

در ادامه مطلب قبلی آموزش [\(jQuery\) جی کوئری #1](#) به ادامه بحث می‌پردازیم.

توابع سودمند

با وجود آنکه انتخاب کردن و ایجاد مجموعه ای از عناصر صفحه یکی از معمول‌ترین و پرستفاده‌ترین کاربردهای تابع (\$) محسوب می‌شود، این تابع توانایی‌های دیگری نیز دارد. یکی از مفیدترین آنها استفاده شدن به عنوان فضای نام گروهی برای توابع سودمند می‌باشد. تعداد زیادی تابع سودمند با استفاده از \$ به عنوان فضای نام قابل دسترسی می‌باشند که اکثر نیازهای یک صفحه را پاسخگو می‌باشند در این پست برخی از آنها را معرفی می‌کنیم در پست‌های آینده سعی می‌کنیم توابع سودمند بیشتری را شرح دهیم.

فراخوانی و استفاده از این توابع در ابتدا ممکن است کمی عجیب به نظر برسد. به مثال زیر دقت کنید که تابع سودمند trim () را فراخوانی کرده ایم.

```
$.trim(someString);
```

در صورتی که نوشتن علامت \$ برای شما عجیب به نظر می‌رسد می‌توانید شناسه دیگر با نام jQuery به کار ببرید. کد زیر دقیقاً مانند بالا عمل می‌کند شاید درک آن راحت‌تر هم باشد.

```
jQuery.trim(someString);
```

بدیهی است که از jQuery یا \$ تنها به عنوان فضای نامی که تابع trim() در آن تعریف شده اند، استفاده شده باشد.

نکته : اگر چه در نوشته‌های آنلاین jQuery، این عناصر به عنوان توابع سودمند در معرفی شده اند اما در حقیقت آنها متدهایی برای تابع (\$) می‌باشند.

عملکرد صفحه آماده (The document ready handler)

هنگامی که از Unobtrusive JavaScript استفاده می‌کنیم، رفتار از ساختار جدا می‌شود، بنابراین برای انجام عملیات روی عناصر صفحه باید منتظر بمانیم تا آنها ایجاد شوند. برای رسیدن به این هدف، ما نیاز به راهی داریم که تا زمان ایجاد عناصر DOM روی صفحه منتظر بماند قبل از آن عملیات را اجرا کند.

به طور معمول از [onload](#) برای نمونه‌های [window](#) استفاده می‌شود، که پس از لود شدن کامل صفحه، دستورهای قابل اجرا می‌باشند. بنابراین ساختار کلی آن کدی مانند زیر خواهد بود:

```
window.onload = function() {  
    $("table tr:nth-child(even)").addClass("even");  
};
```

نوشتن کد به صورت بالا سبب می‌شود که کد پس از بارگذاری کامل صفحه اجرا شود. متأسفانه، مرورگرها تا بعد از ساخته شدن عناصر صفحه صبر نمی‌کنند، بلکه پس از ساخت درخت عناصر صفحه منتظر بارگذاری کامل منابع خارجی صفحه مانند تصاویر نیز می‌مانند و سپس آنها را در پنجره مرورگر نمایش می‌دهند. در نتیجه بازدید کننده زمان زیادی منتظر می‌ماند تا رویداد **onload** تکمیل شود.

حتی بدتر از آن، زمانی است که اگر به طور مثال یکی از تصاویر با مشکل مواجه شود که زمان قابل توجهی صرف بارگذاری آن

شود، کاربر باید تمام این مدت را صبر کند تا پس از آن بتواند با این صفحه کار کند. این نکته می‌تواند دلیلی برای استفاده نکردن از Unobtrusive JavaScript برای شروع کار باشد.

اما راه بهتری نیز وجود دارد، می‌توانیم تنها زمانی که قسمت ساختار عناصر صفحه ترجمه شده و HTML به درخت عناصر تبدیل می‌شود، صبر کنیم. پس از آن کد مربوط به رفتارها را اجرا کنیم. رسیدن به این روش برای استفاده از Cross-Browser کمی مشکل است، اما به لطف jQuery و قدرت آن، این امر به سادگی امکان پذیر است و دیگر نیازی به منتظر ماندن برای بارگذاری منابع صفحه مانند تصاویر و ویدیوها نمی‌باشد. Syntax زیر نمونه ای از چنین حالتی است:

```
$(document).ready(function() {  
    $("table tr:nth-child(even)").addClass("even");  
});
```

ابتدا صفحه مورد نظر را به تابع **\$()** ارسال کرده ایم، سپس هر زمان که آن صفحه آماده شد (Ready)، تابع ارسال شده به آن اجرا خواهد شد. البته می‌توان کد نوشته شده بالا را به شکل مختصرتری هم نوشت:

```
$(function() {  
    $("table tr:nth-child(even)").addClass("even");  
});
```

با ارسال تابع به **\$()**، ما مرورگر را مجبور می‌کنیم که برای اجرای کد تا زمانی که DOM کامل لود شود (فقط DOM لود شود) منتظر بماند. حتی بهتر از آن ما می‌توانیم از این تکنیک چندین بار در همان سند HTML استفاده کرده و مرورگر تمامی تابع‌های مشخص شده توسط ما را به ترتیب اجرا خواهد کرد. (یعنی من در دیک صفحه می‌توانم چنین بار تابع **ready()** را فراخوانی کنم). در مقابل روش **OnLoad** پنجره فقط اجازه اجرای یکبار تابع را به ما می‌دهد. این هم یکی دیگر از کارکردهای دیگر تابع **\$()** می‌باشد. حال به یکی دیگر از امکاناتی که این تابع برای ما فراهم می‌کند دقت کنید.

ساختن اجزای DOM (ساختن عناصر صفحه)

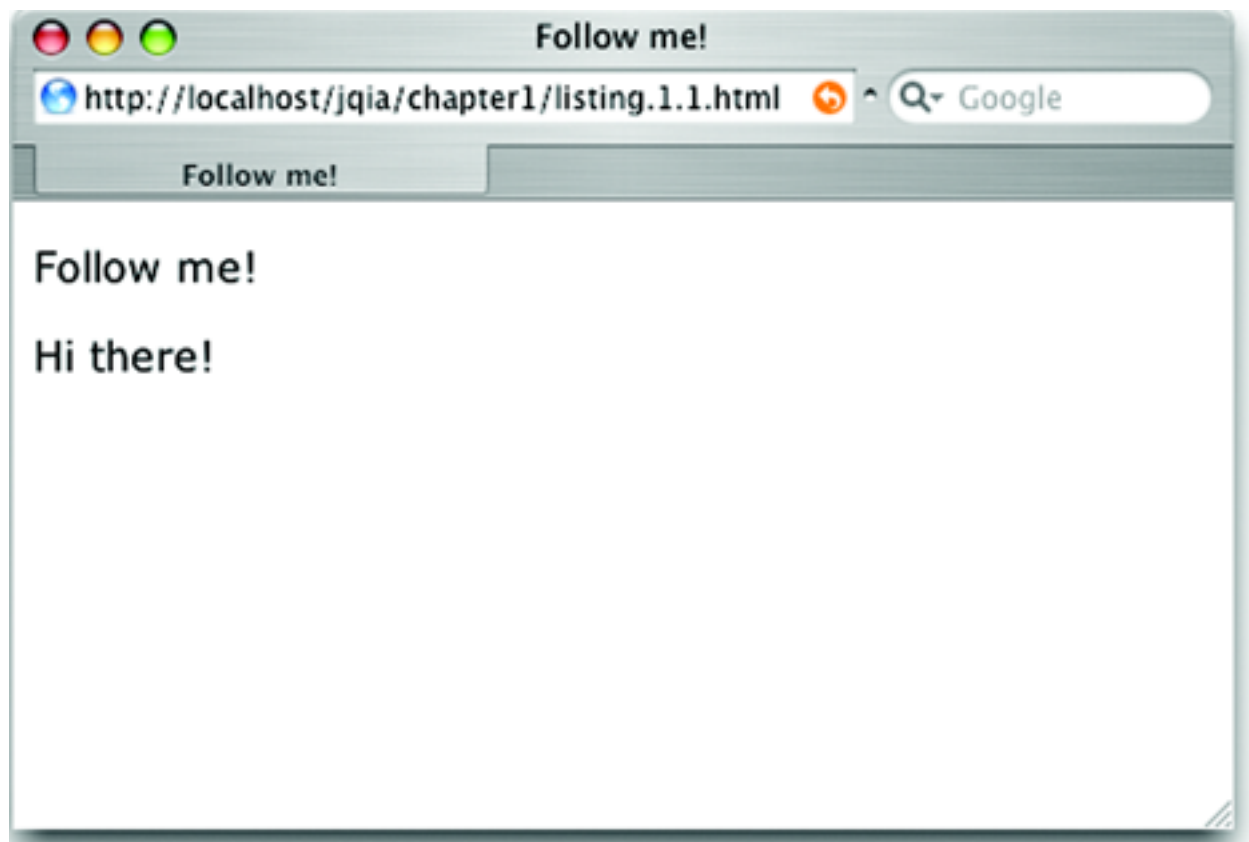
یکی دیگر از کارهایی که تابع **\$()** می‌تواند برای ما انجام دهد ایجاد کردن عناصر صفحه است. به این منظور ورودی تابع **\$()** را یک رشته که حاوی دستور HTML مربوط به ساخت یک عنصر می‌باشد، قرار می‌دهیم. برای مثال دستور زیر یک **تگ p** ایجاد می‌کند:

```
$("<p>Hi there!</p>")
```

اما ایجاد یک عنصر DOM یا (سلسله مراتب عناصر DOM) برای ما به تنهایی سودمند نیست، و هدف ما چیز دیگری است. ایجاد اشیا صفحه توسط **\$()** زمانی برای ما مفید خواهد بود که بخواهیم به هنگام ساخت، تابعی بروی آن اعمال کنیم یا به محض ساخت آن را به تابعی ارسال کنیم به کد زیر دقت کنید:

```
<html>  
  <head>  
    <title>Follow me!</title>  
    <script type="text/javascript" src="../scripts/jquery-1.2.js"></script>  
    <script type="text/javascript">  
      // بودن صفحه عنصر مورد نظر ایجاد می‌شود Reday در زمان  
      $(function(){  
        $("<p>Hi there!</p>").insertAfter("#followMe");  
      });  
    </script>  
  </head>  
  <body>  
    <p id="followMe">Follow me!</p>  
  </body>  
</html>
```

در کد بالا زمانی که صفحه مورد نظر Ready شد تابع مورد نظر ما اجرا شده و در عناصر صفحه بعد از عنصری که id آن followMe می‌باشد یک عنصر p را ایجاد می‌کند. که خروجی آن شبیه تصویر زیر خواهد بود.



مزیت دیگر jQuery این است که در صورتی که امکانی را ندارد شما به آسانی می‌توانید آن را توسعه داده و برای آن [پلاگین](#) طراحی کنید.

برای پایان دادن به این پست همانطور که دیدیم jQuery قادر به انجام کارهای زیر است:
انتخاب عناصر و ایجاد مجموعه ای از آنها که آماده اعمال متدهای مختلف می‌باشند.
استفاده به عنوان یک فضای نام برای توابع سودمند.
ایجاد اشیا مختلف HTML بروی صفحه.
اجرای کد به محض آماده شدن اشیا صفحه.

موفق و موید باشید

در ادامه مطلب قبلی آموزش [\(jQuery\) جی کوئری #2](#) به ادامه بحث می‌پردازیم.

انتخاب عناصر صفحه

در پستهای قبل ([^](#) و [^](#)) با بسیاری از توانایی‌ها و کارکردهای jQuery شامل توانایی‌های آن برای انتخاب عناصر موجود در صفحه تا تعریف توابع جدید و استفاده از آنها به محض آماده شدن صفحه آشنا شدیم.

در این پست و پست بعدی توضیحات تکمیلی در خصوص دو مورد از توانایی‌های jQuery و البته تابع (\$) خواهیم داشت که مورد اول، انتخاب عناصر صفحه با استفاده از انتخاب کننده‌ها و مورد دوم ایجاد عناصر جدید می‌باشد.

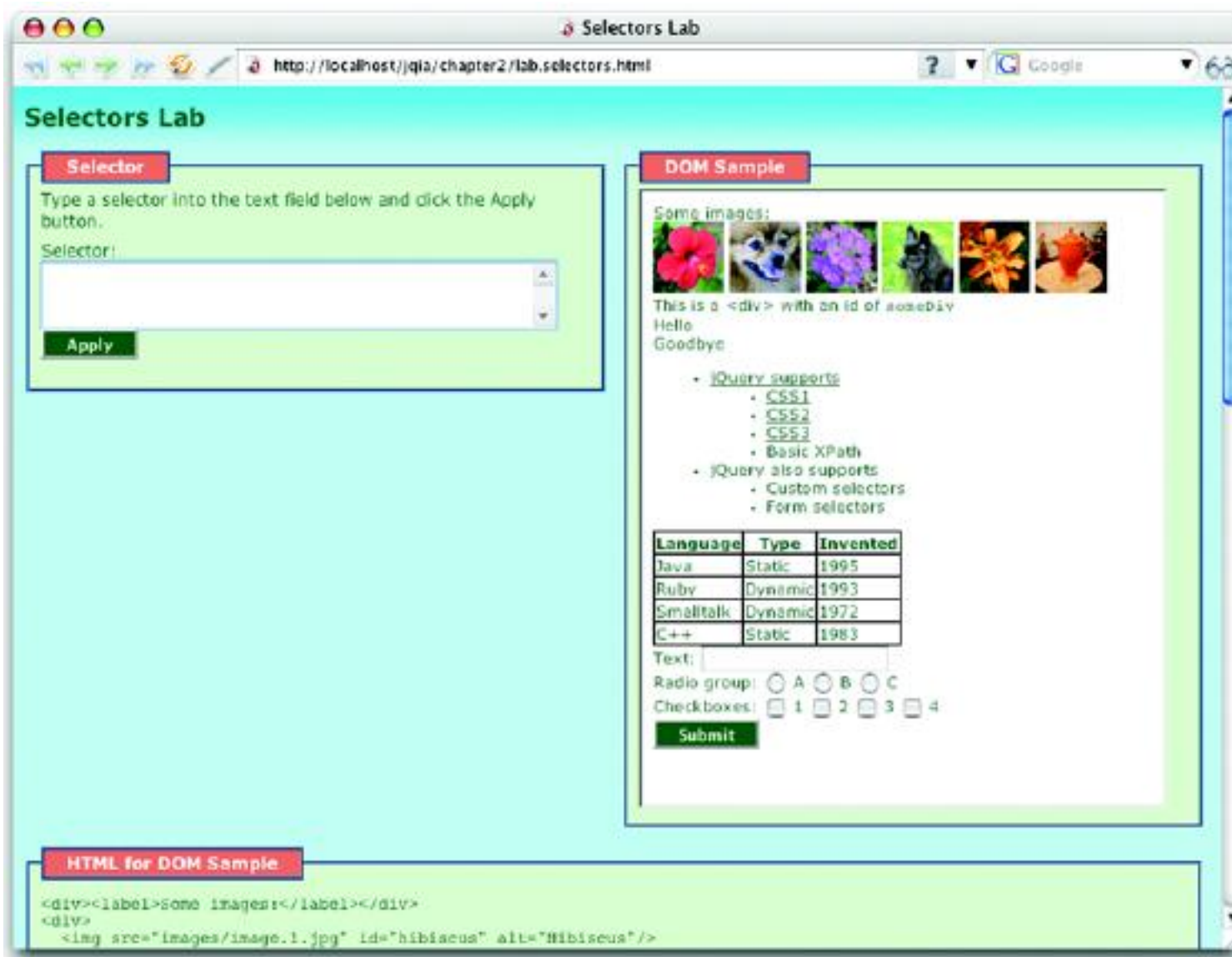
در بسیاری از مواقع برای تعامل با صفحه اینترنتی نیاز به تغییر دادن بخشی از یکی از اشیا موجود در صفحه داریم. اما پیش از آنکه قادر باشیم آنها را تغییر دهیم، ابتدا باید با استفاده از مکانیزمی شی مورد نظر را مشخص و سپس آن را انتخاب کنیم تا پس از آن قادر به اعمال تغییری در آن باشیم. بنابراین اجازه دهید تا به یک بررسی عمیق از راه‌های مختلف انتخاب عناصر صفحه و ایجاد تغییر در آنها بپردازیم.

1- انتخاب عناصر صفحه برای ایجاد تغییر

اولین قدم برای استفاده از هر گونه تابع jQuery، مشخص کردن و انتخاب عناصری است که می‌خواهیم تابع روی آن عناصر اعمال شود. گاهی اوقات انتخاب این مجموعه عناصر با یک توضیح ساده مشخص می‌شود، برای مثال "تمام عناصر پاراگراف موجود در صفحه". اما گاهی اوقات مشخص کردن این مجموعه نیاز به توضیح پیچیده‌تری دارد، برای مثال "تمام عناصر لیست در صفحه که دارای کلاس listElement هستند و لینکی دارند که اولین عضو آن لیست می‌باشد".

خوشبختانه jQuery یک مکانیزم بسیار قوی و قدرتمند ارائه کرده است که انتخاب هر عنصری از صفحه را به سادگی امکان پذیر می‌سازد. انتخاب کننده‌های jQuery از ساختار مربوط به CSS استفاده می‌کنند، بنابراین ممکن است شما هم اکنون با تعداد زیادی از آنها آشنا باشید. در ادامه شمار بیشتر و قدرتمندتری خواهید آموخت.

برای درک بهتر شما از مطالب مربوط به بخش انتخاب کننده‌ها، یک مثال آماده مختص به این مبحث، در قالب یک صفحه اینترنتی، را در [فایل صفحه کارگاهی](#) قرار داده ایم، این فایل در آدرس chapter2/1ab.selector.htm قابل دسترسی می‌باشد. این مثال از پیش آماده و کامل (نوشته شده توسط نویسنده کتاب)، این امکان را به شما می‌دهد تا با وارد کردن یک رشته، به عنوان پارامتر انتخاب کننده، در همان زمان عنصر انتخاب کننده در صفحه را رویت کنید. زمانی که این صفحه را اجرا می‌کنید تصویری مانند زیر ظاهر خواهد شد.



برای درک بهتر مطالب این سلسله پست‌ها می‌توانید فایل‌های کتاب را از [آدرس اصلی](#) آن یا از [این آدرس در همین سایت](#) دانلود نمایید.

این صفحه سه پنجره مجزا دارد. در پنجره سمت چپ، یک textBox و یک دکمه دیده می‌شود، که با وارد کردن یک انتخاب کننده در textBox و فشردن دکمه، عنصر مورد نظر در پنجره سمت راست انتخاب می‌شود. برای شروع در textBox عبارت li را بنویسید و دکمه Apply را کلیک کنید. با انجام این عمل تصویر زیر باید خروجی شما باشد. می‌توانید حالت‌های دیگر را خودتان امتحان کنید.

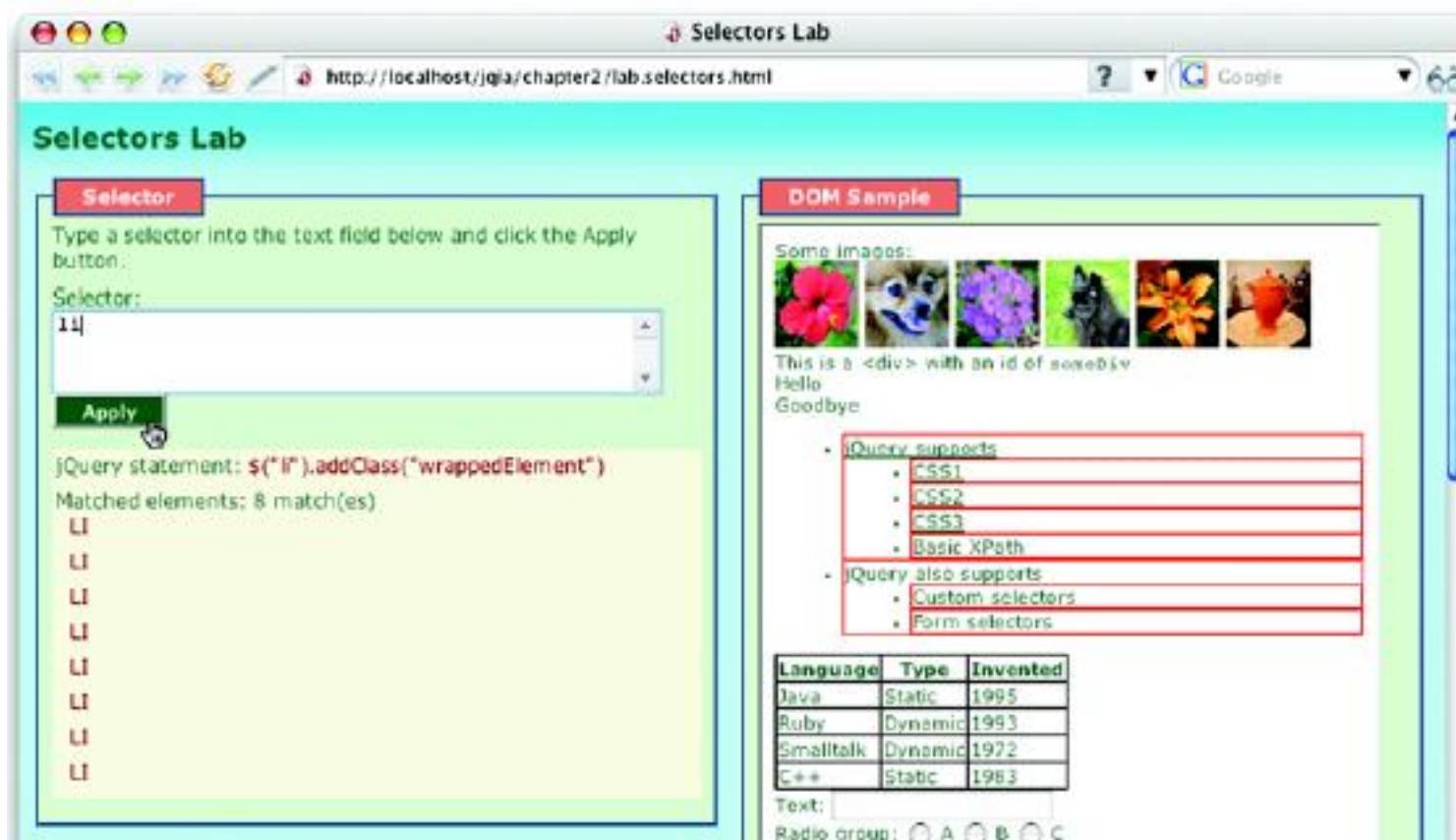


Figure 2.2 A selector value of `li` matches all `` elements when applied as shown by the display results.

1-1- استفاده از انتخاب کننده‌های ابتدایی CSS

برنامه نویسان وب برای اعمال فرمت‌های ظاهری گوناگون به بخش‌ها و عناصر مختلف یک صفحه اینترنتی، از یک راه بسیار ساده، در عین حال قدرتمند و کارا استفاده می‌کنند که در تمام مرورگرهای مختلف نیز جوابگو باشد. این انتخاب کننده‌ها عناصر را بر اساس نام شناسه آنها، نام کلاس و یا ساختار سلسله مراتبی موجود در صفحه انتخاب می‌کنند.

در زیر به معرفی چند نمونه از این انتخاب کننده‌های ساده CSS می‌پردازیم:

a : تمام عناصر `<a>` را انتخاب می‌کند.

#specialID : عنصری را که دارای ID با عنوان specialID باشد انتخاب می‌کند.

.specialClass : عنصری را که دارای کلاس specialClass هستند انتخاب می‌کند.

a#specialID.specialClass : این عبارت عنصری را انتخاب می‌کند که شناسه آن specialID باشد، به شرط آنکه این عنصر `<a>` باشد و دارای کلاس specialClass نیز باشد را انتخاب می‌کند.

p a.specialClass : تمام عناصر لینک (`<a>`) را که دارای کلاس specialClass باشند و درون یک عنصر پاراگراف (`<p>`) قرار گرفته باشند را انتخاب می‌کند.

این انتخاب کننده‌ها شاید ساده به نظر برسند، اما در بسیاری از مواقع پاسخگوی ما می‌باشند؛ به علاوه آنه که با ادغام این انتخاب کننده‌های ساده، ما می‌توانیم انتخاب کننده‌های پیچیده‌تر و تخصصی‌تر ایجاد کنیم.

نکته مثبت در مورد انتخاب کننده‌های CSS این است که از همین انتخاب کننده‌ها می‌توانیم در jQuery نیز استفاده کنیم. برای این کار تنها کافیست انتخاب کننده مورد نظر را به تابع `$()` ارسال کنیم. در زیر یک نمونه را مشاهده می‌کنید:

```
$("p a.specialClass")
```

به جز چند مورد خاص که استثنا وجود دارد، [CSS3](#) و jQuery کاملاً با هم سازگاری دارند. بنابراین انتخاب عناصر به این شکل طبیعی خواهد بود. به عبارتی دیگر هر عنصر که از این طریق توسط CSS انتخاب شود، همان انتخاب حاصل انتخاب کننده jQuery نیز خواهد بود. اما باید به این نکته توجه داشت که jQuery وابسته به CSS نیست و اگر مرورگری پیاده سازی استاندارد برای CSS نداشته باشد، انتخاب کننده jQuery به مشکل بر نمی خورد، بلکه jQuery انتخاب خود را به درستی انجام می دهد، چرا که jQuery از قوانین استاندارد [W3C](#) تبعیت می کند.

2-1- استفاده از انتخاب کننده های فرزند (Child) ، نگهدارنده (Container) و صفت (Attribute)

برای انتخاب کننده های پیشرفته تر، jQuery از جدیدترین مرورگرهایی که CSS را پشتیبانی می کنند، استفاده می کند که می توان به Mozilla Firefox, Internet Explorer 7, Safari و سایر مرورگرهای پیشرفته (مدرن) اشاره کرد. این انتخاب کننده های پیشرفته شما را قادر می سازند تا مستقیماً فرزند یک عنصر را انتخاب کنید و یا از ساختار سلسله مراتبی عناصر صفحه، مستقیماً به عنصر مورد نظر دسترسی داشته باشید و یا حتی تمام عناصری که یک صفت خاص را شامل می شوند، انتخاب کنید. گاهی اوقات انتخاب فرزندی از یک شی برای ما مطلوب است. برای مثال ممکن است ما به چند مورد از یک لیست احتیاج داشته باشیم، نه یک زیر مجموعه ای از آن لیست. به قطعه کد زیر که از صفحه کارگاهی این پست گرفته شده است دقت نمایید:

```
<ul>
  <li><a href="http://jquery.com">jQuery supports</a>
    <ul>
      <li><a href="css1">CSS1</a></li>
      <li><a href="css2">CSS2</a></li>
      <li><a href="css3">CSS3</a></li>
      <li>Basic XPath</li>
    </ul>
  </li>
  <li>jQuery also supports
    <ul>
      <li>Custom selectors</li>
      <li>Form selectors</li>
    </ul>
  </li>
</ul>
```

حال فرض کنید از این ساختار، لینک وب سایت jQuery مد نظر ماست و این کار بدون انتخاب سایر لینک های مربوط به CSS مطلوب است. اگر بخواهیم از دستورهای انتخاب کننده CSS استفاده کنیم، دستوری به شکل `ul.myList li a` خواهیم داشت. اما متأسفانه این دستور تمام لینک های این ساختار را انتخاب میکند، زیرا همه آنها لینک هایی در عنصر `li` می باشند. با نوشتن این دستور در صفحه کارگاهی خروجی به شکل زیر خواهد بود:

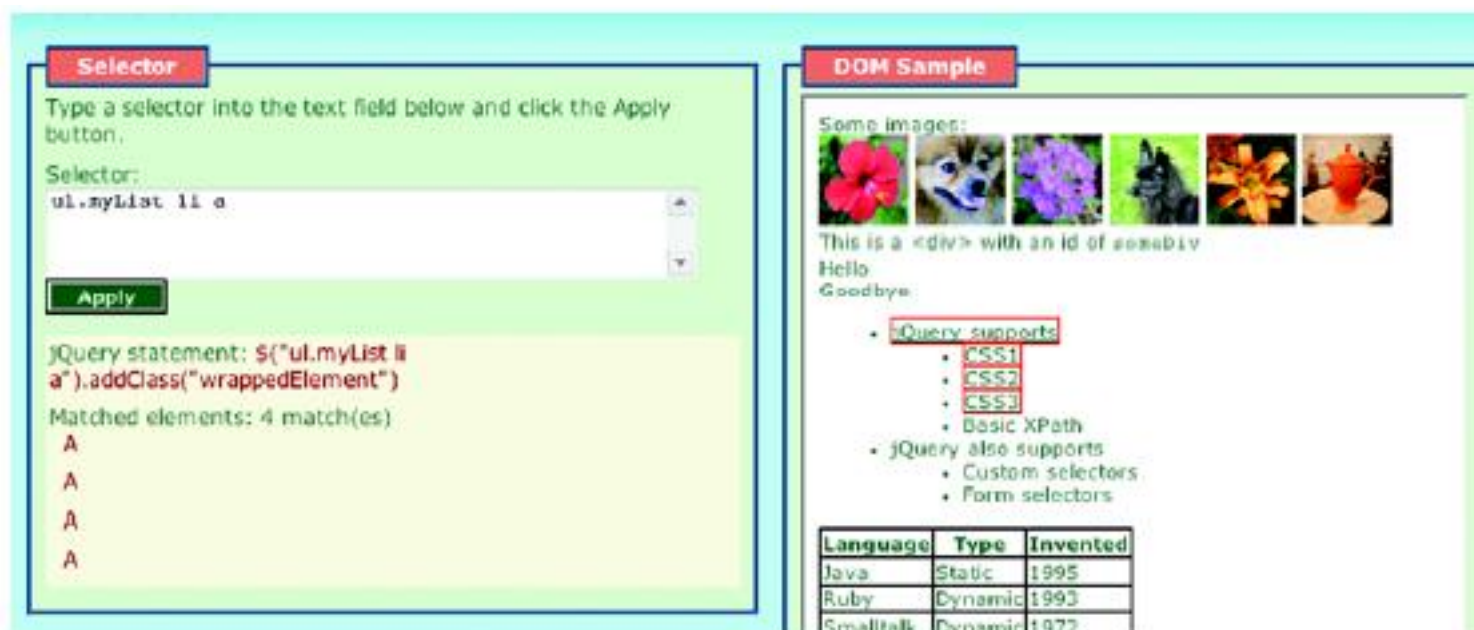


Figure 2.3 All anchor tags that are descendents, at any depth, of an `` element are selected

`ul.myList li a`

راه حل مناسب برای انتخاب چنین حالتی استفاده از **انتخاب فرزند** می باشد که به این منظور Parent (والد) و Child (فرزند)، به وسیله یک کاراکتر `>` از یکدیگر جدا می شوند:

`p > a`

این دستور تنها لینک (`<a>`) هایی را بر می گرداند که فرزند مستقیم یک عنصر `<p>` می باشند. بنابراین اگر در یک `<p>` لینکی در عنصر `` معرفی شده باشد، این لینک انتخاب نمی شود، چرا که فرزند مستقیم `<p>` به حساب نمی آید. در مورد مثال لینک های موجود در لیست، می توانیم دستور زیر را به منظور انتخاب لینک مورد نظرمان استفاده کنیم:

`ul.myList > li > a`

دستور انتخاب فوق از میان عناصر ``، عنصری را که دارای کلاس `myList` می باشد، انتخاب می کند و پس از آن لینک هایی (`<a>`) که فرزند مستقیم گزینه های آن هستند، برگردانده می شوند. همانگونه که در شکل زیر مشاهده می کنید لینک های زیرمجموعه عنصر `` انتخاب نمی شوند، زیرا فرزند مستقیم این عنصر محسوب نمی شوند.

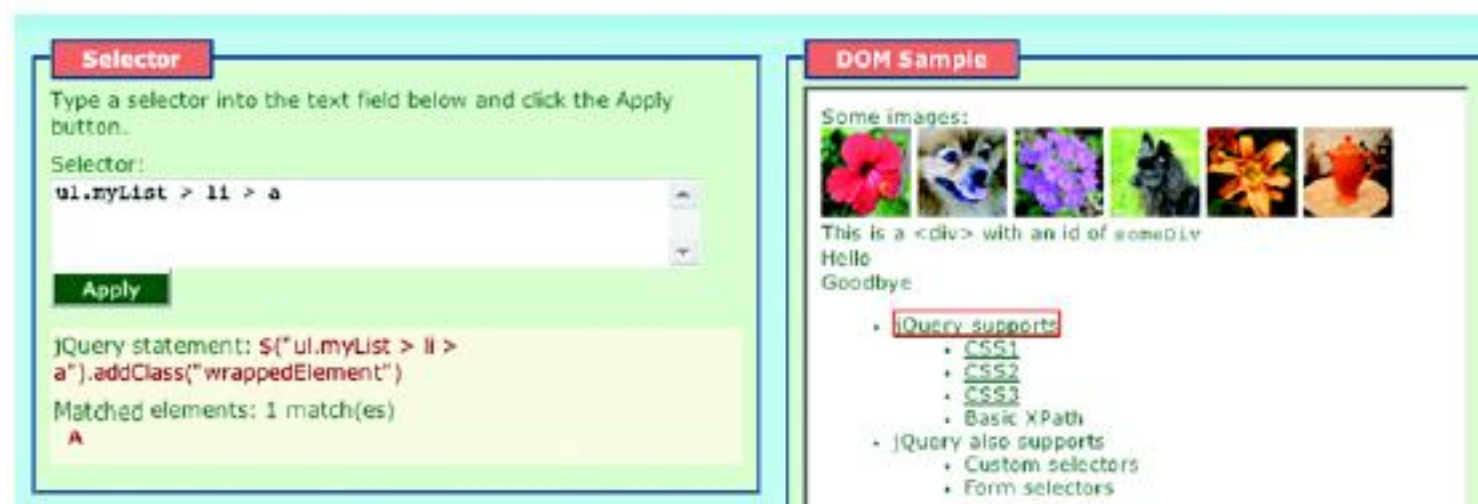


Figure 2.4 With the selector `ul.myList > li > a`, only the direct children of parent nodes are matched.

انتخاب کننده‌های صفت نیز بسیار قدرتمند می‌باشند و ما را توانا تر می‌سازند، فرض کنید برای منظوری خاص قصد دارید به تمام لینک‌های موجود در صفحه که به مکانی خارج از این وب سایت اشاره دارند، رفتاری را اضافه کنید (مثلا مانند همین سایت به کنار آنها یک آیکن اضافه نمایید). فرض کنید این کد (کد موجود در مثال کارگاهی) را در صفحه خود دارید:

```
<li><a href="http://jquery.com">jQuery supports</a>
  <ul>
    <li><a href="css1">CSS1</a></li>
    <li><a href="css2">CSS2</a></li>
    <li><a href="css3">CSS3</a></li>
    <li>Basic XPath</li>
  </ul>
</li>
```

موردی که یک لینک با اشاره به وب سایت خارجی را از سایر لینک‌ها متمایز می‌سازد، شروع شدن مقدار صفت href آن با `http://` می‌باشد. انتخاب لینک‌هایی که مقدار href آنها با `http://` آغاز می‌شود، به سهولت و از طریق دستور زیر صورت می‌پذیرد:

```
a[href^=http://]
```

این دستور باعث انتخاب تمام لینک‌هایی که مقدار صفت href آنها **دقیقا** با `http://` آغاز می‌شود، می‌گردد. علامت `^` موجب می‌شود تا بررسی، لزوماً از ابتدای مقادیر صورت پذیرد و از آنجا که استفاده از این کاراکتر در سایر **عبارات منظم** به همین منظور صورت می‌پذیرد، به خاطر سپردن آن دشوار نخواهد بود. **می‌توانید این کد را در صفحه کارگاهی تست کنید.** راهای دیگری برای استفاده از انتخاب کننده‌های صفت وجود دارد.

```
form[method]
```

این دستور تمام عناصر `<form>` را که یک صفت `method` دارند را انتخاب می‌کند.

```
input[type=text]
```

این انتخاب کننده تمام عناصر input را که type آنها برابر text باشد انتخاب می‌کند.
دستور زیر مثالی دیگر برای بررسی یک مقدار بر اساس کاراکترهای نخست آن می‌باشد:

```
div[title^=my]
```

همانطور که از دستور فوق بر می‌آید، عناصر div که مقدار title آنها با رشته my آغاز می‌شود، هدف این انتخاب کننده خواهد بود. اما اگر بخواهیم تنها بر اساس کاراکترهای انتهایی انتخابی انجام دهیم، دستور مناسب چه خواهد بود؟ برای چنین منظوری مانند زیر عمل می‌کنیم:

```
a[href$=.pdf]
```

این دستور کاربرد زیادی برای شناسایی لینک‌های اشاره کننده به فایل‌های pdf دارد. ساختار زیر نیز زمانی استفاده می‌شود که یک عبارت منظم در جایی از یک صفت قرار گرفته باشد، خواه این عبارت از کاراکتر دوم آغاز شده باشد و یا از هر جای دیگر.

```
a[href*=jquery.com]
```

همانگونه که انتظار می‌رود این انتخاب کننده، تمام لینک‌هایی که به وب سایت jQuery اشاره دارند را برمی‌گرداند. فراتر از خصوصیات، بعضی مواقع ما می‌خواهیم بررسی کنیم که آیا یک عنصر شامل عنصر دیگری هست یا خیر. در مثال‌های قبلی فرض کنید ما می‌خواهیم بدانیم که آیا یک li شامل a هست یا خیر، jQuery با استفاده از انتخاب کننده‌های Containerها این را پشتیبانی می‌کند:

```
li:has(a)
```

این انتخاب کننده همه li‌هایی را برمی‌گرداند که شامل لینک (<a>) هستند. دقت کنید که این انتخاب گر مانند li a نیست، انتخاب گر دوم تمامی لینک‌هایی را که در li هستند بر می‌گرداند اما دستور بالا li‌هایی را بر می‌گرداند که دارای لینک (<a>) هستند.

تصویر زیر انتخاب گرهایی را نشان می‌دهد که ما می‌توانیم در jQuery استفاده نماییم.

Table 2.1 The basic CSS Selectors supported by jQuery

Selector	Description
*	Matches any element.
E	Matches all element with tag name E.
E F	Matches all elements with tag name F that are descendents of E.
E>F	Matches all elements with tag name F that are direct children of E.
E+F	Matches all elements F immediately preceded by sibling E.
E-F	Matches all elements F preceded by any sibling E.
E:has(F)	Matches all elements with tag name E that have at least one descendent with tag name F.
E.C	Matches all elements E with class name C. Omitting E is the same as *.C.
E#I	Matches element E with id of I. Omitting E is the same as *#I.
E[A]	Matches all elements E with attribute A of any value.
E[A=V]	Matches all elements E with attribute A whose value is exactly v.
E[A^=V]	Matches all elements E with attribute A whose value begins with v.
E[A\$=V]	Matches all elements E with attribute A whose value ends with v.
E[A*=V]	Matches all elements E with attribute A whose value contains v.

انشالله در پست‌های بعدی ادامه مباحث را بررسی خواهد شد.

در ادامه مطلب قبلی [آموزش \(jQuery\) جی کوئری #3](#) به ادامه بحث می‌پردازیم.

با توجه به حالت‌های مختلف و گزینه‌های گوناگونی که انتخاب‌کننده‌ها در اختیار ما گذاشته‌اند، اگر هنوز دنبال قدرت بیشتری از انتخاب‌کننده‌ها هستید در ادامه به چند مورد از آنها اشاره خواهیم کرد.

3-1- انتخاب عناصر بر اساس موقعیت

گاهی اوقات انتخاب عناصر با توجه به مکان آنها و یا موقعیت مکانی آن‌ها نسبت به سایر اجزا صورت می‌پذیرد؛ برای مثال اولین لینک صفحه و یا اولین لینک هر پاراگراف و یا گزینه‌ی آخر از لیست، jQuery شیوه‌ای خاص را برای چنین انتخاب‌هایی ارائه کرده است. برای مثال دستور زیر اولین لینک موجود در صفحه را انتخاب می‌کند:

```
a:first
```

دستور زیر چکاری انجام می‌دهد؟

```
p:odd
```

دستور بالا تمامی پاراگراف‌های فرد را انتخاب می‌کند. روش‌های دیگری هم ممکن است بخواهیم استفاده کنیم؛ مثلاً دستور زیر تمامی پاراگراف‌های زوج را انتخاب می‌کند:

```
p:even
```

یا با استفاده از دستور زیر میتوان آخرین فرزند یک والد را انتخاب کرد؛ در زیر آخرین فرزند یک انتخاب می‌شود. علاوه بر انتخاب‌کننده‌هایی که ذکر شد؛ تعداد قابل توجه دیگری نیز وجود دارند که در جدول 2-2 ذکر شده‌اند.

جدول 2-2: انتخاب گره‌های پیشرفته موقعیت عناصر که توسط jQuery پشتیبانی می‌شوند

توضیح	فیلتر
اولین عنصر که با شرط ما مطابقت می‌کند را انتخاب می‌کند، li a:first اولین لینکی را که فرزند لیست به حساب می‌آید؛ را بر می‌گرداند	:first
آخرین عنصری که با شرط ما مطابقت کند را انتخاب می‌کند. li a:last آخرین لینک از فرزندان لیست را برمی‌گرداند.	:last
اولین فرزند عنصر که با شرط ما مطابقت می‌کند را انتخاب می‌کند. li a:first-child اولین عنصر لینک از هر لیست را برمی‌گرداند.	:first-child
آخرین فرزند عنصر که با شرط ما مطابقت می‌کند را انتخاب می‌کند. li a:last-child اولین عنصر لینک از هر لیست را	:last-child

توضیح	فیلتر
برمی گرداند.	
تمام عناصری که پدر آنها تنها همان فرزند را داد، برمی گرداند.	:only-child
n امین فرزند عنصری که با شرط ما مطابقت داشته باشد را انتخاب می کند. <code>li:nth-child(2) //comment</code> دومین عنصر از هر لیست را برمی گرداند.	:nth-child(n)
فرزندان زوج یا فرد عنصر را انتخاب می کند. <code>li:nth-child(even) //comment</code> تمام عناصر زوج لیست ها را برمی گرداند.	:nth-child(even یا odd)
n امین فرزند عنصری که از طریق فرمول ارایه شده به دست می آید را انتخاب می کند. اگر ۷ صفر باشد، نیازی به نوشتن آن نیست. <code>li:nth-child(3n) //comment</code> تمام عناصر ضریب 3 لیست ها را بر می گرداند، در حالی که <code>li:nth-child(5n+1) //comment</code> عناصری از لیست را برمی گرداند که بعد از عنصرهای ضریب 5 لیست ها قرار گرفته باشند.	:nth-child(Xn+Y)
تمام عناصر زوج یا فرد که با شرط ما مطابقت کنند را انتخاب می کند. <code>li:even</code> تمامی عناصر زوج لیست ها را بر می گرداند.	:even یا :odd
n امین عنصر انتخاب شده را برمی گرداند.	:eq(n)
عناصر بعد از n امین عنصر را بر می گرداند. (در واقع عناصری که بزرگتر از عنصر n ام هستند را بر می گرداند)	:gt(n)
عناصر قبل از n امین عنصر را بر می گرداند. (در واقع عناصری که کوچکتر از عنصر n ام هستند را بر می گرداند)	:lt(n)

پ.ن: در جدول بالا در توضیحات بعضی از انتخاب گر ها `//comment` نوشته شده است، اینها جز دستور نبوده و فقط برای نمایش صحیح پرانتز در صفحه اینترنتی نوشته شده است، در عمل نیازی به اینها نیست.

نکته ای که در مورد انتخاب گره های جدول بالا وجود دارد این است که در فیلتر `:nth-child` برای سازگاری با CSS، مقدار شمارشگر از 1 آغاز می شود، اما در سایر فیلترها از قاعده ای که اکثر زبانهای برنامه نویسی استفاده شده است و شمارشگر آنها از صفر آغاز می شود. برای درک این موضوع مثال زیر را در نظر بگیرید:

```
<table id="languages">
  <thead>
    <tr>
      <th>Language</th>
      <th>Type</th>
      <th>Invented</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Java</td>
      <td>Static</td>
      <td>1995</td>
    </tr>
    <tr>
      <td>Ruby</td>
      <td>Dynamic</td>
      <td>1993</td>
    </tr>
    <tr>
      <td>Smalltalk</td>
```

```
<td>Dynamic</td>
<td>1972</td>
</tr>
<tr>
<td>C++</td>
<td>Static</td>
<td>1983</td>
</tr>
</tbody>
</table>
```

حال می‌خواهیم از این جدول، محتویات تمام خانه‌هایی که نام یک زبان برنامه نویسی در آنهاست را انتخاب نماییم. از آنجا که نام این زبانها در اولین ستون از هر سطر قرار دارد. می‌توانیم دستوری مانند زیر بنویسیم:

```
table#languages tbody td:first-child
```

و یا با استفاده از دستور زیر این کار را انجام دهیم:

```
table#languages tbody td:nth-child(1)
```

اما دستور اول مختصرتر و خواناتر است. پس از آن برای دسترسی به نوع هریک از زبانهای برنامه نویسی، دستور انتخاب کننده دوم را به صورت

```
:nth-child(2)
```

تغییر می‌دهیم، و همچنین با تغییر پارامتر 2 به 3 سالی که هر یک از زبانها ابداع شده اند ، انتخاب می‌شوند. بدیهی است در این حالت دو دستور

```
:nth-child(3)
یا
:last-child
```

با یکدیگر برابرند. اما هردوی آنها ستون آخر از هر سطر را انتخاب می‌کنند، در شرایطی که بخواهیم آخرین خانه جدول انتخاب شود (خانه ای با مقدار 1983)، از `td:last` استفاده می‌کنیم. توجه کنید در حالی که دستور `td:eq(2) // comment` خانه ای با مقدار 1995 را انتخاب می‌کند، دستور `td:nth-child(2) // comment` تمام خانه‌های بیان کننده نوع زبانها را انتخاب می‌کند. بنابراین به خاطر داشته باشید که مقدار ابتدایی شمارشگر فیلتر `eq` از صفر است و این مقدار برای فیلتر `:nth-child` یک تعیین شده است.

در پست بعدی انتخاب گره‌های CSS و فیلترهای سفارشی jQuery را بررسی خواهیم کرد.

در ادامه مطلب قبلی [آموزش \(jQuery\) جی کوئری #4](#) به ادامه بحث می‌پردازیم. در پست قبل به بررسی **انتخاب عناصر بر اساس موقعیت** پرداختیم، در این پست به بحث "استفاده از انتخاب کننده‌های سفارشی jQuery" خواهیم پرداخت.

1-4 استفاده از انتخاب کننده‌های سفارشی jQuery

در پست‌های قبلی ([^](#) و [^](#)) تعدادی از انتخاب کننده‌های CSS که هر کدامشان موجب قدرت و انعطاف پذیری انتخاب اشیا موجود در صفحه می‌شوند را بررسی کردیم. با این وجود فیلترهای انتخاب کننده قدرتمندتری وجود دارند که توانایی ما را برای انتخاب بیشتر می‌کنند.

به عنوان مثال اگر بخواهید از میان تمام چک باکس‌ها، گزینه‌هایی را که تیک خورده اند انتخاب نمایید، از آنجا که تلاش برای مطابقت حالت‌های اولیه کنترل‌های HTML را بررسی می‌کنیم، jQuery انتخابگر سفارشی **checked** را پیشنهاد می‌کند، که مجموعه از عناصر را که خاصیت **checked** آنها فعال باشد را برای ما برمی‌گرداند. براس مثال انتخاب کننده **input** تمامی المان‌های **<input>** را انتخاب می‌کند، و انتخاب کننده **input:checked** تمامی **input**هایی را انتخاب می‌کند که **checked** هستند. انتخاب کننده سفارشی **checked**: یک انتخاب کننده خصوصیت CSS عمل می‌کند (مانند `[foo=bar]`). ترکیب این انتخاب کننده‌ها می‌تواند قدرت بیشتری به ما بدهد، انتخاب کننده‌هایی مانند **radio:checked** و **checkbox:checked**.

همانطور هم که قبلاً بیان شد، jQuery علاوه بر پشتیبانی از انتخاب کننده‌های CSS تعدادی انتخاب کننده سفارشی را نیز شامل می‌شود که در جدول 2-3 شرح داده شده است.

جدول 2-3: انتخاب کننده‌های سفارشی jQuery

انتخاب کننده	توضیح
animated:	عناصری را انتخاب می‌کند که تحت کنترل انیمیشن می‌باشند. در پست‌های بعدی انیمیشن‌ها توضیح داده می‌شوند.
button:	عناصر دکمه را انتخاب می‌کند، عناصری مانند <code>input[type=submit]</code> , <code>input[type=reset]</code> , <code>input[type=button]</code> (یا <code>button</code>).
checkbox:	عناصر Checkbox را انتخاب می‌کند، مانند <code>(input[type=checkbox])</code> .
checked:	عناصر checkbox ها یا دکمه‌های رادیویی را انتخاب می‌کند که در حالت انتخاب باشند.
contains(foo) :	عناصری را انتخاب می‌کند که دارای عبارت foo باشند.
disabled:	عناصر در حالت disabled را انتخاب می‌کند.
enabled:	عناصر در حالت enabled را انتخاب می‌کند.
file:	عناصر فایل را انتخاب می‌کند، مانند <code>(input[type=file])</code> .
header:	عناصر هدر مانند h1 تا h6 را انتخاب می‌کند.
hidden:	عناصر مخفی شده را انتخاب می‌کند.
image:	عناصر تصویر را انتخاب می‌کند، مانند <code>(input[type=image])</code> .

انتخاب کننده	توضیح
input:	عناصر فرم مانند input, select, textarea, button را انتخاب می‌کند.
not(filter)	انتخاب کننده‌ها را برعکس می‌کند.
parent:	عناصری که فرزندی دارند را انتخاب می‌کند.
password:	عناصر password را انتخاب می‌کند، مانند (input[type=password]).
radio:	عناصر radio را انتخاب می‌کند، مانند (input[type=radio]).
reset:	دکمه‌های reset را انتخاب می‌کند، مانند (input[type=reset]) یا (button[type=reset]).
selected:	عناصری (عناصر option) را انتخاب می‌کند که در وضعیت selected قرار دارند.
submit:	دکمه‌های submit را انتخاب می‌کند، مانند (input[type=submit]) یا (button[type=submit]).
text:	عناصر text را انتخاب می‌کند، مانند (input[type=text]).
visible:	عناصری را که در وضعیت visible باشند انتخاب می‌کند.

بسیاری از انتخاب کننده‌های سفارشی jQuery بررسی شده برای انتخاب عناصر فرم ورود اطلاعات کاربر استفاده می‌شوند. این فیلترها قابلیت ادغام را دارند، برای مثال در زیر دستوری را به منظور انتخاب آن دسته از گزینه‌های Checkbox که تیک خورده اند و فعال هستند را مشاهده می‌کنید:

```
checkbox:checked:enabled
```

این فیلترها و انتخاب کننده‌ها کاربردهای وسیعی در صفحات اینترنتی دارند، آیا آنها حالت معکوسی نیز دارند؟

استفاده از فیلتر not:

برای آنکه نتیجه انتخاب کننده‌ها را معکوس کنیم می‌توانیم از این فیلتر استفاده کنیم. برای مثال دستور زیر تمام عناصری را که checkbox نیستند را انتخاب می‌کند:

```
input:not(:checkbox)
```

اما استفاده از این فیلتر دقت زیادی را می‌طلبد زیرا به سادگی ممکن است با نتیجه ای غیر منتظره مواجه شویم.

استفاده از فیلتر has:

در [اینجا](#) دیدیم که CSS انتخاب کننده قدرتمندی را ارائه کرده است که فرزندان یک عنصر را در هر سطحی که باشند (حتی اگر فرزند مستقیم هم نباشند) انتخاب می‌کند. برای مثال دستور زیر تمام عناصر span را که در div معرفی شده باشند را انتخاب می‌کند:

```
div span
```

اما اگر بخواهیم انتخابی برعکس این انتخاب داشته باشیم، باید چه کنیم؟ برای این کار باید تمام divهایی که دارای عنصر span می‌باشد را انتخاب کرد. برای چنین انتخابی از فیلتر has: استفاده می‌کنیم. به دستور زیر توجه نمایید، این دستور تمام عناصر div

را که در آنها عنصر span معرفی شده است را انتخاب می‌کند:

```
div:has(span)
```

برای برخی انتخاب‌های پیچیده و مشکل، این فیلتر و مکانیزم بسیار کارا می‌باشد و به سادگی ما را به هدف دلخواه می‌رساند. فرض کنید می‌خواهیم آن خانه از جدول که دارای یک عنصر عکس خاص می‌باشد را پیدا کنیم. با توجه به این نکته که آن عکس از طریق مقدار src قابل تشخیص می‌باشد، با استفاده از فیلتر has: دستوری مانند زیر می‌نویسیم:

```
$('tr:has(img[src$="foo.png"])')
```

این دستور هر خانه از جدول را که این عکس در آن قرار گرفته باشد را انتخاب می‌کند. همانگونه که دیدیم jQuery گزینه‌های بسیار متعددی را به منظور انتخاب عناصر موجود در صفحه برای ما مهیا کرده است که می‌توانیم هر عنصری از صفحه را انتخاب و سپس تغییر دهیم که تغییر این عناصر در پست‌های آینده بحث خواهد شد.

موفق و موید باشید.

نظرات خوانندگان

نویسنده: ^{سید باقر شفیعی}
تاریخ: ۱۳:۴۱ ۱۳۹۲/۰۱/۲۸

سلام مهندس
خیلی عالی بود - امیدارم وقت داشته باشی پست آموزشی بیشتری بذاری.
مرسی

نویسنده: ^{رها}
تاریخ: ۲۰:۳۰ ۱۳۹۲/۱۰/۰۷

سلام؛ من به اسلاید شو ساده را از آموزشهای به سایت انگلیسی زبان ساختم که مدتهاست دنبالش بودم اما هنوز به ایراد کوچولو داره و اونهم اینه که بعد از رسیدن به آخرین عکس برمیگرده به اول یعنی بصورت بک اسلاید میشه و اگر عکسها از سمت راست به چپ اسلاید میشوند وقتی به آخرین عکس میرسه تمام عکسها در کسری از ثانیه از چپ به راست برمیگردند. نمونه کد کوئری رو میزارم و ممنون میشم منو در این زمینه راهنمایی کنید که چطور کاری کنم با رسیدن به آخرین عکس به همون روش از سمت راست به چپ دوباره برگرده به عکس اول نه تمام عکسها رو از چپ به راست برگردونه ؟ اسکریپت فراخوانده شده :

```
< script src = "http://code.jquery.com/jquery-latest.js" ></ script >
```

کوئری نوشته شده :

```
<script type = "text/javascript" >
$(document).ready(function () {
    slideshow();
});

var n = 0;
function slideshow() {
    id = n % 5 + 1;
    leftpost = (1 - parseInt(id)) * 500 + "px";
    $("div.slider-item").animate({ left: leftpost }, 1500);
    n = n + 1;
    s = setTimeout("slideshow()", 3000);
}
</ script >
```

فایل css :

```
<style type = "text/css" >
div#slider {
    width: 500px;
    height: 300px;
    margin: auto;
    overflow: hidden;
    border: 10px solid gray;
}
div#slider-mask {
    width: 500%;
    height: 100%;
}
div.slider-item {
    width: 20%;
    height: 100%;
    position: relative;
    float: left;
}
</ style >
```

فایل html :

```
< div id = "slider" > < div id = "slider-mask" >
< div class = "slider-item" >< img src = "img1.jpg" alt = "1" /></ div >
< div class = "slider-item" >< img src = "img2.jpg" alt = "2" /></ div >
< div class = "slider-item" >< img src = "img3.jpg" alt = "3" /></ div >
< div class = "slider-item" >< img src = "img4.jpg" alt = "4" /></ div >
< div class = "slider-item" >< img src = "img5.jpg" alt = "5" /></ div >
</ div > </ div >
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۰۸ ۰:۵

قسمت اسکریپتی رو اینطوری تغییر بدین

```
<script type="text/javascript">
$(document).ready(function () {
    slideShow();
});

var leftPos = 0;
var numberOfImages = 5;
var sliderWidth = 500;
var ltr = true;

function slideShow() {
    $("div.slider-item").animate({ left: leftPos + "px" }, 1500);

    if(ltr){
        leftPos -= sliderWidth;
    }
    else{
        leftPos += sliderWidth;
    }

    if((Math.abs(leftPos) == (numberOfImages-1) * sliderWidth) || (leftPos == 0)){
        ltr = !ltr;
    }

    //console.log({ leftPos:leftPos , ltr: ltr });
    s = setTimeout("slideShow()", 3000);
}
</script>
```

نویسنده: رها
تاریخ: ۱۳۹۲/۱۰/۲۲ ۱۲:۳۳

سلام؛ وقتی یه جی کوئری یا اسکریپت رو دانلود میکنیم و در ویژوال استادیو باز میکنم بصورتی نوشته شده که تمام فایل در یک خط افقی هست و اگر بخواهیم ویرایشش کنیم همیشه با اسکروول کردن موس در امتدادش حرکت کنیم. میخواستم ببینم آیا در ویژوال استادیو ابزاری هست که اینچنین فایلها رو یکباره از حالت افقی و اینکه در یک خط هست هستند با طول زیاد رو بشکنه و بصورت زیر هم بنویسه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۲۲ ۱۲:۳۸

[JavaScript Deobfuscator](#)

در ادامه مطلب قبلی [آموزش \(jQuery\) جی کوئری #5](#) به ادامه بحث می‌پردازیم.

در پست‌های قبلی مروری بر jQuery داشته و در چند پست انواع روش‌های انتخاب عناصر صفحه وب را توسط jQuery بررسی کردیم. در پست‌های آینده با مباحث پیشرفته‌تری همچون انجام عملیاتی روی المانهای انتخاب شده، خواهیم پرداخت؛ امید است مفید واقع شود.

۲-۲ - ایجاد عناصر HTML جدید

گاهی اوقات نیاز می‌شود که یک یا چند عنصر جدید به صفحه‌ی در حال اجرا اضافه شوند. این حالت می‌تواند به سادگی قرار گرفتن یک متن در جایی از صفحه و یا به پیچیدگی ایجاد و نمایش یک جدول حاوی اطلاعات دریافت شده از بانک اطلاعاتی باشد. ایجاد عناصر به صورت پویا در یک صفحه در حال اجرا کار ساده‌ای برای jQuery می‌باشد، زیرا همانطور که در پست [آموزش \(jQuery\) جی کوئری #1](#) مشاهده کردیم (\$) با دریافت دستور ساخت یک عنصر HTML آن را در هر زمان ایجاد می‌کند، دستور زیر :

```
$("#<div>Hello</div>")
```

یک عنصر div ایجاد می‌کند و آماده افزودن آن به صفحه در هر زمان می‌باشد. تمامی توابع و متدهایی را که تاکنون بررسی کردیم قابل اعمال بروی اینگونه اشیا نیز می‌باشند. شاید در ابتدا ایجاد عناصر به این شکل خیلی مفید به نظر نرسد، اما زمانی که بخواهیم کارهای حرفه‌ای‌تری انجام دهیم؛ برای مثال کار با AJAX، خواهیم دید که تا چه اندازه ایجاد عناصر به این روش می‌تواند مفید باشد. دقت کنید که یک راه کوتاه‌تر نیز برای ایجاد یک عنصر <div> خالی وجود دارد که به شکل زیر است:

```
$("#<div>")
// همه اینها معادل هستند
$("#<div></div>")
$("#<div/>")
```

اما برای ایجاد عناصری که خود می‌توانند حاوی عناصر دیگر باشند استفاده از راههای کوتاه توصیه نمی‌شود مانند نوشتن تگ <script>. اما راههای زیادی برای انجام اینکار وجود دارد.

برای اینکه مزه اینکار را بچشید بد نیست نگاهی به مثال زیر بیندازید (نگران قسمت‌های نامفهوم نباشید به مرور با آنها آشنا خواهیم شد):

```
$("#<div class='foo'>I have foo!</div><div>I don't</div>")
  .filter(".foo").click(function() {
    alert("I'm foo!");
  }).end().appendTo("#someParentDiv");
```

در این مثال ابتدا ما یک المان div ایجاد کردیم که دارای کلاس foo می‌باشد، و خود شامل یک div دیگر است. در ادامه div که دارای کلاس foo بوده را انتخاب کرده و رویداد کلیک را به آن بایند کردیم. و در انتها این div را با محتوایش به المانی با Id=someParentDiv در سلسله مراتب DOM اضافه می‌کند. برای اجرا این کد می‌توانید کد آن را [دانلود](#) کرده و فایل chapter2/new.divs.html را اجرا کنید خروجی مانند تصویر زیر خواهد بود:

جهت تکمیل مطلب فعلی یک مثال کاملتر از این [سایت](#) جهت بررسی انتخاب کردم:

```
$( "<div/>", {
```

```
"class": "test",
text: "Click me!",
click: function() {
    $( this ).toggleClass( "test" );
}
}).appendTo( "body" );
```

در این مثال کمی پیشرفته‌تر یک div ایجاد شده کلاس test را برای آن قرار داده و عنوان آن را برابر text قرار می‌دهد و یک رویداد کلیک برای آن تعریف می‌کند و در نهایت آن را به body سایت اضافه می‌کند.

با توجه به اینکه مطالب بعدی طولانی بوده و تقریباً مبحث جدایی است؛ در پست بعدی به بررسی توابع و متدهای مدیریت مجموعه انتخاب شده خواهیم پرداخت.

پس از انواع روش‌های انتخاب عناصر در jQuery اکنون زمان آشنایی با متدها و توابعی جهت پردازش مجموعه انتخاب شده رسیده است.

۳-۲- مدیریت مجموعه انتخاب شده

هز زمان که مجموعه ای از عناصر انتخاب می‌شوند، خواه این عناصر از طریق انتخاب کننده‌ها انتخاب شده باشند و یا تابع (\$) در صدد ایجاد آن باشد، مجموعه ای در اختیار داریم که آماده دستکاری و اعمال تغییر با استفاده از متدهای jQuery می‌باشد. این متدها را در پست‌های آتی بررسی خواهیم کرد. اما اکنون به این نکته می‌پردازیم که اگر بخواهیم از همین مجموعه انتخاب شده زیر مجموعه ای ایجاد کنیم و یا حتی آن را گسترش دهیم، چه باید کرد؟ به طور کلی در این پست پیرامون این مورد بحث خواهد شد که چگونه می‌توانیم مجموعه انتخاب شده را به آن صورت که می‌خواهیم بهیود دهیم. برای درک مطالبی که قصد توضیح آنها را در این قسمت داریم، یک صفحه کارگاهی دیگر نیز در فایل قابل دانلود این [کتاب](#) موجود می‌باشد که با نام chapter2/lab.wrapped.set.html قابل دسترسی می‌باشد. نکته مهم در مورد این صفحه کارگاهی آن است که می‌بایست عبارات و دستورهای کامل را با ساختار صحیح وارد کنیم در غیر این صورت این صفحه کاربردی نخواهد داشت.

۳-۲-۱- تعیین اندازه یک مجموعه عناصر

قبلا اشاره کردیم که مجموعه عناصر jQuery شباهت هایی با آرایه دارد. یکی از این شباهت‌ها داشتن ویژگی [length](#) می‌باشد که مانند آرایه در جاوااسکریپت، تعداد عناصر موجود در مجموعه را شامل می‌شود. افزون بر این ویژگی، jQuery یک متد را نیز معرفی کرده است که دقیقا شبیه به [length](#) عمل می‌کند. این متد [size\(\)](#) می‌باشد که استفاده از آن را در مثال زیر مشاهده می‌کنید.

```
$('#someDiv')
    .html('There are '+$('a').size()+' link(s) on this page.');
```

این مثال تمام لینک‌های موجود در صفحه را شناسایی می‌کند و سپس با استفاده از متد [size\(\)](#) تعداد آنها را بر می‌گرداند. در واقع یک رشته ایجاد می‌شود و در یک عنصر با شناسه someDiv قرار داده می‌شود. متد [html](#) در پست‌های آتی بررسی می‌شود. فرم کلی متد [size\(\)](#) را در زیر مشاهده می‌کنید.

size()

تعداد عناصر موجود در مجموعه را محاسبه می‌کند

پارامترها

بدون پارامتر

خروجی

تعداد عناصر مجموعه

اکنون که تعداد عناصر مجموعه را می‌دانیم چگونه می‌توانیم به هریک از آنها دسترسی مستقیم داشته باشیم؟

۳-۲-۲- بکارگیری عنصرهای مجموعه

به طور معمول پس از انتخاب یک مجموعه با استفاده از متدهای jQuery، عملی را بروی آن عناصر انتخاب شده انجام می‌دهیم، مانند مخفی کردن آنها با متد [hide\(\)](#)، اما گاهی اوقات می‌خواهیم بروی یک یا چند مورد خاص از عناصر انتخاب شده عملی را اعمال کنیم. jQuery چند روش مختلف را به منظور اینکار ارائه می‌دهد.

از آنجا که مجموعه عناصر انتخاب شده در jQuery مانند آرایه در جاوااسکریپت می‌باشد، بنابراین به سادگی می‌توانیم از اندیس برای دستیابی به عناصر مختلف مجموعه استفاده کنیم. برای مثال به منظور دسترسی به اولین عکس از مجموعه عکس‌های انتخاب

شده که دارای صفت alt می‌باشند از دستور زیر استفاده می‌کنیم:

```
$('#img[alt]')[0]
```

اما اگر ترجیح می‌دهید به جای اندیس از یک متد استفاده کنید، jQuery متد [get\(\)](#) را در نظر گرفته است:

get(index)

برای واکنشی یک یا تمام عناصر موجود در مجموعه استفاده می‌شود. اگر برای این متد پارامتری ارسال نشود، تمام عناصر را در قالب یک آرایه جاوااسکریپت بر می‌گرداند، اما در صورت ارسال یک پارامتر، تنها آن عنصر را بر می‌گرداند.

پارامتر

شماره اندیس یک عنصر که می‌بایست یک مقدار عددی باشد.

خروجی

یک یا آرایه ای از عناصر

دستور زیر مانند دستور قبلی عمل می‌کند:

```
$('#img[alt]').get(0)
```

متد [get\(\)](#) می‌تواند برای بدست آوردن یک آرایه از عناصر پیچیده نیز استفاده شود. مثلاً:

```
var allLabeledButtons = $('label+button').get();
```

خروجی دستور بالا لیست تمام button‌های موجود در صفحه است که بعد از عنصر label قرار گرفته اند، در نهایت این آرایه در متغیری به نام allLabeledButtons قرار خواهد گرفت.

در متد [get\(\)](#) دیدیم که با دریافت شماره اندیس یک عنصر، آن عنصر را برای ما برمی‌گرداند، عکس این عمل نیز امکان پذیر می‌باشد. فرض کنید می‌خواهیم از میان تمام عناصر عکس، شماره اندیس عکسی با شناسه findMe را بدست آوریم. برای این منظور می‌توانیم از کد زیر بهره ببریم:

```
var n = $('img').index($('img#findMe')[0]);
```

فرم کلی متد [index\(\)](#) به صورت زیر است:

index(element)

عنصر ارسالی را در مجموعه عناصر پیدا می‌کند، سپس شماره اندیس آن را بر می‌گرداند. اگر چنین عنصری در مجموعه یافت نشد خروجی 1- خواهد بود.

پارامتر

پارامتر این متد می‌تواند یک عنصر و یا یک انتخاب کننده باشد که خروجی انتخاب کننده نیز در نهایت یک عنصر خواهد بود.

خروجی

شماره اندیس عنصر در مجموعه

۳-۳-۲- برش و کوچک کردن مجموعه ها

ممکن است شرایطی پیش آید که پس از بدست آوردن یک مجموعه عناصر انتخاب شده نیاز باشد که عنصری به آن مجموعه اضافه و یا حتی عنصری را از آن حذف کنیم تا در نهایت مجموعه ای باب میل ما بدست آید. برای انجام چنین تغییرهایی در یک مجموعه jQuery کلیکسیون بزرگی از متدها را برای ما به همراه دارد. اولین موردی که به آن می‌پردازیم، افزودن یک عنصر به مجموعه می‌باشد.

اضافه کردن عناصر بیشتر به یک مجموعه عنصر انتخاب شده

همواره ممکن است شرایطی پیش آید که پس از ایجاد یک مجموعه عناصر انتخاب شده، بخواهیم عنصری را به آن اضافه کنیم. یکی از دلایلی که باعث می‌شود این امر در jQuery بیشتر مورد نیاز باشد توانایی استفاده از متدهای زنجیره ای در jQuery است. ابتدا یک مثال ساده را بررسی می‌کنیم. فرض کنید می‌خواهیم تمام عناصر عکس که دارای یکی از دو خصوصیت alt و یا title می‌باشند را انتخاب کنیم، با استفاده از انتخاب کننده‌های قدرتمند jQuery دستوری مانند زیر خواهیم نوشت:

```
$('img[alt],img[title]')
```

اما برای آنکه با متد [add\(\)](#) که به منظور افزودن عنصر به مجموعه عناصر می‌باشد آشنا شوید این مثال را به صورت زیر می‌نویسیم:

```
$('img[alt]').add('img[title]')
```

استفاده از متد [add\(\)](#) به این شکل موجب می‌شود تا بتوانیم مجموعه‌های مختلف را به یکدیگر متصل کنیم و یک مجموعه کلی‌تر از عناصر انتخاب شده ایجاد کنیم. متد [add\(\)](#) در این حالت مانند متد [end\(\)](#) عمل می‌کند که در قسمت ۲-۳-۶ شرح داده خواهد شد. ساختار کلی متد [add\(\)](#) به صورت زیر است:

add(expression)

ابتدا یک کپی از مجموعه انتخاب شده ایجاد می‌کند، سپس با افزودن محتویات پارامتر expression به آن نمونه، یک مجموعه جدید تشکیل می‌دهد. پارامتر expression می‌تواند حاوی یک انتخاب کننده، قطعه کد HTML، یک عنصر و یا آرایه ای از عناصر باشد.

پارامتر

در این پارامتر مواردی (مانند رشته، آرایه، المان) که می‌خواهیم به مجموعه عناصر انتخاب شده اضافه شوند قرار می‌گیرد. که می‌تواند انتخاب کننده، قطعه کد HTML، یک عنصر و یا آرایه ای از عناصر باشد.

خروجی

یک کپی از مجموعه اصلی به علاوه موارد اضافه شده.

اصلاح عناصر یک مجموعه عنصر انتخاب شده

در قسمت قبل دیدیم که چگونه با استفاده از متد [add\(\)](#) و با بکار گیری آن در توابع زنجیره ای، توانستیم عناصری جدید به مجموعه انتخاب شده اضافه کنیم. عکس این عمل را نیز می‌توان با استفاده از متد [not\(\)](#) در توابع زنجیره ای انجام داد. این متد عملکردی شبیه به فیلتر [:not](#) دارد، اما با این تفاوت که بکار گیری آن مانند متد [add\(\)](#) می‌باشد و می‌توان در هر جایی از زنجیره از آن استفاده کرد تا عناصر مورد نظر را از مجموعه انتخاب شده حذف کنیم.

فرض کنید می‌خواهیم تمامی عناصر عکسی را که دارای خصوصیت title می‌باشند به استثنای آن موردی که واژه puppy در مقدار مربوط به این صفت استفاده کرده اند را انتخاب کنیم. این کار به سادگی و با استفاده از دستوری مانند]]

img[title]:not([title="puppy"]) می‌توان انجام داد. اما برای آن که مثالی از چگونگی کار متد [not\(\)](#) ببینید، این کار را به شکل زیر انجام می‌دهیم:

```
$('img[title]').not('[title="puppy"]')
```

این دستور تمام عکس‌های دارای خصوصیت title را به استثنا titleهایی که مقدار puppy در آنها وجود دارد را انتخاب می‌کند. شکل کلی متد [not\(\)](#) مانند زیر است:

not(expression)

ابتدا یک کپی از مجموعه انتخاب شده ایجاد می‌کند، سپس از آن کپی عناصری را که expression مشخص می‌کند را حذف می‌نماید.

پارامتر

این پارامتر تعیین کننده عناصر در نظر گرفته شده برای حذف می‌باشد. این پارامتر می‌تواند یک عنصر، آرایه ای از عناصر، انتخاب کننده و یا یک تابع باشد.

اگر این پارامتر تابع باشد، تک تک عناصر مجموعه به آن ارسال می‌شوند و هر یک که خروجی تابع را برابر با مقدار true کند، حذف می‌شود.

خروجی

یک کپی از مجموعه اصلی بدون موارد حذف شده.

این شیوه برای ایجاد مجموعه‌هایی که انتخاب‌کننده‌ها قادر به ساخت آن‌ها نمی‌باشند، کاربرد بسیار مناسبی دارد، زیرا از تکنیک‌های برنامه نویسی استفاده می‌کند و دست ما را برای اعمال انتخاب‌های گوناگون باز می‌کند. اگر در شرایطی خاص با حالتی روبرو شدید که احساس کردید عکس این انتخاب برای شما کارایی دارد، باز می‌توانید از یکی دیگر از متدهای jQuery استفاده کنید، متد [filter\(\)](#) عملکردی مشابه با متد [not\(\)](#) دارد با این تفاوت که عناصری از مجموعه حذف می‌شوند که خروجی تابع را false کنند. فرض کنید می‌خواهیم تمام عناصر td که دارای یک عنصر عددی می‌باشند را انتخاب کنیم. با وجود قدرت فوق العاده انتخاب‌کننده‌های jQuery به ما ارایه می‌دهند، انجام چنین کاری با استفاده از انتخاب‌کننده‌ها غیر ممکن است. در این حالت از متد [filter\(\)](#) را به شکل زیر استفاده می‌کنیم:

```
$('td').filter(function(){return this.innerHTML.match(/^\\d+$/)});
```

دستور فوق یک مجموعه از تمام عناصر td انتخاب می‌کند، سپس تک تک عناصر مجموعه انتخاب شده را به تابعی که پارامتر متد [filter\(\)](#) می‌باشد، ارسال می‌کند. این تابع با استفاده از عبارت منظم مقدار عنصر کنونی را می‌سنجد. اگر این مقدار یک یا زنجیره‌ای از ارقام بود، خروجی تابع true خواهد بود، و آن عنصر از مجموعه حذف نمی‌شود، اما اگر این مقدار عددی نبود، خروجی تابع false بوده و عنصر از مجموعه کنار گذاشته می‌شود. شکل کلی متد [filter\(\)](#) به شکل زیر است.

filter(expression)

ابتدا یک کپی از مجموعه انتخاب شده ایجاد می‌کند، سپس از آن کپی عناصری را که expression مشخص می‌کند را حذف می‌نماید.

پارامتر

این پارامتر تعیین‌کننده عناصر در نظر گرفته شده برای حذف می‌باشد. این پارامتر می‌تواند یک عنصر، ارایه‌ای از عناصر، انتخاب‌کننده و یا یک تابع باشد. اگر این پارامتر تابع باشد، تک تک عناصر مجموعه به آن ارسال می‌شوند و هر یک که خروجی تابع را برابر با مقدار false کند، حذف می‌شود.

خروجی

یک کپی از مجموعه اصلی بدون عناصر حذف شده.

ایجاد یک زیر مجموعه از مجموعه عناصر انتخاب شده

گاهی اوقات داشتن یک زیر مجموعه از عناصر یک مجموعه، چیزی است که دنبال آن هستیم. برای این منظور jQuery متد [slice\(\)](#) را ارایه می‌کند که عناصر را بر اساس جایگاه آن‌ها به زیر مجموعه‌هایی کوچکتر تقسیم می‌کند. نتیجه استفاده از این متد یک مجموعه جدید برگرفته از تعدادی عناصر پشت سر هم، از یک مجموعه انتخاب شده خواهد بود: شکل کلی متد [slice\(\)](#) مانند زیر است:

slice(begin, end)

ایجاد و برگرداندن یک مجموعه جدید از بخشی از عناصر پشت سر هم در یک مجموعه اصلی.

پارامتر

begin: پارامتر begin که یک پارامتر عددی می‌باشد و مقدار اولیه آن از صفر آغاز می‌شود، نشان دهنده اولین عنصری است که می‌خواهیم در مجموعه جدید حضور داشته باشد. **end:** پارامتر دوم که آن هم یک پارامتر عددی می‌باشد و از صفر آغاز می‌شود، در این متد اختیاری است. این پارامتر اولین عنصری است که نمی‌خواهیم از آن به بعد در مجموعه جدید حضور داشته باشد را مشخص می‌کند. اگر مقداری برای این پارامتر ننویسیم، به صورت پیش فرض تا انتهای مجموعه انتخاب می‌شود.

خروجی

یک مجموعه عنصر جدید.

اگر بخواهیم از یک مجموعه کلی، تنها یک عنصر را در قالب یک مجموعه انتخاب کنیم می‌توانیم از متد [slice\(\)](#) استفاده کنیم و مکان آن عنصر در مجموعه را به آن ارسال کنیم. دستور زیر مثالی از این حالت می‌باشد:

```
$('.*').slice(2,3);
```

این مثال ابتدا تمام عناصر موجود در صفحه را انتخاب می‌کند، سپس سومین عنصر از آن مجموعه را در یک مجموعه جدید باز می‌گرداند. دقت کنید که دستور فوق با دستور `$('.*').get(2)` کاملاً متفاوت است، چرا که خروجی این دستور تنها یک عنصر است، در حالی که خروجی دستور فوق یک مجموعه است. از همین رو دستور زیر باعث ایجاد یک مجموعه که شامل چهار عنصر اولیه صفحه می‌باشد، می‌شود.

```
$('.*').slice(0,4);
```

برای ایجاد یک مجموعه از عناصر انتهایی موجود در صفحه نیز می‌توان از دستوری مانند زیر استفاده کرد:

```
$('.*').slice(4);
```

این دستور تمام عناصر موجود در صفحه را انتخاب می‌کند، سپس مجموعه ای جدید می‌سازد که تمام عناصر به استثنای چهار عنصر اول را در خود جای می‌دهد.

۲-۳-۴- ایجاد مجموعه بر اساس روابط

jQuery به ما این توانایی را داده است تا مجموعه هایی را انتخاب کنیم، که اساس انتخاب عناصر، رابطه سلسله مراتبی آنها با عناصر HTML صفحه باشد. اکثر این متدها یک پارامتر اختیاری از نوع انتخاب کننده دریافت می‌کنند که می‌تواند برای انتخاب عناصر مجموعه استفاده شود. در صورتی که چنین پارامتری ارسال نگردد، تمام عناصر واجد شرایط متد در مجموعه انتخاب می‌شوند.

جدول ۲-۴- متدهای موجود برای ایجاد مجموعه‌های جدید بر اساس روابط

توضیح	متد
مجموعه ای را برمی گرداند که شامل تمام فرزندان بدون تکرار از عناصر مجموعه می‌باشد.	children()
مجموعه ای شامل محتویات تمام عناصر برمی گرداند. (از این متد معمولاً برای عناصر <code>iframe</code> استفاده می‌شود)	contents()
مجموعه ای شامل فرزندان پدرش که بعد از خود این عنصر می‌باشند را برمی گرداند. این مجموعه عنصر تکراری ندارد.	next()
مجموعه ای شامل تمام فرزندان پدرش که بعد از خود این عنصر می‌باشند را بر می‌گرداند.	nextAll()
مجموعه ای شامل نزدیک‌ترین پدر اولین عنصر مجموعه را بر می‌گرداند.	parent()
مجموعه ای شامل تمام پدران مستقیم عناصر مجموعه را بر می‌گرداند. این مجموعه عنصر تکراری ندارد.	parents()
مجموعه ای شامل فرزندان پدرش که قبل از خود این عنصر می‌باشند را برمی گرداند. این مجموعه عنصر تکراری ندارد.	prev()

توضیح	متد
مجموعه ای شامل تمام فرزندان پدرش که قبل از خود این عنصر می‌باشند را بر می‌گرداند.	() prevAll
مجموعه ای بدون عنصر تکراری را بر می‌گرداند که شامل تمام فرزندان پدر خود عنصر خواهد بود.	() siblings

تمامی جدول بالا غیر از متد [contents\(\)](#) پارامتری از نوع رشته که انتخاب کننده برای متد می‌باشند، استفاده می‌کند.

۳-۵-۲- استفاده از مجموعه‌های انتخاب شده برای انتخاب عناصر

با وجود اینکه تاکنون با شمار زیادی از توانایی‌های انتخاب و انتخاب کننده‌ها در jQuery آشنا شده اید، هنوز چند مورد دیگر نیز برای افزایش قدرت انتخاب باقی مانده است. متد [find\(\)](#) بروی یک مجموعه عناصر انتخاب شده به کار گرفته می‌شود و یک پارامتر ورودی نیز دارد. این پارامتر که یک انتخاب کننده است تنها بروی فرزندان این مجموعه اعمال می‌شود. برای مثال فرض کنید یک مجموعه از عناصر انتخاب و در متغیر `wrapperSet` قرار گرفته است. با دستور زیر می‌توانیم تمام عناصر (تگ) `cite` را که درون یک تگ `p` قرار گرفته اند را انتخاب کنیم، به شرطی که آن‌ها فرزندان عناصر مجموعه `wrapperSet` باشند:

```
wrapperSet.find('p cite')
```

البته می‌توانیم این تکه کد را به صورت زیر هم بنویسیم:

```
$('p cite', wrapperSet)
```

مانند سایر متدهای معرفی شده قدرت اصلی این متد نیز هنگام استفاده در متدهای زنجیره ای مشخص می‌شود. شکل کلی متد [find\(\)](#) مانند زیر است:

find(selector)

یک مجموعه عنصر جدید ایجاد می‌کند که شامل فرزندان عناصر مجموعه قبل می‌شود.

پارامتر

یک انتخاب کننده است که در قالب یک رشته به این متد ارسال می‌شود.

خروجی

یک مجموعه عنصر جدید

جهت پیدا کردن عناصری که داخل یک `wrapperSet` می‌توانیم از متد دیگری به نام `contains()` نیز استفاده کنیم. این متد مجموعه ای را بر می‌گرداند که شامل تمام عناصری است که در انتخاب کننده پارامتر ورودی است. مثلاً

```
$('p').contains('Lorem ipsum')
```

این دستور تمامی عناصر `p` را که شامل `Lorem ipsum` است را بر می‌گرداند. قالب کلی متد مانند زیر است:

contains(text)

مجموعه ای از عناصر که شامل متن ورودی می‌باشند را بر می‌گرداند.

پارامتر

رشته ورودی که می‌خواهیم در عنصر فراخوان متد جستجو شود.

خروجی

مجموعه ای از عناصر از نوع فراخوان متد را بر می‌گرداند که شامل متن ورودی باشد.

آخرین متدی که به بررسی آن می‌پردازیم متد [is\(\)](#) می‌باشد. با استفاده از این متد می‌توانیم اطمینان حاصل کنیم که دست کم یک

عنصر از مجموعه عناصر، شرایط مشخص شده توسط ما را دارا باشد. یک انتخاب کننده به این متد ارسال می‌شود، اگر عنصری از مجموعه عناصر انتخاب شد، خروجی متد true می‌شود و در غیر این صورت مقدار false بر گردانده خواهد شد. برای مثال:

```
var hasImage = $('*').is('img');
```

در صورت وجود دست کم یک عنصر عکس در کل عناصر صفحه، دستور بالا مقدار متغیر hasImage را برابر true قرار می‌دهد. قالب کلی متد [is\(\)](#) مانند زیر است:

is(selector)

بررسی می‌کند که آیا عنصری در مجموعه وجود دارد که انتخاب کننده ارسالی آن را انتخاب کند؟

پارامتر

یک انتخاب کننده است که در قالب یک رشته به این متد ارسال می‌شود.

خروجی

مقدار true در صورت وجود دست کم یک عنصر و false در صورت عدم وجود توسط تابع برگردانده می‌شود.

۲-۳-۶- مدیریت زنجیره‌های jQuery

تاکنون در مورد استفاده از متدها و توابع زنجیره ای زیاد بحث کرده ایم و انجام چندین عمل در یک دستور را به عنوان یک قابلیت بزرگ معرفی کرده ایم و البته از آن هم استفاده کردیم و در ادامه نیز استفاده خواهیم کرد. به کار گیری متدها به صورت زنجیره ای نه تنها موجب نوشتن کدهای قدرتمند و قوی به صورت مختصر و خلاصه می‌شود، بلکه از لحاظ کارایی نیز نکته مثبتی محسوب می‌شود، زیرا برای اعمال هر متد نیازی به محاسبه و انتخاب مجدد مجموعه نخواهد بود.

بنابراین متدهای مختلفی که در زنجیره استفاده می‌کنیم، برخی از آنها ممکن است مجموعه‌های جدیدی تولید کنند. برای مثال استفاده از متد [clone\(\)](#) موجب می‌شود تا مجموعه ای جدید از کپی عناصر در مجموعه اول ایجاد شود. زمانی که یکی از متدهای زنجیره یک مجموعه جدید را تولید می‌کند، دیگر راهی برای استفاده از مجموعه پیشین در زنجیره نخواهیم داشت و این نکته زنجیره ما را به خطر می‌اندازد. عبارت زیر را در نظر بگیرید:

```
$('img').clone().appendTo('#somewhere');
```

این مثال دو مجموعه ایجاد می‌کند و نخست مجموعه ای شامل تمام عناصر عکس صفحه ایجاد می‌شود و مجموعه دوم کپی مجموعه اول است که به انتهای عنصری با شناسه somewhere اضافه می‌شود. حال اگر بخواهیم پس از اعمال کپی بروی مجموعه اصلی عملی مانند افزودن یک کلاس را بروی آن انجام دهیم چه باید بکنیم؟ همچنین نمی‌توانیم مجموعه اصلی را به انتهای زنجیره انتقال دهیم، چون بروی قسمتی دیگر اثر خواهد گذاشت.

برای مرتفع کردن چنین نیازی، jQuery متد [end\(\)](#) را معرفی کرده است. زمانی از این متد استفاده می‌شود، یک نسخه پشتیبان از مجموعه کنونی ایجاد می‌شود. همان مجموعه برگردانده می‌شود. بنابراین اگر متدی پس از آن ظاهر شود اثرش بروی مجموعه اولیه خواهد بود. مثال زیر را در نظر بگیرید:

```
$('img').clone().appendTo('#somewhere').end().addClass('beenCloned');
```

این مثال دو مجموعه ایجاد می‌کند و نخست مجموعه ای شامل تمام عناصر عکس صفحه ایجاد می‌شود و مجموعه دوم کپی مجموعه اول است که به انتهای عنصری با شناسه somewhere اضافه می‌شود. اما با استفاده از متد [end\(\)](#) همان مجموعه اولیه در ادامه زنجیره قرار خواهد گرفت و سپس متد [addClass\(\)](#) بروی تمامی عناصر عکس اعمال می‌شود، نه تنها عکس‌های موجود در مجموعه اول، اگر از متد [end\(\)](#) استفاده نشود متد [addClass\(\)](#) بروی عناصر مجموعه دوم اعمال خواهد شد. قالب کلی متد [end\(\)](#) به شکل زیر است:

end()

در متدهای زنجیره ای استفاده می‌شود و از مجموعه کنونی یک پشتیبان می‌گیرد تا همان مجموعه در زنجیره جریان داشته باشد.

پارامتر

ندارد

خروجی

مجموعه عنصر قبلی

شاید در نظر گرفتن مجموعه‌ها در متدهای زنجیره ای به شکل یک پشته به درک بهتر از متد [end\(\)](#) کمک کند. هر زمان که یک مجموعه جدید در زنجیره ایجاد می‌شود، آن مجموعه به بالای پشته افزوده می‌شود، اما با فراخوانی متد [end\(\)](#) ، بالاترین مجموعه از این پشته برداشته می‌شود و مجدداً مجموعه پیشین در زنجیره قرار می‌گیرد. متد دیگری که توانایی ایجاد تغییر در این پشته خیالی را دارد، متد [andSelf\(\)](#) می‌باشد. این متد دو مجموعه بالای پشته را با یکدیگر ادغام می‌کند و آن‌ها را به یک مجموعه تبدیل می‌کند. شکل کلی متد [andSelf\(\)](#) به صورت زیر است:

andSelf()

دو مجموعه پیشین در یک زنجیره را با یکدیگر ادغام می‌کند.

پارامتر

ندارد

خروجی

مجموعه عنصری ادغام شده

در مباحث بعدی کار با **ویژگی‌های عناصر** بحث خواهد شد.

موفق و موید باشید

نظرات خوانندگان

نویسنده:

منصور جعفری

تاریخ:

۱۶:۲۲ ۱۳۹۳/۰۱/۰۳

سلام

مثلا در مورد طراحی یک سایت که اطلاعاتی بصورت تکراری پشت سر هم تکرار میشن (مثلا کامنت‌های که برای یک موضوع ارسال میشن) چطور باید باید اطلاعات مثلا مربوط به یک فیلد رو دستکاری انجام بدیم برای مثال

```
@foreach(var item in Model)
{
    <td class="text-right itemfarsi">@item.Farsi</td>
}
```

چطور میشه مثلا همین تیبل دیتا رو برای هر کامنت باتوجه به متن اون تغییر داد
من با استفاده از کدهای زیر دستور خودم رو انجام میدم اما در مورد تمام مطالب فقط اطلاعات مربوط به قسمت اول رو برمیگردونه.

```
$(document).ready(function () {
    var content = $(".itemfarsi").text();
    if (content.length >= 50) {
        var mycont = content.substring(0, 50);
        $(".itemfarsi").html(mycont);
    } else {
        $(".itemfarsi").html(content);
    }
});
```

نویسنده:

وحید نصیری

تاریخ:

۱۷:۲ ۱۳۹۳/۰۱/۰۳

از [متد each](#) می‌شود استفاده کرد.