بخش اول

کتاب‌خانه‌ی هسته‌ای[1]

با شنیدن jQuery، به سرعت ذهن‌ها به سمت کتاب‌خانه‌ی هسته‌ای jQuery معطوف می‌شود، درحالی‌که jQuery یک مجموعه‌ی کامل را شامل می‌شود که متشکل از کتاب‌خانه‌های همراه، مانند واسط‌های گرافیکی jQuery (که در بخش دوم کتاب مورد بحث قرار می‌گیرد)، افزونه‌های[2] رسمی (که قابل رویت در آدرس http://plugins.jquery.com/ می‌باشند) و تعداد بی‌شماری از افزونه‌های غیررسمی که توسط اشخاص مستقل ایجاد شده است و از طریق اینترنت قابل دسترسی می‌باشند (با جستجوی عبارت jQuery plugin بالغ بر ۴ میلیون نتیجه به شما برگردانده می‌شود.)

نکته‌ی قابل ذکر دیگر آن که، کتاب‌خانه‌ی هسته‌ای jQuery، قلب اصلی بازار گسترده‌ی برنامه‌ها و محصول‌های ارایه شده برای iPod می‌باشد؛ همان‌طور که خود iPod نیز مدیون کتاب‌خانه‌ی هسته‌ای jQuery می‌باشد.

در بخش اول کتاب، که هشت فصل نخست را شامل می‌شود، کتاب‌خانه‌ی هسته‌ای jQuery را به طور کامل مورد بررسی قرار خواهیم داد. با اتمام هشت فصل اول، شما دید کاملی از کتاب‌خانه‌ی هسته‌ای jQuery خواهید داشت و می‌توانید با تکیه بر یکی از قدرتمندترین ابزارهای سمت کاربر[3]، از پس هرگونه وب سایتی برآیید. هم‌چنین بخش اول کتاب شما را برای استفاده از کتاب‌خانه‌های همراه jQuery نیز، مانند برنامه‌های جانبی iPod، مهیا می‌کند.

پس بخش اول کتاب را آغاز کنید و نه تنها برنامه‌های تحت وب را به سادگی ایجاد کنید، بلکه از آن لذت هم ببرید!

---

1. Core
2. Plugin
3. Client Side

فصل

۱

آنچه در این فصل می‌خوانیم:

- دلیل استفاده از jQuery؛
- جاوا اسکریپت تفکیکی چیست؟؛
- مفاهیم و عناصر پایه‌ای jQuery؛
- استفاده از jQuery به همراه سایر کتاب‌خانه‌های جاوا اسکریپت.

معرفی jQuery

پس از مدت‌ها به سخره گرفتن و نام بردن جاوا اسکریپت به عنوان یک زبان «نه خیلی جدی»، به لطف افزایش تقاضا برای برنامه‌های تحت وب پرتعامل[1] (که معمولاً به نام‌های rich internet application و یا Dim-scripted application شناخته می‌شود) و هم‌چنین تکنولوژی‌های Ajax، جاوا اسکریپت توانسته است جایگاه خود را در چند سال اخیر مجدداً به دست آورد. این زبان توانسته است با ارایه‌ی یک الگوی جدید و بهبود یافته، تولیدکنندگان برنامه‌های سمت کاربر را متقاعد کند تا به جای استفاده‌ی محدود از آن، از تمام کتاب‌خانه‌های جاوا اسکریپت بهره ببرند و مشکلاتی مانند cross-browser را به سادگی برطرف کنند.

یکی از موارد وابسته به دنیای جاوا اسکریپت که به تازگی متولد شده است، jQuery می‌باشد. jQuery، بر خلاف عمر کوتاهش توانسته است محبوبیت زیادی به دست آورد و مورد توجه شمار قابل توجهی از کمپانی‌های بزرگ قرار گیرد. IBM، Netflix، Amazon، Dell، Best Buy، Twitter، Bank of America و تعدادی دیگر از کمپانی‌ها رامی‌توان به عنوان برخی از کاربران دایمی jQuery نام برد. مایکروسافت برای توزیع jQuery به همراه ابزارهای visual studio، کاندید شد و Nokia از jQuery برای تمام گوشی‌هایش، که شامل نرم افزار اجرای تحت وب می‌باشند، استفاده می‌کند. این موارد دیگر بیان کننده‌ی «خیلی جدی نبودن» نیستند!

در مقایسه با سایر ابزارهایی که تأکید عمده‌ای بر روی تکنیک‌های هوشمند جاوا اسکریپت دارند، هدف jQuery 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

1. Highly Interactive

استفاده از زمان و دانش خود در زمینه‌ی CSS، HTML، XHTML و جاوا اسکریپت‌های ساده، عناصر صفحه را مستقیماً دست‌کاری کنند و از همین طریق تغییرهای گسترده و سریعی انجام دهند.

در این کتاب نگاهی دقیق خواهیم داشت به امکاناتی که jQuery برای توسعه‌دهندگان صفحه‌های وب پر تعامل ارایه می‌دهد و این کار را با بررسی این نکته آغاز می‌کنیم که jQuery دقیقاً چه چیزی را برای توسعه‌کنندگان صفحه‌های وب آورده است؟

آخرین و جدیدترین نسخه‌ی jQuery را می‌توان از وب سایت آن، http jQuery.com: //، تهیه کرد. نصب آن نیز تنها مستلزم قرار دادن آن در کنار وب سایت و استفاده از تگ <script> در صفحه‌ی موردنظر می‌باشد:

```
<script type="text/javascript"
       src="scripts/jquery-1.4.js"></script>
```

نسخه‌ای که مثال‌های این کتاب با آن تست شده است، به همراه مثال‌های این کتاب در فایلی به آدرس http://www.manning.com/bibeault2 قابل دانلود می‌باشد.

۱-۱- قدرت کوتاه کردن کد

هر زمان که شما خواسته باشید کارکرد صفحه‌ای را پویاتر کنید، در اکثر اوقات به ناچار این کار را از طریق عناصری بر روی صفحه انجام داده‌اید که با توجه به انتخاب شدن آن‌ها، صفحه کارکردی خاص خواهد داشت. برای مثال جزیی از صفحه را مخفی یا آشکار می‌کنید، خصوصیات آن را تغییرمی‌دهید و یا مشخصات CSS آن را تغییر می‌دهید. به منظور انجام هر یک از تغییرهای ذکر شده، ده‌ها خط کد جاوا اسکریپت احتیاج است، از همین‌رو طراحان و سازندگان jQuery مشخصاً سعی در تسهیل انجام چنین تغییرهایی کرده‌اند. برای مثال هر کسی که با Radio group در جاوا اسکریپت کار کرده باشد، به خوبی می‌داند که پروسه‌ی تشخیص گزینه‌ی انتخاب شده و واکشی مقدار آن بسیار یکنواخت و خسته‌کننده است. ابتدا می‌بایست Radio group انتخاب شود؛ تک‌تک عناصر آن در قالب آرایه‌ای بررسی شوند و مقدار آن عنصری که صفت checked آن true می‌باشد، برگردانده شود. پس از آن، مقدار قابل استفاده می‌باشد. کدی که برای چنین کاری در جاوا اسکریپت نوشته می‌شود، کدی مانند زیر خواهد بود:

```
var checkedValue;
var elements = document.getElementsByTagName('input');
for (var n = 0; n < elements.length; n++) {
       if (elements[n].type == 'radio' &&
```

```
            elements[n].name == 'someRadioGroup' &&
            elements[n].checked) {
        checkedValue = elements[n].value;
        }
}
```

اما اگر بخواهیم همین کار را با استفاده از jQuery انجام دهیم، با نوشتن تنها یک خط کد به خواسته‌ی خود می‌رسیم:

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

ممکن است مثال فوق کمی شما را نگران کند؛ جای هیچ‌گونه نگرانی نیست، به زودی چگونگی کارکرد دستور بالا را فرا خواهید گرفت و قادر به استفاده از آن در صفحه‌های ایجاد شده توسط خودتان خواهید بود. اما اجازه دهید، نگاهی به قسمت‌های مختلف این دستور قدرتمند داشته باشیم.

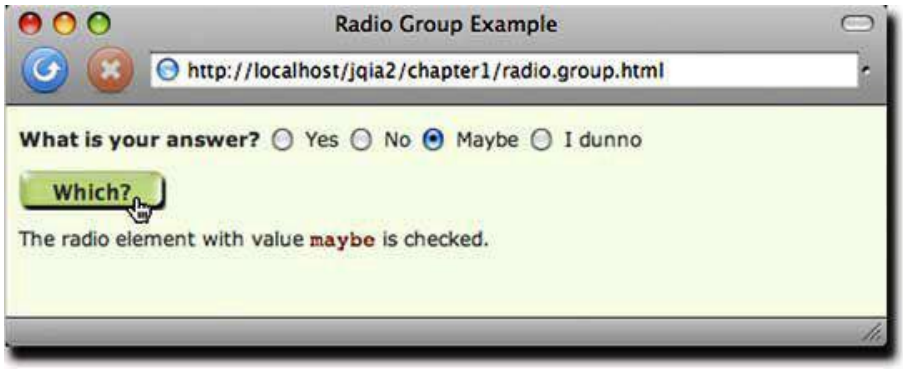
ابتدا تمام عناصری که مقدار صفت name آن‌ها برابر SomeRadioGroup می‌باشند را مشخص می‌کنیم (اگر به خاطر داشته باشید تمام عناصر یک Radio Group تحت یک نام معرفی می‌شوند)، سپس با اعمال فیلتری بر روی آن مجموعه، عناصری که صفت checked آن‌ها در وضعیت مطلوب است، جدا می‌شوند و مقدار آن‌ها برگردانده می‌شود (که در هر لحظه تنها یک عنصر می‌تواند در وضعیت انتخاب قرار داشته باشد).

قدرت اصلی jQuery، مدیون انتخاب‌کننده[1] است. 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 موردنظر را مشخص و به آن دسترسی پیدا کنیم که در مثال فوق، عنصر موردنظر ما گزینه‌ی انتخاب شده از Radio Group بود.

اگر کدهای مربوط به مثال‌های کتاب را دانلود نکرده‌اید، پیشنهاد می‌کنیم اکنون این کار را با مراجعه به آدرس http://www.manning.com/bibeault2 انجام دهید. فایل را از حالت فشرده خارج سازید و پس از آن می‌توانید به هر یک از مثال‌ها به صورت مجزا دسترسی داشته باشید و یا از فهرستی که در فایل index.html تهیه شده است، استفاده کنید.

صفحه‌ی HTML، مربوط به فایل Chapter1/radio.group.html را در مرورگر خود باز کنید. این صفحه که در شکل ۱-۱ نشان داده شده است، با استفاده از همان دستور jQuery که بررسی شد، مشخص می‌کند که کدام دکمه از دکمه‌های Radio Group انتخاب شده است.

1. Selector



شکل ۱-۱- امکان تعیین دکمه‌ی انتخاب شده به سادگی و تنها با یک خط دستور jQuery

حتی همین مثال کوچک نیز می‌بایست شما را متوجه این نکته کرده باشد که jQuery راهی آسان و بی‌دردسر برای ایجاد نسل آینده‌ی برنامه‌های تحت وب پرتعامل می‌باشد. اما صبر کنید تا قدرت‌های دیگر آن را برای ایجاد صفحه‌های موردنظرتان ببینید که با پیشرفت کتاب با آن‌ها آشنا می‌شویم.

به زودی خواهیم دید که چگونه می‌توانیم انتخاب کننده‌هایی برای این قبیل موارد ایجاد کنیم، اما پیش از آن، اجازه دهید نظر سازندگان jQuery را برای استفاده‌ی بهتر از جاوا اسکریپت در صفحه‌هایمان بررسی کنیم.

۱-۲- جاوا اسکریپت تفکیکی[1]

اگر پیش از پیدایش CSS، در کار ایجاد صفحه‌های اینترنتی بوده‌اید حتماً مشکلات و سختی‌های آن دوران را به خاطر دارید. در آن زمان برای فرمت‌دهی به اجزای مختلف صفحه، به ناچار علایم فرمت‌دهی را به همراه دستورهای خودِ اجزا، در صفحه‌های HTML استفاده می‌کردیم.

اما با اضافه شدن CSS به ابزارهای موجود برای ساخت و طراحی صفحه‌های اینترنتی، این امکان فراهم شد تا بخش مربوط به فرمت‌دهی را، از بخش دستورهای اجزا جدا کنیم و در نتیجه دیگر خبری از تگ‌هایی مانند <font> نباشد. جدا کردن بخش‌های فرمت‌دهی و دستورهای اجزا، افزون بر امکان مدیریت مطمئن‌تر و ساده‌تر کد، به ما این توانایی را داد تا صفحه‌هایی با شکل ظاهری بهتر نیز داشته باشیم، زیرا به راحتی امکان مبادله و استفاده‌ی دستورهای مختلف CSS در فایل‌های گوناگون فراهم شد.

---

1. Unobtrusive JavaScript

واژه‌ی unobtrusive به معنی محجوب می‌باشد، اما با توجه به ساختار و مدلی که این گونه کدنویسی معرفی می‌کند، در ترجمه‌ی این کتاب از عبارت جاوا اسکریپت تفکیکی به جای جاوا اسکریپت محجوب استفاده شده است.

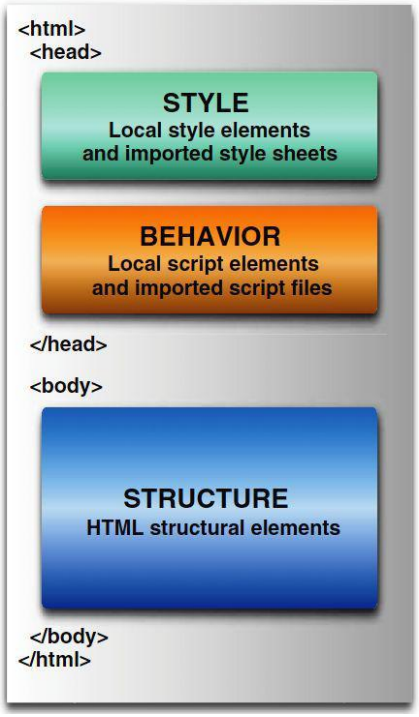
اکنون بسیار بعید به نظرمی‌رسد کسی ترجیح دهد فرمت‌دهی اجزا را به همراه دستورهای HTML آن انجام دهد، اگر چه هنوز دستوری مانند مثال زیر، بسیار معمول و عادی به نظر می‌رسد:

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

نکته‌ای که در مثال فوق دارای اهمیت است، این است که خصوصیات ظاهری دکمه‌ی ایجاد شده، از قبیل فونت و عنوان دکمه، از طریق تگ <font> و یا پارامترهای قابل استفاده در خود دستور دکمه معین نشده است، بلکه CSS وظیفه‌ی تعیین آن‌ها را دارد. اما اگرچه در مثال فوق، فرمت‌دهی و دستور خود دکمه از یک‌دیگر جدا شده‌اند، شاهد ترکیب این دکمه با رفتار آن هستیم. هر زمان که دکمه فشرده شود، جاوا اسکریپت اجرا می‌شود چرا که بخشی از صفت onclick این دکمه در نظر گرفته شده است (که در این مثال رنگ دکمه‌ای با مقدار شناسه‌ی xyz را به قرمز تغییر می‌دهد). اجازه دهید نگاهی داشته باشیم به این مسئله که چگونه می‌توان چنین وضعیتی را بهبود بخشید.

۱-۲-۱- جدا کردن رفتار از ساختار

درست به همان دلایلی که جدا کردن بخش مربوط به فرمت‌دهی و ساختار اجزا از یک‌دیگر، مفید و ضروری تلقی می‌شوند، جدا کردن رفتار اجزا از ساختارشان نیز امری مهم محسوب می‌شود. یک صفحه‌ی ایده‌آل HTML، می‌بایست چیدمانی مانند شکل ۲-۱ داشته باشد؛ فرمت‌دهی‌های ظاهری، رفتار و ساختار، هر یک جدا از هم و در جای مناسب خود باشند.



شکل ۱-۲- با جدا قرار دادن هر یک از قسمت‌های یک صفحه‌ی اینترنتی، خوانایی، مدیریت و نگهداری آن تضمین می‌شود.

این تکنیک که با نام جاوا اسکریپت تفکیکی شناخته می‌شود، توسط سازندگان jQuery مورد توجه قرار گرفت و اکنون شمار زیادی از کتاب‌خانه‌های جاوا اسکریپت این تکنیک را پذیرفته‌اند و از این طریق، ساختار، رفتار و فرمت ظاهری اجزای صفحه‌ها را از یک‌دیگر جدا می‌سازند. بدیهی است که jQuery به عنوان کتاب‌خانه‌ای که جاوا اسکریپت تفکیکی را معروف کرد، تدابیر مناسبی جهت این‌گونه کدنویسی اندیشیده است. جاوا اسکریپت تفکیکی هیچگونه عبارت جاوا اسکریپت در تگ <body> صفحه‌های HTML را مجاز نمی‌داند، خواه این عبارت صفت یکی از اجزا باشد (مانند onclick) و یا عبارتی نوشته شده میان تگ‌های <script> باشد.

با توجه به مطالب فوق تعریف دکمه‌ی مثال قبل به شکل زیر تغییر می‌کند. توجه کنید که onclick از آن حذف و به جای آن یک id (شناسه) در تعریف دکمه جای‌گزین شده است:

```
<button type="button" id="testButton">Click Me</button>
```

تعریف ساده‌تر شد، اما هرچه بر روی این دکمه کلیک کنیم هیچ اتفاقی نخواهد افتاد، زیرا رویدادی برای onclick مشخص نشده است. این مسئله را می‌توان با استفاده از تفکیک بخش‌های کد برطرف کرد.

۱-۲-۲- تفکیک بخش‌های کد

به جای آن که مانند مثال قبل رفتارهای دکمه را در تعریف ساختار آن مشخص کنیم، آن را به خارج از محدوده‌ی body، در تگ <Head> و در یک بلاک script معرفی می‌کنیم. به مثال زیر دقت کنید:

```
<script type="text/javascript">
       window.onload = function() {
              document.getElementById('testButton').onclick = function() {
                     document.getElementById('xyz').style.color   =
              'red';
              };
       };
</script>
```

قطعه کد بالا را به رویداد onload صفحه متصل کرده‌ایم و در همان‌جا با استفاده از یک تابع درون خطی، رفتار مناسب دکمه را در قبال کلیک بر روی آن مشخص کرده‌ایم. نکته‌ی قابل توجه آن‌که می‌توانستیم قطعه کد بالا را به صورت یک تابع خطی معرفی کنیم، اما از آن‌جا که می‌خواهیم به طور قطع دکمه ابتدا ساخته شود و سپس رفتاری به آن افزوده شود، قطعه کد را به رویداد onload صفحه افزوده‌ایم. در بخش ۳-۳-۱ خواهیم دید که jQuery مکانی بهتر برای نوشتن چنین کدی را مهیا کرده است.

نکته: از جهت کارایی بهتر می‌توان بلاک‌های script را پس از تگ < body > نوشت، اما به هر حال نکته‌ی اصلی و مهم آن است که رفتار اجزای صفحه از تعریف ساختارشان جدا باشد.

اگر هر قسمت از کد مثال فوق را متوجه نشده‌اید (برای مثال ساختار تابع درون خطی)، نگران نباشید، در پیوست کتاب تمام آن‌چه که از جاوا اسکریپت برای استفاده‌ی بهینه از jQuery احتیاج دارید، آورده شده است. هم‌چنین در ادامه‌ی فصل روشی دیگر برای نوشتن همین کد را معرفی خواهیم کرد که کوتاه‌تر و خواناتر خواهد بود.

با وجود قدرتی که جاوا اسکریپت تفکیکی در جدا کردن مسئولیت‌های مختلف یک صفحه‌ی اینترنتی موجب می‌شود، به نظر می‌رسد یک عیب نیز به همراه دارد. شاید این نکته نظر شما را به خود معطوف کرده باشد که نوشتن کد به صورت تفکیکی، مستلزم نوشتن خطوط کد بیشتری می‌باشد. جاوا اسکریپت تفکیکی به دلیل ایجاد یک الگوی خاص برای نوشتن کدها تعداد خطوط را بالاتر می‌برد، اما در واقع خود این امر نیز می‌تواند یک مزیت تلقی شود. هر چیز که باعث شود کد نوشته شده‌ی ما منظم‌تر باشد و از ساختار و دقت معینی که معمولاً برای برنامه‌های سمت

سرور لحاظ می‌شود، برخوردار شود، می‌تواند نکته‌ی مثبتی تلقی شود. اما به هر حال با استفاده از jQuery حتی همین افزایش تعداد خطوط کد را نیز نخواهیم داشت. پیش‌تر گفته شد که تمرکز اصلی jQuery بر استفاده‌ی آسان از جاوا اسکریپت تفکیکی می‌باشد. بنابراین در ادامه خواهید دید که چگونه jQuery ما را از نوشتن انبوهی از کدها رها می‌سازد و چگونه می‌توانیم با کدهای کمتر، کارهای بیشتری انجام دهیم.

بدون مقدمه‌ی بیشتر، به چگونگی افزایش کارایی صفحه‌ها با استفاده از سهولت ارایه شده‌ی توسط jQuery نگاهی خواهیم داشت.

۱-۳- مقدمات jQuery

تمرکز اصلی jQuery واکشی اجزایی از صفحه‌ی HTML و انجام عملیاتی بر روی آن‌ها می‌باشد. اگر با CSS آشنایی داشته باشید، از قدرت انتخاب کننده‌ها مطلع هستید که چگونه گروهی از اجزای یک صفحه‌ی HTML را انتخاب می‌کنند و مشخصات ظاهری خاصی را به آن‌ها اختصاص می‌دهند. حال با استفاده از jQuery می‌توانیم همین کار را برای ساده کردن کدهای جاوا اسکریپت انجام دهیم.

jQuery توجه ویژه‌ای به پایداری صفحه‌ها در مرورگرهای مختلف دارد و هم‌چنین برخی از مشکل‌های عدیده‌ی جاوا اسکریپت را پشت پرده برطرف می‌سازد. برای مثال مشکل انتظار تا load شدن کامل صفحه و سپس تعامل با صفحه، به راحتی برطرف شده است.

jQuery متدهای از پیش‌ساخته شده‌ی قدرتمندی برای افزایش کارایی و توانش در کار با افزونه‌ها ارایه می‌دهد، که قدرت و سهولت را برای ما به همراه دارد.

اما اکنون ببینیم که چگونه می‌توانیم از دانش CSS در نوشتن کدی کوتاه و مختصر و در عین حال قدرتمند با jQuery استفاده کنیم.

۱-۳-۱- مجموعه عناصر در jQuery

زمانی‌که CSS به عنوان یک تکنولوژی به منظور جداسازی طراحی از ساختار به دنیای صفحه‌های اینترنتی معرفی شد می‌بایست راهی برای اشاره به اجزای صفحه‌ها از طرف فایل CSS نیز معرفی می‌شد. این امر از طریق انتخاب‌کننده‌ها صورت پذیرفت.

شاید کسانی که با XML آشنایی داشته باشند، انتخاب‌کننده‌های CSS را چیزی مانند XPath فرض کنند. انتخاب‌کننده‌های CSS، در حالی‌که همان قدرت را دارند، طوری طراحی شده‌اند تا برای

کار با صفحه‌های HTML مناسب باشند. به علاوه آن که از اختصار بیشتر و خوانایی بالاتری نیز بهره می‌برند.

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

```
p  a
```

jQuery نیز از چنین انتخاب‌کننده‌هایی استفاده می‌کند، البته نه تنها از انتخاب‌کننده‌هایی که هم‌اکنون در CSS موجود می‌باشند، بلکه برخی از انتخاب‌کننده‌هایی که هنوز در تمام مرورگرها پشتیبانی نمی‌شوند، از جمله شماری از انتخاب‌کننده‌های CSS3 استفاده می‌شود.

برای انتخاب مجموعه‌ای ازعناصر، یکی از دو ساختار[1] زیر را استفاده می‌کنیم:

```
$(selector)
jQuery(selector)
```

ممکن است در ابتدا ()$ کمی نامعمول به نظر آید، اما اکثر کسانی که با jQuery 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 ایجاد می‌کند:

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

تابع ()$، که در حقیقت نام خلاصه‌ای برای ()jQuery می‌باشد، نوع خروجی به خصوصی دارد که شامل یک آرایه از اشیایی می‌شود که انتخاب‌کننده، آن‌ها 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 برنامه‌نویسی به چنین توابعی که گروهی از عناصر را جمع می‌کنند، Wrapper، به معنای بسته‌بند می‌گویند زیرا تمام عناصر مطلوب را تحت یک شی بسته‌بندی می‌کند. در jQuery به آن‌ها jQuery wrapper و یا wrapped set می‌گویند که در این کتاب ما از عنوانِ مجموعه عناصر انتخاب شده، استفاده می‌کنیم. هم‌چنین متدهای قابل اعمال بر روی آن‌ها نیز با نام jQuery wrapper methods شناخته می‌شوند.

در مثال زیر می‌خواهیم تمام عناصر ، در صورتی‌که دارای کلاس notLongForThisWorld باشند، را مخفی کنیم:

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

1. Syntax