BLM3580
Sistem Programlama

2021-2022 GÜZ YARIYILI

DR.ÖĞR.ÜYESI GÖKSEL BIRICIK

DOM

Kolay fark edilemeyen JavaScript

İyi programlama stili tüm JavaScript kodunu ayrı bir dosyaya koymayı ve HTML ile karıştırmamayı gerektirir. Bu stile kolay fark edilmeyen JavaScript adını veriyoruz.

Kolay fark edilen:

- JS: `function goDoIt(){ ... }`
- HTML: `<button onClick="goDoIt();">Go</button>`

Kolay fark edilmeyen:

- JS: `function goDoIt(){ ... }`
- ...
- `document.getElementById("goButton").onclick = goDoIt;`
- HTML: `<button id="goButton">Go</button>`

Not: JS'de olay karşılayıcısı olarak bir fonksiyon ismi verdiğimizde, fonksiyon isminden soonra parantez **()** koyulmaz.

Çalışmayı Ötelemek

Problem: JS kodu <**script>** etiketinden okunduğu anda çalıştırılır. Ancak;

```
document.getElementById("goButton").onclick = goDoIt;
```

Kodu o kadar erken çalışmamalıdır.

goButton elemanı henüz yaratılmadı?

Çalışmasının, sayfanın yüklenmesinden **sonraya** ötelenmesi gereklidir.

Çözüm: **window.onload** olayına bu sorumluluğu yerine getirecek bir fonksiyon atayın.

Anonim Fonksiyonlar

JavaScript olay karşılayıcısı olarak isimsiz fonksiyon atanmasına olanak verir. Daha basit kod ile:

```
window.onload = function ()
{
    // attach event handler
    document.getElementById("goButton").onclick = goDoIt;
}
```

Dikkat edin: olay adlarının hepsi küçük harflidir. Değişken ve fonksiyon isimlerinde sıklıkla kullandığımız gibi deve stili değildir. :

`onload` ve **`onchange`** doğru, **`onLoad`** ve **`onChange`** yanlış.

Bir Elemanı Seçmek

Belirli bir eleman ya da eleman kümesini seçmek için kullanılan fonksiyonlar:

`document.getElementById(id)`
- **id="id"** özelliğine sahip elemanı seçer

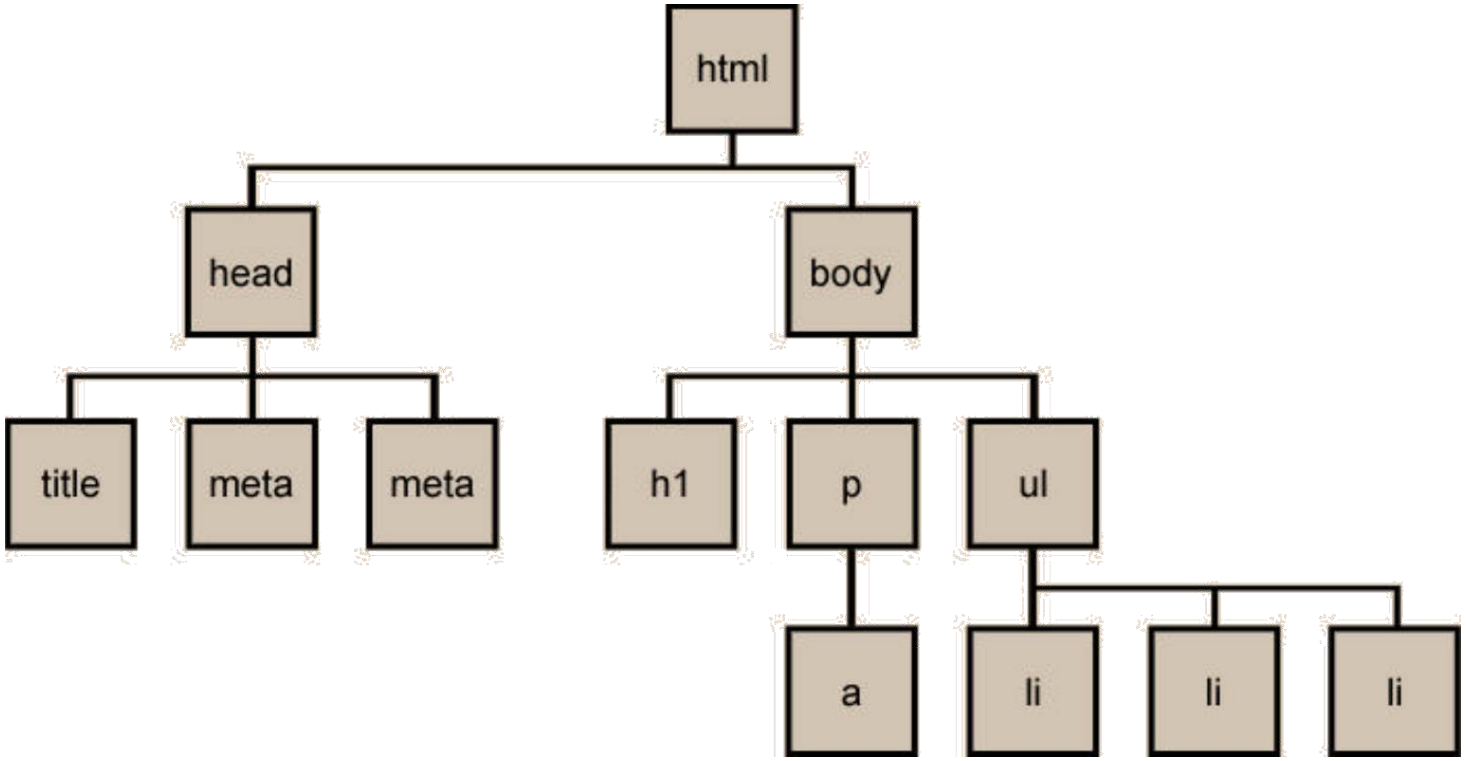
`document.getElementsByName(name)`
- **name="name"** özelliğine sahip tüm elemanları seçer

`document.querySelector(selector)`
- Bir CSS seçicisine uyan ilk elemanı seçer (HTML5)

`document.querySelectorAll(selector)`
- Bir CSS seçicisine uyan tüm elemanları seçer (HTML5)

Belge Nesne Modeli (DOM)

Web sayfasının elemanları bir hiyerarşi oluşturur. Buna DOM ağacı adını veriyoruz:



DOM Düğüm Tipleri (DOM Nodes)

Ana DOM düğüm tipleri:

Eleman düğümü
- HTML etiketine karşılık gelir
- Eleman, metin, özellik ve çocuk düğümler içerebilir.

Metin düğümü
- Bir elemanın bulundurduğu metin içeriği
- İlgili elamanın çocuğudur
- Çocuk düğümler veya özellikler barındıramaz

Özellik düğümü
- Özellik/değer çifti attribute/value pair
- ilgili elamanın çocuğudur
- Çocuk olarak metin barındırabilir

Düğüm Nesnesi

DOM düğümleri nesnelerdir. Özelliklerinden bazıları:

`className` — elemana ait CSS sınıflarının listesi

`innerHTML` — elemanın içinde yer alan içerik. Etiketleri de kapsar

`parentNode` — düğümün ebeveyn düğümü

`firstChild` — Düğümün ilk çocuk düğümü

Ve daha fazlasıdır. Bazıları düğümün tipine bağlıdır.

Bu özelliklere JavaScript kullanılarak erişilebilir ve düğüm değiştirilebilir.

Form alanlarının DOM özellikleri

Form alanı tipinde bir elemanın aşağıdaki özellikleri vardır:

`value` — string, `input` veya `textarea` alanındaki metnin değeri

`selectedIndex` — integer, bir `select` listesinde seçili olan elemanın indeksi (numaralama 0'dan başlar).

`checked` — boolean, bir kutunun seçili olup olmadığı

`disabled` — boolean, br alanın inaktif olup olmadığı

DOM ile içeriği değiştirmek

Bir DOM blok elemanına yeni metin içeriği vermek:

```
elem.innerHTML = "Hello, world!";
```

Önceki örneklerde **`document.write()`** ile tarih ve saati gösteren örneği modern DOM yöntemleri ile tekrar gerçekleştirelim.

`innerHTML` özelliği metin ve HTML etiketi alabilir. Ör:

```
elem.innerHTML = "text <a href="page.html">link</a>";
```

Ancak bu yöntem **KÖTÜDÜR!** Keyfi ve yanlış HTML içeriği sayfaya eklenebilir. Stil ve içeriği birbirine karıştırır, hatalara ve bug'lara* açıktır. **Yapmayın!** Sadece düz metin içeriği girin.

Aynı işi yapan daha temiz DOM tekniklerimiz de mevcuttur.

* *Yukarıdaki mavi yazılı örnekte bir hata var. Bulabilir misiniz?*

DOM ile stilleri değiştirmek

Elemanın içeriği ile birlikte stilini de değiştrmek isteriz. Bir yol: **`style`** özelliğini kullanmak.

Not:

- CSS stil özellik isimleri tirelidir. Bunlar deve stiline döner.
- CSS özellik değerleri CSS ile aynıdır, fakat çift tırnak arasında alınırlar.

CSS: **`font-weight: bold;`**

Şu hale gelir:

JS: **`elem.style.fontWeight = "bold";`**

Yeni Elemanlar eklemek

Sayfadaki bir elemana şu DOM metotları ile yeni çocuklar verilebilir:

`document.createElement("type")`

- — verilen (herhangi isimdeki) tipten yeni bir eleman düğümü yarat

`document.createTextNode("string")`

- Tanımlanan metni içeren yeni bir metin düğümü yarat

`elem.appendChild(childElem)`

- Bir elemanın son çocuğu olarak bir çocuk düğüm ekle.

Bu sayfaya «Inspect element» ile bakın. Gövdedeki elemanlara tıklayarak açın ve çocuklarının JavaScript ile yaratıldığını görün. Bunlar «View Source» ile görünmezler.

`createTextNode` vs. `innerHTML` vs. `innerText` vs. `textContent`

`innerHTML = string` yerine neden **`createTextNode(string)`** kullanıyoruz?

`innerHTML` özelliği metin ve keyfi HTML etiketi alabilir.

Metnin kaynağı güvenilir ise (örneğin JS stringi) bu sorun değildir. Aksi taktirde, sızma saldırısı riskleri doğurur.

Eğer kaynak güvenilmez ise (örneğin doldurulan bir form alanı) **`createTextNode`** kullanın.

`innerText = string` aynı zamanda etketleri görmezden gelir. Ancak W3C standardı değildir, Frefox tarafından desteklenmez.

`textContent = string` W3C standardıdır, fakat IE < 9 tarafından desteklenmez.

Kolay fark edilmeyen stil

İlerleme iyi ancak CSS ve JS hala karışıyor. En iyi yöntem, JavaScript koduna CSS koymamaktır.

Nasıl: **`className`** özelliğini kullanın ve bu sınıf için br CSS kuralı yaratın.

- JS: **`elem.className = "redtext";`**
- CSS: **`.redtext { color: red; }`**

Not: Bir eleman birden fazla sınıfa at olabilir.

- **`className`** boşluklarla birbirinden ayrışmış bir sınıf listesidir.

Bunları yönetmek zahmetlidir:

En iyisi jQuery metotlarını kullanmaktır. Az sonra öğreneceğiz.

DOM Ağacını Dolaşmak

Her DOM elemanı DOM ağacında dolaşabilmek için şu özelliklere sahiptir:

childNodes — elemanın çocukları dizisi

firstChild, **lastChild** — **childNodes** listesinin ilk ve son elemanları.

nextSibling, **previousSibling** — Düğümle aynı seviyede yer alan, aynı ebeveyne sahip önceki ve sonraki düğümler.

parentNode — elemanın ebeveyn düğümü.

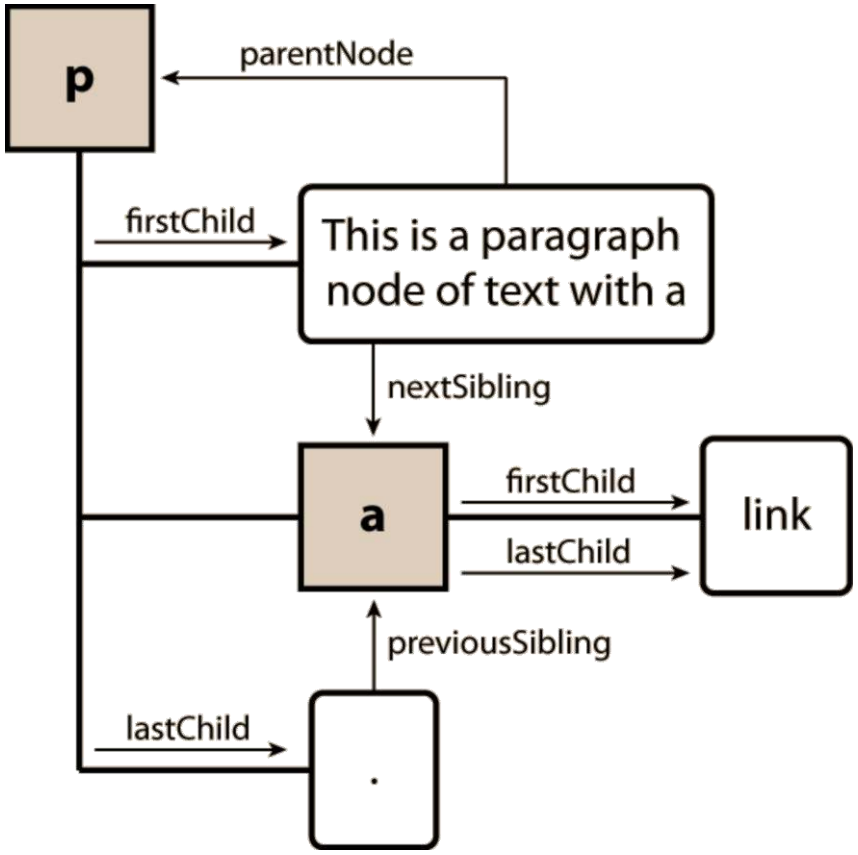
Eski tarayıcılar bazen DOM ağacını biraz farklı üretebilir. W3schools'ta tarayıcı uyumluluk değerlerine göz atın.

DOM Ağacı Dolaşma Örneği

```
<p id="foo">

This is a paragraph of text
with a

<a href="anotherpage.html">

Link

</a>

.

</p>
```

Gri kutular eleman düğümlerdir.

Beyaz kutular metin düğümlerdir.



DOM ağacında tarayıcı uyumlu dolaşma

Aşağıdaki kod (kaynak: w3schools.com) bazı tarayıcıların yaratıp bazılarının yaratmadığı metin düğümleri geçebilmeyi sağlar.

```
function get_nextSibling(n)
{
   y=n.nextSibling;
   while (y.nodeType!=1) // element nodes are type 1
   {
      y=y.nextSibling;
   }
   return y;
}
```

DOM Ağacını Modifiye Etmek

Yeni çocuk düğümleri nasıl yaratıp ekleyeceğimizi gördük. Bunlarla aynı zamanda çocuklar yenileriyle değiştirilebilir ve silinebilir.

Her DOM elemanı nesnesinin aynı zamanda şu metotları da vardır:

`elem.appendChild(node)`
- Bir elemana son çocuk olarak bir çocuk düğüm ekle.

`elem.insertBefore(new, old)`
- Bir elemanda «old» çocuk düğümün önüne «new» çocuk düğümünü ekle

`elem.removeChild(node)`
- Elemanın verilen çocuğunu sil.

`elem.replaceChild(new, old)`
- Eski çocuğu yeni çocuk ile değiştir.

jQuery

jQuery Nedir?

Bir JavaScript kütüphanesidir.

JavaScript programlamayı basitleştirir

- DOM modifikasyonu ve gezintisi
  - Bir daha asla `document.getElementById` yazmamıza gerek yok!
- CSS manipülasyonu
- Olay karşılama ve ele alma
- Efektler ve animasyonlar
- Ajax

Tarayıcılar arası uyumluluk sağlar

jQuery Kullanım Oranı

Popularity

In 2015, jQuery was used on 62.7% of the top 1 million websites (according to BuiltWith), and 17% of all Internet websites.

In 2017, jQuery was used on 69.2% of the top 1 million websites (according to Libscore).

In 2018, jQuery was used on 78% of the top 1 million websites.

In 2019, jQuery was used on 80% of the top 1 million websites (according to BuiltWith), and 74.1% of the top 10 million (per W3Techs).

As of Apr 2021, jQuery is used by 77.8% of the top 10 million websites (according to W3Techs).

jQuery Versiyonları

En son versiyonlar: 1.12.4 ve 2.2.4 ve 3.6.0

jQuery.com diyor ki:

- “jQuery 2.x has the same API as jQuery 1.x, but does not support Internet Explorer 6, 7, or 8... Since IE 8 is still relatively common, we recommend using the 1.x version unless you are certain no IE 6/7/8 users are visiting the site.”

2 versiyonu, 1 versiyonundan %10 daha küçüktür

jQuery.com yine diyor ki:

- “If you need support for the widest variety of browsers including IE8, Opera 12, Safari 5, and the like, use the jQuery-Compat 3.0.0 package. We recommend this version for most web sites, since it provides the best compatibility for all website visitors.”
- “If your web site is built only for evergreen leading-edge browsers, or is an HTML-based app contained in a webview (for example PhoneGap or Cordova) where you know which browser engines are in use, go for the jQuery 3.0.0 package.”

jQuery Sözdizimi

Temelde tüm jQuery ifadeleri şu şekildedir:

- `$(something).method(...)`
- veya `jQuery(something).method(...)`

$ fonksiyonu tanımlayan başka kütüphanelerle çakışabilecekse jQuery ifadesini kullanın (noConflict'e göz atın)

Buradaki "something", neredeyse herşey olabilir. CSS seçicisi, HTML parçacığı, nesne, …

- jQuery kodları, argümanın tipine göre ne yapılacağını belirler

Metot, gerçekleştirilecek işlemdir

Örnek:

- `$("#goButton").hide();` —`id="goButton"` olan elemanı gizler.

$(document).ready Fonksiyonu

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

DOM yaratımı gibidir:

```
window.onload = function() {
  statements
};
```

Ancak, daha erken çalışır (örneğin görüntülerin vs. indirilmesini beklemez)

Doküman yaratılana kadar bekler, bu nedenle içindeki elemanlar gerçekten yer alır.

Genellikle daha az okunabilir olan versiyonu tercih edilir:

```
$( function() { statements } );
```

`$(document).ready`'ye ihtiyacımız var mı?

<script> etiketini neden gövdenin sonuna yerleştirmeyelim?

- Bu sayede, tüm DOM elemanlarının yüklenmesini garanti ederiz
- Esasen, bunu yapan pek çok site var

Scriptleri head bloğunda barındırmanın dezavantajları vardır

- Tarayıcı scriptleri çekip HTML içeriğini işleyip oluşturmaya devam etmeden önce işletmelidir.
  - Document.write içine içerik koyulmuş olabilir (her ne kadar artık yapılmasa da, olabilir)

Bu bekleme, sayfa yüklemesini yavaşlatır

Ancak, scriptleri sonda yüklemek de kötüdür

- Tarayıcı, scritpleri indirmeden önce HTML işlenmesi bitmelidir
- Tüm scriptler indirilip çalışana kadar sayfa işlevsizdir
- Sayfa işlenirken asenkron olarak scriptlerin indirilmesi daha iyidir

Modern Çözüm

<script> etiketini head alanına koyup `$(document).ready`'yi kullanın

HTML5 ile gelen async veya defer özelliklerini kullanın

- Pek çok güncel tarayıcı destekler
- Desteklemiyorsa da sorun yaratmaz

```
<script src="mycode.js" async></script>
```

- Tarayıcı, scripti asenkron yükler
- Script yüklenir yüklenmez çalıştırılır

```
<script src="mycode.js" defer></script>
```

- Tarayıcı, scripti asenkron yükler
- Birden fazla script, belirme sırasında çalıştırılır
  - jQuery ile "$ undefined" hatası almamak için bunu kullanın

Daha ayrıntılı inceleme ve tartışma:

- http://stackoverflow.com/questions/436411/where-is-the-best-place-to-put-script-tags-in-html-markup

jQuery Seçicileri

Herhangi bir CSS seçicisi (tırnak içinde) $(…) içine girebilir

- `querySelectorAll` DOM metodu gibidir
- Ancak dizi olup olmamam endişesi olmaz. Otomatik olarak tüm nesnelere etki eder

Örnekler:

- `$("p").hide()` — tüm paragraph elemanlarını gizler.
- `$(".info").show()` —class="info“ olan tüm elemanları gizler.
- `$("#info").text("message")` —id="info“ olan elemanın metnini değiştirir.
- `$("div > p").css("border", "1px solid gray");` — div içinde yer alan tüm paragraflara kenarlık stili ekler

```
(document).ready( function() {
    $("p").click(toggleHighlighting);
} );


function toggleHighlighting()
{
    $(this).toggleClass( "highlighted"
);
}
```

DOM İçeriksel Arama

DOM metotları bir elemanın çocukları içinde aramaya olanak verir.

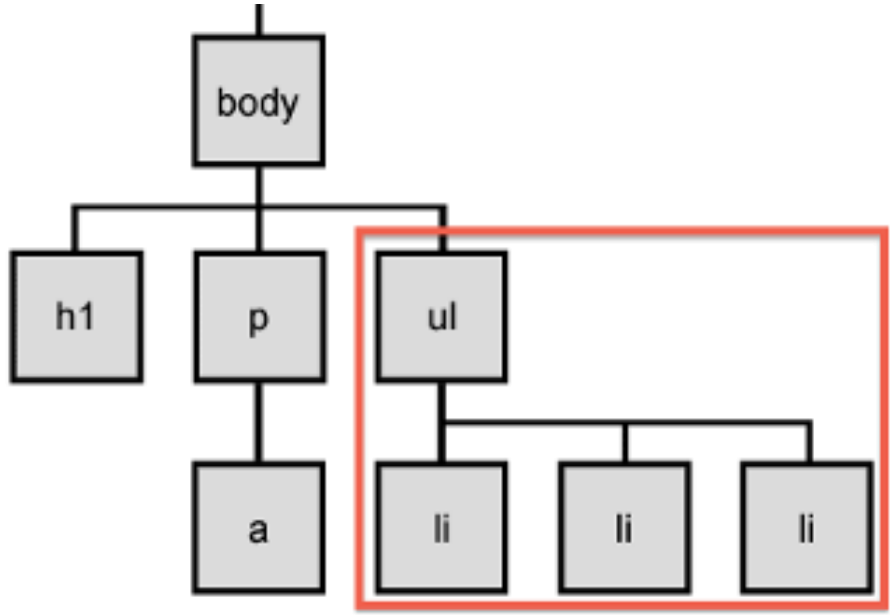
- Örnek: class="foo" olan ve id="bar" olan belirli ul elemanlarının çocuğu olan elemanlar listesini arayalım

DOM yöntemi:

- `var ul = document.getElementById("bar");`
- `var a = ul.querySelectorAll("li.foo");`
- Sonuç: a, istenen elemanların dizisidir

jQuery yöntemi:

- `var ul = $("#bar");`
- `var a = $("li.foo",ul);`



jQuery Nesnesi

$ fonksiyonu her zaman dizi tarzı bir nesne döndürür.

Seçili elemanlar, dizinin gözlerinde yer alır.

Karşılık gelen DOM metotlarından dönen diziler gibi değildir.

Öyleyse;

```
document.getElementById("myid") != $("#myid")
document.getElementById("myid") == $("#myid")[0]

document.getElementsByTagName("p") != $("p")
document.getElementsByTagName("p")[0] == $("p")[0]
```

Nesnelerle jQuery

`$( ... )` Arasına bir nesne girebilir. Ör:

- `$(document)` — document nesnesi
- `$(this)` — bir olay tetikleyen nesne
- `$(myobj)` — kullanıcının tanımladığı bir nesne
- Not: tırnak işareti yok!

Bu sayede bir nesneye «jQuery yükseltmesi» yapmış oluruz.

- Ekstra fonksiyonellik kazanır
- Ancak, dikkat edin: nesne artık bir «jQuery nesnesi»dir
- Orijinal nesne, yeni nesnenin ilk elemanıdır
- Orijinal özelliklerini kullanmanız gerektiğinde, `$(object)[0]` ile erişebilirsiniz.

Yeni Eleman Yaratmak

Yeni eleman yaratmak için `$("<tag>")` kullanılır.

- Ya da isterseniz `$("<tag></tag>")`

Örnek:

```
$("<div>").text("Here's a new div!").appendTo("body");
```

Bu örnek zincirlemeyi de gösterir. Önceki metodun sonucuna yeni bir metot uygulamak.

- Sık kullanılan bir jQuery desenidir.

Gerçekte, < ile başlayan herhangi bir HTML kod parçacığı verilebilir. Ör:

- `$("<div>Here's a <b>new</b> div!</div>").appendTo("body");`
- Bunu karmaşık HTML'de kullanmayın, hataya açıktır.

jQuery DOM İçerik Metotları

Belgenin içeriğini sorgulayıp değiştirmeye yarayan metotlar

Okuyan metotlar:

- `$(elem).text()` — eleman(lar)ın metin içeriğini döndürür
- `$(elem).html()` — etiketler dahil HTML içeriğini döndürür
- `$(elem).val()` — Form alanının değerini döndürür
  - Birden fazla eşleşen sonuç varsa, text metodu birleşik metni, html ve val metotları ilk eşleşeni döndürür

Değiştiren metotlar:

- `$(elem).text("string")` — eleman(lar)ın text içeriğini günceller
- `$(elem).html("string")` — HTML içeriği oluşturur
- `$(elem).val("string")` — Form alanına değer atar

Callback ile içeriği güncellemek

Değiştiren metotlar bir fonksiyona argüman olarak aktarılabilir. Buna callback adı verilir.

Bu fonksiyon, seçili elemanların her biri için tetiklenecektir.

İki argüman alır:

- Seçili elemanların listesinde elemanın indisi
- Değiştirilecek olan orijinal içerik

Fonksiyonun dönüş değeri içerik olarak kullanılır.

```
var itemNames = ["stapler", "pencil box", "notepad", "binder"];

// Attach the click handler

$(document).ready( function() {

    $("#doItButton").click(addItemNames);

} );

// For i'th item, append a colon and the i'th item name in the array to text

function addItemNames() {

    $("ul li").text( function(i,oldText) {

        return oldText + ": " + itemNames[i];

    });

}
```

Özellikleri Okuma/Değiştirme

`$(elem).attr(attr-name)` — ilk eşleşen elemanın verilen özelliğinin değerini döndürür

`$(elem).attr(attr-name,attr-value)` — verilen özelliğin değerini değiştirir

- İkinci argüman olarak bir callback de kullanabilir

```
<img title="hat.jpg">

 <img title="shoe.jpg">

<script>

$( "img" ).attr( "src", function() {     return "resources/" + $( this ).attr("title");  });

 $( "img" ).attr( "alt", function() {     return $( this ).attr("title").split(".")[0]; // get part before the .jpg
});

 </script>
```

Değer almayan özellikleri okuma/değiştirme

`$(elem).prop(prop-name)` — ilk eşleşen eleman için verilen (değer almayan) özelliğin tanımını döndürür

`$(elem).prop(prop-name,prop-value)` — verilen (değer almayan) özelliğin tanımını değiştirir

İkinci argüman olarak bir callback de kullanılabilir

Değer alan (attribute) ve değer almayan (property) özellikler farklıdır. api.jquery.com/prop/ Adresini ziyaret edebilirsiniz

Örneğin bir menünün `selectedIndex` özelliği değer almaz.

Bir seçim kutusu için,

- `.prop("checked")` seçili durumdur, kutu tıklandığında değişir
- `.attr("checked")` ilk halidir, (checked özelliği mevcut anlamında), kutu tıklanınca değeri değişmez

DOM Elemanları Ekleme

`$(selector).append(content)` — seçili eleman(lar)ın sonuna içerik ekler

`$(selector).prepend(content)` — seçili eleman(lar)ın önüne içerik ekler

Bu metotlar, yeni içeriği bir çocuk eleman olarak ekler

- Her metot birden fazla argüman alabilir, her birini ayrı çocuk olarak ekler

Örnek: `$("p").append("Hello");` — Tüm paragrafların sonuna Hello metnini ekler.

`$(selector).after(content)` — Seçili eleman(lar)ın sonrasına içerik ekler

`$(selector).before(content)` — Seçili eleman(lar)ın önüne içerik ekler

Bu metotlar, yeni içeriği kardeş düğüm olarak ekler

- Her metot birden fazla argüman alabilir, her birini ayrı kardeş olarak ekler

Örnek: `$("div").after("<p>Hello</p>");` — her div'i takip eden, Hello içerikli bir paragraf ekler.

Alternatif Metotlar

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

$(content).prependTo(selector)

$(content).insertBefore(selector)

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

Bunlar da önceki slayttaki metotlarla aynı işleri yapar. Ancak, sözdizimi ters çevrilmiştir. Önce içerik, sonra hedef.

Content bir seçici olabilir, bu durumda seçili elemanlar sayfada yeni bir konuma taşınır.

- Hedef seçici birden fazla elemanla eşleşirse, content'in kopyaları ekstra hedefler için de yaratılacaktır.

Elemanları Silmek

`$(selector).remove()` — seçili eleman(lar)ı ve çocuklarını siler

`$(selector).remove(filter)` — Filtre(bir diğer seçici)ye uyan seçili eleman(lar)ı siler

`$(selector).empty()` — seçili eleman(lar)ın tüm çocuklarını siler

Örnek: `class="junk"` olan `div` (ler)i silmek için:

```
$("div.junk").remove();
$("div").remove(".junk");
```

CSS Özelliklerini Değiştirmek

`$(selector).css("property")` — CSS özelliğinin değerini okur

`$(selector).css("property","value")` — CSS özelliğinin değerini değiştirir

- İkinci argüman, indis ve eski CSS özellik değeri argümanlarına sahip bir callback fonksiyonu olabilir

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

- Hatırlayın: `{"name":"value","name":"value",...}` yapısı standart JavaScript nesnesi (JSON) deyimidir.
- Özellik isimleri için, jQuery hem DOM (tırnaksız) e CSS (tırnaklı) şekillerini anlayabilir. Ör:

```
$("body").css({color:"white",backgroundColor:"green"}) veya
$("body").css({"color":"white","background-color":"green"})
```

Tabii ki, dikkati çekmeyen CSS ilkeleri jQuery için de geçerlidir. Tercihen sınıf isimleriyle çalışın.

Sınıfları Değiştirmek

`$(selector).addClass(class-name)` — seçili eleman(lar)a sınıfları ekler

`$(selector).removeClass(class-name)` — seçili eleman(lar)dan sınıfları siler

`$(selector).toggleClass(class-name)` — seçili eleman(lar)da sınıfl tanımlarının durumunu çevirir: yok ise ekler, var ise siler

`$(selector).toggleClass(class-name,addOrRemove)` — boolean addOrRemove true ise sınıfı ekler, false ise siler

Örnek: bir eleman için zincirleme yöntemi ile bir sınıfı kaldırıp başka sınıfı eklemek:

```
$("#feedback").removeClass("low").addClass("high");
```

- Her metot için birden fazla sınıfı aynı anda eklemek/silmek/değiştirmek istiyorsanız, sınıf isimlerini boşluklarla ayrılmış liste olarak verebilirsiniz.

jQuery Olay Metotları

Sık kullanılan olayları karşılamak üzere kendi metotları vardır:

- Mouse düğmesi: `click()`,`dblclick()`
- Mouse hareketi: `mouseenter()`,`mouseleave()`
- Klavye: `keypress()`,`keydown()`,`keyup()`
- Form: `submit()`,`change()`,`focus()`,`blur()`
  - Focus ve blur olayları, alanda klavye aktifliğinin başlaması ve bitmesi ile tetiklenir
- Document/Window: `load(), resize(), scroll(), unload()`
  - `$(window).unload()` Olayı, tarayıcı bir sayfadan ayrılırken tetiklenir

Tüm bu metotlar argüman olarak bir fonksiyon alır ve olay ele alıcısı bu fonksiyondur.

Tüm liste: http://api.jquery.com/category/events

`on()` Fonksiyonu

Önceki sayfadaki tüm metotlar, aslında şunun için kısayoldur:

- `$(elem).on("event",handler);`
- Aynı DOM'daki `addEventListener("event",handler);` gibi.
- Farkı, IE8 gibi bunu desteklemeyen tarayıcılarla da uyumludur

Bu fonksiyonun başka formları da vardır:

- `$(elem).on({"event":handler,"event":handler,...});`
- Eleman için çoklu olaylara karşılayıcı ataması yapar

```
$(document).ready( function () {
   $("div").on( {
      "mouseenter": function () {
         $( this ).text( "Mouse is in" );       },
      "mouseleave": function () {
         $( this ).text( "Mouse is out" );       },
      "click": function () {
         $( this ).text( "You clicked" );       }
      });
});
```

Olay Karşılayıcıların Kaldırılması

İle atanan olaylar, ile kaldırılabilir.

- `$(elem).off("event",handler);`
- Tıpkı DOM `removeEventListener("event",handler);` gibi

Veya;

- `$(elem).off({"event":handler,"event":handler,...});`
- Çoklu olaylardan karşılayıcıları kaldırır

`each()` Fonksiyonu

```
$(selector).each( function )
```

Her seçili elemana fonksiyonu uygular

Fonksiyon iki argüman alır:

- Seçili elemanlar listesindeki elamanın indisi
- Düz DOM nesnesi olarak mevcut eleman
  - `($(this)` jQuery nesnesi de mevcuttur)

jQuery metotlarının tüm seçili elemanlar üzerinde gezdiğine dikkat edin.

- `a.each( function(){ $(this).css("color","blue"); } );` ifadesi şuna eşittir: `a.css( "color","blue" );`
- Bir jQuery metodu ile yapamadığınız bir işi yapmaya ihtiyaç duyduğunuzda `each()` metodunu kullanın

Efektler

jQuery'de görsel efektler için pek çok metot vardır
- Elemanları gizleyip göstermek
- Animasyon
- Yavaş yavaş görünme ve kaybolma, yukarı ya da aşağı kayarak gelme/gitme

Sadece `hide()` `show()` fonksiyonlarına bakacağız.

Daha fazlası:
- http://api.jquery.com/category/effects/.

Elemanları `$(elem).hide()` ile gizlemek
- CSS `display:none` olarak ayarlamak ile aynı etki
- Fakat, önceki değeri hatırlar (ör: `inline` veya `block`)

Önceden gizlenen elemanları `$(elem).show()` ile göstermek:
- Önceki `display` değerini geri oluşturur

AJAX

XMLHttpRequest Nesnesi

Metot ve özellikleri ile AJAX API'sini sağlar

W3C standardı budur

Gerçekte, AJAX'I Microsoft icat etmiştir

- Microsoft.XMLHTTP nesnesine dayanır
- IE olmayan tarayıcılar desteklemez
- XMLHttpRequest, IE7'ye kadar Microsfot tarafından desteklenmemiştir
- IE6 desteklemek için farklı kodlamalar gereklidir
- Neyse ki, her iki nesnenin de metotları aynıdır

Aynı Kaynak Politikası

Güvenlik önlemi olarak, XMLHttpRequest nesnesi sadece şunlardan oluşan aynı kaynaklardan gelen isteklere cevap verir:

- Aynı protocol
- Aynı host
- Aynı port

Örneğin, sayfa http://www.yildiz.edu.tr/~gbiricik/form.php ise

Istekler sadece http://www.yildiz.edu.tr'den olabilir.

Mozilla tanımına bakın: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

İşbirliği içindeki kaynaklar CORS ile (Cross-origin resource sharing) erişime izin verebilir

- Google eklentileri bu şekilde çalışabilir

jQuery AJAX metotları / load

jQuery AJAX için metotlar sunar. Tarayıcı uyumluluğu ile uğraşmayız. İstek-cevap işlemlerini kolaylaştırır. jQuery veya başka bir yazılım çerçevesi kullanın!

```
// Attach handler to the change event for the person_select menu
$(document).ready( function() {
    $("#person_select").change( updateBio );
});
// Handler for change event: use AJAX to load the biographical text into the person_biography div.
function updateBio() {
    var selected_person = $("#person_select option:selected").val(); // selected option value, e.g. "turing"
    // Check for empty string to avoid processing the "Select a person" item
    if( selected_person ) {
        var url = "bios/" + selected_person + ".html"; // construct file path, e.g. "bios/turing.html"
        $( "#person_biography" ).load( url );          // load the content
    }
```

Diğer load opsiyonları

İlave parametreler de sağlanabilir:

```
$(selector).load(url,data,callback)
```

Veri sunucuya istekle birlikte form verisi gibi gönderilir

- Genellikle JSON formatı kullanılır:

```
{ topic: "computers", limit: 25 }
```

- Sunucu, cevabı veriye bağlı olarak özelleştirebilir

Callback fonksiyonu, cevap elemana başarı ile eklendikten sonra çalışır.

XML DOM Gezintisi

XMLHttpRequest `responseXML` nesnesi tıpkı HTML DOM gibi bir belgedir.

Bazı standart JavaScript DOM metot ve özellikleri çalışır:

- `xmlDoc.getElementsByTagName("tag")`
- `node.nodeName, node.nodeValue,`
- `node.childNodes, node.parentNode`
- Ancak, sınıflar, id etiketleri olmadığı için bunlar çalışmaz

Ancak, bunlar düz JS ile karmaşık hale gelir.

- Elemanın içeriği kendisinde değil, metin çocuk düğümünde saklanır:
- `txt=xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue`
- Detaylar için w3schools XML DOM sayfalarına göz atın
- Tercihen jQuery kullanın

Düz metinde ne sorun var?

Düz metin/HTML nedern her zaman yeterli değildir?

Yapılı değildir.

- İçeriğin sadece belirli bir kısmının kulanımı veya,
- Parçalarına özel işlem uygulamak mümkün değildir

Elle geliştirilen yöntemlerle yapı kazandırılabilir ancak,

- Standart olmazlar
- Özel kodlama gerektirir

Daha iyi fikir: XML kullanımı

- Esnek, çok yönlü, yaygın kullanımlı
- Veritabanı sorguları ile PHP gibi dillerle yaratılması kolay
- Düz JS ve jQuery DOM metotları XML ile çalışabilir

jQuery'deki XML DOM Metotları

XML Document nesnesine jQuery güncellemesi yapın:

`$(xmlDoc).find("selector")` çalışır.

- Fakat, `.find(".class")` ve `.find("#id")` çalışmaz.
- Şunlar gibi CSS ilişkisel seçicileri kullanılabilir:
- `.find("name > first")` — ebeveyn-çocuk
- `.find("message[language='english']")` — özelliğe sahip bir eleman

jQuery get/set ve DOM gezinti metotları çalışır:

- `.find("selector").text()`
- `.find("selector").attr("attr-name")`
- `.find("selector").each(callback)`

jQuery AJAX Metotları

XML cevabı ile load fonksiyonu kullanılamaz

- Cevap verisi XML'dir, text/HTML değildir
- XML dokümanını alarak onunla çalışmak gerekir

Bunun için gerekli metotlar:

- `$.ajax(params)` — En genel form
- `$.get(params)` — GET tipi istek için kısayol
- `$.post(params)` — POST tipi istek için kısayol

`params` argümanı bir JSON nesnesidir.

Callback kullanarak cevap işlenir.

jQuery `$.get()` `.post()` Metotları

Küçük bir miktar veri yollanacaksa ya da veri yollanmayacaksa tercih edilebilir.

(Zincirleme yöntem ile) kullanımı:

```
$.get(url)     | $.post(url,data)
  .done( successFunction )
  .fail( failureFunction );
```

`successFunction(data)` Cevap verisini argüman olarak alır

- Metin veya HTML cevabı için, veri düz string formatındadır
- XML cevabı için, veri XML doküman nesnesidir

Istek başarısız olduğunda `failureFunction(xhr,status,errorText)` çağrılır

jQuery `$.ajax()` Metodu

Tam işlevsellik için bunu kullanın. Kullanımı:

`$.ajax(url,settings)` veya

`$.ajax(settings)`

- İkinci formda URL de “setting” değerlerindendir

“Settings” çok seçenekli bir JSON nesnesidir. En sık kullanılanları:

- url: isteğin adresi
- Data: Sunucuya istek ile gönderilecek (JSON tipinde) veri
- Method: istek tipi: GET / POST
- Headers: istekle gidecek ilave başlık bilgileri
- Success: Başarılı cevap karşılayıcısı (ya da .done() kullanın)
- Error: Başarısız cevap karşılayıcısı (ya da .fail() kullanın)

Gelecek Ders

XML, DTD, XSLT