

**İSTANBLU GELİŞİM ÜNİVERSİTESİ**

# MESLEK YÜKSEKOKULU

# ONYX AJAX KÜTÜPHANESİ

**Projeye Gurubu**

**Hakan YOLAT – 120111019**

**Alper AKIN – 120111033**

**Özgür KAN – 120111041**

**Muhammet ÖZSOY - 120111089**

**Projeyi Yöneten**

**Yrd. Doç. Dr. Önder EYECİOĞLU**

**Ocak 2014**

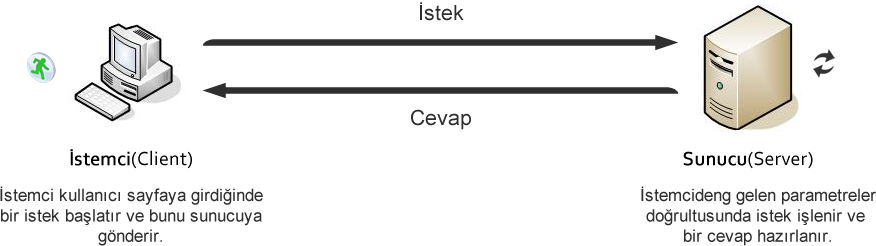
İÇİNDEKİLER

1. Sistem Analizi........................................................................................... 3
   1. Sistemin tanımı.............................................................................. 3
   2. Sistemin parametreleri................................................................... 4
   3. Problemin tanımı........................................................................... 5
   4. Problemin kaynağı......................................................................... 6
   5. Alternatiflerin araştırılması............................................................ 7
   6. Farklılıkların değerlendirilmesi..................................................... 8
2. Tasarım...................................................................................................... 9
   1. Teknoloji seçimi............................................................................ 9
   2. Kullanılan yöntem(teknik)............................................................ 10
   3. Projenin temel yapısı..................................................................... 13
3. Uygulama.................................................................................................. 15
   1. Sistemin test edilmesi................................................................... 15
   2. Test sonuçları................................................................................ 24
4. Kaynaklar.................................................................................................. 16
5. Değerlendirme sayfası

1. Sistem analizi

1.a. Sistemin tanımı

Problemine çözüm üretmeyi amaçladığımız sistem sunucu sistemleridir. Sunucular istemci(client)’den gelen istekleri işledikten sonra istemcinin gönderdiği parametreler doğrultusunda bir cevap verir. Şekil 1.1 de görüldüğü gibi İstemci ve Sunucu arasındaki bu süreçte sunucunun bandwith(bant genişliği) özelliğinden, işlenen veri miktarı kadar kayıp olur.



Şekil 1.1

Yine Şekil 1.1 de görüldüğü gibi İstemci(Client) burada kullanıcıdır. Kullanıcı sayfaya girdiğinde bir istek başlatır ve bunu sunucuya gönderir. Sunucu istemciden gelen parametreler doğrultusunda bu isteği işler ve ortaya bir cevap çıkar. Daha sonra bu cevap istemciye gönderilir. Sunucu sistemleri arasındaki bu süreç böylece sonlanmış olur.

Kullanıcı, bilgisayarındaki tarayıcı aracılığı ile bir sitede gezintiye başladığında sayfalar arası her geçiş esnasında aslında sunucuya yeni istekler göndermiş olur. Ya da kullanıcı bir sitedeki iletişim formunu doldurup gönderdiği zaman yine bir istek başlatmış olur. Bu istek ile birlikte sunucuya iletişim bilgilerini gönderir. Bu istek sunucuya ulaştığında bu bilgiler veritabanına kayıt edilir ve bir mail aracılığı ile yöneticiye iletilir. Daha sonra bu mesajın işlendiğine dair bir bilgi ile birlikte bir cevap istemciye geri gönderilir. Bunun sonucunda ekrana “Mesajınız gönderilmiştir” mesajı verilir. İşte burada yine aynı süreç söz konusudur.

Burada istek ve cevap soyut bir araçtır. Yine de bu istekler arasında yapılan işlemler esnasında bir veri transferi söz konusu olduğu için işlenen veriler bant genişliği miktarını gittikçe azaltır.

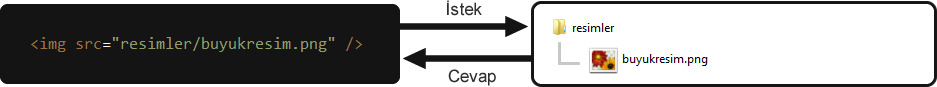
1.b. Sistemin parametreleri

1.b.1. İstek ( Request )

Kullanıcı web sayfasına girdiği zaman web sayfasına ait algoritmaya göre sunucuya bazı istekler gönderir. Örneğin bir veritabanından bir sorgu sonucunda oluşan veri veya sunucuda bulunan bir resim istemci tarafından sunucudan istenebilir.

1.b.2. Cevap ( Response )

İstemciden gelen parametrelere göre istek sunucuda işlenir ve sonuç olarak ortaya çıkan veri cevap olarak istemciye iletilir.



Şekil 1.2

Örneğin; Tarayıcı HTML kodunu yorumlarken bir <img> yani resim etiketine sıra gelir. Bu resmi kullanıcıya tasarımsal olarak göstermek için resim adresini sunucuya gönderir(istek) ve sunucuda böyle bir resim var ise resim’e ait veri tarayıcıya cevap olarak gönderilir ve tarayıcı bunu görsel olarak sunar.

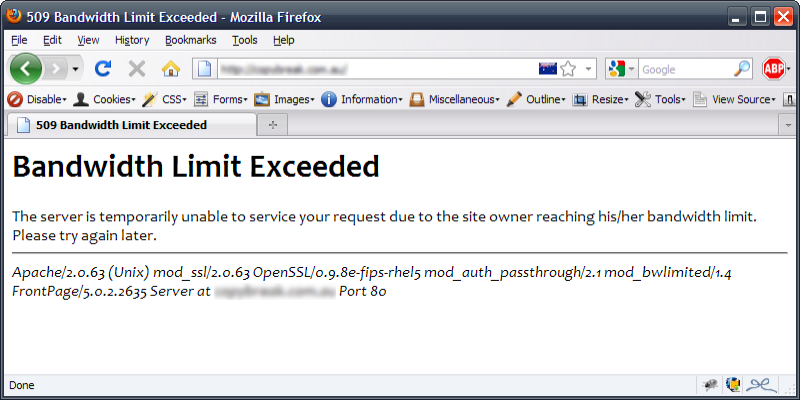
Şekil 1.2 de görüldüğü gibi bir resim transferi esnasında resmin boyutu kadar bant genişliğinden azalma olur. Daha sonra bu resim kullanıcıya ulaşana kadar bir bekleme süresi ortaya çıkar.

1.c. Problemin tanımı

Sunucu sistemlerindeki daha önce anlatılan bu süreç sonucunda ortaya bazı sorunlar çıkar. Bu sorunlardan en önemlisi bant genişliği sorunudur. Diğeri ise sayfa yüklenme süresinin uzamasıdır. Sırayla bu sorunları tanımlayalım.

1.c.1. Bant genişliği

Sistemin parametreleri arasındaki ilişki sonucu oluşan veri transferi miktarı sunucudaki web sayfasının kullanıcı kitlesine göre artmaktadır. Bu durum göz önüne alındığında bir süre sonra sunucunun bant genişliği(bandwidth) miktarı kendi iletişim kanalının kapasitesini aşar ve web sayfası veri transferi yapamaz hale gelir. Sonuç olarak Şekil 1.3 de görüldüğü gibi web sayfasına erişilemez.



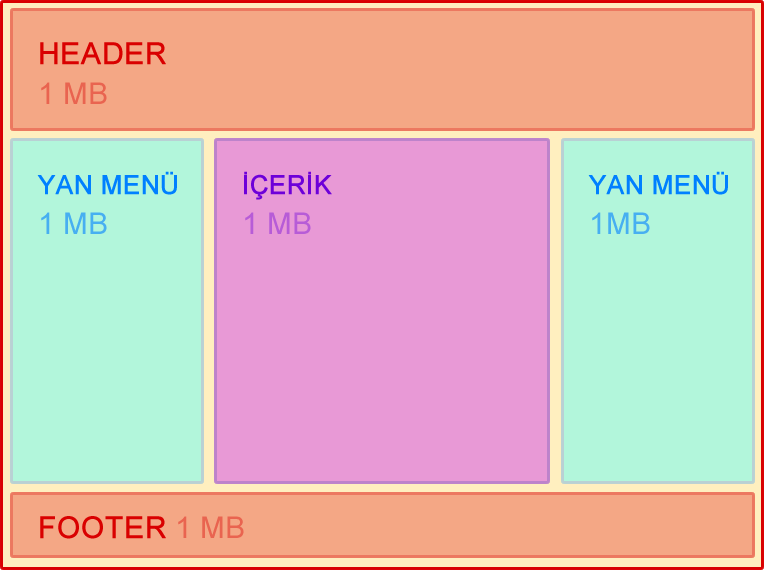
Şekil 1.3

1.c.2. Sayfa yüklenme süresinin uzaması

Bununla birlikte veri transferinin sonuçlandırılması(web sayfasının yüklenme süresi) kullanıcının veri transferi hızına göre değişkenlik göstermektedir. Aynı sayfada birkaç veri transferi için, birlikte yüklenen diğer verilerin tekrar yüklenmesi veri transferinin çoğalmasına yani sayfanın yüklenme hızının gereksiz yere uzamasına sebep olur. Buda genellikle kullanıcı memnuniyetinin olumsuz şekilde sonuçlanmasına neden olur.

1.d. Problemin kaynağı

Şekil 1.4 de örnek bir web sayfasının taslağı bulunmaktadır. Sayfa her yüklemede 5 MB ‘lık veri transferi gerçekleşmektedir.



Şekil 1.4

Şekil 1.4 deki taslağı göz önünde bulundurursak sadece içerik alanının güncellenmesi amaçlandığında :

1. 1 MB’lık veri transferi için toplam 4 MB’lık veri tekrar yüklenecektir. Buda bant genişliği miktarının gereksiz yere harcanması anlamına gelir.
2. 1 MB’lık verinin gösterilmesi için toplam 4 MB’lık verinin tekrar yüklenmesi beklenecektir. Bu bekleme süresi kullanıcı memnuniyetini olumsuz yönde etkiler.

1.e. Alternatiflerin araştırılması

Proje konusu belirlendikten sonra litaratür taraması yapılmıştır. Sonuç olarak projemize benzer aşağıdaki projelere rastlanmıştır.

* **jQuery** : John Resig tarafından 2006 yılında geliştirilmiş bir açık kaynak javascript kütüphanesidir.jQuery bir JavaScript kütüphanesidir. Ya da farklı bir JavaScript Framework'üdür denebilir. [1]



Şekil 1.5 : jQuery’nin logosu

* **Prototype** : Prototype JavaScript library açık kaynak en populer javascipt kütüphanesidir.Genel olarak yeni nesil web uygulaması sayılan ajax uygulamalarında kolaylık sağlar. [2]



Şekil 1.6 : Prototype’in logosu

* **Sardalya** : DOM desteği olan tüm güncel tarayıcılarda çalışmak için tasarlanmış bir 'AJAX Kütüphanesi'dir. [3]

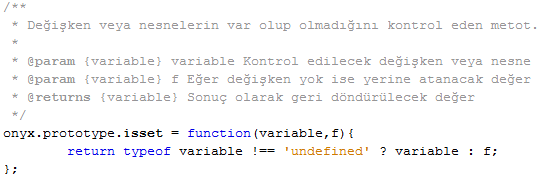
Projemizin diğerlerinden farkları şunlardır ;

1. JSDOC standartları(Yorum satırlarının şekillendirilmesi)
2. Nesne Yönelimli Javascript (Object-Oriented Javascript)
3. Akıcı arabirim (Fluent Interface)
4. Sadece AJAX üzerine yazılmış olması

1.f. Farklılıkların değerlendirilmesi

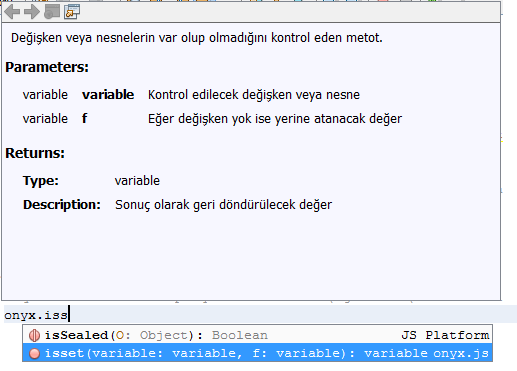
1.f.1. JSDOC standartları(Yorum satırlarının şekillendirilmesi)

JSDOC standardı diğer dökümantasyon stadartları ile aynı amaca hitap eder. Bu standartlar IDE(Integrated Development Environment)’lerin otomatik tamamlama özellikleri sayesinde geliştiricilere sınıflar ve metotlar hakkında bilgiler ve parametreler ile ilgili bilgiler verir.



Şekil 1.7

Şekil 1.7 deki gibi bir standart ile yazılmış kod, geliştiriciler tarafından kullanıldığı zaman Şekil 1.8 de olduğu gibi otomatik tamamlama özellikleri devreye girer.



Şekil 1.8

1.f.2. Nesne Yönelimli Javascript

Metotlar temel bir javascript nesnesine bağlıdır. Benzer projelerde metotlar genellikle constructor(kurucu) metot’un altındaki bir diziye bağlı anahtarlar ile çalışır.

1.f.3. Akıcı arabirim

Metotlar birbiri ardına bağlı yöntemler şeklinde çağırılabilir. Diğer projelerin hepsinde olmasa da bu özellik farklılık göstermektedir.

1.f.4. Sadece AJAX üzerine yazılmış olması

Diğer kütüphaneler projemizden farklı olarak javascript’in tarayıcı nesne modeli(BOM) ve Belge nesne modeli(DOM) gibi konular üzerine kuruludur.

2. Tasarım

2.a. Teknoloji seçimi

Proje temelinde Javascript teknolojisi kullanılmıştır.

JavaScript Netscape Navigator 2.0 ile birlikte Brendan Eich tarafından geliştirilen ve önceleri Mocha daha sonra LiveScript olarak adlandırılan ve en sonunda şu anki adını alan JavaScript dili başlangıçta sadece istemci taraflı (client-side) yorumlanan bir betik dilidir. [4]

Proje gelişimi sürecinde proje tasarımı Nesne tabanlı javascript (Object Oriented Javascript) yöntemi kullanılmıştır.

Nesne tabanlı javascript’de diğer Nesne tabanlı mimarilerden farklı olarak sınıf kavramı yoktur. Bir nesne kurucu(constructor) metot vardır ve kurucu metot kendisinin sınıfı anlamına gelir. Bir metot diğer metotlar’a prototype alanı aracılığı ile kalıtılır. Projemiz bu kalıtım tekniğini kullanmaktadır.

Şekil 2.1 de görüldüğü OOJS ( Object Oriented Javascript) bir başka Nesne tabanlı mimarisi olan PHP(OOP) dili ile kıyaslanmıştır. Görüldüğü üzere Javascript sınıf kavramı yoktur. Diğer mimariden farklı olarak prototype alanı aracılığı ile bir kalıtım söz konusudur.



Şekil 2.1

2.b. Kullanılan yöntem(teknik)

Projemiz tasarlanırken nesne yönelimli javascript teknolojisi kullanılmıştır ve AJAX tekniği projemizin temelidir.

AJAX sunucu sistemleri arasındaki istek-cevap mekanizmasını sayfanın tekrar yenilenmesine gerek kalmadan, daha detaylı ve etkileşimli bir şekilde kullanmamızı sağlar. Bunun için ilk önce bir AJAX Nesnesi oluşturulması gerekir.

AJAX nesnesi javascript kullanılarak oluşturur. AJAX Nesnesi oluştururken dikkat edilecek en önemli husus tarayıcı desteğidir.

AJAX nesnesinin genel adı XMLHttpRequest (XHR) nesnesidir. Fakat Internet Explorer 6+ sürümleri XMLHttpRequest nesnesini desteklememektedir. Bunun için Microsoft’un geliştirdiği ActiveXObject nesnesi devreye girer.

Şekil 2.2 de görüldüğü gibi XMLHttpRequest nesnesi için, Google Chrome, Mozilla Firefox, Safari ve Opera gibi tarayıcıları desteklediğini, Internet Explorer’ın ise ActiveXObject nesnesini desteklediğini söyleyebiliriz.



Şekil 2.2

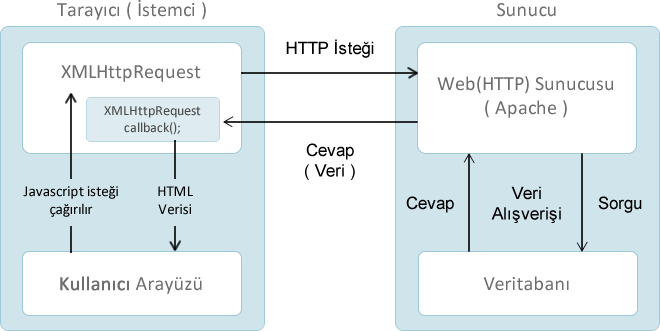
Bu nesnelerin oluşturulması için ise Şekil 2.3 de görüldüğü gibi bir algoritma uygulanır. Burada Tarayıcı uyumluluğu dikkate alınarak bir XHR nesnesi nasıl oluşturulur sorusunun cevabını bulabiliriz.



Şekil 2.3

Şekil 2.3 de görüldüğü üzere tarayıcı XMLHttpRequest nesnesini okuyor. Eğer böyle bir nesne tanımlı dönerse true cevabını alıyor ve nesneyi oluşturuyor. Eğer böyle bir nesne tanımlı değil ise false cevabını alıyor ve ActiveXObject nesnesini oluşturuyor.

Peki bu teknik tam olarak nasıl kullanılır? Aslında temel olarak sunucu sistemlerinde mekanizma buradada mevcuttur. Fakat detaylar AJAX’ı AJAX yapan en önemli unsurdur. Aşağıda bulunan Şekil 2.4 deki grafik tüm bu AJAX mekanizmasının nasıl işlediğini göstermektedir.



Şekil 2.4

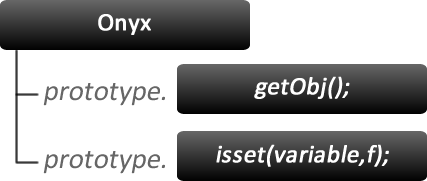
Yukarıdaki Şekil 2.4 de gerçekleşen olaylara sırasıyla bir göz atalım.

1. Buradaki işlem kullanıcının web sayfasında ajax kullanılan bir uygulamayı kullanmasıyla başlıyor.
2. Bu uygulama bir javascript isteği çağırıyor. Bu javascript isteği kendisi için uyarlanmış bir XMLHttpRequest nesnesini oluşturuyor.
3. Daha sonra bir HTTP isteği başlatıyor ve gönderiyor.
4. Bu istek sunucuya ulaştığında sunucu bu isteği kendisine gelen parametreler doğrultusunda değerlendiriyor ve bir cevap oluşturuyor.
5. Bu cevabı istemciye gönderiyor.
6. İsteği başlatan javascript isteği içinde bulunan bir callback metodu ise bu cevabı değerlendirip kullanıcının bulunduğu sayfaya HTML verisi göndererek bir etkileşim sağlıyor.

Callback için “Başka bir koda argüman olarak verilen, çalışabilir kod veya kod parçası.” [5] Diyebiliriz. Bunu bir metot olarak kullandığımızda ise bir nevi kendi kendini çağıran metotlar olarak karşımıza çıkmakta.

2.c. Projenin temel yapısı

Onyx içinde bir kurucu(constructor) metot ve iki prototip metot bulunmaktadır. Kurucu metot aynı isimdedir. Diğer 2 prototip metot ise getObj ve isset metotlarıdır.



Şekil 2.5

Yukarıdaki Şekil 2.5 projemizin temel yapısını en sade şekliyle anlatmaktadır. Sırasıyla temel yapıyı oluşturan bu unsurlara bir göz atalım.

2.c.1. Kurucu (Constructor) metot

Kurucu metot parametre olarak args dizisini işler. Bu diziden gelen değerler ile ajax isteği açılır, olay dinleyicileri için metotlar tanımlanır ve sonuçlar yeniden değerlendirilir. Tanımlamalar hariç tüm olaylar bir hata kontrol mekanizması içerisinde gerçekleşir. Bir hata olursa bu hataya sebep olan komut dizisi yakalanır ve konsol ekranında gösterilir.

Kurucu metot bir args dizisi almak zorundadır. Aksi halde teknik ayarlar yapılamaz ve istek açılamaz.

Kurucu metodun aldığı args parametresi teknik tanımlamarı ve olay dinleyicilerin çağıracağı callback metotları içerir.

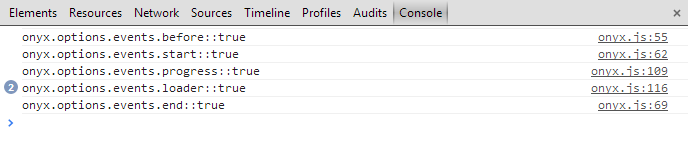
Peki args dizisi hangi değerleri alır ?

1. **method :** İstek türüdür. Bu değer ‘POST’ veya ‘GET’ olabilir.
2. **url :** İstek yapılacak adrestir.
3. **data :** İstek yapılacak adrese gönderilecek veridir. Bu veri bir metin, form elemanı veya bir resim olabilir.
4. **async :** Aynı anda birden fazla istek yapılıp yapılamayacağı hakkındaki karardır. Bu değer ‘true’ veya ‘false’ olabilir.
5. **type :** İstek sonucunda verilen cevabın türüdür. Bu değer ‘text’ veya ‘json’ olabilir.
6. **events :** Kendi başına bir dizidir. İstek ile ilgili olay dinleyicilerine gönderilen metotlar bu dizide tanımlanır.

Peki events dizisi hangi değerleri alır ?

1. **before :** İstek başlamadan önce çağırılacak metot.
2. **start :** İstek başladığında çağırılacak metot.
3. **end :** İstek sonlandığında çağırılacak metot.
4. **error :** İstek bir hata ile karşı karşıya kaldığında çağırılacak metot.
5. **success :** İstek başarı ile sonlandığında çağırılacak metot.
6. **progress :** İstek esnasındaki toplam data miktarını işleyebileceğimiz metot.
7. **loader :** İstek işlenme esnasında çağırılacak metot.

Onyx çağırıldıktan sonra olay dinleyiciler içinde gerçekleşen tüm olayları tarayıcıya log’lar. Aşağıdaki Şekil 2.6 da Google Chrome ‘un Console ekranında Onyx içinde gerçekleşen olayların log’lanması yer almakta.



Şekil 2.6

2.c.2. getObj Metodu

Kurucu metot içinde oluşturulan ajax nesnesini geri döndürür. Bu nesneye kalıtım metoduyla ulaşılır.

2.c.3. isset Metodu

Kendisine gönderilen değerin undefined(tanımsız) olup olmadığını kontrol eder. Eğer tanımsız değil ise kendisine gönderilen değeri, tanımsız ise kendisine gönderilen ikinci parametreyi geri döndürür.

3. Uygulama

3.a. Sistemin test edilmesi

Uygulama bölümün sistemin test edilmesi kısmında Onyx nerelerde nasıl kullanılabileceği hakkında örnek verilecek ve üzerinde çalıştığımız sunucu sistemlerindeki bant genişliği ve sayfa yüklenme süresi sorunlarına ürettiğimiz çözüm için geliştirilen bir uygulama ile birlikte bu sorunların ne kadar giderilip giderilmediği hakkında bilgiler yani “amacımıza ne kadar ulaştık?” sorusuna verilen cevap yer almaktadır.

3.a.1. Onyx nerelerde kullanılabilir?

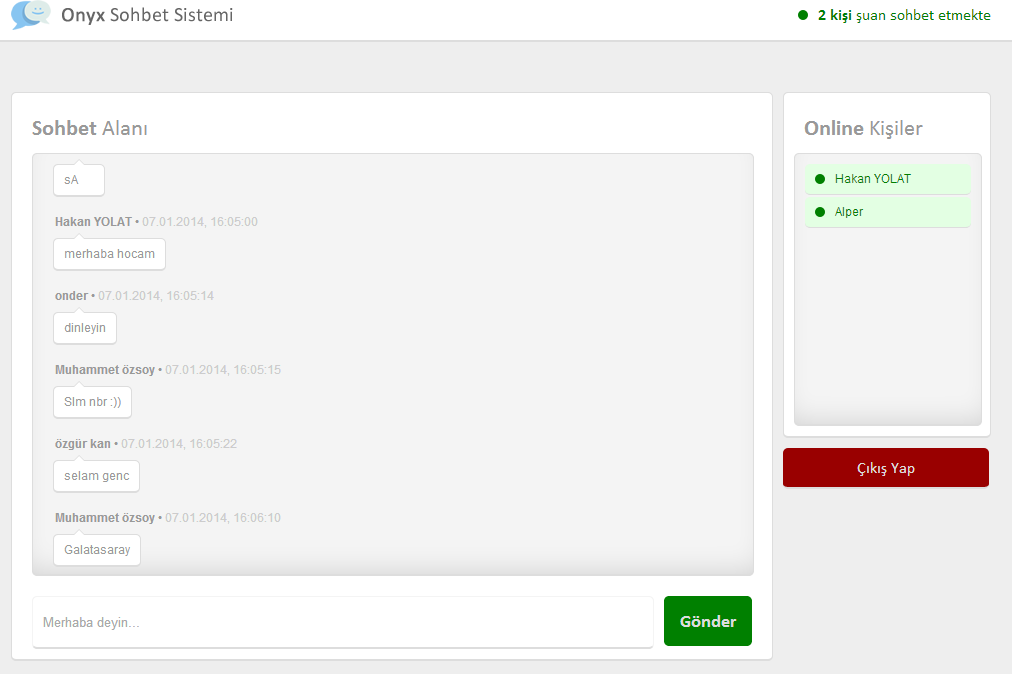
Onyx ‘in temel prensibi ‘veri alışverişi’ olduğu için, veri alışverişi gerektiren ve temel HTTP protokollerinin kullanıldığı tüm uygulamalarda kullanılabilir.

Bu kısımda ‘Onyx kullanılarak neler yapılabilir ?’ sorusunu ve “amacımıza ne kadar ulaştık?” sorusunu cevaplamak için Sistemin Test Edilmesi aşamasında 3 uygulamayı ve bu uygulamalarda Onyx’in nasıl kullanıldığını inceleyeceğiz.

Bu uygulamalar :

1. Chat uygulaması
2. Paint uygulaması
3. Görsel arama motoru uygulaması

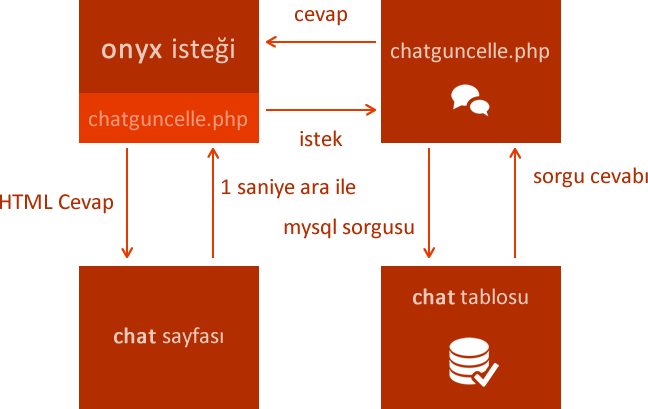
3.a.1.1. Chat uygulaması



Şekil 3.1 : Onyx sohbet sistemi

Chat uygulamaları sayfa yenilenmeden veri alışverişinin olması gerektiği en popüler uygulamalardandır. AJAX tekniği kullanılarak bir çok chat uygulaması geliştirilmiştir. Peki bu uygulamalardaki temel mantık nedir ? Onyx ile geliştirilen Chat uygulamasında Onyx’in kullanıldığı bir çok ayrıntı vardır. Bu ayrıntıları şu şekilde sıralayabiliriz :

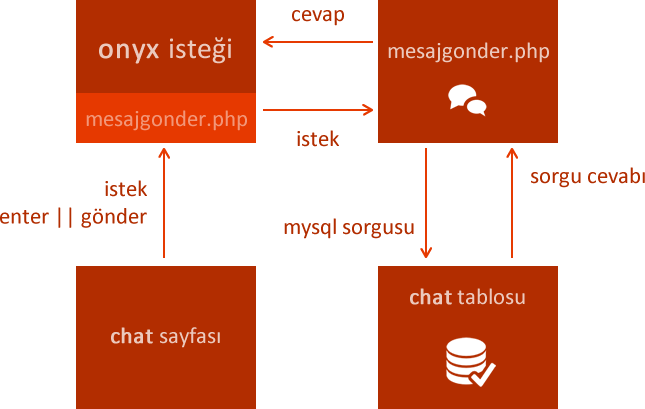
1. **Sohbet alanının yenilenmesi** : Kullanıcıların gönderdiği mesajların canlı bir şekilde gösterilmesi, yani mesajların gösterildiği alanın sürekli yenilenmesi gerekir. Uygulama gereği bu alanın yenilenmesi, o alana veritabanındaki chat tablosundan yeni bilgilerin eklenmesi anlamına gelir. Bunu yapmak için ise bir veritabanı sorgusu çalıştırmak, veritabanı sorgusu çalıştırmak için de sayfayı tekrar yenilemek yani sunucuya bir istek göndermek gerekir. Chat alanının sürekli yenilendiği bu uygulamada ise kullanıcının sayfayı tekrar tekrar yenilemesi söz konusu değildir. İşte bu sorunu aşmak için yine burada devreye Onyx giriyor.



Şekil 3.2

Yukarıda şekil 3.2 de görüldüğü üzere chat sayfasında 1 saniye ara ile bir istek başlatılıyor ve chatguncelle.php sayfası araclılığı ile chat tablosundan mesajlar alınıp chat alanına ekleniyor.

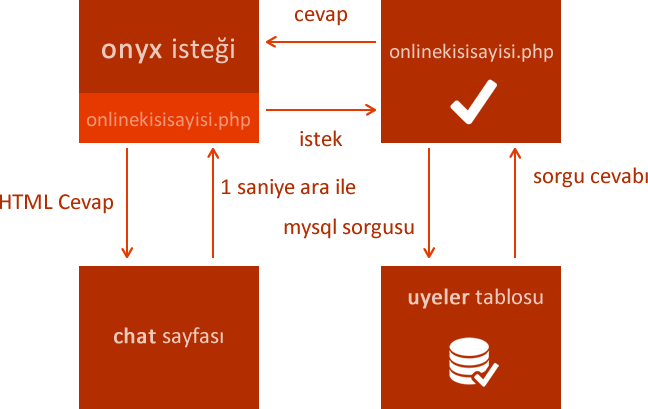
1. **Mesajın Gönderilmesi :** Chat uygulamalarında kullanıcının göndermek isteği mesajı gönderirken mesaj kutucuğuna mesajı yazdıktan sonra “enter” tuşuna basması veya çevresindeki “Gönder” butonuna basması gerekir. Normal bir sayfada bu yapıldığında her mesaj gönderildiğinde sayfa yenilenir. Bu durum böyle bir uygulamada söz konusu olamaz. İşte burada yine devreye Onyx giriyor. Aşağıdaki Şekil 3.3 grafiğini inceleyelim.



Şekil 3.3

Burada kullanıcı mesajı yazdıktan sonra klavyeden “enter” tuşuna bastığı zaman veya mesaj kutucuğunun çevresinden “Gönder” butonuna bastığı zaman kutucukta mesajla birlikte bir onyx isteği başlatılıyor. Bu istek mesajgonder.php sayfasına gönderiliyor ve bu sayfada bu mesaj chat tablosuna ekleniyor. Bir önceki maddede anlatıldığı gibi chat alanı yenilendiği için gönderilen mesajda anında görüntüleniyor.

1. **Çevirim içi kişi sayısı ve kişilerin gösterilmesi** : Çevirim içi kişi sayısının gösterilmesi için, sistemin çalıştığı zaman ile, kullanıcının mevcut sistem üzerinde aktif olduğu zamanın eşit olması veya belirli zaman aşımlarına karşı belirli aralıklarda olması gerekir. Örneğin sistem saati 20:00 ise ve o an o sitede bir kişi var ise, bir algoritma ile o kişinin son online zamanı saniyede bir güncellenir. Böylece sistem saati 20:00 da sitede olan bir kullanıcının son online zamanıda 20:00 olur. Bu durum sorgulanıp online kişi sayısı bulunur.



Şekil 3.4

Şekil 3.4 de 1 saniye ara ile online kullanıcı sayısı bir onyx isteği ile sorgulanır. Chat sayfasında bir javascript metodu ile 1 saniye ara ile bir onlinekisisayisi.php için bir onyx isteği gönderilir. Bu sayfada bir sorgu ile üyeler tablosundan yukarıda anlatılan algoritmaya uygun sonuçlar alınır ve chat sayfasına cevap olarak gönderilir. Daha sonra bu cevap Şekil 3.5 de gözüktüğü gibi sayfaya eklenerek online kişi sayısı gösterilebilir.



Şekil 3.5

Aynı zamanda bu sorgu ile online kişiler isimleri ve diğer bilgileride alınabilir. Bu yüzden aynı istek ile Şekil 3.6 da gözüktüğü gibi online kişilerin isimleride chat sayfasında gösterilebilir.



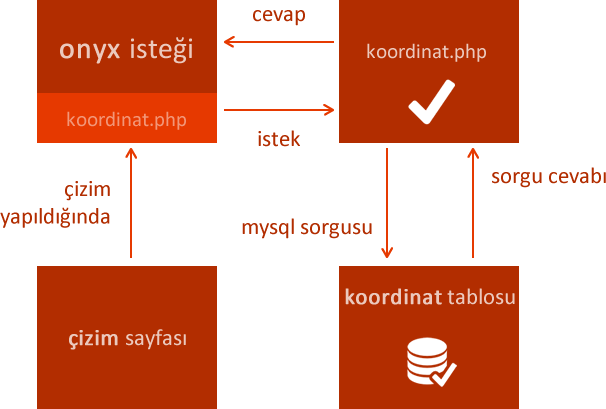
Şekil 3.6

1. Çevirim içi durumunun güncellenmesi : Şekil 3.5 ve Şekil 3.6 da olduğu gibi çevirim içi kişi sayısının ve çevirim içi kişilerin sorgulanması için sayfa içerisinde bulunan kişilerin son online olduğu zamanın güncellenmesi gerektiğinden bahsetmiştik. Böyle bir durumda yine bir istek gerekir. Fakat yine kullanıcının bu sorgu için belirli aralıklarla sayfayı yenilemesi söz konusu değildir. Burada yine onyx devreye girer.

3.a.1.2. Paint uygulaması

Onyx’in mobil sistemlerde de çalıştığını göstermek amacıyla basit bir paint uygulaması geliştirdik. Onyx’in bu uygulamadaki rolü ise çizim koordinatlarının gönderilip alınması ve bu sayede bir sayfada çizilen grafiğin diğer sayfada gösterilmesidir. Onyx paint uygulamasında 2 aşamada kullanılmıştır. Bunlar :

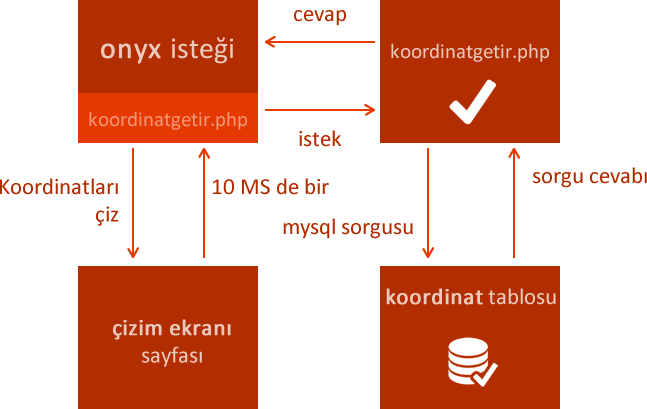
1. **Çizim koordinatlarının kaydedilmesi** : Mobilden, tabletten veya başka bir bilgisayardan, aynı sistem üzerindeki farklı bir sayfadan yapılan çizimin koordinatlarını, her parmak, kalem veya Mouse hareketinde kaydedilmesi gerekir. İşte böyle bir durumda yine Onyx devreye girer.



Şekil 3.8

Şekil 3.8 de görüldüğü gibi çizim ekranı sayfasında bir çizim hareketi (parmak, kalem veya Mouse hareketinde ) olduğu zaman koordinat.php sayfasına bir istek gönderilir. Bu istekte x,y koordinatları parametre olarak gönderilir ve koordinat.php sayfasındaki bir sorgu ile bu koordinatlar koordinat tablosuna kayıt edilir.

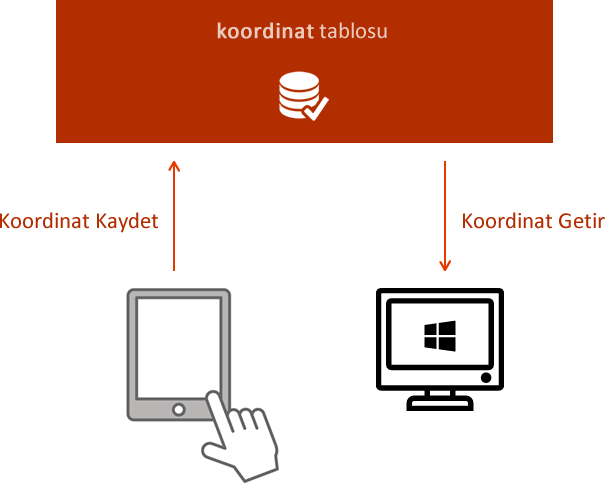
1. **Çizim koordinatlarının getirilmesi** : Şekil 3.8 deki çizim sayfasından çizilen çizim için kaydedilen koordinatların, çizim ekranı sayfasında anında gösterilmesi için çizim koordinatlarının yine onyx isteği ile belirli aralıklarla getirilmesi ve aynı algoritma ile istek sonucunda çizilmesi gerekir.



Şekil 3.9

Şekil 3.9 da görüldüğü gibi 10 MS de bir onyx isteği ile koordinatgetir.php ye istek yapılmaktadır. Bu sayfada koordinatlar tablosuna sorgu yapılarak çizim koordinatları alınmakta ve çizim ekranı sayfasına çizim sayfasındaki aynı algoritma ile çizilmektedir.

Sonuç olarak ortaya şekil 3.10 daki gibi bir büyük resim çıkmaktadır.



Şekil 3.10

Bahsedilen Chat uygulamasını ve paint uygulamasını denemek için Proje lideri Hakan YOLAT’ın internet sitesinde bulunan aşağıdaki adreslere girerek bu uygulamarı test edebilirsiniz.

**Chat uygulaması için**

[**http://hakanyolat.com/onyxtest/chat/**](http://hakanyolat.com/onyxtest/chat/)

adresini kullanabilirsiniz...

**Paint uygulaması için**

*Çizim sayfası*

[**http://hakanyolat.com/onyxtest/paint/**](http://hakanyolat.com/onyxtest/paint/ekran.html)

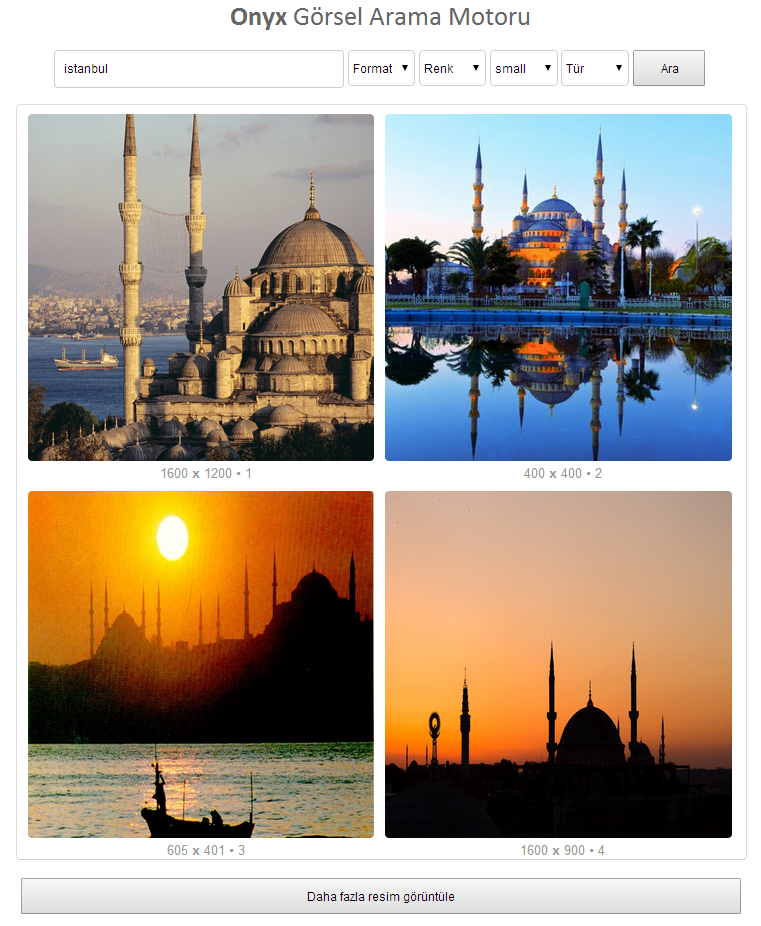
*Çizim ekranı*

[**http://hakanyolat.com/onyxtest/paint/ekran.html**](http://hakanyolat.com/onyxtest/paint/ekran.html)

adreslerini kullanabilirsiniz...

3.a.1.3. Görsel arama motoru uygulaması

Sistemin test edilmesi kısmında bu uygulamada “amacımıza ne kadar ulaştık” sorusuna cevap arayacağız. Bunun için bir görsel arama uygulaması geliştirdik ve onyx’i rakamlarla test ettik.



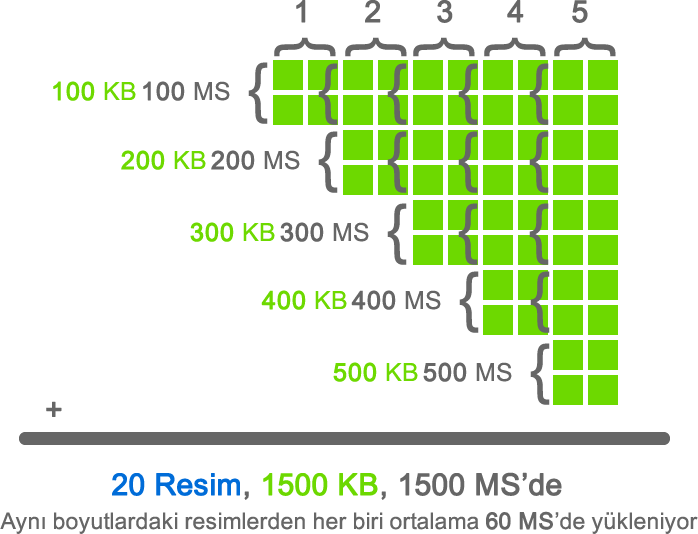
Şekil 3.11

Şekil 3.11 de gördüğünüz bu uygulamada bir resim aranıyor ve sonuçlar “Daha fazla resim görüntüle” butonu ile sırasıyla sonuçların gösterildiği alana ekleniyor. Bu butona her basıldığında 4 adet yeni resim sonuçların gösterildiği alana ekleniyor. Böyle bir uygulamanın normal olarak yapıldığını düşünürsek bu butona her tıklandığında 4 yeni resim için daha önceki resimlerde yeniden yüklenecek. Böyle bir durum söz konusu olduğunda gereksiz yere bant genişliği harcamış oluruz ve kullanıcılar tekrardan diğer resimlerin yüklenmesini bekler, bu durumda sayfa yüklenme süresinden de zarar etmiş oluruz ve bu kullanıcı memnuniyetini negatif yönde etkiler.



Şekil 3.12

Uygulamayı sabit değerlerle test edelim. Bu uygulamanın onyx ile yapıdlığını farzedelim. Bu durumda her “daha fazlasını göster” butonuna bastığımızda Şekil 3.12 de görüldüğü gibi 4 adet resim gösterilecek ve diğer resimlerin tekrardan yüklenmesi gerekmeyecek. Bu durumda sabit değerlerle 20 resim sonuç olarak gösterildiğinde her birinin 100 KB olduğunu ve 100 MS de yüklendiğini varsayarsak toplamda 500 KB ‘lık 20 resim 500 MS’de yüklenmiş olur.



Şekil 3.13

Aynı uygulamanın onyx ile yapılmadığını, standart http isteği ile uygulandığını ve aynı resimlerin şekil 3.12 deki aynı değerler olacak şekilde arandığını varsayalım. Şekil 3.13 de her 4 resim için sayfa tekrardan yenilenecek ve diğer resimler tekrardan yüklenecek ve yine diğer resimlerin yüklenme süresi beklenecektir. Bu durum göz önüne alındığında bu sefer aynı 20 resim için 1.500 KB harcanacak ve bunun için 1.500 MS beklenecek.

Yani genel olarak baktığımızda :

* Standart http isteği ile her bir resim ortalam 75 MS’de yükleniyor. Onyx isteği ile bu rakam 25 MS’dir.
* Bu durumda Bant genişliğinden tasarruf 1.000 KB’tır.
* Sayfa yüklenme süresinden tasarruf ise 1.000 MS’dir.

Bu şekilde sabit değerlerle onyx’in amacına ulaştığını teoride kanıtlamış olduk. Şidmi reel rakamlarla uygulamayı test ederek sonuçları inceleyelim.

**Uygulamanın Onyx ile yapılmış halini**

<http://hakanyolat.com/imgtest/>

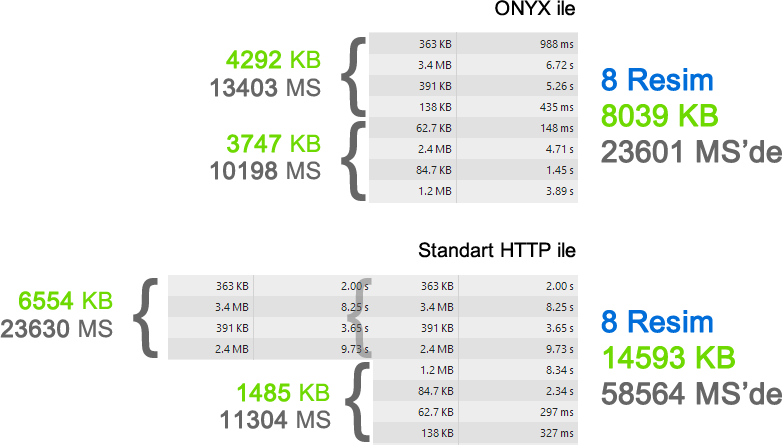
**Uygulamanın Standart http ile yapılmış halini**

<http://hakanyolat.com/imgtest2/>

adreslerinden test edebilirsiniz...

3.b. Test sonuçları

Birazdan göreceğiniz rakamlar Google Chrome’un Network ekranındaki data miktarlarıdır. Görsel arama motoru uygulaması ile “İstanbul” kelimesi ile bir arama yapıyoruz...



Şekil 3.14

Ve sonuçlar alındıktan sonra Chrome Network ekranında data transfer miktarlarını ve yüklenme sürelerini topluyoruz.

Burada aynı 8 resim;

Onyx ile 8.039 KB , 23.609 MS’de transfer edilmiştir.

Standart HTTP isteği ile 14.593 KB , 58.564 MS’de transfer edilmiştir.

Bu durumda ;

Bant Genişliğinden 6554 KB yani ~ 45% tasarruf edilmiştir.

Sayfa yüklenme süresinden 34.963 MS yani ~ 60% tasarruf edilmiştir.

Şekil 3.15 : Onyx ile yüklenen resimlerin boyutsal(KB) grafiği

Şekil 3.16 : Onyx ile yüklenen resimlerin zamansal(MS) grafiği

Şekil 3.17 : Standart HTTP ile yüklenen resimlerin boyutsal(KB) grafiği

Şekil 3.18 : Standart HTTP ile yüklenen resimlerin zamansal(MS) grafiği

Şekil 3.19 : Onyx ve Standart http isteğinin bant genişliği ve sayfa yüklenme süresi açısından grafiği

Şekil 3.20 : Tasarruf miktarlarının grafiği

KAYNAKLAR

1. <http://tr.wikipedia.org/wiki/JQuery>

2. <http://www.frmtr.com/web-dizayn/441205-prototype-javascript-library-ile-neler-yapabilirsiniz.html>

3. <http://www.gigablast.com/get?q=&c=dmoz3&d=197078788781&cnsp=0>

4. <http://tr.wikipedia.org/wiki/JavaScript>

5. <http://en.wikipedia.org/wiki/Callback_(computer_programming)>