بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

متد ()live در jQuery

مدرس : مهندس افشین رفوآ

دوره آموزشی Jquery

متد ()live در jQuery

در دو مبحث قبلی با استفاده از توابع ()bind و ()unbind، event handler ها را به المان های مختلف موجود در صفحه به ترتیب متصل کرده و از آن ها حذف می کردیم. این دو تابع برای المان هایی کاربرد دارد که هم اکنون بر روی صفحه موجود می باشند. اما اگر بخواهیم event handler ها را به المان هایی که در آینده به صفحه اضافه می شوند، متصل کنیم چه اقدامی را باید انجام دهیم؟ به طور معمول این کار را به صورت دستی و به هنگام ایجاد المان های جدید انجام می دهیم که اگرچه امکان پذیر است، اما کارآمد نیست. کتابخانه ی jQuery تابع بسیار پرکاربردی به نام ()live را ارائه می دهد که به واسطه ی آن می توان event handler دلخواه را به المان هایی که ممکن است در آینده اضافه شوند و با selector مشخص شده (مثلا اسم کلاس) منطبق می باشند، bind کرد. به منظور تفهیم بهتر تفاوت دو متد نام برده، ابتدا متد ()bind را برای متصل کردن event handler به المان بکار می بریم و در مثال بعدی همین کار را با ()live انجام می دهیم:

```
<div id="divTestArea1">
  <a href="javascript:void(0);" onclick="AddBox();">Add box</a>
  <div class="test">This is a box</div>
</div>

<script type="text/javascript">
  $(function () {
    $(".test").bind("mouseover", function () {
      $(this).css("background-color", "blue");
    }).bind("mouseout", function () {
      $(this).css("background-color", "white");
    });
  });

  function AddBox() {
```

```
    var div = $("<div></div>").addClass("test").text("Another box");
    $("#divTestArea1").append(div);
  }
</script>
```

در این مثال یک لینک داریم که با کلیک بر روی آن متد AddBox() فراخوانی می شود. علاوه بر آن یک المان div در دست داریم که اسم کلاس آن "test" می باشد. متد AddBox() صرفا یک المان div دیگر با همان نام کلاس به صفحه اضافه می کند تا با کلیک کاربر بر لینک یک کادر دیگر به صفحه اضافه گردد. در رخداد ready، تمامی المان هایی که نام کلاس آن ها "test" می باشد را گزینش کرده و بعد یک handler به دو رخداد mouseover و mouseout متصل می کنیم که طی آن رنگ المان فراخوانده ی رخداد تغییر می یابد.

این مثال را در مرورگر خود امتحان کنید. اولین div افکت mouseover را نمایش می دهد، اما زمانی که برای افزودن کادرهای جدید بر روی لینک کلیک می کنید، کادر های جدید دیگر همان افکت را نخواهند داشت. دلیل آن هم مشخص است: رخدادهای مزبور را قبل از ایجاد کادرهای جدید، bind کردیم.

اکنون مثال زیر را امتحان کنید. در این مثال تنها چیزی که نسبت به نمونه ی بالا تغییر یافته، متد live() بجای bind() می باشد:

```
<div id="divTestArea2">
  <a href="javascript:void(0);" onclick="AddBox();">Add box</a>
  <div class="test">This is a box</div>
</div>

<script type="text/javascript">
  $(function () {
    $(".test").live("mouseover", function () {
      $(this).css("background-color", "blue");
    }).live("mouseout", function () {
      $(this).css("background-color", "white");
    });
  });

  function AddBox() {
    var div = $("<div></div>").addClass("test").text("Another box");
    $("#divTestArea2").append(div);
  }
</script>
```

حال اگر نمونه ی بالا را اجرا کنید، می بینید که اگرچه المان جدید پس از بارگذاری کامل صفحه ایجاد شده، اما jQuery (به واسطه ی متد live()) به صورت خودکار event handler های mouseover و mouseout (تعریف شده در رخداد ready) را به المان جدید نیز متصل می کند. بدین وسیله المان div ای که به انتهای

المان جاری الحاق می شود نیز از افکت تعریف شده در کد برخوردار می باشد (زمانی که موس بر روی div قرار می گیرد رنگ پس زمینه ی آن آبی شده و با ترک موس رنگ پس زمینه ی المان مذکور سفید می شود). متد die() نیز کاربردی مشابه unbind() دارد، اما فقط زمانی باید بکار گرفته شود که رخدادها توسط live() به المان های مورد نظر متصل شده اند.