عنوان: **آموزش (jQuery) جی کوئری #1**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۲۲ ۲۳:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, jQuery, Web Design

با سلام خدمت دوستان عزیز
تصمیم گرفتم در طی چندین پست در حد توانم به آموزش jQuery بپردازم. (مطالب نوشته شده برداشت ازادی از کتاب jQuery in action است)

جی کوئری (jQuery) چیست؟

jQuery یک کتابخانه بسیار مفید برای جاوا اسکریپت است. بسیار ساده و کارآمد است و مشکل جاوا اسکریپت را برای تطابق با مرورگرهای اینترنتی مختلف برطرف نموده است؛ یادگیریjQuery بسیار آسان است. در جی کوئری کد جاوا اسکریپت از فایل HTML جدا شده و بنابراین کنترل کدا و بینه‌سازی آنا بسیار ساده‌تر خواد شد. توابعی برای کار با AJAX فرام نموده و در این زمینه نیز کار را بسیار ساده کرده است. در جی کوئری می‌توان از خصوصیت فراخوانی زنجیره‌ای متدا استفاده نمود و این باعث می‌شود چندین کد فقط در یک سطر قرار گیرد و در نتیجه کد بسیار مختصر گردد. در مقایسه با سایر ابزارهایی که تاکید عمده‌ای بروی تکنیک‌های هوشمند جاوا اسکریپت دارند، هدف جی کوئری تغییر تفکر سازندگان وب سایت‌ها، به ایجاد صفحه‌هایی با کارکرد بالا می‌باشد. به جای صرف زمان برای مقابله با پیچیدگی‌های جاوا اسکریپت پیشرفته، طراحان می‌توانند با استفاده از زمان و دانش خود در زمینه‌ی CSS، HTML، XHTML و جاوا اسکریپت‌های ساده، عناصر صفحه را مستقیما دستکاری کنند و از همین طریق تغییرهای گشترده و سریعی انجام دهند.

نکته : برای استفاده از جی کوئری باید HTML و CSS و جاوا اسکریپت آشنایی داشته باشید.

چگونه از جی کوئری استفاده کنیم؟

برای استفاده از جی کوئری باید ابتدا فایل آن را از سایت آن دانلود کرده و در پروژه خود استفاده نمایید. البته روش‌های دیگری برای استفاده از این فایل وجود دارد که در آینده بیشتر با آن آشنا خواهیم شد. برای استفاده از این فایل در پروژه باید به شکل زیر آن را به صفحه HTML خود معرفی کنیم.

```
<html>
   <head>
       <script type="text/javascript" src="jquery-1.9.1.min.js"></script>
   </head>
   <body>
   </body>
</html>
```

سپس بعد از معرفی خط فوق در قسمت head صفحه باید کدهای خود را در یک تگ script بنویسیم.

کوتاه کردن کد : هر زمان شما خواسته باشید کارکرد یک صفحه وب را پویاتر کنید، در اکثر مواقع به ناچار این کار از طریق عناصری بروی صفحه انجام داده اید که با توجه به انتخاب شدن آنها، صفحه کارکردی خاص خواهد داشت. مثلا در جاوا اسکریپت اگر بخواهیم عنصری را که در یک radioGroup انتخاب شده است را برگردانیم باید کدهای زیر را بنویسیم:

```
var checkedValue;
var elements = document.getElementByTagName ('input');
for (var n = 0; n < elements.length; n++) {
       if (elements[n].type == 'radio' && elements[n].name == 'myRadioGroup' && elements[n].checked) {
              checkedValue = elements[n].value;
       }
}
```

اما اگر بخواهیم همین کد را با جی کوئری بنویسیم:

```
var checkedValue = $ ('[name="myRadioGroup"]:checked').val();
```

ممکن است مثال بالا کمی گنگ باشد نگران نباشید در آینده با این دستورات بیشتر آشنا خواهیم شد.
قدرت اصلی جی کوئری برگفته از انتخاب‌کننده‌ها (Selector) هاست، انتخاب‌کننده ، یک عبارت است که دسترسی به عنصری خاص بر روی صفحه را موجب می‌شود؛ انتخاب‌کننده این امکان را فراهم می‌سازد تا به سادگی عنصر مورد نظر را مشخص و به آن دسترسی پیدا کنیم که در مثال فوق، عنصر مورد نظر ما گزینه انتخاب شده از myRadioGroup بود.
Unobtrusive JavaScript : اگر پیش از پیدایش CSS در کار ایجاد صفحه‌های اینترنتی بوده‌اید حتما مشکلات و مشقات آن دوران را به خاطر می‌آورید. در آن زمان برای فرمت‌دهی به اجزای مختلف صفحه ، به ناچار علائم فرمت‌دهی را به همراه دستورات خود اجزا، در صفحه‌های HTML استفاده می‌کردیم. اکنون بسیار بعید به نظر می‌رسد کسی ترجیح دهد فرمت‌دهی اجزا را به همراه دستورهای HTML آن انجام دهد. اگر چه هنوز دستوری مانند زیر بسیار عادی به نظر می‌آید:

```
<button type="button" onclick="document.getElementById('xyz').style.color='red';">
        Click Me
</button>
```

نکته ای که در مثال فوق حائز اهمیت است، این است که خصوصیات ظاهری دکمه ایجاد شده از قبیل فونت و عنوان دکمه، از طریق تگ <font> و یا پارامترهای قابل استفاده در خود دستور دکمه تعیین نشده است، بلکه CSS وظیفه تعیین آنها را دارد. اما اگرچه در این مثال فرمت‌دهی و دستور خود دکمه از یکدیگر جدا شده‌اند؛ شاهد ترکیب این دکمه با رفتار آن هستیم. در جی کوئری می‌توانیم رفتار را از اجزا به آسانی جدا کنیم.

مجموعه عناصر در جی کوئری :

زمانی که CSS به عنوان یک تکنولوژی به منظور جداسازی طراحی از ساختار به دنیای صفحه‌های اینترنتی معرفی شد، می‌بایست راهی برای اشاره به اجزای صفحات از طرف فایل CSS نیز معرفی می‌شد. این امر از طریق **انتخاب‌کننده‌ها (Selector)** صورت پذیرفت.

برای مثال انتخاب‌کننده زیر، به تمام عناصر <a> اشاره دارد که در یک عنصر <p> قرار گرفته‌اند:

```
p a
```

جی کوئری نیز از چنین انتخاب‌کننده‌هایی استفاده می‌کند، البته نه تنها از انتخاب‌کننده‌هایی که هم اکنون در CSS موجود می‌باشند، بلکه برخی از انتخاب‌کننده‌هایی که هنوز در تمام مرورگرها پشتیبانی نمی‌شوند.
برای انتخاب مجموعه‌ای از عناصر از یکی از دو Syntax زیر استفاده می‌کنیم.

```
$(Selector)
یا
jQuery(Selector)
```

ممکن است در ابتدا ()$ کمی نا معمول به نظر آید، اما اکثر کسانی که با جی کوئری کار می‌کنند از اختصار و کوتاهی این ساختار استفاده می‌کنند.
مثال زیر نمونه‌ای دیگر است که در آن مجموعه‌ای از تمام لینک‌هایی که درون تگ <p> قرار دارند را انتخاب می‌کند:

```
$("p a")
```

تابع ()$ که در حقیقت نام خلاصه‌ای برای ()jQuery می‌باشد، نوع خروجی مخصوصی دارد که شامل یک آرایه از اشیایی می‌شود که انتخاب‌کننده آن را برگزیده است. این نوع خروجی این مزیت را دارد که شمار زیادی متد از پیش تعریف شده را داراست که به سادگی قابل اعمال می‌باشند.
در اصطلاح برنامه نویسی به چنین توابعی که گروهی از عناصر را جمع می‌کنند، Wrapper می‌گویند زیرا تمام عناصر مطلوب را تحت

یک شی بسته‌بندی می‌کند. در جی‌کوئری به آنها Wrapped Set یا jQuery Wrapper می‌گویند و به متدهایی که قابل اعمال بروی اینها به نام jQuery Wrapper Methodes شناخته می‌شوند.
در مثال زیر می‌خواهیم تمام عناصر در صورتی که دارای کلاس notLongForThisWorldباشند را مخفی (با فید شدن) کنیم.

```
$("div.notLongForThisWorld").fadeOut();
```

یکی از مزیت‌های اکثر متدهای قابل اجرا بروی مجموعه عناصر انتخاب شده آن است که خروجی خود آنها مجموعه‌ای دیگر است. به این معنا که خروجی این متد، آماده اعمال یک متد دیگر است.
فرض کنید در مثال بالا بخواهیم پس از مخفی کردن هر بخواهیم یک کلاس به نام removedبه آن بیافزاییم. به این منظور می‌توان کدی مانند زیر نوشت:

```
$("div.notLongForThisWorld").fadeOut().addClass("removed");
```

این زنجیره متدها می‌توانند به هرتعداد ادامه پیدا کند.

چند نمونه انتخاب کننده:

نتیجه		انتخاب کننده
تمام <p>های زوج را انتخاب می‌کند		`$('p:even')`
سطر اول هر جدول را انتخاب می‌کند		`$("tr:nth-child(1)");`
هایی که مستقیما در <body> تعریف شده باشند را انتخاب می‌کند.		`$("body > div");`
لینک هایی که به یک فایل pdf اشاره دارند را انتخاب می‌کند.		`$("a[href$=pdf]");`
تمام هایی که مستقیما در <body> معرفی شده اند و دارای لینک می‌باشند را انتخاب می‌کند.		`$("body > div:has(a)")`

ادامه مطالب در پست‌های بعدی تشریح خواهد شد.

جهت مطالعه بیشتر می‌توانید از این منابع ^ و ^ و ^ و ^ و ^ استفاده کنید.
موفق و موید باشید

نظرات خوانندگان

نویسنده: امیر
تاریخ: ۱۳۹۱/۱۱/۲۳ ۱۱:۲۶

مرسی .اقا خوب ادامه بده

نویسنده: آرآر
تاریخ: ۱۳۹۱/۱۲/۰۱ ۱۳:۳

ممنون. حتما ادامه بدید

عنوان: **آموزش (jQuery) جی کوئری #2**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۰۶ ۱۲:۱۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, jQuery, Web Design, Learning jQuery

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 پراستفاده‌ترین 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 دقیقا 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 JavaScriptاستفاده می‌کنیم، رفتار از ساختار جدا می‌شود، بنابراین برای انجام عملیات روی عناصر صفحه باید منتظر بمانیم تا انها 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 ، دستورها 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. متاسفانه، 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 JavaScriptبرای 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 **()$**، ما مرورگر را مجبور می‌کنیم که برای اجرای کد تا زمانی که DOM کامل لود شود (فقط DOM لود شود) منتظر بماند. حتی بهتر از آن ما می‌توانیم از این تکنیک چندین با در همان سند HTML استفاده کرده و مرورگر تمامی تابع‌های مشخص شده توسط ما را به ترتیب اجرا خواهد کرد. (یعنی من در دیک صفحه می‌توانم چنین بار تابع **()ready** را فراخوانی کنم). در مقابل روش **OnLoad** پنجره فقط اجازه اجرای یکبار تابع را به ما می‌دهد.
این هم یکی دیگر از کارکردهای دیگر تابع **()$** می‌باشد. حال به یکی دیگر از امکاناتی که این تابع برای ما فراهم می‌کند دقت کنید.

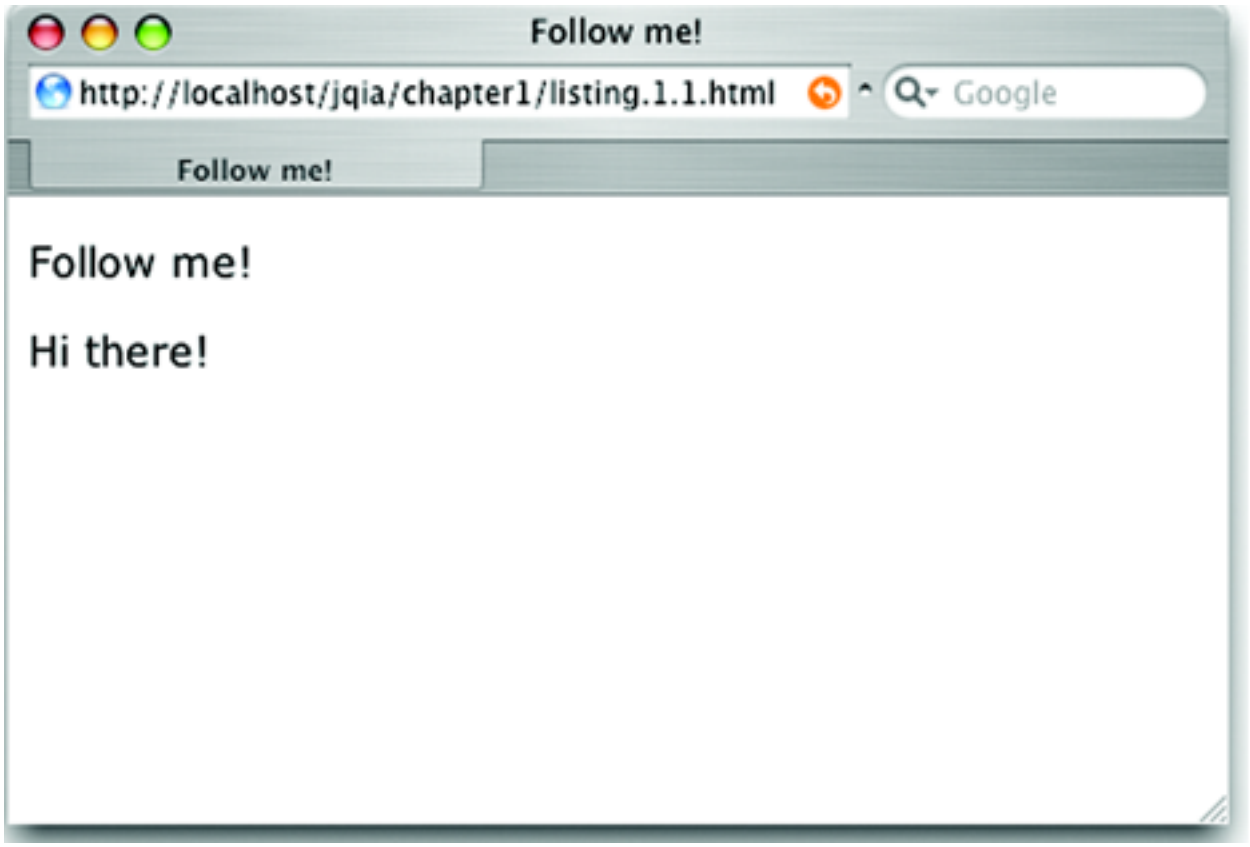
ساختن اجزای DOM (ساختن عناصر صفحه)
یکی دیگر از کارهایی که تابع **()$** می تواند برای ما انجام دهد ایجاد کردن عناصر صفحه است. به این منظور ورودی تابع **()$** را یک رشته که حاوی دستور HTML مربوط به ساخت یک عنصر می‌باشد، قرار می‌دهیم. برای مثال دستور زیر یک تگ p ایجاد می‌کند:

```
$("<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">
           // در زمان Reday می‌شود ایجاد مورد نظر صفحه عنصر بودن
           $(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 اشیای صفحه.

موفق ومومید باشید

عنوان: **آموزش (jQuery) جی کوئری #3**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۰۸ ۱۲:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, jQuery, Web Design, Learning jQuery

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

انتخاب عناصر صفحه

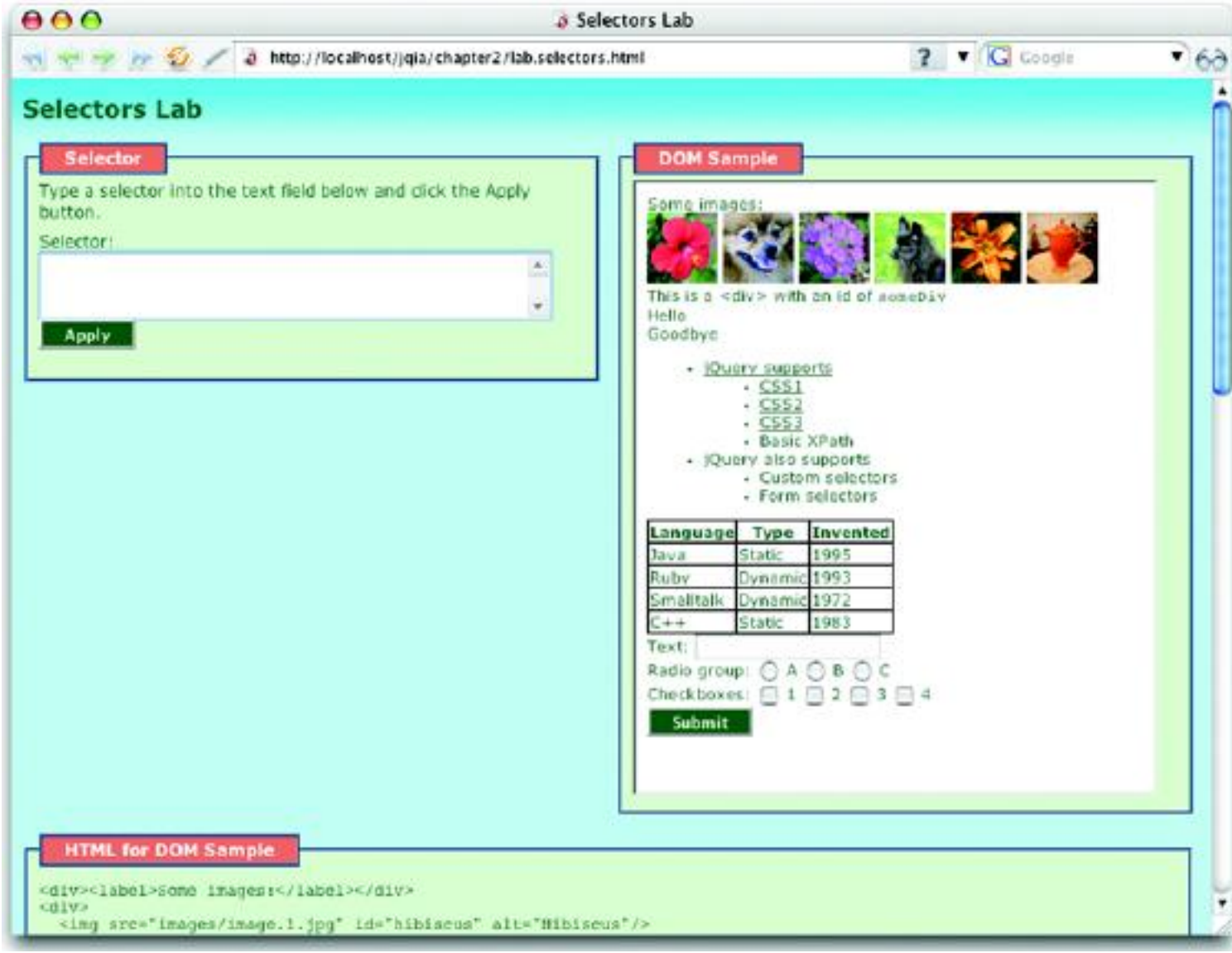
در پستهای قبل (^ و ^) با بسیاری از توانایی‌ها و کارکردهای jQuery شامل توانایی‌های آن برای انتخاب عناصر موجود در صفحه تا تعریف توابع جدید و استفاده از آنها به محض آماده شدن صفحه آشنا شدیم.

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.

-1انتخاب 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 ارایه 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 کننده ها، یک مثال آماده مختص به این مبحث، در قالب یک صفحه اینترنتی، را در فایل صفحه کارگاهی قرار داده ایم، این فایل در ادرس chapter2/lab.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.

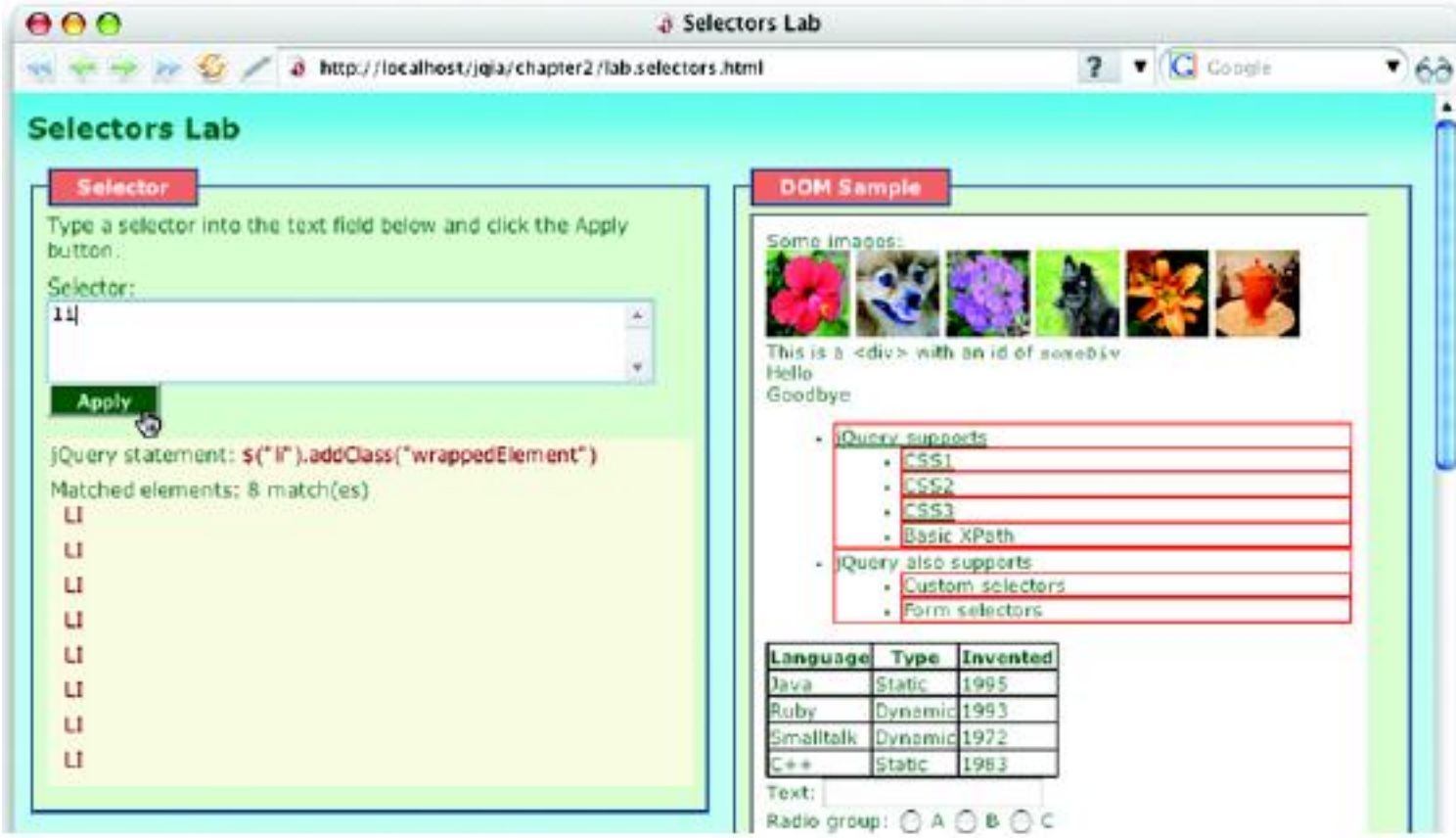


Figure 2.2 A selector value of `li` matches all `<li>` elements when applied as shown by the display results.

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 ایک 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 : عناصری 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.

a.specialClass p: 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 کاملا 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 استانداردی برای CSS نداشته باشد، انتخاب کننده jQuery به مشکل بر نمی‌خورد، بلکه jQuery انتخاب خود را به درستی انجام می‌دهد، چرا که jQuery از قوانین استاندارد W3C تبعیت می‌کند.

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

برای انتخاب کننده‌های پیشرفته تر، jQuery از جدیدترین مرورگرهایی که CSS را پشتیبانی می‌کنند، استفاده می‌کند که می‌توان به Mozilla Firefox, Internet Explorer 7, Safariو سایر مرورگرهای پیشرفته (مدرن) اشاره کرد. این انتخاب کننده‌های پیشرفته شما را قادر می‌سازند تا مستقیما فرزند یک عنصر را انتخاب کنید و یا از ساختار سلسله مراتبی عناصر صفحه، مستقیما به عنصر مورد نظر دسترسی داشته باشید و یا حتی تمام عناصری که یک صفت خاص را شامل می‌شوند، 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 لینک‌های مربوط به CSS مطلوب است. اگر بخواهیم از دستورهای انتخاب کننده CSS استفاده کینم، دستوری به شکل ul.myList li a خواهیم داشت. اما متاسفانه این دستور تمام لینک‌های این ساختار را انتخاب میکند، زیرا همه آنها لینک هایی در عنصر li می‌باشند. با نوشتن این دستور در صفحه کارگاهی خروجی به شکل زیر خواهد بود:

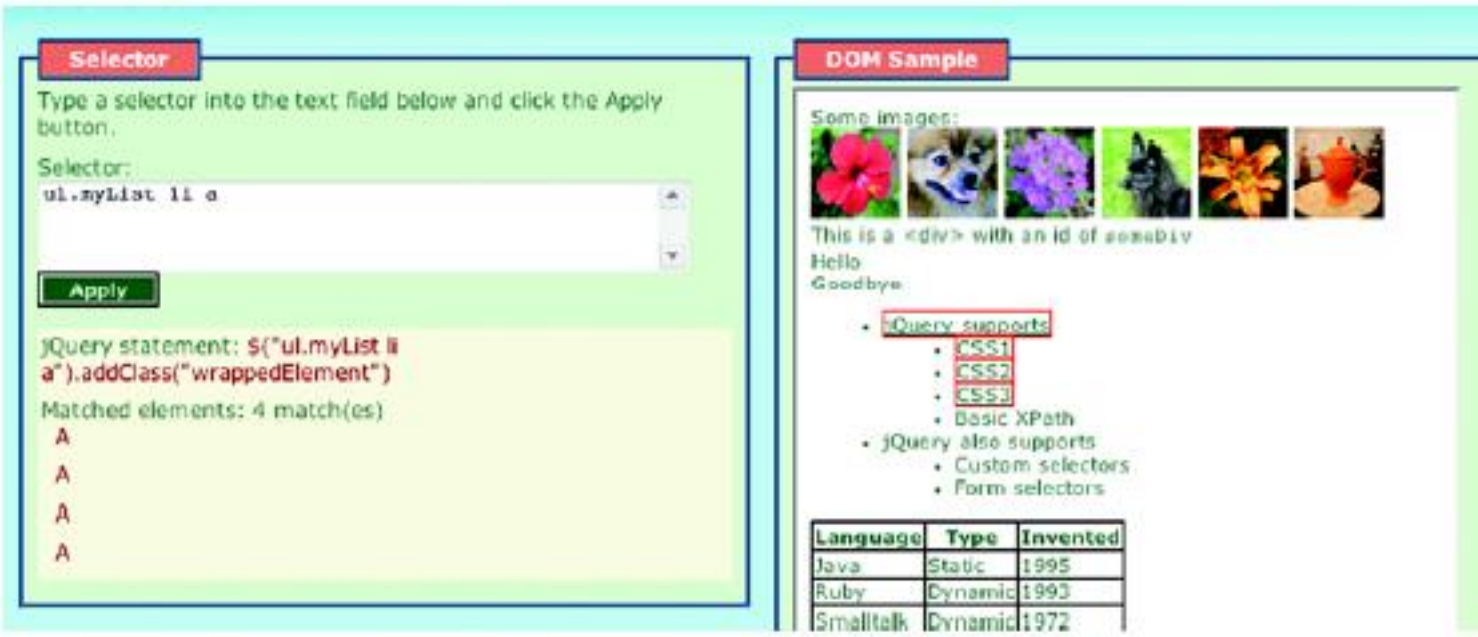


Figure 2.3 All anchor tags that are descendents, at any depth, of an <li> element are selected
ul.myList li a

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 می‌شوند:

```
p > a
```

این دستور تنها لینک (<a>) هایی را بر می‌گرداند که فرزند مستقیم یک عنصر <p> می‌باشند. بنابراین اگر در یک <p> لینکی در عنصر معرفی شده باشد، این لینک انتخاب نمی‌شود، چرا که فرزند مستقیم <p> به حساب نمی‌آید. در مورد مثال لینک‌های موجود در لیست، می‌توانیم 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 می‌باشد، انتخاب می‌کند و پس از آن لینکهایی (<a>) که فرزند مستقیم گزینه‌های آن هستند، برگردانده می‌شوند. همانگونه که در شکل زیر مشاهده می‌کنید لینک‌های زیرمجموعه عنصر <ul> انتخاب نمی‌شوند، زیرا فرزند مستقیم این عنصر محصوب نمی‌شوند.

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 تواناتر 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 لینک هایی 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 لینک هایی که مقدار صفت href آنها **دقیقا** با http:// آغاز می‌شود، می‌گردد. علامت ^ موجب می‌شود تابررسی، لزوما 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.

می توانید این کد را در صفحه کار گاهی 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 با شد 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 اغاز 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 لنک‌های 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 هرجای دیگر.

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

همانگونه که انتظار میرود این انتخاب کننده ، تمام لینک هایی که به وب سایت jQuery اشاره دارند را برمی گرداند.
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 li هایی را برمی گرداند که شامل لینک (<a>) هستند. دقت کنید که این انتخاب گر مانند li a نیست، انتخاب گر دوم تمامی لینک هایی را که در li هستند بر میگرداند اما دستور بالا li هایی 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.

عنوان: **آموزش (jQuery) جی کوئری #4**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۲۷ ۱۱:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, jQuery, Web Design, Learning jQuery

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

با توجه به حالت‌های مختلف و گزینه‌های گوناگونی که انتخاب کننده‌ها در اختیار ما گذاشته اند، اگر هنوز دنبال قدرت بیشتری از انتخاب کننده‌ها 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 شیوه ای خاص را برای چنین انتخاب هایی ارایه 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؛ مثلا 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 <lii> فرزند یک <ul> انتخاب می‌شود.
علاوه بر انتخاب کننده هایی که ذکر شد؛ تعداد قابل توجه دیگری نیز وجود دارند که در جدول 2-2 ذکر شده اند.

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

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

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

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

نکته ای که در مورد انتخاب گرهای جدول بالا وجود دارد این است که در فیلتر nth-child: برای سازگاری با CSS، مقدار شمارشگر از 1 آغاز می‌شود، اما در سایر فیلترها از قاعده ای که اکثر زبانهای برنامه نویسی استفاده شده است و شمارشگر آنها از صفر اغاز می‌شود. 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 خانه هایی که نام یک زبان برنامه نویسی در آنهاست را انتخاب نماییم. از انجا 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 comment // **td:eq(2)** خانه ای با مقدار 1995 را انتخاب می‌کند، دستور comment // **td:nth-child(2)** 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 گرهای CSS و فیلترهای سفارشی jQuery را بررسی خواهیم کرد.

عنوان: **آموزش (jQuery) جی کوئری #5**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۱/۲۳ ۱۸:۲۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Web Design, Learning jQuery

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.

4-1- 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 باکس ها، گزینه هایی 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 برمی گرداند. براس مثال انتخاب کننده input تمامی المان‌های <input> را انتخاب می‌کند، و انتخاب کننده input:checked تمامی inputهایی 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 کننده هایی مانند :radio:checked و :checkbox:checked .
همانطور هم که قبلا بیان شد، jQuery علاوه بر پشتیبانی از انتخاب کننده‌های CSS تعدادی انتخاب کننده سفارشی را نیز شامل می‌شود که در جدول 3-2 شرح داده شده است.

جدول 3-2: 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 input[type=submit]، input[type=reset]،) (input[type=button]، یا button)
checkbox:	عناصر Checkbox را انتخاب می‌کند، مانند (input[type=checkbox]).
checked:	عناصر checkboxها یا دکمه‌های رادیویی را انتخاب می‌کند که در حالت انتخاب باشند.
contains(foo) :	عناصری ر انتخاب می‌کند که دارای عبارت foo باشند.
disabled:	عناصر در حالت disabled را انتخاب می‌کند.
enabled:	عناصر در حالت enabledرا انتخاب می‌کند.
file:	عناصر فایل را انتخاب می‌کند، مانند (input[type=file]).
header:	عناصر هدر مانند h1 تا h6 را انتخاب می‌کند.
hidden:	عناصر مخفی شده را انتهاب می‌کند.
image:	عناصر تصویر را انتخاب می‌کند، مانند (input[type=image]).

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]).
raset:	دکمه‌های reset را انتخاب می‌کند، مانند (input[type=reset] یا button[type=reset]).
selected:	عناصری (عناصر option) را انتخاب می‌کند که در وضعیت selected قراردارند.
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 visibleباشند 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 ارایه کرده است که فرزندران 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: ۱۳۹۲/۰۱/۲۸ ۱۳:۴۱

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

نویسنده: رها
تاریخ: ۱۳۹۲/۱۰/۰۷ ۲۰:۳۰

سلام؛ من یه اسلاید شو ساده را از اموزشهای یه سایت انگلیسی زبان ساختم که مدتهاست دنبالش بودم
اما هنوز یه ایراد کوچولو داره و اونهم اینه که بعد از رسیدن به اخرین عکس برمیگرده به اول یعنی بصورت بک اسلاید میشه و اگر عکسها از سمت راست به چپ اسلاید میشوند وقتی به اخرین 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 اخرین 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 ویژوآل استادیو باز میکنم بصورتی نوشته شده که تمام فایل در یک خط افقی هست و اگر بخواهیم ویرایشش کنیم نمیشه با اسکرول کردن موس در امتدادش حرکت کنیم. میخواستم ببینم آیا در ویژوآل 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

عنوان: **آموزش (jQuery) جی کوئری #6**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۴/۰۲ ۲۱:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, jQuery, Web Design, Learning jQuery

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 ساده ای 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 حرفه ای‌تری 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 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 محتویاتش 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.htmlرا 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 ان 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 تقریبا مبحث جدایی است؛ در پست بعدی به بررسی توابع و متدهای **مدیریت مجموعه انتخاب شده** خواهیم پرداخت.

عنوان: **آموزش (jQuery) جی کوئری #7**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۴/۰۴ ۲۱:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Web Design, Learning jQuery

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

۲-۳- مدیریت مجموعه انتخاب شده

هز زمان که مجموعه ای از عناصر انتخاب 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 بهبود 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 اینصورت این صفحه کاربردی نخواهد داشت.

۲-۳-۱-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 آراه در جاوااسکریپت ، تعداد عناصر موجود در مجموعه را شامل می‌شود.
افزون بر این ویژگی، jQueryیک 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 بااستفاده 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؟

۲-۳-۲-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 ارایه 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 واکشی 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 ارایه از عناصر پیچیده نیز استفاده شود. مثلا:

```
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 برمی گرداند، 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 ان 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

۲-۳-۳-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 کلکسیون 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 ارایه 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 ستفاده از متد not() در توابع زنجیره ای انجام داد. این متد عملکرد 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، ارایه 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 مجموعه هایی که انتخاب کننده‌ها 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 انتخاب کننده‌های jQuery به ما ارایه می‌دهند، انجام چنین کاری با استفاده از انتخاب کننده‌ها 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 زنجیره ای از ارقام بود، خروجی تابع true خواهد بود، و ان 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 تعیین کننده عناصر در نظر گرفته شده برای حذف می‌باشد. این پارامتر می‌تواند یک عنصر، ارایه 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 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 مجموعه هایی 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) کاملا 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.

۲-۳-٤-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 ۲-٤-متدهای موجود برای ایجاد مجموعه‌های جدید بر اساس روابط

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

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

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

۲-۳-۵-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:

```
wrappedSet.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. مثلا

```
$('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 می‌کندو 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 شودف اثرش بروی مجموعه اولیه خواهد بود. مثال زیر را در نظر بگیرید:

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

این مثال دو مجموعه ایجاد می‌کندو 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 مجدادا 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 صفت‌ها و ویژگی‌های عناصر** بحث خواهد شد.

موفق و موید باشید

نظرات خوانندگان

نویسنده: منصور جعفری
تاریخ: ۱۳۹۳/۰۱/۰۳ ۱۶:۲۲

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: ۱۳۹۳/۰۱/۰۳ ۱۷:۲

از متد each می‌شود استفاده کرد.

عنوان: **شروع کار با Dart - قسمت 1**
نویسنده: میثم خوشبخت
تاریخ: ۱۳۹۳/۰۲/۰۲ ۱۴:۵۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Web Design, Dart

Dart کتابخانه ای است که توسط شرکت گوگل ارائه شده است و گفته می‌شود، قرار است جایگزین جاوا اسکریپت گردد و از آدرس https://www.dartlang.org قابل دسترسی می‌باشد. این کتابخانه، دارای انعطاف پذیری فوق العاده بالایی است و کد نویسی Java Script را راحت‌تر می‌کند. در حال حاضر هیچ مرورگری به غیر از Chromium از این تکنولوژی پشتیبانی نمی‌کند و جهت تسهیل در کدنویسی، باید از ویرایشگر Dart Editor استفاده کنید. این ویرایشگر کدهای نوشته شده را به دو صورت Native و JavaScript Compiled در اختیار مرورگر قرار می‌دهد. در ادامه با نحوه‌ی کار و راه اندازی Dart آشنا خواهید شد.

ابتدا Dart و ویرایشگر مربوط به آن را توسط لینک‌های زیر دانلود کنید:

دانلود
نسخه 64 بیتی دارت + ویرایشگر

دانلود
نسخه 32 بیتی دارت + ویرایشگر

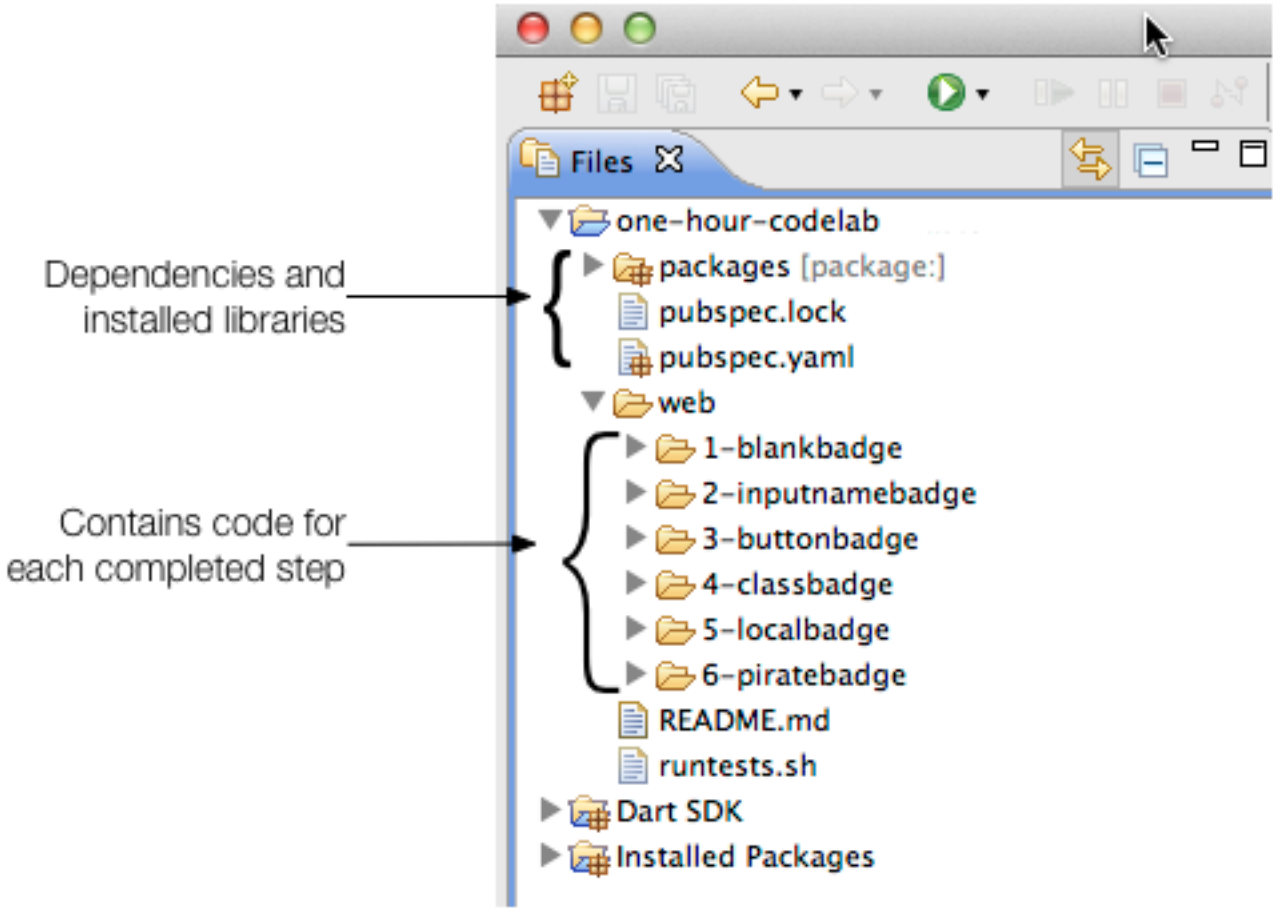
بعد از اینکه فایلهای فوق را از حالت فشرده خارج کردید، پوشه ای با نام dart ایجاد می‌نماید. وارد پوشه dart شده و DartEditor را اجرا کنید.

توجه: جهت اجرای dart به JDK 6.0 یا بالاتر نیاز دارید

در مرحله بعد نمونه کدهای Dart را از لینک زیر دانلود نمایید و از حالت فشرده خارج کنید. پوشه ای با نام one-hour-codelab ایجاد می‌گردد.

دانلود
نمونه کدهای دارت

از منوی File > Open Existing Folder... پوشه one-hour-codelab را باز کنید .



توضیحات

- پوشه packages و همچنین فایلهای pubspec.yaml و pubspec.lock شامل پیش نیازها و Package هایی هستند که جهت اجرای برنامه‌های تحت Dart مورد نیاز هستند. Dart Editor این نیازمندی‌ها را به صورت خودکار نصب و تنظیم می‌کند.

توجه: اگر پوشه Packages را مشاهده نکردید و یا در سمت چپ فایلها علامت X قرمز رنگ وجود داشت، بدین معنی است که package ها به درستی نصب نشده اند. برای این منظور بر روی pubspec.yaml کلیک راست نموده و گزینه Get Pub را انتخاب کنید. توجه داشته باید که بدلیل تحریم ایران توسط گوگل باید از ابزارهای عبور از تحریم استفاده کنید.

- 6 پوشه را نیز در تصویر فوق مشاهده می‌کنید که نمونه کد piratebadge را بصورت مرحله به مرحله انجام داده و به پایان می‌رساند.

- Dart SDK شامل سورس کد مربوط به تمامی توابع، متغیرها و کلاس هایی است که توسط کیت توسعه نرم افزاری Dart ارائه شده است.

- Installed Packages شامل سورس کد مربوط به تمامی توابع، متغیرها و کلاس‌های کتابخانه‌های اضافه‌تری است که Application به آنها وابسته است.

گام اول: اجرای یک برنامه کوچک

در این مرحله سورس کدهای آماده را مشاهده می‌کنید و با ساختار کدهای Dart و HTML آشنا می‌شوید و برنامه کوچکی را اجرا

می‌نمایید.

در Dart Editor پوشه blankbadge-1 را باز کنید و فایلهای piratebadge.html و piratebadge.dart را مشاهده نمایید.

کد موجود در فایل piratebadge.html

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Pirate badge</title>
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="piratebadge.css">
  </head>
  <body>
    <h1>Pirate badge</h1>

    <div>
      TO DO: Put the UI widgets here.
    </div>
    <div>
      <div>
        Arrr! Me name is
      </div>
      <div>
        <span id="badgeName"> </span>
      </div>
    </div>

    <script type="application/dart" src="piratebadge.dart"></script>
    <script src="packages/browser/dart.js"></script>
  </body>
</html>
```

توضیحات

- در کد HTML ، اولین تگ <script> ، فایل piratebadge.dart را جهت پیاده سازی دستورات dart به صفحه ضمیمه می‌نماید

- Dart VM (Dart Virtual Machine) کدهای Dart را بصورت Native یا بومی ماشین اجرا می‌کند. Dart VM کدهای خود را در Dartium که یک ویرایش ویژه از مرورگر Chromium می‌باشد اجرا می‌کند که می‌تواند برنامه‌های تحت Dart را بصورت Native اجرا کند.

- فایل packages/browser/dart.js پشتیبانی مرورگر از کد Native دارت را بررسی می‌کند و در صورت پشتیبانی، Dart VM را راه اندازی می‌کند و در غیر این صورت JavaScript کامپایل شده را بارگزاری می‌نماید.

کد موجود در piratebadge.dart

```
void main() {
  // Your app starts here.
}
```

- این فایل شامل تابع main می‌باشد که تنها نقطه ورود به application است. تگ <script> موجود در piratebadge.html برنامه را با فراخوانی این تابع راه اندازی می‌کند.

- تابع main() یک تابع سطح بالا یا top-level می‌باشد.

- متغیرها و توابع top-level عناصری هستند که خارج از ساختار تعریف کلاس ایجاد می‌شوند.

جهت اجرای برنامه در Dart Editor بر روی piratebadge.html کلیک راست نمایید و گزینه Run in Dartium را اجرا کنید. این فایل توسط Dartium اجرا می‌شود و تابع main() را فراخوانی می‌کند و صفحه ای همانند شکل زیر را نمایش می‌دهد.

گام دوم: افزودن فیلد input

توجه داشته باشید که در این مرحله یا می‌توانید تغییرات مورد نظر خود را در طی آموزش بر روی پوشه‌ی blankbadge-1 اعمال کنید و یا به پوشه‌های تهیه شده در نمونه کد موجود در همین پروژه مراجعه نمایید.

در این مرحله یک تگ <input> به تگ <”div class=”widgets> اضافه کنید.

```
...
<div>
  <div>
    <input type="text" id="inputName" maxlength="15">
  </div>
</div>
...
```

سپس کتابخانه dart:html را به ابتدای فایل piratebadge.dart اضافه کنید.

```
import 'dart:html';
```

توضیحات

- دستور فوق کلاس‌ها و Resource های موجود در کتابخانه dart:html را اضافه می‌کند.

- از حجیم شدن کدهای خود نگران نباشید، زیرا فرایند کامپایل کدهای اضافی را حذف خواهد کرد.

- کتابخانه dart:html شامل کلاسهایی جهت کار با عناصر DOM و توابعی جهت دسترسی به این عناصر می‌باشد.

- در مباحث بعدی یاد می‌گیرید که با استفاده از کلمه کلیدی show فقط کلاسهایی را import کنید که به آن نیاز دارید.

- اگر کتابخانه ای در هیچ بخش کد استفاده نشود، خود Dart Editor به صورت warning اخطار می‌دهد و می‌توانید آن را حذف

کنید.

دستور زیر را در تابع main بنویسید تا رویداد مربوط به ورود اطلاعات در فیلد input را مدیریت نمایید.

```
void main() {
  querySelector('#inputName').onInput.listen(updateBadge);
}
```

توضیحات

- تابع querySelector() در کتابخانه dart:html تعریف شده است و یک المنت DOM را جستجو می‌نماید. پارامتر ورودی آن یک selector می‌باشد که در اینجا فیلد input را توسط #inputName جستجو نمودیم که یک ID Selector می‌باشد.

- نوع خروجی این متد یک شی از نوع DOM می‌باشد.

- تابع onInput.Listen() رویدادی را برای پاسخگویی به ورود اطلاعات در فیلد input تعریف می‌کند. زمانی که کاربر اطلاعاتی را وارد نماید، تابع updateBadge فراخوانی می‌گردد.

- رویداد input زمانی رخ می‌دهد که کاربر کلیدی را از صفحه کلید فشار دهد.

- رشته‌ها همانند جاوا اسکریپت می‌توانند در " یا ' قرار بگیرند.

تابع زیر را به صورت top-level یعنی خارج از تابع main تعریف کنید.

```
...

void updateBadge(Event e) {
  querySelector('#badgeName').text = e.target.value;
}
```

توضیحات

- این تابع محتوای المنت badgeName را به محتوای وارد شده در فیلد input تغییر می‌دهد.

- پارامتر ورودی این تابع شی e از نوع Event می‌باشد و به همین دلیل می‌توانیم این تابع را یک Event Handler بنامیم.

- e.target به شی ای اشاره می‌کند که موجب رخداد رویداد شده است و در اینجا همان فیلد input می‌باشد

- با نوشتن کد فوق یک warning را مشاهده می‌کنید که بیان می‌کند ممکن است خصوصیت value برای e.target وجود نداشته باشد. برای حل این مسئله کد را بصورت زیر تغییر دهید.

```
...

void updateBadge(Event e) {
  querySelector('#badgeName').text = (e.target as InputElement).value;
}
```

توضیحات

- کلمه کلیدی as به منظور تبدیل نوع استفاده می‌شود که e.target را به یک InputElement تبدیل می‌کند.

همانند گام اول برنامه را اجرا کنید و نتیجه را مشاهده نمایید. با تایپ کردن در فیلد input به صورت همزمان در کادر قرمز رنگ نیز نتیجه تایپ را مشاهده می‌نمایید.

عنوان: **شروع کار با Dart - قسمت 2**
نویسنده: میثم خوشبخت
تاریخ: ۱۳۹۳/۰۲/۰۲ ۱۸:۱۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Web Design, Dart

لطفا قسمت اول را در اینجا مطالعه بفرمائید

گام سوم: افزودن یک button

در این مرحله یک button را به صفحه html اضافه می‌کنیم. button زمانی فعال می‌شود که هیچ متنی در فیلد input موجود نباشد. زمانی که کاربر بر روی دکمه کلیک می‌کند نام Meysam Khoshbakht را در کادر قرمز رنگ می‌نویسد.

تگ <button> را بصورت زیر در زیر فیلد input ایجاد کنید

```
...
<div>
  <div>
    <input type="text" id="inputName" maxlength="15">
  </div>
  <div>
    <button id="generateButton">Aye! Gimme a name!</button>
  </div>
</div>
...
```

در زیر دستور import و بصورت top-level متغیر زیر را تعریف کنید تا یک ButtonElement در داخل آن قرار دهیم.

```
import 'dart:html';

ButtonElement genButton;
```

توضیحات

- ButtonElement یکی از انواع المنت‌های DOM می‌باشد که در کتابخانه dart:html قرار دارد
- اگر متغیری مقداردهی نشده باشد بصورت پیش فرض با null مقداردهی می‌گردد

به منظور مدیریت رویداد کلیک button کد زیر را به تابع main اضافه می‌کنیم

```
void main() {
  querySelector('#inputName').onInput.listen(updateBadge);
  genButton = querySelector('#generateButton');
  genButton.onClick.listen(generateBadge);
}
```

جهت تغییر محتوای کادر قرمز رنگ تابع top-level زیر را به piratebadge.dart اضافه می‌کنیم

```
...

void setBadgeName(String newName) {
  querySelector('#badgeName').text = newName;
}
```

جهت مدیریت رویداد کلیک button تابع زیر را بصورت top-level اضافه می‌کنیم

```
...

void generateBadge(Event e) {
  setBadgeName('Meysam Khoshbakht');
}
```

همانطور که در کدهای فوق مشاهده می‌کنید، با فشردن button تابع generateBadge فراخوانی میشود و این تابع نیز با فراخوانی تابع setBadgeName محتوای badge یا کادر قرمز رنگ را تغییر می‌دهد. همچنین می‌توانیم کد موجود در updateBadge مربوط به

رویداد input فیلد input را بصورت زیر تغییر دهیم

```
void updateBadge(Event e) {
  String inputName = (e.target as InputElement).value;
  setBadgeName(inputName);
}
```

جهت بررسی پر بودن فیلد input می‌توانیم از یک if-else بصورت زیر استفاده کنیم که با استفاده از توابع رشته ای پر بودن فیلد را بررسی می‌کند.

```
void updateBadge(Event e) {
  String inputName = (e.target as InputElement).value;
  setBadgeName(inputName);
  if (inputName.trim().isEmpty) {
    // To do: add some code here.
  } else {
    // To do: add some code here.
  }
}
```

توضیحات

- کلاس String شامل توابع و ویژگی‌های مفیدی برای کار با رشته‌ها می‌باشد. مثل trim که فواصل خالی ابتدا و انتهای رشته را حذف می‌کند و isEmpty که بررسی می‌کند رشته خالی است یا خیر.

- کلاس String در کتابخانه dart:core قرار دارد که بصورت خودکار در تمامی برنامه‌های دارت import می‌شود

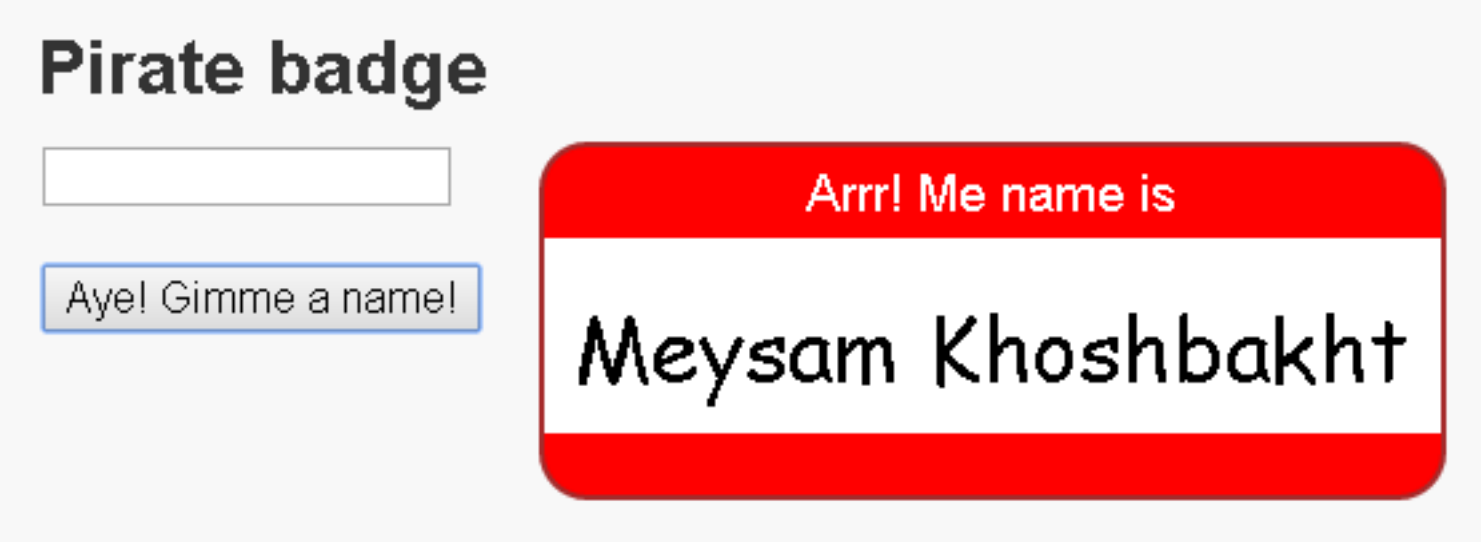
حال جهت مدیریت وضعیت فعال یا غیر فعال بودن button کد زیر را می‌نویسیم

```
void updateBadge(Event e) {
  String inputName = (e.target as InputElement).value;
  setBadgeName(inputName);
  if (inputName.trim().isEmpty) {
    genButton..disabled = false
             ..text = 'Aye! Gimme a name!';
  } else {
    genButton..disabled = true
             ..text = 'Arrr! Write yer name!';
  }
}
```

توضیحات

- عملگر cascade یا آبشاری (..)، به شما اجازه می‌دهد تا چندین عملیات را بر روی اعضای یک شی انجام دهیم. اگر به کد دقت کرده باشید با یک بار ذکر نام متغیر genButton ویژگی‌های disabled و text را مقدار دهی نمودیم که موجب تسریع و کاهش حجم کد نویسی می‌گردد.

همانند گام اول برنامه را اجرا کنید و نتیجه را مشاهده نمایید. با تایپ کردن در فیلد input و خالی کردن آن وضعیت button را بررسی کنید. همچنین با کلیک بر روی button نام درج شده در badge را مشاهده کنید.



گام چهارم: ایجاد کلاس PirateName

در این مرحله فقط کد مربوط به فایل dart را تغییر میدهیم. ابتدا کلاس PirateName را ایجاد می‌کنیم. با ایجاد نمونه ای از این کلاس، یک نام بصورت تصادفی انتخاب می‌شود و یا نامی بصورت اختیاری از طریق سازنده انتخاب می‌گردد.

نخست کتابخانه dart:math را به ابتدای فایل dart اضافه کنید

```
import 'dart:html';

import 'dart:math' show Random;
```

توضیحات

- با استفاده از کلمه کلیدی show، شما می‌توانید فقط کلاسها، توابع و یا ویژگی‌های مورد نیازتان را import کنید.

- کلاس Random یک عدد تصادفی را تولید می‌کند

در انتهای فایل کلاس زیر را تعریف کنید

```
...

class PirateName {
}
```

در داخل کلاس یک شی از کلاس Random ایجاد کنید

```
class PirateName {
  static final Random indexGen = new Random();
}
```

توضیحات

- با استفاده از static یک فیلد را در سطح کلاس تعریف می‌کنیم که بین تمامی نمونه‌های ایجاد شده از کلاس مشترک می‌باشد

- متغیرهای final فقط خواندنی می‌باشند و غیر قابل تغییر هستند.

- با استفاده از new می‌توانیم سازنده ای را فراخوانی نموده و نمونه ای را از کلاس ایجاد کنیم

دو فیلد دیگر از نوع String و با نام‌های _firstName و _appelation به کلاس اضافه می‌کنیم

```
class PirateName {
  static final Random indexGen = new Random();
  String _firstName;
  String _appellation;
}
```

متغیرهای خصوصی با (_) تعریف می‌شوند. Dart کلمه کلیدی private را ندارد.

دو لیست static به کلاس فوق اضافه می‌کنیم که شامل لیستی از name و appelation می‌باشد که می‌خواهیم آیتمی را بصورت تصادفی از آنها انتخاب کنیم.

```
class PirateName {
  ...
  static final List names = [
    'Anne', 'Mary', 'Jack', 'Morgan', 'Roger',
    'Bill', 'Ragnar', 'Ed', 'John', 'Jane' ];
  static final List appellations = [
    'Jackal', 'King', 'Red', 'Stalwart', 'Axe',
    'Young', 'Brave', 'Eager', 'Wily', 'Zesty'];
}
```

کلاس List می‌تواند شامل مجموعه ای از آیتم‌ها می‌باشد که در Dart تعریف شده است.

سازنده ای را بصورت زیر به کلاس اضافه می‌کنیم

```
class PirateName {
  ...
  PirateName({String firstName, String appellation}) {
    if (firstName == null) {
      _firstName = names[indexGen.nextInt(names.length)];
    } else {
      _firstName = firstName;
    }
    if (appellation == null) {
      _appellation = appellations[indexGen.nextInt(appellations.length)];
    } else {
      _appellation = appellation;
    }
  }
}
```

توضیحات

- سازنده تابعی همنام کلاس می‌باشد

- پارامترهایی که در {} تعریف می‌شوند اختیاری و Named Parameter می‌باشند. Named Parameterها پارمترهایی هستند که جهت مقداردهی به آنها در زمان فراخوانی، از نام آنها استفاده می‌شود.

- تابع nextInt() یک عدد صحیح تصادفی جدید را تولید می‌کند.

- جهت دسترسی به عناصر لیست از [] و شماره‌ی خانه‌ی لیست استفاده می‌کنیم.

- ویژگی length تعداد آیتم‌های موجود در لیست را بر می‌گرداند.

در این مرحله یک getter برای دسترسی به pirate name ایجاد می‌کنیم

```
class PirateName {
  ...
  String get pirateName =>
    _firstName.isEmpty ? '' : '$_firstName the $_appellation';
}
```

توضیحات

- Getterها متدهای خاصی جهت دسترسی به یک ویژگی به منظور خواندن مقدار آنها می‌باشند.

- عملگر سه گانه :? دستور میانبر عبارت شرطی if-else می‌باشد

- $ یک کاراکتر ویژه برای رشته‌های موجود در Dart می‌باشد و می‌تواند محتوای یک متغیر یا ویژگی را در رشته قرار دهد. در رشته '$_appellation_$firstName the' محتوای دو ویژگی _firstName و _appellation در رشته قرار گرفته و نمایش می‌یابند.

- عبارت (=> expr;) یک دستور میانبر برای { return expr; } می‌باشد.

تابع setBadgeName را بصورت زیر تغییر دهید تا یک پارامتر از نوع کلاس PirateName را به عنوان پارامتر ورودی دریافت نموده و با استفاده از Getter مربوط به ویژگی pirateName، مقدار آن را در badge name نمایش دهد.

```
void setBadgeName(PirateName newName) {
  querySelector('#badgeName').text = newName.pirateName;
}
```

تابع updateBadge را بصورت زیر تغییر دهید تا یک نمونه از کلاس PirateName را با توجه به مقدار ورودی کاربر در فیلد input تولید نموده و تابع setBadgeName رافراخوانی نماید. همانطور که در کد مشاهده می‌کنید پارامتر ورودی اختیاری firstName در زمان فراخوانی با ذکر نام پارامتر قبل از مقدار ارسالی نوشته شده است. این همان قابلیت Named Parameter می‌باشد.

```
void updateBadge(Event e) {
  String inputName = (e.target as InputElement).value;

  setBadgeName(new PirateName(firstName: inputName));
  ...
}
```

تابع generateBadge را بصورت زیر تغییر دهید تا به جای نام ثابت Meysam Khoshbakht، از کلاس PirateName به منظور ایجاد نام استفاده کند. همانطور که در کد می‌بینید، سازنده‌ی بدون پارامتر کلاس PirateName فراخوانی شده است.

```
void generateBadge(Event e) {
  setBadgeName(new PirateName());
}
```

همانند گام سوم برنامه را اجرا کنید و نتیجه را مشاهده نمایید.

نظرات خوانندگان

نویسنده: محمد 92
تاریخ: ۱۳۹۳/۰۲/۰۳ ۱۰:۲۲

سلام و ممنون از مطلب خوبتون، فقط امکانش هست لینک هایی برای بنچمارک دارت و جاوا اسکریپت بزارید تا ببینیم کدوم بهتر عمل می‌کنند و کدوم حجم کمتری برای دانلود نهایی دارند

نویسنده: میثم خوشبخت
تاریخ: ۱۳۹۳/۰۲/۰۳ ۱۸:۱۵

لینک بنچمارک Dart، dart2js و Java Script برای مقایسه Performance هریک از آنها که نشون میده Dart امتیاز بالاتری رو کسب کرده

عنوان: **شروع کار با Dart - قسمت 3**
نویسنده: میثم خوشبخت
تاریخ: ۱۳۹۳/۰۲/۰۳ ۹:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Web Design, Dart

لطفا قسمت دوم را در اینجا مطالعه بفرمایید

خدمت دوستان عزیز مطلبی را عرض کنم که البته باید در ابتدای این سری مقالات متذکر می‌شدم. این سری مقالات Dart مرجع کاملی برای یادگیری Dart نمی‌باشد. فقط یک Quick Start یا Get Started محسوب می‌شود برای آشنایی مقدماتی با ساختار Dart. از عنوان مقاله هم این موضوع قابل درک و تشخیص می‌باشد. همچنین فرض شده است که دوستان آشنایی مقدماتی با جاوااسکریپت و مباحث شی گرایی را نیز دارند. البته اگر مشغله کاری به بنده این اجازه را بدهد، مطالب جامع‌تری را در این زمینه آماده و منتشر می‌کنم.

گام پنجم: ذخیره سازی اطلاعات در فضای محلی یا Local

در این گام، تغییرات badge را در فضای ذخیره سازی سیستم Local نگهداری می‌نماییم؛ بطوری که اگر دوباره برنامه را راه اندازی نمودید، badge با داده‌های ذخیره شده در سیستم Local مقداردهی اولیه می‌گردد.
کتابخانه dart:convert را به منظور استفاده از کلاس مبدل JSON به فایل piratebadge.dart اضافه نمایید.

```
import 'dart:html';
import 'dart:math' show Random;

import 'dart:convert' show JSON;
```

همچنین یک Named Constructor یا سازنده‌ی با نام را به کلاس PirateName بصورت زیر اضافه کنید.

```
class PirateName {
  ...
  PirateName.fromJSON(String jsonString) {
    Map storedName = JSON.decode(jsonString);
    _firstName = storedName['f'];
    _appellation = storedName['a'];
  }
}
```

توضیحات

- جهت کسب اطلاعات بیشتر در مورد Json به این لینک مراجعه نمایید
- کلاس JSON جهت کار با داده هایی به فرمت Json استفاده می‌شود که امکاناتی را جهت دسترسی سریعتر و راحت‌تر به این داده‌ها فراهم می‌کند.
- سازنده‌ی PirateName.fromJSON، از یک رشته حاوی داده‌ی Json، یک نمونه از کلاس PirateName ایجاد می‌کند.
- سازنده‌ی PirateName.fromJSON، یک Named Constructor می‌باشد. این نوع سازنده‌ها دارای نامی متفاوت از نام سازنده‌های معمول هستند و بصورت خودکار نمونه ای از کلاس مورد نظر را ایجاد نموده و به عنوان خروجی بر می‌گردانند.
- تابع JSON.decode یک رشته‌ی حاوی داده‌ی Json را تفسیر نموده و اشیاء Dart را از آن ایجاد می‌کند.
یک Getter به کلاس PirateName اضافه کنید که مقادیر ویژگی‌های آن را به یک رشته Json تبدیل می‌کند

```
class PirateName {
  ...
  String get jsonString => JSON.encode({"f": _firstName, "a": _appellation});
}
```

جهت ذخیره سازی آخرین تغییرات کلاس PirateName در فضای ذخیره سازی Local، از یک کلید استفاده می‌کنیم که مقدار آن محتوای PirateName می‌باشد. در واقع فضای ذخیره سازی Local داده‌ها را به صورت جفت کلید-مقدار یا Key-Value Pairs نگهداری می‌نماید. جهت تعریف کلید، یک متغیر رشته ای را بصورت top-level و به شکل زیر تعریف کنید.

```
final String TREASURE_KEY = 'pirateName';
```

```
void main() {
  ...
}
```

زمانیکه تغییری در badge name صورت گرفت، این تغییرات را در فضای ذخیره سازی Local، توسط ویژگی window.localStorage ذخیره می‌نماییم. تغییرات زیر را در تابع setBadgeName اعمال نمایید

```
void setBadgeName(PirateName newName) {
  if (newName == null) {
    return;
  }
  querySelector('#badgeName').text = newName.pirateName;
  window.localStorage[TREASURE_KEY] = newName.jsonString;
}
```

تابع getBadgeNameFromStorage را بصورت top-level تعریف نمایید. این تابع داده‌های ذخیره شده را از Local Storage بازیابی نموده و یک شی از نوع کلاس PirateName ایجاد می‌نماید.

```
void setBadgeName(PirateName newName) {
  ...
}

PirateName getBadgeNameFromStorage() {
  String storedName = window.localStorage[TREASURE_KEY];
  if (storedName != null) {
    return new PirateName.fromJSON(storedName);
  } else {
    return null;
  }
}
```

در پایان نیز تابع setBadgeName را به منظور مقدار دهی اولیه به badge name، در تابع main، فراخوانی می‌نماییم.

```
void main() {
  ...
  setBadgeName(getBadgeNameFromStorage());
}
```

حال به مانند گامهای قبل برنامه را اجرا و بررسی نمایید.

گام ششم: خواندن نام‌ها از فایل‌های ذخیره شده به فرمت Json

در این گام کلاس PirateName را به گونه‌ای تغییر می‌دهیم که نام‌ها را از فایل Json بخواند. این عمل موجب می‌شود تا به راحتی اسامی مورد نظر را به فایل اضافه نمایید تا توسط کلاس خوانده شوند، بدون آنکه نیاز باشد کد کلاس را مجددا دستکاری کنید.

به منوی File > New File... مراجعه نموده و فایل piratenames.json را با محتوای زیر ایجاد نمایید. این فایل را در پوشه 1-blankbadge و در کنار فایلهای HTML و Dart ایجاد کنید.

```
{ "names": [ "Anne", "Bette", "Cate", "Dawn",
        "Elise", "Faye", "Ginger", "Harriot",
        "Izzy", "Jane", "Kaye", "Liz",
        "Maria", "Nell", "Olive", "Pat",
        "Queenie", "Rae", "Sal", "Tam",
        "Uma", "Violet", "Wilma", "Xana",
        "Yvonne", "Zelda",
        "Abe", "Billy", "Caleb", "Davie",
        "Eb", "Frank", "Gabe", "House",
        "Icarus", "Jack", "Kurt", "Larry",
        "Mike", "Nolan", "Oliver", "Pat",
        "Quib", "Roy", "Sal", "Tom",
        "Ube", "Val", "Walt", "Xavier",
        "Yvan", "Zeb"],
  "appellations": [ "Awesome", "Captain",
        "Even", "Fighter", "Great", "Hearty",
        "Jackal", "King", "Lord",
```

```
        "Mighty", "Noble", "Old", "Powerful",
        "Quick", "Red", "Stalwart", "Tank",
        "Ultimate", "Vicious", "Wily", "aXe", "Young",
        "Brave", "Eager",
        "Kind", "Sandy",
        "Xeric", "Yellow", "Zesty"]}
```

این فایل شامل یک شی Json با دو لیست رشته ای می‌باشد.

به فایل piratebadge.html مراجعه نمایید و فیلد input و المنت button را غیر فعال نمایید.

```
...
  <div>
    <input type="text" id="inputName" maxlength="15" disabled>
  </div>
  <div>
    <button id="generateButton" disabled>Aye! Gimme a name!</button>
  </div>
...
```

این دو المنت پس از اینکه تمامی نام‌ها از فایل Json با موفقیت خوانده شدند فعال می‌گردند.

کتابخانه dart:async را در ابتدای فایل دارت import نمایید

```
import 'dart:html';
import 'dart:math' show Random;
import 'dart:convert' show JSON;

import 'dart:async' show Future;
```

توضیحات

- کتابخانه dart:async برنامه نویسی غیر همزمان یا asynchronous را فراهم می‌کند

- کلاس Future روشی را ارئه می‌کند که در آن مقادیر مورد نیاز در آینده ای نزدیک و به صورت غیر همزمان واکشی خواهند شد.

در مرحله بعد لیست‌های names و appellations را با کد زیر بصورت یک لیست خالی جایگزین نمایید.

```
class PirateName {
  ...
  static List<String> names = [];
  static List<String> appellations = [];
  ...
}
```

توضیحات

- مطمئن شوید که کلمه کلیدی final را از تعاریف فوق حذف نموده اید

- [] معادل new List() می‌باشد

- کلاس List یک نوع Generic می‌باشد که می‌تواند شامل هر نوع شی ای باشد. اگر می‌خواهید که لیست شما فقط شامل داده هایی از نوع String باشد، آن را بصورت List<String> تعریف نمایید.

دو تابع static را بصورت زیر به کلاس PirateName اضافه نمایید

```
class PirateName {
  ...

  static Future readyThePirates() {
    var path = 'piratenames.json';
    return HttpRequest.getString(path)
        .then(_parsePirateNamesFromJSON);
  }

  static _parsePirateNamesFromJSON(String jsonString) {
    Map pirateNames = JSON.decode(jsonString);
    names = pirateNames['names'];
    appellations = pirateNames['appellations'];
  }
}
```

توضیحات - کلاس HttpRequest یک Utility می‌باشد که داده‌ها را از یک آدرس یا URL خاص واکشی می‌نماید.

- تابع getString یک درخواست را به صورت GET ارسال می‌نماید و رشته ای را بر می‌گرداند
- در کد فوق از کلاس Future استفاده شده است که موجب می‌شود درخواست GET بصورت غیر همزمان ارسال گردد.
- زمانیکه Future با موفقیت خاتمه یافت، تابع then فراخوانی می‌شود. پارامتر ورودی این تابع، یک تابع می‌باشد که پس از خاتمه درخواست GET اجرا خواهد شد. به این نوع توابع که پس از انجام یک عملیات خاص بصورت خودکار اجرا می‌شوند توابع CallBack می‌گویند.
- زمانیکه Future با موفقیت خاتمه یافت، اسامی از فایل Json خوانده خواهند شد
- تابع readyThePirates دارای نوع خروجی Future می‌باشد بطوری که برنامه اصلی در زمانی که فایلها در حال خوانده شدن هستند، به کار خود ادامه میدهد و متوقف نخواهد شد

یک متغیر top-level از نوع SpanElement در کلاس PirateName ایجاد کنید.

```
SpanElement badgeNameElement;

void main() {
  ...
}
```

تغییرات زیر را در تابع main ایجاد کنید.

```
void main() {
  InputElement inputField = querySelector('#inputName');
  inputField.onInput.listen(updateBadge);
  genButton = querySelector('#generateButton');
  genButton.onClick.listen(generateBadge);

  badgeNameElement = querySelector('#badgeName');
  ...
}
```

کد زیر را نیز به منظور خواندن نام‌ها از فایل Json اضافه کنید. در این کد اجرای موفقیت آمیز درخواست و عدم اجرای درخواست، هر دو به شکلی مناسب مدیریت شده اند.

```
void main() {
  ...

  PirateName.readyThePirates()
    .then((_) {
      //on success
      inputField.disabled = false; //enable
      genButton.disabled = false;  //enable
      setBadgeName(getBadgeNameFromStorage());
    })
    .catchError((arrr) {
      print('Error initializing pirate names: $arrr');
      badgeNameElement.text = 'Arrr! No names.';
    });
}
```

توضیحات

- تابع readyThePirates فراخوانی شده است که یک Future بر می‌گرداند.
- زمانی که Future با موفقیت خاتمه یافت تابع CallBack موجود در تابع then فراخوانی می‌شود.

- (_) به عنوان پارامتر ورودی تابع then ارسال شده است، به این معنا که از پارامتر ورودی صرف نظر شود.
- تابع then المنت‌های صفحه را فعال می‌کند و داده‌های ذخیره شده را بازیابی می‌نماید
- اگر Future با خطا مواجه شود، توسط تابع catchError که یک تابع CallBack می‌باشد، پیغام خطایی را نمایش می‌دهیم.

برنامه را به مانند گامهای قبل اجرا نموده و نتیجه را مشاهده نمایید