve parametreler. Aynı zamanda, komutları kullanabilmek için gerekli adımları öğrendik: dökümanın hazır olduğundan emin olmak, elemanları seçmek ve onların üzerinde değişiklikler yapmak. Şimdi bütün bu bilgileri faydalı efektler yapabilmek için kullanacağız ve jQuery temellerimizi güçlendireceğiz.

Elemanları Gizleme veya Gösterme

Müşteri vazgeçme/red belgelerini sevmiyor çünkü ürünü kötü etkilediğini düşünüyor. Ancak onun avukatı bunun gerekli olduğu konusunda ısrar ediyor. Bu yüzden, müşteri bir buton eklememizi istiyor. Bu butonun özelliği; kullanıcı metni okuduktan sonra isterse silebilsin.

```
<input type=”button” id=”hideButton” value=”hide” />
```

hideButton numarasıyla sayfamıza bir HTML butonu ekledik. Kullanıcı bu butona tıkladığında bir vazgeçme elemanı isteniyor, bu elemanın numarası “disclaimer” ile gizlenecek:

```
$('#hideButton').click(function(){
  $('#disclaimer').hide();
});
```

Bu kodu çalıştırın ve butona tıkladığınızda elemanın yok olduğunu gözlemleyin. Burada bu işi yapan “hide” faaliyeti. İşin yapılmasını sağlayan şey ise herhangi bir nesnenin faaliyette olup olmayacağına karar veren “Event Handler” yani olay işleyicisidir.

Olay İşleyicisi(Event Handler)

Bir olay gerçekleştiğinde o “fired”(yandı) oldu deriz ve o olayı işlemek için bazı kodlar yazdığımızda olayı “caught”(yakalandı) ettik deriz.

Bir internet sayfasında herhangi bir anda binlerce olay “fired” olmakta; kullanıcı fareyi hareket ettirdiğinde, bir butona tıkladığında ya da bir pencerenin boyutunu değiştirdiğinde vs. bu olayları yakalayıp üzerlerinde işlem yapabiliriz.

Konularımızın başından beri ilk gördüğümüz olay işleyicisi, dökümanın hazır olduğunu söyleyen olaydı. Döküman ben hazırım dediğinde bir olay “fired” oluyordu ve bizim jQuery komutumuz olayı yakalıyordu.

Biraz önce incelediğimiz kod parçasında, butona tıklandığında disclaimer'ı gizlemesi için “click” olay işleyicisini kullandık:

```
$('#hideButton').click(function(){
  $('#disclaimer').hide();
});
```

this kullanımı

Örneğin, kullanıcının tıkladığı bir butonu bir şekilde modifiye etmek isteyebiliriz. Böyle bir işlem, bizim olay işleyicimizin içinde JavaScript'in “this” anahtar kelimesiyle mümkün olabilir. Bir JavaScript nesnesini jQuery nesnesine dönüştürmek için, onu jQuery seçicisinde paketleriz:

```
$('#hideButton').click(function(){
  $(this).hide();   // ilginç, gözükmeyen bir buton
});
```

$(this), o elemanı yeniden seçmek yerine doğrudan event'i “fired” eder ve elemandan bahsedebilmemizi sağlar.

Gizli Elemanları Yeniden Görünür Hale Getirmek

Müşterimiz aynı zamanda, eğer kullanıcı yanlışlıkla bir bilgiyi/nesneyi yok eder ve tekrar onu görüntülemek isterse, bunun için de bir çözüm bulmamızı istiyor. O zaman HTML'e yeni bir buton daha ekleyelim. Bu butonun numarası(id) da showButton olsun:

```
<input type="button" id="showButton" value="show" />
```

Aynı zamanda, script dosyamıza başka bir jQuery komutu ekleriz:

```
$('#showButton').click(function(){
  $('#disclaimer').show();
});
```

Bu kodu eklediğimizde kaybolan “disclaimer” numaralı yazı tekrar görünür hale gelir,“hide” butonunu da tekrar görünür hale getirmek istersek kodumuzu şu şekilde düzenlemeliyiz:

```
$('#showButton').click(function(){
  $('#disclaimer').show();
  $('#hideButton').show();
});
```

Elemanları Değiştirme

disclaimer'ı gizlemek veya göstermek için ayrı ayrı butonlar kullanmak ekranda boş yere fazladan alan kullanılması demektir. Tek butona sahip olmak ve bu iki görevi de tek butona yaptırmak çok daha iyi bir çözüm olacaktır. Bunu yapabilmenin yolu,

elemanın görünür olup olmadığını kontrol etmek ve sonra bu bilgiye göre gizlemek ya da göstermektir. Şimdi önceden eklediğimiz butonları sileceğiz ve yerine şöyle bir buton ekleyeceğiz:

```
<input type="button" id="toggleButton" value="toggle" />
```

Bu butona tıklanıldığında, disclaimer'ın gizleneceğini ya da gösterileceğini kontrol edeceğiz:

```
$('#toggleButton').click(function(){
   if($('#disclaimer').is(':visible')) {
     $('#disclaimer').hide();
   } else {
     $('#disclaimer').show();
   }
 });
```

Burada “is” fonksiyonunu görüyoruz. Bu fonksiyon, jQuery fonksiyonuna yazdığımız aynı seçicilerden herhangi birini alır ve çağrıldığı elemanla eşleşip eşleşmediğini kontrol eder. Buradaki durumda, seçili olan #disclaimer'ın aynı zamanda sahte seçici(visible) tarafından da seçilip seçilmediğini kontrol ediyoruz. Bu başka özellikler için de kullanılabilir; seçimimiz div formunda mı yoksa etkin mi vs.

“is” fonksiyonu, elemanın seçiciyle eşleşip eşleşmediğine bağlı olarak true(doğru) veya false(yanlış) döndürecek. Bizim kullanımımızda iki durum var: gizli/görünür. Gizliyse görünür yapacağız, görünürse gizli yapacağız. Böyle durumlara iki durumlu (toggle) deniyor ve bu çok yararlı bir yapı.
İki durumlu yapılara “toggle” dendiğini belirttik. Şimdi, en son yazdığımız kodu buna göre düzenleyecek olursak:

```
$('#toggleButton').click(function(){
   $('#disclaimer').toggle();
 });
```

Butona her tıkladığımızda, elemanımız görünür ve görünmez durumları arasında gidip gelecek. Bu güzel bir kullanım ama ya kullanıcımız her butona tıklanıldığında buton üzerindeki yazının da
değişmesini isterse? Bunu yapmanın birkaç yolu var, bunlardan birisi şu şekilde:

```
$('#toggleButton').click(function(){
   $('#disclaimer').toggle();
   if($('#disclaimer').is(':visible')) {
```

```
    $(this).val('Hide');
  } else {
    $(this).val('Show');
  }
});
```

Burada $(this) butonumuzdan bahsediyor. Kodu inceleyip uygulamayı deneyin, daha fazla detaya ileride değineceğiz.

Aşamalı Geliştirme

Müşterimizin tüm isteklerini yerine getirdik, fonksiyonlarımız gayet iyi çalışıyor ama düşünemediğimiz bir durum var. Ya kullanıcı JavaScript desteklemeyen bir sunucu kullanıyorsa? O zaman hazırladığımız butonu sayfada görecek fakat bu butona tıkladığında herhangi bir değişiklik göremeyecek. Bu kullanıcı için olumsuz bir durum. Şimdi, JavaScript desteği olmayan bir sunucu olabilir mi diye düşünebilirsiniz. Ancak çok eski bilgisayarlar kullanan veya sınırlı uygulamalara sahip cep telefonu kullananlarda böyle bir sorun oluşabilir.

Çözüm, bu kullanıcılara kabul edilebilir bir deneyim sağlamak. Bu uygulama aşamalı geliştirme olarak biliniyor.

Bizim “disclaimer” fonksiyonumuzun tüm kullanıcılara görünür olmasını istiyoruz, bu yüzden HTML'imizin içinde ona yer veriyoruz. Sonra, JavaScript ile tüm kullanıcılara gizleme becerisini ekliyoruz. Yani, göster/gizle butonunu tercih etmemeye çalışıyoruz ki bunu kullanamayan kullanıcılarımıza da hitap edebilsin.

Bunu başarabilmemizin bir yolu, butonumuzu CSS ile gizlemek ve jQuery'i sadece CSS komutuyla gösterilebilir hale getirmek. Ancak, eğer kullanıcının sunucusu CSS'i de desteklemiyorsa bu da bir problem yaratacak. En iyisi, butonu jQuery aracılığıyla ekleyelim, sadece JavaScript'i destekleyen sunucular kullanan kullanıcılar butonu görebilsin. Mükemmel!

Yeni Elemanlar Ekleme

Bu zamana kadar seçme işlemi için kullanılan jQuery fonksiyonlarını gördük ama başka önemli bir fonksiyon daha var: yeni elemanlar yaratma. Aslında, jQuery fonksiyonunun içine koyduğunuz herhangi geçerli bir HTML dizesi yaratılacaktır ve sayfaya koyabilmeniz için hazır duruma getirilecektir. Aşağıdaki örnekte basit bir paragraf elemanının nasıl yaratılacağını görebilirsiniz:

```
$( ' <p>Yeni bir paragraf!</p> ' )
```

Bu kodu yazdığınızda jQuery birçok yararlı faaliyeti gerçekleştirecektir: HTML'i DOM parçalarına çözümler ve onu seçer(sıradan bir jQuery seçicisinin yaptığı gibi). Bunun anlamı, daha fazla jQuery aşaması için hazır olduğudur. Örneğin, yeni yarattığımız elemanımıza bir sınıf eklemek için basitçe şöyle bir komut yazabiliriz:

```
$( ' <p> Yeni bir paragraf! </p> ' ).addClass( 'new' );
```

Bu metodu kullanarak, ihtiyacınız olan yeni elemanları HTML biçimlemenizde tanımlamak yerine jQuery aracılığıyla yaratabilirsiniz. Bu yol ile, sayfamızı aşamalı geliştirmeyle tamamlayabiliriz.

Yeni elemanlarımızı yarattıktan sonra, onları sayfamızda istediğimiz yere koymamızı sağlayacak bir yola ihtiyacımız var. Bu amaç için kullanılabilecek bir çok jQuery fonksiyonu var. İlk bakacağımız fonksiyon: insertAfter. Bu fonksiyon, o anki jQuery seçimimizi alacak ve fonksiyona parametre olarak yazacağımız başka bir seçili elemandan sonra ekleyecek.

Bunun nasıl çalıştığını kolayca anlayabilmek için aşağıdaki örneği inceleyelim. Aşağıdaki kod parçası jQuery ile nasıl toggle butonu yaratabileceğimizi gösteriyor:

```
$('<input type="button" value="toggle"
id="toggleButton">').insertAfter('#disclaimer');
  $('#toggleButton').click(function(){
    $('#disclaimer').toggle();
  });
```

insertAfter fonksiyonu disclaimer elemanından sonrasına yeni bir eleman ekliyor buradaki örnekte. Eğer butonun disclaimer elemanından önce gözükmesini isterseniz, ya elemanı disclaimer'dan önce etiketleyip insertAfter kullanın ya da daha iyisi sadece insertBefore fonksiyonunu kullanın:

```
$('<input type="button" value="toggle"
id="toggleButton">').insertBefore('#disclaimer');
  $('#toggleButton').click(function(){
    $('#disclaimer').toggle();
  });
```

Küçük bir hatırlatma: DOM'u düşündüğümüzde, aynı seviyede olan elemanları kardeş(siblings) olarak nitelendiririz. Eğer sizin bir “div” elemanınız varsa ve iki tane “span” elemanına sahipse, bu “span” elemanları kardeş olurlar.

Eğer var olan bir elamanın kardeşi olarak yeni bir eleman eklemek isterseniz, prependTo ya da appendTo fonksiyonlarını kullanabilirsiniz:

```
$('<strong>START!</strong>').prependTo('#disclaimer');
$('<strong>END!</strong>').appendTo('#disclaimer');
```

Burada, prependTo fonksiyonu disclaimer'ın başına START! kelimesini ekler, appendTo fonksiyonu ise disclaimer'ın sonuna END! kelimesini ekler:

START! Disclaimer! This service is not intended for the those with criminal intent. Celebrities are kind of like people so their privacy should be respected. **END!**

Eklemek ve silmek için çok daha fazla faaliyet(action) var ama şimdilik bu kadarını bilmemiz yeterli.

Varolan Elemanları Silme

Müşterimizi sunucusu JavaScript'i desteklemeyen kullanıcıların eklediğimiz özellikleri görüntüleyemeyeceği konusunda uyardık ve o da bize şöyle bir öneride bulundu: bu kullanıcılarımız için ekranda bir uyarı mesajı göstersek ve JavaScript desteğinin olması konusunda bilgilendirsek? Bu mesaj JavaScript'i destekleyen sunucular kullanan kullanıcılara gözükmeyecek.

Bu senaryo jQuery kullanarak HTML elemanlarının bir sayfadan nasıl silineceğini öğrenmek için güzel bir örnek gibi duruyor. Mesajımızı HTML'e koyacağız ve onu jQuery'den sileceğiz: bu sayede sadece JavaScript'i olmayan ziyaretçiler bu mesajı görebilecekler.

Şimdi HTML sayfamıza yeni bir uyarı ekleyelim:

```
<p id="no-script">
    We recommend that you have JavaScript enabled!
</p>
```

Şimdi, elemanları sayfadan silmek için kodumuzu çalıştırmalıyız. Eğer kullanıcı JavaScripte sahip değilse, jQuery komutlarımız çalışamayacak ve mesaj ekranda kalacak. Jquery'deki elemanları silmek için, ilk önce onları seçiciyle seçeriz ve sonra remove(sil) metodunu çağırırız:

```
$( ' #no-script ' ).remove();
```

“remove” faaliyeti bütün seçili elemanları DOM'dan silecek ve aynı zamanda o elemanlara bağlı bulunan bilgileri ve olay işleyicileri(event handler) de siler. Bu

fonksiyon herhangi bir parametre gerektirmiyor. Ama seçimi daha güzel bir hale getirmek için bir ifade tanımlayabilirsiniz. Şu örneği deneyin:

$('#celebs tr').remove(':contains("Singer")');

Bu kod, #celebs div'in içindeki tüm tr'leri silmek yerine sadece “Singer” metnini içeren satırları siliyor.

Yaptığımız tüm değişiklikler sayesinde şu an sitemiz hem JavaScript'li hem JavaScript'siz kullanıcılar için iyi bir şekilde çalışıyor. Bu, aşamalı geliştirmenin çok basit bir örneği ama aynı zamanda çok temel bir fikri anlamamızı sağlıyor: jQuery'i kullanıcı arayüzümüzün(UI) destekleyicisi olarak kullanmak yerine, işlemekte olan deneyimlerimizi/fonksiyonlarımızı daha güzel bir hale getirmek için kullanın.

İçeriği Değiştirme

Bir elemanın asıl içeriğini değiştirmek istersek? Bunun için JQuery bir kaç fonksiyona sahip:“text” ve “html”. Bu iki faaliyet birbirine çok benziyor çünkü ikisi de seçtiğimiz elemanın içeriğini ayarlıyor. Basitçe iki fonksiyona da birer dize(string) yazalım:

$('p').html('paragraflar gitti!');
$('h2').text('Bütün başlıklar bizim!');

Bu örnekte, her iki fonksiyon da eşleştiği elemanların içeriğini verilen cümle ile değiştiriyor, kendiniz deneyip sonucu inceleyin. “text” ve “html” arasındaki farkı içerik dizesine HTML eklemeye çalışırsak görebiliriz:

$('p').html('Uyarı! Text değiştirildi.');
$('h2').text('Uyarı! Başlıkta komutlar da gözüktü.');

Bu örneği uyguladığınızda, başlıkların “Uyarı!” halinde olduğunu gözlemlerken paragraflarda koyu renkle “Uyarı!” yazıldığını görebilirsiniz.

Sadece bu yeni faaliyetler mi içeriği değiştirebilir? diye bir soru gelebilir aklınıza. Aynı faaliyetleri kullanarak jQuery seçimlerimizdeki içerikleri atabiliriz:

alert($('h2:first').text());

“text” faaliyetini herhangi bir parametre vermeden kullanabiliriz. Bu bize sayfadaki ilk h2 etiketinin(tag) metin içeriğini döndürür(“Welcome!”).

Temel Animasyon: Yetenek ile Gizleme ve Geri Getirme/Yeniden Gösterme

Şu ana kadar tüm yaptıklarımız yararlı ama etkileyici değil. Şimdi canlandırılmış jQuery tekniklerine geçelim. Ana jQuery kütüphanesi, sayfamızı daha güzel görünürmlü bir hale getirmek için kullanabileceğimiz temel efektler içerir. Bunlara yeteri kadar sahip olduktan sonra, yüzlerce çılgın efektlere ulaşabileceğiniz eklenti deposunda(plugin repository) gezinebilirsiniz.

FadeIn ve FadeOut

JQuery'deki en yaygın efektlerden birisi gömme/yerleşik görünen efektlerdir(built-in fade effect).Göstermeyi(Fading) en basit halde kullanmak için, show'u “fadeIn” ile, hide'ı ise “fadeOut” ile yer değiştirin:

```
$('#hideButton').click(function(){
   $('#disclaimer').fadeOut();
 });
```

Kodu uygulamadan önce HTML'inize butonu eklemeyi unutmayın:

```
<input type="button" id="hideButton" value="hide" />
```

Efekti değiştirmek için aynı zamanda bazı opsiyonel parametreler de var. Bunlardan ilki görünmenin tamamlanması için geçen zamanı kontrol etmek için kullanılır. Çoğu jQuery efekti ve animasyonu zaman parametresini kabul eder. Bu parametreyi dize olarak veya tamsayı(integer) olarak yazabilirsiniz.

Zaman dilimini önceden tanımlanan bazı kelimeleri kullanan bir dize olarak belirtebiliriz: yavaş(slow), hızlı(fast) ve normal. Örneğin: fadeIn('fast'). Eğer animasyonun süresi üzerinde daha detaylı bir kontrol belirtmek istiyorsanız, zamanı milisaniyelerle de gösterebilirsiniz: fadeIn(1000).

Efektleri ve Animasyonları Değiştirme

JQuery'nin görüntüleri kullanarak değişiklik yapmak için belli bir faaliyeti olmamasına rağmen, küçük bir sır var: eğer “toggle” faaliyetimize parametre olarak bir zaman dilimi yazarsak, onun animasyon ekleme özelliği olduğunu göreceğiz:

```
$('#toggleButton').click(function(){
   $('#disclaimer').toggle('slow');
 });
```

Burada HTML dosyamızda şu kodun bulunduğundan emin olalım:

```
<input type="button" id="toggleButton" value="toggle" />
```

Kodu çalıştırdığımızda, toggle butonuna tıkladığımızda disclaimer'ın yavaşça kaybolduğunu, tekrar tıkladığımızda ise yavaşça geri geldiğini gözlemleyebiliriz.

Başka bir jQuery animasyon efekti daha var: sliding. Bu efekt de yerleşik(built-in) toggle faaliyetlerini içeriyor. “fade” ile aynı şekilde kullanılıyor ama şu faaliyetlere(action) sahip; slideDown, slideUp ve slideToggle. Aynı şekilde bir zaman dilimi de belirtebiliriz:

```
$('#toggleButton').click(function(){
   $('#disclaimer').slideToggle('slow');
 });
```

Bildiri/Geri Çağrım Fonksiyonları(Callback Functions)

Çoğu efekt “callback function” olarak bilinen özel bir parametre kabul eder. Geri çağrımlar, efektlerin işi bittikten sonra çalıştırılması gereken kodu belirtirler. Bizim durumumuzda, kayma işlemi(slide) bittikten sonra bizim callback kodumuzu çalıştıracak:

```
 $('#toggleButton').click(function(){
   $('#disclaimer').slideToggle('slow', function(){
     alert('The slide has finished sliding!')
   });
 });
```

Görüldüğü gibi callback fonksiyonu efekt faaliyetine ikinci parametre olarak yazıldı. Eğer disclaimer'ın sliding işlemi bittikten sonra butonun kaybolmasını istersek:

```
$('#hideButton').click(function(){
   $('#disclaimer').slideUp('slow', function(){
     $('#hideButton').fadeOut();
   });
 });
```

Birkaç Püf Noktası

Müşterimizin bizden istediği özellikleri hallettik, şimdi siteyi güzelleştirecek farklı bir şeyler yapalım. Bu kısımda yeni yapılar da olacak o yüzden örnekleri kendinizin de denemesinde fayda var, özellikle jQuery dünyasıyla yeni tanışıyorsanız.

Üzerinde/Etrafında Dolaşılırken Vurgulama(Highlighting)

Tablonun satırlarının renklerini değiştirmemize ek olarak, kullanıcı faresini tablo üzerinde gezdirirken bir vurgu yapmak istiyoruz.
Bu efekti "mouseover" ve "mouseout" olaylarıyla ilgilenen "event handlers"ı tabloya ekleyerek uygulayabiliriz. Sonra, üzerinde farenin dolaşacağı elemanların arkaplan rengini içeren bir CSS sınıfı ekleyebilir veya çıkarabiliriz:

```
$('#celebs tbody tr:even').addClass('zebra');

$('#celebs tbody tr').mouseover(function(){
   $(this).addClass('zebraHover');
  });

$('#celebs tbody tr').mouseout(function(){
   $(this).removeClass('zebraHover');
  });
```

Şimdi, stili de CSS dosyamıza eklememiz gerekiyor:

```
tr.zebraHover {
  background-color: #FFFACD;
}
```

Bu kodu denediğinizde sorunsuz çalıştığını göreceksiniz ama aynı işlemi yapmanın daha basit bir yolu var: JQuery bir "hover" fonksiyonu içeriyor ve bu fonksiyon mouseover ile mouseout faaliyetlerini bir araya getiriyor:

```
$('#celebs tbody tr:even').addClass('zebra');
  $('#celebs tbody tr').hover(function(){
    $(this).addClass('zebraHover');
  }, function(){
    $(this).removeClass('zebraHover');
  });
```

Görüldüğü gibi iki fonksiyonu birden parametre olarak yazıyoruz; biri mouseover olayı için diğeri ise mouseout olayı için.

Sınıf özellikleri ekleyip çıkarmayı daha etkin kullanabilir hale geldik bu yüzden başka bir faydalı sınıfa ilişkin bir faaliyeti inceleyebiliriz: toggleClass. Bu faaliyet, eğer eleman henüz sahip değilse bir sınıfı ekliyor, eğer sahipse de siliyor. Örneğin, kullanıcılarımızın tablodan birden fazla satır seçebilmesini istiyoruz. Bir satıra

tıkladığımızda belirginleştirilsin ve tekrar tıklanıldığında vurgu yok olsun. Bunu uygulamak yeni jQuery becerimizle çok kolay:

```
$('#celebs tbody tr:even').addClass('zebra');
  $('#celebs tbody tr').click(function(){
    $(this).toggleClass('zebraHover');
  });
```

Spoiler Revealer

Sitemizin en son haberler bölümü ünlülerimiz arasındaki dedikoduları içeriyor. Haberler sitedeki en çok ilgi geçen kısım, bir çok kullanıcı her gün yeni haberlere göz atıyor. Bunun için bir "spoiler revealer" öneriyoruz: kullanıcı bir haberin hangi ünlü ile ilgili olduğunu tahmin edebilir ve ondan sonra cevap için tıklayabilir.

Bu tarz bir fonksiyon siteye aynı zamanda iyi eklemeler yapabilir: mesela film eleştirileri. Hikayenin detaylarını içeren eleştirileri gizleyebilirsin ama filmi izlemiş olan kullanıcılar için görüntülenebilir hale getirebilirsin.

Spoiler revealer'ımızı kurmak için, sitemizin "news" bölümüne yeni bir eleman eklemeliyiz.Varsayılan olarak gizlenen bazı sırlar "span" elemanının içinde, "spoiler" sınıfına bağlanarak paketlenebilir:

Who lost their recording contract today? (Bugün kimler sözleşme kayıtlarını kaybetti?)

```
<span class='spoiler'>The Zaxntines!</span>
```

Script'imizin nelere ihtiyacı var? İlk olarak, cevapları gizlemeliyiz ve eğer kullanıcı isterse yeniden görünebilir hale getirebilmemiz için yeni bir eleman eklemeliyiz. O eleman tıklanıldığında, cevabı görünür hale getirmeliyiz. Gizlemek? Eklemek? Tıklamaları idare etme? Biz tüm bunların nasıl yapılacağını biliyoruz:

```
$('.spoiler').hide();
$('<input type="button" class="revealer" value="Tell
Me!"/>').insertBefore('.spoiler');
$('.revealer').click(function(){
   $(this).hide();
   $(this).next().fadeIn();
});
```

Burada yeni kullanımlar var ama kodu inceleyip uyguladığınızda neyin ne olduğunu rahatlıkla anlayabileceksiniz. İlk olarak, bütün spoiler elemanlarını gizliyoruz ve

“insertBefore” faaliyetini, her birinden önce yeni bir buton eklemek için kullanıyoruz. Bu noktada sayfada yeni “Tell me!” butonlarımız gözükecek ve orjinal spoiler span'leri gizlenecek. Sonra, bu yeni butonlarımızı seçeceğiz ve “click event handler”larımızı onlara ekleyeceğiz. Butonlardan herhangi birine tıklanıldığında, yeni “revealer” elemanımızı sileceğiz ve onun yanına “spoiler” elemanımızı geçireceğiz. “next” yeni kullandığımız bir faaliyet, DOM'u dolaşmak(traverse) için kullanılıyor ve bize bir elemanın yanındaki kardeşine ulaşma imkanı sağlıyor(aynı container içinde yanındaki ilk eleman).

```
▼ <p>
    Who lost their recording contract today?
     <input class="revealer" type="button" value="Tell Me!"/>
     <span class="spoiler" style="display: none;">The Zaxntines!</span>
  </p>
```

(Modifiye edilmiş DOM)

Eğer değiştirilmiş DOM'a bakarsak, “revealer” butonundan sonraki ilk elemanın “spoiler span” olduğunu görürüz. Sonraki faaliyet basitçe bizim seçimimizi o elemana götürür. JQuery bize aynı zamanda bir önceki faaliyete ulaşmayı da sağlar.

Aslında jQuery, DOM etrafında dolaşabilmemiz için bir düzine farklı faaliyete sahip. “previous” ve “next” en yararlı ikisi. Anlatımımız boyunca bunların bir çoğuna değineceğiz ama eğer şu an hepsini görmek isterseniz http://docs.jquery.com/Traversing adresini ziyaret edebilirsiniz.

Diğer bölüme geçmeden önce kısa bir bilgi verelim. Burada öğrendiğimiz basit jQuery yapı bloklarını alacağız ve butonlar, efektler, kullanıcı etkileşimi yaratmak için kullanacağız.

BÖLÜM 3: ANİME ETME, KAYDIRMA VE YENİDEN BOYUTLANDIRMA

Anime Etme

jQuery animasyon eklemek amaçlı oluşturuldu. Geniş bir eklenti dizesiyle çoğaltılmış güçlü metodlar sayesinde bir çok özellik yaratabiliriz.

CSS Özelliklerini Anime Etme

Şu ana kadar bazı animasyon örnekleri konusunda bilgi sahibi olduk: sliding, fading

ve bazı süslü gizleme ve gösterme örnekleri gibi. Ancak, tam olarak neyin anime edildiği ve bunun nasıl olduğu konusuna yeterince hakim olamadık. Şimdi çok ilgi çekici bir jQuery fonksiyonuna giriş yapacağız: animate. Bu fonksiyon, bazı ilgi çekici efektleri kendi kendimize biçimlendirebilmek için, CSS özelliklerinin tüm sunucularını anime etmemizi sağlıyor. Şimdi örnek bir kullanımını görelim:

```
$('p').animate({
   padding: '20px',
   borderBottom: '3px solid #8f8f8f',
   borderRight: '3px solid #bfbfbf'
 }, 2000);
```

Bu kodu çalıştırdığınızda, sayfadaki tüm paragrafların nasıl bir düzen içine gireceğini rahatlıkla gözlemleyebilirsiniz.

“animate”i kullanmak için, anime etmek istediğimiz, key/value çiftleri olarak belirtilen özellikleri içeren bir nesne yazıyoruz(çoğu css fonksiyonuyla atanan çoklu özelliklere benzer). Hatırlamanız gereken bir uyarı var: özellik isimlerini her bir kelimenin baş harfi büyük olacak şekilde yazarak oluşturmalıyız ki animate fonksiyonu tarafından kullanılabilsin. Örneğin; margin-left yazmak yerine marginLeft yazmalısınız.

Zaman dilimi parametresi 2. bölümde gördüğümüz basit animasyonlar gibi çalışmaktadır. Milisaniyeler yazabilirsiniz ya da “slow,fast,normal” dizelerinden birini geçirebilirsiniz. CSS özellikleri için değerler px, em, % ya da pt cinsinden yazılabilir: 100px, 10em, 50%, 16pt.

Daha ilginci, tanımladığınız değerler, elemanın o anki değeriyle alakalı olabilir. Tek yapmanız gereken değerin önüne += ya da -= yazmanız. Böylece, belirttiğiniz değer o elemana eklenebilir veya o elemandan çıkarılabilir. Şimdi bu özelliği, navigasyon menümüzün salınım yapabilmesi için kullanalım. Fareyi menü seçenekleri üzerine getirdiğimizde salınım yapsın istiyoruz, bunu hover fonksiyonuyla yaparız:

```
$('#navigation li').hover(function(){
   $(this).animate({paddingLeft: '+=15px'}, 200);
 }, function(){
   $(this).animate({paddingLeft: '-=15px'}, 200);
 });
```

Kodu uyguladığınızda menü seçeneklerinin nasıl sallandığını gözlemleyebilirsiniz. “animate” fonksiyonunu 2.bölümde gördüğümüz showing,hiding ve toggling fonksiyonlarını daha iyi kontrol edebilmek için de kullanabilirsiniz. Bir özelliğin animasyon değerini sayısal bir miktar olarak belirlemek yerine show,hide ya da

toggle olarak belirleyebiliriz:

```
$("#hideButton").click(function() {
   $('#disclaimer').animate({
     opacity: 'hide',
     height: 'hide'
   }, 'slow');
 });
```

Kendiniz daha farklı animasyon örnekleri uygulamaya çalışabilirsiniz. “animate” fonksiyonu daha bir çok güçlü ileri seviyede seçenekler sunuyor, bu bölümü ilerleyen konularda daha detaylı inceleyeceğiz.

Renk Animasyonu

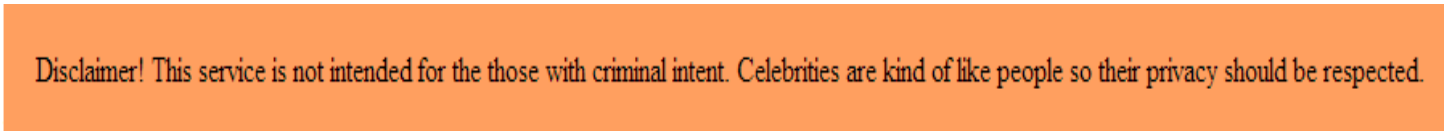
“animate” fonksiyonunu kullanmayı kavradıktan sonra, bir elemanın rengini anime etmek isteyebilirsiniz. Ancak, rengi anime etmek biraz ince bir iş çünkü başlangıç ve bitiş renklerinin değerleri özel bir şekilde hesaplanmalı. Örneğin, açık mavi ile turuncu arasında 3/4'lük bir oran gitmek gibi.

Bu renk hesaplama fonksiyonu ana kütüphaneden çıkarıldı. Bu durum bize şunu düşündürebilir: çoğu projede bu fonksiyona gerek yok bu yüzden jQuery kütüphane boyutunu küçültebilmek için bunu çıkardı. Eğer siz renkleri anime etmek isterseniz, http://plugins.jquery.com/color/ linkinden gerekli eklentiyi indirmelisiniz(Color Animations Plugin).

Eklentiyi indirdikten sonra, jQuery animasyon kodunuzdaki renk özelliklerini şimdi anime edebilirsiniz. Sayfa yüklendikten 2 saniye sonra disclaimer mesajımızın üzerinin turuncu olarak işaretlenmesini sağlayalım:

```
$('#disclaimer').animate({'backgroundColor':'#ff9f5f'}, 2000);
```

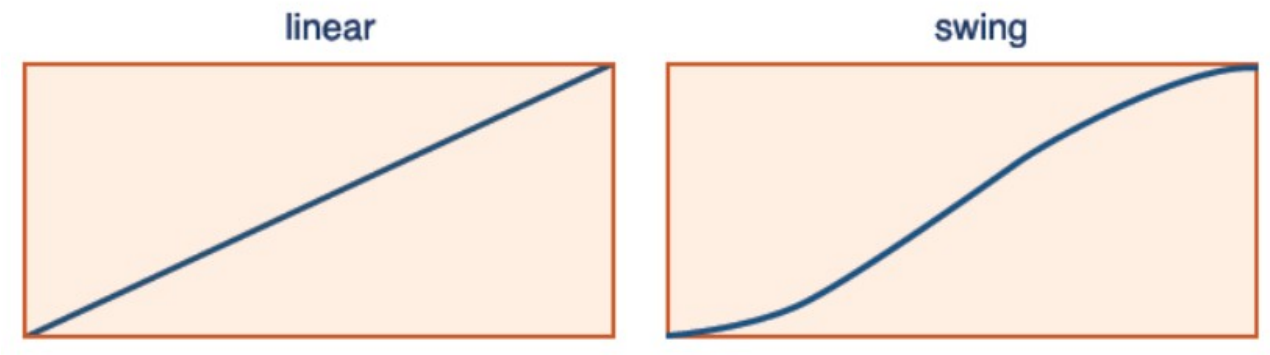
Kodu uygulayıp mesajın nasıl belirginleştirildiğini gözlemleyin. Sonuçta şöyle bir görüntü olacak:



“Easing”

“Easing”: Bir animasyonun daha doğal olabilmesi için ona uygulanan hızlandırma veya yavaşlatma işlemidir. Bir animasyonun ilerleme süresince hızını göstermek için

matematiksel bir algoritma uygular. JQuery'de iki tip easing vardır: linear(doğrusal) ve swing(yön değiştiren/dalgalı). İkisi arasındaki fark aşağıdaki şekilde görülebilir:



“swing easing” hız kazanmaya başlamadan önce yavaşça başlar, animasyonun sonuna doğru tekrar yavaşlar. Görsel olarak, “linear swing”e göre daha doğal gözükür ve eğer herhangi bir parametre belirtilmediyse jQuery varsayılan olaran “swing” kullanır.

“linear easing”de herhangi bir hızlanma veya yavaşlama yoktur, animasyon sabit bir oranda gerçekleşir. Çoğu durumda sıkıcı ve sabit gözükür ancak denemeye değerdir, bazı amaçlar için bunun kullanılması uygun olabilir.

Örnek olarak, ilk paragrafı anime edeceğiz. Tıklanıldığında büyüsün ve sonra büzülüp iyice küçülsün istiyoruz. Büyürken “linear easing” kullanacağız ve büzülürken de “swing” tercih edeceğiz. Aralarındaki fark hemen göze çarpmayabilir ama eğer animasyonu birkaç defa tekrar ederseniz o farkı ayırt edebilir hale geleceksiniz; büzülme animasyonu daha doğal gözükecek:

```
$('p:first').toggle(function() {
  $(this).animate( {'height':'+=150px'}, 2000, 'linear')
}, function() {
  $(this).animate( {'height':'-=150px'}, 2000, 'swing');
});
```

Bu kod parçasında bir çok komut var, şimdi durup bunların her birinin ne için kullanıldığını anladığınızdan emin olalım:

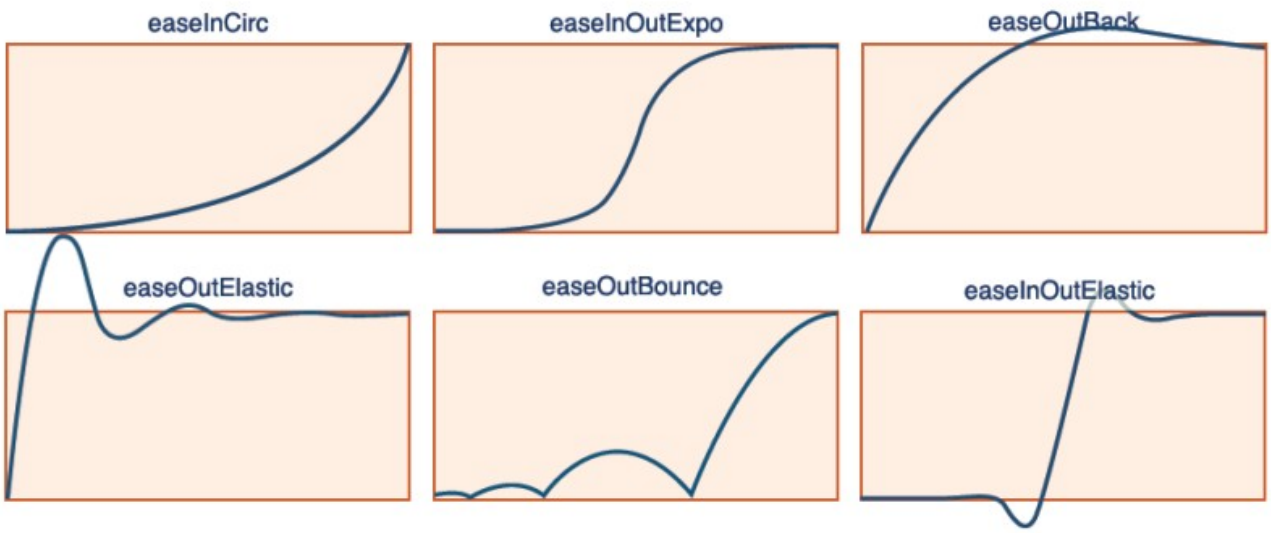
- Sadece ilk paragrafın etiketini yakalayabilmek için, seçiciyle birlikte bir filtre kullanıyoruz.
- Paragrafa bir “toggle event handler”ı bağlıyoruz.
- Faaliyeti paragrafın kendisine uygulayacağımız için “this” kullanıyoruz.
- İlk olayda, paragrafın boyunu 300 piksel büyütebilmek için “+=” formatını “linear easing” ile birlikte kullanıyoruz.

- İkincisinde ise 300 piksel küçültmek için “-=” formatını “swing easing” fonksiyonuyla birlikte kullanıyoruz.

Gelişmiş/İleri “Easing”

Bahsettiğimiz iki ana tip easing dışında, ana jQuery kütüphanesinde bir çok çeşit daha var. Bunlara easing eklentilerinden ulaşabilirsiniz:
http://plugins.jquery.com/jquery.easing/

İndirin ve eklentilerin JavaScript dosyalarını HTML sayfanıza dahil edin. Her birinin ayrı ayrı ne yaptığını açıklamak yerine dikkatinizi aşağıdaki grafiklere çekmek istiyoruz:



Şekillerden bazılarında grafiğin alan dışına taştığını görüyorsunuz. Anime edilen elemanlar bu kısma ulaştığında, eski konumlarına geri dönüyorlar ve sonunda oraya yerleşiyorlar. Yeni algoritmalardan birini kullanmak için, sadece animate fonksiyonumuza onun ismini yazmalıyız. Çok fazla örnek var, birkaçını kullanmaya çalışalım:

```
$('p:first').animate({height: '+=300px'}, 2000, 'easeOutBounce');
$('p:first').animate({height: '-=300px'}, 2000, 'easeInOutExpo');
$('p:first').animate({height: 'hide'}, 2000, 'easeOutCirc');
$('p:first').animate({height: 'show'}, 2000, 'easeOutElastic');
```

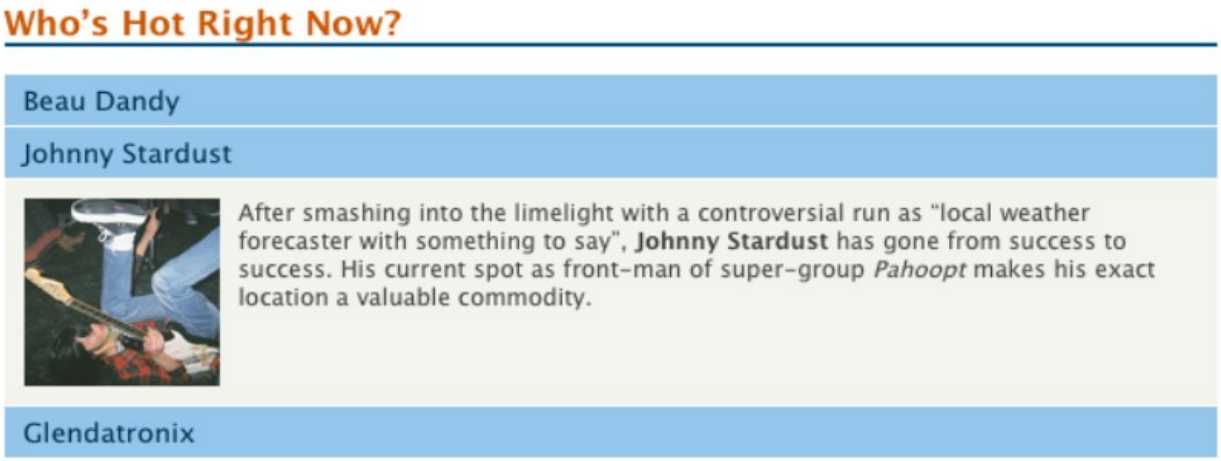
Kodu uygulayın ve paragrafın nasıl hareket ettiğine bakın!

Dinamik İçerik Çerçeveleri

Şu ana kadar “animate” fonksiyonlarının nasıl çalıştığını öğrendik. Şimdi,

müşterimizin bizden yeni bir isteği daha var: Şu anda hangi ünlü daha gözde, hangileri gözden düştü, bunun bilgisini ve güncellemelerini içeren bir liste oluşturmamızı istiyor(bir fotoğraf ve kısa bir biyografi de içerecek): Who is hot rigth now?. Çerçeveler bağımsız bir şekilde açılıp kapanabilsin diye listemize yeni öğrendiğimiz animasyonları ve “easing” tekniklerini uygulayacağız.

Görsel bileşenin sayfada nasıl gözükeceğini inceleyin:



HTML dosyamızda, bütün ünlülerimizi içeren bir “div” elemanı uygulayacağız. Her ünlünün bölümü h3 olarak işaretlenecek, kısa bir paragraf ve resim içeren “div” elemanıyla takip edilecek:

```
<div id="bio">
      <h2>Who's Hot Right Now?</h2>
      <h3>Beau Dandy</h3>
  <div>
       <img src="../../images/beau_100.jpg" width="100" height="100 "/>
       <p>Content about Beau Dandy</p>
 </div>
      <h3>Johny Startdust</h3>
      <div>
       <img src="../../images/johnny_100.jpg" width="100" height="100 "/>
       <p>After smashing into the limelight with a controversial run as &#147;local
weather forecaster with something to say&#148;, <strong>Johnny
Stardust</strong> has gone from success to success. His current spot as front-man of
super-group <em>Pahoopt</em> makes his exact location a valuable
commodity.</p>
      </div>
      <h3>Glendatronix</h3>
      <div>
       <img src="../../images/glenda_100.jpg" width="100" height="100 "/>
       <p>Content about Glendatronix</p>
```

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

Kullanıcı başlıklardan herhangi birine tıkladığında toggle edilmesini, yani açıksa kapatılmasını, kapalıysa açılmasını istiyoruz. Kodu çalıştırıp bunu gözlemleyin. Tarz (stil) üzerinde istediğiniz gibi değişiklik yapıp bu örneği tekrar deneyebilirsiniz.

Sayfa yüklenirken hiçbir içeriğin gösterilmesini istemiyoruz. Bu yüzden *$('#bio > div').hide();* komutunu kullandık. Eğer varsayılan olarak sadece ilk kısmın gösterilmesini isterseniz *$('#bio > div:first').show();* komutunu eklemeniz yeterli.

Burada yeni bir ifadeyle karşılaştık: > işareti. Bunun anlamı; çocuk seçicidir(child selector). Bunun görevi ise seçiciyle eşleşen tüm çocukları seçmektir. Eğer child seçiciyi kodumuzdan çıkarırsak "bio div" elemanının altındaki tüm div elemanlarını seçmiş oluruz(başka elemanların içine bağlanmış olsalar da). Bu seçicinin kullanımıyla ilgili daha fazla detay ve örnekler için http://docs.jquery.com/Selectors/child adresini inceleyebilirsiniz.

```
$('#bio h3').click(function() {
    $(this).next().animate(
      {'height':'toggle'}, 'slow', 'easeOutBounce'
);
```

Bu kod parçası ise başlıklara tıklanıldığında içeriğin gösterilmesi için uyguladığımız easing'i ve animasyonu içeriyor.

Animasyon Sırası

Animasyon konusunda göreceğimiz son başlık, "animate" fonksiyonunun başka bir gelişmiş kullanımı. Bu fonksiyon bir dizi ekstra seçenekle şu şekilde çağrılabilir:

```
animate(parameters, options);
```

"options" parametresi, anahtar/değer çiftleriyle oluşturulmuş bir çok seçeneğin bir araya getirilerek oluşturulduğu bir pakettir. Bunların bir çoğu şu ana kadar tanıdık: duration, easing ve complete(callback metodu). Ancak, bazı yeni olanlar da var: step ve queue. Onları açıklamadan önce, "options" parametresiyle "animate" fonksiyonunun nasıl çağrıldığını ve yazılım kuralını inceleyelim:

```
 $('p:first').animate(
  {
    height: '+=100px',
    backgroundColor: 'green'
```

```
  },
  {
    duration: 'slow',
    easing: 'swing',
    complete: function() {alert('done!');},
    queue: false
  }
);
```

Bir eleman için uygulanmasını istediğimiz her animasyon, istediğimiz sırayla uygulanır. Animasyonların bu şekilde bekletilmesi queue sayesinde olur, queue'da bir sıra oluşturulur ve o sıraya göre animasyonlar uygulanır. Ancak, bazı durumlarda birden fazla animasyonun bir elemana aynı anda uygulanmasını isteyebilirsiniz, bu durumda queue'yu devre dışı bırakmanız gerekir, bu sayede animasyonlar paralel olarak çalışabilir.

Animasyon sırası “queue” seçeneği kullanarak kontrol edilebilir. Aynı zamanda jQuery'nin şu faaliyetleri de vardır: stop, queue ve dequeue. Bu faaliyetler animasyonlar üzerinde istediğimiz gibi oynayabilmemizi sağlarlar. Ancak, gerçekten ileri seviye işler yapmaya başlamadan önce en önemli jQuery tekniklerinden birini açıklamanın vakti geldi.

Chaining(Zincirleme) Faaliyetleri

Şu ana kadar komutlarımızı hep birbiri ardına yazdık ya da callback fonksiyonları içinde biraraya getirdik. Seçicilerimizi hep yeniden yazdık ya da etiketlerimizi tekrar bulabilmek için “this” kelimesini kullandık. Ancak, aynı eleman üzerinde birbiri ardına birçok jQuery komutunu çalıştırabilmemizi sağlayan bir teknik var. Bu tekniğe “chaining” diyoruz.

“Chaining” tek bir elemana 2 veya daha fazla jQuery faaliyetini bağlar. Bir faaliyeti zincirlemek için, onu basitçe bir öncekine eklememiz gerekir. Örneğin birlikte “hide, slideDown ve fadeOut” faaliyetlerini zincirleyelim:

```
$('p:first').hide().slideDown('slow').fadeOut();
```

Kodumuzun daha anlaşılabilir olması için şu şekilde düzenlememiz faydalı olacaktır:

```
$('p:first')
  .hide()
  .slideDown('slow')
  .fadeOut();
```

Zincirlemeyi Duraklatmak

Uygulanan faaliyetlerin herhangi bir yerinde duraklatmak isterseniz, “delay”i kullanmanız gerekir. Ona bir sayı verin ve o kadar milisaniye geçene kadar bekletsin:

```
$('p:first')
.hide()
.slideDown('slow')
.delay(2000)
.fadeOut();
```

Anime Edilmiş/Canlandırılmış Navigasyon

Müşterimiz üst seviye bir navigasyon istiyor, Flash kontrolünde ve kullanıcıyla etkileşim içinde olsun istiyor.

Şimdi, yeni keşfettiğimiz jQuery özelliklerini Flash benzeri bir navigasyon kalıbı yaratmak için uygulayalım. Kullanıcı üzerinde gezerken menü seçeneğini vurgulamak için çevresinde titreşen bir “blob” arkaplanına sahip olacak. Ve bütün bunları Flash kullanarak değil, HTML, CSS ve JavaScript kullanarak yapacağız!

Navigasyon menümüz dikey yerine yatay olarak yerleştirebilmemiz için CSS özelliklerimizi modifiye ettik. HTML görüntümüz şu şekilde olacak:

```
<div id="navigation">
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Buy Now!</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Gift Ideas</a></li>
      </ul>
</div>
```

Arkaplan rengi blob'umuz, kullanıcının üzerine geldiği linkin arkasında duracak boş bir “div” elemanı olacak. Bunun için yapacağımız ilk iş, elemanı yaratıp dökümana eklemek:

```
$('<div id="navigation_blob"></div>').css({
 width: $('#navigation li:first a').width() + 10,
 height: $('#navigation li:first a').height() + 10
}).appendTo('#navigation');
```

“blob”u yerleştirdikten sonra, onu harekete geçirecek bir tetikleyiciye ihtiyacımız var.

Bu da kullanıcı navigasyon linklerinin üzerinde dolaştığı zaman olmalı. Bu yüzden "hover" fonksiyonunu kullanacağız. Bu fonksiyonun 2 tane parametre aldığını hatırlayın: biri fare üzerindeyken çalışacak diğeri ise üzerinde değilken. Genel olay işleyişini görelim:

```
$('#navigation a').hover(function() {
// Mouse over function
⋮
}, function() {
// Mouse out function
⋮
});
```

Fare linkin üzerindeyken şöyle bir fonksiyonumuz olmalı:

```
// Mouse over function
$('#navigation_blob').animate(
 {width: $(this).width() + 10, left: $(this).position().left},
{duration: 'slow', easing: 'easeOutElastic', queue: false}
);
```

Kullanıcı linklerin üzerindeyken blob'un iki özelliğini anime etmeliyiz: genişliği ve pozisyonu.

Sayfa üzerindeki linklerin pozisyonu "position" olarak adlandırılan bir jQuery metodunu kullanarak belirlenebilir. Bu fonksiyon kendi başına hiçbir şey yapmaz fakat çağrıldığında iki özellik açığa çıkarır: left ve top. Bunlar seçilen elemanın ebeveynine göre olan sol ve üst uzaklıklarıdır. Bu durumda, biz "left" özelliğini istiyoruz bu yüzden navigasyon menümüzde blob'un nereye hareket edeceğini biliyoruz.

"queue" opsiyonunu false yaptık çünkü kullanıcımız linkler üzerinde dolaşmaktan memnunsa animasyonlarımızın çalıştırılmayı beklerken yığılıp kalmayacağından emin olmak istiyoruz. Başka bir linke geçtiğinde, o anki animasyonun bitip bitmediğine bakmaksızın yeni animasyon başlayacak.
JQuery'e fare linklerin üzerinde değilken ne yapması gerektiğini hala söylemedik. Bu kodumuz birkaç zincirleme faaliyet içermesine rağmen önceki kodumuza çok benziyor:

```
// Mouse out function
$('#navigation_blob')
  .stop(true)
  .animate(
```

```
 {width: 'hide'},
 {duration: 'slow', easing: 'easeOutCirc', queue: false}
 )
  .animate(
{left: $('#navigation li:first a').position().left;},
  'fast'
);
}
```

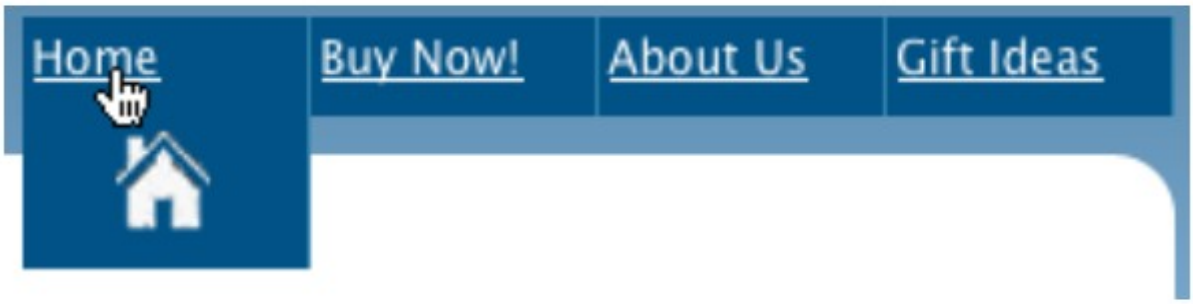
Bu kez 2 animate faaliyetini zincirledik: ilki “blob”u güzel bir easing uygulamasıyla gizliyor, ve ikincisi ise onu navigasyondaki ilk linkin pozisyonunun yanına götürüyor. Ayrıca, burada ekstra bir faaliyetin animasyonumuza zincirlendiğini farkedebilirsiniz: “stop”. Bu faaliyet animasyonu durdurur. İki parametre kabul eder: clearQueue ve gotoEnd. ClearQueue parametresine true değerini veriyoruz ki sırada bekleyen tüm animasyonlar silinsin. “gotoEnd” parametresi: eğer jQuery'nin bir elemanın o anki animasyon sırasının sonundaki durumunun(state) belirlemesini ve sonra hemen o state'e gitmesini istiyorsanız bu parametreyi kullanırsınız. Burada kullanmıyoruz çünkü animasyonlarımızın “blob”un o anki pozisyonundan başlamasını istiyoruz.

Bu uygulamayı kendiniz de çalıştırıp gözlemleyin, üzerinde değişiklikler yapmaya çalışın. Örneğin easing çeşitlerini deneyebilirsiniz ya da “blob”un arkaplan rengini değiştirebilirsiniz vs.

Anime Edilmiş/Canlandırılmış Navigasyon – 2

Müşterimiz oluşturduğumuz navigasyonu görmeden önce probleme farklı bir açıdan bakalım ve neyle karşılaşabileceğimizi görelim.

Bu animasyon için, menü maddelerimizin her biri gizli bir simgeye sahip olacak ve fare o linklin üzerine geldiğinde görüntülenecek(şekildeki gibi):



Bu efektin yapımı çok basit; yatay bir menu gibi görünen sırasız bir liste. Ancak, biz görünen alanın alt tarafına gizli bir simge yerleştirdik. Bu sadece varsayılan olarak metin linkini görebilesin diye liste maddelerinin yüksekliğini ayarlama hamlesi. Ancak, biz yüksekliği anime ettiğimizde simge ortaya çıkacak ve sorun olmayacak.

CSS ile başlayacağız. Menümüzü “container div”in sağ üstüne götürdük(simgelerin genişleyebilmesi için boşluk vardı orada). Karşılaştırılabilsinler diye simgelere arkaplan rengi atadık ve arkaplan resimlerini gizleyebilmek için akıllıca bir hile kullandık: liste maddelerinin 20px yüksekliğinde olduğuna ama arkaplanın üstten 30px uzaklıkta olduğunu biliyormuydunuz?. Sonuç olarak, list maddelerinin yüksekliğini genişletene kadar onlar görünmez olacak. Aynı zamanda, her linkin arkaplan resmini farklı bir simgeye atadık:

```
#container {
position: relative;
}
#navigation {
  position:absolute;
  width: inherit;
  top: 0;
  right: 0;
  margin: 0;
}
#navigation li {
  height:20px;
  float:left;
  list-style-type: none;
  width:70px;
  padding:3px;
  border-right:1px solid #3687AF;
  background-color: #015287;
  background-repeat: no-repeat;
  background-position: center 30px;
}
#navigation li a {
  color: #FFFFFF;
}

#navigation #home{
  background-image:url('icon_home.png');
}

⋮
```

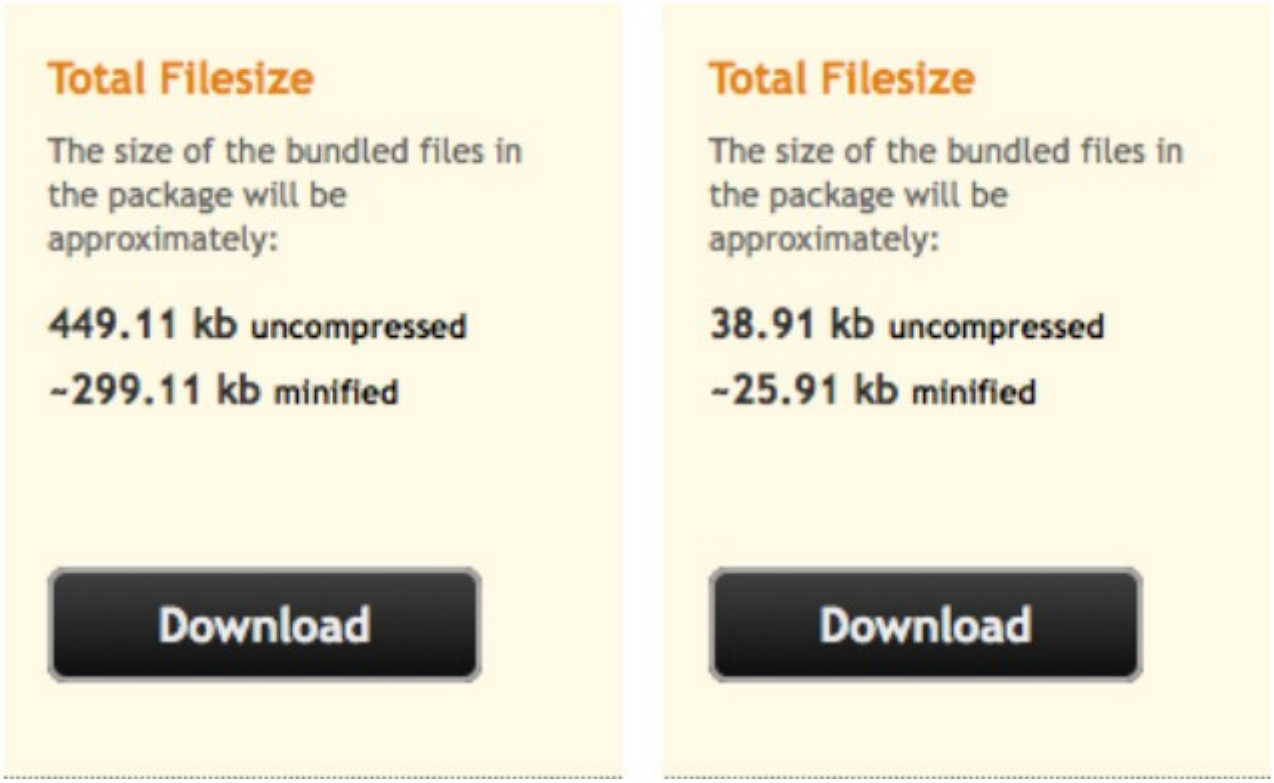
Bu efektin kodlamasındaki tek yeni olay, hem mouseover hem mouseout olaylarında “stop” faaliyetinin kullanılmasıdır. Sonra, biz sadece yüksekliği anime ederiz ki gizlenmiş simgeleri gösterebilelim ve “hover” bittiğinde normal haline döndürebilelim. Kodu inceleyelim:

```
$('#nav_stretch li').hover(
function() {
  $(this)
      .stop(true)
      .animate(
      {height: '60px'},
      {duration: 500, easing: 'easeOutBounce'}
)},
function() {
  $(this)
      .stop(true)
      .animate(
      {height:'20px'},
      {duration: 500, easing: 'easeOutCirc'}
)});
```

jQuery Kullanıcı Arayüzü Kütüphanesi (The jQuery User Interface Library)

İlk bölümde bahsedildiği gibi, jQuery UI kütüphanesi, ileri düzey jQuery görsel bileşenleri, efektleri ve etkileşimlerinin toplamıdır. Bunların hepsi web geliştirme için çok uygun. Şimdi bu kütüphaneyi nasıl indireceğimize ve kullanacağımıza bakalım. Şu ana kadar indirdiğimiz kütüphaneler ve eklentiler çok basitti, indirmesi de kolaydı fakat jQuery UI kütüphanesi öyle değil çünkü yaklaşık 500 KB boyutunda(minimize edilmiş hali 300KB). İçinde çok fazla kod var. Şanslıyız ki, herhangi bir proje normal olarak bu kodların sadece küçük bir kısmını kapsayacak ve jQuery web sitesi bize kendi jQuery UI versiyonumuzu yaratabilmemiz için kullanışlı bir araç sunuyor. İndirmek için http://jqueryui.com/ sitesini ziyaret ediniz. Birçok parçaya bölünmüştür: Core, Interaction, Widgets, Effects ve Themes. Core, ana jQuery UI kütüphanesi. Widgets ve Interaction bileşenlerinin hepsi ona dayanır. Oluşturucuya en iyi yaklaşım her şeyi seçilmemiş hale getirmek ve sadece kendi istediklerimizi eklemek. Eğer bir bileşen başka bir bileşene bağlı ise o otomatik olarak sizin için seçilecektir.

Geliştirirken, tüm kütüphaneyi elimizde tutmak güvenlidir. Bu sayede ihtiyacımız olan her şey bizim için orada olur. Internet sitemizin işlevselliğinden memnun olduktan sonra, indirme oluşrucuya dönebilir ve kullanılmayan her şeyi ayırarak kendimize özel bir kütüphane oluşturabiliriz. Asıl kütüphane ile bizim yaratacağımız kütüphane arasındaki dosya boyutu farklı şekilde görülmektedir:



Özel download arşivi birçok dosya içeriyor. “development-bundle” alt dizininde denemeniz için bir yığın demo ve belge var ama jQuery UI'yi kullanmak için “jQuery-ui-1.7.2-min.js” dosyasına ya da bunu okuduğunuz zaman en güncel haline ihtiyacınız var ve temasını(theme) sizin seçtiğiniz bir alt dizine ihtiyacınız var.

Temanızın alt dizinini HTML sayfanızın ulaşabileceği bir yere koymalısınız. Bu kitabın örneği için, jQuery UI “lib” alt dizininde ve tema dosyaları “css” alt dizininin içinde.

JQuery UI kütüphanesi, bazı ilginç efektleri uygulayabilmemiz için bir “Effects” paketi içerir. Aynı zamanda ileri seviye animasyonlar yapabilmemiz için faydalı fonksiyonlar ve işlevsellik de içerir. Bu işlevselliklerin bazılarını gördük(Renk Animasyonu eklentisi ve Easing eklentisinde). Bunlar “Effects” paketinde saklanıyor, bu yüzden eğer jQuery UI effects kütüphanesini kullanıyorsanız, onları dahil etmenize gerek kalmıyor.
Konuyu bitirmeden önce, efektlere bir göz atalım. İlk paragrafımızı alıp önce sallayacağız, sonra sırıyla vurgulayıp ardından parçalara ayıracağız:

```
$('p:first')
.effect('shake', {times:3}, 300)
.effect('highlight', {}, 3000)
.hide('explode', {}, 1000);
```

Nasıl etkileyici bir efekt olduğuna bakın! Tabiki, bu basit bir örnek. Efektlerin çoğu “effect” faaliyeti aracılığıyla bu şekilde kullanılıyor. Bazıları ise “hide, show ya da toggle” parametrelerinin yerinde kullanılıyor. Sonraki örneklerin bazıları: “blind, clip, fold ve slide”. Burada daha fazla detaya girmeyeceğiz fakat siz zaman ayırıp farklı türde animasyonlar yaratmaya çalışabilirsiniz.

Kaydırma

Bu bölümde, kullanıcı kaydırma yaparken sabit kalan menülere, özel temalı kaydırma çubuklarına ve hatta öğrendiğimiz animasyon tekniklerini nasıl kullanacağımıza ve dökümanları kaydırırken onlara nasıl uygulayacağımıza bakacağız.

“scroll” Olayı

Kullanıcılarımızın kaydırma deneyimini geliştiremeden önce, onların ne zaman ve ne kaydıracaklarını bilmemiz gerekir. Bu olaya “scroll” denir ve kullanıcı bir elemanın scroll pozisyonunu güncellediği zaman “fire” eder / uyarı verir. Bu yüzden sitenizde kullanıcılarınızdan biri “scroll bar”kaydırma çubuğu ile her etkileşime girdiğinde, sizin yakalayıp cevaplayabileceğiniz bir olay meydana gelecektir.

“scroll” olayını yakalamak için, kaydırma çubuğuna sahip olan bir elemana olay işleyicisi(event handler) eklemeliyiz(çoğunlukla pencere elemanlarına). Pencerenin kendisi bir JavaScript nesnesidir, bu yüzden tüm yapmamız gereken onu seçmesi için JQuery fonksiyonumuza paketlemektir.

Tabiki, “scroll” olayını faaliyet halinde görmek için, kaydırma çubuklarıyla bir alan hazırlamalıyız! Kullanıcıya uygun şekilde kaydırma efektleri hakkında birkaç fikrimiz var ama “scroll” olayının nasıl çalıştığını öğrenmek için taşma sınırlarını ayarlayarak kurmaca bir kaydırma ortamı oluşturalım. “scroll” sayfamızdaki div elemanlarından birinin üzerinde (css dosyamız):

```
#news {
   height: 100px;
   width: 300px;
   overflow: scroll;
}
```

Bu kod sayfamızın içinde “news” bölümünün küçük bir kaydırma penceresiyle çevrelenmesini sağlayacak. Şimdi, “scroll” olayını yakalayalım ve bu olay her gerçekleştiğinde sayfaya rastgele bir metin ekleyelim:

```
$('#news').scroll(function() {
  $('#header')
  .append('<span class="scrolled">You scrolled!</span>');
});
```

Şimdi, “news” bölümündeki çubuğu her kaydırdığında sayfanın başında kırmızı renkte “You Scrolled!” yazısı çıkacak.

Gezici Navigasyon

Şimdi kullanıcının ne zaman kaydırma işlemini yapacağını biliyoruz, sitemizi geliştirebilmek için bu bilgiyi nasıl kullanabiliriz? Kullanıcıların kaydırma işlemi ile etkileşime giren en yaygın sayfa örneklerinden biri, gezici navigasyon bölümüdür. Bu örnekte, scroll pozisyonuna bakmaksızın navigasyon elemanını hep sayfanın en tepesinde görürsünüz. Bu işlemi “scroll” olayını kullanarak uygulamak çok kolay: basitçe sayfanın neresinde olduğumuzu bulmalıyız ve sonra navigasyonu oraya hareket ettirmeliyiz.

İlk yapacağımız iş, anime edilmiş navigasyon için CSS dosyamızı hazırlamak. Bir pozisyon ekleyeceğiz: “relative”.Bu egzersizin amacına uygun olması için, içeriğe çok büyük bir yükseklik özelliği uyguluyoruz ki onu “scroll bar”(kaydırma çubuğu) oluşturması için zorlayalım:

```
#navigation {
position: relative;
}
#content {
height: 2000px;
}
```

Şimdi, gezici bölümümüze odaklanalım. İlk bakışta hiç kolay değilmiş gibi görünüyor ama sadece navigasyon bloğunun en üst pozisyonunu güncelleyerek ana pencerenin “scroll” olayına bakalım:

```
$(window).scroll(function() {
   $('#navigation').css('top', $(document).scrollTop());
 });
```

Sunucunuzda deneyin ve sonucun istediğimiz gibi olduğunu gözlemleyin. Bu durumda aklınıza bir soru takılabilir: En üst seviyenin neresi olduğunu nasıl biliyor? Kodu incelediğimizde scrollTop faaliyetini kullanmış olduğumuzu göreceğiz. Bu fonksiyon, eşleştiği elemanın en üste olan uzaklığını döndürür. Bizim örneğimizde, tüm dökümanın en üst pozisyonunu sorduk:

$(document).scrollTop(). Bu her zaman ekranın en üstünde olacak.
Oluşturduğumuz kod çalışıyor fakat çok fazla sıçramaya hazır ve gösterişsiz. Bunun sebebi; kullanıcı kaydırma çubuğunu her kaydırdığında çok fazla "scroll" eventi gerçekleşiyor ve bunlar queue'da bekletiliyor. Bunların her biri kodumuzun navigasyon pozisyonunu sürekli güncellemesine neden oluyor. Daha önceki konularımızsa "stop" komutuyla animasyonların durdurulabildiğini görmüştük. Şimdi hem bunu kullanarak hem de bazı "easing" yapıları kullanarak kodumuzu daha iyi bir hale getirelim:

```
$(window).scroll(function() {
  $('#navigation')
  .stop()
  .animate({top: $(document).scrollTop()},'slow','easeOutBack');
});
```

Dökümanları Kaydırma

Birbiriyle ilişkili konular hakkındaki bilgilerin uzun bir listesi tek bir HTML sayfasında gösterilmek istendiği zaman, sayfanın en üstünde konuların bir köprü listesi şeklinde eklemek yaygın olarak kullanılır.

Bu dahili linkler, seçtiğiniz menü seçeneği için doğru pozisyona götürecektir. İçerikten sonra, genelde bir link olur ve bu link sizin sayfanın en başına dönmenizi sağlar. Böylece başka bir menü seçeneği seçebilirsiniz. Şimdi bu fonksiyonelliği sayfamıza eklemeye çalışalım.

İlk işimiz, sayfamızın alt bilgisine linki eklemek. Sayfanın en üstüne dönebilmemiz için, tüm yapmamız gereken linkimizin href özelliğini #'ye atamak.

Düşündüğümüzde, tüm yapmak istediğimiz sayfanın kaydırma pozisyonunu anime etmek(jQuery'de scrollTop ile ifade edilir.). Aynı zamanda, varsayılan link faaliyetine ulaşmamız gerekiyor; aksi halde animasyonumuzun gerçekleşebilmesi için gerekli zaman elde edilemeden sayfamız atlar. Eğer JavaScript'de yeniyseniz, bunu yapmanın basit bir yolu var: bir linklin tıklanma olayını üstlenen herhangi bir fonksiyon, o linkin varsayılan özelliğine çıkmak için "return false" komutunu içermeli:

```
$('a[href=#]').click(function() {
   $('html').animate({scrollTop: 0},'slow');
    return false; // Return false to cancel the default link action
}
```

Bu kodda yeni bir seçici tipi göreceğiz: "attribute selector"(özellik seçici). Bir özelliği ve kare parantez([]) içerisinde aramak istediğimiz değeri çevreleyerek, belli

bir özellik değeri ile tek bir elemanı içeren bir seçimi daraltabiliriz. Bu durumda, # href değeriyle sadece o linkleri arıyoruz.

Bu kod çalışır ama küçük bir problemle karşılaşır. Eğer kullanıcının sunucusu “quirks mod”da çalışıyorsa(HTML kodunuzda doctype tanımı yoksa browser kodu neye göre yorumlayacağını bilemez ve “quirks mode”una geçiş yapar.), *$('html')* seçicisi işini yapamayacaktır. Yukarıdaki kodu “quirks mode”da çalışabilir hale getirmek için, *$ ('body')* seçicisini kullanmalısınız. Emin olmak için ikisini de etiketliyoruz: *$('html, body')*. Bu, iki elemanı da aynı zamanda kaydırmaya çalışan Opera sunucusunda bazı problemlere sebep olabilir.

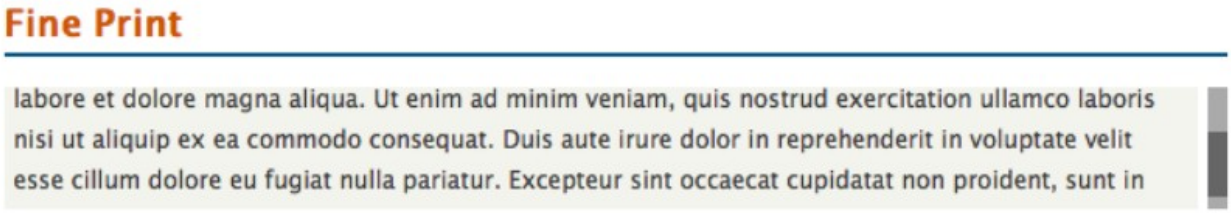
JQuery çapraz tarayıcılar sorununu çözüyor demiştiniz, diye itiraz edebilirsiniz. Adil olmak gerekirse, çoğu sorunu çözüyor. Bu yanıltıcı olanlarından. Ana jQuery kütüphanesi de bu sorunu çözecek bilgiler içermiyor ama şanslıyız ki biri bunu çözüp bir paket halinde “ScrollTo” eklentisinin içinde sunmuş: http://plugins.jquery.com/scrollto/ adresinden indirebilirsiniz. İndirip dahil ettikten sonra, yukarıdaki link fonksiyonelliğini yeniden yazabilirsiniz ve sunucu hataları hakkında endişelenmeyi bırakabilirsiniz:

```
$('a[href=#]').click(function() {
  $.scrollTo(0,'slow');
  return false;
});
```

Bu kod parçası size biraz değişik gelebilir çünkü scrollTo'yu direk jQuery'nin takma adından çağırıyoruz. Ama eklentimiz akıllı ve bu şekilde çağırdığımızda tüm pencereyi kaydırmak istediğimizi anlıyor.

Özel Kaydırma Çubukları

Müşterimiz projemizin son halini eline alıyor ve bize memnuniyetsiz bir şekilde bakıyor. Bu çubuklar gri ve çok kötü gözüküyor, böyle kalmayacak değil mi? diye soruyor. Standart işletim sisteminin bir özelliği olarak gözüken çubukları özelleştirmek/değiştirmek için farkında olmanız gereken bazı uygulamalar var. Bu tip UI özelleştirmeleri çok dikkatli bir şekilde yapılmalı. Bunun için geliştirilmiş uygun bir eklenti var: jScrollPane. Bu eklenti, sunucunun varsayılan dikey kaydırma çubuklarını, özel olanlarla değiştirmemize imkan veriyor. Eklentinin güncel haline http://code.google.com/p/jscrollpane/ adresinden ulaşabilirsiniz. Dahil etmeniz gereken iki dosya var, biri JavaScript diğeri de css dosyası. CSS dosyasında kendi çubuğunuzu yaratmak için kullanabileceğiniz özellikler var, oradaki renkleri, stilleri kullanarak değişik çubuklar yaratmaya çalışın. Biz örneğimizde ince ve şık görünümlü gri bir çubuk stili kullanacağız(herhangi bir siteye rahatlıkla uyum sağlayabilmesi için). Şekilde görebilirsiniz:



Özel kaydırma çubuklarını aktive edebilmeniz için jScrollPane fonksiyonunu çağırmalısınız. Bu fonksiyonun parametreleri düzenlenebilir. Yapabileceklerinizin tüm listesini görebilmek için şu adresi ziyaret edin: http://www.kelvinluck.com/assets/jquery/jScrollPane/jScrollPane.html Bizim örneğimizde, içeriğimizle kaydırma çubuğumuz arasında bir pay bırakıyoruz, kaydırma çubuğunun genişliğini belirliyoruz ve üst ve alt okları gizlemeyi seçiyoruz:

```
$('#fine_print').jScrollPane({
  scrollbarWidth: 10,
  scrollbarMargin: 10,
  showArrows: false
});
```

Yeniden Boyutlandırma

Kullanıcı arayüzünde yeniden boyutlandırmanın birkaç farklı anlamı var. İlk akla gelen; sunucu penceresinin yeniden boyutlandırılması. Ama uygulamalar içerisindeki pencerelerin ve resimlerin ya da diğer elemanların yeniden boyutlandırılması da yaygın olarak kullanılır.

JQuery, kullanıcı tarafından başlatılan pencere boyutlandırması hakkında bilgi toplamanın bir yolunu içeriyor ve aynı zamanda kullanıcıya sayfa üzerindeki herhangi bir elemanın boyutunu istediği gibi yeniden boyutlandırabilme imkanı sunuyor. Şimdi bunun detaylarına bakalım.

“resize” Olayı

“resize”, herhangi bir dökümanın görüntüsü yeniden boyutlandırıldığında uyarı veren ana jQuery olayıdır. Bu olayla etkileşim içine girmenizi gerektiren bir çok sebep var ama basit bir örnek incelemeden önce, olayın nasıl çalıştığını anladığımızdan emin olalım:

```
$(window).resize(function() {
  alert("You resized the window!");
```

```
});
```

index.html sayfasını sunucunuza yükleyin ve pencerenin boyutunu her değiştirdiğinizde bir uyarı aldığınızı gözlemleyin. Bunu her yaptığında uyarı almak kullanıcıyı sıkabilir, şimdi daha basit bir uygulama yapalım.

Düzen Değiştirici

CSS ile az ya da çok ilgilendiyseniz, değişken düzenlerin mi yoksa sabit genişlikli düzenlerin mi daha iyi olduğu konusunda kuşkuların olduğunu bilirsiniz. Değişken düzenler bir kullanıcının ekranının gerçek alanının maksimum kullanımını sağlarken, sabit genişlikli düzenler bir kullanıcı başka bir görüş alanı boyutuyla görüntüleğinde bile görüntünün bozulmayacağı mükemmel pikselli bir dizayn yapabilmenizi sağlarlar.

Bizim sitemiz için ikisinin de en iyisini sunabiliriz: iki ayrı sabit genişlikli düzen dizayn edelim ve “resize” olayını yakalayarak onlar arasında bir switch/değiştirme yapalım. Şu ana kadar çalıştığımız site 650px genişliğindeydi. İlk görevimiz, onu daha geniş yapabilmek için bazı stiller yazmak. 850px versiyonu için css dosyamız:

```
body #container {
  width: 850px;
}
body #container p {
  width: 650px;
}
 body #header {
   background-image: url('../../css/images/header-corners-wide.png');
}
body #celebs table {
  width: 650px;
  margin-left: 5px;
}
```

burada her birinin başına “body” koyduk ki her işleme göre önceliği olsun, aynı elemana uyguladığımız başka faaliyetler de olabilir ama önce bu özellik uygulansın istiyoruz. Şimdiki adımımız yeni stilimizi eklemek ya da silmek için jQuery kodumuzu yazmak. Tüm test etmemiz gereken body elemanının genişliği 900px'den büyük mü, eğer öyleyse head elemanımıza stilimizi eklemeliyiz değilse silmeliyiz:

```
if ($('body').width() > 900) {
   $('<link rel="stylesheet" href="wide.css" type="text/css" />')
.  appendTo('head');
```

```
} else {
        $('link[href=wide.css]').remove();
}
```

Ancak bu sefer bu kodu bir fonksiyon olarak JavaScript kodumuza yazıyoruz:

```
$(document).ready(function() {
  stylesheetToggle();
  $(window).resize(stylesheetToggle);
});
function stylesheetToggle() {
  if ($('body').width() > 900) {
        $('<link rel="stylesheet" href="wide.css" type="text/css" />')
        .appendTo('head');
  } else {
        $('link[href=wide.css]').remove();
    }
}
```

Fonksiyonumuzun ismini stylesheetToggle olarak atadık ve iki defa çağırdık: ilk olarak doküman ilk yüklenirken ve daha sonra da ne zaman yeniden boyutlandırılırsa.

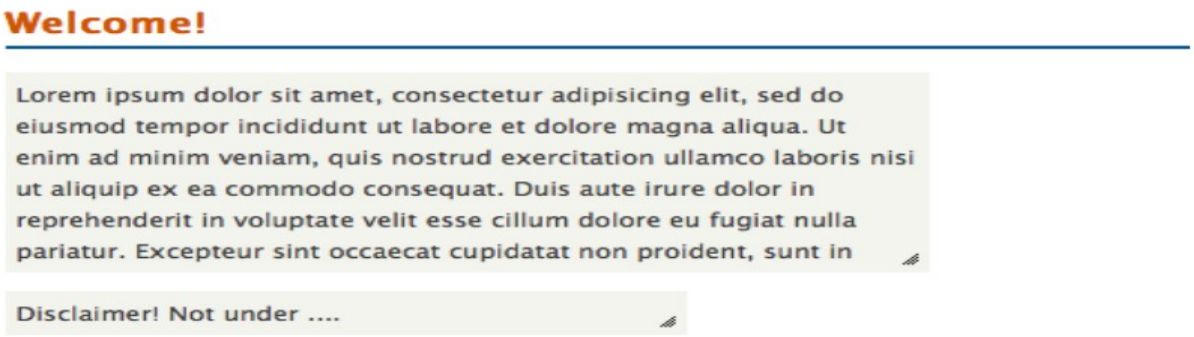
Yeniden Boyutlandırılabilir Elemanlar

jQuery UI kütüphanesi “Resizable” eklentisini içeriyor. Bu eklenti, seçtiğiniz elemanların alt köşesine küçük bir kol ekleyerek yeniden boyutlandırılabilir yapmanızı sağlıyor. Bu, kendi işletim sisteminizin penceresi gibi fare ile esnetilebilir. Daha önceden jQuery UI kütüphanesini indirdiyseniz, sınıf kullanıma hazır demektir. Aksi halde Resizable bileşeni indirip dahil etmeniz gerekir.

Bu bileşenin kullanımı çok basittir. Modifiye ettiğimiz elemanı veya elemanları seçip resizable fonksiyonunu çağırıyoruz:

```
$('p').resizable();
```

Eğer bunu kendi sitemizde çalıştırırsak, bazı alışılmadık sonuçlar alabiliriz: her bir paragraf elemanı yeniden boyutlandırılabilir olur! Sitenin genel görünümü şu şekilde oldu:



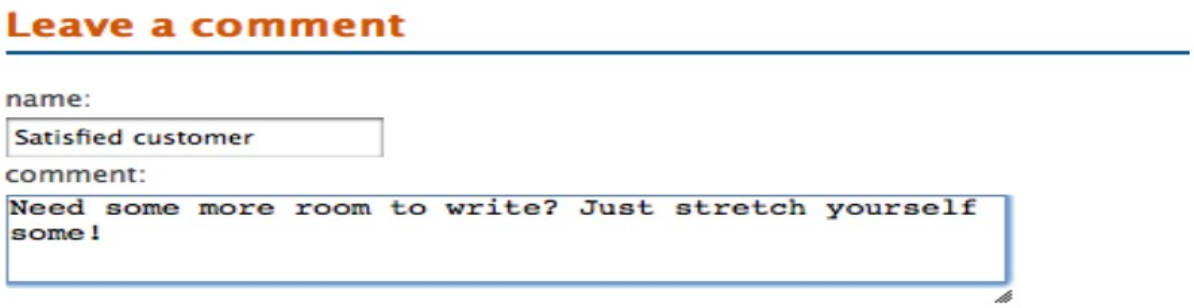
Yeniden Boyutlandırılabilir Metin Alanı(textarea)

Bazen bir arayüzün kullanılabilir olmasını sağlamakla dizaynın dengeli ve güzel olmasını çelişebilir. Fakat, jQuery bize bunun için de kolaylık sağlıyor.

Form ve fonksiyonun sık sık çarpıştığı bir alan HTML form dizaynının içinde. Bunun nedeni kısmen, kullanıcıların sitenize çılgınca farklı gereksinimlerinin olmasındandır. Örneğin, eğer siz geribildirim için bir alan sunuyorsanız, kullanıcılar ya hiç bir şey yazmamayı tercih eder, ya çok az yazar ya da çok fazla yazabilir. Dengeyi kurabilmek için küçük bir metin alanı oluşturursunuz ama onu yeniden boyutlandırılabilir yaparsınız. Şimdi, bunu jQuery UI'in Resizable fonksiyonelliğini kullanarak bunu nasıl yapabileceğimizi görelim:

```
$('textarea').resizable({
  grid : [20, 20],
  minWidth : 153,
  minHeight : 30,
  maxHeight : 220,
  containment: 'parent'
});
```

Bu kod, tüm metin alanlarımızı yeniden boyutlandırılabilir yapar.Efekt aşağıdaki şekilde gösterilmiştir. Ancak, etkiyi geliştirebilmek için birkaç yeni paremetre belirlememiz ve Resizable bileşenlerin esnekliğini göstermemiz gerekir.Daha fazla yapılandırma seçenekleri var,bunları detaylıca http://docs.jquery.com/UI/API/1.7/Resizable adresinde bulabilirsiniz.



Aynı zamanda, elemanın ne kadar gerdirilebileceğini, minHeight, minWidth ve maxHeight özellikleriyle sınırladık. Fark edeceksiniz ki, maxWidth özelliğini containment parametresinin lehine olsun diye dahil etmedik. Bu sayede resizable elemanı kısıtlayan bir container belirleyebilirsiniz.

Resizable nesnesinin belirli büyüklüğünü kapsaması için “grid” opsiyonunu kullandık. Bazıları için, resizing etkileşimine güzel bir etki eklemek gibi gözükür. “grid” iki eleman içeren bir dize(array) olarak belirlenir.Yatay(horizontal) grid size ve dikey(vertical) grid size.

İncelemek isteyeceğiniz bir başka parametre de “handles” parametresi. Bu, elemanın hangi tarafına kolun ekleneceğini ve sonuç olarak hangi yönde elemanın gerdirileceğini belirler. Parametre şu seçenekleri kabul eder: n, e, s, w, ne, se, sw, nw, ve all. Virgülle ayırarak istediğiniz sayıda bunlardan kullanabilirsiniz. Örneğin, { handles : 'n', 'se' }, elemanın üst ve sağ alt köşesine ekler.

Çerçeve Ayırıcı

Müşterimiz ana sayfada çok fazla gösterilmek istenen şey olduğunda sorun olacağını düşünüyor. Bu sorunu ayırıcı(splitter) ile çözebiliriz. Splitter, sayfa üzerindeki birçok alanı kullanıcılar için resize elemanlar haline getiren bir UI bileşeni, bu yolla, kullanıcılar her bir alan için ne kadar boşluk istediklerine karar verebilirler. Ayırıcılar, masaüstü uygulamalarında yaygın bir yere sahiptir ve zengin Internet uygulamaları sayesinde Web'de de yerini almışdır. Basit bir ayırıcıyı göstermek için Resizable bileşeniyle deneyimimizi geliştirebiliriz. Görsel bileşeni şu şekilde görebilirsiniz:

Şu an için sadece resizing fonksiyonelliğine odaklanacağız. Her bölümün içeriğini çerçeve içine dinamik olarak yüklemek 5. bölümde anlatılacak.

Ayırıcımız, her bir çerçeveyi temsil eden iki “div” elemanı içerecek(sabit boyutlara sahip bir elemanla içiçe bağlı). İçeriklerin tablosunu blok seviyesindeki bir elemanla kaplayacağız ki kullanıcı çerçeveleri yeniden boyutlandırdığında, metin bağlı

listelerimizi karıştırmasın:

```
<div id="splitter">
  <div class="pane" id="tocPane">
    <div class="inner">
      ⋮
    </div>
  </div>
  <div class="pane" id="contentPane">
    <div class="inner">
      ⋮
    </div>
  </div>
</div>
```

Şimdi, splitter.css stil dosyasına bazı basit stiller ekleyeceğiz:

```
#splitter{
  height:150px;
  margin-top:30px;
  margin-bottom:50px;
}
#splitter .pane {
  width:50%;
  height:100%;
  float:left;
}
#splitter h2 {
   margin-bottom:0;
   padding-bottom:0;
}
#tocPane {
  overflow: hidden;
  background:#d6dde5 url(handle.png) no-repeat right center;
}
#tocPane .inner {
  width: 300px;
}
#contentPane {
  overflow: auto;
}
#contentPane .inner {
  padding: 0 5px;
```

```
}
```

Yatay bir ayırıcı yaratmak için, ilk elemanı resizable yaparız ve doğuya bakan bir kol belirleriz. Böylece “div”in sadece sağ kenarı resizable olur.

Eğer sadece “resizable” komutunu içeren örneği daha önce çalıştırdıysanız, hemen hemen orada olduğumuzu farketmişsinizdir: iki eleman bir ayırıcı pencere gibi davranıyor(sağdaki elemanın genişliği kolu sürüklediğinizde kalan tüm boşluğu doldurmasının yerine sabit kalması hariç). Bunu incelemek için, resizable görsel bileşenin “resize” fonksiyonunun içinde bazı hesaplamalar yapmamız gerekiyor. Bu, bileşen yeniden boyutlandırılırken bir “event handler”ın uyarı verir:

```
$('#splitter > div:first').resizable({
   handles: 'e',
   minWidth: '100',
   maxWidth: '400',
   resize: function() {
     var remainingSpace = $(this).parent().width() - $(this).outerWidth();
     var divTwo = $(this).next();
     var divTwoWidth = remainingSpace - (divTwo.outerWidth() - divTwo.width());
     divTwo.css('width', divTwoWidth + 'px');
}
});
```

Burada, sol tarafın boyutunun değişmesi sağ tarafı da etkileyeceğinden(biri büyürken diğeri küçülecek) basit bir matematik bilgisi uygulandı koda. Kullanıma ne kadar boşluğun kaldığı hesaplandıktan sonra, ilk elemanın genişliğini belirlemek için hazırız. Düşünmemiz gereken bir şey daha var: eğer ikinci “div” bordera ya da paddinge sahipse, bunları da hesaba katmamız gerekir. Malesef outerWidth fonksiyonu sadece okuyabiliyor, bu yüzden toplam yüksekliği hesaplatmak için onu kullanamıyoruz. Bir elemanın ne kadar genişlikte olduğunu hesaplayabilmemiz için, elemanın outerWidth'inden width'ini çıkarmamız gerekir. Bu değeri remainingSpace değişkeninden çıkarırsak ikinci “div” elemanının tam olarak kaç piksel alana ihtiyacı olduğunu bulmuş oluruz. Böylece yatay ayırıcımız tamamlanmış olur.

Eğer dikey bir ayırıcı uygulamak istersek, çok küçük bir değişiklik yapmamız gerekir: çerçeve elemanlarımız yanyana dizilmek yerine üstüste dizilir ve “resizable” çağrımız güney yöne bakan bir kol kullanır. Kod neredeyse aynı ama bu sefer elemanın genişliği yerine boyuyla ilgileniriz:

```
$('#splitter > div:first').resizable({
  handles: 's',
  minHeight: '50',
```

```
  maxHeight: '200',
  resize: function() {
    var remainingSpace = $(this).parent().height() - $(this).outerHeight();
    var divTwo = $(this).next();
    var divTwoHeight = remainingSpace - (divTwo.outerHeight() - divTwo.height());
    divTwo.css('height', divTwoHeight + 'px');
  }
});
```

Bu ayırıcılar faydalı, az kod gerektiriyor ve birçok amaç için gayet uygun. Ancak, eğer sizin için daha ayrıntılı ayırıcı davranışları gerekiyorsa(çoklu ayırıcı çerçeveler ya da bağlı çerçeveler gibi), jQuery Splitter eklentisini inceleyin.

BÖLÜM 4: GÖRÜNTÜLER VE SLAYT GÖSTERİLERİ

Lightboxes

Bir Web 2.0 efektidir. Model bir diyalog içinde, bir görüntünün küçük resminin tam boyutlu versiyonunu göstermek için kullanılır.

Custom(Özel) Lightbox

Kendimiz için yaratacağımız lightbox çok basit olacak: herhangi bir HTML linki lightbox'ın sınıf ismine sahip olacak ve tıklanıldığında, linkin işaret ettiği görüntü çıkacak. Resim ekranın ortasında gözükecek ve diğer tüm alanlar kullanım dışı ve karanlık olacak. Oluşacak efekti aşağıdaki resimde görebilirsiniz:

HTML linkimizle işe başlayalım:

```
<a href=”/images/celeb01.jpg”class=”lightbox”>Pic</a>
```

Görüntü gösterildiğinde tüm ekranın karartılmasını istiyoruz, bunu nasıl yaparız? Bunun en basit yolu sayfanın tümünü içerecek büyüklükte bir “div” elemanı eklemek. Bunun içine başka bir “div” elemanı daha ekleyeceğiz ve görüntüyü buna yükleyeceğiz.
CSS dosyamız:

```
#lightbox_overlay {
  position:absolute;
  top:0;
  left:0;
  height:100%;
  width:100%;
  background:black url(loader.gif) no-repeat scroll center center;
}
#lightbox_container {
  position:absolute;
}
```

Stilimizde yükseklik ve genişliği %100, arkaplan rengini de siyah olarak ayarlıyoruz. Sonra, elemanın opaklığını ayarlayacağız ve bu sayede görüntünün gölgeli özelliğini sağlamış olacağız.

Sonra, lightbox linkimize “click handler” ekleyeceğiz ki tıklanıldığında görüntünün altında uzanan koyuluğu, “image container”ı ve görüntünün kendisini ekleyebilelim. Container şu an bizim için önemli değil fakat eğer lightbox'ınızın fonksiyonelliğini arttırmak isterseniz(border ve açıklamalar eklemek ya da Next-Previous butonları gibi), sizin için yararlı olacak. JavaScript dosyamız:

```
$('a.lightbox').click(function(e) {
// hide scrollbars!
  $('body').css('overflow-y', 'hidden');
  $('<div id="overlay"></div>')
  .css('top', $(document).scrollTop())
  .css('opacity', '0')
  .animate({'opacity': '0.5'}, 'slow')
  .appendTo('body');
  $('<div id="lightbox"></div>')
  .hide()
  .appendTo('body');
```

```
  $('<img />')
  .attr('src', $(this).attr('href'))
  .load(function() {
  positionLightboxImage();
})
.click(function() {
  removeLightbox();
})
.appendTo('#lightbox');
return false;
});
```

“Overlay”(kaplama) ekranın başında yerini alıyor ve görünmez halden %50 opaklığa ulaşıyor(arkaplan efektini sağlayabilmek için). Lightbox container sayfaya ekleniyor ve hemen gizleniyor. Container'a görüntü ekleniyor ve “src” özelliği görüntünün pozisyonuna göre ayarlanıyor. Bunu yapabilmek için jQuery'nin güçlü “attr” metodunu kullanırız(bu metod herhangi bir DOM elemanının herhangi bir özelliğini ayarlamak ya da geri almak için kullanılır). Sadece bir elemanla çağrıldığında($ (this).attr('href') gibi), o özelliğinin değerini döndürür.İkinci bir parametreyle(örneğin, *$('<img />').attr('src', …)*,sağlanan değer için bir özellik ataması yapar.
Sonra, görüntüye bir kaç “event handler” ekleriz. Bu olaylardan biri bizim için yeni: load . “ready” olayıyla bağlantılı ama bir eleman %100 yüklendiğinde uyarıyor.

Son olarak, “return false” ekleriz ki HTML linkinin default(varsayılan) davranışının olmasını önleyelim. Aksi halde, kullanıcı sayfamızdan ve görüntüden çok daha farklı bir yere yönlendirilir.

```
function positionLightboxImage() {
  var top = ($(window).height() - $('#lightbox').height()) / 2;
  var left = ($(window).width() - $('#lightbox').width()) / 2;
  $('#lightbox')
  .css({
  'top': top + $(document).scrollTop(),
  'left': left
})
.fadeIn();
}
```

Görüntü yüklendiğinde, “positionLightboxImage” fonksiyonu çağrılır. Bu fonksiyon, görüntütün ekranın tam ortasında gösterilmesiyle ilgilenir. Pencerenin yüksekliğini ve genişliğini alıp görüntünün yükseklik ve genişliğinden çıkardıktan sonra bulduğu sonucun yarısını alarak merkez noktayı hesaplar. Bu şekilde görüntünün gösterilmesi

için en uygun yer bulunmuş olur.
Geriye kalan son iş, kullanıcı görüntüye tıkladığında lightbox'ın silinmesini sağlamak. Basitçe yeni elemanlarımızı oluştururuz ve sonra sileriz ki bir sonraki görüntüye de aynı şeyler uygulanabilsin:

```
function removeLightbox() {
  $('#overlay, #lightbox')
  .fadeOut('slow', function() {
    $(this).remove();
    $('body').css('overflow-y', 'auto'); // show scrollbars!
   });
}
```

Şimdi bir lightbox'ın nasıl oluşturulabileceğini, isteğinize göre nasıl şekillendirilebileceğini öğrenmiş oldunuz. Her ne kadar çok eklenti olsa da bazen kendinizin üretmesi çok daha tatmin edicidir!

console.log ile Aksaklıkları Giderme(Troubleshooting)

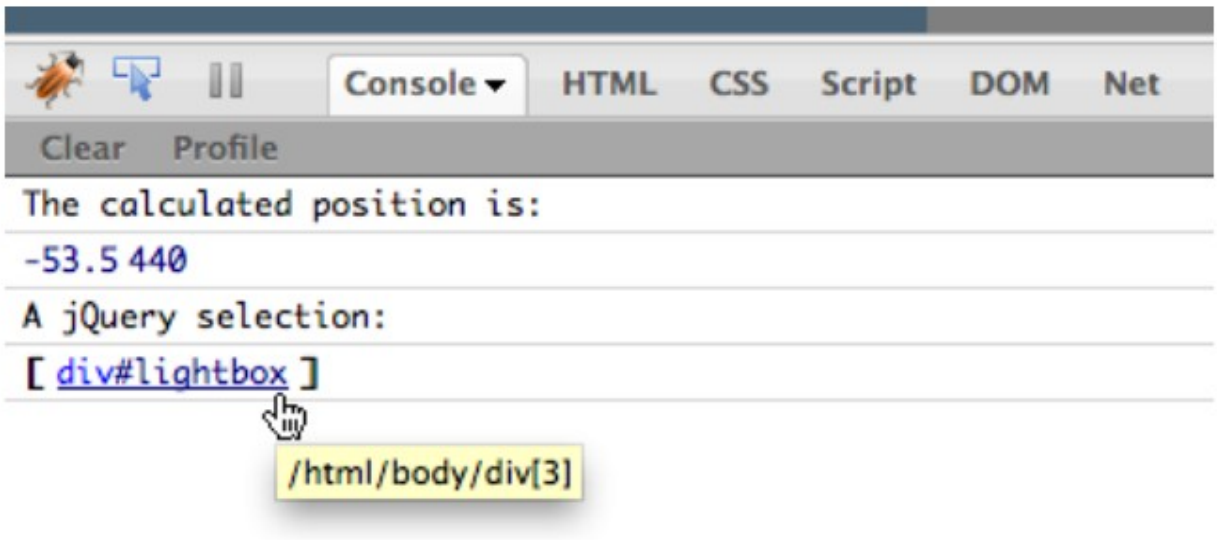
Bu basit lightbox'ı geliştirmek için uğraştıysanız eğer, bazı sorunlarla karşılaşmış olabilirsiniz, kodunuz sizin beklediğiniz gibi davranmamış olabilir. Bazen kodunuzdaki bir fonksiyonun çağrıldığında ne yaptığını ya da herhangi bir anda bir değişkenin aldığı değeri bilmek isteyebilirsiniz.

Genel olarak, bu tip bir hata ayıklama(debugging) eski bir uyarı metoduyla yapılır. Örneğin, en üstteki değişkende kodun hangi değeri sakladığını bilmeniz gerekebilir, “alert(top);” yazarsınız. Ancak, bu programın akışını engeller ve devam etmek için sizi uyarıyı kapatmaya zorlar. Ve eğer sizin ilgilendiğiniz kod parçası bir döngünün içerisindeyse, çok fazla uyarıyla uğraşmak zorunda kalabilirsiniz.

Şanslıyız ki, web geliştirme araçları gelişiyor, ve eğer siz Firefox için iyi bir Firebug kullanıyorsanız(2. Bölümde bahsedilmişti), birçok yerleşik hata ayıklama seçeneğinin avantajlarından faydalanabilirsiniz. Firebug'ın en kullanışlı özelliklerinden biri “console”(konsol)dur. Burada değişkenlerin değerlerini uyarı olarak göstermek yerine console.log komutunu kullanabilirsiniz:

```
console.log(top,left);
```

Sadece Firebug'ın Console sekmesini açın ve gösterilen değerleri göreceksiniz. Bir komut satırı içerisinde istediğiniz kadar değişkeni virgülle ayırarak belirtebilirsiniz. Farklı tip log komutları tarafından üretilen çıktılar aşağıdaki resimde gösterilmiştir:



Bir Lightbox Eklentisi: ColorBox

http://colorpowered.com/colorbox/ adresinden ColorBox'ı indirin ve içeriğini inceleyin. İçerisinde hem sıkıştırılmış hem sıkıştırılmamış hali mevcut fakat daha iyi performans alabilmek için sıkıştırılmış halini indirin, nasıl çalıştığı neleri kapsadığını incelemek için sıkıştırılmamış halini inceleyebilirsiniz. İndirdiğiniz dosyanın içinde aynı zamanda birkaç örnek var ve bunların her birinde lightbox görüntüsü farklı. En beğendiğiniz CSS dosyasını belirleyin ve uygulamanıza dahil edin.

Örneklerden birinin CSS ile görüntü dosyalarını ve sıkıştırılmış eklenti dosyasını HTML'imize dahil ettik:

```
<link rel="stylesheet" href="colorbox.css" type="text/css">
<script src="jquery.colorbox-min.js" type="text/javascript"></script>
```

ColorBox daha önce yaptığımız gibi tek bir görüntü üzerinde de çalışabilir, ama slayt gösterileri ve stil galerilerinde daha kullanışlı. Kullanıcının görüntüler arasında dolaşabilmesi için aşağıda gösterilen şekildeki gibi bir galeri hazırlayalım. Bunu yapabilmek için göstermek istediğimiz görüntüleri gruplamalıyız ve ColorBox'ın bizden beklediği gibi bunu linklerimizin “rel” özelliğiyle yapmalıyız.

Biçimlendirmede, gruplamak istediğimiz tüm görüntüler üzerinde rel=”celeb” i dahil ettik. Şimdi, o görüntüleri bulmak için jQuery özellik seçicilerini kullanabiliriz: a[rel=”celeb”]. Seçimimizde colorbox metodunu çağırmak bize hoş görümlü bir lightbox sunar:

```
$(document).ready(function() {
  $('a[rel="celeb"]').colorbox();
});
```

Bu tamamen varsayılan olarak çalışır ve o şekilde gözükür ancak değiştirebileceğimiz/üzerinde oynama yapabileceğimiz çok fazla seçenek var. Sıradaki örnekte “fade” geçisinin süresini belirleyecek olan bir hız opsiyonu veriyoruz. (Daha fazlası için ColorBox sitesini ziyaret edin ve kullanılabilir seçenekleri inceleyin. JavaScript dosyamız:

```
$('a[rel=celeb]').colorbox({
  transition: 'fade',
  speed: 500,
  current: "{current} of {total} celebrity photos"
});
```

ColorBox'ın en iyi yanı, özelleştirilebilir olmasıdır. Davranış ayarlarını

değiştirebilirsiniz, callback'ler ekleyebilirsiniz ve biçimlendirmenizi ya da eklentinin kaynak dosyasını değiştirmeden olay kancalarını(event hooks) kullanabilirsiniz.
Son olarak, ColorBox MIT Lisans izni altında kullanılabilir, bu yüzden projelerinizde bunu kullanabilirsiniz(http://creativecommons.org/licenses/MIT/)

Görüntüleri Jcrop ile Kırpmak

Jcrop eklentisi, bir görüntünün bölgelerini tanımlamak için kullanılır. Görüntü üzerine bir lightbox stili ekler ve kullanıcının bir dikdörtgeni sürükleyerek görüntü için gerekli olan alanı seçebilmesini sağlar. Bu fonksiyonellik, kullanıcıların profil resimleri için yükledikleri resimleri kırpma imkanı veren çoğu internet sitesinde kullanılıyor.

Eğer Web üzerinde görüntü işleme konusuna aşinaysanız, bu tip bir işlemenin sunucu tarafında gerçekleştiğini biliyorsunuzdur. Jcrop eklentisi aslında görüntüleri kırpmaz, kullanıcının görüntüyü kırpmak istediği sınırları tanımlamak için sezgisel bir arayüz sağlar. Eklentiden dönen sonuçlar, asıl görüntü işlemini gerçekleştirmek üzere sunucuya iletilir. Jcrop ile kırpılan görüntüyü aşağıdaki şekilde görebilirsiniz:

Jcrop eklentisinin tipik iş akışı şu şekildedir: kırpılması gereken görüntü kullanıcıya

gösterilir, ve Jcrop arayüzü oluşturulur. Kullanıcı seçimini yaptıktan sonra koordinatlar sunucuya gönderilir. Burada görüntünün son hali yaratılır ve gösterilmek/indirilmek üzere kaydedilir.

Jcrop etkileşimini uygulayabilmek için, önce indirmeli ve dosyaları çıkarmalısınız(extract). Bu işlem için http://deepliquid.com/content/Jcrop.html adresini ziyaret ediniz. Pakette bir Jcrop JavaScript dosyası, küçük bir CSS dosyası, anime edilmiş bir GIF ve tüm Jcrop özelliklerini gösteren bazı demo sayfalar var.

CSS ve JavaScript dosyalarını dahil etmelisiniz. Jcrop.gif görüntüsü de CSS dosyanızla aynı dizinde olmalı. HTML dosyamızda:

```
<link rel="stylesheet" href="css/jquery.Jcrop.css" type="text/css">
<script src="jquery.Jcrop.min.js" type="text/javascript"></script>
```

Her şey yerine oturtulduktan sonra, sayfa için seçilebilir yapmak istediğiniz bir görüntüyü eklemelisiniz. Jquery ile kolayca seçilebilir olması için görüntüye bir numara(id) verdik. Eğer görüntüler üzerindeki seçimin iyi bir şekilde yapıldığı konusunda kullanıcıya uyarı vermek isterseniz, bir buton da ekleyebilirsiniz:

```
<div id="crop">
<img id="mofat" src="../../images/mofat_400.jpg" alt="Mo'Fat"/>
<input type="button" value="crop"/>
 </div>
```

En basit formda, görüntüye jQuery eklentisini uygulamalısınız. Sayfayı yeniden yüklediğinizde, görüntü, sürükleme çubukları ve bir kaplamayla büyütülmüş olacak:

```
$('#mofat').Jcrop();
```

Arkaplan kaplamasının opaklığını,rengini ve en-boy oranını sınırlandırabileceğiniz gibi bunu kırpma alanı ve minimum-maksimum seçme boyutu üzerinde de yapabilirsiniz:

```
var jcrop = $('#mofat).Jcrop({
  setSelect: [10,10,300,350],
  minSize:[50,50],
  onChange: function(coords) {
  // use the coordinates
},
onSelect: function(coords) {
  // use the coordinates
}});
```

Burada bazı varsayılan özellikleri dahil ettik. “setSelect” varsayılan bir kırpma alanı tanımlamamızı sağlar; bunun için [x1, y1, x2, y2] formatında bir dizi koordinat yazmalıyız. “minSize” seçeneği seçimin minimum genişlik ve yüksekliğini içeren bir dizi seçeneğidir. Aynı zamanda, “onChange” ve “onSelect” olaylarını nasıl yakalayacağınızı da gösterdik. “onChange” olayı, kullanıcı sürükleme kollarını her kullandığında uyarı verecek. “onSelect” olayı ise sadece bir seçim tanımlandığında uyarı verecek(kullanıcı sürükleme işlemini bitirdiğinde).

Olay işleyiciler x, y, x2, y2, w ve h özelliklerini içeren koordinat nesnesini alacak. Bu yüzden, “handler” kodunuzda, o anki seçimin genişliğini almak için “coords.w” yazmalısınız.

Kullanıcının seçme işlemini bitirdiğinden emin olmak için bir buton koyuyoruz ki bitirdiğinde tıklayıp bizi bilgilendirsin. Bunu yapabilmek için, orijinal kodumuzu biraz değiştirmemiz gerekecek. Yukarıda yaptığımız gibi bir jQuery nesnesi üzerinde Jcrop'u çağırdığınızda, başka metodlarda da zincirleme yöntemiyle kullanılabilecek bir jQuery nesnesi döner. Ancak bu, seçim koordinatlarına ulaşmamızı sağlamaz. Bunu yapabilmek için, Jcrop'u ayrıca çağırmalıyız(direk $'dan). Bu yolla çağrıldığında, özel bir Jcrop nesnesi döner. Bu nesne seçilen koordinatlara ulaşabilmek için özellikler ve metodlar içerir. Onu hem kırpılacak nesne için bir seçici olarak hem de seçeneklerin kümesi olarak ekleriz:

```
var jcrop = $.Jcrop('#mofat',{
  setSelect: [10,10,300,350],
  minSize:[50,50]
});

$('#crop :button').click(function() {
  var selection = jcrop.tellSelect();
  alert('selected size: ' + selection.w + 'x' + selection.h);
})
```

O anki seçimi elde edebilmek için “tellSelect” metodunu kullanıyoruz. Bu metod, olay koordinatlarıyla aynı özelliklere sahip, bu yüzden onları sunucuya gönderebiliriz ve resmimizi kırpılmış hale getirebiliriz. Sunucunun olmaması durumunda, basitçe uyarı vermeyi seçtik ki neler olup bittiği anlaşılabilsin. Jcrop çok fazla seçenek ve metoda sahip bu yüzden indirdiğiniz pakette bulunan demoları incelemeniz sizin için faydalı olacaktır.

Slayt Gösterileri

Müşterimize birkaç farklı slayt gösterisi fikrimizin olduğunu söyledik. İlk olarak, çapraz solma görüntülerinin birkaç yoluna bakacağız. Bunun anlamı bir görüntü görünmeye başlarken diğer görüntünün solması/gözden kaybolmaya başlamasıdır. Sonra birkaç kayan galeriye ve son olarak daha gelişmiş dönen kitap stilindeki galerilere bakacağız. Tüm bunlarla uğraşırken yeni jQuery hilelerini de yakalamış olacağız.

Çapraz Solma(Cross-fading) Slayt Gösterileri

Web'de bunun farklı teknikleri var, gerekli olduğunda herhangi birini seçebilmeniz için size birkaç farklı metod göstereceğiz.

Rollover Fader

Daha önce incelediğimiz “hover” efektlerine benzer fakat bu kez iki durum arasında aşamalı bir “fade” uygulayacağız. İlk olarak, “hover” görüntüyü nasıl ve nerede saklayacağımız problemini çözmeliyiz. Bunu, iki görüntüyü de bir “span” içerisine koyarak çözebiliriz. “hover” görüntü ilk görüntünün üzerinde konumlandırılır ve kullanıcı fareyi üzerinden çekene kadar saklanır. Sonra saklanan görüntü ortaya çıkar. Başlamak için “rollover container”ımızı kurarız, html sayfamız:

```
<span id="fader">
<img src="../../images/glenda_200.jpg" alt="Glendatronix"/>
<img class="to" src="../../images/fader_200.jpg"
 alt="Darth Fader"/>
</span>
```

“hover” görüntüyü gizleyebilmek için “position” ve “display” özelliklerini kullanırız:

```
#fader {
position: relative;
}
#fader .to {
display: none;
position: absolute;
left: 0;
}
```

“container”ın içinde iki görüntünün tutulduğunu bilerek, onlara “eq” filtresi ile ulaşabiliriz. “image 0” bizim görünür resmimiz, “image 1” ise bizim “hover” resmimiz.
“rollover”ı uygulamak için kullanacağımız kodu inceleyiniz:

```
$('#fader').hover(function() {
  $(this).find('img:eq(1)').stop(true,true).fadeIn();
}, function() {
  $(this).find('img:eq(1)').fadeOut();
})
```

Burada “stop” komutunun gelişmiş versiyonunun kullanılması dışında yeni bir şey yok. “clearQueue” ve “gotoEnd” için true tanımlıyoruz, böylece “fade” animasyonumuz kuyrukta bekleyen herhangi bir animasyon olduğunda duracak ve nerede belirtildiyse oraya gidecek. Bu, fare hızlıca hareket ettirildiğinde animasyonların yedeklenmesini önler.

Bu efekti navigasyon butonları için kullanmayı düşünebilirsiniz. Başka bir düşünce de CSS'de bağlantıların(link) “hover” durumu arkaplan görüntüsü olarak “hover image” eklemek. Bu şekilde, “rollover”ınız JavaScript'siz geleneksel bir “hover” butonu gibi davnır.

JavaScript Zamanlayıcılar(Timers)

Web, olay güdümlü bir ortamdır. Elemanlar sayfa üzerinde durur ve kullanıcının onlar üzerinde bir işlem yapmalarını beklerler(tıklama,kaydırma,seçme vb). Bu beklenen olay gerçekleştiğinde, kodumuz hayata geçer ve bizim istediğimiz fonksiyonelliği yürütür. Fakat bazı durumlarda kullanıcının herhangi bir eylem gerçekleştirmesini beklemeden belirli bir sıklıkta kendiliğinden uygulanmasını istediğimiz görevler olabilir. Bu şimdi göreceğimiz slayt gösterilerini yapılandırmamız için geçerli olan durum. Otomatik olarak bir dizi görüntünün birkaç saniyede bir değiştirilerek gösterilmesini istiyoruz. Kullanacağımız iki metod var: “setTimeout” ve “setInterval”.

Zamanlayıcı fonksiyonlarının ikisi de aynı şekilde çalışıyor.Onlara verdiğimiz kodu çalıştırmadan önce belirli bir süre bekliyorlar.İki kodunda yazım kuralı birbirine benzer şekilde:

```
setTimeout(<code to run>, <number of milliseconds to wait>);
setInterval(<code to run>, <number of milliseconds to wait>);
```

Ana farkları ise şudur; setTimeout belirlenen bir zaman aralığı kadar bekler, ona verdiğimiz kodu çalıştırır ve sonra durur. SetInterval ise bekler, kodu çalıştırır sonra tekrar bekler tekrar çalıştırır(daima tekrar eder ya da ona durmasını söylediğimiz zaman durur). Eğer bu metodlara yazdığımız kod, bir elemanın görünür bir özelliğini güncelliyorsa ve ona atadığımız gecikme süresi çok azsa, animasyonun yanılsamasını(göz aldatmacasını) elde edebiliriz. Aslında bu, herhangi bir animasyon

fonksiyonunu kullandığımızda jQuery'nin sahne arkasında ne yaptığıdır.

Bir Zamanlayıcı Ayarlama

Burada zamanlayıcıların çalışma şeklini göstermek için basit bir örnek yapacağız. Basitçe ekran boyunca yeşil bir kutuyu hareket ettireceğiz. Tabiki bunu yapmak için jQuery'nin "animate" metodunu kullanabiliriz ama JavaScript'in içinde nasıl gerçekleştiğini öğrenmek istiyoruz. Şimdi kutularımızı oluşturalım:

```
<div>
<div id="green" class="box">Go!</div>
<div id="red" class="box">Go!</div>
</div>
```

Kutular hala yerinde duruyor, onlar anime edebilmemiz için bir zamanlayıcıya ihtiyacımız var. setInterval zamanlayıcısını kullanacağız çünkü kodumuzun sürekli çalıştırılmasını istiyoruz.JavaScript kodu:

```
var greenLeft = parseInt($('#green').css('left'));
  setInterval(function() {
    $('#green').css('left', ++greenLeft);
}, 200);
```

"div" elemanımız yavaşça ekranda hareket ediyor. Her 200 milisaniyede onu 1 piksel sağa itiyoruz. Gecikmenin süresini değiştirmek, animasyonun hızını etkiler. Dikkat etmeniz gereken bir şey var: eğer gecikme süresi çok düşük olursa(50 milisaniyeden daha az) ve her bir döngü için çok fazla DOM değişikliği yaparsanız, ortalama kullanıcıların tarayıcısı duracaktır. Bunun sebebi, sizin istediğiniz her şeyi yapmak için bilgisayarlarının yeterince zamana sahip olmamasıdır.

"setInterval"ın fonksiyonelliğini "setTimeout" fonksiyonunu kullanarak oluşturmak da mümkün. Kodumuzu biraz değiştiriyoruz:

```
var redLeft = parseInt($('#red').css('left'));
  function moveRed() {
    setTimeout(moveRed, 200);
    $('#red').css('left', ++redLeft);
}
moveRed();
```

Burada moveRed isimli bir fonksiyon çağrılıyor bunun içerisinde bir setTimeout zamanlayıcısı var. "setTimeout" bir kez çalıştıktan sonra sadece moveRed fonksiyonunu çağıracak. Ancak, bu fonksiyon içerisinde zamanlayıcı çağırdığı için,

kendisini tekrar ve tekrar çağıracak. Böylece “setInterval” ile aynı işi yapmış olacak.

Zamanlayıcıları Durdurmak

Genellikle, zamanlayıcılarımızın daima çalışması istenmeyen bir durumdur. Çalıştırmaya başladığınız zamanlayıcıları, uygun JavaScript komutlarından birini çağırarak durdurulabilirsiniz.“clearInterval” ya da “clearTimeout” kullanılabilir:

```
clearInterval(<timer id>);
clearTimeout(<timer id>);
```

Bu fonksiyonlardan herhangi birini çağırmak için, zamanlayıcının numarasını yazmalıyız. Numarasının ne olduğunu nasıl bilebiliriz? Numara, siz onu yarattığınızda atadığınız sayıdır. Eğer zamanlayıcıyı ileriki bir zamanda durdurmayı düşünüyorsanız, o sayıyı bir değişkende tutmalısınız:

```
var animationTimer = setInterval(animate, 100);
```

Şimdi zamanlayıcı herhangi bir zamanda aşağıdaki kod ile durdurulabilir:

```
clearInterval(animationTimer);
```

Solma(Fading) Slayt Gösterileri

İki görüntü arasında çapraz solma yapmak çok basittir; biri görünmeye başlarken diğeri gözden kaybolacak. Eğer bu düşüncemizi daha fazla görüntü üzerinde uygulamayı düşünürsek, örneğin dönen resim galerisi, bunun yapımı daha zor olur. Şimdi, sıradaki gösterilecek olan görüntüyü bulmaya çalışmalıyız.

Jquery resim galerilerinde göreceğimiz en yaygın hile, o anki resim dışındaki tüm resimleri gizlemektir. Değiştirme zamanı geldiğinde, o anki resim gözden kaybolurken sıradaki resim görünmeye başlar. Bu gerçek bir çapraz solma değil fakat uygulaması kolaydır bu yüzden ilk olarak bu örneğe bakacağız, sonraki örnekte ise gerçek çapraz solmanın nasıl olduğunu göreceğiz. Slayt gösterimiz bir “div” elemanı içerisinde birkaç tane görüntü içerecek. “show” sınıfını atayarak resimlerden birini görünür olarak belirleyeceğiz:

```
<div id="photos">
  <img alt="Glendatronix" class="show" src="../../images/glenda_200.jpg" />
  <img alt="Darth Fader" src="../../images/fader_200.jpg" />
  <img alt="Beau Dandy" src="../../images/beau_200.jpg" />
  <img alt="Johnny Stardust" src="../../images/johnny_200.jpg" />
  <img alt="Mo' Fat" src="../../images/mofat_200.jpg" />
```

```
</div>
```

Varsayılan olarak tüm görüntüleri gizleyeceğiz. Bu slayt gösterisinde “show” sınıfının iki amacı var: gösterme işlemini gerçekleştirebilmemiz için CSS'de etiketleyebilmeyi sağlar, ve o anki resim için bize bir kol/tutacak verir.

Şimdi, resimlerimiz üzerinde bir döngü yaratabilmek için JavaScript zamanlayıcımızı çalıştırmaya başlamalıyız. Her 3 saniyede bir kendisini çağıran bir fonksiyon yazacağız:

```
$(document).ready(function() {
  slideShow();
});

function slideShow() {
  var current = $('#photos .show');
  var next = current.next().length ? current.next() : current.parent().children(':first');
  current.hide().removeClass('show');
  next.fadeIn().addClass('show');
  setTimeout(slideShow, 3000);
}
```

O anki resmin “show” sınıfına ait olduğunu biliyoruz bu yüzden onu kolayca seçebiliriz ama sıradaki resme nasıl ulaşacağız? Eğer sırada bir kardeş varsa seçeriz yoksa ilk resmi seçeriz. Böylece slayt gösterimiz kendi etrafında dönebilir. Son olarak, o anki resmi gizleriz ve bir sonrakini gösteririz. Aynı zamanda, “show” sınıfını eski fotoğraftan kaldırıp yenisine ekleriz ve slideShow metodu için bir zaman aşımı belirleriz ki her 3 saniyeden sonra kendisini tekrar çağırabilsin.

Gerçek Çapraz Solma

Bu uygulama için izleyeceğimiz adımlar şu şekilde:

1) En yüksek z-indeksine sahip olan resim gösterilebilsin diye tüm resimleri birbiri üzerine kümeleyin.
2) Sıradaki resim görüntülenebilsin diye en üstteki resmi görünmez yapın.
3) Solma işlemi gerçekleştikten sonra, o anki resim en üste gelebilsin diye görüntülerin z-indekslerini güncelleyin.
4) 2. ve 3. adımları tekrar edin.

Bu tekniğin dezavantajı şu: bütün resimleri üstüste kümelediğimiz için hepsi aynı boyutta olmalı. Bu genellikle küçük bir sorun çünkü Web üzerinde belirli bir alan oluşturmak çok zor değildir.

Resimlerimizi kümelendirerek işe başlayalım:

```
<p id="photos">
  <img src="images/BarronVonJovi.png" />
  <img src="images/TheComputadors.png" />
  <img src="images/Bosom.png" />
  ⋮
</p>
```

Resimleri tamamen yerleştirmeli ve hepsini sınırlayıcı bir kutu içerisine koymalıyız:

```
#photos img {
  position: absolute;
}
#photos {
  width: 180px;
  height: 180px;
  overflow: hidden;
}
```

Şimdi, resimlerimiz kümelendi, geriye dönüp bazı planlamalar yapalım. 4. adımı gerçekleştirebilmek için(sıradaki resim için solma işlemini tekrarlama), solma(“fading”) kodumuzu bir çeşit döngü içine koymalıyız. Döngü o anki resmi izlemek zorunda olduğu için, kodumuzu ayrı bir fonksiyona taşıyoruz ve bu fonksiyonu *“$(document).ready”* bloğumuzda çağırıyoruz:

```
$(document).ready(function() {
  rotatePics(1);
}
```

Şimdi “rotatePics” fonksiyonunu yaratalım. Bu metod, o anki fotoğrafın indeksi olacak numarayı alıyor. Bunun kısaca nasıl kullanıldığını göreceğiz ama önce, tüm fotoğrafların sayısını bir değişkende tutuyoruz(çünkü bunu kodun farklı yerlerinde birkaç kez kullanacağız):

```
function rotatePics(currentPhoto) {
  var numberOfPhotos = $('#photos img').length;
  currentPhoto = currentPhoto % numberOfPhotos;
  ⋮
}
```

Kodumuzun ikinci satırı, değerlerin verilen aralıkta sınırlandırıldığından emin olmak için yapılan en yaygın hilelerden biridir. “currentPhoto”nun hiçbir zaman

fotoğrafların toplam sayısından daha büyük olmasını istemeyiz, bu yüzden fotoğraf indeksinin geçerli olduğundan emin olmak için bir hesaplama yaparız(uzunluğun modunu alarak). Mod, JavaScript'de "%" sembolüyle gösterilir ve bölme işleminin kalanını döndürür. Bu yüzden, eğer toplamda 5 fotoğrafımız varsa, ve index olarak 6 yazıyorsak, işlem 6%5 = 1 sonucunu verir. Bu sayede, var olmayan bir resmi göstermeye çalışmayacağımızdan emin oluruz.

Artık gerçek çapraz solma işlemimizi yürütebiliriz. O anki fotoğrafı yakalayabilmek için jQuery'nin "eq" komutunu kullanırız(eq = equals).Bu komut, bir grup içerisindeki elemanlardan indeksini yazdığımızla diğerleri arasından indeks numarasına eşit olanını seçer. JavaScript kodumuz:

```
$('#photos img').eq(currentPhoto).fadeOut(function() {
  // re-order the z-index
  $('#photos img').each(function(i) {
    $(this).css('zIndex', ((numberOfPhotos - i) + currentPhoto) % numberOfPhotos);
  });
  $(this).show();
  setTimeout(function() {rotatePics(++currentPhoto);}, 4000);
});
```

Fonksiyonu tekrar çağırdığımızda currentPhoto'nun 1 arttırılmış olmasını istediğimiz için ++ currentPhoto yazdık(önce arttırıyor sonra o değeri döndürüyor). Eğer currentPhoto++ yazsaydık o anki değeri döndürüp daha sonra arttıracaktı.

Eklentilerle İleri Düzey Solma(Fading) İşlemleri

Şimdi, slayt gösterisi efektini elde etmek için kullanabileceğiniz iki eklentiye bakacağız.Biri basitken diğeri daha fazla özelliğe sahip.

"InnerFade" ile Kayan Yazı Bandı

InnerFade, bir dizi eleman arasında geçiş yapmanızı sağlayan küçük bir eklentidir. Bize sağladığı birkaç kolaylık: parçaları rastgele bir sırada gösterebilme, aktif elemana bir sınıf ismi verme ve farklı animasyon tipleri seçme. İndirmek için şu adresi ziyaret edin http://medienfreunde.com/lab/innerfade/ ve sonra HTML kodunuza ekleyin:

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

Sonra, elemanlarımızı ve kaydırmak istediğimiz maddeleri ayarlıyoruz. "Container" olarak sırasız bir liste ve eleman olarak da liste maddelerini kullanacağız:

```
<div id="news">
  <h2>News</h2>
  <ul>
  <li><a href="#">Barron Von Jovi spotted …</a></li>
  <li><a href="#">Mo'Fat signs up-and-coming rapper …</a></li>
  <li><a href="#">Glendatronix rumored to be …</a></li>
  <li><a href="#">Man claims to be Darth Fader's son …</a></li>
  </ul>
</div>
```

Dökümanımız hazır olduğunda, liste üzerinde eklentinin “innerfade” metodunu kullanırız. Bir slayt efekti belirleyeceğiz(solma efekti yerine) ki kayan yazı bandımızı tamamlayabilelim. Aynı zamanda, rastgele dönen elemanlara sahip olacağız:

```
$('#news ul').innerfade({
  animationtype: 'slide',
  speed: 750,
  timeout: 2000,
  type: 'random'
});
```

Basit ve güzel bir efekt yaratmış olduk.

Döngü(Cycle) Eklentisi:

Döngü(Cycle) sınırsız özellikli ve gelişmiş bir eklentidir. İçeriğinin bir sonucu olarak eklentinin sıkıştırılmış hali 25 KB olsa da kullanmaya değer çünkü etkileyici geçiş efektleri sunuyor. Resim galerilerini daha ilginç bir biçimde göstermek için çok uygun.

Eklentiyi indirin (http://jquery.malsup.com/cycle/)ve JavaScript dosyasını HTML'inize ekleyin.

Önceki slayt gösterilerimizle aynı şekilde başlayacağız. Cycle eklentisiyle kolayca çapraz solma yapabilirsiniz ancak bu eklentinin bize sunduğu daha süslü seçenekler var, şimdi onlardan birini deneyelim,resimleri karalım:

```
$('#photos').cycle({
  fx: 'shuffle'
});
```

Bu efektin sonucunu aşağıdaki şekilde görebilirsiniz:

Eklenti bize 20den fazla seçenek sunuyor; *shuffle, fade, zoom, wipe, toss, curtainX, growY,* ... Ayrıca, kendi geçiş efektlerinizi ekleyerek eklentiyi özelleştirebilirsiniz (eğer kendi ihtiyacınıza uygun bir şey bulamazsanız).

Şimdi daha karışık bir örnek deneyelim:

```
$('#photos').cycle({
   fx: 'scrollDown',
   speedIn: 2500,
   speedOut: 500,
   timeout: 0,
   next: '#photos'
});
```

Zamanaşımı(timeout), geçişler arasındaki zamanı kontrol eder ama 0 değerinin anlamı nedir? Bu durumda “anime etme” anlamına gelir. Bunun yerine, bir elemanı seçmek için “next” seçeneğini kullandık ki tıklanıldığında bir sonraki slayta geçsin. Bu seçici, slayt gösterisinin ta kendisi! Bu yüzden sıradaki resme geçmek için, sadece resmin üzerine tıklarsınız.

Ek olarak, “in” ve “out” animasyonlarının süresini belirlemek için “speedIn” ve “speedOut” seçeneklerini kullandık sıradaki resim yavaşça gelirken son resmin hızlı

bir şekilde gözden kaybolmasını seçtik.

Slayt Gösterilerini Kaydırma

Görüntüler arasında geçiş yapabilmenin tek yolunun çapraz solma işlemi olmadığını Cycle eklentisini kullanarak öğrenmiş olduk. Bundan sonraki birkaç örnekte, etkileşimli slayt gösterileri yaratmanın başka tekniklerini keşfedeceğiz. Bütün resimlerimizi büyük bir "container" içerisine koyacağız ve tümünü gizleyebilmek için kapsayıcı bir eleman kullanacağız. Sonra ne zaman farklı bir resim göstermek istesek, sadece elemanı istenen pozisyona kaydıracağız.

Küçük Resim Kaydırıcı

Bir kaydırma galerisinde ilk odaklanacağımız iş, küçük resimlerin yatay bir listesini tutmak. Siz tıkladıkça liste daha fazla resim göstermek üzere kayacak.

Bu kontrolü yapabilmek için iki iç içe geçmiş elemana ihtiyacımız olacak. Çocuk eleman büyük olacak ve tüm resimleri içerecek. Ebeveyn eleman ise sadece görüntü alanı kadar büyük olacak(kullanıcıya göstermek istediğimiz alan kadar). Çocuk eleman hareket ettikçe, kullanıcı içerik kayıyormuş gibi görecek. Html dosyamızı inceleyelim:

```
<div id="photos">
  <div id="photos_inner">
  <img alt="Glendatronix" src="../../images/glenda_100.jpg" />
  <img alt="Darth Fader" src="../../images/fader_100.jpg" />
   ⋮
  <img alt="Beau Dandy" src="../../images/beau_100.jpg" />
  </div>
 </div>
```

En dıştaki eleman, artan içeriği gizlemeli yani bir taşma(overflow)ya ihtiyacı var: hidden. Kaydırıcımız için, en içteki elemanı 15 küçük resmimize yetecek genişlikte tanımlıyoruz:

```
#photos {
  overflow: hidden;
  width: 600px;
 }
#photos_inner {
  height: 100px;
  width: 1500px;
  overflow: hidden;
```

```
  position: relative;
}
#photos_inner img {
  float: left;
  width: 100px;
  height: 100px;
}
```

“container”ımız tamamen ayarlandıktan sonra, resimleri kaydırma işlemi için ilk adımı atalım:

```
$('#photos_inner').toggle(function() {
  var scrollAmount = $(this).width() - $(this).parent().width();
  $(this).animate({'left':'-=' + scrollAmount}, 'slow');
}, function() {
  $(this).animate({'left':'0'}, 'slow');
});
```

İlk olarak, ne kadar kaydıracağımızı hesaplamalıyız. Resimleri içeren ve taşmış olan elemanın genişliğini alıp ebeveyn “container”ın genişliğinden çıkarırız. Ebevenyn faaleyeti(action) bir elemanın o anki(en yakın) ebeveynini seçer. Bu bilgiyi, resimlerin en sonuna ulaşmak için ne kadar kaydırmamız gerektiğini bulmak için kullanırız. Resimlere tıkladığımızda, kaydırma efektimiz, resimlerin başlandıgıcı ve bitişi arasında değişir.

1. Eğer resimler iki ekran genişliğinden daha azsa, bu yaklaşım mükemmel fakat daha fazlaysa farklı bir yaklaşımda bulunmalıyız. Bu durumda kaydırma işlemi için en iyi yol, belirli bir zamanda yarım ekranın görüntü değerini kullanmaktır. Listenin sonuna ulaştığımızda, başlangıca dönmek için kaydırırız. Şimdi, daha elverişli bir kod elde edebilmek için önceki kodumuzu biraz genişletelim:

```
$('#photos_inner').click(function() {
  var scrollAmount = $(this).width() - $(this).parent().width(); 1
  var currentPos = Math.abs(parseInt($(this).css('left'))); 2
  var remainingScroll = scrollAmount - currentPos; 3

  // Scroll half-a-screen by default
  var nextScroll = Math.floor($(this).parent().width() / 2); 4

  // But if there isn't a FULL scroll left,
  // only scroll the remaining amount.
  if (remainingScroll < nextScroll) { 5
    nextScroll = remainingScroll;
  }

  if (currentScrollPos < scrollAmount) { 6
    // Scroll left
    $(this).animate({'left':'-=' + nextScroll}, 'slow');
  }
  else{
    // Scroll right
    $(this).animate({'left':'0'}, 'fast');
  }
});
```

Bu kod çok daha uzun fakat satır satır incelediğinizde hiç de karışık olmadığını göreceksiniz. Bir çok farklı değişken kullanıyoruz fakat hepsine anlamlı isimler verdiğimiz için kodun daha okunabilir olmasını sağlıyoruz:

1) Önceki örnekteki gibi, sahip olduğumuz toplam kaydırma boşluğumuzu hesaplıyoruz: “scrollAmount” değişkenimiz.

2) Yeni kaydırıcımız iki ekran genişliğinden daha fazla görüntü değerini kapsamak zorunda olduğu için, o anda ne kadar uzaklıkta olduğumuzu da anlamamız gerekiyor. O anki kaydırma pozisyonumuzu pozitif bir sayıya çevirmek için, bir JavaScript fonksiyonu olan Math.abs()'yi kullanıyoruz(çünkü sola kaydırmanın anlamı elemanları negatif bir bölgeye kaydırmak). Demek istediğimiz, bu işlem sonucunda sayının kendisi pozitif de olsa negatif de olsa gerçek değeri daima onun pozitif değeri olarak dönecektir. Örneğin; Math.abs(3)=3 iken Math.abs(-3) de 3'tür.

3) Toplamda ne kadar boşluk olduğunu ve ne kadar uzaklıkta olduğumuzu biliyoruz. Bu yüzden ne kadar uzağa gidebileceğimizi bulmak çok kolay. Sadece sonuncu sayıyı ilk sayıdan çıkar.

4) Şimdi, ne kadar uzağa kaydıracağımızı hesaplamalıyız. Varsayılan olarak bu değer, resim “container”ımızın toplam genişliğinin yarısı kadar olmalı. Kaydırmak istediğimiz uzaklığı nextScroll değişkeninde tutuyoruz.

5) Eğer kaydırmak istediğimizden daha az boşluk varsa, nextScroll değişkenimizi bize sadece resimlerin sonunu getirmesi için değiştiriyoruz.

6) Son olarak, kaydırıyoruz. Eğer zaten resimlerin sonuna ulaşmadıysak, hesapladığımız değer kadar resimleri sola kaydırıyoruz. Aksi halde(eğer resimlerin sonundaysak), resimlerin başlangıcına kaydırıyoruz.

Eğer basit bir efekt için çok fazla kod yazdığımızı düşünüyorsanız, haklısınız! Keşke tüm bu matematik işlemlerini yapmadan kaydırma içeriğini yapabilmenin bir yolu olsaydı...

“scrollTo” ile Bir Kaydırma Galerisi

Daha önce 3. bölümde sayfayı kaydırmak için faydalı bir eklentiden söz etmiştik: scrollTo. Bu eklenti, sayfa kaydırmak kadar taşan elemanları kaydırmak konusunda da başarılı. İlk kaydırma işlemi girişimimizde bir çok hesaplama yapmıştık, scrollTo eklentisi bunların çoğunu otomatik olarak bizim için hesaplıyor, bu yüzden daha karmaşık özellikler kullanabiliriz.

Bu demoda, küçük resim listemizi sileceğiz ve yerine daha büyük resimler koyacağız. Büyük resimler bir ızgara(grid)da tutulacak ama herhangi bir anda resimlerden sadece bir tanesini göreceğiz. Kullanıcı resme tıkladığı zaman, ızgarayı kaydıracağız ve yeni bir resimde duracağız(rastgele).

Başlarken, resimlerin bir listesine ihtiyacımız olacak. Kolaylık olsun diye, onları “div container”ının içinde resim etiketlerinin bütünü olarak tutacağız. Html kodumuz:

```
<div id="pic_container">
  <div id="pic_scroller">
    <img alt="Glendatronix" src="../../images/glenda_200.jpg" />
    <img alt="Darth Fader" src="../../images/fader_200.jpg" />
    ⋮
    <img alt="Beau Dandy" src="../../images/beau_200.jpg" />
  </div>
</div>
```

“pic_container div” in genişliğini 3 resim kadar, yüksekliğini de 4 resim kadar yapacağız. Yani, resimler 3x4'lük bir ızgara üzerinde olacak. Bu demo için, 200x200 piksellik resimler kullanacağız. Bu yüzden “container”ımız 800 piksel genişliğinde ve 600 piksel yüksekliğinde olmalı. Sonra, görünür alanı tek bir resmin boyutu kadar yapacağız ve geri kalanını saklayacağız:

```
#pic_container {
  overflow: hidden;
  height: 200px;
  width: 200px;
```

```
  margin-bottom: 15px;
}
#pic_scroller {
  height: 600px;
  width: 800px;
  overflow: hidden;
}
#pic_scroller img {
  float: left;
}
```

Izgaramız hazır. Şimdi, tüm resimleri tutup rastgele bir tane seçeceğiz ve scrollTo kullanarak kaydıracağız:

```
$('#pic_scroller').click(function() {
  var numberOfPics = $(this).find('div > img').length;
  var next = Math.floor(Math.random() * numberOfPics);
  $(this)
  .scrollTo(
  '#photos_inner>img:eq(' + next + ')',
  {duration: 1000}
  );
});
```

Bu kod bi önceki örneğimizden çok daha basit! Eklenti neredeyse tüm karmaşık bölümlerle ilgileniyor. Burada iki yeni jQuery elemanı var: “find” faaliyeti ve “:eq” filtresi. “find” fonksiyonu, ana $ seçicisiyle aynı, tek farkı sadece o anki seçilen elemanlar içinde arıyor olması(tüm doküman yerine). “:eq” filtresi ise daha önce gördüğümüz “eq” faaliyeti gibi çalışıyor, tek farkı bunun filtre olması. Böylece onu bir seçici dize(string) içinde kullanabilirsiniz. Ona 0 ile seçilecek olan resimlerin toplam sayısı arasında rastgele bir sayı yazarız.

“data” Faaliyetiyle Daha Akıllıca Bir Kaydırma İşlemi

Bir önceki örnekte tasarladığımız bileşende büyük bir problem var: eğer çok az sayıda resmini varsa, bir sonraki olarak rastgele seçilen resim o anki resimle aynı olabilir! Bu durumda herhangi bir kayma işlemi gerçekleşmez ve hata gibi gözükebilir. Bunu çözmek için, o anki görüntünün ne olduğunu bilmemiz gerekir:

```
$('#pic_scroller').click(function() {
  var numberOfPics = $(this).find('div > img').length;
  var last = $(this).data('last');
  var next = Math.floor(Math.random() * numberOfPics);
```

```
  if (next == last) {
  next = (next + 1) % numberOfPics;
}
$(this)
  .data('last', next)
 .scrollTo(
 '#photos_inner>img:eq(' + next + ')',
 {duration: 1000});
});
```

Yeni ve güçlü bir jQuery faaliyeti kullanıyoruz: data. Bu faaliyet bu ana kadar gördüğümüz faaliyetlerden farklı çünkü herhangi bir jQuery nesnesinde bilgi saklamamızı sağlıyor. İlk parametre veri öğesinin ismi olur ve ikincisi de saklamak istediğimiz değer olur. Sonra, veriyi tekrar alabilmek için, sadece bir parametre yazarız: veri öğresinin ismi.

Gelişmiş örneğimizde, kaydırma yapacağımız resmi bulduktan sonra, $(this).data('last', next) komutuyla eleman sayısını tutarız. Kaydırıcı elemana bir sonraki tıklanışta, $(this).data('last') komutuyla onu tekrar okuruz. Eğer yeni elemanımız öncekiyle aynıysa, bir sonraki resme kaydırmak için 1 ekleriz(hala toplam resim sayısının içinde çalıştığımızdan emin olmak için mod operatörü kullanırız).

Yukarıda koyu olarak belirttiğimiz satırları dikkatlice çalışın. Bu satırlarda veriyi yeniden alıyoruz ve set ediyoruz.

“iPhoto” - Slayt Gösterisi Görsel Bileşeni Gibi

iPhoto, Mac OS X'a dahil edilen resim galerisi uygulamasıdır. İphoto'yu kullanarak daha gelişmiş bir slayt gösterisi oluşturacağız. Resmin sol ya da sağ tarafına fareyi getirmek, önceki ya da sonraki resme geçmeyi sağlayacak, böylece kullanıcı isteğine göre galeride dolaşabilecek. Bu, şu ana kadar gördüğümüz en gelişmiş jQuery ve bunu kavramak ilk bakışta biraz zor gelebilir ama endişelenmeyin.

Tanıdık bir resim listesiyle slayt gösterimizin tabanını oluşturalım. Listeyi bir “div” içerisinde tutacağız. Bu eleman küçük resimleri sınıflandırabilmemize ve bir tutturucu eklememize olanak sağlayacak:

```
<h2>Around town last night</h2>
<div id="photos">
  <a href="#" class="trigger">Image Gallery</a>
  <ul id="photos_inner">
    <li>
```

```
      <img alt="Glendatronix" src="../../images/glenda_400.jpg" />
    </li>
    <li>
      <img alt="Darth Fader" src="../../images/fader_400.jpg" />
    </li>
     ⋮
  </ul>
</div>
```

Eğer kullanıcı sayfamızı CSS ve JavaScript yokken görüntülerse, büyük bir resim yığını olarak görecek. Onlar için de içeriğimize tam ulaşım sağlamalıyız. Eğer sadece JavaScript kullanım dışıysa, resimlerden sadece biri görünecek ama galerinin üzerinde tamamına erişebilmek için bir link koyacağız. Böylece, kullanıcı ona tıklayarak basit bir HTML galeri sayfasını görüntülemiş olacak. CSS kullanarak slayt gösterisi geliştirmelerimize başlayalım:

```
#photos {
  border: 1px solid #BEBEBE;
  height: 400px;
  overflow: hidden;
  position: relative;
  width: 400px;
}
#photos ul {
  left: 0;
  list-style-type: none;
  margin: 0;
  padding: 0;
  position: absolute;
  top: 0;
  width: 2400px;
}
#photos li {
  float: left;
}
#photos .trigger {
  left: 0;
  position: absolute;
  top: 0;
  z-index: 10;
  text-indent: -9999px;
  height: 400px;
  width: 400px;
```

```
  display: block;
}
```

“container div” ile başlayarak, görünüm sınırlandırmalarını ayarlarız. Resimlerimiz 400x400 piksel, bu yüzden “container”ımızın genişliği de bu şekilde.

Görsel Bileşen Yaratma

Görsel bileşen için çok farklı tanımlamalar yapılabilir fakat bizim burada kullanacağımız anlamı; fonksiyonelliğin bağımsız bir parçasıdır(çünkü gelecekteki projelerimizde yeniden kullanacağız). Bir görsel bileşenin gerçek amacı; kodumuzu kullanışlı bir paket haline getirmektir.

Görsel bileşenimizin temeli için kalıcı bir JavaScript nesnesi(görsel bileşenimizin ismini tanımlamak için) : var gallery = {} . Şu ana kadar gördüğümüz gibi, kalıcı nesneler süslü parantez({}) içerisinde gösteriliyor. En başta nesnemiz boş ama birazdan dolduracağız(karmaşayı önlemek adına baştan tüm kodu içerisine koymak mantıklı değil bu yüzden boş bırakıyoruz).

Sonra görsel bileşenimizi fonksiyonları gerçekleştirebilir hale getirmek için özellikler ve metotlar ekleyebiliriz. Nesnemize özellikler ekleyerek, sayfadaki herhangi bir değişkenin diğer scriptler tarafından yeniden yazılma riskini önlemiş oluruz:

```
gallery.trigger = $("#photoshow .photoshow-trigger");
gallery.content = $("#photoshow .photoshow-content");
gallery.scroll = false;
gallery.width = 240;
gallery.innerWidth = gallery.content.width();
gallery.timer = false;
```

“gallery.timer = false” yazmakla şu şekilde yazmamız aynı şey:

```
var gallery = {
  var timer = false;
}
```

“.(dot)” notasyonu, nesnenin beyan edildiği yerin dışından o nesnenin özelliklerini okuma ve yazma işlemleri yapabilmek için bir kısayoldur.

Şimdi kendimiz için neler yaptığımıza bakalım: “gallery.trigger”, tetikleyici(trigger) bağlantımızın jQuery seçicisi için bir referans. “gallery.content” ise resimlerimizin listesi. Şimdi bu kısa isimlerin faydasını görebiliriz(onları her kullanmak istediğimizde tüm seçici dizesini yazmaktan kurtulduk). Başka bir faydası; sadece bu

değerleri değiştirerek farklı sayfada ve farklı bir galerideki scriptleri kolayca gösterebiliriz.

Galeri nesnelerimize atadığımız diğer özelllik; fonksiyonlar, “gallery.offset” kayan listemizin ne kadar uzağa hareket edeceğini ayarlar, “gallery.slide” listeyi hareket ettirir veya hareket etmeyi sürdürmesini sağlar, “gallery.direction” slayt gösterisinin kayma yönünü belirler ve “gallery.init” başlangıç değerlerini veren fonksiyondur. Şimdi hepsine bakalım:

```
gallery.offset = function() {
  var left = gallery.content.position().left;
  if (gallery.scroll == '>') {
    if (left < 0) {
    left += gallery.width;
    }
  } else {
    if (left <= 0 && left >= ((gallery.innerWidth * -1) + (gallery.width * 2))) {
      left -= gallery.width;
    }
  }
  return left + "px";
}
```

“gallery.offset”de yapacağımız ilk iş, “div” elemanına bağlı olarak listemizin “left” özelliğini tutmak için bir değişken ayarlamak. “offset” yerine “position” kullanarak, görüntü kapısına bağlı olarak çalışan galeri sorunundan kurtulmuş oluyoruz.

Sonra, küçük resimleri kaydırmak istediğimiz yönü kontrol ediyoruz ve kaydırmak için hala bir odamız olduğunu görüyoruz. Eğer herşey düzgünse/normalse yeni bir “left” değeri üretiyoruz ve onu döndürüyoruz. Dönmeden önce değere “px” eklediğimizden emin oluyoruz çünkü onu bir CSS özelliği olarak kullanacağız. Yani, “gallery.offset” sadece galerinin ne kadar uzağa kayacağını hesaplıyor. Fonksiyonu inceleyiniz:

```
gallery.slide = function() {
  if (gallery.timer) {
    clearTimeout(gallery.timer);
  }
  if (gallery.scroll) {
    $(gallery.content)
    .stop(true,true)
    .animate({left: gallery.offset()}, 500);
    gallery.timer = setTimeout(gallery.slide, 1000);
```

```
  }
}
```

“slide” metodumuzun ilk işi “gallery.timer”ın ayarlanıp ayarlanmadığını kontrol etmek. Eğer ayarlanmışsa(set edilmişse), “setTimeout” zaten çağrılmıştır, yani “clearTimeout”u çağırabiliriz(sadece güvenli tarafta olabilmek için). Emin olmadığımız sürece kayma işleminin gerçekleşmesine izin vermemeliyiz. Sonra, “gallery.scroll”u kontrol ederiz(kayma işlemi olmalı mı diye). Eğer kullanıcı kayma bölgesinin dışında bir yere gelirse, kayma işlemini durdurmak isteriz. Eğer kullanıcı hala görsel bileşen üzerinde geziniyorsa, “gallery.content”in “left” özelliği üzerindeki jQuery'nin “animate” metodunu çağırırız. Bu şekilde galerimiz sorunsuz bir şekilde kayacaktır. Animasyonların kuyrukta yığılmasını önlemek için yine “stop(true,true)” kullanırız.

Hangi yöne kaydıracağımızı nasıl kontrol edeceğiz? “gallery.direction” bunu bizim için şu şekilde çözüyor:

```
gallery.direction = function(e,which) {
  var x = e.pageX - which.offset().left;
  gallery.scroll = (x >= gallery.width / 2) ? ">" : "<";
}
```

Başlangıç değerlerini atayan metodumuz “gallery.direction”ı çağırdığında, iki parametre yazar: “e” olayın yer aldığı sayfanın koordinatlarına ulaşmamızı sağlar. “e.pageX”i kullanarak, tetikleyici(trigger) bağlantının sol tarafından imlece kadar olan uzaklığı hesaplayabiliriz. Bu sayı , “gallery.scroll”a değer atamak için üçlü bir operatör kullanmaya yarar.Eğer imleç merkezin sağındaysa “>” değilse “<”.

Bakmamız gereken son metot, “gallery.init”:

```
gallery.init = function() {
  $(gallery.trigger)
  .mouseout(function() {gallery.scroll = false;})
  .mousemove(function(e) {gallery.direction(e,gallery.trigger);})
  .mouseover(function(e) {
  gallery.direction(e,gallery.trigger);
  gallery.slide();});
}
```

Şu an yaptığımız her şeye alışmış olmalısınız. “gallery.init” sadece “gallery.trigger” tarafından seçilen jQuery nesnesi üzerindeki jQuery metotlarını zincirler. Şimdi bu zincirlemeyi biraz parçalayalım.

“mouseout”, “gallery.scroll”u “false” olarak ayarlar. Bu özelliği, kullanıcının kaydırma işlemini devam ettirmemizi istediğinden emin olmak için kontrol ederiz. “mousemove” “gallery.direction”ı çağırır. Son olarak, “mouseover” da “gallery.direction”ı çağırır ama aynı zamanda “gallery.slide”ı da ekler.

Diğer bölüme geçmeden önce, geri çağrımız(callback)dan “mouseover” olayının sayfa üzerindeki pozisyonuna nasıl ulaştığımıza hızlıca bir göz atalım.

Olay İşleyicisi(Event Handler) Parametreleri

Bu zamana kadar gördüğümüz olay işleyicilerine baktığımızda, yukarıdaki “mouseover” işleyicisinde bir farklılık olduğunu görüyoruz: “e”yi tüm jQuery olay işleyicilerinin alabileceği opsiyonel bir parametre olarak yazdık. Bunu daha önce hiç kullanmamıştık çünkü uyguladığımız efektlerin hiçbirinde böyle birşeye ihtiyacımız yoktu.

Bir olay işleyicisi çağrıldığında, jQuery oluşan olayla ilgili veri ve detayları içeren bir olay nesnesi oluşturur. Verinin tipi, olayın tipine bağlıdır.Bir “keypress” olayı, hangi tuşa basıldığıyla ilgili bilgiler içerecekken, bir “click” olayı tıklamanın konumunun bilgisini içerecektir vb.

Koda dönelim. Kullanıcı ne zaman faresini navigasyon parçaları üzerinde hareket ettirse “mouseover” olayı uyarı verir. Olay işleyicimizde “e” parametresini yakalarız ve “gallery.direction” üzerine yazarz. Gördüğümüz gibi, bu olayla ilgili spesifik bilgiler içerir(nerede gerçekleştiği gibi). Bu bilgiyi, galeri görsel bileşeni üzerindeki farenin yatay pozisyonunu hesaplamak için kullanırız:

```
var x = e.pageX – which.offset().left;
```

Bu bize bir piksel değeri verir. Sonra, bu değeri galerinin yarısının genişliği ile karşılaştırırız. Eğer sayı daha küçükse, sol taraftayız demektir; değilse, sağ taraftayızdır. Bu, hangi yöne kaydıracağımıza karar vermemizi sağlar:

```
gallery.scroll = (x >= gallery.width / 2) ? ">" : "<";
```

“pageX” olay nesnelerinin kullanılabilir özelliklerinden sadece biri. Daha fazlasını bir sonraki bölümde göreceğiz.

BÖLÜM 5: MENÜLER, SEKMELER

Jquery DOM konusunda kesinlikle bir ustadır. Çok çaba sarfetmeden nesneleri haraket ettirme, CSS özelliklerini anime etme ve statik içeriğimizi daha güzel bir

hale getirmek için elamanlarımızın davranışlarını işletme konusunda yardımcı olur. Fakat bu statik içerik web'in çok küçük bir parçasıdır ; daha fonksiyonel uygulamalar günden güne artmaktadır. Bu bölüm statik içerikten ziyade Zengin İnternet Uygulamaları 'nın (RIA) büyülü dünyasına girmemize yardımcı olacaktır. Müşterimizin Zengin İnternet Uygulamaları (RIA) hakkında bilgi sahibi olduğunu varsayalım. Uygulamalarla dolu eğlenceli bir site isteğinde. Bu durumda bize de uygulamayı yazmak düşer.

Menüler

Aslında şu ana kadar birkaç menü tasarladık fakat bunlar çok basit menülerdi. Bu bölümde ise daha karmaşık yapılar üzerinde duracağız.(Açılıp kapanabilen menüler gibi)

Sitemiz büyüdükçe karmaşıklaşacak ve kullanıcı aradığını bulmakta zorlanacaktır.İyi kategorize edilmiş bir menü sistemiyle bu karmaşanın önüne geçilebilir.

Genişletilebilir/Katlanabilir Menüler

En yaygın kullanılan sistem başlıkların benzerliklerine göre gruplaştırıldığı alt menü sistemidir. Bu menü sistemi hem güzel görünür hem de aradığınızı kısa zamanda bulmanızı sağlar. Ayrıca ilişkili konular aynı başlıklarda toplandığı için daha az yer kaplar.İlk olarak küçük bir menü tasarlayıp üzerine özellikler ekleyelim.

Sitemize uyarlayacağımız genişletilebilir menü için iç içe geçmiş listeler ayarlayalım :

```
<ul id="menu">
  <li><a href="#">What's new?</a>
    <ul class="active">
      <li><a href="#">Weekly specials</a></li>
      <li><a href="#">Last night's pics!</a></li>
      <li><a href="#">Users' comments</a></li>
    </ul>
  </li>
<li><a href="#">Member extras</a>
  <ul>
```

```
    <li><a href="#">Premium Celebrities</a></li>
    <li><a href="#">24-hour Surveillance</a></li>
```

Kodun görünüşünü iyileştirelim :

```
#menu, #menu ul {
  list-style-type: none;
  padding: 0;
  margin: 0;
}
#menu li {
    cursor: pointer;
   background: #94C5EB;
   border-bottom: 1px solid #444;
}
#menu li a { text-decoration: none; }
#menu > li > a {
   padding: 2px 10px;
    font-weight: bold;
}
#menu li li {
    cursor: auto;
   border: 0;
   padding: 0 14px;
   background-color: #fff;
}
```

En üst seviyedeki linklere iç içe olanlardan farklı bir stil verebilmek için CSS çocuk seçicileri kullanıyoruz. Çoğu modern sunucu bunu destekliyor ancak Internet Explorer 6 kullanıyorsanız sıkıntıya düşebilirsiniz. Üst seviyedeki linklerin tıklanabilir olduğunu anlmamız için işaretleyici imleçler kullanabiliriz.

Tüm bunlar sağlandığında üst seviye bir menü tasarımı elde ederiz. Bütün elemanlar JavaScript desteği olmayan sunucularda bile görünür. Şimdi aşama aşama bu özelliği geliştirelim. İlk olarak kategori elemanlarının hepsini gizleyelim:

```
$('#menu > li > ul')
  .hide()
  .click(function(e) {
  e.stopPropagation();
});
```

Kazara alt kategorideki elamanların gizlenmesini önlemek için çocuk seçicileri

kullanıyoruz. Eğer menünüzü bir seviye daha açmak isterseniz kodumuz aynı şekilde çalışacaktır.

Kodumuzdaki e.stopPropagation() komutunu anlamamış olabilirsiniz. Bu komut kısaca açıklanacaktır ancak öncelikle efektimizi toggle fonksiyonunu kullanarak bitirelim.

```
$('#menu > li').toggle(function() {
    $(this).find('ul').slideDown();
        }, function() {
    $(this).find('ul').slideUp();
});
```

Final kodu sunucunuzda çalıştırdığınızda çok işlevli bir menü tasarlamış olduğunuzu göreceksiniz. Daha farklı özelliklerle geliştirmeye devam etmeden önce stopPropagation fonksiyonunu inceleyelim.

Olay Yayılımı

Olay yayılımı, bir olayın DOM hiyerarşisi boyunca akışını tanımlar. Bir eleman üzerinde bir olay gerçekleştiğinde, o eleman üzerindeki herhangi bir işleyici için o olayı yakalama şansı verilir. Bu aşama gerçekleştikten sonra, olay DOM ağacına geçirilir ve ebeveyn elemanlara o olayı yürütmek için bir şans verilir. Bu olay anlamlı gelir: paragraf içindeki bir linke tıkladığınızda aynı zamanda paragrafın kendisine de tıklamış olursunuz, yani her elemandaki olay işleyicisinin harekete geçmesi için bir şansı olmalı.

Olay yayılımını anlayabilmenin en kolay yolu uygulama üzerinde görmek.Bu konsepti göstermek için hızlı bir deney yapalım. Deneyimiz şu başlıklardan oluşsun : iki adet div ,biri iç biri dış olsun. Dıştaki div ve içteki div “id” lere sahip olsun.

```
<div id="outer">
  Click Outer!
   <div id="inner">
     Click Inner!
   </div>
</div>
```

Şimdi de , div elemanlarımıza tıklanabilme özelliği ekleyelim. Böylece bir elemanın üzerine tıkladığımızda uyarı açılır ve bize elemanın adını söyler.

```
$('div').click(function() {
lert('Hello from ' + $(this).attr('id'));
```

```
});
```

İlk olarak dıştaki div'e tıkladığınız zaman karşınıza “Hello from outer” yani “Dışardakinden selam” yazısı çıkacaktır. İçteki div'e tıkladığınız zaman “Hello from inner ” yazısının çıkmasınız beklersiniz ancak “Hello from outer” yazısıda belirir. Bir kere tıklamamıza rağmen neden iki tıklama olayı görürüz?

Tahmin ettiğiniz üzere iki tıklama olayı yerine tek tıklama iki farklı yerde gerçekleşir. Olay , içteki div'imizden başlar ve buna bağlı bir olay olup olmadığını kontrol eder. Böylece tıklama olayı div'in ebeveyni olan elemana sıçrar(Bu durumda ebevyn dıştaki div'dir). Olay bu şekilde ebeveyn eleman kalmayıncaya kadar sıçrayarak devam eder .

Olay sıçraması çoğu durumda yararlıdır. Aynı olayı DOM'un farklı seviyelerinde işlemek isteyebiliriz. Herhangi bir çocuğa tıklandığında ebeveynin sınıfını ayarlamak istersek herbir çocuğa ayrı ayrı işlemek yerine ebeveyne işlemek daha verimli olacaktır. Ama biz hala her bir çocuğa ayrı ayrı tıklama işleyicisi eklemek istiyoruz.

Diğer taraftan , sıçrama olayı bazı durumlarda istenmeyen bir nitelik olabilir. Bazen çocuğu başka bir olaya atlamadan sonlandırmak isteyebiliriz. Örnek olarak tavşanların sırayla kafalarını deliklerden çıkardığı , sizinde vurmaya çalıştığınız Whac-A-Mole oyunu yaptığımızı hayal edin. Bu oyun iki parçadan oluşur : oyun tahtası ve delikler. Delikler ekranda sadece bir kaç saniyeliğine belirir ve kaybolurlar. Deliklere yapılan direk vuruşları incelemek için her bir deliğe işleyici eklememiz gerekebilir ve kaçan vuruşları incelemek için de oyun tahtasına başka bir işleyici eklememiz gerekebilir. Olay yayılımı ile yapılan ve kaybalon vuruşlar hesaplanır ve oyun tahtasına kaydedilir.

Olay yayılımını kontrol etmek için bazı teknikler vardır. Genel olarak kullanılan JavaScript tekniği ise olay işleyicisinden “false(yanlış) döndürür. Bu durum çoğu sunucu tarafından desteklenen ve oldukça iyi bir tekniktir. Ancak , Jquery'nin olay sistemi bütün olayları W3C standartlarına uyarlar. Bu da farklı sunuclarda beklenmedik durumları önler.

Jquery kullanırak olay yayılımını durdurmak için stopPropagation metodunu kullanırız. Olayı tutabilmek için anonim geri çağırım fonksiyonuna “e” parametresini geçiririz. Daha sonra , bu olaydaki stopPropagation fonksiyonunu çağırırız ve bu da DOM 'un daha yukarıya çıkmasını engeller.

Varsayılan Olay Faaliyetleri

Olay akışını kontrol etmek için yaygın olarak kullanılan diğer bir metod da bakma

zamanı : preventDefault . PreventDefault komutu,sunucunun bir olayın varsayılan faaliyetini yürütmesini durdurmasında kullanılır . En genel kullanımı bir linke tıklandığında hedefi yüklemeyi durdurmasıdır.

```
$('a').click(function(e) {
   e.preventDefault();
});
```

Bu kod etkili bir şekilde sayfadaki linkleri devre dışı bırakır. Tabiki de bunu her link için istemek saçmadır fakat bir linkin faaliyetini bu şekilde düzenlenmesi yaygındır.JavaScript desteği olmadığında link normal bir şekilde görünürken JavaScript aktif olduğunda jQuery fonksiyonelliği ortaya çıkar.

Eskilerden kalan bir JavaScript tekniği ise varsayılan faaliyetleri önlemek için olay işleyiciden “false ” döndürmektir. Jquery'de bu metodu kullanmak ile preventDefault ve stopPropagation fonksiyonlarını çağırmak aynı etkiyi gösterir.

Ayrıca olay akışının değişip değişmediğini test etmek için isDefaultPrevented ve isPropagationStopped komutlarını kullanabilirsiniz. İsimlerinden de anlayacağımız üzere bu fonksiyonlar , varsayılan faaliyet durdurulduysa “true” aksi takdirde “false” döndürür.

Açık/Kapalı Göstericiler

Menü kontrolümüz planlandığı gibi çalışmaktadır yani kaçınılmaz jQuery kurallarına uymanın zamanı gelmiştir. İlk olarak resimdeki simgeleri eklemekle işe başlayalım.

Bu şekildeki bir göstericiyi mutluka internet sitelerinde görmüşsünüzdür. Göstericiler menünün açılacağı ya da kapanacağı yöne göre değişiklik gösterir(Şekilde görüldüğü üzere).Böylece kullanıcıya yol göstermiş oluruz.

Menümüze göstericileri eklemek için CSS sprite tekniğini(resim yer değiştirme) kullanacağız. Menü bölümümüzdeki oklar aşağı yönlü ve yukarı yönlü olabilirler.

Başlangıçta , bütün menü bölümlerimiz kapalı olduğu için oklarımızın yönü aşağı bakacaktır. Arkaplan resmimizi menü içindeki “li” elemanımıza uygulayalım ve daha derindeki iç içe geçmiş öğelerden silelim.

```
#menu li {
cursor:pointer;
border-bottom:1px solid #444;
background: #94C5EB url(arrows.png) no-repeat right top;
}
⋮
#menu li li {
cursor:auto;
border:0;
padding:0 14px;
background-color:#fff;
background-image: none;
}
```

Yerleştirdiğimiz arkaplandaki resmimizle ne zaman bir menü öğesi değiştirilse CSS sprite'ımızın pozisyonunu ona uydurmalıyız. Menü öğesi aşağıya kaydığında genişletilmiş durumu , tekrar yukarı kaydığında sınırlandırılmış durumu gösterir. “ul” elemanını göstermek veya gizlemek için alt bölümlere geçmeden önce css faaliyetini uygulamak adına akıllıca bir zincirleme sistemi kullanacağız.

```
$('#menu > li').toggle(function() {
  $(this)
    .css('background-position', 'right -20px')
    .find('ul').slideDown();
  }, function() {
   $(this)
     .css('background-position', 'right top')
     .find('ul').slideUp();
});
```

“Hover” Durumunda Menü Genişletme

Bir sonraki hilemiz için , hover olayına ve tıklama olayına cevap veren bir menü yapacağız. Kullanıcı ebeveyn menü ögelerinden birinin üzerine geldiğinde kısa bir süre durduracağız ve menüyü genişleteceğiz. Bu aşama boyunca yeterli olduğunu gördünüz. Jquery “hover” etkisisiz fareyi hedef bölgeye haraket ettirdiğiniz an başlayacaktır.Ancak bu etki gerçekleştirme zamanını geçiktireceğinden dolayı kısa olmalıdır. Diğer türlü de fareyi hızlı haraket ettirirken pencereler çok hızlı açılıp

kapanıcak ve kontrol edilmesi zor bir menü oluşacaktır.

Bu ufak ancak önemli bir değişikliktir. Eğer menüde gösterildiği gibi değişiklikler yaparsanız , kontrolün önemli bir şekilde değiştiğini göreceksiniz :

```
$('#menu > li').hover(function() {
  $(this).addClass('waiting');
  setTimeout(function() {
     $('#menu .waiting')
       .click()
       .removeClass('waiting');
  }, 600);
}, function() {
   $('#menu .waiting').removeClass('waiting');
});
```

Kullanıcı ilk olarak fareyi menü elemanının üzerine getirir ve menüye “waiting” isimli bir sınıfı ekleriz böylece zamanlayıcıyı 600 milisaniyeye ayarlamış oluruz. Eğer kullanıcı fareyi gecikme zamanı ayarlanmadan menü elemanından uzağa hareket ettirirse sınıfı çıkarırız.

Gecikme aranırken , “waiting” sınıfını da barındıran menüye bakılır(kullanıcı fareyi menü elemanından uzağa hareket ettirmediyse). Kullanıcı hala bekliyor ise menü simgesine tıklar ve etkinin başlamasına sebep olur. Bu tıklama olayını bu şekilde ilk görüşümüz. Parametresiz çağırıldığında olay işleyicisini kurmak yerine tıklama olayı hedef elemandaki etkileri başlatır. Son olarak, etki başladığında sınıfı çıkarıyoruz böylece kare olana geri dönüyoruz. Ayrıca tıklama olayına daha önceden tanımladığımızı değiştirmemiz gerekiyor. Eğer bir kullanıcı faresini menü üzerine getirir ve tıklarsa, “waiting” sınıfı zamanlayıcı bittiğinde menünün kapanmasına neden olacak. Tıklama işleyicisine “removeClass” çağrısını eklemeliyiz.

Yaptığımız şey daha önceki etkilerde de yapıldığı gibi, sınıflar kullanarak durum yönetimini sağlamaktır. Durum yönetimi , hangi durumda(state) olduğumuzu hatırlama ile kontrol etmemizi sağlamanın eğlenceli bir yoludur. Gecikmeyi incelemeden kullanıcının menüden uzaklaşıp uzaklaşmadığını bilmek isteriz. Böylece “waiting” sınıfını doğru bir şekilde ekleyebilir ve çıkarabiliriz. Durum yönetimi için sınıf isimlerini kullanmak , şık bir hiledir. Fakat tek veya en iyi yol değildir.JQuery tarafından sağlanan basit durum bilgilerinin veri fonksiyonelliğini daha önceden görmüştük.

Açılabilir Menü

Eğer eskiden kullanılan internet dili (DHTML) üzerinde açılabilir menü kodlamaya

çalıştıysanız ne kadar sinir bozucu bir deneyim olduğunu biliyorsunuzdur. Online olarak yavaş ilerliyen bir çok script vardı ama CSS sayesinde bu sorunlar çözüldü. “Suckerfish drop-down”(http://www.alistapart.com/articles/dropdowns) yani hem aşağıya hemde yana açılabilen menüler , açılabilir menülerin dikkatli bir şekilde biçimlendirilmiş halidir. Bu menüler çocuk menü simgelerini de saklarlar. JavaScript kullanabilen mükemmel bir çözümdür. JavaScript kullanışlı olmasına rağmen uygun olduğunda CSS tekniklerinide kullanmalıyız.

CSS açılabilir menüleri kullanırken bazı sorunlar yaşanır. Eski sunucular bağlantısız elemanlar üzerindeki “hover” seçicilerini stilize etmek için yeterli değildir. Bu yapılsa bile gösterme/gizleme efektleri aniden olabilir. Suckerfish açılabilir menüler geliştirebilirlik için iyi bir taban oluştururlar. Jquery ile geliştirilebilen uygun çözümler sağlarlar. “hover” desteklemeyen sunucularda çalışmak için “Suckerfish” tekniğini öğreneceğiz. Aynı zamanda aşağıya doğru açılabilen menüleri Jquery animasyonları ile geliştireceğiz.

Çapraz Tarayıcı Suckerfish Menüleri

İlk olarak taban çizgimiz olarak basit bir Suckerfish açılabilir menü kurgulayalım. Açılabilir / kapanabilir menüler bölümündeki genişletilebilir navigasyon için kullandığımız biçimlendirmenin aynısını kullanacağız. CSS sırasız listelerini yatayda konumlandırılmış menü olarak biçimlendirecek. Dikkat etmemiz gereken tek yer yeni eklediğimiz “:hover” CSS sınıfıdır. Açılabilir menümüzün görünürlüğü için bu sınıfa ihtiyacımız var.

```
#container {
position: relative;
}
#menu {
position: absolute;
top: 0;
right: 0;
}
#menu, #menu ul {
padding: 0;
margin: 0;
list-style: none;
}
#menu li {
float: left;
background: #FFF;
}
#menu a {
```

```
display: block;
padding: 4px;
width: 10em;
}
#menu li ul {
position: absolute;
width: 10em;
left: -999em;
}
#menu li:hover ul, #menu li ul:hover {
left:auto;
}
```

Bunu sunucuların çoğunda denerseniz sürpriz bir şekide JavaScript olmadan tam olarak çalışan bir menünüz olduğunu göreceksiniz. Şimdi yapmak istediğimiz şey efektin üst kısımlarına jQuery ekleyelim ki eski sunucularda daha tatmin edici sonuçlar alalım. Aşağıdaki şekilde gösterilmiştir:

Bu efekti jQuery kullanarak nasıl yapabiliriz? Basit , sadece hover faaliyetini aşağıya doğru kayma efektiyle ekleyeceğiz diyebilirsiniz. Haklısınız ancak bir problem var. Hem CSS'imize hem de scriptimize aynı fonksiyonelliği uygularsak ne olur? Kim kazanır dersiniz? Aslında bu bakımdan eşleşirler ve kazanan yoktur.
Geçici bir çözüm olarak hover olay işleyicimizdeki CSS sınıfımızı yeniden yazabiliriz. Ancak bu bir sonuç vermeyecektir , çünkü CSS özelliklerini ayarlamak için jQuery'deki taslak seçicileri hedef gösteremezsiniz.

```
$('#nav li:hover ul').css('left', '-999px'); // doesn't work!
```

Peki Suckerfish CSS menülerimizin güzel görünümünü jQuery iyileştirmelerini uygulayarak nasıl sürdürebiliriz? CSS özelliklerini kaldırmak çözümlerden biri olabilir ama hover olayının olup olmadığına bakmadan çocuk menünün CSS özelliklerini devralmak daha basit bir yoldur.

```
$('#menu li ul').css({
display: "none",
```

```
left: "auto"
});
$('#menu li').hover(function() {
$(this)
.find('ul')
.stop(true, true)
.slideDown('fast');
}, function() {
$(this)
.find('ul')
.stop(true,true)
.fadeOut('fast');
});
```

Scrip'tin ilk kısmı daha önceden ayarladığımız CSS 'i yeniden yazar ve gösterir “none” kullanarak alt menüleri gizler. Kullanıcı fareyle liste simgesinin üzerine geldiğinde bağlantılı çocuk listenin container'ını bulup gösteririrz. Burada slideDown ve fadeOut kullandık ancak başka “easing” seçenekleriyle karşılaşabilirsiniz. stop(true ,true) komutu , 3. bölümde gördüğümüz gibi fareyi menüde çok hızlı haraket ettirdiğimizde animasyonların kuyrukta yığılmasını önler.

Menü kontrolünün en iyi yönü JavaScript desteği olmayan bir kullanıcı siteyi ziyaret ettiğinde bizim menümüz hala iyi bir şekilde çalışıyor olacaktır.

Hover Intent Fonksiyonu

“moueseover” olayı ile tetiklenen bazı efektler gördük ve onların bazen aktif olmak için çok istekli olduğunu farketmiş olabilirsiniz. Farenizi hedefin üzerinden kısa süreliğine geçirdiğinizde bile aktif olurlar. Akıcı anime edilmiş efektleri yapmanın en iyi yoludur ama sonradan farklı sonuçlarla karşılaşabiriz. Açılabilir menülerimizle, yanlış başlangıç gereksiz ve rahatsız edici gözükebilir. Kullanıcının hedef elemanı aktif etmek isteyeceğinden emin olana kadar efekti geciktirecek bir metoda ihtiyacımız var.

Önceden, kullanıcı fareyi elemanın üzerine getirdiğinde bu isteğimizi gerçekleştirebilmek için bir zamanlayıcı kullanmıştık. Ancak bunun bazı yan etkileri var. Örneğin, küçük bir zamanlayıcı ve büyük bir hedef alanınız varsa, zamanlayıcının süresi dolmadan önce elemana ulaşamayabilir.

Brian Cherne bu sorun için, Hover Intent eklentisinin(http://cherne.net/brian/resources/jquery.hoverIntent.html) yapısında iyi bir geçici çözüm uyguladı. Fare elemanların üzerlerinden geçerken farenin hızını hesapladı.Efekt ancak eklenti kullanıcının orda durmak istediğini anlayacak kadar

farenin hareketinin yavaşladığını fark ederse faaliyete geçer.

Eklentiyi dahil ettiğinizde , hover faaliyetini kullandığınız herhangi bir yerde kullanabilirsiniz. Örnek olarak açılabilir menümüze bir gecikme eklemek istersek hover komutu yerine hoverIntent komutunu koymamız yeterli olacaktır.

```
$('#menu li').hoverIntent(function() {
⋮
}, function() {
⋮
});
```

Menü sadece kullanıcı onu görüntülemek istediğinde görünecektir. Bu eklentinin ne yaptığını anlamak için en iyi yol önceki örneğimizle karşılaştırmak olacaktır. Önceki örneğimizde menüler üzerlerine gelir gelmez açılıyordu ama hoverIntent kullandığımızda fare menü simgesini üzerinde durduğu zaman açılır hale geldi.

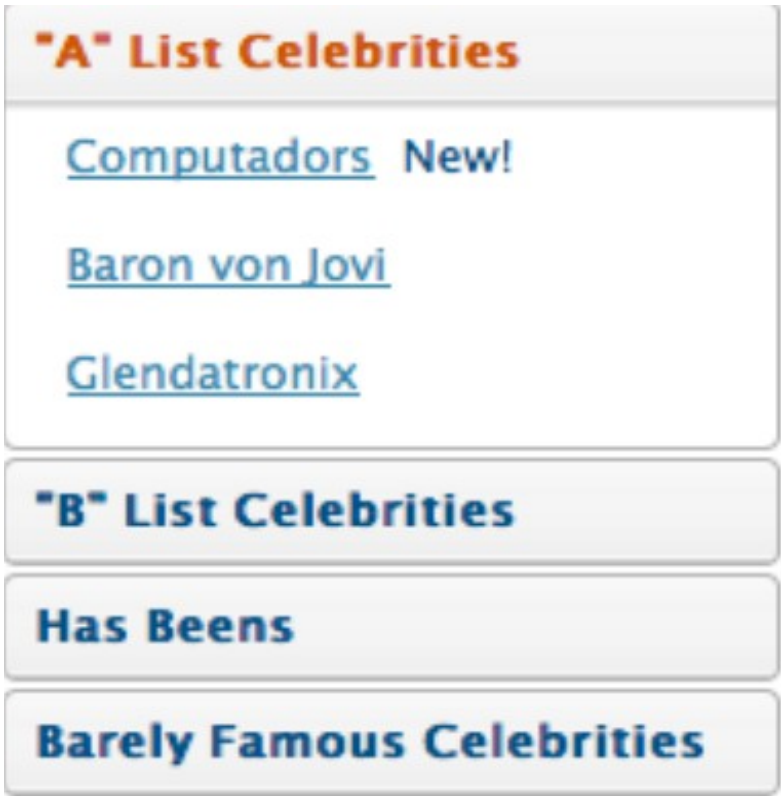
Akordeon Menüler

Akordeon menüler adını müzik aleti alan akordeona benzetildiği için almıştır. Bazı akordeonlar bütün simgeleri gizleyebilmeniz için o anda açık olan alanı kapatabilmenize olanak sağlarken tipik olarak tüm akordiyonlar herhangi bir zamanda sadece bir alanın görünebileceği şekilde ayarlanır.

Basit Akordeonlar

Akordeonlar beklediğinizden daha yanıltıcı olabilirler. Daha önceden açılabilen ve kapanabilen menülerin jQuery 'nin bir kaç satırını kodumuzda işleyerek nasıl basitleşebileceklerini gördük. Şimdi menü elemanlarını kısıtlarsak (sadece bir tanesini kullanarak) daha mantıklı olur.

Örneğin , ünlüleri popülerliklerine göre gruplayacak basit bir akordeon inşa edelim. Bu bize sitemizin önceki ekranını kaydetmemizi sağlar. Sonucu aşağıdaki şekilden görebilirsiniz.



Sanal olarak HTML biçimlendirmesinin herhangi bir yapısını kullanarak bir akordeon oluşturabilirsiniz. Bütün ihtiyacımız olan net olarak tanımlanabilecek bir “header” kümesi(her biri bir içerik ile alakalı olacak şekilde). Bu örnek için iç içe sıralı listelerin bir kümesini şu şekilde kullanacağız:

```
<ul>
  <li class="active">
    "A" List Celebrities
    <ul>
      <li><a href="#">Computadors</a>  New!</li>
      <li><a href="#">Johny Stardust</a></li>
      <li><a href="#">Beau Dandy</a></li>
    </ul>
  </li>
<li>
  "B" List Celebrities
  <ul>
    <li><a href="#">Sinusoidal Tendancies</a></li>
    <li><a href="#">Steve Extreme</a></li>
  </ul>
</li>
:</ul>
```

Liste biçimlendirmesi bir menü için ideal bir HTML yapısıdır ama başka seçenekler de vardır; en basit şekilde iç içe sıralı div elemanlarını kullanabilirsiniz(her bir bölümün başlığını header elemanlarıyla ekleyebilirsiniz). Header tetikleyicilerinin tümünü ve ilgili içeriği seçmenizi sağlayan herhangi bir yapı tutarlı olduğu sürece uygun olabilir. Listeyi CSS 'in bazı özellikleriyle stilize ettik. Örnek kodlarımızı inceleyebilirsiniz.

Sayfamız yüklendiğinde içerik alanlarımızın hepsi görünür. Şimdide varsayılan simgemiz dışında bütün içeriği gizlemeliyiz.

```
$('#celebs ul > li ul')
  .click(function(e) {
    e.stopPropagation();
})
  .filter(':not(:first)')
  .hide();
```

Bunu daha öncekinden biraz farklı olarak yaptık. Bu durumda daha önceden olay sıçramasında karşılaştırdığımız problemi etkisizleştirdik. Bu komut çeşitliliği de bize jQuery'nin gücünü gösterir. Her bir içerik alanına olay dinleyicisi ekleyerek başlıyoruz. Seçimimizi ilk alanı çıkaracak şekilde daraltıyoruz ve daha sonra geriye kalan her şeyi gizliyoruz.

“filter” komutu bir seçimi daraltabilmenin en kullanışlı yoludur. Kriterlerimize uymayan elemanlar seçimimizden çıkarılır ve bunun sonucunda diğer jQuery komutlarından da etkilenmezler. Kriteri bir seçici veya fonksiyon olarak belirleyebilirsiniz. Önceki örnekte “filter “ seçicilerini kullandık(:not ve :first).Sonradan elelmanları bulabilmenizde yardımcı olacak herhangi bir jQuery seçicisini kullanabilirsiniz.

“:not” düzenli ve işe yarar bir seçicidir çünkü parantez içindekilerin zıddını seçer. Yani
$(':not(p)') komutu paragraf olmayan bütün elemanları seçer . Aynı şekilde $('p:not(.active)') komutuda aktif olmayan paragrafları seçer.

“filter” metodunun tersi “add” metodudur. Filter komudu seçicilerden elemanları çıkarken add komutu seçicilere yeni elemanlar eklerken kullanılır. Filter ve add metodunu birleştirerek çok sayıda uygulama yapabilrisiniz.

Akordeon efekti için yazdığımız kod açık olan bütün simgeleri kapatmalı sonra tıkladığımızı açmalıdır.Ancak burada bir mantık hatası var. Eğer zaten açık olan bir simgeye tıklarsak gereksiz yere aşağı yukarı kayacaktır. Bu yüzden ilk olarak simgenin açık olup olmadığı kontrol edilir.

```
$('#celebs ul > li').click(function() {
  var selfClick = $(this).find('ul:first').is(':visible'); 1
  if (!selfClick) { 2
    $(this)
      .parent()
      .find('> li ul:visible')
      .slideToggle();
  }
  $(this)
    .find('ul:first')
    .slideToggle(); 3
});
```

Kodu parçalara ayıralım:

1) .is('visible') yapısını kullanarak iç içe geçmiş ul elemanlarının görünür olup olmadığını kontrol ederiz ve sonucunu selfClick isimli değişkende tutarız.

2) Eğer görünür bölümler tıklanılmadıysa gizlemek için if komutunda "!" operatörünü kullanırız. Böylece iç içe kod blokları selfClick "true" değilse çalıştırılacaktır.

3) Son olarak , tıkladığımız simgenin durumunu değiştiriyoruz . Eğer açıksa yukarı değilse aşagıya kayıyor.

Çözümümüzde kodladığımız yolla kullanıcı akordeon menümüzdeki açık olan bölümleri kapatabilir.Bir simgenin sürekli görünür kalmasını isterseniz , kodunuzu buna göre düzeltebilirsiniz böylece açık simgeler de etklilenmez. Bunu basit ve temel JavaScript ile yapabilirsiniz. Eğer selfClick "true" ise JavaScript 'te "return" komutunu kullanarak fonksiyondan çıkabiliriz.

```
$('#celebs ul > li').click(function() {
  var selfClick = $(this).find('ul:first').is(':visible');
  if (selfClick) {
     return;
}
$(this)
  .parent()
  .find('> li ul:visible')
  .slideToggle();
```

```
$(this)
  .find('ul:first')
  .stop(true, true)
  .slideToggle();
});
```

Çok Katmanlı Akerdeonlar

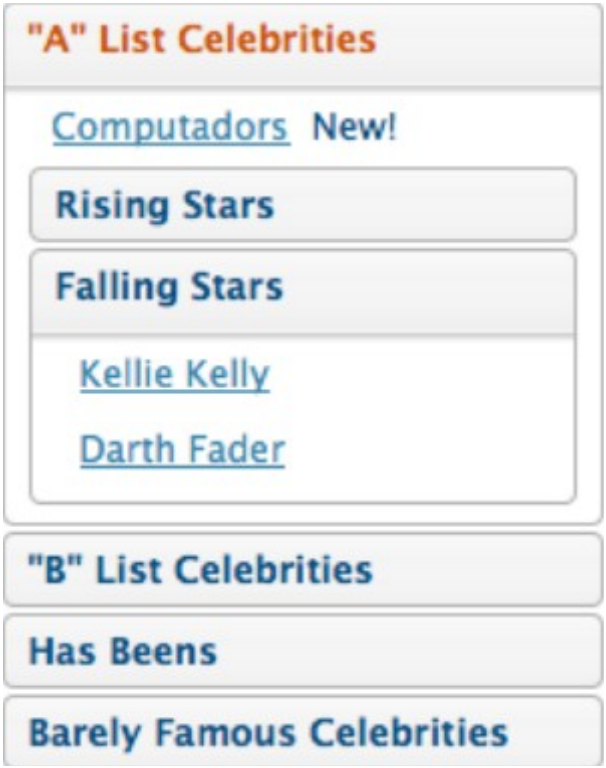
İlk olarak , akerdeonunuzun HTML yapısını sürekli bir şekilde kurmalısınız demiştik şimdi nedenini göreceğiz. Eğer jQuery seçicilerimiz yeterince belli olduysa akerdeonumuza başka bir katman eklemek olay işleyicimize yeni bir seviye eklemek kadar basittir. İlk olarak , menü simgelerinin ikinci katmanını ekleriz. Birinci katmanda kullandığımız yapının aynısını kullanırız ama bunu ilk katmandaki liste simgesinin içine koyarız.

```
<ul>
  <li class="active">"A" List Celebrities
    <ul>
       <li><a href="#">Computadors</a>  New!</li>
      <li>Rising Stars
        <ul>
  <li><a href="#">Johny Stardust</a></li>
  <li><a href="#">Beau Dandy</a></li>
   </ul>
 </li>
 <li>Falling Stars
   <ul>
     <li><a href="#">Kellie Kelly</a></li>
     <li><a href="#">Darth Fader</a></li>
⋮
```

Tek seviyeli akordeonumuz için, $('#accordion > li').click(…) kodunu kullanarak listemizin kökündeki(root) ilk seviye çocukların hepsine akordeon kodumuzu ekliyoruz. İçi çe geçmiş listemizin yapısı öncekiyle aynı olduğu için, tek yapmamız gereken aynı kodu içi çe geçmiş elemanlara uygulamak:

```
$('#accordion > li, #accordion > li > ul > li').click(…);
```

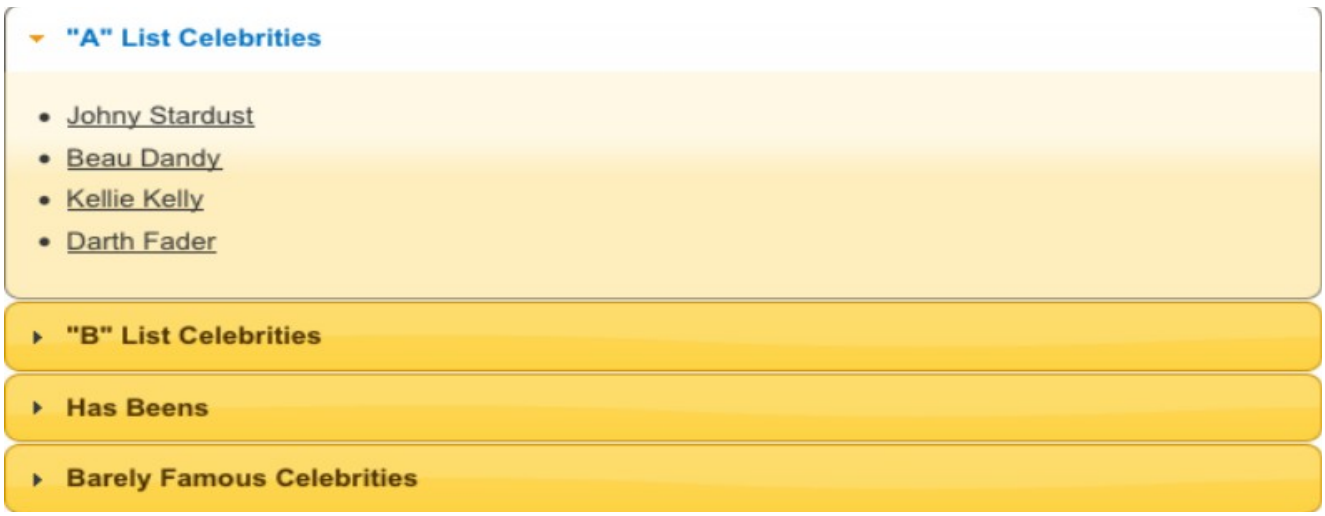
Eğer bu seçici zincirini takip ederseniz, akordeon kodunu doğru liste simgelerine eklediğimizi göreceksiniz. Şu an sadece her liste elemanına onu ekledik, iç içe geçmiş içeriklerde farklı davranışlara neden olabilir(HTML yapınıza bağlı olarak). Sonuçta oluşan menüyü aşağıdaki şekilde görebilirsiniz:



Eğer akordeona daha fazla katman eklemek isterseniz, sadece bu işlemi tekrar etmeniz yeterli. Eğer uzun seçiciler kontrolden çıkarsa, her seviyenin/katmanın köküne ekstra bir sınıf ekleyebilirsiniz.

Jquery UI(Kullanıcı Arayüzü) Akordeon

Gördüğünüz gibi, jQuery kullanarak tamamen kontrollü bir akordeon yaratmak basit. Ancak, jQuery UI kütüphanesi de bir akordeon kontrolü içermektedir ve çok geniş bir seçenek aralığı sunar. Bunlara örnek olarak şunları gösterebiliriz; değişen simgeler, “mouseover” olayını tetikleme ve belli yollarla akordeonun içeriğindeki elemanları faaliyete geçirmek. Jquery UI akordeonunun bir özelliği, “Sunny” temasını kullanmak. Şekli inceleyiniz:



Özel akordeonlarımız gibi, jQuery UI akordeonlarının biçimlendirmesi de “header”ları ve içerik elemanlarını gerektirir. Varsayılan olarak, “header”ların link etiketleri olduğu ve içeriğin de onları takip ettiği kabul edilir. Ancak, içerikte linkler varsa biraz karmaşıklaşabilir. Bu yüzden, içeriğinizin hangi bölümünün başlık olacağına karar vermesine yardım edecek bir seçici belirlemek en iyisidir:

```
$('#accordion').accordion({header: 'h3'});
```

Bu, içeriğinizi tamamen fonksiyonel bir akordeona dönüştürebilmeniz için gerekli olan tüm kod(h3'ün bitişiğindeki kardeş sizin gizlemek veya göstermek istediğiniz içerik bölümü olduğu sürece kullanılır).

Akordeon aynı zamanda kontrolle programlı bir etkileşim için gerekli olan fonksiyonelliği sağlar. Örneğin, “activate” seçeneğini kullanarak özel bir içerik bölümü açabilirsiniz(açmak istediğiniz panelin indeksi boyunca). Üçüncü bölümü açmak için(indekslerin 0'dan başladığını unutmayın), aşağıdaki kodu yazmalısınız:

```
$("#accordion").accordion('activate', 2);
```

Akordeonla etkileşimi ve yapılandırmayı ayarlamak için daha bir çok kullanılabilir seçenek var.İncelemek için; http://jqueryui.com/demos/accordion/

BÖLÜM 6: YAPI (CONSTRUCTION), AJAX VE ETKİLEŞİM (INTERACTIVITY)

Şu ana kadar müşterimiz için bir çok görsel efekt hazırladık ve sitesine hoş bir görünüm kazandırdık. Ancak müşterimiz biraz kurnazlaşıyor, sayfalarının Web 2.0 gibi görünmesinin yanı sıra Web 2.0 gibi davranmasını da istiyor. Bunun anlamı: Ajax!

Kullanıcı tarafındaki Ajax fonksiyonelliklerini uygulamak kolay, özellikle de uygulama iskeletimiz(framework) jQuery ise. Ama bu yeni özellikler karmaşıklığı da biraz arttırıyor. Bu yüzden konunun derinlerine inmeden önce bu karmaşayı nasıl yönetebileceğimizin bir kaç yoluna bakalım ve etkileyici kodlar nasıl yazılır görelim.

Yapı ve En iyi Uygulamalar

jQuery, Ajax ve Dom ile ilgili aşamaları basite indirgedi ama iyi ve temiz bir JavaScript kodu yazmanın yararlarını değiştirmedi. JavaScript kodu yazmakta uzman olmamıza gerek yok fakat projelerimize daha okunabilir ve sürdürebilir kodlar kazandırabilmemiz için takip etmemiz gereken bir kaç adım var.

Temizleyici jQuery

Kod Yorumları

HTML ve CSS 'te olduğu gibi JavaScript kodunuzu yorumlamaya olanak sağlar. Yorum olduğunu belirtmek için satır başlarına // koymak yeterlidir.

```
// Assign the value '3' to the variable 'count':
var count = 3;
```

Yorumunuz birden fazla satırı kaplayacaksa şu şekilde yapabilirsiniz:

```
/* An example of
 a multiline
comment
*/
var count = 3;
```

Harita (Map) Objeleri

Tek bir jQuery faaliyetine birden fazla seçenek yazmak için kullanırız. Örneğin :

```
$('p').css({color:'green', padding:'3px'});
```

Özel bir Jquery yapısı değildir ama kendi fonksiyon ve görsel bileşenlerinize yazmak için bilgilerinizi kapsamakta idealdir. Örneğin bir veriyi çalışma alanından dışarı çekerseniz , onları anahtar/değer olarak haritalandırılmış çiftler olarak paketleyebilirsiniz. Böylece onlar üzerinde sonradan daha fazla işlem yapabilirsiniz.

```
var id = $('input#id').val();
var name = $('input#name').val();
var age = $('input#age').val();
var data = {
type: 'person',
id: id,
name: name,
age: age
}
```

Bütün sarmalanan verileri kolayca yazabiliriz ve istediğimiz gibi kullanabiliriz. Bir nesnenin herhangi bir değerine ulaşmak için basitçe nesnenin isminden sonra .type yazmalıyız.Örneğin data objesine ulaşmak ve person'u içerip içermediğini kontrol etmek için aşağıdaki gibi bir yol izleriz :

```
if (data.type == 'person') {
alert('Hello ' + data.name);
}
```

Kapsam

Kapsam kelimesi proglamlama dilinde değişkenlerin tanımlandığı alan anlamına gelir.
Örneğin , bir değişkeni herhangi bir yapının(fonksiyon veya nesne) veya döngünün dışında tanımladığınızda evrensel(global) olarak tanımlamış olursunuz. Benzer şekilde , yapının içinde tanımladığınızda yerel (local) değişken tanımlamış olursunuz.

Bu olay basit görünebilir ancak Ajax istekleri için geri çağırım metotlarını tanımlayarak başladığımızda daha karmaşık olabilir. Çünkü geri çağırım tanımlandığından daha farklı bir içerikte yürütülecektir. Geri çağırımda görsel bileşenlerinizi gruplandırdığınızı göstermek için "this" komutunu kullanmayı tercih ederseniz beklenemdik bir süprizle karşılaşabilirsiniz "undefined" olabilir yada tamamen başka bir şey anlamına geliyor olabilir.Örneğin :

```
var WIDGET = {};
WIDGET.delay = 1000;
WIDGET.run = function() {
alert(this.delay); // 1000 …good!
$(p).click(function() {
alert(this.delay) // undefined! bad!
});
};
```

Bir p etiketine tıklanıldığında olay işleyicimiz görsel bileşen nesnesinin kendisi yerine daha farklı bir içerikte çalışır.Bu yüzden this.delay büyük ihtimalle var olmayacaktır. Bununla başa çıkabilmemiz için bir kaç yol vardır ancak çok JavaScript kullanmadan en basit yol görsel bileşenin kapsamını bir değişken içinde tutmaktır:

```
var WIDGET = {};
WIDGET.delay = 1000;
WIDGET.run = function() {
  alert(this.delay); // 1000 …good!
  var _widget = this;
  $(p).click(function() {
    alert(_widget.delay) // 1000 …yes!
});
};
```

İstemci Tarafı Şablonu

Şimdiye kadar gördüğümüz menülerin çoğu statik içeriğe sahip menülerdi.Çoğu menü yazıları değişmezler ama Ajax uyumlu görsellerde keşfettiğimiz gibi yazıları dinamik olarak tekrar yerleştirmek ve dahil etmek büyük bir problem teşkil etmektedir. Buda bize bir problem oluşturur.Sayfanıza eklediğiniz içerikleri yapılandırmak için nereleri biçimlendirmelisiniz?

Bu probleme farklı şekillerde yaklaşabilirsiniz. En basiti tüm içerik levhasını her göründüğünde yeniden yerleştirmek(html aracılığıyla). Ne zaman içerik levhasını güncellemek istersek tüm içerikleri yeni biçimlendirmeyle yeniden yerleştireceğiz. Örneğin:

```
$('#overlay').html("<p>You have " + cart.items.length +" items in your cart.</p>");
```

Eğer küçük sayılarda biçimlendirmemiz varsa HTML içeriklerini direkt olarak yazmak iyi olacaktır. Ancak daha ayrıntılı içerikler için okunması ve sürdürebilmesi zor olan JavaScript veya jQuery kodları çabuk bir şekilde düzenlenebilir. Kod birleştirme aracılığıyla daha karmaşık tablolar yapmaya çalışırsanız sorunla karşılaşabilirsiniz.

Bu problemi HTML içeriğinize kancalar ekleyerek engelleyebilirsiniz.

```
<div id='overlay'>
  <p>You have <span id='num-items'>0</span> items in your cart.</p>
  <p>Total cost is $<span id='total-cost'>0</span></p>
</div>
```

Şimdi HTML'e biraz container alanları ekleyelim. Verilerin güncellenmesi gerektiğinde containerların yazılarının güncellenmesi için jQuery kullanırız.

```
$(this).find('#num-items').text(cart.items.length);
$(this).find('#total-cost').text(cart.getTotalCost());
```

Bu ilk girişimimizden daha iyidir HTML sayfamızdaki bütün biçimlendirmeleri sahip olduğu yerde bırakır ve böylece kodun ne yaptığını daha kolaylıkla görebiliriz.

jQuery ile değiştirilecek biçimlendiricileri yönetmek için başka bir seçenek vardır. Sonuçların listesini döndürecek uygulamarla çalışırken , her simge için bir taslak gibi davranan eleman dahil edebilirsiniz ve basitçe o elemanı kopyalayıp , ne zaman yeni bir simge eklemeniz gerekse o elelmanın içeriğini ekleyebilirsiniz.

Bu söylediklerimizi uygulamaya dökelim. JQuery 'i kullanarak simgeleri ekleyip

çıkarabiliriz. Simgeler HTML tablosunda durur ve biz tablonun bütün satırlarını jQuery kodumuza dahil etmekten kaçınırız. Kaç tane simgenin önceden taşınacağını bilmediğimiz için yertutucuları kullanmak da zordur. Yani basitçe jquery tarafından içerikleri doldurulacak olan boş tablo satırları hazırlayın .

İlk görevimiz şablonumuzu harekete geçirmek için “display:none” kullanarak boş bir satır yaratmak olacak:

```
<table id="cart">
  <thead>
   <tr>
     <th>Name</th>
     <th>Qty.</th>
     <th>Total</th>
   </tr>
</thead>
<tr class="template" style="display:none;">
  <td><span class="item_name">Name</span></td>
  <td><span class="item_qty">Quantity</span></td>
  <td><span class="item_total">Total</span>.00</td>
 </tr>
</table>
```

Şimdi de şablonlamayı bizim adımıza yapan yardımcı bir fonksiyon yaratalım. Değişiklikleri şablonlaştırdığımızda sürdürebilirlik kolay olsun diye bu kodu merkezi tutulur. “template” fonksiyonu bir tablo satırının jQuery seçimini kabul ediyor. Sonuç , HTML dokümanımıza dahil edilebilcek hazır doldurulmuş bir şablondur.

```
function template(row, cart) {
  row.find('.item_name').text(cart.name);
  row.find('.item_qty').text(cart.qty);
  row.find('.item_total').text(cart.total);
  return row;
}
```

Her bir veri satırı için , şablonumuzu kopyalamalı , değerlerimizi yerine geçirmeli ve tablonun sonuna eklemeliyiz. Seçili bir elemanı kopyalamanın en kullanışlı yolu clone metoduyla olur. “clone” metodu , o anki jQuer seçiminin kopyasını oluşturur. Bir elemanı klonladıktan sonra , seçim size yeni elemanı DOM ağacına yerleştirmenizi sağlayacak şekilde değişir.

```
var newRow = $('#cart .template').clone().removeClass('template');
var cartItem = {
```

```
  name: 'Glendatronix',
  qty: 1,
  total: 450
};
template(newRow, cartItem)
 .appendTo('#cart')
 .fadeIn();
```

“template” sınıfı silindi çünkü o artık bir şablon değil! Daha sonra, “cart” elemanımızı ayarlıyoruz ve veriyi satır içine koymak için “template” metodumuzu kullanıyoruz. Yerine koyma işlemi tamamlandıktan sonra, var olan tabloya satırı ekliyoruz ve açıyoruz. Kodumuz basit kalıyor ve HTML dosyalarımızdaki biçimlendirmelerimizin hepsi ait oldukları yerlerde kalır.

Tarayıcı Bulma

Tarayıcı bulma, JavaScript aracılığıyla kullanıcının hangi tarayıcıyı kullandığını tespit etmektir. Bu bilgiyle, tarayıcıda var olan hataların farkında olursunuz ve bu sayede sayfanızın daha düzgün çalışmasını sağlarsınız.

Tarayıcıyı bulmamızı sağlayacak iki jQuery fonksiyonu vardır:

$.browser, kullanıcının tarayıcısının Internet Explorer, Safari, Opera, ya da Mozilla olup olmadığına karar veren birkaç işarete sahiptir. Bazen sinir bozucu çapraz tarayıcı hataları üzerinde çalışmaktan kaçamayabilirsiniz.

Öte yandan, $.browser.version ise önerilmeyen, kullanmaktan kaçınmanız gereken bir faaliyet. Kullanıcının tarayıcısının o anki versiyonunu raporlar. Bu komutları kullanarak koşullara bağlı kodu şu şekilde çalıştırabilirsiniz:

```
if ($.browser.mozilla && $.browser.version.substr(0,3)=="1.9") {
// Only do this code in Firefox with version 3 (rv 1.9)
}
```

Özellik Algılama

Bu olayı bir örnekle anlatalım : Internet Explorer “opacity” CSS özelliği için direkt olarak bir desteğe sahip değil. Kodumuzda opacity komutunu kullanmadan , kullanıcının Internet Explorer kullanıp ona bağlı olarak davrandığını görmek için test edebiliriz. Ancak sorun Internet Explorer değil , sorun opacity 'nin kendisidir.

Tarayıcı algılamayı yerleştirmek için , jQuery $.support metodunu yaratmıştır. “Tarayıcı Internet Explorer mı?” diye sormak yerine “Tarayıcı opacity stilini

destekliyor mu?” diye sormamız gerekir. Şöyle ki:

```
if (!$.support.opacity) {
// Doesn’t support opacity: apply a workaround
}
```

Bu yaklaşımın güzelliği ise gelecekte opacity desteği olmayan yeni tarayıcılar ortaya çıkarsa veya Internet Explorer birden desteklemeye başlarsa eski kodunuz hala eskisi gibi çalışıyor olacaktır.

Opacity 'nin yanında test edebileceğiniz bir düzine özellik var. Örneğin ; $.support.boxModel eğer tarayıcı quirk mod'da ise “false”dödürür. “$.support.leadingWhitespace innerHTML” de alfabe dışı elemanların yazılmasını önlüyorsa “true” döndürür.Eğer gerekiyorsa Appendix A daki tüm listeyi kontrol ettiğinizden emin olun.

Ajax Nedir ?

Ajax terimi 2005 yılında Asynchronous JavaScript and XML deyiminin kısaltılarak bilgisayar kavramı olarak hayatımıza girmiştir. Tanımlanan kısaltmanın teknolojilerini herzaman kullanmayan Ajax benzeri işler yaparlarken çoğu insan bu terimi sorunsal bulmuşlardır. Mevcut sayfay bozmadan bir kullanıcının tarayıcısı bir sunucu ile etkileşim sağlayan bir teknoloji ve teknikle bu terim hayatımıza yerleşti.

Sunucu ile etkileşen ajax olmayan metodlar web'te alışık olduğumuz modele benzer.Kullanıcı linkin üzerine tıklayıp veya bir form doldurup sunucuya istek gönderir. Sunucu da buna tarayıcının yüklediği veya gösterdiği yeni bir HTML sayfası ile cevap verir. Sayfa yüklenirken kullanıcı beklemek zorunda kalır.

Ajax tarayıcıdan sunucuya sayfa yeniden yüklenmeden bir istek göndermemize olanak sağlar ve böylece kullanıcı çalışmaya devam ederken sayfanın bir kısmını güncelleyebilir. Bu da sunucunun kullanıcıya karşı daha cevaplayıcı olmasını sağlar.

Ajax tarayıcı üzerinde çalışırken, sunucu ile dinamik olarak etkileşim için bir yol gerekir. Her tarayıcı bunu sağlamak için az çok aynı metodlar kullanır. JQuery burada devreye girer ve bu farklıları düşünmemize gerek kalmaz.

Uzaktan HTML yükleme

Kodlayarak bu konuya başlamak iyi bir yol olabilir. Varsayalım ki sitemizi tasarladığımız adam gittikçe işimize karışmaya başladı ve Ajax uygulamalarını sitesinde istiyor olsun. Bu Ajax geliştirmelerini sitemize en kolay şekilde uygulamak için jQuery Ajax fonksiyonu olan “load” komutunu kullanabiliriz.

“load” komutu mucizevei bir şekilde sunucunun HTML dosyasını kapar ve dosyanın içeriklerini o anki web sayfasının arasına sokar. Statik HTML dosyalarını veya dinamik HTML çıktılarını yükleyebilirsiniz. Nasıl kullandığımızı anlatan küçük bir örnek:

```
$('div:first').load('test.html');
```

Bu komut satırı Ajax'ın fonksiyonelliğini gösteren çok küçük bir parçadır. Dinamik olarak test.html dosyasının tüm içeriğini sayfadaki ilk div'in arasına sokar. HTML'in nereye gideceğini söylemek için herhangi bir seçiciyi kullanılır ve hatta aynı anda birden fazla pozisyonlara yükleyebilirsiniz.

Hijax ile Hyberlink'leri Geliştirme

Sitemize ünlülerin biyogrofisi(celebrities’ biographies) anahtarını ekleyerek bir ana sayfa tasarlayalım. Ana sayfamızda biyografi sayfalarına götürecek standart bir grup köprüler yer alacak. Yeni Ajax donanımlarımızla kullanıcı linklere tıkladığında linkleri durduracak ve kullanıcıyı biyografi sayfasına göndermek yerine bilgileri linklere yükleyeceğiz.

Harici bilgileri yüklemek için bu mükemmel bir tekniktir. Ana sayfamız hızlı bir şekilde yüklenirken sitemizi JavaScript 'siz ziyaret eden kullanıcılar biyografi sayfalarını da hala normal link gibi ziyaret ediyor olacaklardır. Bu şekilde aşamalı olarak artan hyperlinklere hijax denir.

Sayfamıza bunu uygulamak için bazı HTML kodlarına ihtiyacımız var. Şimdilik bunu basit tutup sadece açıklama ve başlığımızı tasarlayalım:

```
<body>
<h1>Baron von Jovi</h1>
<p id="description">
It's a little known fact that Baron von Jovi …
</p>
</body>
```

Her bir ünlü için ayrı bir web sayfasına ihtiyacımız olacak. Eğer milyonlarca ünlüden oluşan bir sayfa hazırlamak isterseniz büyük ihtimalle hepsi için ayrı sayfa hazırlamak akıl karı olmayacaktır. Böyle bir durumda bilgileri veri tabanından alabilir ve işiniz kolaylaştırabilirsiniz.

Kataloğumuzu tamamladığımızda eski sitemizdeki listemize koyalım ki tıklama işlemi gerçekleştiğinde doğru biyoğrafi açılsın.

```
<ul id="biographies">
  <li><a href="barronVonJovi.html">Baron von Jovi</a></li>
  <li><a href="computadors.html">The Computadors</a></li>
  < li><a href="darthFader.html">Darth Fader</a></li>
  <li><a href="moFat.html">Mo' Fat</a></li>
  </ul>
<div id="biography">
Click on a celeb above to find out more!
</div>
```

Listemizin altına fazladan bir div daha ekledik. Bu bizim Ajax çağrılarına yanıt ekleyeceğimiz yerdir. Şimdiki adımımız ise biraz Ajax yaparak bağlantıları kesmek.

```
$('#biographies a').click(function(e) {
var url = $(this).attr('href');
$('#biography').load(url);
e.preventDefault();
});
```

İlk olarak sırasız listemizdeki tüm elemanları seçtik ve varsayılan olayın oluşumunu engelledik.Linkin orijinal hedefini yakaladık (tıkladığımız baglantıların href niteliğini alarak) ve load fonksiyonunun yazdık.

Kodumuz iyi bir şekilde çalışıyor ama tüm sayfanın içeriklerini yazmamız biraz problemli olabilir. Yeni içeriğimiz olan <h1> etiketi tüm sayfanın başlığını vermesi için kullanılmalıdır.

Seçiciler ile HTML Toplama

“load” olayımız URL kodlarının bir kısmı olarak jQuery seçicileri özelleştirmemize yarar. Sadece seçicilerle eşleşen sayfa elemanları döndürülmelidir.”load” seçicilerini kullanan formlar oldukça basittir yüklemek istediğiniz dosyadan sonra seçici kodunuzu eklemek ve herbirini boşluklarla ayırmanız yeterli olacaktır :

```
$('#biography').load('computadors.html div:first');
```

Kullandığınız seçicinizi istediğiniz kadar karışık yapabilirsiniz. Sitemiz için açıklama(description) kısmındaki bilgileri görüntülemek istiyoruz :

```
var url = $(this).attr('href') + ' #description';
```

Ancak yazdığınız kodunuzda dosya ismini boşlukla ayırdığınızdan emin olun. Eğer

bu versiyonu çalıştırırsanız , sadece yüklenen div içindeki açıklamaları görürsünüz.

İleri Seviye Yüklemeler

Genel olarak istenen şey sunuculara yazmak için bazı verileri özelleştirmektir. Örneğin ; quer kodu temellendirilmiş ve bilgi döndüren bir fonksiyonunuz olsa şöyle bir çağrı yapmanız gerekir :

```
$('div#results').load('search.php', 'q=jQuery&maxResults=10');
```

“load” fonksiyonuna yazılan ikinci parametre ise “data” 'dır. Eğer bir string içinde geçirirseniz(yukarıda yaptığımız gibi) jQuery Get isteğini çalıştıracaktır. Bunun yerine ,verinizi içeren bir obje yazarsanız da POST isteği aktif olacaktır.

Ek olarak , bir istek geri çağırım fonksiyonu ile birlikte bitirildiğinde işlemi bitirebiliriz. Geri çağırım fonksiyonu 3 parametre ile çalışır : cevap metni(asıl veri) , cevap durumunu içeren bir kod ve tüm cevap objesinin kendisi.

```
$('div#result').load('feed.php', function(data, status, response) {
// Post-processing time!
});
```

İsteğe “data” ve “callback” parametrelerinin yazılması isteğebasğlıdır

Gelecek İçin Hazırlayın : “live” ve “die”

Bütün Ajax kısımları çalıştırabilir ve artık sayfamızı müşteriye gösterme zamanı. Adam heycanlı. “Fareyi biyografinin üzerine getirdiğinde arka planı renklendirebilir misiniz? ” diyor müşteri. “Tabiki de “ diyorsunuz sizde. “2 saiyelik bir iş” Ajax kodunuza basit bir jQuery komutu ekleyerek bunu yapabilirsiniz:

```
$('p#description').mouseover(function() {
  $(this).css('background-color','yellow');
});
```

Kendinize güvenen bir sesle “İşte!” diyorsunuz ve sayfayı yenilediğinizde kodun çalışmadığını görüyorsunuz. Neden? Eğer bu konu üzerinde kafa yorarsanız p#description elemanına doküman yüklenirken olay dinleyicisi eklemek mantıklı gelebilir. Ama döküman ilk yüklenirken p#description elemanımızn yok. Ajax bunu daha sonra dinamik olarak ekleyecek. Peki ne yapmamız gerekiyor?

Bu problem için sadece Ajax kodumuza geri çağırım fonksiyonundaki “mouseover” olayını ekleyebilirdik ancak daha iyi bir yol var “live” metodu. “live” şimdiye kadar

yaptığınıza benzer bir şekilde işleyiciyi bağlamanızı sağlar . Jquery seçim kriterinizi hatırlayacak ve herhangi bir yeni elemanla eşleşme görürse olay işleyicimiz ekleyecektir.

Yazılış şekli biraz farklı ama live , Ajax kullanırken kullanışlı bir komuttur. Yukarıdaki örneğimizi düzeltmek için live metodunu kullanarak “mouseover” olayımızı tekrardan yazacağız.

```
$('#description').live('mouseover', function() {
  $(this).css('background-color', 'yellow');
});
```

Bu kodu çalıştırmamızla birlikte ne zaman #description ile eşleşen bir eleman eklense, olay işleyici kodumuz da eklenecektir. Live kullanmak için işlemek istediğiniz olayı belirginleştirir. Olay gerçeşkleştikten sonra durdurmak isterseniz die metodunu kullanmanız gerekir.

```
$('p#description').die('mouseover');
```

$.getJSON ile Verileri Yakalama

Birkaç yıl önce “mashup” deyimi birden fazla 3. taraf web sitelerinden veri kapan site ve uygulamar için icat edildi. Çoğu web site sahipleri verilerini diğer insanlara açmanın ve programcıların erişebileceği şekilde XML alanlarını açmanın yararının farkına vardı. Bu tip açık veri kaynakları API (Aplication Programming Interface)'ye karşılık gelir.

Son zamanlarda JSON(JavaScript Object Notation) ajax uygulamarındaki verileri formatlamak için popüler bir yol haline geldi. JSON XML için biraz hafif kalmıştır. Hiçbir ayrıştırma veya yorumlanması gerekmemektedir .

Jquery JSON verilerini yakalamamız için kullanışlı bir metod sunar “$.getJSON..” Bu metodun basit versiyonu URL ve geri çağırım fonksiyonunu kabul eder. Örnek olarak “celebs” etiketini delicious.com sitesinde yakalamaya çalışalım :

```
$.getJSON(
  'http://feeds.delicious.com/v2/json/tag/celebs?callback=?',
   function(data) {alert('Fetched ' + data.length + ' items!');
     alert('Fetched ' + data.length + ' items!');
});
```

Twitter Arayıcısı

Neden Delicious ile zamanımızı harcıyoruz ? Birleştirmemiz için bir Twitter akışımız var ve bunu akıllı bir şekilde yapmalıyız. Twitter ana sayfasından API bağlantısını takip etmek bize bu yolda ilerlerken yol gösterecektir. Ünlüler hakkındaki güncel toplu tweetleri JSON veri olarak döndürmek için URL araması yapacağız.

var searchTerm = "celebs";

```
var baseUrl = "http://search.twitter.com/search.json?q=";
$.getJSON(baseUrl + searchTerm + "&callback=?", function(data) {
  $.each(data.results, function() {
  $('<div></div>')
    .hide()
    .append('<img src="' + this.profile_image_url + '" />')
    .append('<span><a href="http://www.twitter.com/'
      + this.from_user + '">' + this.from_user
      + '</a> ' + this.text + '</span>')
    .appendTo('#tweets')
    .fadeIn();
});
```

Arama sonuçları “result” içeren bir obje formunda döncürülecektir. “$.each” yardımcı metoduyla her bir simge için başlangıç dizisi yazlıdığında fonksiyon yürütülecektir. Yine her simge için kullanıcının profil resmini, Twitterdaki profiline bağlanan bir link ve ünlülerler alakalı tweetlerden oluşan yeni bir div yaratır.

JQuery Ajax İlişkisi

Bir parça maceracı ve jQuey ana kütüphanesi kaynak kodlarındaki load fonksiyonunu ele geçirmiş gibi hissediyorsanız , bu fonksiyonun gölgesindeki $.ajax metodunu bulmak isteyeceksinizdir. Eğer jQuery'nin Ajax fonksiyonlarını araştırırsanız hepsinin birbirleriyle aynı olduğunu görürsünüz.

$.ajax metodu jQuery ve Ajax 'ın çok önemli bir yardımcısıdır. Bu metodun yazım kuralı şu ana kadar gördüklerimizden biraz daha karmaşıktır. Yapmak istediğiniz her şeyi söylemeniz gerekir. Tüm bunlara rağmen hala elimizdeki en kullanışlı fonksiyon $.ajax'tır. Aşağıda GET isteğinin basit bir örneği bulunmaktadır:

```
$.ajax({
type: 'GET',
url: 'getDetails.php',
data: { id: 142 },
```

```
success: function(data) {
// grabbed some data!
};
});
```

Genel Ajax Ayarları

Birden fazla Ajax çağrısına aynı script içinde aynı ayarları uygulamak isteyebiliriz. Jquery $.ajaxSetup ile ayarları özelleştirmenize olanak sağlar.

```
$.ajaxSetup({
type: 'POST'
url: 'send.php',
timeout: 3000
});
```

Diğer Ajax çağrılarını varsayılan olarak atar ama yinede ihtiyacınız olduğunda üzerine yazabilirsiniz. Yaygın varsayımları sayfanız için özelleştirerek , arzu edilen kodu , istek göndemek için oldukça basitleştirebilirsiniz.

```
$.ajax({
  data: { id: 142 }
});
```

Bu çağrı 3,000 milisaniyeden fazla sürünce zaman aşımına uğrayacak olan send.php için POST isteğini gönderecektir. Bazı ayarlar $.ajaxSetup metodu ile ayarlanamaz: error, complete, ve success geri çağırım fonksiyonları...

Harici Komut Dosyalarını $.getScript ile Yükleme

Sitemizdeki uygulamalar büyüdükçe , sayfanın yükleniş hızı da buna bağlı olarak düşer.Bu yüzden uygulamalarımız olabildiğince düzenli tutmamız gerekiyor. Tabiki de jQuery bu noktada bize yardım uzatan el olacaktır.

$.getScript fonksiyonu Ajax aracılığıyla JavaScript dosyasını yükleyecek ve yürütecektir. Eğer dikkatli bir şekilde plan yaparsak , sayfa daha fazla dosyayı sıraya sokup yüklerken kodumuzun küçük bir parçasını yükleyebiliriz.

Örneğimizde 3. bölümde gördüğümüz Jquery eklentileri ile Color(renk) eklentisi yükleyelim:

```
$.getScript('http://view.jquery.com/trunk/plugins/color/jquery.color.js', function() {
$('body').animate({'background-color': '#fff'}, 'slow');
```

```
});
```

GET ve POST İstekleri

Son olarak jQuery , istekleri yerine getirmek için bir çift yardımcı fonkisiyon içermektedir: GET ve POST. Bunlar $.ajax tarzı fonksiyonlardır ancak sadece URL veya istediğiniz herhangi bir veriyi seçmekte daha kullanışlı fonksiyonlardır.

Bu iki çağrı HTTP istek farkı dışında neredeyse aynıdır. $.get GET isteğini, $.post ise POST isteğini harekete geçirir. İki istek de aynı parametrelerle çalışır :

```
$.get(url, data, callback, dataType);
$.post(url, data, callback, dataType);
```

“data” parametreleri sunucuya gönderilmesi gereken tüm parametreleri barındırmalıdır. $.ajax fonksiyonu ile belirttiğimiz geri çağırı veri durumlarını cevaplıyacaktır. “type” parametresi geri çağırım fonksiyonuna geçecek veri tiplerini belirlemenize olanak sağlar. (xml ,html , script, json , jsonp ,text bunlardan biri olabilir). Aşağıdaki örnekte bunların kullanımlarını görebilirsiniz :

```
$.get("getInfo.php", function(data) {
alert("got your data:" + data);
});
$.post("setInfo.php", {id: 2, name: "DJ Darth Fader"});
```

Evet , bu metodları kullanmak kolay ve uygun olabilir ancak dikkatli olmalısınız! $.get ve $.post metodlarının hızlı olması gerekmektedir. Geri çağırım fonksiyonu ilk çağırıldığında her şey yolunda gider. Eğer küresel bir olay işleyici ile izlemiyorsanız çağrınızın başarısız olduğunu hiç bir zaman fark edemezsiniz. Bu yüzden bazı durumlarda daha karmaşık olan $.ajax metodunu kullanmak zorunda kalabilirsiniz.

JQuery Ajax Olayları

JQuery 'de yerel ve evrensel olarak iki tip Ajax olayı vardır. Yerel olaylar sadece bireysel Ajax isteklerine yanıt verir. Diğer olaylarda gördüğümüz gibi yerel olayları geri çağırımlar gibi işleyebilirsiniz. Evrensel olaylar herhangi dinlenen bir işleyiciyi yayınlar ve böylece bütün istekleri yanıtlamak için daha fonksiyonel bir yol sağlar.

Örnek olarak , Jquery error'u yerel olay , ajaxError'u ise evrensel olay olarak tanımlanabilir.
“error “ olayı bireysel bir istek gibi işlenir ancak “ajaxError” olayı ise herhangi bir istekte hata olarak belirebilir. Yerel olaylar $.ajax çağrısında şu şekilde işlenirler :

```
$.ajax({
  url: "test.html",
  error: function() {
  alert('an error occurred!');
}
});
```

Global olaylar ise genellikle kullanıcıyı cevaplamak isteğiniz Dom düğümüne eklenir. Örneğin ; görüntülenen Ajax error mesajınız için bir div elemanınız olabilir:

```
$("#msg").ajaxError(function(event, request, settings) {
$(this).html("Error requesting page " + settings.url + "!");
});
```

Ne zaman sayfadaki herhangi bir Ajax çağrısında bir hata oluşsa , işleyici çağrılır.

Etkileşim:Ajax Kullanmak

Şu ana kadar bir çok konu anlattık. Ajax araçları hakkında genel bilgi sahibi olduk. Şimdi müşterimizin siteyi dünyadaki en büyük fotograf paylaşım sitesi olmasını arzuladağını ve bu konu ile ilgili bir kaç araştırma yapıp böyle bir sitenin karşılaşabileceği sorunlar hakkında bilgi sahibi olduğunu varsayalım. Tüm dünyadan bu siteye fotograf transferi yapılacağını düşünün. Böyle bir siteyi güçlendirmek için neler yapabiliriz?

Resim Etiketleme

Etiketleme içeriğinizle ilgili bir metadata koleksiyonu olduğunu mükemmel bir yol ile ispatlamıştır. Kullanıcıların açıklama yapmak istedikleri simgelere yazılar eklemesi ile çalışan bir sistemdir. Eğer bir çok insan aynı kelimeyi kullanırsa ikisi arasında bir bağlantı olduğundan emin olabilirsiniz. Bu da diğer kullanıcıların alakalı içerikler için sitenize göz atmasında yardımcı olabilir.

XML Kullanma

JSON Web üzerinde veri değişimi formatları konusunda geleceğin altın çocuğu olmasına rağmen XML 'in daha çok web hizmeti bulduğunu söyleyebilirsiniz. XML , JSON 'dan daha yaşlıdır bu yüzden daha fazla kütüphaneye sahiptir. Ayrıca JSON'la oynamak daha kolay olmasına rağmen , Jquery XML'i data işlemede daha iyi kullanır.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<celebs>
  <celeb id="421">
    <name>Johnny Stardust</name>
    <image>johnny_200.jpg</image>
  </celeb>
  <celeb id="422">
   <name>Kellie Kelly</name>
   <image>kellie_200.jpg</image>
  </celeb>
</celebs>
```

Şimdi , elimizdeki dataları ilk uygulamamızı kullanışlı kılmak için güncellememiz gerkiyor. İlk geçiş olarak dosya isimlerini bölüp her birini $.each kullanıp yeniliyoruz. Ama şimdi istekler biraz karmaşık. Bize gerekli olan bilgileri ayıklayıp XML düğümlerini iletiyoruz. İyi haber ise jQuery bize XML dökümanları ile DOM arasında bir anlaşma sağlar.

Bu jQuery etkilerinin tümümü kullanabilmemiz anlamına gelir. Örneğin; düğümleri isimlerine göre bulmak için find()komutunu kullanbiliriz.(Next ve prev komutlarıyla)

```
success: function(data) {
$(data)
.find('celebs')
.children()
.each(function() {
var node = $(this);
var id = node.attr('id');
var name = node.find('name').text();
var image = node.find('image').text();
_gallery.display({'id': id, 'image': image, 'name': name});
});
}
```

Her celeb düğümünü dönğü içine alıyoruz ve bu düğümlerin ID,isim ve resim URL'lerini çıkarıyoruz. Daha sonra display fonksiyonunun kodunu yeni veri objemizi kabul etmesi için değiştirmek zorundayız. Jquery data fonksiyonu aracılığıyla“img” etiketinin kendi içindeki değeri saklıyacağız.

Verilerimizdeki bir ünlünün isimine eriştik. Aynı zamanda erişebilirliği ve standart uyum sorununu önceki kodumuzla çözebiliriz. Ünlülerin isimlerini içeren resimlerimize “alt” özelliği ekleyebiliriz :

```
display: function(dataItem) {
```

```
$('<img />')
.attr({
src: '../../images/' + dataItem.image,
alt: dataItem.name
})
.hide()
.data('id', dataItem.id)
.load(function() {
$(this).fadeIn();
})
.click(function() {
CELEB.load($(this).data('id'));
})
.appendTo('#gallery');
}
```

Verilerle DOM düğümleri artırmak çok yararlıdır ; iç “click” işleyicisi için etiket verilerini yüklemek için ihtiyaç duyduğumuz ID 'leri kolayca bilebiliriz. ID'lerimizi elde ettikten sonra resim etiketlemede bir sonraki adıma hazırız demektir.Etiket verisini kabul etme ve gösterme.

Bu mantığı GALLERY görsellerindede uygulayabilirdik ancak biz yepyeni bir içerikle uğraşıyoruz. Bunun yerine güzel ve okunabilir tutmak için yeni bir CELEB görseli içine ayıracağız.

```
var CELEB = {
  url: 'celebs.json',
  load: function(image_id) {
    var _celeb = this;
    $('#details input').attr('disabled', 'disabled');
    $.getJSON(
      this.url,
      function(data) {
        $('#details input').removeAttr('disabled');
        _celeb.display(data);
});
},
display: function(data) {
  $('#id').val(data.id);
  $('#name').val(data.name);
  $('#tags').val(data.tags.join(" "));
}
}
```

Veriyi yakalamak için $.getJSON kullanıyoruz. Ancak bu beforeSend veya complete olay işleyicilerini yoksunlaştırır. Bunun yerine istek göndermeden önce form alanlarını geçersiz yapıp ,veriler geri geldiğinde geçerli kılacağız. “disable” özelliğini ayarlamak için “attr” metodunu kullanacağız.

Veriler geri geldiğinde onları display fonksiyonuna yazıyoruz ve alanlara yerleştiriyoruz. Değişik sunucu cevaplarını görmek için “celebs.json “ nın içeriklerini değiştirmeyi deneyebilirsiniz.

Form Verisi Gönderme

Yapmamız gereken tek şey gönderilecek olan formu verimizle birleştirmek. Bölgelerdeki değerlerin hepsini bir kodda birleştirebilir(elverişsiz olur) veya formdaki tüm anahtar/ değer çiftlerini tutan bir obje yaratabilirdik. İkincisi daha zahmetsiz olur ancak Jquery bize sihirli Ajax kodu ile bir yol daha sunuyor..Formdaki verileri “serialize” metoduyla kolayca birleştirebilir ve gönderilmeye hazır hale getirebiliriz.

“serialize” metodu “input” alanlarını çekecektir. Bu özelliği avantaja çevirmek istiyorsak alanların isimlendirilmiş olduğundan emin olunuz.

```
<form>
  <input type="text" name="name" />
  <input type="text" name="tags" />
  <input type="hidden" name="id" />
  <input type="button" value="update" />
</form>
```

Biçimlendirmemizi tam anlamıyla yapabilmek için formun kendisinin jQuery seçicisinde serialize metodunu çağırmamız gerek.

```
var form_data = $("form").serialize();
```

Veriyi serileştirmek , kodu bölge ismi ve değerlerinin & ile ayrıldığı bir formata çevirir.

```
name=Kellie+Kelly&tags=b-grade+has-been+rich&id=8
```

Eğer veriniz organize edilmiş bir formatta ise “serializeArray” komutunu kullanabilirsiniz. Tuhaf olarak bu fonksiyon “array” değilde form alanındaki anahtar/değer çiftlerini içeren bir obje döndürür.

```
update: function() {
  var form_data = $('form').serialize();
  $.post(this.set_url, form_data, function() {
    $('#status').text('Update successful!');
});
}
```

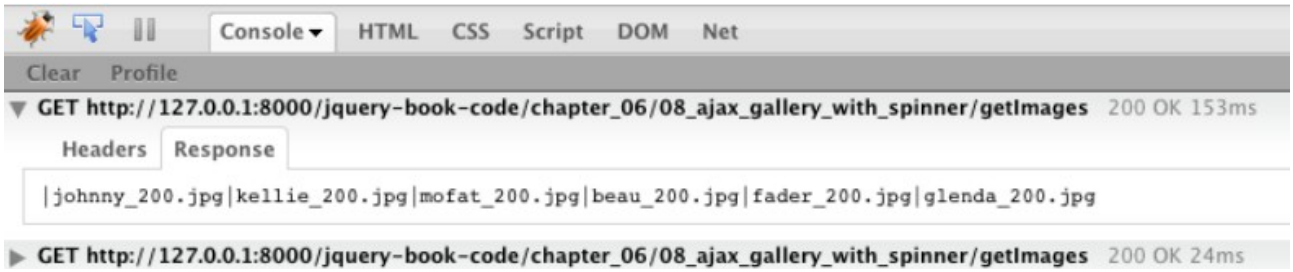
“$.post” metodu kullanması kesinlikle daha kolaydır. Ancak daha önceden bahsettiğimiz gibi , neyin yanlış gittiğini hiç bir şekilde anlayamayız. Yeni arkadaşımız “$.ajax” yi çağırarark kodumuzu değiştirelim. Bu şekilde “Güncelleme Başarılı” mesajını ve hata mesajını class'ımıza ekleyelim :

```
$.ajax({
  type: "POST",
  url: this.url,
  data: form_data,
  beforeSend: function() {
    $('#ajaxDetails').addClass('progress');
},
error: function() {
  $('#status').text('Update failed—try again.').slideDown('slow');
},
success: function() {
  $('#status').text('Update successful!');
},
complete: function() {
  $('#ajaxDetails').removeClass('progress');
  setTimeout(function() {
    $('#status').slideUp('slow');
}, 3000);
}
});
```

“setTimeout” metodunu ekleyerek Submit butonuna tıklandığı zaman birkaç saniye içinde mesaj çıkmasını sağlayabiliriz.

```
$('#update').click(function() {
  CELEB.update();
});
```

Son olarak tüm bu satırları bağladığımızda şekildeki gibi bir şekil elde ederiz :



BÖLÜM 7 : FORMLAR , KONTROLLER VE DİYALOGLAR

Web ilk başlarda salt-okunur bir dildi, neredeyse sonsuz sayıdaki belgerle.Web geliştiricileri daha fazlasını istediler. Özellikle tasarladıkları web sayfalarını insanların sadece okumasını istemiyolardı. Daha fazlasını isteyen geliştiriciler insanların ziyaretci defterine kayıt olup sitelerini nasıl yarattıklarını görmelerini istediler.

JavaScript , HTML form elemanlarına yardım etmek için devreye girdi. Ancak bu amaçla üretilen kodlar genellikle şişirilmiş olurdu. Jquery daha basit kontrol oluşumlarına olanak sağlar ve fonksiyonel kontrolleri daha hızlı gerçekleştirmeye yarar.

Şimdi müşterimiz bazı satır içi düzenlemeleri, eğlenceli form doğrulama mesajlarını, havalı iletişim kutularını ve her şeyi sitesinde istiyor. Neyseki jQuery bize tüm bu düzenlemeler için olanak sağlıyor.

Formlar

HTML formları oldukça eski oldukları için sunucuda görüntülenirken biraz kaba durabilirler. JavaScript sayesinde bu eski düzen formları daha havalı web uygulamaları haline getirebiliriz.Eğer bunu JavaScript ile yapıyorsak jQuery ile eğlenceli hale getirebiliriz.

Formlar DOM elemanı olduğu için jQuery ile rahatça işleyebiliriz. Ancak form elemanları tipik DOM elemanlarınız değildir bu yüzden bu işlemeyi daha verimli kulmak için bazı jQuery hilelerinden faydalanmamız gerekir. Şuana kadar bunlardan bazılarını gördük zaten, ama bu bölümde bahsettiğimiz hileleri daha yakından inceleyeceğiz .

Basit Fom Onayı

Form onaylama sıkıcı gibi görünsede temel bir kavramdır. İyi tasarlanmış ve uygulanmış formlar kullanıcıyı sitenize girme konusunda ikna edebilir.

JQuery form doğrulamanın asıl meselesini anlamamızı engellemesine rağmen , form değerlerine erişebilmemiz için kullanışlı metodlara olanak sağlar. Form alanlarını herhangi başka bir elemanmış gibi seçebiliriz ama kodu daha verimli ve okunabilir kılmak için bazı filtrelemeler kullanmamız gerekir.

“input” filtresi bütün elemanları seçer ; seçim kutularını , yazı alanlarını , düğmeleri... Filtrelemeyi bir örnek ile anlayalım :

```
$('#myForm:input').css('background-color', 'lemonchiffon')
```

Elemanı seçme konusunda daha eleyici olmak isterseniz , başka filtreleme seçeneklerine de göz atmalısınız ; text, password, radio, checkbox, submit, button, image (resim düğmeleri için) ve file. Birden fazla filtre kullanabilrisiniz.

Ayrıca , elemanları değerlerine ve durumlarına göre seçmek isterseniz “enable” ve “disable” filtrelerinden de faydalanabilirsiniz. “checked” ve “select” filtreleriylede radyo düğmelerini ve seçim kutusu simgelerini seçebiliriz.

Elemanınızı seçtikten sonra elemanların değerleri bulmanız gerekir böylece isteklerinize göre onları geçerli kılabilirsiniz. “val” fonksiyonu form alanındaki değeri döndürür. Daha sonra basit onaylamaları gerçekleştirebiliriz. Örneğin :

```
$(':submit').click(function(e) {
  $(':text').each(function() {
    if ($(this).val().length == 0) {
      $(this).css('border', '2px solid red');
}
});
e.preventDefault();
});
```

Formunuzu bir iki harfle doldurmaya çalışın , boş bıraktığınız her kutucukta harf kırmızı renge dönecektir.

“val” fonksiyonu kutuları ve cinsiyet işaretlerini seçmek içinde kullanılabilir. Örnek olarak ; kullanıcı seçimi değiştirdiğinde cinsiyet seçimini(genel adı radyo tuşu) alarma geçirelim.

```
$(':radio[name=sex]').change(function() {
  alert($(this).val());
});
```

“change” olayı formdaki bir değer değiştiğinde çalışmaya başlıyacaktır. “input” veya “textarea” alanlarındaki yazılar için kullanıcı önceki değeri değiştirğinde “change “ komutu çalışmaya başlar.

Onaylama örneğimize tekrar dönelim. Şimdi boş alanları kullanıcı bir sonraki alana geçtiğinde test edelim. “blur” fonksiyonunu kullanarak bunu test edebiliriz. Bu satır içi onaylama için kullanışlı bir fonksiyondur.

```
$(':input').blur(function() {
  if ($(this).val().length == 0) {
    $(this)
      .addClass('error')
      .after('<span class="error">This field must …</span>');
  }
});
$(':input').focus(function() {
  $(this)
    .removeClass('error')
    .next('span')
    .remove();
});
```

Doldurulan alanları kontrol ettik ama minumum veya maximum karakter sayısı , geçersiz karakterler , şifre onayıda kontrol edilmelidir.

“submit” Olayı

Formun submit edilmesi(gönderilmesi) sırasında “submit” fonksiyonunuda kullanabilirsiniz. Submit tuşu tıklama olayını kontrol etmekten daha iyi bir çözümdür. Aynı zamanda bu fonksiyon sayesinde Enter tuşu ile de submit olayı gerçekleştirilir. Eğer submit olay işleyicisinden false döndürürseniz form gönderilmeyecektir. Eğer herhangi bir alan boş bırakılırsa ekrana mesaj çıkacaktır.

```
$("form").submit(function() {
  var error = false;
  $(this).find(":text").each(function() {
    if ($(this).val().length == 0) {
      alert("Textboxes must have a value!");
      $(this).focus();
```

```
    error = true;
    return false; // Only exits the “each”loop
  }
 });
 if (error) {
  return false;
}
return true;
});
```

Onay Eklentisi Kullanrak Form Onaylama

Her alan doldurduktan sonra bir sonraki satıra geçirilken doldurulan satırın onaylanıp onaylanmadığının kontrol edilmesidir.Aşağıdaki örnekte bir siteye kayıt olma formu bulunmaktadır :

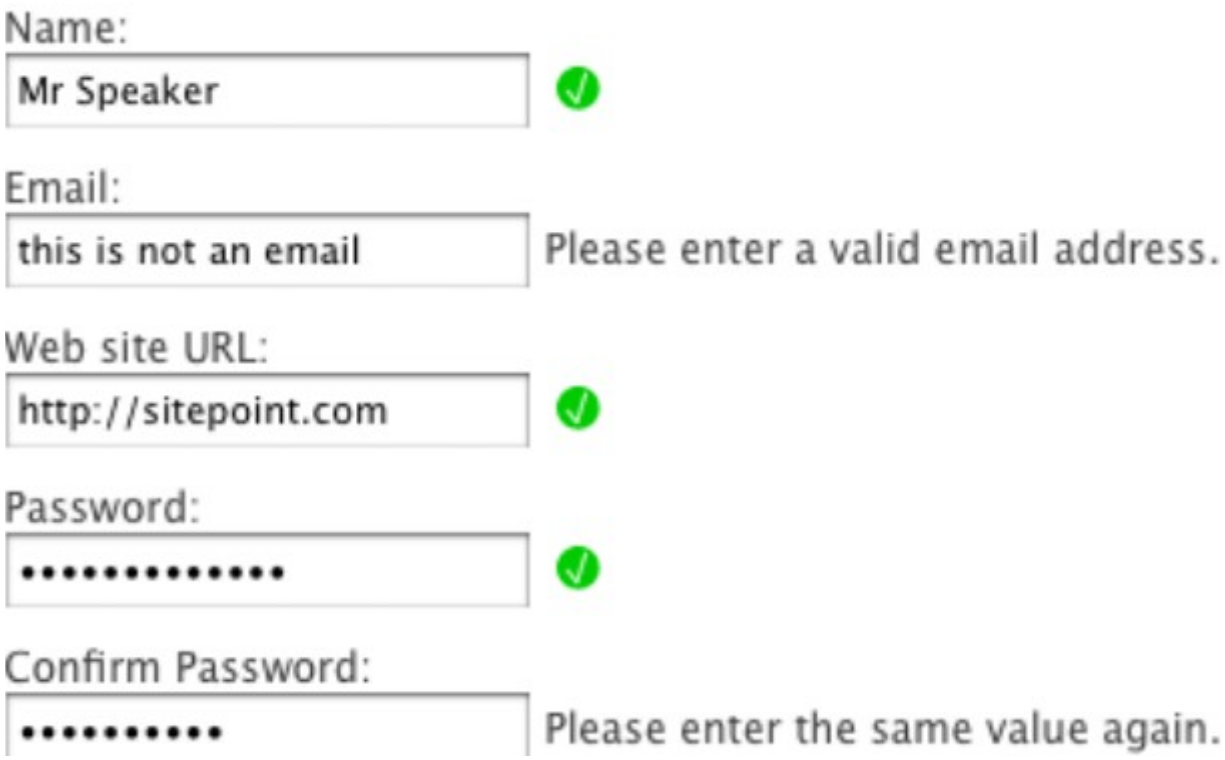
```
<div id="signup">
 <h2>Sign up</h2>
 <form action="">
  <div>
   <label for="name">Name:</label>
   <input name="name" id="name" type="text"/>
 </div>
 <div>
  <label for="email">Email:</label>
  <input name="email" id="email" type="text"/>
 </div>
 <div>
   <label for="website">Web site URL:</label>
   <input name="website" id="website" type="text" />
 </div>
 <div>
  <label for="password">Password:</label>
  <input name="password" id="password" type="password" />
</div>
<div>
     <label for="passconf">Confirm Password:</label>
     <input name="passconf" id="passconf" type="password" />
</div>
  <input type="submit" value="Submit!" />
</form>
</div>
```

Onaylama eklentisini kullanmak için “validate” fonksiyonunu çağırmamız gerekir. Bu fonksiyona bir çok seçenek yazabiliriz ama en önemlisi “rule” seçeneğidir. Kullanıcının girdiği verilerin geçerliliği için tanımladığınız kurallardan oluşur.

```
$('#signup form').validate({
rules: {
name: {
required: true,
},
email: {
required: true,
email: true
},
website: {
url: true
},
password: {
minlength: 6,
required: true
},
passconf: {
equalTo: "#password"
}
},
success: function(label) {
label.text('OK!').addClass('valid');
}
});
```

Tabikide doğrulama için kullanabileceğiniz bir çok kural vardır ve bunlardan istediğinizi seçebilirsiniz. Biz örnek kodumuzda gerekli olduğunu düşündüğümüz required , email , url , minlength ,equalto fonksiyonlarını kullandık. “Required” eğer alan doldurulmamış ise bir hata döndürecektir. Email “@” içerip içermediğini url ise http:// ile başlayıp başlamadığını kontrol eder. “rule” objemizin içinde her bir alan için ayrı objeler tanımladık. Son olarak “equalTo” o alan ile jquery seçicisinin işaretlediği başka bir form alanının eşit olup olmadığına bakar.

Bir alan hata mesajı verirse onay eklentisi yeni bir “label” ekleyecektir. Bu mesajı değiştirebilirsiniz.



Maksimum Uzunluk Göstergesi

Müşterimiz geri dönüşleri 130 karakterle sınırlamak istiyor. Böylece kullanıcıların gereksiz şeyler yazmasını engellemek istiyor. Textarea'mızda "maxlengt" sınıfı yaratarak hedeflenen işi gerçekleştirebiliriz. Script kodumuza yeni bir olay işleyicisi olan "span" i ekleyerek bu görevi tamamlayacağız.

```
$('.maxlength')
.after("<span></span>")
.next()
.hide()
.end()
.keypress(function(e) {
// handle key presses;
});
```

"span"ı ekledikten sonra textarea hala seçili elemanımız. "span"ı değiştirmek için next olayını saklamalıyız bunun için hide olayını kullanıyoruz.

Şimdi form elemanımıza geri döndük. Basılan tuşları algılaması için "keypress" olay işleyicisini kodumuza ekledik. Şimdide olay işleyici fonksiyonun içinde hala bir karakter girilip girilmeyeceğini kontrol edelim :

```
var current = $(this).val().length;
```

```
if (current >= 130) {
  if (e.which != 0 && e.which != 8) {
    e.preventDefault();
}
}
```

“lenght” fonksiyonu ile elde ettiğimiz uzunluk izin verilen maksimum değerden büyükse harf girişini preventDefault komutuyla önlüyoruz.

“keypress” olay işleyicisi faaliyetteyken basılan tuşların ASCII karşılıkları “which” komutuyla elde edilir. Silme ve boşluk bırakma işlemlerine izin verildiğine dikkat ediniz (delete =0 , backspace =8). Eğer bu olayı engellemezseniz kullanıcıya izin verilen sınırı aşmasıyla ilgili bir mesaj gönderilebilir.

```
$(this).next().show().text(130 – current);
```

Son olarak yapmamız gereken şey yarattığımız “span”lerdeki kalan karakter sayısını göstermektir. Bir sonraki elemana haraket edip, görünür olduğundan emin olur ve ne kadar daha karaktere izin verildiğini göstermek için yaptığımız hesaplamaları sonuç olarak ekranda gösteriyoruz.

Form İpuçları

Formun sayfada kapladığı alanı azaltmanın en iyi yolu form alanlarının yanlarındaki açıklamaları kutucuğun içine taşımaktır. Kullanıcı kutucuğu doldurmak için tıkladığında açıklamanın kaybolup kullanıcının doldurmasına izin verilmelidir.

Eğer bunu “data” ile yaptığımızı düşündüyseniz yanılmadınız. Her silinebilen nesne için varsayılan değeri “data” da saklayıyoruz ve eğer kullanıcı buradan çıktığında kutucuk hala boşsa yazıyı kutucuğa tekrar yazıyoruz.

```
$('input.clear').each(function() {
  $(this)
    .data('default', $(this).val())
    .addClass('inactive')
    .focus(function() {
      $(this).removeClass('inactive');
       if ($(this).val() == $(this).data('default') || '') {
         $(this).val('');
       }
  })
.blur(function() {
  var default_val = $(this).data('default');
  if ($(this).val() == '') {
   $(this).addClass('inactive');
    $(this).val($(this).data('default'));
   }
  });
});
```

Her bir elemana gidip döküman yüklenirken varsayılan değeri kaydetmemiz gerekiyor. Kullanıcı “input”larımıza girdiğinde veya çıktığında focus ve blur olayları harekete geçiyor. “focus” ta kutucuktaki yazıların varsayılan (ilk açıklamalarla) yazılarla aynı olup olmadığı kontrol edilir. Eğer öyleyse yazılar silinir. Eğer kutucuk boşsa varsayılan yazılar tekrar yerleştirilir.Örneğimizde bu varsayılan yazıları normalden biraz daha açık şekilde kullanmayı tercih ettik.

Tüm Onay Kutuları Kontrol Etme

Girdiler ve yazılar bizim kontrolümüz altında olmasına rağmen başka kontrol seçeneklerini uygulamamız gerekebilir. Tikleme gerektircek kontrol kutularını ele alalım. Her birini ayrı ayrı işaretleyebilmeli ve altda bütün kutucukları seçebilen bir “check all” seçeneği olmalıdır.

Jquery form filtreleriyle bu istekleri gerçekleştirebiliriz. Aynı grupta olan bütün kontrol kutularını seçip , kontrol edebilir veya temizleyebiliriz. HTML formlarında kontrol kutularını alakalı elemanlarla aynı isim altında gruplandırıyoruz.

```
<div class="stats">
  <span class="title">Reason for Celebrity</span>
  <input name="reason"
    type="checkbox" value="net" />Famous on the internet<br/>
  <input name="reason"
```

```
   type="checkbox" value="crim" />Committed a crime<br />
  <input name="reason"
   type="checkbox" value="model" />Dates a super model<br />
  <input name="reason"
   type="checkbox" value="tv" />Hosts a TV show<br />
  <input name="reason"
   type="checkbox" value="japan" />Big in Japan<br />
  <hr />
  <input class="check-all"
   name="reason" type="checkbox" /><span>Check all</span>
</div>
```

Son onay kutusunu özel bir “class” olan “check-all” olarak tayin ettik. Bu kutu bizim ana onay kutumuzmuş gibi davranıcaktır. Kutu işaretliğinde veya işaret kaldırıldığında kodumuz devreye girecektir.
İlk olarak ana kontrol kutumuzla aynı isimdeki kontrol kutularımız seçecek bir seçici kod oluşturuyoruz. Bu birkaç kodu bir araya yapıştırmayı gerektirir. Kodumuz şu şekilde görünecektir : “checkbox[name=reason]”.

Daha sonra ana kontrol kutumuzdaki ile aynı kontrol değerlerini ilgili kontrol kutularımızda ayarlıyoruz. Kodumuz kullanıcı değeri değiştirdikten sonra çalışmaya başladığı için “checked” özelliği kontrol kutumuzun yeni durumunu yansıtacaktır. Bu da ilgili elemanların duruma göre seçilip seçilmemesine yol açar.

```
$('.check-all:checkbox').change(function() {
  var group = ':checkbox[name=' + $(this).attr('name') + ']';
$(group).attr('checked', $(this).attr('checked'));
});
```

Satır İçi Düzenlemeler

Satır içi düzenlemeler Ajax gücünü göstermek için faydalı kontroller sağlarlar. Eğer ilk defa satır içi düzenleme kutusu kullanıyorsanız büyülenebilirsiniz çünkü tam olarak çalışmasını istediğiniz gibi çalışırlar.

Efektleri oluşturmak için bir sürü yol bulunmaktadır. En kolay olanı etiket olarak form alanlarını gizlemektir .Kenarları kaldırmak , sayfanızla aynı arka plan rengini vermek ve kullanıcı buraya tıkladığında kenarları tekrar görünür kılmak. Ancak eğer sayfanın birçok farklı parçalarının düzenlenebilir olmasını istiyorsanız, bunu gerçekleştirmek için ekstra biçimlendirme ve stillere çok ihtiyacınız olabilir.

Sonuç olarak form dışı elemanları düzenlemek daha yaygın bir yaklaşımdır. Paragraf ve başlık etiketleri örneklerden bazıları. Kullanıcı bu etiketlere tıkladığında

“textarea” ile kullanıcının etkileşime geçeçeği yazılar yazı kutusunda belirecektir. Görev tamamlandığında , orijinal etiketler yeni içerikle yer değiştirilecektir.

İçerikleri düzenlenebilir kılmak için sınıfları kullanırız. Basit tek satırlar ve uzun mesajlar için “input” elemanını kullanacağız . Ayrıca her bir elemana özel tek id verdiğimizden emin olmalıyız. Veri tabanını güncellemek ve sayfa dolarken yeni verileri yüklemek için sunucuya verileri gönderebiliriz.

```
<h3 id="celeb-143-name" class="editable">Glendatronix</h3>
<p id="celeb-143-intro" class="editable-area">
Glendatronix floated onto the scene with her incredible debut …
</p>
```

Bu kodu çalışabilir kılmamız için birkaç olayı daha eklememiz lazım. İlk olarak kullanıcın elemanın düzenlenebilir olduğunu anlaması için “hover” olayını eklemeliyiz.

Aynı zamanda kullanıcı düzenlenebilir içeriğe tıkladığında çalışmaya başlıyacak “click” ve düzenlemenin sonlandığını işaret edecek “blur” olaylarınıda ele almalıyız :

```
$(".editable, .editable-area")
  .hover(function() {
    $(this).toggleClass("over-inline");
})
.click(function(e) {
  // Start the inline editing
}).blur(function(e) {
  // End the inline editing
});
```

Kullanıcı düzenlenebilir alanlara tıkladığında kodumuz aktif olucak. Öncelikle “$editable” değişkeninde tıklanan ve saklanan referansı ele aldık. Bu bizim sürekli seçmemizi önlüyecektir. “hasClass” ile active-inline class'ı kontrol edeceğiz. Eğer eleman zaten active-inline class ise kullanıcı düzenleme kutusunda demektir. Düzenleme kutusunu başka bir düzenleme kutusu ile değiştirmeyi tercih etmeyiz :

```
// Start the inline editing
   var $editable = $(this);
   if ($editable.hasClass('active-inline')) {
     return;
}
```

Dizi içeriğinin başındaki ve sonundaki boşlukları $.trim metoduyla kaldıracağız.

Bunu HTML'nizin nasıl yerleştirileceğine bağlı olarak metin kutusunda engellemek isteyeceğiniz boşluklar olabileceği için kullanmanız gerekiyor.

Daha sonra aktif sınıfımızı ekliyoruz ve “empty” olayı ile o anki elemanı temizliyoruz. Bu komut “remove” ile benzerlik gösterir ama fark olarak “empty” fonksiyonunu çağırmamız elemanın kendisini değilde çocuklarını kaldıracaktır :

```
var contents = $.trim($editable.html());
$editable
  .addClass("active-inline")
  .empty();
```

Sonunda yeni “textarea” mızı göstermemizin zamanı geldi. Hangi form elemanlarının birleştirilceğini anlamak için editable class ile kontrol edeceğiz. Çıkardığımız içeriklerin elemanlarına yeni değerlerimizi atacayacağız ve hedef eleman ile birleştireceğiz :

```
// Determine what kind of form element we need
var editElement = $editable.hasClass('editable') ?
 '<input type="text" />' : '<textarea></textarea>';
// Replace the target with the form element
$(editElement)
  .val(contents)
  .appendTo($editable)
  .focus()
  .blur(function(e) {
    $editable.trigger('blur');
});
```

“trigger” kullanışı konusunda meraklı olabilirsiniz. Olayı harekete geçirmek için farklı bir yoldur. “trigger” olayı benzerlerine göre daha kullanışlıdır.
Kodumuzu .editable ve .editable-area elemanlarını kullanarak tekrar yazalım :

```
$('.editable, .editable-area').bind({
hover: function(e) {
// Hover event handler
},
click: function(e) {
// Click event handler
},
blur: function(e) {
// Blur event handler
}
```

```
});
```

Kodumuza geri dönecek olursak üzerinde düzenleme yaparak varsayılan durumumuza geri dönebiliriz. Form elemanlarının değerleri çekeceğiz ve $.post ile sunucuya gönderceğiz. POST işlemi gerçekleştiğinde yazıları güncelleyeceğiz ve kaydedildiğine dair mesaj ileteceğiz. Daha önceki bölümlerimizde gördüğümüz Ajax fonksiyonelliği ile boş bir "save" dosyasının sunucu-tarafı yanıtlarını taklit ediyoruz. Gerçek hayat uygulamalarında veritabanına kaydedilen değişiklikleri kontrol etmek isteriz :

```
.blur(function(e) {
  // end the inline editing
  var $editable = $(this);
  var contents = $editable.find(':first-child:input').val();
  $editable
    .contents()
    .replaceWith('<em class="ajax">Saving …</em>');
// post the new value to the server along with its ID
$.post('save',
  {id: $editable.attr('id'), value: contents},
  function(data) {
    $editable
      .removeClass('active-inline')
      .contents()
       .replaceWith(contents);
});
});
```

Yukarıda kod parçamıda yeni olarak iki tane jQuery fonksiyonu bulunmaktadır. Bu fonksiyonların ikiside tüm DOM içeriklerini döndürür.

Otamatik Tamamlama

Söylemek istediğimiz , aratmak istediğimiz değerleri , sitede var olan değerlerden baş harfleriyle çıkarım yapıp seçenek olarak kutucuğumuzun altında çıkmasıdır. Şekilde otomatik tamamlamanın bir örneğini görmekteyiz :

İlk olarak bir eklentiye ihtiyacımız vardır. Örneklere hızlıca göz atalım ve sonra sayfamıza dahil edelim.

Aynı zamanda bazı CSS stillerinede ihtiyacımız var. Burada eklenti içeren örnek bir CSS dosyası bulunmaktadır. Böylece eklenen sınıflarınlar hakkında bilgi sahibi olursunuz. Biz bir çok seçenek arasından, aşağıda sıralanmış halde beliren modeli tercih ettik. Otomatik tamamlama menüsü üzerine bir seçim kutusu iliştirir. Basit formumuzdaki alana bunu uygulayın:

```
<label for="location">Last known whereabouts:</label>
<input type="text" id="location"/>
```

Şimdi otomatik tamamlamanın bizim için neler yapabileceğini görelim. Varsayılan olarak bir dizideki saklanmış yerel bir koleksiyon gerektirir . Sayfamızda HTML listesinden gelen verilerimizi saklamak istediğimizden bizim için mükemmeldir.:

```
var cities = ['New York', 'Melbourne', 'Montreal', 'London' …];
$('#location').autocomplete(cities,{
  autoFill: true,
  selectFirst: true,
  width: '240px'
```

Basitçe bir JavaScript dizisi yazdık ama otomatik tamamlama aynı zamanda Url yazmamızada da olanak sağlamaktadır. Otomatik tamamlama bir düz metin dönüşü beklemektedir.Yukarıdaki kod bu haliyle çalışabilir ancak başka seçenekleri de yok sayamayız.

“aoutoFill” önden yazma etkisi sunar(ok ile seçeneklerde dolaşırken okun üzerinde olduğu seçenek ile kutucuğu doldurur). “matchContains “ kelimeleri alt dizinleriyle eşleştirir. Burada ince ayar yapabilirsiniz bu yüzden seçenek listesinde dolaşırk hızlı olmalısınız.

Otomatik tamamlama eklentisi aynı zamanda kullanıcı bir seçenek seçtiğinde “result” olayınıda faaliyete geçirir. Kullanıcının seçtiği otomatik tamamlama seçeneğini ikinci bir parametre olarak olay işleyicimize gönderiririz.

```
$('#location')
  .autocomplete(cities)
  .result(function(event, selection) {
    alert(selection);
});
```

Kontroller

Konumuzun bu bölümünde müşterimizin en başta gelen korkusu formların kullanabilirliği. Şimdi eğlenceli bir kaç işlem yaparak bu sorunu giderebiliriz. JQuery ve jQuery UI ilkel HTML formlarını haraket ettirmek için çok iyi birer araçtırlar. Bir kez Web'in antik tarihin havasız sınırları geride bıraktığınızda yeni kontrolleri yaratmanın sınırının sadece hayal gücünüz olduğunu göreceksiniz. Söylemek istediğimiz web sitesi ile iletişime geçmenin tek yolu bir kutucuk değildir.

Tarih Seçici

Müşterimiz sitemize kullanıcıların ünlüleri nerede ve ne zaman gördüğünü söyleyebilcekleri bir “CelebSpotter” bölümü eklemek istiyor. Tabiki , bunu yapmak için kullanıcıların ünlüyü hangi tarihte gördüğü bilgisine ihtiyacımız var. Daha önceki deneyimlerimizden kullanıcıların boşluğa tarihi girerken istenen formatta girip girmediğinden emin olamadığımızı biliyoruz. Bu yüzden müşteri tarafından da onaylanmış sabit bir tablodan tarihi seçtirmek en akıllı çözüm olucaktır.

Eğer JavaScript 'te bir tarih seçici fonksiyonu yazmaya çalışırsanız muhtemelen tekrar bunu yapmak istemeyeceksiniz. Kontrol edilmesi oldukça zordur. Tarih seçicileri kontrol etmek oldukça önemlidir çünkü eğer yanlış yapılırlarsa sizi deliye çevirebilirler. Neyseki jQuery UI çok uygun ve tam özellikli bir tarih seçici kontrolü içermektedir. Bu da hata payımızı en aza indirger. Bu kontrole bir örnek olarak aşağıdaki şekli inceleyelim :

Tarih olarak güncel kullanılan girdi alanı ile başlayalım :

<input type="text" id="date" />

Eğer basit bir tarih seçici arıyorsanız jQuery 'nin bu tek satırlık kodu tam size göre demektir :

$("#date").datepicker();

Tarih seçici , giriş kutusu odağı aldığı zaman tetiklenir ve seçili olan tarihe gider. Kutucuk odak olmaktan çıktığında tarih seçici kaybolur. Tabi jQuery sayesinde oldukça güzel görünüyor ama peki bizden tarih seçici ile uğraşmaktan başka ne istenmişti ? (jQuery kullanıyor olmanız demek JavaScriptin uygulanabilir kısımlarını görmezden geleceğiniz anlamına gelmez.).

JQuery UI 'deki tarih seçici bileşeni paketlenmiş özelliktedir. Paketlenmiş ! Bütün olarak konumlandırılabilinir , herhangi bir tarih formatı seçilebilir , bir seferde birden fazla ay görüntülenebilir , şık bir tasarım elde edilebilir , yapılandırabilir tuşlara izin verilir , ctrl+ ok tuşları ile haraket ettirilir ve bir çok daha özelliğe izin verilir.

Yukarıdaki resimde olduğu gibi bir takvim hazırlamak için bu kodları kullanmalıyız :

```
$('#date').datepicker({
  showOn: 'button',
  buttonText: 'Choose a date',
  buttonImage: 'calendar.png',
```

```
  buttonImageOnly: true,
  numberOfMonths: 2,
  maxDate: '0d',
  minDate: '-1m -1w',
  showButtonPanel: true
});
```

“showon” takvim açıldığında seçim işlemine olanak sağlar. Erişilebilir seçenekler ; “focus ”(fare ile kutucuğu seçtiğinde) , “button” (kullanıcının takvimi açabilmesi için olan tuş) yada “both” (iki seçeniği de sağlayan) . Tuşda resim kullanmak için buttonImage fonksiyonunu tanımladık. Ayrıca buttonImageOnly fonsksiyonuna true atadık bu da standart buton yerine seçtiğimiz resimin görüntülenmesini sağlar.

“numberOfMonth” deyimine 2 sayısını atayarak takvimimizde kullanıcının yanyana 2 ayı görebilmesini sağladık. Eğer [3,3] deyimi kullanırsanız 3'e 3'lük bir ay karesi elde edersiniz.

“maxDate” ve “minDate” seçenekleri kullanıcının seçeceği tarih anlamına gelir. JavaScript tarih objesini tanımlayabilir veya alakalı tarihleri bir kodla belirleyip kullanabilirsiniz. Burada yaptığımız diğer seçenek genellikle daha kullanışlıdır. MaxDate'i 0 olarak ayarlamak bugünün tarihine denk gelmektedir. minDate'i -1m ve -1w olarak ayarlıyarak kullanıcının sadece 1 hafta ve 1 ay öncesini seçmesine izin verdik.

Tarih Seçici Özellikleri

Jquery UI kütüphanesi tarihlerle oynamanızı kolaylaştırmak için birkaç evrensel tarih seçici özelliği sunar.

“$.datepicker.setDefaults” metodu tarih seçicinin ayarlarının birleşiminden meydana gelen bir nesne olarak kabul edilir. Tanımladığınız bütün değişiklikler sayfadaki tüm tarih seçicilerine yanıt verebilir. Mesela bütün tarih seçiciler zaman olarak 2 ayı göstersin isterseniz :

```
$.datepicker.setDefaults({
  numberOfMonths: 2
});
```

Geri kalan özellik fonksiyonları tarih formatları olarak işlemek yada işlenmesine yardım etmek içindir. “$.datepicker.iso8601Week” fonksiyonu tarih alır ve 1 den 53 e kadar olan yıl içindeki haftaları döndürür. “$.datepicker.parseDate “ fonksiyonu verilen diziden bir tarih çıkarır ; bunu bir kod ve tarih formatına çevirmeniz gerekir(örneğin “aa-gg-yy”) ve JavaScript tarih objesine geri erişirsiniz. Son olarak

“$.datepicker.formatDate” bu işlemin tam tersini yapar. Tarih objesini belirlediğiniz formata göre formatlar.

Sürgü(Slider)

Müşterimiz siteyi ziyaret eden insanların ünlüleri kolaylıkla ve çabucak bulmasını istiyor. Müşteri aynı zamanda giren kullanıcıların , ünlülerin sahne fiyat aralıklarını seçerek bir liste görüntülemelerini istiyor. O zaman Jquery UI parçası olan slider'la tanışmanın tam zamanıdır.

İki tane seçim kutusu olan basit bir form ile başlayalım biri maksimum fiyat bir diğeri ise minimum fiyat için. Şimdi de bu kutuları kontrol etmek için Jquery'deki sürgüleri ekleyelim. Sonuç şekilde gösterilmiştir.

Basit iyileştirmeler göz atalım :

```
<div id="price-range">
 <form>
  <label for="min">Minimum Price:</label>
  <select id="min">
   <option value="0">0</option>
   <option value="10">10</option>
   <option value="20">20</option>
    ⋮
   <option value="80">80</option>
   <option value="90">90</option>
</select>
<br/>
<label for="max">Maximum Price:</label>
<select id="max">
   <option value="10">10</option>
   <option value="20">20</option>
   <option value="30">30</option>
```

```
    ⋮
   <option value="100">100</option>
  </select>
 </form>
</div>
```

Sayfa yüklendiğinde ilk olarak seçim kutularında o anki en büyük ve en küçük değerleri elde ederiz. Yaratılan yeni boş “div” elemanında “slider” metodunu çağırarak sürgümüzü başlatıyoruz :

```
var max = $('#max').val();
var min = $('#min').val();
var rangeSlider = $('<div></div>')
  .slider({
    min: 0,
    max: 100,
    step: 10,
    values: [min, max],
    range: true,
    animate: true,
    slide: function(e,ui) {
      $('#min')
        .val(ui.values[0]);
       $('#max')
        .val(ui.values[1]);
      showCelebs();
}
})
    .before('<h3>Drag the slider to filter by price:</h3>');
$('#price-range').after(rangeSlider).hide();
```

Kodumu biraz parçalayalım “min” ve “max” sürgümzün maksimum ve minumum değerleridir. “step” sürgümzün artışının adımlarıdır. “values” slider'a varsayılan değer atamak için kullanılır. İki değerin kodunu tanımladığımız için sürgümzün iki tane sürükleme çubuğu vardır.

“range” ve “animate” sürgümüzde birden fazla sürükleme çubuğu yaratırken faydalı seçeneklerdir. Bizim burada yaptığımız olayda “range” sürükleme çubuğunun belirlediğimiz alanını renklendirmek veya gölgelendirmek oldu. “animate” ise sürükleme çubuğunun tıkladığımız yere zıplaması yerine fare ile tutup bırakılarak hareket etmesini sağlar.

Son olarak “slide” metodu kullanıcı sürgünün üstüne tıkladığı zaman faaliyete geçen bir olay işleyicisi tanımlar. Bu olay işleyicisi bazı slider özelliklerine erişmenizi sağlayacak seçilebilir bir “ui” parametresidir. Biz seçim kutularının değerlerini ayarlamak için “values” özelliğini kullandık. Ayrıca seçtiğimiz aralıktaki fiyat değerlerine göre sergilenecek liste için “showCelebs” metodunu kullandık.

“slide” olayı ile aynı olan“change” olayıda işimize yarıyabilir. Fark olarak ,“slide” olayı sadece kullanıcı direkt olarak “slider” ile etkileşime geçtiğinde faaliyete geçerken “change” olayı sürgünün değerleri programlı olarak düzenlendiğinde de faaliyete geçer.

Jquery UI “slider” kısmı varsayılan olarak yatay bir sürgü yaratır. Eğer dikey bir sürgü isterseniz orientation: 'vertical' komutunu kullanmanız gerekir.

Sürgümüze başlık eklemek ve bunu sayfaya yansıtmak için “before” ve “after” metodunu kullandık ve orijinal seçim kutularını sakladık. Kodu şimdi denediğinizde daha organize bir slider elde ettiğinizi göreceksiniz.

Ünlüleri filtrelemek için “showCelebs” metodunu inceleyelim:

```
function showCelebs() {
  var min = $('#min').val();
  var max = $('#max').val();
  $('.data tr').each(function() {
    var price = parseInt($(this).find('td:last').text().substring(1));
    if (price >= min && price <= max) {
      $(this).fadeIn();
      } else {
        $(this).fadeOut();
}
});
}
```

Seçim kutularının değerlerini çıkarıp ünlü tablosundadaki satırların her birini döngüye aldık ve seçtiğimiz fiyat aralığına göre ünlüleri gösterip sakladık. Burada yaptığımız hileli kısım ise biraz JavaScript kodu kullanmak ve satırlardan fiyatı çıkarmak oldu. “substring(1)” i kullanarak ilk karakterden itibaren her şeyi çıkardık. “parseInt” ile de kodu sayılara çevirerek fiyatı yazabildik.

Aynı zamanda “showCelebs” komutuyla formdaki varsayılan değerleri “base” alarak listeyi önfiltrelemiş olduk.

Bu çalışma istenilen sonucu verdi ve kullanıcının ünlüleri fiyat aralığına göre

filtrelemesini sağladı. Sürgüler mükemmel UI görselleridir çünkü çok basit yapılardır yani kullanıcıya nasıl kullanılacağını anlatmanıza gerek kalmadan onlar zaten anlarlar. Muhtemelen artık sürgüler hakkında bilgi sahibisiniz.

Sürükle ve Bırak

Sürükle ve bırak kavramının temeli çok eskilere dayanmaktadır. Her zaman arka planda olmasına rağmen yazı kutucuklarının ve işaretleme kutucuklarının önüne geçememiştir. Eğer sürükle ve bırak özelliğini iyi bir şekilde işlerseniz ürününüzün özelliklerini oldukça arttırmış olursunuz.

Eğer bilgisayar kullanmaya yeni başlayan insanların bile bilmesi gereken bir görev varsa bu simgeleri çöp kutusuna sürüklemektir. Şimdi müşterimiz kontrol kutucuklarına tıklayıp sonra sil butonuna basmak yerine öğelerin tutup sürükleyerek silinmesini ve çöp kutusundan dumancıklar çıkmasını istiyor.
Şekil arzu edilenlerin resmedilmiş halidir. Kullanıcı bir resmi seçti ve onu çöp kutusu simgesinin üzerine bıraktı. Sürüklemek için herhangi bir nesneyi seçebilirsiniz.


```
<div class="trash">
  <img src="trash.png" alt="trash"/>
  <span id="trash-title">Drag images here to delete</span></div><div
id="photo-grid">
  <img src="../../images/fader_100.jpg" />
<img src="../../images/beau_100.jpg" /></div>
```

Sürükle ve bırak özelliğini çapraz sunucularda çalıştırmak biraz sorunlu olabilir.Bu yüzden güvenilir dostumuz jQuery'nin jQuery UI eklentisine göz atalım. Çapraz sunucularda sorunların üstesinden gelmemiz için iki yararlı fonkiyon barındırır ; "draggable" ve "droppable".

```
 $(document).ready(function() { $('#photo-grid > div').draggable({
      revert: 'invalid' }); $('.trash').droppable({
```

```
    activeClass: 'highlight', hoverClass: 'highlight-accept', drop: function(event, ui)
    {
    puffRemove($
  (ui.draggable)); } });});
```

```
function puffRemove(which) { // Implement the “puff-of-smoke” effect}
```

Bu bizim etkileşimlerimizin iskeleti. “puff” animasyonunu yapabilmemiz için birçok şey eklemeliyiz. Ama diğer taraftan sürükle ve bırak faaliyeti için her şeyi yapmış bulunmaktayız. Şimdi fonksiyonlarımıza daha yakından göz atalım

Sürükleme Elemanı(Draggable)

“draggable” etkileşimi fareyle seçtiğiniz simgenin sürüklenmesini sağlar. “$ (‘p’).draggable()” fonksiyonu sayfadaki tüm <p> etiketlerini sürüklenebilir yapar. Bu komutu örnekteki gibi daha da özelleştirebilirsiniz.

```
$('p').draggable({axis: 'y', containment: 'parent'});
```

“axis” deyimi nesnenin sadece x veya y eksenlerde sürüklenebilir olmasını sağlar. “containment” ise nesneyi parent , document ,window parametrelerine göre sınırlar . Ayrıca iki elemanı kodu belirtilerek, bir şebekeye sürükleyerek sınırlandırmak için “grid” seçeneğini kullanabilirsiniz.

Başka bir örneğe göz atalım:

```
 $('#dragIt').draggable({
    handle: 'p:first',
    opacity: 0.5,
    helper: 'clone'
});
```

Bu sonraki seçenekler için , en az bir adet <p> etiketi içeren div üzerindeki “dragIt” ile çalışıyoruz.”handle” seçeneğiyle ilk p elemanını kullanıcı elemanı sürükleyebilsin diye tasarlıyoruz. Ayrıca sürüklenebilir elemanları belirtmesi için “hepler” fonksiyonunu kullandık. Bu seçeneği “clone” a atamamız ve elemanların sürükleme sırasında aynı zamanda ilk yerindede görünmesi için kopyalamamız gerek.

Dikkat etmemiz gereken diğer bir seçenek ise “revert” seçeneğidir. Eğer buna “invalid” atarsanız (fotoğraf sürükleme işlemimizde yaptığımız gibi) , sürüklediğiniz elemanı hedef alan dışında bir yere bıraktığında ilk yerine geri dönecektir.
Ek olarak fark edebileceğiniz üç olay daha var: “start” , “stop” ve “drag” sürüklemeyi başlattığınız, durdurduğunuz ve sürükleme yaptığınız olaylar.

Bırakma Elemanı(Droppable)

Bırakma elemanları süreklenen simgelerin hedef noktalarıdır. Bırakma elemanları sürükleme elemanlarından biraz daha fazla seçeneğe sahiptirler. Yukarıda önemli olarak “activeclass” ve “hoverClass” ı kullandık. “activeClass” sürükleme elemanı sürüklenmeye başlandığında elemanımıza eklenir . Benzer olarak “hoverClass” sürüklenen eleman bırakma elemanının üzerindeyken elemanımıza eklenir.

Ayrıca “acept” seçeneği için sürüklenebilir elemanların bırakılacağı yerleri sınırlamak için bir seçici tanımlayabilirisiniz. Bu sizin birden fazla bırakma noktası belirtmenize olanak sağlar.

Sürüklenebilir elemanlar ile bırakma elemanlarının olayları oldukça benzerdir. “start”, “stop” ve “drag” yerine “over” ,“out” ve “drop” kullanılır. Fotoğraf örneğimizde sürüklenen elemanın ne zaman yok edilceğini anlamak için “drop” olayını kullandık.

“Puff” Etkisi

Sürükleme ve bırakma işlemlerinde birkaç satır kodla öğrendiklerimizle birlikte etkili ve güçlü bir kontrol sağladık. Ancak şu ana kadar kendimiz yazmak yerine var olan sürükleme ve bırakma fonksiyonlarını kullandık.

JQuery animasyon fonksiyonlarını kullanmak yerine kendi animasyon rulomuzu yaratacağız. Bunu yapmak için 5 adet aynı boyutlu PNG uzantılı resimi birbirlerinin üzerine yığıyoruz. Daha sonra resimleri sıra haline getirmek için yerlerini değiştireceğiz. Eğer “animate” kullansaydık , yerleri yavaş yavaş değiştirecek ve buda resimlerin çercevelere tam oturmamasına neden olucaktı.

```
// Implement the “puff-of-smoke” effect
var $this = $(which);
var image_width = 128;
var frame_count = 5;
```

“image_width” animasyon görüntüsünün pixellerinin genişliğini ayarlar.”frame_count” animasyondaki toplam çerceve sayısıdır.(resmin toplam uzunluğu image_width*frame_count olmalıdır.) Tabiki bunlar başka animasyon resimleri eklemek isterseniz onlar için de geçerli olur. Değiştirmek için ihtiyacınız olan sayıları komut dosyasının sağ üst köşesinde bulabiliriz.

Öyleyse bir çöp kutusu resmi ayarlayalım. Çöp kutumuz sildiğimiz elemanla aynı yerde ve aynı boyutta olmalı.

```
// Create container
var $puff = $('<div class="puff"></div>')
  .css({
    height: $this.outerHeight(),
    left: $this.offset().left,
    top: $this.offset().top,
    width: $this.outerWidth(),
    position: 'absolute',
    overflow: 'hidden'
})
.appendTo('body');
```

Çöp kutumuzun “overflow” una “hidden” atadığımız için resimin sadece bir çercevesi görünecektir. Resimi çöp kutusuna oturtmak için (sildiğimiz elemanla aynı boyutta olan) resimi boyutlandırmamız lazım. Boyutlandırmayı çöp kutusunun genişliğini resimin genişliğine bölerek buluruz.

```
var scale_factor = $this.outerWidth() / image_width;
var $image = $('<img class="puff" src="epuff.png"/>')
  .css({
    width: image_width * scale_factor,
     height: (frame_count * image_width) * scale_factor
})
.data('count', frame_count)
.appendTo($puff);
```

Ayrıca resmimize “data” olayı aracılığıyla “count “ özelliğini ekliyoruz. Bu geriye kalan gösterilcek çerçevelerin toplam sayısıdır. Tüm bunlar yapıldıktan sonra bırakılan asıl elemanı silmekle devam edelim :

```
// Remove the original element
$this.animate({
  opacity: 0
}, 'fast').remove();
```

Şimdi animasyonumuzu başlatalım. Başlatmamız için biraz JavaScript'e ihtiyacımız var. Animasyonumuzu oynatırken kendi çalışır ve kendinden devamlı döngüler ayarladık :

```
    setTimeout(animate, 75); 5
  } else {
    $image.parent().remove(); 6
  }
})();
```

```
  if (count) { 3
    var top = frame_count - count;
    var height = $image.height() / frame_count;
    $image.css({
      "top": - (top * height),
      'position': 'absolute'
    });
    $puff.css({
      'height': height
    })
    $image.data("count", count - 1); 4
```

```
    setTimeout(animate, 75); 5
  } else {
    $image.parent().remove(); 6
  }
})();
```

1) (function myFunction(){}) yapısıyla bir fonksiyon yaratıyoruz ve çalıştırıyoruz. Bu eski bir JavaScript yapısı olup , tam olarak anlamanızı gerektirmez. Bu durum için kendi kendine devam edebilen fonksiyonları yaratmamızı sağladığını bilmeniz yeterlidir.

2) Hangi çerçevenin bizim için uygun olacığına “count” verilerini kontrol ederek karar verdik.

3) Eğer hala gösterilcek çerçeve kaldıysa , görüntünün uzaklığını hesaplayıp doğru çerçeveye haraket ettiririz.

4) Çerçeve sayılarını azalltık ve böylece döngü bir dahaki sefere çalıştığında serilerdeki bir sonraki çerçeveyi gösterir.

5) Son olarak fonksiyonumuzu geri çağırım fonksiyonu olarak özelleştirecek “setTimeout”u çağırıyoruz .

6) Sayaç sıfıra ulaştığında ve animasyon sonlandığında puff çöp kutumuzu DOM dan çıkarıyoruz.

JQuery UI Sıralama(Sortable)

jQuery'nin bir başka kullanışlı özelliği “sortable” davranışıdır. Sortable olarak tanımladığınız elemanlar çocuklarına bırakma hedefi haline gelir(çocukların hepsi sürüklenebilir özelliğe sahiptir). Sonuç olarak uygun gördüğünüz çocukları yeniden yaratabilirsiniz. Çöp kutusu arasındaki elemanları sıralarken , aslında hiç bir şey sıralanmaz sıralama kullanıcıya bağlıdır.

Sıralanmanın yönetilmesi gereken durumlarda bu elemanların listeleri için mükemmel bir sonuç çıkarır. “move up the list” veya “move down the list” butonlarını kullanmak yerine “sortable” davranışını uygulayıp kullanıcın listelirini daha basit bir yolla yeniden sıralamasını sağlarız.

Sitemizde haftanın en popüler ünlüleri gösteren iki adet listemiz var. Bir tanesi A-ünlüler diğeri ise B-ünlüler. Bu müşterimize havalı bir hile göstermenin tam zamanı.
Haydi bu listeleri kullanıcı tarafından tekrar sıralanabilir hale getirelim. Yukarı aşağı hareket ettirebilelim ve hatta A/B statülerindeki elemanları bile kendi aralarında değiştirebilelim. Kullanıcı sıralama işleminden memnun kaldığı zaman “Accept” butonuna basıp listenin son halini onaylayabilsin.

Sıralanabilir(sortable) davranışını kullanmak için listeler en güzel hedeflerdir. Bir “div” elemanı kullanarak birkaç iyileştirme yapalım :

```
<ul id="a-list" class="connected">
<li><a href="#">Glendatronix</a></li>
<li><a href="#">Baron von Jovi</a></li>
⋮
</ul>
<ul id="b-list" class="connected">
<li><a href="#">Mr Speaker</a></li>
⋮
</ul>
```

Aynı “droppable” ve “dragabble” da yatıklarımız gibi bir elemanı sıralanabilir kılmak basittir.

```
$("#a-list, #b-list").sortable();
```

Bir eleman sıralanabilir hale gelirken kullanabileceğimiz sayısız metod , olay ve seçenek vardır. İlginç anları kontrol edebilmek için bunları birleştirebiliriz.

```
$("#a-list, #b-list").sortable({
  connectWith: '.connected',
  placeholder: 'ui-state-highlight',
  receive: function(event, ui) { adopt(this) },
   remove: function(event, ui) { orphan(this) }
}).disableSelection();
```

Olay işleyici olarak atadığımız iki metoda göz atalım :

```
function adopt(which) {
  if ($(which).hasClass('empty')) {
     $(which).removeClass('empty').find('.empty').remove();
}
}
function orphan(which) {
  console.log(which);
   if ($(which).children().length == 0) {
     $(which)
       .append($('<li class="empty">empty</li>'))
       .addClass('empty');
}
}
```

Bu metodlar listedeki son eleman da başka bir listeye taşınıp çıkarıldığında listeye "empty(boş) yazmasını sağlarlar. "receive" olayı ise sıralanabilir eleman bağlıtılı listeden bir elemana sahip olduğu zaman devreye girer. "adopt" metodumuzu çağırmak içinse boş yazısı bulunduğunda işlemeye başlar.

Listeden bir çocuğu çıkarmak "remove" olayını tetikler(aynı zamanda "orphan" fonksiyonuda denir). Bu metod sıralanacak bir çocuk olup olmadığını söyler. Bu metodda bir çocuk, <li> yazdığımız da ve "empty class" atadığımızda boş olmalıdır.

Diyaloglar

Eski günlerde sitelerin kullanıcıya meaj göndermesini gerektirecek çok fazla durum yoktu. Ancak son zamanlarda Ajax destekli web siteleri daha karmaşık hale geldi, iletilmesi gereken bilgiler oldukça büyüdü ; onay mesajları ,durum güncellemeleri , hata mesajları ve daha nicesi. Bunları kullanıcıyı rahatsız etmeden yapmak ise bir ustalık işi haline geldi.

Basit Modal Diyaloglar

“Modal diyaloglar” kullanıcıya mesaj gösteren ve kullanıcı devam etmek istediği sürece sürmesi gereken diyalog tipleridir. Aslında bir çeşit taciz sayılabirlir. İnsanlar bu aniden çıkan bildirimlerden hoşlanmadıkları için sadece gerekli oldukları zaman kullanılmalılardır. Müşterimiz siteye giren herkesin End User License Agreement (EULA) adında bir sözleşmeyi onaylamasıı istiyor. Şimdi bunu nasıl yapabileceğimize odaklanalım.

Modal diyalogların bir lightbox' a(arka plan karardıktan sonra daha aydınlık olarak çıkan bilgilendirme kutuları) çok benzer olduğunu fark etmiş olabilirisiniz. İstenenleri yapabilmemiz için gizli bir div içinde HTMl kodumuzu işleyelim. Göstermek istediğimiz zamanlar ise içeriklerin bir kopyasını alıp arka planı soldurarak gösterebiliriz. Aynı lightbox elemanını kullanarak birden fazla diyolog gösterebiliriz.

```
<div id="overlay">
  <div id="blanket"></div>
</div>
<!-- the dialog contents -->
  <div id="eula" class="dialog">
     <h4>End User License Agreement</h4>
⋮
<div class="buttons">
   <a href="#" class="ok">Agree</a>
   <a href="#" class="cancel">Disagree</a>
  </div>
</div>
```

Diyaloğumuzun alt kısmına bir çift buton eklediğimizi göreceksiniz. Bunları kullanmamızın sebebi ise kullanıcı etkileşiminin aşamalarını görmektir. Bu gerçekten basit bir HTML kalıbıdır ve tahmin edebileceğiniz gibi diyaloğumuzun görünüşündeki asıl rolü CSS oynamaktadır. Tüm ekranı karartıp daha sonra modal diyaloğumuzu göstermek istiyoruz :

```
  #overlay {
display:none;
top: 0;
right: 0;
bottom: 0;
left: 0;
margin-right: auto;
margin-left: auto;
```

```
position: fixed;
width: 100%;
z-index: 100;
}
  #blanket {
background-color: #000000;
top: 0;
bottom: 0;
left: 0;
display: block;
opacity: 0.8;
position: absolute;
width: 100%;
}
  .dialog {
display: none;
margin: 100px auto;
position: relative;
width: 500px;
padding: 40px;
background: white;
-moz-border-radius: 10px;
}
```

Şimdi diyaloğumuzu ekrana getirelim. HTML diyoloğunu alacak bir “openDialog” fonksiyonu yaratacağız. Bunu örtülü yapıya taşıyıp göstereceğiz. Taşıma işlemi “clone” sayesinde gerçekleştirilir. Bu da bize o anki jQuery seçiminin bir kopyasını yaratır ve orijinal alana bırakır. Diyaloğu kapattığımız zaman içerikleri kaldıracağız böyelece sözleşmeyi bir kere ekrana getirmiş olacağız :

```
function openDialog(selector) {
$(selector)
.clone()
.show()
.appendTo('#overlay')
.parent()
.fadeIn('fast');
}
```

Fonksiyona bir davranış eklediğimiz için ne zaman diyaloğu açmaya ihtiyaç duysak , onu çağırabilir ve göstermek istediğimiz elemanın seçicisini yazabiliriz.

```
$("#eulaOpen").click(function() {
  openDialog("#eula");
});
```

İkinci kısım diyalog kapandığında her şeyi eski durumuna getirir . Bu örtüyü bularak , kararmayı kaldırıp diyalog içeriklerini kaldırarak gerçekleşir. :

```
function closeDialog(selector) {
$(selector)
  .parents("#overlay")
  .fadeOut('fast', function() {
    $(this)
      .find(".dialog")
        .remove();
    });
}
```

“closeDialog” fonksiyonunu o anki diyalog arasından çağırmamız gerekir. Ancak kapattığımız zaman , diyalog üzerindeki butonlar başka etkilere sahip olmalılar. HTML diyaloglarına ekstradan buton ekleyerek ve bunları dökümanlanmaya hazır kısımlara ekleyerek olay işleyicilerinin herhangi birini çağırabilir ve ihtiyacınız olduğunda işleyebilirsiniz :

```
$('#eula')
  .find('.ok, .cancel')
  .live('click', function() {
  closeDialog(this);
})
.end()
.find('.ok')
.live('click', function() {
   // Clicked Agree!
})
.end()
.find('.cancel')
.live('click', function() {
   // Clicked disagree!
});
```

Bu kod parçasındaki en önemli kısım “live” komutuduır. DOM düğümlerini kopyalamak için “clone” kullandığımız zaman onun olay işleyicisi aşamalarda kaybolur ama “live” düğümleri silmemize ve çoğaltmamıza bağlı kalmadan her şeyi yerinde tutar.

Buton olaylarını işlememiz için bu basit ancak etkili bir yoldur. Burada kullandığımız metodun avantajı kısmi ihtiyaçlarımızı hedefleyip vurgulamamızı sağlamasıdır. Ama elle butonları yaratmanız ve ilişkili olayları işlemeniz eğer çok karmaşık diyaloglarınız varsa gerçekten yıpratıcı bir hale gelecektir.

Bölüm :8 TABLOLAR

HTML listeleri yeni internetin duyulmamış kahramanlarıdır. Tablolar ise kötü çocuk olarak bu durumu sona erdirmişlerdir. Ancak tabloların kendilerini suçlamamak gerkir. Çapraz sunucularda kullanılması problemli olduğu için yakın zamana kadar hep suçlanmıştır ama CSS çağı ile birlikte çizelge halindeki verileri büyüleyici bir şekilde tablolaştırabiliriz.

Sitemizde sergilememiz gereken bir veri topluluğu var. Bu bilgilerimizi de kolayca düzenlemek ve sıralama işlemini yapmadan sergilememiz için tablolar bize yardım edecekler.

Sabit Tablo Başlıkları

Tablolarda başlık satırının önemi oldukça büyüktür. Başlık satırı olmadan tablonuz anlamsız verilerle dolu gibi görünecektir. Ancak tablonuzda çok fazla satır varsa başlık satırının önemi azalacaktır çünkü kullanıcı bilgileri görmek için aşağı doğru kaydıkça başlığı unutacaktır. Verilerin sayfalanması genellikle problem oluşturur. Bütün verilerinizin aynı anda sayfada görünmesini istiyorsanız başka bir seçenek düşünmek zorundasınız.

Tablonuzun güzel görünmesi ve sütünların neyle alakalı olduğunu söylemeniz için en etkili yol balşığı tablonun üstünde sabit tutmaktır. Kullanıcı tabloyu görmek için aşağı doğru indikçe başlıkta sizinle gelir. Bunu örneğimizde görebilirsiniz :

Id	Name	Occupation	Approx. Location	Price
007F	Kellie Kelly	Singer	Omaha	$11.95
8A05	Darth Fader	DJ	London	$19.95

Eğer tablonuz sayfadaki tek elemansa “position : fixed” “thead” elemanını sabitlemek için kullanılabilir. Fakat “position : fixed” içerdiği elemanı konumlandırmak yerine sadece görüntü kapısını ile ilgili elemanları konumlandırır. Yani , içinde başka elemanlar içeren tablolar için , jQuery'e dönmemiz gerekir.

Şimdi bu etkiyi nasıl gerçekleştirrebilceğimize bakalım :

```
<table id="celebs">
  <thead>
    <tr>
     <th>Id</th>
     <th>Name</th>
     <th>Occupation</th>
     <th>Approx. Location</th>
     <th>Price</th>
    </tr>
  </thead>
<tbody>
⋮
  </tbody>
</table>
```

“thead” etrafında gezinmek biraz zordur. Bazı sunucularda zorluk çıkarırken bazılarında sorunlu çalışabilir. Bu yüzden bir hile ekleyelim sırasız listenin formundaki “thead” içeriklerini kopyalayalım , böylece gerçek bir “thead” gibi görünür. Kullanıcı sayfayı kaydırırken ekranda gezinimesi için “position : absolute” kullanıyoruz.

Kodumuza “fixHeader” kullanarak TABLO göselini ekleyerek başlayalım. Bu metod sayfadaki tabloyu işaret eden bir gösterici bekler. :

```
var TABLE = {};
TABLE.fixHeader = function(table) {
$(table).each(function() {
var $table = $(this);
var $thead = $table.find('thead');
var $ths = $thead.find('th');
$table.data('top', $thead.offset().top);
$table.data('left', $thead.offset().left);
$table.data('bottom', $table.data('top') + $table.height() -$thead.height());
```

:

İlk olarak görselimizin içeriğini tutmak için bir yaklaşım sergiledik. Daha sonra seçici ile eşleşen “table”ları seçip metodumuza yazdık. “each” ile bir döngüye soktuk ve bilgileri depoladık(table ve thread ve th). Son olarak $thead'in soldaki ve baştaki uzaklıkları kaydedip başlığın en alt noktayı geçmesini önledik.
“Faux” başlığımızı yarttık ve tablonun başlığının içeriklerini kopyaladık :

```
var $list = $('<ul class="faux-head"></ul>');
$ths.each(function(i) {
  _th = $(this);
  $list.append($("<li></li>")
   .addClass(_th.attr("class"))
    .html(_th.html())
    .width(_th.width())
     .click(function() {
     _th.click()
})
    ).hide().css({left: _table.left});
});
$('body').append($list);
```

“$ths” de toplanan gerçek th elemanları ile “each” kullandık.Esas elemanların class,HTML içeriklerini , genişliğini ve “click” olay işleyicisini kopyaladık . Bunlardan bazıları örneğimizde gereksizdi ancak pratik için iyi olacağını düşünüyoruz.

“thead” taklidimizle kaydırma olayını işlemeliyiz :

```
$(window).scroll(function() {
  setTimeout(function() {
    if ($table.data('top') < $(document).scrollTop() && $(document).scrollTop()
<$table.data('bottom')) {
$list
  .show()
  .stop()
  .animate({
   top: $(document).scrollTop(),
  opacity: 1
});
} else {
   $list.fadeOut(function() {
    $(this).css({top: $table.data('top')});
```

```
});
}
}, 100);
});
```

Kaydırma(scroll) için varsayılan zamanı 100 milisaniye olarak ayarladık. Bu çok kısa bir zaman ama kullanıcı kaydırma işlemini gerçekleştirirken animasyonu önlemek için de yeterli bir zamandır.

İşte oldu. “TABLE.fixHeader(“#celebs”)” çağırabilirisiniz ve sayfayı aşağı kaydırdığınızda yeni “thead” sizi takip edecektir.

Başlığı Tekrarlama

Başlığın kaybolması çözümüne bir başka yaklaşım ise başlığı belirli aralıklarla tekrar etmektir. Eğer veriler yazdırılıcaksa oldukça kullanışlı bir çözümdür. Amaç ; tablonun ilk satırını alıp her aralığa (10 satırda bir diyelim) yapıştırmaktır. Şekilde yapılmak istenen gösterilmektedir :

Id	Name	Occupation	Approx. Location	Price
203A	Johny Stardust (bio)	Front-man	Los Angeles	$39.95
6636	Glendatronix	Keytarist	London	$39.95
141B	Beau Dandy (pic)	Singer	New York	$39.95
2031	Mo' Fat	Producer	New York	$19.95
007F	Kellie Kelly	Singer	Omaha	$11.95
8A05	Darth Fader	DJ	London	$19.95
203A	Johny Stardust (bio)	Front-man	Los Angeles	$39.95
6636	Glendatronix	Keytarist	London	$39.95
141B	Beau Dandy (pic)	Singer	New York	$39.95
2031	Mo' Fat	Producer	New York	$19.95
Id	Name	Occupation	Approx. Location	Price
007F	Kellie Kelly	Singer	Omaha	$11.95

Başlık satırını kopyalayıp onu biryerlere koymak size biraz eski iş gibi gelebilir şu an için. Peki her 10 satıra nasıl düzenli bir şekilde bunu ekleyeceğiz? Her satırı döngüye alıp indekslerinimi inceleyeceğiz? Evet , belki yapabiliriz ancak yine Jquery filtreleri ile bize yardıma geliyor :

```
$('#celebs')
  .find('tr:first')
  .clone()
  .insertAfter('#celebs tr:nth-child(10n)');
```

“:nth-child” seçicisi aynı zamanda başka değerleride kabul edebilir ve böylece aynı anda birden fazla satır seçebilir. Eğer yazı şeklinde “even” ve “odd” yazarsanız bütün tek veya çift sayıdaki satırları seçmiş olursunuz. En güzeli de seçim parametresi olarak bir denklem bile yazabilirsiniz.

“n” harfini tekrarlama için kullanabilirsiniz “nth-child(10n)” her 10. satırı seçicektir. “10n” ifadesinin yanına artı veya eksi ekleyip denklem yaratabilir ve buna göre seçim yapabilirsiniz. Örneğin her 10 satırdan sonraki üçüncü satırları seçmek istiyorsanız “nth-child(10n+3)” ifadesini kullanabilirsiniz.

Bunların hepsi olağanüstü bir şekilde çalışır ancak bir sorun var eğer son satırımız denklemle eşleşiyorsa başlık satırı son satır olarak görüntülenecek ve biraz saçma bir görüntüye yol açacaktır. Şimdi basit bir görsel nesne ile yolumuza devam edelim :

```
var TABLE = {};
TABLE.repeatHeader = function(table, every) {
  $(table).each(function() {
   var $this = $(this);
   var rowsLen = $this.find('tr:not(:first)').length;
  $(this).find('tr:first')
    .clone()
    .insertAfter($this.find('tr:nth-child(' + every + 'n)'));
 if ((rowsLen) % every === 0) {
    $this.find('tr:last').remove();
}
});
}
```

Tablomuz için seçici bir dizi ve başlıklar arasında kaç satır bırakacağımızı belirten sayıyı kabul eden bir fonksiyon tanımladık . Daha sonra aynı sayfada başka tablolar içinde aynı fonksiyonu kullanabilmemiz için seçicimizin bulduğu elemanları döngüye aldık. Her bir eleman için $(this) elemanına bir kısayol atadık ve tablodaki satır sayılarını elde ettik. Satır sayılarını önceden saklamak bizim için önemli çünkü tablomuzun başlığını tekrar ettikçe satır sayımız artıcak.
İlk satırımızı kopyaladık ve her “n”. satıra yapıştırdık. Bu “find” olayı ile yeniden yazılmayı gerektirebilir böylece seçici tarafından eşleşen tüm tablolarla çalışabilir. Son kısım , başlığımızı tabloya son satır olarak ekleyip eklemediğimize karar vermek için bir parça matematik içermekte. Eğer tablodaki toplam satır sayısı tekrar eden rakamımıza tam olarak bölünüyorsa son satırı çıkarmamız gerekir.

Veri Sistemleri

Sayfanızın son halini müşterinize gösterdiğinizde değişikliklerin mükemmel göründüğünü söyledi. Ancak herzaman ki gibi ek olarak istedikleri var. “Bunlar güzel ancak sanatçıların menejerlerinide aynı veri tabanında farklı hücrelerde piyasaya sunsak nasıl olur?” diyor. Bunu yapıp yapamıcağınızı sorduğunda tabikide diyorsunuz ve müşteriniz eklentileri görmek için ne zaman gelmesi gerektiğini sorup size güvendiğini söyleyip sırtınıza vurarak oradan ayrılıyor.

Veri sistemlerinin ne olduğu hakkında kesin bir tanım yoktur ama yaygın özellikler sıralama , filtreleme , arama , sayfalandırma , sütün ve satır düzenlemeleridir. İlk olarak sayfalandırmaya göz atalım.

Sayfa Numaralandırma

Çok uzun tablolar siteye girmek için çok zahmetli olabilirler. Ekran değerlerle doldukça gözünüz korkabilir.Numaralandırma ekleyerek küçük veri kümeleri elde edebilir ve onları bu şekilde gösterebiliriz. Bu sayede kullanıcılar aradıklarını daha rahat bulabilir veya listelere daha kolay odaklanabilirsiniz.

Sayfa numaralandırma genellikle sunucu tarafından işlenir. Kullanıcı bir sayfa numarası girer ve sunucu bu sayfayı kullanıcıya döndürür. Eğer çok sayıda veri ile uğraşıyorsanız bu çok mantıklı olabilir : 10,000 tablo satırını sunucuya yüklemek çok can sıkıcı olabilir.Fakat küçük kümeler için her şeyi sayfaya bir kerede yüklemek daha mantıklıdır.jQuery sayfalandırma animasyonumuz şekilde görülmektedir :

Tablo sayfalandırma görselimizde yaptığımız şey “NEXT” ve “PREVIOUS” düğmeleri yaratarak sayfalar arası geçiş sağlamaktır. HTML yapısı burada oldukça önemlidir :

```
<div class="table-wrapper">
  <div class="wrapper-paging">
   <ul>
     <li><a class="paging-back">&lt;</a></li>
     <li><a class="paging-this"><b>0</b> of <span>x</span></a></li>
```

```
    <li><a class="paging-next">&gt;</a></li>
   </ul>
  </div>
  <div class="wrapper-panel">
   <table id="celebs">
⋮
   </table>
  </div>
</div>
```

Uygun gördüğünüz kontrollleri kendi tarzına göre stilize edebilirisiniz ama CSS'deki kontrol kutularını saklamak iyi bir fikirdir. Daha sonra sayfa yüklenirken bu kontrolleri jQuery ile gösterin. Bu JavaScript uzantısı olmayan kişilerin gereksiz kontrolleri görmesini engelleyecektir.

Görselimizin iskeleti bunun gibi görünür :

```
var TABLE = {};
TABLE.paginate = function(table, pageLength) {
// 1. Set up paging information
// 2. Set up the navigation controls
// 3. Show initial rows
  pagination = function (direction) {
  reveal = function (current) {
// 5. Reveal the correct rows
};
// 4. Move previous and next
}
};
```

Daha işlenecek çok şey varmış gibi görünebilir ancak emin olun ki geri kalanlar zaten bildiğiniz şeyler. İlk olarak sayfalandırmak istediğimiz tabloları ve satırları ele alalım ve kaç sayfa olucağını hesaplamak için bir kaç işlem yapalım :

```
// 1. Set up paging information
var $table = $(table);
var $rows = $table.find('tbody > tr');
var numPages = Math.ceil($rows.length / pageLength) - 1;
var current = 0;
```

Şimdi yerleştirme kontrollerimizi devreye sokalım. Bu yapımızın önemli bir kısmıdır çünkü DOM' a doğru “table” seçimimizden div'e tırmanırak kontrolleri bulabilir ve daha sonra navigasyon bölümüne geri dönebilirsiniz. Bu sizin aynı kozu farklı

tablolar içinde uyguluyabilminizi sağlar.

```
// 2. Set up the navigation controls
var $nav = $table
   .parents('.table-wrapper')
   .find('.wrapper-paging ul');
var $back = $nav.find('li:first-child a');
var $next = $nav.find('li:last-child a');
```

Daha sonra gösterge kutucuklarına o anki sayfanın numarasnı ve sayfaların toplam uzunluğunu yerleştirdik. “Previous” ve “Next” butonları için olay işleyicileri ekledik. Butonlara tıklandığında haraket etmek istediğimiz sayfa yönünde “pagination” fonksiyonunu çağırırız :

```
$nav.find('a.paging-this b').text(current + 1);
$nav.find('a.paging-this span').text(numPages + 1);
$back
   .addClass('paging-disabled')
   .click(function() {
      pagination('<');
});
$next.click(function() {
    pagination('>');
});
```

Son kısımda da başlangıçta da kullanıcının kaç sayfa göreceğini sınırlandrırırz. Bunu yapmanın en kolay yolu tablonun bütün satırlarını saklamak ve sadece ilgilendiğimiz aralıktaki satırları göstermektir. Peki bu aralığı jQuery ile nasıl seçebiliriz? “:lt()” ve “:gt()” filtrelerini kullanabiliriz ama gösterme zamanı geldiğinde 10dan 20ye satırlar demek seçicinin kafasını karıştırır. Neyse ki “slice” olayı sayesinde başlangıç indeksini ve bitiş indeksini alarak o aralıktaki nesneleri döndürebiliriz :

```
// 3. Show initial rows
$rows
  .hide()
  .slice(0, pageLength)
  .show();
```

Şimdi herşey yolunda gibi görünüyor.Navigasyon kontrollerimiz verileri , bulunduğumuz ve toplam sayfayı doğru bir şekilde gösteriyor. Ancak sayfalandırma butonlarımızın henüz bir faaliyeti yok. Şimdi bulunduğumuz sayfadan haraket etmek için birkaç bişey ekleyelim :

```
// 4. Move previous and next
if (direction == "<") { // previous
   if (current > 1) {
     reveal(current -= 1);
}
else if (current == 1) {
   reveal(current -= 1);
   $back.addClass("paging-disabled");
}
} else { // next
  if (current < numPages - 1) {
    reveal(current += 1);
 }
else if (current == numPages - 1) {
   reveal(current += 1);
   $next.addClass("paging-disabled");
}
}
```

“current” değişkenini güncelleyelim ama tablonun kendisini değil. Bunu yapmak için içsel bir fonksiyon çağırıyoruz : reveal.

```
// 5. Reveal the correct rows
var reveal = function (current) {
  $back.removeClass("paging-disabled");
  $next.removeClass("paging-disabled");
  $rows
    .hide()
     .slice(current * pageLength, current * pageLength + pageLength)
     .show();
$nav.find("a.paging-this b").text(current + 1);
}
```

“reveal” ilk olarak görünmüyen sınıfları temizler böylece butonlarımız görünmezlik durumuna sıkışıp kalmaktan kurtarır. Daha sonra yine doğru satırları seçmek için “slice” metodunu kullanıyoruz.

Bir Satırı Düzenlemek

Şuana kadar tek elemanda satır içi işlemeler yaptık ama eğer tüm tabloyu düzenlenebilir yapmak isterseniz ne yapmanız gerekir ? Veri tiplerine her satırın sonuna düzenle butonu ekleyerek düzünleyici fonksiyonelliği kazandıralım.

Çizelge halindeki verilerinizi içeren hücreler bilgileri saklamak için harika bir fırsat sağlar . Çalışma alanlarını gruplandırmak için “vir” satır elemanını şu ana kadar görmedik. Şimdi nasıl kullanılacağını görelim :

```
<form action="null" method="post">
 <table id="celebs">
  <thead>
   <tr>
     <th>ID</th>
     <th>Name</th>
     <th>Occupation</th>
     <th>Approx. Location</th>
     <th>Price</th>
   </tr>
</thead>
<tbody>
 <tr>
    <td>203A</td>
     <td>Johny Stardust</td>
     <td>Front-man</td>
     <td>Los Angeles</td>
     <td>$39.95</td>
  </tr>
⋮
```

Çok normal görünüyor değil mi? Olmalıda. Ne kadar çok özelliği düzenlenebilir yaparsanız kullanıcılarınız size memnun bir şekilde geri döneceklerdir.Başka bir sorunumuz daha var tablonun HTML düzenleme kontrollerinden hiç birini dahil etmeden jQuery sayesinde kolayca satır ekleyebiliriz.

Tablomuza ilk olarak değer atamak için kurulum metodunu ile başlıyoruz ve gerekli butonları ekleyeliyoruz :

```
TABLE.formwork = function(table) {
var $tables = $(table);
 $tables.each(function () {
   var _table = $(this);
   _table
     .find('thead tr')
      .append($('<th class="edit"> </th>'));
    _table
      .find('tbody tr')
      .append($('<td class="edit"><input type="button" value="Edit"/></td>'))
});
$tables.find('.edit :button').live('click', function(e) {
  TABLE.editable(this);
  e.preventDefault();
});
}
```

TABLE.formwork metodumuz yazdığımız seçiciyi arar “thead” ve “tbody” 'deki satırlara düzenle (edit) hücresi ekler. “thead” hücresi boştur ama tablonun sütünlarında bir kayıp yaşanmaz. “Tbody” eklentisi de satırı düzenleme moduna geçirir. Butonumuzun birinci dereceden “click” olayı ile bağlantılı olduğunu bilmemize rağmen , “live” buradaki en hayati olaydır. Bunu kullanmamızın nedeni ise butonlarımızı nasıl haraket ettirdiğimizin fark etmemesi , onları kaldırabilmemizi garantilemektir.

Burada hatırlamamız gereken bir başka nokta ise “live” metodunu “each” döngüsünün dışında kullanışımız. Şimdi güzel bir kısım geliyor. Butonlarımızı iki durumda çalışır yapmak istiyoruz(düzenle ve kaydet modu) . Bunu nasıl uyguladığımıza bakalım :

```
TABLE.editable = function (button) {
var $button = $(button);
var $row = $button.parents('tbody tr');
var $cells = $row.children('td').not('.edit');
if ($row.data('flag')) { // in edit mode, move back to table
// cell methods
$row.data('flag', false);
$button.text('Edit');
}
else { // in table mode, move to edit mode
// cell methods
$row.data('flag', true);
```

```
$button.text('Save');
}
};
```

“TABLE.formwork”la gelen canlı “click” olayı bizim düzenlemek istediğimiz satır için düğmeye yazılır. $button nesnesi üzerinde “parent”ları kullanarak satır buluyoruz ve “$row” üzerindeki çocuklar ve diğer hücreleri buluyoruz.

Kodumuzu daha basit anlatabilmek için tersten bakalım. Yapmamız gerekn en basit kısım çıkış(exit) moduna bakalım :

```
$cells.each(function () {
  var _cell = $(this);
  _cell.html(_cell.find('input').val());
});
```

Şimdide düzenleme modunu değiştiren kodumuzu inceleyelim .

```
$cells.each(function () {
  var _cell = $(this);
  _cell.data('text',_cell.html())
     .html('');
var $input = $('<input type="text" />')
  .val(_cell.data('text'))
  .width(_cell.width() - 16);
_cell.append($input);
});
```

Hücreleri düzenlenebilir yapmak bir parça zordur. Hücrelere girdi ekleyebilmemiz için geçerli içeriklerini boşaltacağız veya yerleşimi bozacağız. Hücrelerin içindekileri boşaltmadan önce daha sonra kullanabilmek için geçerli içerikleri depolayacağız.

Daha sonra oluşturduğumuz bu girdileri birleştireceğiz ve bunları DOM 'a ekleyeceğiz. Giriş genişliğini herhangi bir düzen sorunu yaşanmaması için hücrenin genişliğinden daha küçük ayarlayın ve sadece giriş elemanı varsayılan değer olarak atayın.

EK: REFERANS METARYELLERİ

Jquery bir çok duruma uygulanabilirliği , esnekliği ile gündemdedir. Bu bir çok seçeneğe imkan sağlamaktadır. Hepsini ezberlemenize gerek yok internet üzerinden bu seçeneklerin dökümüne ulaşabilir ve buradan kontrol edebilirsiniz. Bu jQuery

veya eklenti havuzuna dalmaktan korkmayın.

$.ajax Seçeneği

Ajax fonksiyonlarının en güçlü dizisi tek bir metod altında toplanmıştır “$.ajax”. Bu metod bir çok seçeneği içerir .Bu seçeneklerden bazılarına göz atalım ama unutmayın ki jQuery'de bunun gibi bir çok seçenek bulunmaktadır.

Bayraklar(Flags)

“flaggable” seçeneği parametre olarak boolean değerler kabul eder (true ve false). Çoğu zaman varsayılan değerler yeterli olacaktır ama isteklerinize göre şekillendirmeniz kolaydır.

“async”

“async” varsayılan olarak “true” değerine sahiptir. Eğer uyumlu bir hale (synchronous) getirmek istiyorsanız “false” atamanız yeterli olucaktır.

“cache”

Ön belleğe alınan veriler Ajax isteklerini gerçekleştirirken sorun olabilirler. Eğer tarayıcınız eski istekleri depolıyorsa son verileri yakalayamıyacaktır. Ön belleğe alınmayı önlemek için “cache” seçeneğini “false.Script” olarak ayarlayın.

“global”

“global” olay işleyicimizi kullanarak Ajax isteklerinden gelen olayları işleyebildiğimizi gördük. Aynı zamanda global parametresi için “false” tanımlarsanız evrensel olay işleyicilerinden gelen olayları durdurabilirsiniz.

“ifModified”

“ifmodified” bayrağını kullanarak, bir isteği yalnızca bellek son istekten bu yana güncellenmiş ise “başarılı” olmaya zorlayabilirsiniz. Yeni verileri görüntülemek için işlem yapmak istiyorsanız bu sizin işinizi görecektir.

“processData”

Eğer isteğinizle göndereceğiniz verileriniz varsa Jquery bunları işleyecek ve bir sorgu koduna dönüştürecektir. Eğer bu istenmeyen bir durumsa “processData” bayrağına “false” atıyabilirsiniz.

Ayarlar(Sets)

Ajax seçeneklerinin çoğu aç/kapa değerinden daha fazlasını tanımlamanıza olanak sağlar. Çoğunlukla kodu veya nesneleri yazabilirsiniz.

“contentType”

Bu sizin isteğiniz için bir içerik tipi ayarlamanıza olanak sağlar. Çoğu durum için ideal olan varsayılan değeri application/x-www-form-urlencoded 'dir.

“context”

Ajax geri çağırımları çalıştırılırken içerikleri genellikle kullanışsız olan “window” nesnesidir. Bu seçeneği çalıştırarak bir geri çağırımdaki “this” özelliği JavaScript objesi veya jQuery elemanı gibi daha kullanışlı hale gelebilir.

“data”

Sunucuya veri yazmak için “data” ya bir katar , JavaScript kodu veya bir nesne yazarsınız.

“dataType”

Sunucuya gönderdiğiniz veriye contentType'ına içerik tipini yazarken “dataType” parametre olarak sunucunuzdan iletilmesini beklediğiiniz veri titpini kabul eder. Varsayılan olarak “everything (*/*)” tanımlıdır ancak xml, html, script, json, veya text olarak özelleştirebilirsiniz.

“jsonp”

JSONP çağrısı yaptığınız zaman geri çağırım fonksiyonunun adının “callback” olması beklenir. Eğer kullandığınız servis başka bir isim gerektiriyorsa jsonp ayarları ile karalaştırabilirsiniz.

“password”

Kimlik doğrulama kullanıcı adı ve şifre gerektirir. “Password” ayarları ile bu atamaları yapabiliriz.

“scriptCharset”

Bu özellik script etiket özelliklerinden bir karakter kümesi tanımlamanıza olanak sağlar.

“timeout”

Bu özellik ajax ayarlarından rakam kabul eden tek parametre olmanın gururunu taşımaktadır. Bir ajax isteğini iptal etmeden kaç milisaniye geçmesi gerektiğini tanımlar.

“type”

Ajax istekleri için en önemli ayarlardan bir tanesi HTTP isteği tipi olan “type” özelliğidir. (GET, POST ,PUT ,veya DELETE)

“url”

Diğer bir önemli özellik ise çağrı yapmak istediğiniz adresin konumunu tanımlayan url 'dir.

“username”

Kimlik doğrulmanın gerektirdiği kullanıcı adını tanımlamamızı sağlar.

Geri Çağırım ve Fonksiyonlar

“complete” işleyici “$.ajax” çağrısının, başarılı veya başarısız olmasına bakılmaksızın , devreye girer. “eror”işleyici ise çağrı başarısız olduğu zaman çalışır.

“beforeSend” fonksiyonu gönderilen mesaj faaliyete geçirilmeden çalışır ve isterseniz yapılan isteği ayarlamanızı sağlar.

“datafilter” fonksiyonu başarılı bir şekilde geri dönüş yapan isteklerden sonra çağırılır.

Olaylar

Şu ana kadar zamanımızın çoğunu olayları inceliyerek geçirdik. Jquery, çapraz tarayıcı olay tutucularını W3C standartlarını kullanarak basitleştirir. Normalleştirilmiş olay objesi kullanışlı özellikler ve fonksiyonlar bütünüdür. Kitabımızda bunlardan bazılarını zaten kullandık.

Olay Özellikleri

Bir olay tutucu çağrıldığı zaman olayın kaynağı olan DOM elemanını elde edersiniz. Bu aynı zamanda sinir bozucu DOM seçicilerinden mutlu jQuery arazimize geri

kaçmak içinde bir yoldur. Ek olarak eğer olay bir fare hareketiyse bu elemanı releatedTarget özelliği ile bir önceki hedeften bulabilirsiniz. Ve eğer olay hiyerarşik olarak ilerliyorsa curretnTarget sizi o anki hedef olarak bilgilendirir.

Son ama oldukça kullanışlı bir başka özellik is timeStamp'tir. Bu size olayın ortaya çıktığı kesin zamanı verir ve zaman tabanlı etkileri işlememizde oldukça yaygın olarak kullanılır. Örneğin eğer özel bir 3-tıklama olayı yaratmak istiyorsanız verilen zamanda 3tıklama olayı gerçekleşmişmi diye olayların timestamps'lerine danışabilirsiniz.

Olay Metodları

Kullanabiliceğimiz birçok olay metodunu şu ana kadar gördük. “preventDefault” ve “stopPropagation” tarayıcı tarafından olayların nasıl sıçrayıp nasıl durdurulduğunu kontrol ettik. Başka metodlara örnek verecek olursak “stopImmediatePropagation” metodu , “stopPropagation” gibi ebevyn tutucuları durdurmak yerine çalışan tüm olay tutucularını durdurur.

“isDefaultPrevent” , “isPropagationStopped”, ve “isImmediatePropagationStopped” metodları boolean bir değer olarak “false” döndürürler.

DIY Olay Objeleri

Olaylar hakkında öğrenmekte hoşlanacağınız son bir bilgi.Kendi olay nesnenizi yaratabilir ve onalara direkt olarak tutucu ekleyebilirsiniz. Kodu kontrol edin :

```
// Regular event binding
$('p').bind('click', function(e) {
  $(this).text(e.pageX);
});
// Home-made event object!
var e = $.Event('click');
e.pageX = 100;
e.pageY = 100;
$('p').trigger(e);
```

Sanal bir tıklama olayı yarattık ve “ pageX” ve “pageY” özelliklerini atadık. Bu bize, bir olay işleyicisi tetikleyen olayların detayları üzerinde nihai kontol sağlar.