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.

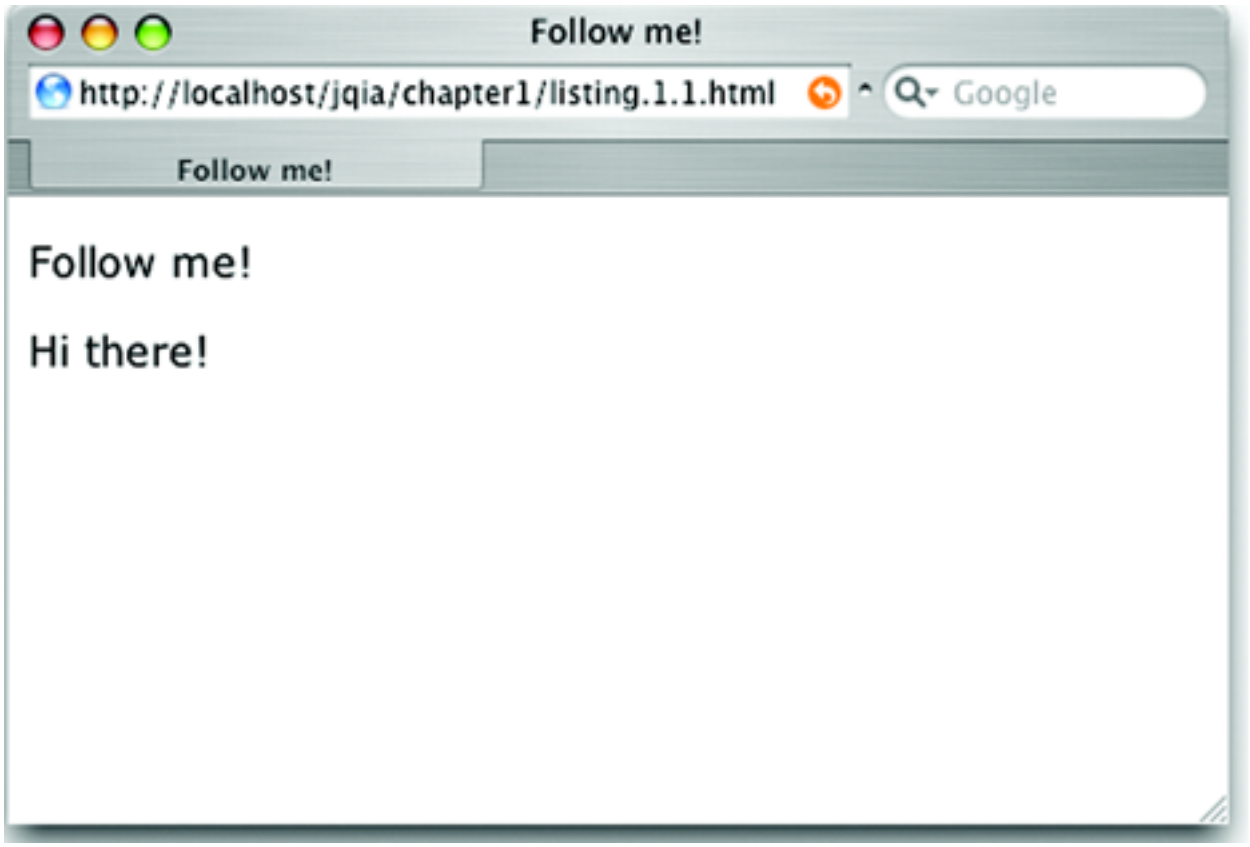
ساختن اجزای DOM (ساختن عناصر صفحه)
یکی دیگر از کارهایی که تابع **()$** می تواند برای ما انجام دهد ایجاد کردن عناصر صفحه است. به این منظور ورودی تابع **()$** را یک رشته که حاوی دستور HTML مربوط به ساخت یک عنصر می‌باشد، قرار می‌دهیم. برای مثال دستور زیر یک تگ p ایجاد می‌کند:

```
$("<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 عنصر صفحه 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 وموید باشید