DRUPALIHA.COM

مرجع کامل

براساس مستندات نسخه **1.6.2**

گرد آوری و تالیف:
هیمن حسین پنا
آرمان فیضی

زمستان ۱۳۹۰

Powerd By
Drupaliha.com

jQurey چیست؟

jQeury یک نوع جدید از کتابخانه های جاوا اسکریپت است. jQuery کتابخانه ای سریع و سبک که با قدرت تمام پیمایش اسناد HTML، مدیریت رخدادها ، انیمشن کردن عناصر ، آمایش داده ها و تعامل با AJAX را ارائه می کند.به صورت کلی jQuery برای تغییر در شیوه اسکریپت نویسی جاوا اسکریپت طراحی شده است.

مطالب در یک نگاه

بخش اول:
مبانی jQuery

- **پیش نیازها :**
  - تجربه کاری با جاوا اسکریپت (فراخوانی توابع ، ایجاد اشیا و دستکاری آنها و...)
  - درک کامل HTML (وظیفه jQuery آمایش تگ ها و عناصر HTML است)
  - تجربه کاری با CSS (jQuery امکانات مناسبی برای کار با استایل ها دارد)

jQuery یک کتابخانه رایگان جاوا اسکریپت است که دارای خصوصیاتی همچون پشتیبانی از مرورگرهای مختلف و نتیجه کاری بیشتر در مقابل نوشتن کد کمتر است. کار اصلی آن با انتخاب عناصر HTML آغاز و با رهگیری رخدادها و کارهایی همچون انیمیشن کردن عناصر ، فراخوانی های AJAX ، تغییر استایل ها و... ادامه می یابد.

در یک تعریف کلی jQuery را میتوان اینگونه معرفی کرد :

Free JavaScript library ,Cross Browser, Reduce Code and Write Less ,Do More

▪ ارجاع به کتابخانه jQuery

نخستین کار برای شروع با jQuery دانلود کتابخانه از سایت www.jquery.com و ارجاع به آن در فایل کاری خودمان است.کتابخانه به دوصورت برای دانلود در سایت قابل دسترس است.

Production: این نسخه شامل تمام ویژگی های اصلی کتابخانه با حجم بسیار پایین است.دلیل اصلی حجم پایین این نوع حذف فاصله های خالی و توضیحات در فایل است.

Developer : این نسخه شامل توضیحات کامل و امکان خواندن راحت آسان است ، بالطبع دارای حجم بیشتری است.

توصیه میشود در هنگام توسعه و برنامه نویسی وب جهت راحتی کار و اشکال زدایی سریع تر از نسخه دوم استفاده شده و در هنگام آپلود وب سایت جهت استفاده از نسخه دوم استفاده شود.

پس از دانلود بایستی از طریق دستور زیر در قسمت هدر HTML این دستور اضافه شود.

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

در قسمت src آدرس و نام فایل دانلود شده را می نویسیم.

یکی از گزینه های دیگر ارجاع به کتابخانه استفاده از CDN های شرکت هایی همچون مایکروسافت و گوگل است . این شرکت های فایل کتابخانه را برای استفاده در و CDN خود قرار داده اند که میتوان به جای ارجاع محلی از آنها استفاده کرد.مزیت اصلی این روش کش کردن اسکریپت است . به عنوان مثال فرض کنید سایتی مانند CNN نیز از این کتابخانه استفاده کرده باشد (از طریق CDN) و شما نیز آن سایت را مشاهده کرده باشید ،

اسکریپت مورد نظر کش شده و سریعتر لود خواهد شد.آدرس این CDN ها به شرح زیر است:

- Google Ajax API CDN
  - http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js
- Microsoft CDN
  - http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.6.2.min.js
- jQuery CDN (via Media Temple)
  - http://code.jquery.com/jquery-1.6.2.min.js Minified version
  - http://code.jquery.com/jquery-1.6.2.js Source version

یک نکته حائز اهمیت در استفاده از این CDN ها اگر در لود فایل از طریق آنها مشکلی پیش بیاد اسکریپت لود نخواهد شد. در این صورت از طریق کد زیر میتوان در صورت وجود چنین مشکلی میتوان از دستورات زیر استفاده نمود و آن را به صورت محلی لود کرد.

```
<script type="text/javascript"
src="http://ajax.microsoft.com/ajax/jquery/jquery-
1.4.2.min.js"></script>
<script type="text/javascript">
if (typeof jQuery == 'undefined') {
    document.write(unescape("%3Cscript
src='/js/jquery-1.4.2.min.js'
type='text/javascript'%3E%3C/script%3E"));
}
</script>
```

پس از ارجاع گام بعدی چک کردن لود DOM است .نکته حائز اهمیت در این مورد این است که در رویداد onload که قبلا استفاده می شد لود همه چیز اعم از مطالب و عکس ها مهم بود اما در این کتابخانه منظور از لود شدن آماده شدن درخت DOM قبل از لود تصاویر و ... است.نقطه شروع کار با jQuery چک کردن این موضوع و اجرای دستورات مورد نظر خودمان است.وظیفه متد ready چک کردن این موضوع است.شکل کلی متد به صورت زیر است:

- `$(document).ready(handler)`
- `$().ready(handler) (this is not recommended)`
- `$(handler)`

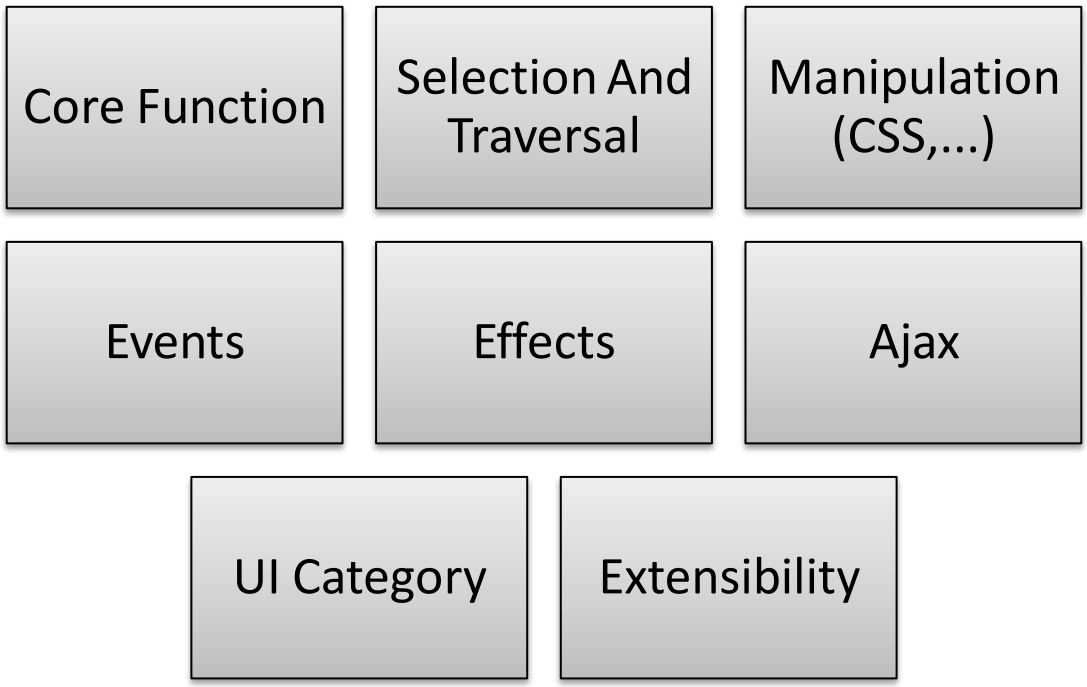
هر سه این موارد معادل هستند. نکته دیگر معادل بودن $ با jQuery است که میتوان از آن نیز استفاده نمود.در زیر مثال های از این متد آمده است :

```
$(document).ready(function() {
  // Handler for .ready() called.
});
$(function() {
 // Handler for .ready() called.
});
```

دستورات زیر متن تگ پاراگراف را تعیین می کند:

```
$(document).ready(function () {
  $("p").text("The DOM is now loaded and can be
manipulated.");
});
```

ویژگی هایjQuery را می توان در ۸ بخش زیر طبقه بندی کرد که در ادامه به معرفی هر کدام از آنها خواهیم پرداخت.



بخش دوم:
انتخاب کننده های jQuery

در قسمت قبل گفته شد که اساس کار jQuery با انتخاب عناصر آغاز می شود.در این قسمت مکانیزم های انتخاب عناصر در jQuery را معرفی خواهیم کرد. این انتخاب کننده ها به ما اجازه انتخاب تگ و عناصر DOM را به ما می دهند. به یاد داشته باشید خروجی اغلب این انتخاب کننده ها آرایه ای از اشیا jQuery میباشد که طول آن ممکن است صفر ، یک یا بیشتر باشد.

- **انتخاب همه عناصر :**

عملگر * جهت انتخاب سراسر و همه عناصر است. این نکته را بخاطر داشته باشید که این عملگر از نظر کارایی کند عمل میکند.شکل کلی آن به صورت زیر است :

```
$('*')
```

مثال :

```
var elementCount = $("*"). length;
```

در مثال بالا طول آرایه که عناصر آن همه محتوی DOM است را در متغیر قرار می دهد.نکته قابل توجه در مورد مثال ها این است که دستورات بکار رفته صرفاً جهت تشریح عملکرد انتخاب کننده هاست با این وجود هرکدام از آنها در ادامه کامل توضیح داده خواهد شد.

- **انتخاب با نام تگ یا عنصر :**

از طریق این انتخاب کننده میتوان تگ یا عنصر خاصی را انتخاب کردم. به صورت کلی همه عناصر با نام تگ داده شده را انتخاب می کند.در حقیقت متد `getElementsByTagName()` جاوا اسکریپت را فراخوانی میکند. شکل کلی آن به فرم زیر است :

```
jQuery('element')
```

مثال : همه عناصر div را انتخاب و یک خصوصیت css آن را ست میکند.

```
$("div").css("border","9px solid red");
```

- **انتخاب با نام ID:**

یک عنصر با ID مورد نظر را انتخاب میکند.در حقیقت این انتخاب کننده متد getElementById() جاوا اسکریپت را فراخوانی می کند که از نظر کارایی بهتر از آن عمل میکند بویژه زمانی که با موارد دیگری همچون h2#pageTitle ترکیب شده باشد.هر عنصری در صفحه بایستی دارای شناسه منحصر به فرد باشد. اگر در DOM چند عنصر با یک شناسه موجود باشد این انتخاب کننده تنها اولین مورد را انتخاب می کند.شکل کلی آن به صورت زیر است :

```
jQuery('#id')
```

مثال :

```
$("#myDiv").css("border","3px solid red");
```

- **انتخاب با نام کلاس :**

عناصرعضو کلاس موردنظر را انتخاب میکند. در حقیقت این انتخاب کنند متد getElementsByClassName() جاوا اسکریپت را فراخوانی می کند. شکل کلی آن به صورت زیر است :

```
jQuery('.class')
```

مثال :

```
$(".myClass").css("border","3px solid red");
```

- **انتخاب فرزندان یک عنصر :**

برای انتخاب فرزندان مستقیم یک پدر از شکل کلی زیر استفاده می شود :

```
jQuery('parent > child')
```

مثال : حاشیه ای را دور عناصر فرزند <ul class=”top”> میکشد.

```
$("ul.top > li").css("border", "3px double red");
```

- **انتخاب نسل های یک عنصر :**

برای انتخاب نسل های یک عنصر از شکل کلی زیر استفاده می شود:

```
jQuery('ancestor descendant')
```

مثال :انتخاب همه نسل های ازنوع input که فرزند فرم هستند.

```
$("form input").css("border", "2px dotted blue");
```

- **انتخاب نسل بعدی در یک سیبل :**

عنصر بعدی در درخت که والدشان یکی است را برمیگرداند.

```
jQuery('prev + next')
```

عنصر بعدی در درخت که والدشان یکی است را برمیگرداند.

- **انتخاب فرزند بعدی در یک سیبل :**

```
jQuery('prev ~ siblings')
```

- **انتخاب چندگانه :**

برای استفاده از چندین انتخاب کننده و ترکیب نتایج به صورت زیر عمل میکنیم :

```
jQuery('selector1, selector2, selectorN')
```

مثال : انتخاب چندین عنصر و اعمال استایل :

```
$("div,span,p.myClass").css("border","3px solid
red");
```

- **انتخاب بر اساس صفت های یک عنصر :**

برای انتخاب از طریق صفت های یک عنصر روش های متفاوتی وجود دارد که شکل کلی هریک از آنها به شرح زیر است :

```
jQuery('[attribute]')
```

عناصری که داری صفت مشخص شده هستند را انتخاب می کند.

```
jQuery('[attribute="value"]')
```

عناصری که داری صفت مشخص شده و مقداری مساوی با مقدار مشخص شده هستند را انتخاب می کند.

```
jQuery('[attribute!="value"]')
```

عناصری که داری صفت مشخص شده و مقداری مساوی با مقدار مشخص شده نیست را انتخاب می کند.

```
jQuery('[attribute*="value"]')
```

عناصری که داری صفت مشخص شده و مقداری شامل (حاوی) مقدار مشخص شده هستند را انتخاب می کند.

```
jQuery('[attribute^="value"]')
```

عناصری که داری صفت مشخص شده و مقدار آن با مقدار مشخص شده شروع می شوند را انتخاب می کند.

```
jQuery('[attribute$="value"]')
```

عناصری که داری صفت مشخص شده و مقدار آن به مقدار مشخص شده ختم می شوند را انتخاب می کند.

```
jQuery('[attribute|="value"]')
```

عناصری که داری صفت مشخص شده و مقدار آن برابر با مقدار مشخص شده و یا با آن مقدار و سپس بدنبال آن (-) شروع شود را انتخاب می کند.

```
jQuery('[attribute~="value"]')
```

عناصری که داری صفت مشخص شده و مقدار آن شامل مقدار یا درمیان کلمات آن باشد(که با فاصله از هم جدا شده اند) را انتخاب می کند.

مثال : در میان تگ های input تگی را که دارای صفت نام و مقداری شامل کلمه man است را یافته و مقدار آن را ست میکند

```
$('input[name~="man"]').val('mr. man is in it!');
```

- **انتخاب چندگانه بر اساس صفت ها :**

برای استفاده از چندین انتخاب کننده و ترکیب نتایج به صورت زیر عمل میکنیم :

```
jQuery('[attributeFilter1][attributeFilter2][attribut
eFilterN]')
```

مثال : انتخاب همه عناصر ورودی با خصوصیت id و خصوصیت name که به man ختم می شود. :

```
$('input[id][name$="man"]').val('only this one');
```

- **انتخاب بر اساس عناصر فرم:**

انتخاب همه هدرها (..h1 ,h2)

```
jQuery(':header')
```

jQuery راه هایی را برای انتخاب عناصر فرم بر اساس نوع فیلد قرار داده است که به شرح زیر است:

```
jQuery(':text')
```

همه عناصر textbox موجود در فرم را انتخاب میکند.کار آن معادل حالت `$('[type=text]')` است.

```
jQuery(':submit')
```

همه عناصر submit را در فرم انتخاب میکند .

```
jQuery(':reset')
```

همه عناصر reset را در فرم انتخاب میکند .

```
jQuery(':selected')
```

این انتخاب کننده برای کار با <option> است و همه عناصر انتخاب شده را بر میگرداند.

```
jQuery(':radio')
```

دکمه های رادیویی را در فرم انتخاب میکند .

```
jQuery(':password')
```

همه عناصر از نوع password را انتخاب می کند

```
jQuery(':visible')
```

همه عناصر visible را انتخاب میکند. عناصر میتوانند در شرایطی مانند خصوصیت css به صورت display:none یا طول و عرض صفر و یا input های مخفی و یا مخفی بودن پدر پنهان شود.

```
jQuery(':hidden')
```

همه عناصر مخفی را انتخاب میکند.

```
jQuery(':input')
```

همه عناصر ورودی را انتخاب میکند.

```
jQuery(':image')
```

همه عناصر تصویر را انتخاب میکند.

```
jQuery(':focus')
```

عنصری که دارای فوکاس است را برمیگرداند.

```
jQuery(':file')
```

همه عناصر از نوع فایل (آپلود) را انتخاب میکند.

```
jQuery(':button')
```

همه عناصر دکمه راانتخاب میکند.

```
jQuery(':checkbox')
```

همه عناصر چک باکس را انتخاب می کند.

```
jQuery(':checked')
```

همه عناصر انتخاب شده در چک باکس و دکمه های رادیویی را برمیگرداند.

```
jQuery(':disabled')
```

همه عناصر غیرفعال را برمیگرداند.

```
jQuery(':enabled')
```

همه عناصر فعال را برمیگرداند.

- **انتخاب بر اساس درخت DOM:**

این گروه از انتخاب کننده ها بر اساس ساختار درخت DOM عمل میکند.

```
jQuery(':parent')
```

همه عناصر شامل فرزند اعم از متن را انتخاب میکند.

```
jQuery(':empty')
```

همه عناصر بدون فرزند را بر میگرداند.

```
jQuery(':only-child')
```

همه عناصری که تنها فرزند والدشان هستند را انتخاب میکند.

```
jQuery(':last-child')
```

همه عناصری که آخرین فرزند والدشان هستند را می یابد.

```
jQuery(':first-child')
```

همه عناصری که اولین فرزند والدشان هستند را می یابد.

```
jQuery(':even')
```

عناصر زوج را برمیگرداند.عناصر از صفر اندیس گزاری می شوند.

```
jQuery(':odd')
```

عناصر فرد را برمیگرداند.عناصر از صفر اندیس گزاری می شوند.

```
jQuery(':nth-child(index/even/odd/equation)')
```

انتخاب n امین عنصر از فرزندان یک عنصر . در شکل کلی index شماره اندیس عنصر ، even عناصر زوج ،odd عناصر فرد و equation معادله ای به فرم Mx+n

- **سایر انتخاب کننده ها:**

```
jQuery(':animated')
```

انتخاب همه عناصری که در زمان انتخاب در حال انیمیشن هستند

```
jQuery(':first')
```

در میان عناصر مطابق با انتخاب کننده اولین را بر میگرداند.

```
jQuery(':has(selector)')
```

انتخاب عناصری که حداقل دارای یک عنصر در میان انتخاب کننده selector هستند.

```
jQuery(':contains(text)')
```

انتخاب همه عناصری که شامل text مورد نظر هستند.

```
jQuery(':not(selector)')
```

انتخاب عناصری که در یک انتخاب کننده وجود ندارند.

```
jQuery(':eq(index)')
```

انتخاب عنصری با اندیس index در لیست مطابق ها

```
jQuery(':gt(index)')
```

انتخاب عناصری که اندیسشان از index بزرگتر است .

```
jQuery(':it(index)')
```

انتخاب عناصری که اندیسشان از index کوچکتر است .

```
jQuery(':last')
```

آخرین عنصر موجود در لیست مطابق ها بر میگرداند.

مثال :آخرین سطر جدول را استایل دهی میکند.

```
$("tr:last").css({backgroundColor: 'yellow',
fontWeight: 'bolder'});
```

بخش سوم:
پیمایش اشیای انتخاب شده
توسط jQuery

در بخش قبل چگونگی انتخاب عناصر مورد نظر خود را در میان همه عناصر صفحه آموختیم . بر اساس آنچه که گفته شد ما از قصل قبل آرایه ای از عناصر jQuery را در اختیار داریم .در این بخش قصد داریم چگونگی پیمایش این آرایه از طریق توابع کتابخانه ای در دسترس مورد بررسی قرار دهیم.برخی از این توابع بر اساس درخت ، برخی بر اساس DOM و برخی نیز داده ها را فیلتر می کنند.در ادامه هریک از این توابع را تشریح خواهیم کرد.

- ***.add()***

این تابع عنصری را مجموعه عناصر مطابق اضافه می کند. شکل کلی این تابع به فرم زیر است.

```
.add( selector )
.add( elements )
.add( html )
.add( jQuery object )
.add( selector, context )
```

این تابع بر اساس پارامترهای ارسال شده شی جدیدی ساخته و آن را برمیگرداند.

مثال :

```
$("p").add("div").addClass("widget");
var pdiv = $("p").add("div");
$("div").css("border", "2px solid red")
          .add("p")
          .css("background", "yellow");
```

- ***.andSelf()***

مجموعه عناصر قبلی را به پشته عناصر جاری اضافه می کند.شکل کلی آن به فرم زیر است :

.andSelf()

مثال : انتخاب همه div **ها و پاراگراف های داخل آن و ست کردن کلاس برای آنها**

```
$("div").find("p").andSelf().addClass("border");
$("div").find("p").addClass("background");
```

- ***.children()***

فرزندان هریک از عناصر موجود در مجموعه مطابق را برمیگرداند.پارامتر اختیاری selector **نیز قابل ارسال به تابع است.شکل کلی آن به فرم زیر است:**

```
.children( [selector] )
```

مثال :

```
$('ul.level-2').children().css('background-color',
'red');
```

مثال : یافتن همه فرزندان عناصر کلیک شده

```
$("#container").click(function (e) {
    $("*").removeClass("hilite");
    var $kids = $(e.target).children();
    var len = $kids.addClass("hilite").length;

    $("#results span:first").text(len);
    $("#results span:last").text(e.target.tagName);

    e.preventDefault();
    return false;
});
```

- ***.closest()***

نخستین پدر که با انتخاب کننده مطابق است را گرفته و با شروع از عنصر جاری درخت DOM را پیمایش میکند.شکل کلی این دستور به صورت زیر است:

```
closest( selector  )

    .closest( selector )

    .closest( selector, [context] )

    .closest( jQuery object )

    .closest( element )

closest( selectors, [ context ]  )

    .closest( selectors, [context] )
```

- ***.content()***

فرزندان هر عنصر مجموعه مطابق شامل متن و عناصر توضیحات را برمی گرداند. شکل کلی آن به صورت زیر است :

```
.contents()
```

مثال: یافتن همه عناصر متنی در پاراگراف و bold کردن آن :

```
$("p").contents().filter(function(){
return this.nodeType != 1; })
.wrap("<b/>");
```

- ***.each()***

یکی از توابع پرکاربرد در jQuery است.در میان عناصر jQuery تکرار کرده و تابعی را بر روی هر کدام از آنها اجرا می کند.شکل کلی به صورت زیر است:

```
.each( function(index, Element) )
```

index اندیس عنصر جاری و element خود عنصر است.در درون تابع با استفاده از this نیز میتوان به شی جاری دسترسی پیدا کرد.

مثال :

```
$('li').each(function(index) {
    alert(index + ': ' + $(this).text());
  });
```

- ***.end()***

آخرین عملیات فیلتر را در زنجیر جاری خاتمه داده و مجموعه عناصر مطابق حالت قبلی را برمیگرداند.خروجی شی قبلی jQuery است. شکل کلی به صورت زیر است:

```
.end()
```

مثال :انتخاب همه عناصر پاراگراف و یافتن span میان آنها و سپس بازگرداندن انتخاب به پاراگراف

```
$("p").find("span").end().css("border", "2px red
solid");
```

- ***.eq()***

عناصر مجموعه مطابق را با اندیس ذکر شده کاهش می دهد.شکل کلی آن به صورت زیر است:

```
.eq(index)
.eq(-index)
```

حالت دوم عددی است که نشان دهنده موقعیت عنصر است از آخرین عنصر تا این عنصر شمرده و برمیگرداند.

مثال :

```
$('li').eq(5).css('background-color', 'red');
```

- ***.filter()***

مجموعه عناصر مطابق را بر اساس انتخاب کننده دیگری فیلتر میکند.شکل کلی آن به صورت زیر است.

```
.filter( selector )
.filter( element )
.filter( function(index) )
```

تابعی است که بر روی همه عناصر تست خاصی انجام می دهد

```
.filter( jQuery object )
```

مثال :

```
$('li').filter(function(index) {
  return $('strong', this).length == 1;
}).css('background-color', 'red');
```

- ***.find()***

نسلی از هر عنصر مجموعه مطابق را گرفته و با انتخاب کننده فیلتر میکند.شکل کلی آن به صورت زیر است:

```
.find( selector )
.find( jQuery object )
.find( element )
```

مثال:

```
$("p").find("span").css('color','red');
```

- ***.first()***

مجموعه عناصر مطابق را به اولین عنصر تقلیل می دهد.شکل کلی آن به صورت زیر است .

```
.first()
```

مثال :

```
$('li').first().css('background-color', 'red');
```

- ***.last()***

مجموعه عناصر مطابق را به آخرین عنصر تقلیل می دهد.شکل کلی آن به صورت زیر است .

```
.last()
```

مثال :

```
$('li').last().css('background-color', 'red');
```

- ***.has()***

مجموعه عناصر مطابق را به عناصری که دارای شرط مورد نظر هستند تقلیل می دهد.شکل کلی آن به صورت زیر است .

```
.has( selector )
.has( contained )
```

پارامتر حالت دوم یک عنصر DOM است.

مثال :

```
$("ul").has("li").addClass("full");
```

- ***.is()***

براساس پارامتر ارسال شده چک میکند تا مشخص کند دارای شرایط مذکور هست یا نه. خروجی آن بولی است. شکلی کلی آن به صورت زیر است:

```
.is( selector )
.is( function(index) )
.is( jQuery object )
.is( element )
```

مثال :

```
$("li").click(function() {
  var $li = $(this),
    isWithTwo = $li.is(function() {
      return $('strong', this).length === 2;
    });
  if ( isWithTwo ) {
    $li.css("background-color", "green");
  } else {
    $li.css("background-color", "red");
  }
});
```

- ***.map()***

هریک از عناصر موجود در مجموعه مطابق را از تابعی عبور داده و شی جدید شامل نتیجه را بر میگرداند. شکلی کلی آن به فرم زیر است :

```
.map( callback(index, domElement) )
```

مثال: ایجاد لیستی از همه عناصر موجود در فرم

```
$("p").append( $("input").map(function(){
      return $(this).val();
    }).get().join(", ") );
```

مثال : انتخاب کردن همه چک باکس ها

```
$(':checkbox').map(function() {
return this.checked = true;
});
```

- ***.next()***

عنصر بعدی را در سیبل بر میگرداند. شکل کلی آن به صورت زیر است :

```
.next( [selector] )
```

- ***.nextAll()***

عناصر بعدی را در سیبل بر میگرداند. شکل کلی آن به صورت زیر است :

```
.nextAll( [selector] )
```

- ***.nextUntil()***

عناصر بعدی در سیبل را تارسیدن به عنصری که در انتخاب کننده نباشد برمیگرداند:

```
.nextUntil( [selector], [filter] )
.nextUntil( [element], [filter] )
```

پارامتر اول شرط توقف و پارامتر دوم شرط انتخاب برای مقایسه هستند.هردو انتخابگر هستند.

مثال :

```
$("#term-2").nextUntil("dt")
.css("background-color", "red");
var term3 = document.getElementById("term-3");
$("#term-1").nextUntil(term3, "dd")
.css("color", "green");
```

- ***.prev()***

عنصر قبلی را در سیبل بر میگرداند. شکل کلی آن به صورت زیر است :

```
.next( [selector] )
```

- ***.prevAll()***

عناصر قبلی را در سیبل بر میگرداند. شکل کلی آن به صورت زیر است :

```
.prevAll( [selector] )
```

- ***.prevUntil()***

عناصر قبلی در سیبل را تارسیدن به عنصری که در انتخاب کننده نباشد برمیگرداند:

```
.prevUntil( [selector], [filter] )
.prevUntil( [element], [filter] )
```

پارامتر اول شرط توقف و پارامتر دوم شرط انتخاب برای مقایسه هستند.هردو انتخابگر هستند.

- ***.not()***

عناصری را از مجموعه مطابق ها حذف میکند.شکل کلی آن بدین صورت است :

```
not( selector )
not( elements )
.not( function(index) )
```

مثال : اضافه کردن حاشیه به div هایی که سبز یا آبی نیستند:

```
$("div").not(".green, #blueone")
            .css("border-color", "red");
```

- ***.offsetParent()***

نزدیک ترین والدی که عنصر در آن قرار دارد را بر میگرداند:

```
.offsetParent()
```

- ***.parent()***

والد هریک از عناصر موجود در مجموعه مطابق را برمیگرداند.شکلی آن به صورت زیر است :

```
.parent( [selector] )
```

مثال :

```
$("p").parent(".selected").css("background",
"yellow");
```

- ***.parents()***

والدین هریک از عناصر موجود در مجموعه مطابق را برمیگرداند.شکلی آن به صورت زیر است :

```
.parents( [selector] )
```

مثال :

```
var parentEls = $("b").parents()
            .map(function () {
                  return this.tagName;
                })
            .get().join(", ");
$("b").append("<strong>" + parentEls + "</strong>");
```

- ***.parentsUntil()***

والدین عناصر موجود در مجموعه را با این شرط که در انتخاب کننده نباشند را برمیگرداند:

```
.parentsUntil( [selector], [filter] )
.parentsUntil( [element], [filter] )
```

- ***.siblings()***

سیبل ها (هم جد ها) را در مجموعه مطابق برمیگرداند :

```
.siblings( [selector] )
```

مثال :

```
$("p").siblings(".selected").css("background",
"yellow");
```

- ***.slice()***

عناصر مجموعه مطابق را با استفادع از اندیس های ارسال شده کاهش می دهد.

```
.slice( start, [end] )
```

مثال پر کردن Div ها به صورت تصادفی

```
function colorEm() {
      var $div = $("div");
```

```
    var start = Math.floor(Math.random() *
    $div.length);
    var end = Math.floor(Math.random() *
   ($div.length - start)) + start + 1;
    if (end == $div.length) end = undefined;
    $div.css("background", "");
    if (end)
   $div.slice(start, end).css("background", "yellow");
     else
      $div.slice(start).css("background", "yellow");
    $("span").text('$("div").slice(' + start +
     end ? ', ' + end : '') +
').css("background","yellow");');
  }
  $("button").click(colorEm);
```

بخش چهارم:
کار با CSS

بر اساس آنچه که تابحال گفته شد کار jQuery با انتخاب عناصرمورد انتظار آغار شده و سپس در ادامه بر روی این عناصر عملیات مورد نظر خود را اجرا می کنیم.در این بخش فرض می شود با استفاده از ابزارهای معرفی شده در فصل های قبل ، لیست یا مجموعه عناصر دلخواه یا مورد نظر انتخاب شده و موجود است.در این بخش قصد داریم تا عملیات مربوط به استایل ها و CSS را معرفی کنیم.توابع ارائه شده به این منظور به شرح زیر می باشند.

- ***.addClass()***

کلاس یا کلاس هایی را به عنصر موردنظر اضافه میکند.شکل کلی این تابع به صورت زیر است :

```
.addClass( className )
.addClass( function(index, currentClass) )
```

در حالت اول نام کلاس به صورت رشته برای تابع ارسال می شود .اگر چند کلاس باشند با فاصله از هم جدا می شوند.در حالت دوم تابع وظیفه تولید نام کلاس را برعهده دارد.اندیس ارسال شده موقعیت عنصر در مجموعه را نشان می دهد.

مثال:

```
$("ul li:last").addClass(function() {
  return "item-" + $(this).index();
});
```

مثال :

```
$("p:last").addClass("selected");
```

- ***.removeClass()***

کلاس یا کلاس هایی را از عنصر موردنظر حذف میکند.شکل کلی این تابع به صورت زیر است:

```
.removeClass( className )
.removeClass( function(index, currentClass) )
```

در حالت اول نام کلاس به صورت رشته برای تابع ارسال می شود .اگر چند کلاس باشند با فاصله از هم جدا می شوند.در حالت دوم تابع وظیفه تولید نام کلاس را برعهده دارد.اندیس ارسال شده موقعیت عنصر در مجموعه را نشان می دهد.

مثال:

```
$("ul li:last").removeClass(function() {
  return "item-" + $(this).index();
```

```
});
```

مثال :

```
$("p:last").removeClass("selected");
```

- ***.toggleClass()***

کلاس یا کلاس هایی را از عنصر موردنظر بر اساس پارامتر ارسالی حذف یا اضافه میکند.شکل کلی آن به صورت زیر است :

```
.toggleClass( className )
.toggleClass( className, switch )
```

پارامتر سوئیچ که بولی است حذف یا اضافه شدن کلاس را تعیین میکند.

```
.toggleClass( function(index, class, switch),
[switch] )
```

مثال :

```
$('div.foo').toggleClass(function() {
  if ($(this).parent().is('.bar')) {
    return 'happy';
  } else {
    return 'sad';
  }
```

مثال :

```
  $(this).toggleClass("highlight");
```

- ***.css()***

این تابع به صورت خواندنی /نوشتنی استایل خاصی را ست یا برمیگرداند. شکل کلی آن به صورت زیر است:

```
.css( propertyName )

.css( propertyName, value )

.css( propertyName, function(index, value) )

.css( map )
```

مثال :

```
var color = $(this).css("background-color");
```

مثال :

```
$("#box").one( "click", function () {
    $( this ).css( "width","+=200" );
  });
```

مثال :

```
$("p").hover(function () {
    $(this).css({'background-color' : 'yellow',
'font-weight' : 'bolder'});
  }, function () {
    var cssObj = {
      'background-color' : '#ddd',
      'font-weight' : '',
      'color' : 'rgb(0,40,244)'
    }
    $(this).css(cssObj);
  });
```

- ***.hasClass()***

عنصر جاری برای داشتن کلاسی خاص چک میکند . خروجی آن بولی است.شکل کلی آن به صورت زیر است:

```
.hasClass( className )
```

مثال :

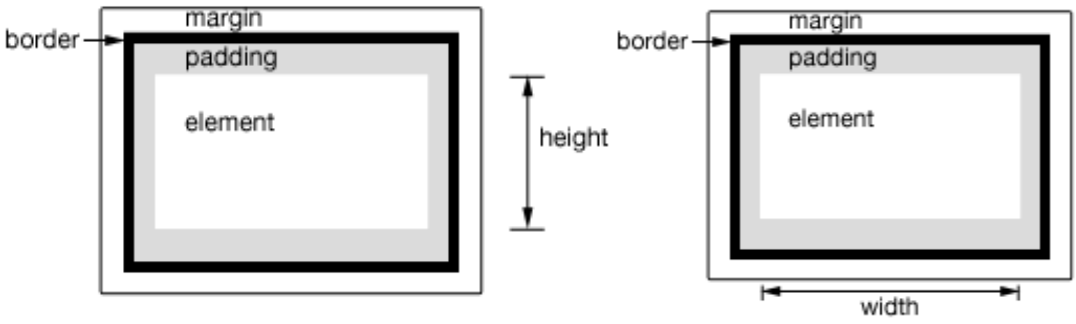
```
$('#mydiv').hasClass('bar')
```

- ***.height()***
- ***.width()***

این توابع به صورت خواندنی و نوشتنی طول و عرض عنصری را ست یا برمیگردانند.شکل کلی آنها به صورت زیر است :

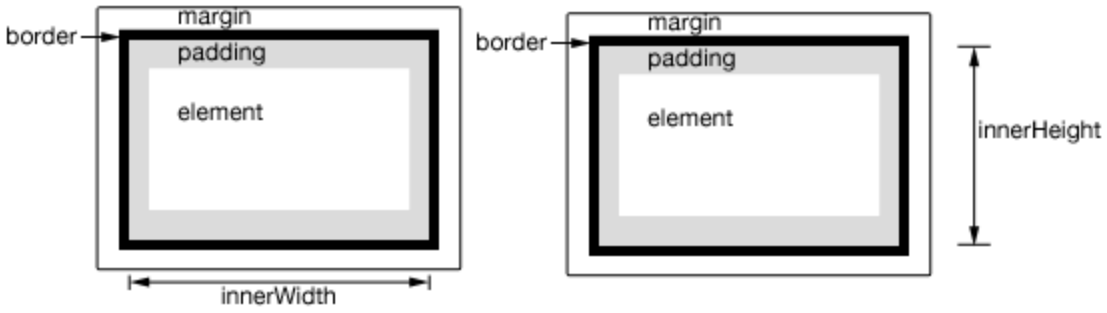
```
.height()
.height ( value )
.height( function(index, width) )

.width()
.width( value )
.width( function(index, width) )
```

تصاویر زیر مقادیر برگشتی را به خوبی نشان داده است :



- ***.innerHeight()***
- ***.innerWidth()***

فاصله اولین عنصر مجموعه شامل pad و بدون border را برمیگرداند . در تصویر مقدار کاملا مشخص است :

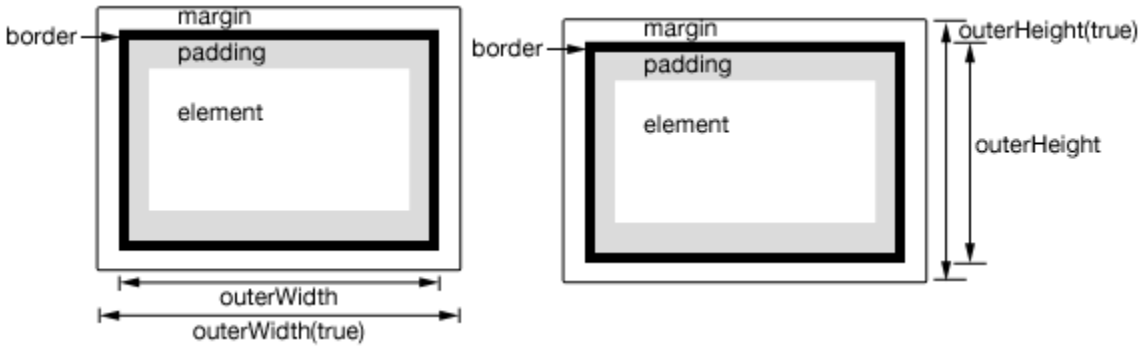


شکل کلی این توابع به صورت زیر است :

```
.innerHeight()
.innerWidth()
```

- ***.outerHeight()***
- ***.outerWidth()***

فاصله اولین عنصر مجموعه شامل pad و border و در صورت انتخاب margin را برمیگرداند . در تصویر مقدار کاملا مشخص است :



شکل کلی این توابع به صورت زیر است :

```
.outerHeight([includeMargin])
.outerWidth([includeMargin])
```

- ***.offset()***

مختصات اولین عنصر موجود در مجموعه را ست یا برمیگرداند.شکل کلی آن به صورت زیر است :

```
.offset()

.offset( coordinates )
```

مختصات عنصر شامل top و left

```
.offset( function(index, coords) )
```

مثال :

```
var offset = p.offset();
p.html( "left: " + offset.left + ", top: " + offset.top
);
```

مثال :

```
$("p:last").offset({ top: 10, left: 30 });
```

- ***.position()***

موقعیت نخستین عنصر مجموعه را نسبت به آفست والدش برمیگرداند.شکل کلی آن به صورت زیر است.

```
.position()
```

مثال :

```
var position = p.position();
$("p:last").text( "left: " + position.left + ", top:
" + position.top );
```

- ***.scrollLeft()***
- ***.scrollTop()***

موقعیت افقی یا عمودی اسکرول بار اولین عنصر مجموعه را گرفته یا ست می کند.شکل کلی این توابع به صورت زیر است :

```
.scrollLeft()
.scrollLeft( value )
.scrollTop()
.scrollTop( value )
```

مثال :

```
$("p:last").text( "scrollTop:" + p.scrollTop() );
```

مثال :

```
$("div.demo").scrollLeft(300);
```

بخش پنجم:
آمایش اشیای انتخاب شده
توسط jQuery

بر اساس آنچه که تابحال گفته شد کار jQuery با انتخاب عناصرمورد انتظار آغار شده و سپس در ادامه بر روی این عناصر عملیات مورد نظر خود را اجرا می کنیم.در این بخش فرض می شود با استفاده از ابزارهای معرفی شده در فصل های قبل ، لیست یا مجموعه عناصر دلخواه یا مورد نظر انتخاب شده و موجود است.در این بخش بیشتر در مورد خود عناصر و اضافه یا حذف کردن عنصر بحث خواهیم کرد.توابعی که jQuery برای آمایش داده ها ارائه کرده است به صورت زیر است...

- **آمایش صفت ها :**

  - ***.attr()***

صفتی را به مجموعه عناصر اضافه کرده یا صفت نخستین عنصر را برمیگرداند.

  - ***.removeAttr()***

صفت خاصی را از همه عناصر پاک میکند.

شکل کلی این دستورات به صورت زیر است.

```
.attr( attributeName )

.attr( attributeName, value )

.attr( attributeName, function(index, attr) )

.removeAttr( attributeName )
```

مثال :

```
var title = $("em").attr("title");
  $("div").text(title);
```

مثال :

```
$("img").attr({
  src: "/images/hat.gif",
  title: "jQuery",
  alt: "jQuery Logo"
});
$("div").text($("img").attr("alt"));
```

در این مثال از حالت map که از اشیا JSON استفاده میکند بهره گرفته شده است.

مثال :

```
$("button:gt(1)").attr("disabled","disabled");
```

مثال :

```
$("button").click(function () {
  $(this).next().removeAttr("disabled")
  .focus()
  .val("editable now");
});
```

- ***.prop(propertyName)***

خصوصیتی را به مجموعه عناصر اضافه کرده یا خصوصیت نخستین عنصر را برمیگرداند.

- ***.removeProp()***

خصوصیت خاصی را از همه عناصر پاک میکند.

شکلی کلی این دستورات به شکل زیر است.

```
.prop( propertyName )

.prop( propertyName, value )

.prop( map )

.prop( propertyName, function(index,
oldPropertyValue)

.removeProp( propertyName )
```

مثال :

```
$("input").change(function() {
  var $input = $(this);
  $("p").html(".attr('checked'): <b>" +
$input.attr('checked') + "</b><br>"
              + ".prop('checked'): <b>" +
$input.prop('checked') + "</b><br>"
              + ".is(':checked'): <b>" +
$input.is(':checked') ) + "</b>";
}).change();
```

مثال :

```
$("input[type='checkbox']").prop({
  disabled: true
});
```

مثال :

```
var $para = $("p");
$para.prop("luggageCode", 1234);
$para.append("The secret luggage code is: ",
String($para.prop("luggageCode")), ". ");
$para.removeProp("luggageCode");
$para.append("Now the secret luggage code is: ",
String($para.prop("luggageCode")), ". ");
```

- ***.html()***

محتوی html **یک عنصر را برمیگرداند یا ست میکند. شکل کلی این دستور به صورت زیر است :**

```
.html()

.html( htmlString )

.html( function(index, oldhtml) )
```

مثال :

```
$("p").click(function () {
      var htmlStr = $(this).html();
      $(this).text(htmlStr);
    });
```

مثال:

```
$("div").html("<span class='red'>Hello
<b>Again</b></span>");
```

مثال :

```
$("div").html("<b>Wow!</b> Such excitement...");
    $("div b").append(document.createTextNode("!!!"))
     .css("color", "red");
```

- ***.val()***

مقدار عنصر جاری را برگردانده یا ست میکند.(مقدار عنصر اول را برمیگرداند ویا همه عناصر را ست میکند).

شکل کلی به صورت زیر است:

```
.val()

.val( value )

.val( function(index, value) )
```

مثال :

```
function displayVals() {
      var singleValues = $("#single").val();
      var multipleValues = $("#multiple").val() ||
[];
      $("p").html("<b>Single:</b> " +
                  singleValues +
                  " <b>Multiple:</b> " +
                  multipleValues.join(", "));
    }

    $("select").change(displayVals);
    displayVals();
```

مثال :

```
$("input").keyup(function () {

      var value = $(this).val();

      $("p").text(value);

    }).keyup();
```

مثال :

```
$("button").click(function () {
      var text = $(this).text();
```

```
    $("input").val(text);
  });
```

- اضافه و حذف کردن عناصر

- ***.append()***
- ***.appendTo()***

تابع اول محتویی را به هریک از عناصر مطابق اضافه می کند. (به انتهای عنصر)

همه عناصر مطابق را به انتهای عنصر موردنظر اضافه می کند.

.شکل کلی آن به صورت زیر است :

```
.append( content, [content] )
.append( function(index, html) )
.appendTo( target )
```

مثال :

```
$("p").append("<strong>Hello</strong>");
```

مثال :

```
$("p").append( $("strong") );
```

مثال :

```
$("span").appendTo("#foo");
```

- ***.prepend()***
- ***.prependTo()***

تابع اول محتویی را به هریک از عناصر مطابق اضافه می کند. (به ابتدای عنصر)

همه عناصر مطابق را به ابتدای عنصر موردنظر اضافه می کند.

.شکل کلی آن به صورت زیر است :

```
.prepend( content, [content] )
.prepend( function(index, html) )
.prependTo( target )
```

مثال :

```
$("p").prepend("<b>Hello </b>");
```

مثال :

```
$("p").prepend( $("b") );
```

مثال :

```
$("span").prependTo("#foo");
```

- ***.after()***
- ***.before()***

محتوی مورد نظر را به قبل و بعد از همه عناصر مطابق اضافه می کنند.شکل کلی به صورت زیر است :

```
.after( content, [content] )
.after( function(index) )
.before( content, [content] )
.before( function )
```

مثال :

```
$("p").before("<b>Hello</b>");
```

مثال:

```
$("p").before( $("b") );
```

مثال :

```
$("p").after("<b>Hello</b>");
```

- ***.insertAfter()***
- ***.insertBefore()***

همه عناصر موجود در مجموعه مطابق را به قبل یا بعد target اضافه می کند.شکل کلی بدین صورت است:

```
.insertBefore( target )
.insertAfter( target )
```

مثال :

```
$("p").insertAfter("#foo");
```

مثال :

```
$("p").insertBefore("#foo");
```

- ***.clone()***

یک کپی عمقی از اشیا ایجاد میکند.

```
.clone( [withDataAndEvents] )
.clone( [withDataAndEvents], [deepWithDataAndEvents]
)
```

پارامتر اول تعیین کننده کپی داده ها و رخدادهاست.پارامتر دوم تعیین کننده کپی داده و رخدادها برای همه فرزندان است.این پارامتر به صورت پیش فرض مساوی با پارامتر اول است.

مثال :

```
$("b").clone().prependTo("p");
```

مثال :

```
$('#copy').append($('#orig .elem')
          .clone()
          .children('a')
```

```
            .prepend('foo - ')
            .parent()
            .clone());

$('#copy-correct')
            .append($('#orig .elem')
            .clone()
            .children('a')
            .prepend('bar - ')
            .end());
```

- ***.detach()***

مجموعه عناصررا از DOM حذف می کند.

```
.detach( [selector] )
```

مثال:

- ```
  $("p").click(function(){
          $(this).toggleClass("off");
      });
      var p;
      $("button").click(function(){
        if ( p ) {
          p.appendTo("body");
          p = null;
        } else {
          p = $("p").detach();
        }
      });
  ```

- ***.empty()***

همه فرزندان مجموعه عناصر را در DOM حذف می کند

```
.empty()
```

مثال :

```
$("p").empty();
```
```

- ***.remove()***

مجموعه عناصر را از DOM حذف میکند.تفاوت این تابع با empty در این است که این تابع خود عنصر را نیز پاک میکند اما empty فرزندان را حذف می کند.

```
.remove( [selector] )
```

مثال :

```
$("p").remove(":contains('Hello')");
```

مثال :

```
$("p").remove();
```

- ***.replaceWith()***

هر عنصر مجموعه مطابق را با عنصر ارسال شده جایگزین می کند.

```
.replaceWith( newContent )
```

```
.replaceWith( function )
```

مثال :

```
$("button").click(function () {
  $(this).replaceWith( "<div>" + $(this).text() +
"</div>" );
});
```

مثال :

```
$('button').bind("click", function() {
  var $container =
$("div.container").replaceWith(function() {
    return $(this).contents();
  });

  $("p").append( $container.attr("class") );
});
```

مثال :

```
$("p").replaceWith( "<b>Paragraph. </b>" );
```

مثال :

```
$("p").click(function () {
  $(this).replaceWith( $("div") );
});
```

- ***.replaceAll()***

هر عنصر مقصد را با مجموعه مطابق جایگزین می کند.

```
.replaceAll( target )
```

مثال :

```
$("<b>Paragraph. </b>").replaceAll("p");
```

- ***.wrap()***

یک ساختار HTML را به عناصر مجموعه مطابق WRAP میکند.(عنصر HTML عنصر مجموعه مطابق را در میان می گیرد)

```
.wrap( wrappingElement )
```

```
.wrap( function(index) )
```

مثال :

```
$("p").wrap("<div></div>");
```

مثال :

```
$("span").wrap("<div><div><p><em><b></b></em></p></di
v></div>");
```

مثال :

```
$("p").wrap($(".doublediv"));
```

- ***.unwrap()***

والد عناصر مجموعه مطابق را حذف و آنها را در هر همان مکان نگه می دارد.

```
.unwrap()
```

مثال:

```
$("button").toggle(function(){
  $("p").wrap("<div></div>");
}, function(){
  $("p").unwrap();
});
```

- ***.wrapAll()***

یک ساختار HTML را گرداگرد همه عناصر موجود در مجموعه مطابق حلقه می کند.

```
.wrapAll( wrappingElement )
```

مثال :

```
$("p").wrapAll("<div></div>");
```

مثال:

```
$("p").wrapAll($(".doublediv"));
```

مثال :

```
$("p").wrapAll(document.createElement("div"));
```

- ***.wrapInner()***

یک ساختار HTML را گرداگرد محتوی همه عناصر موجود در مجموعه مطابق حلقه می کند.

```
.wrapInner( wrappingElement )
.wrapInner( function(index) )
```

مثال :

```
$("body").wrapInner("<div><div><p><em><b></b></em></p
></div></div>");
```

مثال:

```
$("p").wrapInner($("<span class='red'></span>"));
```

مثال :

```
$("p").wrapInner(document.createElement("b"));
```

بخش ششم:
مدیریت رخدادها

رخدادها بخش اساسی تعامل با کاربر در برنامه های کامپیوتری است.در یک برنامه زمانی که کارب عمل خاصی مثلا کلیک را انجام میدهد منجر به رخ دادن رویدادی میشود.پس از این رویداد برنامه بایستی بتواند به خوبی رویداد اتفاق افتاده را ضبط و عملیات متناظر و مطابق با آن را انجام دهد . در گذشته جاوا اسکریپت برای مدیریت رخداد ها مشکلات فراوانی داشت که اغلب آنها از تنوع مرورگرها و استانداردهایش ناشی می شد.کتابخانه jQuery با ارائه توابع و ابزارهای قدرتمندش مدیریت رخدادها را بسیار آسان نموده به طوری که کاملا برنامه نویس را از جزئیات مرورگر فارغ کرده و تنها از او میخواهد منطق برنامه را مشخص کند.در این بخش رخدادهای قابل برنامه ریزی در jQuery مورد بررسی قرار می دهیم ...

رویدادهای بارگذاری صفحه

- ***.load()***

برای رویداد load جاوا اسکریپت یک دستگیره تعیین میکند.(منظور از دستگیره تابعی است که در زمان روی دادن رخداد اجرا می شود) .این رویداد زمانی اتفاق می افتد که همه عناصر صفحه کامل لود شده باشد.شکل کلی این دستور به صورت زیر است :

```
.load( handler(eventObject) )

.load( [eventData], handler(eventObject) )
```

پارامتر eventdata داده های اختیاری است که به تابع ارسال می شود . handler نام تابع و پارامتر ارسالی اطلاعاتی در مورد رخدادی است که اتفاق افتاده است.این شی دارای اطلاعاتی است که در انتهای این فصل کاملا توضیح داده شده است.

این تابع به نوعی میانبر برای دستور زیر است :

```
.bind('load', handler).
```

مثال : اجرای تابعی در زمان لود صفحه (همه عناصر حتی تصاویر)

```
$(window).load(function () {
  // run code
});
```

مثال :

```
$('img.userIcon').load(function(){
  if($(this).height() > 100) {
    $(this).addClass('bigImg');
  }
});
```

- ***.ready()***

این رویداد در زمان لود DOM و قبل از تصاویر رخ می دهد. معمولا نقطه شروع کار با jQuery پس از این رخداد است.

```
.ready( handler )
```

مثال :

```
$(document).ready(function () {
  $("p").text("The DOM is now loaded and can be
manipulated.");
});
```

- ***.unload()***

برای رویداد unload **جاوا اسکریپت یک دستگیره تعیین میکند.(منظور از دستگیره تابعی است که در زمان روی دادن رخداد اجرا می شود) .این رویداد زمانی که کاربر از صفحه خارج می شود یا صفحه دیگری می رود اتفاق می افتد.شکل کلی این دستور به صورت زیر است :**

```
.unload( handler(eventObject) )

.unload( [eventData], handler(eventObject) )
```

پارامتر eventdata **داده های اختیاری است که به تابع ارسال می شود .** handler **نام تابع و پارامتر ارسالی اطلاعاتی در مورد رخدادی است که اتفاق افتاده است.این شی دارای اطلاعاتی است که در انتهای این فصل کاملا توضیح داده شده است.**

این تابع به نوعی میانبر برای دستور زیر است :

```
.bind('unload', handler).
```

مثال :

```
$(window).unload(function() {

  alert('Handler for .unload() called.');

});
```

مثال:

```
$(window).unload( function () { alert("Bye now!"); }
);
```

▪ رویدادهای فرم

- ***.blur()***

این رویداد زمانی اتفاق می افتد که عنصر فوکاس را از دست دهد .شکل کلی این دستور به صورت زیر است :

```
.blur( handler(eventObject) )
.blur( [eventData], handler(eventObject) )
.blur()
```

مثال :

```
$('#target').blur(function() {
  alert('Handler for .blur() called.');
});
```

مثال :

```
$('#other').click(function() {
  $('#target').blur();
});
```

- ***.change()***

این رویداد زمانی اتفاق می افتد که عنصر مقدارش تغییر کند . این رویداد برای <input><textarea><select> اتفاق می افتد.ذکر این نکته ضروری است که برای textbox این رویداد پس از از دست دادن فوکاس روی میدهد در حالی که برای چک باکس و دکمه رادیویی بلافاصله پس از انتخاب روی می دهد. شکل کلی این دستور به صورت زیر است :

```
.change( handler(eventObject) )
.change( [eventData], handler(eventObject) )
.chane()
```

مثال : اعتبار سنجی

```
$("input[type='text']").change( function() {
  // check input ($(this).val()) for validity here
});
```

مثال :

```
$("select").change(function () {
          var str = "";
          $("select option:selected").each(function
() {
                str += $(this).text() + " ";
              });
          $("div").text(str);
        })
        .change();
```

- ***.focus ()***

این رویداد زمانی اتفاق می افتد که عنصر فوکاس را بدست بگیرد.

```
.focus( handler(eventObject) )

.focus( [eventData], handler(eventObject) )

.focus()
```

مثال :

```
  $("input").focus(function () {
        $(this).next("span").css('display','inline').fad
eOut(1000);
    });
```

مثال : جلوگیری از گرفتن فوکاس در یک عنصر

```
$("input[type=text]").focus(function(){
  $(this).blur();
});
```

- ***.select ()***

این رویداد زمانی اتفاق می افتد که در textbox **و** textarea **متنی انتخاب شود.**

```
.select( handler(eventObject) )

.select( [eventData], handler(eventObject) )
```

```
.select()
```

مثال :

```
$(":input").select( function () {
      $("div").text("Something was
selected").show().fadeOut(1000);
    });
```

- ***.submit ()***

این رویداد زمانی اتفاق می افتد که کاربر سعی در تحویل فرم از طریق دکمه یا عناصری که میتوانند فرم راتحویل دهند روی می دهد.

```
.submit( handler(eventObject) )

.submit( [eventData], handler(eventObject) )

.submit()
```

مثال :

```
<form action="javascript:alert('success!');">
    <div>
      <input type="text" />

      <input type="submit" />
    </div>
  </form>


    $("form").submit(function() {
      if ($("input:first").val() == "correct") {
        $("span").text("Validated...").show();
        return true;
      }
      $("span").text("Not
valid!").show().fadeOut(1000);
      return false;
    });
```

- **رویدادهای ماوس :**

  - ***.click ()***

این رویداد زمانی اتفاق می افتد که کاربر ماوس را بر روی عنصری قرار داده و کلید ماوس را فشارداده و رها میکند.

```
.click( handler(eventObject) )

.click( [eventData], handler(eventObject) )

.click()
```

مثال :

```
$("p").click(function () {
    $(this).slideUp();
});
$("p").hover(function () {
  $(this).addClass("hilite");
}, function () {
  $(this).removeClass("hilite");
});
```

  - ***.click ()***

این رویداد زمانی اتفاق می افتد که کاربر ماوس را بر روی عنصری قرار داده و کلید ماوس را فشارداده و رها میکند.

```
.click( handler(eventObject) )

.click( [eventData], handler(eventObject) )

.click()
```

مثال :

```
$("p").click(function () {
    $(this).slideUp();
});
$("p").hover(function () {
  $(this).addClass("hilite");
}, function () {
```

```
    $(this).removeClass("hilite");
  });
```

- ***.dblclick ()***

این رویداد زمانی اتفاق می افتد که کاربر ماوس را بر روی عنصری قرار داده و بر روی آن دوبار کلیک کند.

```
.dblclick( handler(eventObject) )

.dblclick( [eventData], handler(eventObject) )

.dblclick()
```

مثال :

```
 var divdbl = $("div:first");
   divdbl.dblclick(function () {
     divdbl.toggleClass('dbl');
   });
```

- ***.focusin ()***
- ***.focusout()***

این رویدادها زمانی اتفاق می افتند که عنصر خود یا فرزندانش فوکاس را دریافت یا از دست دهند.

```
.focusin( handler(eventObject) )

.focusin( [eventData], handler(eventObject) )

.focusout( handler(eventObject) )

.focusout( [eventData], handler(eventObject) )
```

مثال :

```
$("p").focusin(function() {
        $(this).find("span").css('display','inline').fad
eOut(1000);
    });
```

مثال :

```
var fo = 0, b = 0;
$("p").focusout(function() {
  fo++;
  $("#fo")
    .text("focusout fired: " + fo + "x");
}).blur(function() {
  b++;
  $("#b")
    .text("blur fired: " + b + "x");

});
```

- ***.hover ()***

این رویداد زمانی اتفاق می افتد که کاربر را بر روی عنصر قرار می دهد. از طریق این رویداد میتوان هم برای وارد شد به شی و هم خارج شدن از آن تابع تعیین کرد.

```
.hover( handlerIn(eventObject),
handlerOut(eventObject) )

.hover( handlerInOut(eventObject) )
```

در حالت اول برای دو وضعیت تابع تعریف و در حالت دوم از یک تابع برای هر دو وضعیت استفاده می شود.

مثال :

```
 $("td").hover(
  function () {
    $(this).addClass("hover");
  },
  function () {
    $(this).removeClass("hover");
  }
```

برای غیر فعال کردن رخداد بالا میتوان از دستور زیر استفاده کرد.

```
$("td").unbind('mouseenter mouseleave');
```

مثال :

```
$("li")
.filter(":odd")
.hide()
 .end()
.filter(":even")
.hover(
  function () {
    $(this).toggleClass("active")
      .next().stop(true, true).slideToggle();
  }
);
```

- ***.mouseenter()***
- ***.mouseleave()***

این رویدادها زمانی اتفاق می افتند که ماوس وارد محدوده یک عنصر یا از آن خارج شود.

```
. mouseenter ( handler(eventObject) )
. mouseenter ( [eventData], handler(eventObject) )
. mouseenter ()
. mouseleave ( handler(eventObject) )
. mouseleave ( [eventData], handler(eventObject) )
. mouseleave ()
```

مثال :

```
var i = 0;
    $("div.overout").mouseover(function(){
      $("p:first",this).text("mouse over");
      $("p:last",this).text(++i);
    }).mouseout(function(){
      $("p:first",this).text("mouse out");
    });

    var n = 0;
    $("div.enterleave").mouseenter(function(){
```

```
    $("p:first",this).text("mouse enter");
    $("p:last",this).text(++n);
  }).mouseleave(function(){
    $("p:first",this).text("mouse leave");
  });
```

مثال :

```
var i = 0;
  $("div.overout").mouseover(function(){
    $("p:first",this).text("mouse over");
  }).mouseout(function(){
    $("p:first",this).text("mouse out");
    $("p:last",this).text(++i);
  });

  var n = 0;
  $("div.enterleave").mouseenter(function(){
    $("p:first",this).text("mouse enter");
  }).mouseleave(function(){
    $("p:first",this).text("mouse leave");
    $("p:last",this).text(++n);
  });
```

- ***.mousedown()***
- ***.mouseup()***

این رویدادها زمانی اتفاق می افتند که ماوس وارد محدوده یک عنصر شده و به ترتیب به محض فشرده شدن و رها شدن کلید ماوس اجرا می شوند.

```
. mousedown ( handler(eventObject) )
. mousedown ( [eventData], handler(eventObject) )
. mousedown ()
. mousedown ( handler(eventObject) )
. mousedown ( [eventData], handler(eventObject) )
. mousedown ()
```

مثال :

```
  $("p").mouseup(function(){
      $(this).append('<span style="color:#F00;">Mouse
up.</span>');
    }).mousedown(function(){
      $(this).append('<span style="color:#00F;">Mouse
down.</span>');
    });
```

- ***.mousemove()***

این رویداد زمانی اتفاق می افتد که ماوس در محدوده عنصر حرکت کند.

```
. mousemove ( handler(eventObject) )

. mousemove ( [eventData], handler(eventObject) )

. mousemove ()
```

مثال :

```
$("div").mousemove(function(e){
      var pageCoords = "( " + e.pageX + ", " +
e.pageY + " )";
      var clientCoords = "( " + e.clientX + ", " +
e.clientY + " )";
      $("span:first").text("( e.pageX, e.pageY ) - "
+ pageCoords);
      $("span:last").text("( e.clientX, e.clientY ) -
" + clientCoords);
    });
```

- ***.mouseout()***

این رویداد زمانی اتفاق می افتد که ماوس ازمحدوده عنصر خارج شود.

```
. mouseout( handler(eventObject) )

. mouseout ( [eventData], handler(eventObject) )

. mouseout ()
```

مثال :

```
var i = 0;
  $("div.overout").mouseout(function(){
    $("p:first",this).text("mouse out");
    $("p:last",this).text(++i);
  }).mouseover(function(){
    $("p:first",this).text("mouse over");
  });

  var n = 0;
  $("div.enterleave").bind("mouseenter",function(){
    $("p:first",this).text("mouse enter");
  }).bind("mouseleave",function(){
    $("p:first",this).text("mouse leave");
    $("p:last",this).text(++n);
  });
```

- ***.mouseover()***

این رویداد زمانی اتفاق می افتد که ماوس در محدوده عنصر قرار گیرد.

```
. mouseover ( handler(eventObject) )

. mouseover ( [eventData], handler(eventObject) )

. mouseover ()
```

مثال :

```
var i = 0;
  $("div.overout").mouseover(function() {
    i += 1;
    $(this).find("span").text( "mouse over x " + i );
  }).mouseout(function(){
    $(this).find("span").text("mouse out ");
  });

  var n = 0;
  $("div.enterleave").mouseenter(function() {
    n += 1;
    $(this).find("span").text( "mouse enter x " + n
);
```

```
  }).mouseleave(function() {
    $(this).find("span").text("mouse leave");
  });
```

- ***.toggle()***

این تابع دو یا چند دستگیره را برای اجرا در هنگام کلیک تعیین میکند.

```
.toggle( handler(eventObject), handler(eventObject),
[handler(eventObject)] )
```

اگر دو پارامتر برای تابع ارسال شود در کلیک اول پارامتر اول و در کلیک دوم پارامتر دوم اجرا می شود. و اگر سه پارامتر ارسال شود به صورت دوره ای اجرا می شوند.

مثال :

```
$("li").toggle(
      function () {
        $(this).css({"list-style-type":"disc",
"color":"blue"});
      },
      function () {
        $(this).css({"list-style-type":"disc",
"color":"red"});
      },
      function () {
        $(this).css({"list-style-type":"",
"color":""});
      }
    );
```

مثال :

```
$("td").toggle(
  function () {
    $(this).addClass("selected");
  },
  function () {
    $(this).removeClass("selected");
  }
);
```

رویدادهای صفحه کلید

- ***.keydown()***
- ***.keyup()***

این رویدادها به ترتیب زمانی اتفاق می افتند که کلیدی از صفحه کلید فشرده (کلید پایین باشد) یا رها شود(کلید آزاد شده باشد).

```
. keydown ( handler(eventObject) )
. keydown ( [eventData], handler(eventObject) )
. keydown ()
. keyup ( handler(eventObject) )
. keyup ( [eventData], handler(eventObject) )
. keyup ()
```

مثال :

```
var xTriggered = 0;
$('#target').keydown(function(event) {
  if (event.keyCode == '13') {
     event.preventDefault();
   }
   xTriggered++;
   var msg = 'Handler for .keydown() called ' +
xTriggered + ' time(s).';
  $.print(msg, 'html');
  $.print(event);
});
$('#other').click(function() {
  $('#target').keydown();
});
```

مثال:

```
var xTriggered = 0;
$('#target').keyup(function(event) {
  if (event.keyCode == '13') {
     event.preventDefault();
   }
```

```
    xTriggered++;
    var msg = 'Handler for .keyup() called ' +
xTriggered + ' time(s).';
  $.print(msg, 'html');
  $.print(event);
});

$('#other').click(function() {
  $('#target').keyup();
});
```

- ***.keypress ()***

این رویداد زمانی اتفاق می افتد که وروردی صفحه کلید در مرورگر ثبت شود.تقریبا شبیه keydown **است با این تفاوت که اگر کلیدی را فشرده و نگه دارد** keydown **یکبار و** keypress **به تعداد کاراکتر های وارد شده روی می دهد.**

```
. keypress ( handler(eventObject) )

. keypress ( [eventData], handler(eventObject) )

. keypress ()
```

مثال :

```
$("#target").keypress(function(event) {
  if ( event.which == 13 ) {
     event.preventDefault();
   }
   xTriggered++;
   var msg = "Handler for .keypress() called " +
xTriggered + " time(s).";
  $.print( msg, "html" );
  $.print( event );
});

$("#other").click(function() {
  $("#target").keypress();
});
```

رویدادهای مرورگر

- ***.error ()***

این رویداد اصولاً برای عناصری که به درستی لود نشده باشد روی می دهد .به عنوان مثال تصویری که آدرسش صحیح نباشد.

```
. error ( handler(eventObject) )
. error ( [eventData], handler(eventObject) )
```

مثال :

```
$("img")
  .error(function(){
    $(this).hide();
  })
  .attr("src", "missing.png");
```

- ***.resize ()***

این رویداد زمانی اتفاق می افتد که اندازه مرورگر تعییر کند .

```
. resize ( handler(eventObject) )
. resize ( [eventData], handler(eventObject) )
. resize ()
```

مثال :

```
$(window).resize(function() {
  $('body').prepend('<div>' + $(window).width() +
'</div>');
});
```

- ***.scroll ()***

این رویداد زمانی اتفاق می افتد که دریک عنصر اسکرول به مکانی دیگر برود(تغییر کند)

```
. scroll ( handler(eventObject) )
```

```
. scroll ( [eventData], handler(eventObject) )
. scroll ()
```

مثال :

```
  $("p").clone().appendTo(document.body);
    $("p").clone().appendTo(document.body);
    $("p").clone().appendTo(document.body);
    $(window).scroll(function () {
      $("span").css("display",
"inline").fadeOut("slow");
    });
```

- ضمیمه کردن توابع به متد

این سری از توابع رویداد خاصی را با تابعی مرتبط می کنند

- ***.bind ()***

این تابع دستگیره رخدادی را به یک رویداد خاص مقید میکند.شکل کلی این دستور به فرم زیر است :

```
.bind( eventType, [eventData], handler(eventObject) )
```

پارامتر اول رشته ای حاوی نام رویداد است.اگر این نام استاندارد نباشد یک رخداد سفارشی خواهد بود اگرچه این رویداد هرگز توسط مرورگر رخ نخواهد داد اما میتوان آن را با توابعی همچون trigger اجرا کرد.

```
.bind( eventType, [eventData], preventBubble )
```

پارامتر سوم در صورت false بودن از رویدادن رخداد جلوگیری میکند و آن را در bubbling رخدادها متوقف می کند.

```
.bind( events )
```

در این حالت events یک شی جاوا اسکریپت برای نگاشت یک یا چند رخداد است.

مثال :

```
$('#foo').bind('click', function() {
```

```
  alert('User clicked on "foo."');
});
```

مثال :

```
$('#foo').bind('mouseenter mouseleave', function() {
  $(this).toggleClass('entered');
});
```

مثال :

```
$('#foo').bind({
  click: function() {
    // do something on click
  },
  mouseenter: function() {
    // do something on mouseenter
  }
});
```

مثال: یک رویداد سفارشی

```
$("p").bind("myCustomEvent", function(e, myName,
myValue){
$(this).text(myName + ", hi there!");
$("span").stop().css("opacity", 1)
.text("myName = " + myName)
.fadeIn(30).fadeOut(1000);
});
$("button").click(function () {
$("p").trigger("myCustomEvent", [ "John" ]);
});
```

- ***.unbind ()***

رخدادهای مقید شده قبلی را پاک میکند.شکل کلی آن به صورت های زیر است.

```
.unbind( [eventType], [handler(eventObject)] )
```

```
.unbind( eventType, false )
```

این حالت معادل استفاده از `.bind( eventType, false )` **است.**

```
.unbind( event )
```

در این حالت events **یک شی جاوا اسکریپت برای نگاشت یک یا چند رخداد است.**

```
.unbind()
```

همه رخدادها را پاک میکند.

مثال :

```
$('#foo').unbind('click');
```

مثال :

```
function aClick() {
$("div").show().fadeOut("slow");
}
$("#bind").click(function () {
// could use .bind('click', aClick) instead but for
variety...
$("#theone").click(aClick)
  .text("Can Click!");
});
$("#unbind").click(function () {
$("#theone").unbind('click', aClick)
  .text("Does nothing...");
});
```

- ***.delegate ()***

دستگیره ای را برای یک یاچند رویداد که ممکن است الان وجود داشته باشد یا در آینده تولید شود تعیین میکند.منظور از آینده عناصری است که در زمان اجرا ایجاد می شود.شکل کلی به صورت های زیر است.

```
.delegate( selector, eventType, handler )
```

```
.delegate( selector, eventType, eventData, handler )
```

```
.delegate( selector, events )
```

مثال :

```
$("body").delegate("p", "click", function(){
      $(this).after("<p>Another paragraph!</p>");
    });
```

مثال:

```
    $("body").delegate("p", "myCustomEvent",
function(e, myName, myValue){
      $(this).text("Hi there!");
      $("span").stop().css("opacity", 1)
               .text("myName = " + myName)
               .fadeIn(30).fadeOut(1000);
    });
    $("button").click(function () {
      $("p").trigger("myCustomEvent");
    });
```

- ***.undelegate()***

دستگیره های مقید شده توسط تابع delegate **را پاک می کند.**

```
.undelegate()

.undelegate( selector, eventType )

.undelegate( selector, eventType, handler )

.undelegate( selector, events )

.undelegate( namespace )
```

مثال :

```
function aClick() {
  $("div").show().fadeOut("slow");
}
$("#bind").click(function () {
  $("body").delegate("#theone", "click", aClick)
    .find("#theone").text("Can Click!");
});
```

```
$("#unbind").click(function () {
  $("body").undelegate("#theone", "click", aClick)
    .find("#theone").text("Does nothing...");
});
```

در نسخه های قبلی از تابع live() برای مقید کردن و از تابع die() برای پاک کردن رخدادها استفاده می شد.

- ***.one ()***

دستگیره ای را به رخداد مورد نظر مقید میکند .آن دستگیره حداقل یکبار به ازای هر عنصر اجرا می شود.به عبارتی این تابع همان bind() است با این تفاوت که پس از اجرا unbind می شود.

```
.one( eventType, [eventData], handler(eventObject) )
```

مثال :

```
var n = 0;
$("div").one("click", function() {
  var index = $("div").index(this);
  $(this).css({
    borderStyle:"inset",
    cursor:"auto"
  });
  $("p").text("Div at index #" + index + " clicked."
+
      "  That's " + ++n + " total clicks.");
});
```

- ***.trigger ()***

همه دستگیره ها و رفتارهای مقید شده به یک عنصر را برای نوع مشخص شده اجرا می کند.به عبارتی میتوان رخداد خاصی را اجرا کرد

```
.trigger( eventType, extraParameters )
```

```
.trigger( event )
```

مثال :

```
$("button:first").click(function () {
update($("span:first"));
});
$("button:last").click(function () {
$("button:first").trigger('click');

update($("span:last"));
});

function update(j) {
var n = parseInt(j.text(), 10);
j.text(n + 1);
}
```

- ***.triggerHandler ()***

همه دستگیره ها و رفتارهای مقید شده به یک عنصر را اجرا میکند.رفتار آن شبیه به تابعtrigger است با این تفاوت که :

- این تابع منجر به انجام رفتار پیش فرض عنصر نمی شود .(مثلا تحویل فرم)
- trigger بر روی همه عناصر اجرا ولی این تابع تنها بر روی اولین عنصر اجرا می شود.
- رخداد رخ داده شده در سلسله مراتب DOM قرار نمی گیرد.(bubble up نمی شود)
- برخلاف trigger که شی jQuery برمیگرداند این تابع مقدار برگشتی آخرین دستگیره را برمیگرداند.(توانایی اجرای زنجیره ای دستورات را ندارد)

```
.triggerHandler( eventType, extraParameters )
```

مثال :

```
$("#old").click(function(){
$("input").trigger("focus");
});
$("#new").click(function(){
$("input").triggerHandler("focus");
});
$("input").focus(function(){
```

```
$("<span>Focused!</span>").appendTo("body").fadeOut(1
000);
});
```

▪ شی Event و خصوصیات آن

در اغلب متدهای بالای دیده شد که شی به نام eventObject به تابع دستگیره رخداد ارسال می شود.این شی شامل اطلاعات کاملی در مورد رخداد است که در این قسمت آن را به تفصیل شرح خواهیم داد.

برای ایجاد یک نمونه از این شی میتوان به صورت زیر عمل کرد: (عملگر new اختیاری است)

```
//Create a new jQuery.Event object without the "new"
operator.
  var e = jQuery.Event("click");
  // trigger an artificial click event
  jQuery("body").trigger( e );
```

مثال :

```
  // Create a new jQuery.Event object with specified
event properties.
  var e = jQuery.Event("keydown", { keyCode: 64 });
  // trigger an artificial keydown event with keyCode
64
  jQuery("body").trigger( e );
```

این شی دارای خصوصیاتی است که ممکن است بر اساس نوع رخداد میتواند متفاوت باشد.برخی از آنها به شرح زیر است .

```
altKey, attrChange, attrName, bubbles, button,
cancelable, charCode, clientX, clientY, ctrlKey,
currentTarget, data, detail, eventPhase, fromElement,
handler, keyCode, layerX, layerY, metaKey, newValue,
offsetX, offsetY, originalTarget, pageX, pageY,
```

```
prevValue, relatedNode, relatedTarget, screenX,
screenY, shiftKey, srcElement, target, toElement,
view, wheelDelta, which
```

با این وجود jQuery چند خصوصیات زیر را برای پشتیبانی از مرورگرهای متفاوت نرمالیزه کرده است :

- ***target***
- ***relatedTarget***
- ***pageX***
- ***pageY***
- ***which***
- ***metaKey***

در ادامه مهم ترین خصوصیات این شی را بررسی می کنیم

- ***event.currentTarget***

عنصر جاری DOM که فاز Bubbling رخداد قراردارد را مشخص میکند.

مثال:

```
$("p").click(function(event) {
  alert( event.currentTarget === this ); // true
});
```

- ***event.data***

در زمان مقید کردن رخداد و تعیین دستگیره میتوان پارامتری اضافی به تابع ارسال کرد که این خصوصیت شامل آن داده است .

مثال :

```
$("a").each(function(i) {
  $(this).bind('click', {index:i}, function(e){
     alert('my index is ' + e.data.index);
  });
});
```

- ***event.namespace***

فضای نام مشخص شده در هنگام تریگر کردن رویداد را برمیگرداند.این خاصیت بیشتر برای نوشتن پلاگین و ... کاربردی تر است.

مثال:

```
$("p").bind("test.something", function(event) {
  alert( event.namespace );
});
$("button").click(function(event) {
  $("p").trigger("test.something");
});
```

- ***event.pageX***
- ***event.pageY***

مختصات فعلی ماوس در هنگام روی دادن رویدادن رخداد نشان میدهد.(نسبت به document)

مثال :

```
$(document).bind('mousemove',function(e){
            $("#log").text("e.pageX: " + e.pageX + ",
e.pageY: " + e.pageY);
});
```

- ***event.preventDefault()***
- ***event.isDefaultPrevented()***

اگر تابع اول فراخوانی شود رفتار پیش فرض رویداد لغو خواهد شد و مقدار تابع دوم true می شود.به عبارتی تابع دوم مشخص میکند که آیا تابع preventDefault فراخوانی شده است یا خیر .

مثال:

```
$("a").click(function(event) {
  event.preventDefault();
  $('<div/>')
    .append('default ' + event.type + ' prevented')
    .appendTo('#log');
});
```

مثال:

```
$("a").click(function(event){
  alert( event.isDefaultPrevented() ); // false
  event.preventDefault();
  alert( event.isDefaultPrevented() ); // true
});
```

- ***event. event.stopPropagation()***
- ***event. isPropagationStopped ()***

اگر تابع اول فراخوانی شود از Bubbleup **شدن رویداد خود رویداد و همه والدهایش جلوگیری کرده و مقدار تابع دوم** true **می شود.به عبارتی تابع دوم مشخص میکند که آیا تابع** stopPropagation **فراخوانی شده است یا خیر .**

مثال:

```
$("p").click(function(event){
  event.stopPropagation();
  // do something
});
```

مثال:

```
function propStopped(e) {
  var msg = "";
  if ( e.isPropagationStopped() ) {
    msg =  "called"
  } else {
    msg = "not called";
  }
  $("#stop-log").append( "<div>" + msg + "</div>" );
```

```
}

$("button").click(function(event) {
  propStopped(event);
  event.stopPropagation();
  propStopped(event);
});
```

- ***event. stopImmediatePropagation ()***
- ***event. isImmediatePropagationStopped ()***

اگر تابع اول فراخوانی شود از Bubbleup **شدن خود رویداد جلوگیری کرده (والدها را به کار خود ادامه می دهند) و مقدار تابع دوم** true **می شود.به عبارتی تابع دوم مشخص میکند که آیا تابع** stopImmediatePropagation **فراخوانی شده است یا خیر .**

مثال:

```
$("p").click(function(event){
  event.stopImmediatePropagation();
});
$("p").click(function(event){
  // This function won't be executed
  $(this).css("background-color", "#f00");
});
$("div").click(function(event) {
  // This function will be executed
    $(this).css("background-color", "#f00");
});
```

مثال:

```
function immediatePropStopped(e) {
  var msg = "";
  if ( e.isImmediatePropagationStopped() ) {
    msg =  "called"
  } else {
    msg = "not called";
  }
  $("#stop-log").append( "<div>" + msg + "</div>" );
}
```

```
$("button").click(function(event) {
  immediatePropStopped(event);
  event.stopImmediatePropagation();
  immediatePropStopped(event);
});
```

- ***event.relatedTarget***

هر عنصر دیگری که شامل رخداد شده باشد

```
$("a").mouseout(function(event) {
  alert(event.relatedTarget.nodeName); // "DIV"
});
```

- ***event.result***

آخرین مقدار برگشتی دستگیره رخداد که توسط این رخداد trigger **شده باشد را برمیگرداند**

```
$("button").click(function(event) {
  return "hey";
});
$("button").click(function(event) {
  $("p").html( event.result );
});
```

- ***event.target***

عنصری که رویداد را پایه ریزی کرده است برمیگرداند.

```
function handler(event) {
  var $target = $(event.target);
  if( $target.is("li") ) {
    $target.children().toggle();
  }
}
$("ul").click(handler).find("ul").hide();
```

- ***event.type***

نوع رویداد را مشخص می کند.

```
$("a").click(function(event) {
  alert(event.type); // "click"
});
```

- ***event.which***

برای کلید یا دکمه این خصوصیت شامل مشخصات دکمه یا کلید فشرده شده است.

```
$('#whichkey').bind('keydown',function(e){
  $('#log').html(e.type + ': ' +  e.which );
});
```

- ***event.timeStamp***

شامل تفاوت بین زمان رخ دادن رویداد با اول ژانویه ۱۹۷۰ است (به میلی ثانیه)

```
$('div').click(function(event) {
  if ( last ) {
    diff = event.timeStamp - last
    $('div').append('time since last event: ' + diff
+ '<br/>');
  } else {
    $('div').append('Click again.<br/>');
  }
  last = event.timeStamp;
});
```

بخش هفتم:
افکت ها و جلوه های بصری

- **مقدمه**

کتابخانه jQuery شامل تکنیک های متفاوتی برای ایجاد انیمیشن در صفحه وب است.این افکت ، افکت های استانداردی هستند که در اکثر صفحات وب دیده می شوند.در این بخش توابع ایجاد انیمیشن در jQuery را به تفصیل بررسی می کنیم...

- **افکت های پایه ای :**

- ***.hide()***

عناصر موجود در مجموعه مطابق را مخفی می کند.

```
.hide()
.hide( duration, [callback] )
```

duration **زمان اجرای انیمیشن و** callback **تابعی است که به محض خاتمه اجرای انیمیشن اجرا می شود.**

```
.hide( [duration], [easing], [callback] )
```

easing **استفاده شده برای** transition **تعیین می کند.**

مثال :

```
for (var i = 0; i < 5; i++) {
      $("<div>").appendTo(document.body);
    }
    $("div").click(function () {
      $(this).hide(2000, function () {
        $(this).remove();
      });
    });
```

مثال :

```
  $("button").click(function () {
      $("p").hide("slow");
    });
```

- ***.show()***

عناصر موجود در مجموعه مطابق را نمایان میکند.

```
.show()
.show( duration, [callback] )
```

duration زمان اجرای انیمیشن و callback تابعی است که به محض خاتمه اجرای انیمیشن اجرا می شود.

```
.show( [duration], [easing], [callback] )
```

easing استفاده شده برای transition تعیین می کند.

مثال :

```
$("button").click(function () {
    $("p").show("slow");
    });
```

مثال :

```
$("#showr").click(function () {
  $("div:eq(0)").show("fast", function () {
    /* use callee so don't have to name the function
*/
    $(this).next("div").show("fast",
arguments.callee);
  });
});
$("#hidr").click(function () {
  $("div").hide(2000);
});
```

- **.toggle()**

عناصر موجود در مجموعه مطابق را مخفی یا نمایان میکند.

```
.toggle(showOrHide)
```

متغیر بولی نشان دهنده نمایش یا عدم نمایش شی است.

```
.toggle( [duration], [callback] )
```

duration زمان اجرای انیمیشن و callback تابعی است که به محض خاتمه اجرای انیمیشن اجرا می شود.

```
.show( [duration], [easing], [callback] )
```

easing استفاده شده برای transition تعیین می کند.

مثال :

```
$("button").click(function () {
$("p").toggle("slow");
});
```

مثال :

```
var flip = 0;
$("button").click(function () {
$("p").toggle( flip++ % 2 == 0 );
});
```

انیمیشن های Fade

- ***.fadeIn()***
- ***.fadeOut()***

عناصر موجود در مجموعه به ترتیب نمایش و محو میکند.

```
.fadeIn( [duration], [callback] )
```

duration زمان اجرای انیمیشن و callback تابعی است که به محض خاتمه اجرای انیمیشن اجرا می شود.

```
.fadeIn( [duration], [easing], [callback] )
```

easing استفاده شده برای transition تعیین می کند.

```
.fadeOut( [duration], [callback] )
```

duration زمان اجرای انیمیشن و callback تابعی است که به محض خاتمه اجرای انیمیشن اجرا می شود.

```
.fadeOut( [duration], [easing], [callback] )
```

easing استفاده شده برای transition تعیین می کند.

مثال :

```
$(document.body).click(function () {
        $("div:hidden:first").fadeIn("slow");
      });
```

مثال :

```
$("span").click(function () {
  $(this).fadeOut(1000, function () {
  $("div").text("'" + $(this).text() + "' has
faded!");
  $(this).remove();
  });
  });
  $("span").hover(function () {
  $(this).addClass("hilite");
  }, function () {
  $(this).removeClass("hilite");
  });
```

- ***.fadeTo()***

Opacity شی را تعیین می کند.

```
.fadeTo( duration, opacity, [callback] )
```

opacity عددی بین صفر و یک که شفافیت عنصر را تعیین میکند.

```
.fadeTo( duration, opacity, [easing], [callback] )
```

مثال :

```
$("div").click(function () {
$(this).fadeTo("fast", Math.random());
});
```

مثال :

```
var getPos = function (n) {
return (Math.floor(n) * 90) + "px";
};
$("p").each(function (n) {
```

```
var r = Math.floor(Math.random() * 3);
var tmp = $(this).text();
$(this).text($("p:eq(" + r + ")").text());
$("p:eq(" + r + ")").text(tmp);
$(this).css("left", getPos(n));
});
$("div").each(function (n) {
      $(this).css("left", getPos(n));
    })
.css("cursor", "pointer")
.click(function () {
      $(this).fadeTo(250, 0.25, function () {
            $(this).css("cursor", "")
                  .prev().css({"font-weight":
"bolder",
                              "font-style":
"italic"});
          });
    });
```

- ***fadeToggle ()***

عناصر موجود در مجموعه مطابق را با انیمیشن کردن شفافیتش نمایان یا محو می کتد.

```
.fadeToggle( [duration], [easing], [callback] )
```

مثال :

```
$("button:first").click(function() {
  $("p:first").fadeToggle("slow", "linear");
});
$("button:last").click(function () {
  $("p:last").fadeToggle("fast", function () {
    $("#log").append("<div>finished</div>");
  });
});
```

انیمیشن های Sliding

.slideUp()

.slideDown()

عناصر موجود در مجموعه مطابق را با انیمیشن slide محو یا نمایان میکند.(همانند پرده به بالا یا پایین)

```
.slideDown( [duration], [callback] )

.slideDown( [duration], [easing], [callback] )

.slideUp( [duration], [callback] )

.slideUp( [duration], [easing], [callback] )
```

مثال :

```
$(document.body).click(function () {
    if ($("div:first").is(":hidden")) {
      $("div").show("slow");
    } else {
      $("div").slideUp();
    }
  });
```

مثال :

```
$("button").click(function () {
    $(this).parent().slideUp("slow", function () {
      $("#msg").text($("button", this).text() + " has
completed.");
    });
  });
```

مثال :

```
$("div").click(function () {
$(this).css({ borderStyle:"inset", cursor:"wait" });
$("input").slideDown(1000,function(){
$(this).css("border", "2px red inset")
.filter(".middle")
 .css("background", "yellow")
 .focus();
```

```
$("div").css("visibility", "hidden");
});
});
```

- ***slideToggle ()***

عناصر موجود در مجموعه مطابق را با sliding نمایان یا محو می کتد.

```
.slideToggle( [duration], [callback] )
```

```
.slideToggle( [duration], [easing], [callback] )
```

مثال :

```
$("#aa").click(function () {
    $("div:not(.still)").slideToggle("slow", function
() {
      var n = parseInt($("span").text(), 10);
      $("span").text(n + 1);
    });
  });
```

▪ انیمشن های سفارشی

- ***.animate()***

یک انیمیشن سفارشی را بر اساس یک مجموعه از خصوصیات CSS اجرا می کند.

```
.animate( properties, [duration], [easing],
[complete] )
```

properties **مجموعه ای از خصوصیات css است که انیمیشن بر اساس آن اجرا می شود.** complete **تابعی است که پس از اجرای انیمیشن اجرا می شود.**

```
.animate( properties, options )
```

option **گزینه هایی است که برای تابع ارسال می شودو دارای گزینه های زیر است :**

```
duration, easing, complete, step, queue, specialEasing
```

step در هرگام از اجرای انیمیشن اجرا می شود.queue مقدار بولی است که تعیین میکند انیمیشن در صف اجرا قرار گیرد یا بلافاصله اجرا شود.(اگر false باشد بلافاصله اجرا می شود).specialEasing یک یا چند خصوصیت CSS است که برای تابع easing استفاده می شود.

اغلب خصوصیاتCSS که عددی است قابل استفاده هستند.

مثال:

```
$('#clickme').click(function() {
  $('#book').animate({
    opacity: 0.25,
    left: '+=50',
    height: 'toggle'
  }, 5000, function() {
    // Animation complete.
  });
});
```

مثال :

```
$('li').animate({
  opacity: .5,
  height: '50%'
},
{
  step: function(now, fx) {
    var data = fx.elem.id + ' ' + fx.prop + ': ' +
now;
    $('body').append('<div>' + data + '</div>');
  }
```

```
});
```

مثال :

```
$("#right").click(function(){
  $(".block").animate({"left": "+=50px"}, "slow");
});

$("#left").click(function(){
  $(".block").animate({"left": "-=50px"}, "slow");
});
```

مثال :

```
$( "#go1" ).click(function(){
  $( "#block1" ).animate( { width: "90%" }, { queue:
false, duration: 3000 })
     .animate({ fontSize: "24px" }, 1500 )
     .animate({ borderRightWidth: "15px" }, 1500 );
});

$( "#go2" ).click(function(){
  $( "#block2" ).animate({ width: "90%" }, 1000 )
     .animate({ fontSize: "24px" }, 1000 )
     .animate({ borderLeftWidth: "15px" }, 1000 );
});

$( "#go3" ).click(function(){
  $( "#go1" ).add( "#go2" ).click();
});

$( "#go4" ).click(function(){
  $( "div" ).css({ width: "", fontSize: "",
borderWidth: "" });
});
```

- ***.stop()***

انیمیشن جاری درحال اجرا را متوقف میکند.

```
.stop( [clearQueue], [jumpToEnd] )
```

پارامتر اول تعیین میکند که انیمیشن از صف خارج شود یا خیر(پیش فرض False است) و پارامتر دوم در صورت true انیمیشن را مجبور به اتمام میکند (پیش فرض false است)

مثال :

```
/* Start animation */
$("#go").click(function(){
$(".block").animate({left: '+=100px'}, 2000);
});

/* Stop animation when button is clicked */
$("#stop").click(function(){
$(".block").stop();
});
/* Start animation in the opposite direction */
$("#back").click(function(){
$(".block").animate({left: '-=100px'}, 2000);
});
```

- ***.delay()***

یک تایمر را برای تاخیر در اجرای دنباله ای از عناصر در صف ست می کند.

```
.delay( duration, [queueName] )
```

مثال :

```
  $("button").click(function() {
      $("div.first").slideUp(300).delay(800).fadeIn(4
00);
      $("div.second").slideUp(300).fadeIn(400);
    });
```

بخش هشتم:
کار با AJAX

- **مقدمه**

کتابخانه jQuery شامل یک سوئیت کامل برای به کارگیری قابلیت های AJAX است.توابع و متدهای موجود در این کتابخانه شما را قادر می سازد تا داده ها را بدون اینکه صفحه رفرش شود ، از سرور دریافت کنید.در ادامه هریک از این توابع را مورد بررسی قرار می دهیم

- **شی jqXHR**

این شی نوع برگشتی از *$.ajax()* **است.به عبارتی نتیجه درخواست** *ajax* **را می توان در این شی جستجوکرد.این شی دارای خصوصیاتی است که مهم ترین آنها به صورت زیر است.**

- *readyState*
- *status*
- *statusText*
- *responseXML* and/or *responseText*
- *setRequestHeader(name, value)*
- *getAllResponseHeaders()*
- *getResponseHeader()*
- *abort()*

پس از اجرای تابع ajax **توابعی موسوم به** callback **اجرا خواهند شد که به شرح زیر هستند:**

beforeSend: **شی** *jqXHR* **را دریافت و پارامترهای** *map* **را ست می کند.**

error: **در زمان روی دادن خطا به ترتیب رجیستر شدن اجرا می شود.شی** *jqXHR* **و رشته ای شامل نوع خطا و در صورت وجود یک استثنا را بر میگرداند.**

dataFilter : **به محض دریافت اعلام وصول پاسخ اجرا می شود.داده برگشتی و نوع آن را دریافت و به** *success* **تحویل می دهد.**

Success : **اگر درخواست موفقیت آمیز باشد اجرا می شود.این تابع داده برگشتی کد و شی** *jqXHR* **را دریافت میکند.**

complete:**پس از پایان یافتن درخواست با هر وضعیتی اجرا می شود.**

- **واسط سطح پایین** AJAX

این متدها برای ایجاد درخواست های AJAX **به کار گرفته می شود.**

- *jQuery.ajaxSetup()*

مقادیر پیش فرض را برایی درخواست بعدی AJAX تعیین میکند.

```
jQuery.ajaxSetup( options )
```

پارامتر ارسال شده به تابع در متد(ajax. کامل توضیح داده خواهد شد.

مثال :

```
$.ajaxSetup({
   url: "/xmlhttp/",
   global: false,
   type: "POST"

 });
 $.ajax({ data: myData });
```

- ***jQuery.ajax ()***

یک درخواست آسنکرون HTTP یا به عبارتی AJAX را صادر می کند.

```
jQuery.ajax( url, [settings] )
```

url مسیری است که درخواست به آنجا ارسال می شود.setting مجموعه از جفت های کلید/مقدار است که همگی اختیاری است .در ادامه همه این مقادیر تشریح شده اند.

```
jQuery.ajax( settings )
```

setting مجموعه ای از جفت های کلید / مقدار به صورت زیر است:

accepts: نوع محتوی ارسال شده در هدر درخواست است و به سرور می گوید که چه نوع برگشتی را می پذیرد.

async:به صورت پیش فرض همه درخواست ها به صورت آسنکرون ارسال می شود.در صورت false بودن این مقدار درخواست به صورت سنکرون ارسال شده و مرورگر قفل شده و نمیتوان کار دیگری انجام داد.(پیش فرض true است).

beforeSend(jqXHR, settings): یک تابع callback پیش از درخواست است که برای تغییر در jqXHR به کار گرفته می شود.مقدار برگشتی false از این تابع درخواست را لغو می کند.

cache: اگر این مقدار false باشد از کش شدن صفحه درخواستی در مرورگر جلوگیری می کند.در صورت false بودن عبارت "_=[TIMESTAMP]" به url اضافه می شود.(پیش فرض برای نوع های script و false jsonp و برای نوع های دیگر true است.)

complete(jqXHR, textStatus):یک تابع callback است که پس از پایان درخواست اجرا می شود (پس از اجرای success و error) .پارامتر اول شی jqXHR و پارامتر دوم رشته ای شامل یکی از موارد زیر است:

("*success*", "*notmodified*", "*error*", "*timeout*", "*abort*", or "*parsererror*").

contents:نگاشتی از جفت های رشته / عبارت باقاعده است که تعیین کننده چگونگی تجزیه پاسخ از طرف سرور توسط jQuery است.

contentType:در زمان ارسال درخواست نوع محتوی را تعیین میکند که در اغلب موارد و پیش فرض

'application/x-www-form-urlencoded' است.

context:شی است که بستر همه callback های مربوط به ajax را تعیین میکند.

```
$.ajax({
  url: "test.html",
  context: document.body,
  success: function(){
    $(this).addClass("done");
  }
});
```

converters:یک نگاشت مبدل بین نوع داده/نوع داده است . تابعی است که خروجی را تبدیل می کند. پیش فرض به صورت زیر است:

```
{"* text": window.String, "text html": true, "text
json": jQuery.parseJSON, "text xml": jQuery.parseXML}
```

crossDomain:اگر قصد استفاده از درخواست های crossDomain را بر روی یک domain دارید (مثلا jsonp) این خصوصیت را true قرار دهید.

data: داده ارسال شده به سمت سرور است.این داده تبدیل به querystring می شود و اگر رشته نباشد به دستور GET اضافه می شود.

dataFilter(data, type):تابعی است که برای کار با داده های خام XMLHttpRequest به کار گرفته می شود.

dataType :نوع داده ایست که انتظار دارید از سمت سرور دریافت کنید.اگر تعیین نشود معمولا از MIME استفاده می شود.

error(jqXHR, textStatus, errorThrown): تابعی است که در زمان بروز خطا یا ناموفق بودن در خواست اجرا می شود.

global: تعیین کننده رخ دادن رویدادهای global Ajax event است.پیش فرض true است و در صورت false بودن از روی دادن این رخدادها جلوگیری می کند.

headers:نگاشتی از جفت های کلید/مقدار است که در کنار درخواست ارسال می شود.این مقدار قبل از beforesend تعیین می شود و میتوان در آن تابع آن را بازنویسی کرد.

ifModified:تعیین میکند که موفقیت آمیز بودن درخواست را منوط به تغییر پاسخ سرور نسبت به آخرین درخواست میکند.بدین ترتیب در صورت false بودن هدر را چک نمی کند.(پیش فرض false است).

isLocal:اجازه میدهد که محیط اجرایی به صورت محلی قابل شناسایی باشد.(قابلیت استفاده از Filesystem)

Jsonp: نام تابع callback را با یک درخواست jsonp بازنویسی می کند.

jsonpCallback: نام تابع را برای درخواست jsonp تعیین میکند.

mimeType: نوع mime شی XHR را تعیین می کند.

Password:کلمه عبوری است که در پاسخ به اعتبارسنجی HTTP ارسال می شود.

processData:به صورت پیش فرض داده های ارسال شده به یه صورت data است .به عبارتی چیزی بیشتر از رشته.اگر قصد ارسال DOMDocument یا non-processed data را false قرار دهید.

scriptCharset:برای درخواست های با نوع داده jsonp و script و نوع GET یک کاراکترست تعیین می کند.

statusCode:نگاشتی بین توابع و کدهای عددی برگشتی از سوی HTTP است. مثال زیر را در نظر بگیرید.

```
$.ajax({
  statusCode: {
    404: function() {
      alert('page not found');
    }
  }
});
```

success(data, textStatus, jqXHR):در صورت موفقیت آمیز بودن درخواست این تابع اجرا می شود.

timeout:یک زمان انقضا برای تابع تعیین میکند.

traditional: اگر نیاز به استفاده از حالت اولیه پارامتر را دارید این خاصیت را true تعیین کنید.

type: نوع درخواست را تعیین میکند (POST,GET). برخی از دستورهای دیگری همچون PUT و DELETE نیز قابل استفاده است اما باید توجه داشت که در همه مرورگرها قابل استفاده نیست.(پیش فرض GET)

url: آدرسی است که درخواست به آنجا ارسال می شود.

username: نام کاربری است که در پاسخ به اعتبارسنجی HTTP ارسال می شود.

xhr:تابعی است که برای ایجاد شی XMLHttpRequest است.اگر IE باشد ActiveXObject ساخته می شود در غیر اینصورت XMLHttpRequest ایجاد می شود.

xhrFields:نگاشتی است بین فیلد/مقدار برای تعیین خصوصیات شی XHR به کار میرود.به عنوان مثال کد زیر را در نظر بگیرید:

```
$.ajax({
   url: a_cross_domain_url,
   xhrFields: {
```

```
        withCredentials: true

    }

});
```

مثال :

```
$.ajax({
   type: "GET",
   url: "test.js",
   dataType: "script"
 });
```

مثال :

```
$.ajax({
   type: "POST",
   url: "some.php",
   data: "name=John&location=Boston",
   success: function(msg){
     alert( "Data Saved: " + msg );
   }
 });
```

مثال :

```
$.ajax({
  url: "test.html",
  cache: false,
  success: function(html){
    $("#results").append(html);
  }
});
```

مثال:

```
var bodyContent = $.ajax({
      url: "script.php",
      global: false,
      type: "POST",
      data: {id : this.getAttribute('id')},
      dataType: "html",
      async:false,
      success: function(msg){
```

```
            alert(msg);
        }
    }
).responseText;
```

- ***ajaxPrefilter()***

از یک شی سفارشی option برای تغییر option جاری برای هر درخواست قبل از ارسال و قبل از پردازش توسط ajax(). تعیین میکند.

```
jQuery.ajaxPrefilter( [dataTypes], handler(options,
originalOptions, jqXHR) )
```

مثال :

```
var currentRequests = {};

$.ajaxPrefilter(function( options, originalOptions, jqXHR
) {

  if ( options.abortOnRetry ) {

    if ( currentRequests[ options.url ] ) {

      currentRequests[ options.url ].abort();

    }

    currentRequests[ options.url ] = jqXHR;

  }

});
```

مثال :

```
$.ajaxPrefilter( function( options ) {

  if ( options.crossDomain ) {

    options.url = "http://mydomain.net/proxy/" +
encodeURIComponent( options.url );

    options.crossDomain = false;

  }
```

```
});
```

مثال :

```
$.ajaxPrefilter(function( options ) {
  if ( isActuallyScript( options.url ) ) {
    return "script";
  }
});
```

▪ دستگیره های رخدادهای سراسری AJAX

- **.ajaxStart()**

هرگاه درخواست AJAX در حال ارسال باشد .jQuery چک می کند تا ببیند آیا درخواست معلق دیگری وجود دارد یا نه.اگر وجود نداشته باشداین رویداد اجرا می شود.

```
.ajaxStart( handler() )
```

مثال:

```
$("#loading").ajaxStart(function(){
   $(this).show();
 });
```

- ***.ajaxSend()***

هرگاه درخواست AJAX در حال ارسال باشد .jQuery قبل از ارسال این رویداد را اجرا می کند.

```
.ajaxSend( handler(event, jqXHR, ajaxOptions) )
```

مثال:

```
$("#msg").ajaxSend(function(evt, request, settings){
        $(this).append("<li>Starting request at " +
```

```
settings.url + "</li>");
      });
```

مثال :

```
$('.log').ajaxSend(function(e, jqxhr, settings) {
  if (settings.url == 'ajax/test.html') {
    $(this).text('Triggered ajaxSend handler.');
  }
});
```

- ***.ajaxStop()***

هرگاه یک درخواست AJAX کامل شده باشد .jQuery چک می کند تا ببیند آیا درخواست معلق دیگری وجود دارد یا نه(کامل نشده) .اگر وجود نداشته باشداین رویداد اجرا می شود.

```
.ajaxStop( handler() )
```

مثال:

```
$('.log').ajaxStop(function() {
  $(this).text('Triggered ajaxStop handler.');
});
```

مثال :

```
$("#loading").ajaxStop(function(){
      $(this).hide();
      });
```

- ***.ajaxError ()***

هرگاه یک درخواست AJAX با خطا خاتمه یافته و کامل شود این رویداد رخ می دهد.

```
.ajaxError( handler(event, jqXHR, ajaxSettings,
thrownError) )
```

مثال:

```
$("#msg").ajaxError(function(event, request,
settings){
  $(this).append("<li>Error requesting page " +
settings.url + "</li>");
});
```

- ***.ajaxSuccess ()***

هرگاه یک درخواست AJAX با موفقیت خاتمه یافته و کامل شود این رویداد رخ می دهد.

```
.ajaxSuccess( handler(event, XMLHttpRequest, ajaxOptions)
)
```

مثال:

```
$("#msg").ajaxSuccess(function(evt, request,
settings){
      $(this).append("<li>Successful Request!</li>");
      });
```

- ***.ajaxComplete ()***

هرگاه یک درخواست AJAX کامل شود این رویداد روی می دهد.

```
.ajaxComplete( handler(event, XMLHttpRequest,
ajaxOptions) )
```

مثال:

```
$("#msg").ajaxComplete(function(event,request,
settings){
   $(this).append("<li>Request Complete.</li>");
 });
```

توابع کمکی

- ***jQuery.param()***

این تابع یک نمایش سریالیزه شده از از یک شی یا یک آرایه تولید می کند.این نمایش سریالیزه برای استفاده در یک درخواست AJAX یا querystring مناسب است.

```
jQuery.param( obj )
```

```
jQuery.param( obj, traditional )
```

traditional یک متغیر بولی که تعیین کننده استفاده از حالت سریالیزه کردن shallow است.به صورت پیش از فرض از حالت deep استفاده می شود.

مثال:

```
var params = { width:1680, height:1050 };
  var str = jQuery.param(params);
  $("#results").text(str);
```

خروجی :

width=1680&height=1050

مثال :

```
var myObject = {
  a: {
    one: 1,
    two: 2,
    three: 3
  },
  b: [1,2,3]
};
var shallowEncoded = $.param(myObject, true);
var shallowDecoded =
decodeURIComponent(shallowEncoded);
```

```
alert(shallowEncoded);

alert(shallowDecoded);
```

- ***.serialize()***

مجموعه از عناصر را انکد می کند.(رشته ای)

```
.serialize()
```

مثال:

```
 function showValues() {
      var str = $("form").serialize();
      $("#results").text(str);
    }
    $(":checkbox, :radio").click(showValues);
    $("select").change(showValues);
    showValues();
```

مثال :

```
$('form').submit(function() {

  alert($(this).serialize());

  return false;

});
```

خروجی :

```
a=1&b=2&c=3&d=4&e=5
```

- ***.serializeArray()***

مجموعه از عناصر را به صورت آرایه از نام ها و مقادیر انکد می کند.(رشته ای)

```
.serializeArray()
```

مثال:

```
function showValues() {
      var fields = $(":input").serializeArray();
      $("#results").empty();
      jQuery.each(fields, function(i, field){
        $("#results").append(field.value + " ");
      });
    }

    $(":checkbox, :radio").click(showValues);
    $("select").change(showValues);
    showValues();
```

مثال :

```
$('form').submit(function() {

  console.log($(this).serializeArray());

  return false;

});
```

خروجی :

```
[  {

    name: "a",

    value: "1"

  }, {

    name: "b",

    value: "2"

  },{

    name: "c",

    value: "3"

  },{

    name: "d",

    value: "4"

  },{
```

```
    name: "e",
    value: "5"
  }]
```

▪ متدهای سریع

- ***.load()***

داده ها را از سرور دریافت میکند و در عنصر HTML مطابق قرار میدهد.

```
.load( url, [data], [complete(responseText,
textStatus, XMLHttpRequest)] )
```

مثال:

```
$("#new-nav").load("/ #jq-footerNavigation li");
```

مثال :

```
$("#success").load("/not-here.php",
function(response, status, xhr) {
  if (status == "error") {
    var msg = "Sorry but there was an error: ";
    $("#error").html(msg + xhr.status + " " +
xhr.statusText);
  }
});
```

- ***jQuery.get()***
- ***jQuery.post()***

داده ها را از سرور از طریق POST یا GET دریافت می کند.

```
jQuery.post( url, [data], [success(data, textStatus,
jqXHR)], [dataType] )
```

```
jQuery.get( url, [data], [success(data, textStatus,
jqXHR)], [dataType] )
```

url آدرس موردنظر ،data نگاشت یا رشته ارسالی به سرور است.success تابعی است که در صورت موفقیت اجرا می شود و datatype نوع داده ای است که از انتظار می رود سرور آن را برگرداند.

مثال :

```
$.get("test.php",
   function(data){
     $('body').append("Name: " + data.name); // John
              .append("Time: " + data.time); //  2pm
   }, "json");
```

مثال:

```
$.get("test.cgi", { name: "John", time: "2pm" },
   function(data){
     alert("Data Loaded: " + data);
   });
```

مثال :

```
/* attach a submit handler to the form */
  $("#searchForm").submit(function(event) {

    /* stop form from submitting normally */
    event.preventDefault();

    /* get some values from elements on the page: */
    var $form = $( this ),
        term = $form.find( 'input[name="s"]' ).val(),
        url = $form.attr( 'action' );

    /* Send the data using post and put the results in a
div */
    $.post( url, { s: term },
      function( data ) {
          var content = $( data ).find( '#content' );
          $( "#result" ).empty().append( content );
      }
    );
  });
```

- ***jQuery.getScript()***

یک فایل اسکریپت جاوا اسکریپت را از طریق GET HTTP سرور دریافت می کند و آن را اجرا می کند.

```
jQuery.getScript( url, [success(data, textStatus)] )
```

مثال:

```
$.getScript("http://dev.jquery.com/view/trunk/plugins
/color/jquery.color.js", function() {
  $("#go").click(function(){
    $(".block").animate( { backgroundColor: "pink" },
1000)
      .delay(500)
      .animate( { backgroundColor: "blue" }, 1000);
  });
});
```

- ***jQuery.getJSON()***

یک شی JSON انکد شده را از سرور با استفاده از GET HTTP دریافت میکند.

```
jQuery.getJSON( url, [data], [success(data,
textStatus, jqXHR)] )
```

مثال:

```
$.getJSON("http://api.flickr.com/services/feeds/photo
s_public.gne?jsoncallback=?",
  {
    tags: "cat",
    tagmode: "any",
    format: "json"
  },
  function(data) {
    $.each(data.items, function(i,item){
      $("<img/>").attr("src",
item.media.m).appendTo("#images");
      if ( i == 3 ) return false;
    });
  });
```

بخش نهم:
توابع سودمند

- مقدمه

کتابخانه jQuery شامل یک سوئیت کامل برای به کارگیری قابلیت های AJAX است.توابع و متدهای موجود در این کتابخانه شما را قادر می سازد تا داده ها را بدون اینکه صفحه رفرش شود ، از سرور دریافت کنید.در ادامه هریک از این توابع را مورد بررسی قرار می دهیم ...

- شی jqXHR

این شی نوع برگشتی از `$.ajax()` است.به عبارتی نتیجه درخواست `ajax` را می توان در این شی جستجوکرد.این شی دارای خصوصیاتی است که مهم ترین آنها به صورت زیر است.

- `readyState`
- `status`
- `statusText`
- `responseXML` and/or `responseText`
- `setRequestHeader(name, value)`
- `getAllResponseHeaders()`
- `getResponseHeader()`
- `abort()`

پس از اجرای تابع ajax توابعی موسوم به callback اجرا خواهند شد که به شرح زیر هستند:

`beforeSend`: شی `jqXHR` را دریافت و پارامترهای `map` را ست می کند.

`error`: در زمان روی دادن خطا به ترتیب رجیستر شدن اجرا می شود.شی `jqXHR` و رشته ای شامل نوع خطا و در صورت وجود یک استثنا را بر میگرداند.

`dataFilter` : به محض دریافت اعلام وصول پاسخ اجرا می شود.داده برگشتی و نوع آن را دریافت و به `success` تحویل می دهد.

`Success` : اگر درخواست موفقیت آمیز باشد اجرا می شود.این تابع داده برگشتی کد و شی `jqXHR` را دریافت میکند.

`complete`:پس از پایان یافتن درخواست با هر وضعیتی اجرا می شود.

- **واسط سطح پایین AJAX**

این متدها برای ایجاد درخواست های AJAX به کار گرفته می شود.

- ***jQuery.ajaxSetup()***

مقادیر پیش فرض را برایی درخواست بعدی AJAX تعیین میکند.

```
jQuery.ajaxSetup( options )
```

پارامتر ارسال شده به تابع در متدajax(). کامل توضیح داده خواهد شد.

مثال :

```
$.ajaxSetup({
   url: "/xmlhttp/",
   global: false,
   type: "POST"

 });
 $.ajax({ data: myData });
```

- ***jQuery.ajax ()***

یک درخواست آسنکرون HTTP یا به عبارتی AJAX را صادر می کند.

```
jQuery.ajax( url, [settings] )
```

url مسیری است که درخواست به آنجا ارسال می شود.setting مجموعه از جفت های کلید/مقدار است که همگی اختیاری است .در ادامه همه این مقادیر تشریح شده اند.

```
jQuery.ajax( settings )
```

setting مجموعه ای از جفت های کلید / مقدار به صورت زیر است:

accepts: نوع محتوی ارسال شده در هدر درخواست است و به سرور می گوید که چه نوع برگشتی را می پذیرد.

async:به صورت پیش فرض همه درخواست ها به صورت آسنکرون ارسال می شود.در صورت false بودن این مقدار درخواست به صورت سنکرون ارسال شده و مرورگر قفل شده و نمیتوان کار دیگری انجام داد.(پیش فرض true است).

beforeSend(jqXHR, settings): یک تابع callback پیش از درخواست است که برای تغییر در jqXHR به کار گرفته می شود.مقدار برگشتی false از این تابع درخواست را لغو می کند.

cache: اگر این مقدار false باشد از کش شدن صفحه درخواستی در مرورگر جلوگیری می کند.در صورت false بودن عبارت "_=[TIMESTAMP]" به url اضافه می شود.(پیش فرض برای نوع های script و false jsonp و برای نوع های دیگر true است.)

complete(jqXHR, textStatus):یک تابع callback است که پس از پایان درخواست اجرا می شود (پس از اجرای success و error) .پارامتر اول شی jqXHR و پارامتر دوم رشته ای شامل یکی از موارد زیر است:

("success", "notmodified", "error", "timeout", "abort", or "parsererror").

contents:نگاشتی از جفت های رشته / عبارت باقاعده است که تعیین کننده چگونگی تجزیه پاسخ از طرف سرور توسط jQuery است.

contentType:در زمان ارسال درخواست نوع محتوی را تعیین میکند که در اغلب موارد و پیش فرض 'application/x-www-form-urlencoded' است.

context:شی است که بستر همه callback های مربوط به ajax را تعیین میکند.

```
$.ajax({
  url: "test.html",
  context: document.body,
  success: function(){
    $(this).addClass("done");
  }
});
```

converters:یک نگاشت مبدل بین نوع داده/نوع داده است . تابعی است که خروجی را تبدیل می کند. پیش فرض به صورت زیر است:

```
{"* text": window.String, "text html": true, "text
json": jQuery.parseJSON, "text xml": jQuery.parseXML}
```

crossDomain:اگر قصد استفاده از درخواست های crossDomain را بر روی یک domain دارید (مثلا jsonp) این خصوصیت را true قرار دهید.

data: داده ارسال شده به سمت سرور است.این داده تبدیل به querystring می شود و اگر رشته نباشد به دستور GET اضافه می شود.

dataFilter(data, type):تابعی است که برای کار با داده های خام XMLHttpRequest به کار گرفته می شود.

dataType :نوع داده ایست که انتظار دارید از سمت سرور دریافت کنید.اگر تعیین نشود معمولا از MIME استفاده می شود.

error(jqXHR, textStatus, errorThrown): تابعی است که در زمان بروز خطا یا ناموفق بودن در خواست اجرا می شود.

global: تعیین کننده رخ دادن رویدادهای global Ajax event است.پیش فرض true است و در صورت false بودن از روی دادن این رخدادها جلوگیری می کند.

headers:نگاشتی از جفت های کلید/مقدار است که در کنار درخواست ارسال می شود.این مقدار قبل از beforesend تعیین می شود و میتوان در آن تابع آن را بازنویسی کرد.

ifModified:تعیین میکند که موفقیت آمیز بودن درخواست را منوط به تغییر پاسخ سرور نسبت به آخرین درخواست میکند.بدین ترتیب در صورت false بودن هدر را چک نمی کند.(پیش فرض false است)

isLocal:اجازه میدهد که محیط اجرایی به صورت محلی قابل شناسایی باشد.(قابلیت استفاده از Filesystem)

Jsonp: نام تابع callback را با یک درخواست jsonp بازنویسی می کند.

jsonpCallback: نام تابع را برای درخواست jsonp تعیین میکند.

mimeType: نوع mime شی XHR را تعیین می کند.

Password:کلمه عبوری است که در پاسخ به اعتبارسنجی HTTP ارسال می شود.

processData:به صورت پیش فرض داده های ارسال شده به یه صورت data است .به عبارتی چیزی بیشتر از رشته.اگر قصد ارسال DOMDocument یا non-processed data را false قرار دهید.

scriptCharset:برای درخواست های با نوع داده jsonp و script و نوع GET یک کاراکترست تعیین می کند.

statusCode:نگاشتی بین توابع و کدهای عددی برگشتی از سوی HTTP است. مثال زیر را در نظر بگیرید.

```
$.ajax({
  statusCode: {
    404: function() {
      alert('page not found');
    }
  }
});
```

success(data, textStatus, jqXHR):در صورت موفقیت آمیز بودن درخواست این تابع اجرا می شود.

timeout:یک زمان انقضا برای تابع تعیین میکند.

traditional: اگر نیاز به استفاده از حالت اولیه پارامتر را دارید این خاصیت را true تعیین کنید.

type: نوع درخواست را تعیین میکند (POST,GET). برخی از دستورهای دیگری همچون PUT و DELETE نیز قابل استفاده است اما باید توجه داشت که در همه مرورگرها قابل استفاده نیست.(پیش فرض GET)

url: آدرسی است که درخواست به آنجا ارسال می شود

username: نام کاربری است که در پاسخ به اعتبارسنجی HTTP ارسال می شود.

xhr:تابعی است که برای ایجاد شی XMLHttpRequest است.اگر IE باشد ActiveXObject ساخته می شود در غیر اینصورت XMLHttpRequest ایجاد می شود.

xhrFields:نگاشتی است بین فیلد/مقدار برای تعیین خصوصیات شی XHR به کار میرود.به عنوان مثال کد زیر را در نظر بگیرید:

```
$.ajax({
   url: a_cross_domain_url,
   xhrFields: {
      withCredentials: true
   }
});
```

مثال :

```
$.ajax({
   type: "GET",
   url: "test.js",
   dataType: "script"
 });
```

مثال :

```
$.ajax({
   type: "POST",
   url: "some.php",
   data: "name=John&location=Boston",
   success: function(msg){
     alert( "Data Saved: " + msg );
   }
 });
```

مثال :

```
$.ajax({
  url: "test.html",
  cache: false,
  success: function(html){
    $("#results").append(html);
  }
});
```

مثال:

```
var bodyContent = $.ajax({
      url: "script.php",
      global: false,
      type: "POST",
      data: {id : this.getAttribute('id')},
      dataType: "html",
      async:false,
      success: function(msg){
         alert(msg);
      }
   }
).responseText;
```

- ***ajaxPrefilter()***

از یک شی سفارشی option **برای تغییر** option **جاری برای هر درخواست قبل از ارسال و قبل از پردازش توسط** ajax(). **تعیین میکند.**

```
jQuery.ajaxPrefilter( [dataTypes], handler(options,
originalOptions, jqXHR) )
```

مثال :

```
var currentRequests = {};

$.ajaxPrefilter(function( options, originalOptions, jqXHR
) {

  if ( options.abortOnRetry ) {

    if ( currentRequests[ options.url ] ) {

      currentRequests[ options.url ].abort();

    }

    currentRequests[ options.url ] = jqXHR;

  }

});
```

مثال :

```
$.ajaxPrefilter( function( options ) {
```

```
  if ( options.crossDomain ) {
    options.url = "http://mydomain.net/proxy/" +
encodeURIComponent( options.url );
    options.crossDomain = false;
  }
});
```

مثال :

```
$.ajaxPrefilter(function( options ) {
  if ( isActuallyScript( options.url ) ) {
    return "script";
  }
});
```

▪ دستگیره های رخدادهای سراسری AJAX

- ***.ajaxStart()***

هرگاه درخواست AJAX در حال ارسال باشد .jQuery چک می کند تا ببیند آیا درخواست معلق دیگری وجود دارد یا نه.اگر وجود نداشته باشداین رویداد اجرا می شود.

```
.ajaxStart( handler() )
```

مثال:

```
$("#loading").ajaxStart(function(){
   $(this).show();
 });
```

- ***.ajaxSend()***

هرگاه درخواست AJAX در حال ارسال باشد .jQuery قبل از ارسال این رویداد را اجرا می کند.

```
.ajaxSend( handler(event, jqXHR, ajaxOptions) )
```

مثال:

```
$("#msg").ajaxSend(function(evt, request, settings){
        $(this).append("<li>Starting request at " +
settings.url + "</li>");
      });
```

مثال :

```
$('.log').ajaxSend(function(e, jqxhr, settings) {
  if (settings.url == 'ajax/test.html') {
    $(this).text('Triggered ajaxSend handler.');
  }
});
```

- ***.ajaxStop()***

هرگاه یک درخواست AJAX کامل شده باشد .jQuery چک می کند تا ببیند آیا درخواست معلق دیگری وجود دارد یا نه(کامل نشده) .اگر وجود نداشته باشداین رویداد اجرا می شود.

```
.ajaxStop( handler() )
```

مثال:

```
$('.log').ajaxStop(function() {
  $(this).text('Triggered ajaxStop handler.');
});
```

مثال :

```
$("#loading").ajaxStop(function(){
      $(this).hide();
      });
```

- ***.ajaxError ()***

هرگاه یک درخواست AJAX با خطا خاتمه یافته و کامل شود این رویداد رخ می دهد.

```
.ajaxError( handler(event, jqXHR, ajaxSettings,
thrownError) )
```

مثال:

```
$("#msg").ajaxError(function(event, request,
settings){
  $(this).append("<li>Error requesting page " +
settings.url + "</li>");
});
```

- ***.ajaxSuccess ()***

هرگاه یک درخواست AJAX با موفقیت خاتمه یافته و کامل شود این رویداد رخ می دهد.

```
.ajaxSuccess( handler(event, XMLHttpRequest, ajaxOptions)
)
```

مثال:

```
$("#msg").ajaxSuccess(function(evt, request,
settings){
      $(this).append("<li>Successful Request!</li>");
      });
```

.ajaxComplete ()

هرگاه یک درخواست AJAX کامل شود این رویداد روی می دهد.

```
.ajaxComplete( handler(event, XMLHttpRequest,
ajaxOptions) )
```

مثال:

```
$("#msg").ajaxComplete(function(event,request,
settings){
```

```
  $(this).append("<li>Request Complete.</li>");
});
```

▪ توابع کمکی

- ***jQuery.param()***

این تابع یک نمایش سریالیزه شده از از یک شی یا یک آرایه تولید می کند.این نمایش سریالیزه برای استفاده در یک درخواست AJAX یا querystring مناسب است.

```
jQuery.param( obj )
```

```
jQuery.param( obj, traditional )
```

traditional **یک متغیر بولی که تعیین کننده استفاده از حالت سریالیزه کردن** shallow **است.به صورت پیش از فرض از حالت** deep **استفاده می شود.**

مثال:

```
var params = { width:1680, height:1050 };
  var str = jQuery.param(params);
  $("#results").text(str);
```

خروجی :

width=1680&height=1050

مثال :

```
var myObject = {
  a: {
    one: 1,
    two: 2,
    three: 3
  },
```

```
  b: [1,2,3]
};
var shallowEncoded = $.param(myObject, true);
var shallowDecoded =
decodeURIComponent(shallowEncoded);
alert(shallowEncoded);
alert(shallowDecoded);
```

- ***.serialize()***

مجموعه از عناصر را انکد می کند.(رشته ای)

```
.serialize()
```

مثال:

```
 function showValues() {
      var str = $("form").serialize();
      $("#results").text(str);
    }
    $(":checkbox, :radio").click(showValues);
    $("select").change(showValues);
    showValues();
```

مثال :

```
$('form').submit(function() {
  alert($(this).serialize());
  return false;
});
```

خروجی :

```
a=1&b=2&c=3&d=4&e=5
```

- ***.serializeArray()***

مجموعه از عناصر را به صورت آرایه از نام ها و مقادیر انکد می کند.(رشته ای)

```
.serializeArray()
```

مثال:

```
function showValues() {
      var fields = $(":input").serializeArray();
      $("#results").empty();
      jQuery.each(fields, function(i, field){
        $("#results").append(field.value + " ");
      });
    }

    $(":checkbox, :radio").click(showValues);
    $("select").change(showValues);
    showValues();
```

مثال :

```
$('form').submit(function() {

  console.log($(this).serializeArray());

  return false;

});
```

خروجی :

```
[  {

    name: "a",

    value: "1"

  }, {

    name: "b",

    value: "2"
```

```
},{
  name: "c",
  value: "3"
},{
  name: "d",
  value: "4"
},{
  name: "e",
  value: "5"
}]
```

▪ متدهای سریع

.load()

داده ها را از سرور دریافت میکند و در عنصر HTML مطابق قرار میدهد.

```
.load( url, [data], [complete(responseText,
textStatus, XMLHttpRequest)] )
```

مثال:

```
$("#new-nav").load("/ #jq-footerNavigation li");
```

مثال :

- ```
  $("#success").load("/not-here.php",
  function(response, status, xhr) {
    if (status == "error") {
      var msg = "Sorry but there was an error: ";
      $("#error").html(msg + xhr.status + " " +
  xhr.statusText);
    }
  });
  ```
  ```

- ***jQuery.get()***
- ***jQuery.post()***

داده ها را از سرور از طریق POST یا GET دریافت می کند.

```
jQuery.post( url, [data], [success(data, textStatus,
jqXHR)], [dataType] )

jQuery.get( url, [data], [success(data, textStatus,
jqXHR)], [dataType] )
```

url آدرس موردنظر ،data نگاشت یا رشته ارسالی به سرور است.success تابعی است که در صورت موفقیت اجرا می شود و datatype نوع داده ای است که از انتظار می رود سرور آن را برگرداند.

مثال :

```
$.get("test.php",
   function(data){
     $('body').append("Name: " + data.name); // John
              .append("Time: " + data.time); //  2pm
   }, "json");
```

مثال:

```
$.get("test.cgi", { name: "John", time: "2pm" },
   function(data){
     alert("Data Loaded: " + data);
   });
```

مثال :

```
/* attach a submit handler to the form */
  $("#searchForm").submit(function(event) {

    /* stop form from submitting normally */
    event.preventDefault();

    /* get some values from elements on the page: */
    var $form = $( this ),
        term = $form.find( 'input[name="s"]' ).val(),
        url = $form.attr( 'action' );

    /* Send the data using post and put the results in a
```

```
div */
    $.post( url, { s: term },
      function( data ) {
          var content = $( data ).find( '#content' );
          $( "#result" ).empty().append( content );
      }
    );
  });
```

- ***jQuery.getScript()***

یک فایل اسکریپت جاوا اسکریپت را از طریق GET HTTP سرور دریافت می کند و آن را اجرا می کند.

```
jQuery.getScript( url, [success(data, textStatus)] )
```

مثال:

```
$.getScript("http://dev.jquery.com/view/trunk/plugins
/color/jquery.color.js", function() {
  $("#go").click(function(){
    $(".block").animate( { backgroundColor: "pink" },
1000)
      .delay(500)
      .animate( { backgroundColor: "blue" }, 1000);
  });
});
```

- ***jQuery.getJSON()***

یک شی JSON انکد شده را از سرور با استفاده از GET HTTP دریافت میکند.

```
jQuery.getJSON( url, [data], [success(data,
textStatus, jqXHR)] )
```

مثال:

```
$.getJSON("http://api.flickr.com/services/feeds/photo
s_public.gne?jsoncallback=?",
  {
    tags: "cat",
    tagmode: "any",
```

```
      format: "json"
    },
    function(data) {
      $.each(data.items, function(i,item){
        $("<img/>").attr("src",
item.media.m).appendTo("#images");
        if ( i == 3 ) return false;
      });
    });
```

Powerd By
Drupaliha.com