عنوان: **Syntax highlighting در بلاگر!**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۸/۰۷ ۱۶:۱۱:۳۳
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

اگر علاقمند باشید که syntax highlighting را به سورس کدهای ارسالی در بلاگر اضافه کنید، روش کار به صورت زیر است:
از آنجائیکه دسترسی به سرور و راه حل‌های سمت سرور را نخواهیم داشت، تنها راه حل باقیمانده استفاده از روش‌های سمت کلاینت است. کتابخانه زیر این امر را میسر می‌سازد:

http://code.google.com/p/syntaxhighlighter/

این کتابخانه، کار Syntax highlighting سمت کلاینت را با استفاده از JavaScript انجام می‌دهد.

پس از دریافت آن (احتمالا به یک پروکسی نیاز پیدا خواهید کرد ...)، فایل‌ها را در یک سرور قرار دهید. (برای مثال در Google pages)
سپس به قسمت ویرایش html قالب سایت مراجعه کنید و کدهای زیر را به آن اضافه نمائید (درصورت نیاز مسیرهای فایل‌ها را ویرایش کنید):

```
<link href='http://vahid.nasiri.googlepages.com/SyntaxHighlighter.css' rel='stylesheet'
type='text/css'/>
<script src='http://vahid.nasiri.googlepages.com/shCore.js' type='text/javascript'/>
<script src='http://vahid.nasiri.googlepages.com/shBrushCpp.js' type='text/javascript'/>
<script src='http://vahid.nasiri.googlepages.com/shBrushCSharp.js' type='text/javascript'/>
<script src='http://vahid.nasiri.googlepages.com/shBrushCss.js' type='text/javascript'/>
<script src='http://vahid.nasiri.googlepages.com/shBrushJava.js' type='text/javascript'/>
<script src='http://vahid.nasiri.googlepages.com/shBrushJScript.js' type='text/javascript'/>
<script src='http://vahid.nasiri.googlepages.com/shBrushSql.js' type='text/javascript'/>
<script src='http://vahid.nasiri.googlepages.com/shBrushXml.js' type='text/javascript'/>

<script class='javascript'>
//<![CDATA[
  function FindTagsByName(container, name, Tag)
  {
      var elements = document.getElementsByTagName(Tag);
      for (var i = 0; i < elements.length; i  )
      {
          if (elements[i].getAttribute("name") == name)
          {
              container.push(elements[i]);
          }
      }
  }
  var elements = [];
  FindTagsByName(elements, "code", "pre");
  FindTagsByName(elements, "code", "textarea");

for(var i=0; i < elements.length; i  ) {
if(elements[i].nodeName.toUpperCase() == "TEXTAREA") {
 var childNode = elements[i].childNodes[0];
 var newNode = document.createTextNode(childNode.nodeValue.replace(/<br\s*\/?>/gi,'\n'));
 elements[i].replaceChild(newNode, childNode);

}
else if(elements[i].nodeName.toUpperCase() == "PRE") {
 brs = elements[i].getElementsByTagName("br");
 for(var j = 0, brLength = brs.length; j < brLength; j  ) {
  var newNode = document.createTextNode("\n");
  elements[i].replaceChild(newNode, brs[0]);
 }
}
}
// dp.SyntaxHighlighter.ClipboardSwf =
//"http://vahid.nasiri.googlepages.com/clipboard.swf";
dp.SyntaxHighlighter.HighlightAll("code");
//]]>
</script>
```

خطوط فوق باید پس از تگ‌های زیر در قالب استاندارد قرار داده شوند:

```
</div></div> <!-- end outer-wrapper -->
```

از این پس جهت استفاده از این قابلیت تنها کافی است از تگ‌های pre یا textarea استفاده کنید (در قسمت html ارسال مطلب) و name را مساوی code قرار داده و language را مساوی زبان مورد نظر. برای مثال:

```
<div align="left" dir="ltr">
<pre name='code' language='sql'>
--get login time
SELECT login_time FROM master..sysprocesses WHERE spid = 1
</pre>
</div>
```

که نتیجه نهایی به صورت زیر خواهد بود:

```
--get login time
SELECT login_time FROM master..sysprocesses WHERE spid = 1
```

همچنین باید دقت داشت که مجاز به ارسال کاراکترهای غیرمجاز در xml (&'\<>) در کدهای خود نیستید و این کاراکترها سبب خواهند شد که کد شما نمایش داده نشوند. به همین جهت همانطور که پیشتر نیز ذکر شد می‌توان از سرویس سایت http://www.elliotswan.com/postable/ استفاده کرد.

ماخذ:

http://developertips.blogspot.com/2007/08/syntaxhighlighter-on-blogger.html

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۷/۰۴ ۱۷:۴۰:۲۶

تصحیح!
در کد فوق، ++i هست و نه i (در هر دو حلقه). همچنین یک ++j هم در حلقه بعدی هست. یا به اصل کد مراجعه کنید در ماخذ یاد شده.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۷/۰۴ ۱۷:۵۲:۰۵

و در کل این روش از همه بهتر است:
http://copysourceashtml.codeplex.com/

نویسنده: mo30
تاریخ: ۱۳۹۰/۰۱/۱۴ ۱۳:۰۴:۳۸

نه نه.اشتباه نکن.
برنامه شما بسیار عالی کار می کنه.
اما من از ترفند این
http://www.techknowl.com/show-html-and-java-script-inside.html
سایت استفاده کردم و ترفندی که توی سایت شماست اصلا برای من کار نمی کنه.
اما همین ترفند خودم هم به صورتی که توی تصویر نشون دادم مشکل داره و جالب اینجاست که فقط برای یه پوسته خاص این مشکل وجود داره.

http://025.img98.net/out.php/i217666_Untitled1.jpg

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۴ ۱۳:۳۲:۴۱

مشکل در پوسته انتخابی شما است که قسمت ارسال مطالب دوبار در آن تکرار شده چون اگر به اچ تی ام ال صفحه نهایی رندر شده خودتون مراجعه کنید می‌بینند که دوبار این قسمت مطالب هست یعنی قالب ایراد دارد و یک قسمت تکراری دارد که باید حذف شود
مشکل از اسکریپت یا جای دیگر نیست. مشکل از قالب اصلی انتخابی است

عنوان: **استفاده از popBox برای کوچک کردن خودکار تصاویر بزرگ**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۱۷ ۱۱:۲۶:۲۳
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

یکی از دوستان در قسمت تماس با من پیغام گذاشته بود که چگونه تصاویر یک صفحه را می‌شود به صورت موقت کوچک کرد؟ برای نمونه، این مشکلی است که در فوروم‌ها زیاد وجود دارد. گاهی از اوقات یکی از کاربران تصویری را ارسال می‌کند که اندازه‌ی آن در حد یک تابلوی دیواری است!
با استفاده از محصول سورس بازی به نام popBox می‌توان تصاویر دلخواهی را ابتدا با اندازه‌ی کوچک در صفحه نشان داد و پس از کلیک بر روی آن، تصویر به اندازه اصلی آن نمایش داده خواهد شد. به این صورت صفحه از شکل نخواهد افتاد. این اسکریپت از آدرس زیر قابل دریافت است:
http://www.c6software.com/products/popbox/default.aspx
نحوه استفاده و مثال‌های مربوطه هم در همان آدرس قابل مشاهده است و نکته خاصی ندارد.

اما این روش یا روش‌های مشابه آن یک مشکل دارند. اعمال آن‌ها باید به صورت دستی صورت گیرد. یعنی باید دقیقا مشخص کنیم که این تصویر معلوم در ابتدا به اندازه‌ی مشخصی نمایش داده شود و سایر تنظیمات مربوطه اعمال گردد. این روش در سایت‌های پویا عملا پاسخگو نخواهد بود. زمانیکه کاربری مطلبی را ارسال می‌کند یا باید در سورس‌های موجود دست برد و به تصاویر ارسالی به صورت خودکار تنظیمات مربوط به popbox را اضافه کرد، یا اینکه هر بار مطالب را پس از ارسال به صورت دستی ویرایش کرد یا روش‌هایی مانند این.

با استفاده از jQuery ، اسکریپت selectAll.js را به این مجموعه اضافه کردم. ابتدا تمامی تصاویر صفحه پیدا می‌شوند و سپس به آنهایی که طول و عرض بزرگتری از حد مجاز دارند به صورت خودکار ویژگی‌های popBox افزوده خواهد شد. چون این روش اسکریپتی است : الف) نیازی به تغییر در سورس‌های موجود نیست. ب) توسط هر تکنولوژی سمت سروری نیز قابل استفاده است. ج) به صورت خودکار به تمامی تصاویر بزرگ صفحه اعمال می‌شود و نیازی به ویرایش دستی هیچ موردی جهت اعمال ویژگی‌های مورد نظر وجود نخواهد داشت.

نکته جالبی که با jQuery وجود دارد، روال رخداد گردان document.ready آن است. این رخداد پیش از window.onload استاندارد و زمانیکه DOM آماده استفاده است فراخوانی می‌شود.

برای استفاده از آن ابتدا باید دو فولدر scripts و images به سرور کپی شوند. سپس سه سطر زیر را به هدر صفحه اضافه کنید (به سورس صفحه sample.html فایل پیوست می‌توان مراجعه کرد):

```
<script src="scripts/jquery.min.js" type="text/javascript"></script>
<script src="scripts/PopBox.js" type="text/javascript"></script>
<script type="text/javascript" src="scripts/selectAll.js"></script>
```

در مورد سایر تنظیمات، فایل selectAll.js را گشوده و موارد زیر را در صورت لزوم ویرایش نمائید:

```
popBoxWaitImage.src = "images/spinner40.gif";
popBoxRevertImage = "images/magminus.gif";
popBoxPopImage = "images/magplus.gif";
popBoxRevertText = "کوچک نمایی";
popBoxPopText = "بزرگ نمایی";
popBoxCaptionMoreText = "بیشتر";
popBoxCaptionLessText = "کمتر";
MaximumAllowedWidth = 500;
MaximumAllowedHeight = 480;
```

در این فایل قسمتی از تابع rescaleImage از سایت stack overflow قرض گرفته شد و تغییرات لازم جهت اعمال ویژگی‌های popbox به آن اعمال گردید. با استفاده از این تابع نسبت‌های طول و عرض تصویر به میزان ثابتی کوچک می‌شوند و تصویر کوچک شده با کیفیت بهتری نمایش داده خواهد شد.

موارد ذکر شده را از این آدرس می‌توانید دریافت نمائید.

نظرات خوانندگان

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۰۹/۱۷ ۲۰:۲۰:۰۰

A day without Learning is a lost day
وحید نصیری همچنان وحید نصیری هست.
تشکر فراوان از پشتکار شما و به اشتراک گذاشتن روزانه ی دانسته هایتان
درود

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۱۷ ۲۱:۲۱:۰۰

با سپاس از لطف شما
موفق باشید

نویسنده: نیما
تاریخ: ۱۳۸۷/۰۹/۱۹ ۱۰:۲۹:۰۰

سلام استاد نصیری
با تشکر از زحمات شما
به یه موردی که برخوردم اینه که اگر سرعت پایین باشه و عکسها زودتر از کدهای جاوا اسکریپت دانلود شده باشن این افکت عمل نمیکنه والبته من این مشکل رو قبلا با ایجکس هم داشتم.اگر شما به این سایت مراجعه کنین http://converter.telerik.com/ چنانچه کل صفحه بطور کامل لود نشده باشد با زدن دکمه به یک خطای هندل نشده برخورد میکنیم که بدلیل عدم دانلود کامل کدهای جاوا اسکریپتی است.چکار باید کرد که این مشکلت رخ ندهد؟امیدوارم منظورم رو درست بیان کرده باشم.
موفق و موید در پناهحق باشید

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۱۹ ۱۰:۵۵:۰۰

سلام
- در این مثال، خواص مربوط به popbox در document.ready مربوط به جی کوئری ست می‌شود، یعنی قبل از اینکه عکس‌ها load شوند تنظیمات ما اعمال شده.
- اسکریپت مربوط به popbox را من فشرده نکردم. با فشرده سازی آن حجم آن نصف می‌شود که در مورد سرعت load شدن آن خیلی مؤثر است. برای این منظور می‌تونید به آدرس زیر مراجعه کنید:
http://www.codeproject.com/KB/cs/jscompress.aspx

نویسنده: نیما
تاریخ: ۱۳۸۷/۰۹/۲۰ ۲۲:۳۸:۰۰

سلام استاد نصیری
نظرتون راجع به UpdatePanelچیه؟میشه ازش استفاده کرد در اون زمینه ای که بالا پرسیدم؟یعنی کل قسمتی که عکس داره رو بندازیم تو یه اپدیت پنل و وقتی کاملا لود شد ازش استفاده کنیم.آیا با کدهای جاوا تداخل ایجاد نمیکنه؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۲۱ ۰۲:۳۰:۰۰

سلام

استفاده از update panel برای اینکار مقرون به صرفه نیست. شما از یک طرف از سرعت پائین خط شکایت می‌کنید و از طرف دیگر می‌خواهید از حجم بالای اسکریپت‌های اجکس هم استفاده کنید که فقط به کاربر نشان دهید این تصویر در حال load شدن است؛ کمی صبر کنید الان نمایش داده می‌شود. مثلا چیزی شبیه به این:
http://www.codeproject.com/KB/ajax/DelayedContentLoading.aspx

- راه حل‌های زیادی برای حل این سربار هست. برای مثال در یک CMS که خودم نوشتم، تصاویری را که یوزر آپلود می‌کند، به صورت خودکار تبدیل به thumbnail می‌کنم (System.Drawing.Image.GetThumbnailImage). یعنی دو نسخه نگهداری می‌شود. یکی کوچک و یکی در اندازه اصلی. نسخه کوچک در صفحه اصلی گالری تصاویر نمایش داده می‌شود. پس از کلیک کاربر، با همین پاپ باکس یا یک نمونه دیگر که استفاده کردم به نام light box ، تصویر در اندازه اصلی نمایش داده می‌شود. به این صورت دیگر من نگران نحوه نمایش اولیه نخواهم بود و نمایش اولیه اسکریپتی نیست.
سایت لایت باکس است:
/http://www.huddletogether.com/projects/lightbox2
استفاده از آن هم در ASP.Net ساده است. اسکریپت‌های آن به هدر اضافه می‌شوند. سپس به تگ‌های یک کنترل ایمیج یا هایپرلینک که مثلا در یک DataList قرارگرفته‌اند خواص آن اضافه می‌شود. به این صورت می‌شود یک گالری پویا را ایجاد کرد که با دیتابیس هم کار کند.

- راه دیگر (اگر نمی‌خواهید thumbnail درست کنید) این است که از ابتدا اندازه تصویری را که قرار است نمایش داده شود مشخص کنید (طول و عرض کوچک). البته اگر تصویر شما همانطور که در این مقاله ذکر شد از نوع قرار گرفته شده داخل متن نیست و کاملا کنترل شده مثلا توسط یک دیتالیست رندر می‌شود، به این صورت اندازه ابتدایی کوچک خواهد بود و پس از لود شدن پاپ باکس، با کلیک کاربر روی عکس، اندازه اصلی را خواهید دید. (اسکریپتی که برای پاپ باکس نوشتم اینکار را به صورت خودکار انجام می‌دهد ولی خوب اسکریپت باید لود شده باشد که به طور قطع قبل از لود شدن تصویر اینکار تمام شده)

- اگر تصویر شما مخلوط داخل متن است و کنترلی روی آن ندارید باید با regular expressions آنها را پیدا کنید و سپس خواص مورد نظر خودتان را به تگ‌های تصویر اضافه کنید. (روش سمت سرور قبل از نمایش تصاویر)

عنوان: **تاریخ شمسی برای blogger !**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۳:۴۰:۱۴
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

تاریخ میلادی بلاگر واقعا روی اعصاب بود! این مشکل با استفاده از jQuery به صورت زیر قابل حل است.

تاریخ انگلیسی بلاگر به صورت زیر است:
البته در قسمت تنظیمات تاریخ بلاگ ، فرمت را به این صورت انتخاب کردم تا بدون مشکل تبدیل شود.

```
<h2 class='date-header'>2008/12/17</h2>
```

یعنی ما باید متن هرچی heading شروع شده با h2 و دارای کلاس date-header را پیدا کنیم و بعد معادل فارسی آن‌را جایگزین کنیم.
این‌کار را با استفاده از jQuery به صورت زیر می‌توان انجام داد:

```
<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.2.6/jquery.min.js'
type='text/javascript'></script>
<script src='http://vahid.nasiri.googlepages.com/farsidate.js' type='text/javascript'></script>
<script type='text/javascript'>
$(document).ready(function() {
$("h2.date-header").each(function() {
var obj = $(this);
obj.html(ToPersianDate(new Date(obj.text())));
});

$("a[title='comment permalink']").each(function(){
 var obj = $(this);
 obj.html(ToPersianDateLong(obj.text()));
});

$("a.post-count-link").each(function() {
 var obj = $(this);
 obj.html(getBloggerPMonthNames(obj.text()));
});

$("a.post-count-link").each(function() {
  var obj = $(this);
  obj.html(getBloggerPYear(obj.text()));
});
});
</script>
```

قسمت ویرایش html چیدمان وبلاگ را باید ویرایش و چند سطر بالا را به آن اضافه کرد (بعد از title صفحه).

پ.ن.
Farsidate.js برای تبدیل تاریخ میلادی به شمسی جاوا اسکریپتی از اینجا قرض گرفته شد.

به روز رسانی ،
نام ماه و سال سمت راست صفحه هم فارسی شد.
تاریخ کامنت‌ها هم فارسی شد.

نظرات خوانندگان

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۰۹/۲۸ ۱۴:۲۸:۰۰

برای قسمت آرشیوش کاری نمیشه کرد آخه! اشکال اینجا هست. من هم از چنین چیزی استفاده می کنم که تیم بلاگر فارسی درستش کرده بود.اما خوب خیلی دلم می خواهد که آرشیو هم خورشیدی بشه...

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۲۸ ۱۶:۳۷:۰۰

چرا، اتفاقا بعد از اینکه این‌کار را انجام دادم یادم افتاد قسمت سمت راست فراموش شده :)
این مورد را هم درست کردم. مطلب ارسالی هم به روز شد.

نویسنده: احمد
تاریخ: ۱۳۸۷/۰۹/۲۹ ۰۰:۳۰:۰۰

سلام،
کار بسیار عالی انجام دادید، تا حالا نمونه ساخته شده با jquery را ندیده بودم.

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۲:۴۹:۰۰

آقا خیلی حال دادی! الان اینو گذاشتم روی وبلاگم. مبدل این با اون قلبیه فرقی نداره. فقط چیکار کنم که ساعت رو باسه تاریخ کامنت نشون بده و اسم قدیمی روز رو برای پست ها نشون بده؟

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۳:۱۴:۰۰

ببخشید من انقدر اذیت می کنم، اون اسکریپتی که قبلا استفاده کرده بودم رو آردسش رو برات گذاشتم. کدوم یکی از این دو تا سریع تر اجرا می شن؟ چون تو این یکی jquery.min.js نداره، حجمش کمتره! اما خوب آرشیو رو خورشیدی نمی کنه! فعلا که این اسکریپت شما بهتره اما اون مشکل ساعت و اسم قدیمی روز رو دارم و اینکه نمی دونم چقدر تو سرعت لود صفحه ها تاثیر داره چون متاسفانه حجم صفحات وبلاگم تقریبا زیاده. بازهم متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۳:۴۳:۰۰

- ساعت کامنت‌ها رو درست کردم. اسکریپت farsidate.js و متن ارسالی فوق به روز شد.
- در مورد حجم، چون این اسکریپت‌های لینک داده شده توسط مرورگر کش خواهند شد، فقط بار اول 50 کیلوبایت سربار دارد. در دفعه بعد بازدید، از کش خوانده می‌شود.

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۳:۵۹:۰۰

50 کیلوبایت خیلی زیاده رو قالب وبلاگ من چون خودش سنگینه! با این روشی که اینجا توضیح داده نمی شه نمایش آرشیو رو هم خورشیدی کرد:

http://bahramshahrfa.blogspot.com/2007/12/persian-date-for-blogger.html

نویسنده: وحید نصیری

تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۴:۱۱:۰۰

این اسکریپت رو تبدیل به یک اسکریپت گریس مانکی کردم که امروز پست می‌کنم. به این صورت نیازی به تغییر در سایت نخواهد بود. خیلی ساده‌تر است و سرباری برای سایت ندارد.

نویسنده: باغ بان باشی
تاریخ: ۱۳۸۷/۰۹/۳۰ ۰۶:۵۹:۰۰

آیا با این کدها فرمت آرشیو به تاریخ شمسی تبدیل میشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۳۰ ۰۹:۰۰:۰۰

بله. سمت راست صفحه مشخص است.

نویسنده: باغ بان باشی
تاریخ: ۱۳۸۷/۰۹/۳۰ ۱۵:۱۲:۰۰

آقای نصیری شرمنده . میخواستم بدونم اگر کدهای بخش دوم رو به وبلاگ اضافه کنیم ، آرشیو به خورشیدی تبدیل میشه؟ من این کار رو کردم و تغییریی نکرد

من متوجه این بخش از توضیحاتتون نمیشم :ممکن بیشتر توضیح بدهید؟

"یعنی ما باید متن هرچی heading شروع شده با h2 و دارای کلاس date-header را پیدا کنیم و بعد معادل فارسی آن‌را جایگزین کنیم."

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۳۰ ۱۶:۱۸:۰۰

- لطفا آدرس بلاگی را که پس از اعمال کدهای فوق تغییری نکرده اینجا جهت بررسی ذکر بفرمائید.
این فارسی سازی با قالب استاندارد بلاگر (که به نظر در اکثر موارد همین است) سازگار است.

- اون مطلب رو درباره توضیح در مورد برنامه نویسی آن نوشتم و هنگام استفاده کاربردی ندارد و صرفا کپی کردن اطلاعات اسکریپت قسمت دوم مطلب بالا در header صفحه کفایت می‌کند.

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۰۹/۳۰ ۲۰:۵۷:۰۰

من فکر کنم مشکل مجتبی در $("h2.date-header").each(function() باشه... مثلا توی قالب من مطلب ها توی Span هستند و نویسنده از h2 استفاده نکرده!!!!! هرچند خیلی کاره اشتباهی است اما خوب من مجبور شدم موقتا span را توی اسکریپپ جایگزین کنم.

مجتبی جا داره اگه این اسکریپت اندک مشکلاتی که داره حل شه بره توی بلاگر فارسی اون هم به عنوان بهترین مطلب بلاگر فارسی.

نویسنده: کورش اسلام زاده
تاریخ: ۱۳۸۷/۰۹/۳۰ ۲۱:۳۴:۰۰

سلام
دست شما درد نکند و کارتان آفرین دارد.

یکی دو سالی است که از تاریخ فارسی در بلاگر استفاده میکنم. علت هم روشن است چون بازدیدکنندگان ساکن ایران درک روشنی از تاریخ میلادی ندارند. از طرف دیگر مشکل دیگری در اینصورت به وجود »یاید . متاسفانه واقعیت اینست و کراراً مشاهده شده است که ساکنین خارج از کشور تاریخ شمسی را گم میککنند و درک درستی از زمان به تاریخ شمسی ندارند.

راه حل من دستکاری در قالب بلاگر و قرار دادن هر دو در کنار هم بوده است.

نمیدانم! اگر برایتان امکان پذیر باشد که ورژن دیگری را هم تهیه کنید که هر دو را داشته باشد بسیار مطلوب است و جهت به کار گیری کار خوب شما من مجبور به سر و کله زدن با قالب بلاگر نمیشوم.

کمان میکنم موضوعی که گفتم یعنی داشتن هر دو تاریخ شمسی و میلادی در کنار هم یک نیاز واقعی است که تقریباً در اینباره بحثی نشده است ئ وراه حل آسانی هم برای آن نبوده است
بازهم ممنون و موفق باشید

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۳۰ ۲۳:۳۳:۰۰

سلام
- میلادی را هم به روش فارسی اضافه کردم :)

- ضمنا اسکریپت گریس مانکی هم به روز شد تا با آن تداخل نکند.

نویسنده: باغ بان باشی
تاریخ: ۱۳۸۷/۰۹/۳۰ ۲۳:۵۸:۰۰

آقای نصیری سلام . من تغییرات رو در وبلاگم اعمال کردم و جواب گرفتم . فقط یک نکته اینکه میشه توی این کد تغییری اعمال کنید که فقط آرشیو رو به شمسی تبدیل کنه و تاریخ پستها تبدیل نشه؟ اینطوری بهتره

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۱ ۰۰:۴۹:۰۰

برای این‌کار کافی است چهار سطر مربوط به h2.date-header و چهار سطر a[title='comment permalink'] را حذف کنید.

نویسنده: افشار
تاریخ: ۱۳۸۷/۱۰/۰۱ ۰۹:۰۰:۰۰

سلام،
ممنون از این مطلب خوب. برای من همه قسمت‌ها را به جز commentها درست کرد.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۱ ۰۹:۱۴:۰۰

سلام
کامنت‌های بلاگ شما چون در خارج از صفحه وبلاگ صورت می‌گیرد اسکریپت‌های فارسی سازی روی آن اثری ندارند. (فکر کنم این مورد در تنظیمات وبلاگ قابل تنظیم است که کامنت‌ها کجا باز شوند.)
اینجا توضیح داده شده:
/http://web3b.wordpress.com/2008/09/07/blogger-admin
قسمت "کادر نظرات در وبلاگهای بلاگر در ذیل مطالب، مانند وردپرس"

نویسنده: افشار

تاریخ: ۱۳۸۷/۱۰/۰۲ ۰۸:۴۸:۰۰

سپاس از توجه شما،
به نظر می‌آید که یک باگ کوچولو هم در محاسبه تاریخ آرشیوها وجود دارد. من دیروز (۱ دی) یک مطلب به بلاگم پست کردم که خود بلاگ هم تاریخ آن را ۱ دی نمایش می‌دهد ولی این مطلب در قسمت آرشیو وبلاگ در مدخل آذرماه آمده است نه دی ماه.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۲ ۰۹:۰۶:۰۰

بله. این همان موردی است که دوستان در بالا هم به آن اشاره کردند. در نحوه رندر کردن گوگل تغییری حاصل نشده. بنابراین تا آخر برج میلادی در همین آذر باقی خواهد ماند.
نظر شما چیست؟
1- ماه به صورت آذر-دی نمایش داده شود
2- ماه اگر مربوط به سال قبل شمسی بود ، عدد سال هم به آن اضافه شود. (مثلا اسفند سال قبل که دوستان به آن اشاره کردند)

یا پیشنهاد بهتری دارید؟

نویسنده: افشار
تاریخ: ۱۳۸۷/۱۰/۰۳ ۰۸:۱۷:۰۰

به نظر من مورد اول بهتر است چون اگر سال را هم به آن اضافه کنید زیادی شلوغ می‌شود.

نویسنده: فاروق
تاریخ: ۱۳۸۷/۱۲/۲۵ ۱۳:۵۴:۰۰

این کار شما برای کسانی که از template کلاسیک استفاده نمیکنن کار میکنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۲۵ ۱۴:۴۲:۰۰

قالب‌های غیر رسمی هم اگر سلکتورهای بلاگر را حفظ کرده باشند، بله کار می‌کند.
برای مثال اگر یک قالب غیر رسمی در تگ h2 کلاس استاندارد date-header را حفظ کرده، بله کار می‌کند.

در کل اگر می‌خواهید تست کنید که این روش روی سایت شما هم جواب میده یا نه، اسکریپت گریس مانکی زیر را نصب کنید:
http://www.dotnettips.info/2008/12/blog-post_19.html

نویسنده: hamed
تاریخ: ۱۳۸۸/۰۴/۰۸ ۲۳:۲۲:۴۱

سلام من این رو بین head و /head قرار دادم ولی کار نکرد.
آدرس من :
mortalfun.blogspot.com

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۴/۰۹ ۰۱:۰۴:۰۳

سلام،
اگر کد را مستقیما کپی کردید یک سری شماره خط اضافه شده که باید پاک شود.
.1
.2

.3

...

.25

این‌ها اضافی است و نباید باشد.

نویسنده: hamed
تاریخ: ۰۹:۲۳:۳۲ ۱۳۸۸/۰۴/۱۰

مورد قبلی درست شد(مرسی)
بعد این که نمیشه کاری کرد که به جای این که مثلا می رو با خرداد یکی بگیره تو آرشیو ماه های صحیح فارسی نوشت
اطلا یه کار دیکه آیا بایگانی کاملا فارسی فارق از بایگانی بلاگر وجود داره؟

نویسنده: وحید نصیری
تاریخ: ۱۰:۰۰:۵۵ ۱۳۸۸/۰۴/۱۰

این باگ بایگانی هنوز برطرف نشده ...

نویسنده: کلاغ سفید
تاریخ: ۰۳:۳۳:۱۶ ۱۳۸۸/۰۴/۱۹

از زحمتی که کشیده اید ممنونم.

نویسنده: جواد
تاریخ: ۱۰:۲۸:۲۵ ۱۳۸۸/۰۵/۲۲

آقای نصیری! ممنون!

نویسنده: وحید نصیری
تاریخ: ۱۷:۵۳:۱۲ ۱۳۸۸/۰۶/۱۹

سلام،
قسمت "اگر کد را مستقیما کپی کردید یک سری شماره خط اضافه شده .... اضافی است" را مطالعه بفرمائید (2 پاسخ بالاتر بنده).

نویسنده: PanaAli
تاریخ: ۲۳:۵۶:۰۶ ۱۳۸۸/۰۷/۰۳

سلام وحید جان . آقا کدت عالی بود . فقط این آخرین نظرات رو از کجا بذارم ؟ widget خودش کار نمی کنه .
و اینکه چون بلاگ شیر نیست نمی شه از فید استفاده کرد .

نویسنده: وحید نصیری
تاریخ: ۰۰:۱۷:۲۶ ۱۳۸۸/۰۷/۰۴

سلام
راه استاندارد آن‌را اینجا توضیح داده‌ام:
http://www.dotnettips.info/2009/06/blog-post.html
نیاز به فید نظرات سایت دارد و اگر share نیست با این راه نمی‌شود.
راه دیگر هم اینجا توضیح داده شده است:
http://www.google.com/support/blogger/bin/answer.py?hl=en&answer=42522

نویسنده: رز سیاه
تاریخ: ۱۳۸۸/۰۷/۰۵ ۱۷:۴۰:۱۴

سلام آقای نصیری
واقعاً خسته نباشید
خیلی عالی و کاربردیه

ببخشید
فقط من یه مشکلی داشتم
برای من زمان ثبت پست ها هنوز به صورت میلادی هستش
یعنی اشکال کار کجاست ؟؟؟

البته این 2 نکته رو اضافه کنم :
1 - من از قالب فارسی سازی شده خود بلاگر فارسی استفاده می کنم
2 - من کد ها رو درست در بخش انتهایی کد های قالب و قبل از html آخر وارد کردم

آدرس من اینه :
/http://persianblackrose.blogspot.com

ممکنه مشکل از این 2 تا باشه ؟؟؟

.
.
.

و یک سوال هم از خدمتتون داشتم
آیا این امکان وجود داره که بشه قسمتی به ساید بار اضافه کرد که در اون بازدید کنندگاه امکان گذاشتن کامنت رو داشته باشند ؟
تقریباً یه چیز تو مایه های بخش کامنت 360 یاهو ؟؟؟

.
.
.

پیشاپیش از لطفتون ممنونم

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۰۵ ۲۲:۰۷:۲۷

- در قالب شما ابتدای صفحه یک سطر طولانی هست ... اون همون کل فایل jQuery است که به این صورت لحاظ شده و علت آن‌را هم نمی‌دانم و لزومی هم نداشته و صفحه را بی‌خود سنگین کرده.
- شما می‌تونید اسکریپت‌ها را قبل از بسته شدن تگ body قرار بدید.
- ضمنا کار ارائه شده در این مقاله، با قالب‌های جدید تهیه شده سازگار نیست. خبری از date-header و غیره در آن نیست و منصف باشید که فرصت این وجود ندارد که به ازای تک تک قالب‌ها یا تک تک افراد بلاگر این موارد بخواهند اصلاح شوند یا تغییر کنند.
در کل تهیه کنندگان قالب‌ها می‌تونند این مورد را لحاظ کنند یا خیر.

عنوان: **اسکریپت گریس مانکی برای تاریخ فارسی بلاگر و ایمیل یاهو**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۴:۲۴:۵۱
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

پیرو مطلب قبلی که ارسال کرده بودم، خوب اگر بلاگ‌داری این موارد را اعمال کند، تاریخ شمسی را به وبلاگ خودش اضافه کرده است. اما در مورد دیگران چطور؟ برای این منظور، اسکریپت مطلب ارسالی را به یک اسکریپت گریس مانکی تبدیل کردم که به‌عنوان افزونه‌ای در فایرفاکس نصب خواهد شد و تمامی وبلاگ‌های بلاگری را پوشش می‌دهد (حتی آنهایی که در آدرسشان بلاگر یا بلاگ اسپات نیست).

نحوه استفاده: (همانند روش عنوان شده در این مطلب)
گریس مانکی را نصب کنید
سپس به صفحه زیر مراجعه کرده و بر روی دکمه install در بالای صفحه کلیک نمائید.
Persian Date for Blogger

همچنین ستون تاریخ انگلیسی ایمیل‌های یاهو را هم به تاریخ شمسی تبدیل کردم که جزئی از این مجموعه است (با قالب کلاسیک ایمیل یاهو (نه این قالب جدید) درست کار می‌کند).

Date	Size
۱۳۸۷ آذر ۲۷	7KB
۱۳۸۷ آذر ۲۷	839KB
۱۳۸۷ آذر ۲۶	11KB
۱۳۸۷ آذر ۲۶	3KB

مزایای این روش:
بلاگ‌دار لازم نیست چیزی را نصب کند.
تمام بلاگ‌های مربوط به بلاگر (در دومین و خارج از دومین آن) تحت پوشش قرار می‌گیرند.
سریع است. (چون یکبار بر روی مرورگر شما نصب شده است و هیچ فایلی قرار نیست از اینترنت دریافت شود)

شایان ذکر است که گریس مانکی در مرورگرهای دیگر هم پشتیبانی می‌شود.

پ.ن.
اگر علاقمند به برنامه نویسی اسکریپت‌های گریس مانکی هستید، کتابچه‌ای رایگان در این باره را از این آدرس می‌توانید دریافت نمائید.

نظرات خوانندگان

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۰۹/۲۹ ۱۸:۰۹:۰۰

سلام، توی آرشیو یک مشکل وجود داره؛ تغییر ماه های میلادی با خورشیدی هم زمان نیست، این باعث شده بود که پستی که مثلا در اواخر مارچ پست شده بود در اسفند سال بعد قرار بگیره در صورتی که باید در اسفند سال قبل باشه! گریس مانکی جالبه اما جایگزین اون اسکریپت نمیشه خوب!

نویسنده: نیما
تاریخ: ۱۳۸۷/۰۹/۲۹ ۲۳:۵۷:۰۰

سلام جناب نصیری
من خیلی به جاوااسکریپت وارد نیستم اما همین کار شما رو برای تقویم گوگل انجام دادم اما دو تا مشکل داره:
-1 سال میلادی رو نمی فهمه و باید برای هر سال میلادی بصورت دستی این اسکریپت رو تغییر داد.
-2 فقط برای هفته ی جاری، هفته ی بعد و هفته ی قبل درست کار می کنه.

پیشنهاد می کنم یه نگاهی بهش بندازی و اصلاحش کنی. بدرد من یکی که خیلی می خوره.

http://nima.rasouli.org/2008/08/google-calendar-persian.html

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۳۰ ۰۱:۴۸:۰۰

سلام
و با تشکر از دوستان.

- سعی می‌کنم در هفته بعد این مشکل ماه اسفند سال 87 رو حل کنم :) البته بهتره فقط یک سال 86 به اون اضافه بشه تا مشخص باشه. چون عملا دستکاری رندرینگ گوگل شاید ممکن نباشه یا لزومی نداشته باشه.

- تقویم گوگل واقعا حرفه‌ای است. با تشکر از نیماخان. من قبلا نتونسته بودم این اسکریپت شما رو استفاده کنم که حالا مشخص شد دلیلش چی هست. در اسکریپت گریس مانکی بالا یک سری از فرمت‌ها رو دستی درست کردم تا با فرمت تاریخ استاندارد جاوا اسکریپت همخوانی پیدا کند. به این صورت با اکثر فرمت‌ها سازگاری پیدا کرد. یا یک سری از جاها با استفاده از isNan بد نیست تست بشه آیا خروجی معتبر هست یا نه. یا کلا تبدیل تاریخ درست انجام شده یا نه.

پروژه جدید تقویم گوگل هم ارزشش رو داره ;)

نویسنده: بابک
تاریخ: ۱۳۸۷/۰۹/۳۰ ۰۶:۴۶:۰۰

سلام جناب نصیری . درخواستی داشتم . شما چگونه فرمت آرشیو وبلاگتون رو به تاریخ شمسی تبدیل کردید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۹/۳۰ ۰۸:۵۹:۰۰

سلام
در این‌جا توضیح دادم:
http://www.dotnettips.info/2008/12/blogger.html

نویسنده: علی
تاریخ: ۱۳۸۹/۰۶/۱۶ ۰۱:۴۱:۴۵

مرسی آقای نصیری مفید بود!

عنوان: **جمع آوری آمار لینک‌های خروجی از سایت توسط Google analytics**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۲ ۱۶:۵۱:۳۷
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

چندی قبل مطلب کوتاهی را در مورد Google analytics نوشتم. در حین جستجو درباره‌ی jQuery در وب، به نحوه ردیابی لینک‌های خروجی از سایت توسط Google analytics برخوردم که نحوه پیاده سازی آن به صورت زیر است.
بدیهی است قبل از هر کاری باید اسکریپت مربوط به Google analytics را به انتهای صفحه و جایی که تگ body بسته می‌شود اضافه کنید (قابل دریافت درقسمت Add Website Profile . شماره این اسکریپت برای هر پروفایلی که ایجاد می‌کنید متفاوت است).
سپس:
الف) افزودن ارجاعی از کتابخانه jQuery به هدر صفحه که آن‌را در مطلب شمسی کردن تاریخ بلاگر ملاحظه کردید.
ب) افزودن چند سطر زیر به هدر صفحه

```
<script type="text/javascript">
$(document).ready(function() {
$("a").click(function() {
  var $a = $(this);
  var href = $a.attr("href");

  // see if the link is external
  if ( (href.match(/^http/)) && (! href.match(document.domain)) ) {

    // if so, register an event
    var category = "outgoing";
   var event = "click";
   var label = href;

   pageTracker._trackPageview('/outgoing/' + href);
   pageTracker._trackEvent(category, event, href);
 }
});
});
</script>
```

البته اگر قبلا اسکریپت شمسی کردن تاریخ بلاگر را اضافه کرده بودید فقط محتویات تابع document.ready را باید اضافه کنید (جهت مشاهده نمونه اعمال شده، روی صفحه جاری کلیک راست کنید و سورس صفحه را مشاهده نمائید).

توضیحاتی در مورد کد فوق:
این اسکریپت به روال رخ داد گردان onclick هر لینکی که به خارج از سایت ختم می‌شود (مثلا لینک به یک فایل یا یک سایت خارجی (خارج از سایت))، به صورت خودکار تابع trackPageview مربوط به Google analytics را اضافه می‌کند. این کار تاثیری در عملکرد سایت ندارد و کاربر چیزی را متوجه نخواهد شد، اما به این طریق لینک‌های خروجی در آمار Google analytics ظاهر می‌شوند (مطابق تصاویر زیر).

100.	/outgoing/http://afsharm.blogspot.com/2008/05/how-to-confi	1
101.	/outgoing/http://content.zdnet.com/2346-12554_22-255234-1	1
102.	/outgoing/http://feeds.feedburner.com/vahidnasiri	1
103.	/outgoing/http://weblogs.asp.net/pwilson/archive/2005/01/18	1
104.	/outgoing/http://www.dumpanalysis.org/blog/	1

از این پس آمار تمام لینک‌های خروجی از سایت ، متمایز شده با outgoing ، جمع آوری و نمایش داده خواهند شد.

امکانات بیشتری مانند event tracking نیز قرار است به Google analytics اضافه شود که هنوز در مرحله آزمایشی است و بر روی تمامی اکانت‌ها فعال نشده است.

نظرات خوانندگان

نویسنده: Armin
تاریخ: ۲۰:۴۰:۰۰ ۱۳۸۷/۱۰/۰۲

سلام وحید جان
ممنون بسیار جالب بود
امروز اینجا پیداتون کردم و خیلی از مطالب استفاده بردم
پاینده باشید

نویسنده: PAYAM
تاریخ: ۲۳:۵۲:۰۰ ۱۳۸۷/۱۰/۰۲

سلام وحید خان.

راه انداختن وبلاگ کار بسیار پسندیده‌ای بود.
خوشحالم که جای دیگه‌ای هم هست که بتونم از حضورتون استفاده کنم.

ارداتمند!

نویسنده: وحید نصیری
تاریخ: ۰۰:۰۷:۰۰ ۱۳۸۷/۱۰/۰۳

شما لطف دارید.
موفق باشید

نویسنده: Anonymous
تاریخ: ۰۵:۴۰:۰۰ ۱۳۸۷/۱۰/۰۳

سلام آقای نصیری.
قصد دارم جاوااسکریپت یادبگیرم.
میخواستم بدونم یک هلپ فایل برایش موجود هست؟ که بشه مثل MSDN تقریبا؟
مرسی

نویسنده: وحید نصیری
تاریخ: ۱۰:۵۶:۰۰ ۱۳۸۷/۱۰/۰۳

سلام
عبارت زیر را در گوگل جستجو کنید
JavaScript complete reference rapidshare

عنوان: **چگونه یک اسکریپت گریس مانکی بنویسیم؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۳ ۱۷:۳۲:۵۸
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

گریس مانکی
یکی از افزونه‌های فایرفاکس است که توسط آن می‌توان اسکریپت‌هایی را بر روی صفحات وب در حال مشاهده، جهت تغییر آن‌ها اجرا کرد. این نوع تغییرات بیشتر در جهت بالا بردن کارآیی یا خوانایی سایت‌ها صورت می‌گیرد. مثلا بررسی وجود لینک‌های ارائه شده در یک صفحه وب (قبل از اینکه به آن‌ها رجوع کنیم، مشخص شود که آیا وجود دارند یا خیر) و هزاران مثال شبیه به این که در سایت اسکریپت‌های آن قابل دریافت هستند.
گریس مانکی به خودی خود کار خاصی را انجام نمی‌دهد و فقط میزبان اجرایی اسکریپت‌هایی است که برای آن تهیه شده اند. به این اسکریپت‌ها user scripts گفته می‌شود و جهت تهیه آن‌ها از زبان جاوا اسکریپت استفاده می‌گردد.
مطابق اصول نامگذاری آن، فایل این اسکریپت‌ها حتما باید به user.js. ختم شود تا توسط افزونه گریس مانکی قابل شناسایی باشد.

اسکریپت سلام دنیای گریس مانکی!

```
// ==UserScript==
// @name Hello World
// @namespace http://diveintogreasemonkey.org/download/
// @description example script to alert "Hello world!" on every page
// @include *
// @exclude http://diveintogreasemonkey.org/*
// @exclude http://www.diveintogreasemonkey.org/*
// ==/UserScript==
alert('Hello world!');
```

فرمت کلی و آناتومی یک اسکریپت گریس مانکی مطابق چند سطر فوق است. در ابتدا نامی را که برای اسکریپت مشخص کرده‌اید، ذکر خواهید نمود. سپس فضای نام آن مشخص می‌گردد. این فضای نام یک آدرس وب خواهد بود (مثلا سایت شخصی شما) . به این طریق گریس مانکی می‌تواند اسکریپت‌های هم نام را بر اساس این فضاهای نام مختلف مدیریت کند. سپس توضیحات مربوط به اسکریپت ارائه می‌شود. در قسمت include مشخص خواهید کرد که این اسکریپت بر روی چه سایت‌ها و آدرس‌هایی اجرا شود و در قسمت exclude سایت‌های صرفنظر شونده را تعیین خواهید نمود. در مثال فوق، اسکریپت بر روی تمامی سایت‌ها اجرا خواهد شد، منهای دو سایت و زیر سایت‌هایی که در قسمت exclude مشخص شده‌اند.
در پایان مثلا نام آن‌را hello.user.js خواهیم گذاشت (همانطور که ذکر شد قسمت user.js آن باید رعایت شود). برای نصب آن فقط کافی است این فایل را به درون پنجره فایر فاکس کشیده و رها کنیم (drag & drop).

استفاده از jQuery در اسکریپت‌های گریس مانکی

عمده کاربردهای اسکریپت‌های گریس مانکی در جهت اعمال تغییرات بر روی document object model صفحه (DOM) هستند.
کتابخانه jQuery اساسا برای این منظور تهیه و بهینه سازی شده است.

مثال:
فرض کنید قصد داریم به نتایج خروجی جستجوی گوگل، fav icon سایت‌های یافت شده را اضافه کنیم (در کنار هر لینک، آیکون سایت مربوطه را نمایش دهیم). گوگل این‌کار را انجام نداده است. اما ما علاقمند هستیم که این قابلیت را اضافه کنیم!

```
// ==UserScript==
// @name            Google FavIcon
// @namespace       http://userscripts.org
// @description     Shows favicon for Google searches results
// @include         http://*.google.*/search?*
```

```
// ==/UserScript==

loadJquery();

$(document).ready(function(){
   $("h3.r > a.l").each(function(){
       var $a = $(this);
       var href = $a.attr("href");
       var domain = href.replace(/<\S[^><]*>/g, "").split('/')[2];
       var image = '<img src="http://' + domain + '/favicon.ico"  style="border:0;padding-right:4px;"
/>';
       //GM_log(">image:> " + image);
       $(this).prepend(image);
   });

});
```

نحوه پیاده سازی تابع loadJquery را در سورس مربوطه می‌توانید مشاهده نمائید.
دریافت سورس این مثال

در اینجا ابتدا لینک‌های حاصل از جستجو پیدا می‌شوند. سپس نام دومین مربوطه استخراج می‌گردد (با استفاده از regular expressions) و در ادامه از این نام دومین یک آدرس استاندارد http://domain/favico.ico ساخته شده و از آن یک تگ img درست می‌شود. در آخر این تگ به قبل از لینک‌های گوگل اضافه می‌شود.

شاید سؤال بپرسید از کجا مشخص شد که باید به دنبال h3.r > a.l گشت؟ به تصویر زیر دقت نمائید (نمایی است از یکی از توانایی‌های افزونه fireBug فایرفاکس).

هنگامیکه صفحه بارگذاری شد، بر روی آیکون سوسک موجود در status bar فایرفاکس کلیک کنید تا FireBug ظاهر شود (البته من اینجا سوسک دیدم شاید موجود دیگری باشد :))
سپس بر روی دکمه inspect در نوار ابزار آن کلیک کنید. در همین حال اشاره‌گر ماوس را به یکی از لینک‌های نتیجه جستجوی گوگل نزدیک کنید. بلافاصله تگ‌ها و کلاس‌های مورد استفاده آن به شکل زیبایی ظاهر خواهند شد. به این صورت صرفه جویی قابل ملاحظه‌ای در وقت صورت خواهد گرفت.

نتیجه اجرای اسکریپت فوق (پس از نصب) به صورت زیر است:

نکته: نحوه دیباگ کردن اسکریپت‌های گریس مانکی

اگر نیاز به مشاهده مقدار متغیرها در لحظه اجرای اسکریپت داشتید، یکی از راه‌حل‌های موجود استفاده از تابع GM_log مربوط به API گریس مانکی است که خروجی آن‌را در قسمت messages مربوط به error console فایرفاکس می‌توان دید.

Error Console
All
Errors
Warnings
Messages
Clear
Code:
http://userscripts.org/Google FavIcon: >domain:> www.experts-exchange.com
http://userscripts.org/Google FavIcon: >domain:> www.experts-exchange.com
http://userscripts.org/Google FavIcon: >domain:> www.eggheadcafe.com
http://userscripts.org/Google FavIcon: >domain:> desk.surpasshosting.com
http://userscripts.org/Google FavIcon: >domain:> lists.evolt.org
http://userscripts.org/Google FavIcon: >domain:> www.longtailvideo.com

نظرات خوانندگان

نویسنده: web3b
تاریخ: ۱۳۸۷/۱۰/۰۷ ۲۳:۳۲:۰۰

تشکر از وقتی که واسه مطالب میگذارید.
گویا اجرای استایلها توسط استایلیش بهتر باشه چون همزمان با لود صفحات اعمال میشوند. اما برای کارکردهایی غیر از تغییر استایل گریزمانکی حرف نداره! ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۸ ۰۱:۴۰:۰۰

با تشکر.
البته جسارتا تا جایی که دقت کردم تلفظ آن همان گریس است و گریز نیست.
http://upload.wikimedia.org/wikipedia/commons/a/a3/En-us-grease.ogg
http://en.wiktionary.org/wiki/grease

عنوان: **ویدیوهای رایگان آموزش jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۶ ۱۷:۱۰:۳۶
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

آموزش مقدماتی jQuery

روز 1 : مشاهده سایت اصلی ، دریافت
Day 1: Downloading the Library

روز 2 : مشاهده سایت اصلی ، دریافت
Day 2: Fade, Slide, and Show Methods

روز 3 : مشاهده سایت اصلی ، دریافت
Day 3: The Animate Method

روز 4 : مشاهده سایت اصلی ، دریافت
Day 4: Advanced Selectors

روز 5 : مشاهده سایت اصلی ، دریافت
Day 5: Creating and Removing Elements

روز 6 : مشاهده سایت اصلی ، دریافت
Day 6: The toggle() and toggleClass() Methods

روز 7 : مشاهده سایت اصلی ، دریافت
Day 7: The hover() Methods

روز 8 : مشاهده سایت اصلی ، دریافت
Day 8: User Request - Image Slides

روز 9 : مشاهده سایت اصلی ، دریافت
Day 9: Resizing Text

روز 10 : مشاهده سایت اصلی ، دریافت
Day 10: Intro to AJAX: Using the Load Method

روز 11 : مشاهده سایت اصلی ، دریافت
Day 11: Fun Image Hovering

روز 12 : مشاهده سایت اصلی ، دریافت
Day 12: Advanced Tooltips: Part 1

روز 13 : مشاهده سایت اصلی ، دریافت

Day 13: Submitting Information to a Database Asynchronously

نظرات خوانندگان

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۰/۲۰ ۰۹:۳۷:۰۰

باتشکر.
دو قسمت دیگرش هم آماده شده.

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۰/۲۰ ۱۴:۰۳:۰۰

لینک آخر کار نمیکنه

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۰ ۱۵:۴۹:۰۰

برای تمام ویدیوها فوق mirror تهیه کرده‌ام:
http://www.dotnettips.info/2008/12/mirror.html

و mirror برای ویدیوهای جدید این مجموعه:
Day 14: Implementing Your First Plugin
http://rapidshare.com/files/181363327/v3bg115310.flv

Day 15: Building a jQuery Style-Switcher: Part 1
http://rapidshare.com/files/181363189/n62ldrzuyo.flv

Day 15: Building a jQuery Style-Switcher: Part 2
http://rapidshare.com/files/181363345/qkrwv8jzsf.flv

نویسنده: MicroDump
تاریخ: ۱۳۸۷/۱۰/۲۵ ۱۵:۵۵:۰۰

سلام
بابت لینک ها دستت درد نکنه.
اگه قسمت جدیدی هم داده اون ها رو هم میرور کنی ممنون می شم

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۵ ۱۷:۰۶:۰۰

سلام،
به نظر دوره مقدماتی jQuery همین ها است. الان دوره جدید PHP مقدماتی رو شروع کرده.

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۲/۱۵ ۰۷:۴۰:۰۰

خیلی متشکرم
فیلم ها خیلی مفید هستند
ولی از فیلم شماره 7 به بعد رو نمی تونم ببینم.فقط صدا شنیده می شه
با VLC و GOMPlayer هم تست کردم

Vista Codec Pack هم روی سیستم نصبه

ممکنه بگین برای دیدن فیلم چه کدکی لازمه؟

متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۵ ۰۹:۳۷:۰۰

سلام،
لطفا از برنامه‌ی رایگان زیر استفاده کنید:
http://cdn.pandora.tv/KMP/Download/beta/The_KMPlayer_1434.exe
http://www.kmplayer.com/forums/showthread.php?t=12142

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۲/۱۶ ۰۸:۰۱:۰۰

خیلی متشکرم
با KMPlayer درست پخش می شه

ولی لینک روز 12 و 13 خرابه.لطفا درست کنید.
متشکرم D:

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۶ ۱۰:۳۰:۰۰

mirror برای ویدیوهای فوق در چند مطلب بالاتر ارسال شده. (کامنت سوم)

نویسنده: علی
تاریخ: ۱۳۸۸/۰۶/۱۳ ۱۰:۳۶:۵۸

برای دانلود آموزشهای تصویری jquery به زبان فارسی به سایت www.nano3oft.com مراجعه کنید.

عنوان: **نمایش خودکار آیکون لینک‌های سایت‌های خارجی با استفاده از jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۰ ۰۱:۱۰:۰۱
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

اگر دقت کرده باشید در کنار هر لینک خارجی (هر لینک ختم شده به خارج از سایت) در این وبلاگ، آیکون آن سایت هم به صورت خودکار نمایش داده می‌شود. از نظر من جالب بوده (خصوصا جهت مشخص کردن وجود یک لینک در میان انبوهی متن، یا برای نمایش لیستی از لینک‌ها)، نظر شما رو نمی‌دونم!

این‌کار در ادامه‌ی مطلبی است که در مورد نوشتن اسکریپت‌های گریس مانکی چندی قبل منتشر کردم. البته بهبود یافته‌ی آن است. برای ایجاد یک چنین قابلیتی در سایت فقط کافی است چند سطر اسکریپت زیر را به هدر سایت خود اضافه کنید.

```
<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.2.6/jquery.min.js' type='text/javascript' />
<script type="text/javascript">
   $(document).ready(function(){
       $("a").each(function(){
           var $a = $(this);
           var href = $a.attr("href");
           // see if the link is external
           if (href && href.match(/^http/))
               if (!href.match(document.domain)) {
                   var domain = href.replace(/<\S[^><]*>/g, "").split('/')[2];
                   var image = '<img src="http://' + domain +
                   '/favicon.ico" width="0" ' +
                   '
onload="this.width=16;this.height=16;this.style.paddingLeft=\'3px\';this.style.paddingRight=\'1px\';" '
+
                   ' style="border:0" ' +
                   ' onerror="this.src=\'http://vahid.nasiri.googlepages.com/weblink.gif\';" />';
                   $(this).prepend(image);
               }
       });
   });
</script>
```

توضیحاتی در مورد جزئیات برنامه نویسی آن:

با استفاده از jQuery ، زمانیکه document ما ready شد (document.ready)، تمام لینک‌های موجود در صفحه را پیدا می‌کنیم (یک به یک). سپس مقدار href آن‌ها را می‌خوانیم (ببینید با jQuery انجام این نوع کارها چقدر ساده شده است). اکنون روی لینک دریافت شده باید پردازش صورت گیرد و نام دومین آن جدا شود (اگر ختم به خارج از سایت بود). سپس بر اساس این دومین ، یک تگ img منتهی به آیکون آن سایت طراحی و نمایش داده خواهد شد (به قبل از لینک اضافه می‌شود).

عموما هر پروژه‌ای هر چند کوچک و به ظاهر کم اهمیت، نکات خاصی را به همراه دارد.

برای مثال، در ابتدای کار width=0 در نظر گرفته نشد. مشکلی را که ایجاد کرد این بود که یک سایت ممکن است اصلا favicon.ico نداشته باشد. فایرفاکس محترم اصلا به روی مبارک هم نخواهد آورد و شما متوجه نخواهید شد که در صفحه کمبود تصویری وجود دارد. اما در IE حتما جای خالی آیکون و تصویر به نحوه واضحی گوشزد می‌شود. بنابراین در ابتدای کار قرار نیست چیزی را نمایش دهیم (width=0). سپس در رخداد onload تگ img، اگر تصویر واقعا وجود داشت و بارگذاری شد، طول و عرض آن‌را 16 در 16 قرار خواهیم داد (این مورد هم لازم است. چون بعضی از سایت‌ها اندازه‌های بسیار بزرگی را برای آیکون خود در نظر می‌گیرند که اصلا مقصود ما را برآورده نخواهند کرد).

این مورد (عدم نمایش تصاویری که وجود ندارند) مشکلی را که در ادامه ایجاد خواهد کرد، عدم یکنواختی در سایت است. یک سری از لینک‌های خارجی آیکون دارند و یک سری خیر. نکته جالبی که در مورد تگ img وجود دارد رخداد onerror آن است. اگر مرورگر نتواند تصویر مورد نظر را پیدا یا بارگذاری کند، این روال رخداد گردان فراخوانی می‌شود. همینجا از موقعیت استفاده شده و src تصویر جاری به یک آیکون مشخص و از قبل تعیین شده تنظیم می‌شود.

نظرات خوانندگان

نویسنده: مسعود
تاریخ: ۱۳۸۷/۱۰/۲۱ ۰۰:۵۷:۰۰

جالب بود.من قبلا سعی کرده بودم که یه جورایی از صفحه عکس بگیرم و به عنوان تول تیپ نمایش بدم که سرور قوی می خواست و نشد.

این روش خیلی خوبیه.خیلی جالب بود

مرسی استاد

(;

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۱ ۰۱:۱۸:۰۰

این هم فکر جالبیه.
سایت زیر سرویس مورد نظر شما رو به رایگان ارائه میده:
/http://w.googlepreview.com/preview?s=http://www.dotnettips.info

البته در سایتش نوشته که مجاز نیستید بجز در افزونه مورد نظر فایرفاکس ازش استفاده کنید، ولی خوب دیگه، محض اطلاع ;)

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۲ ۱۲:۰۸:۰۰

این سایت هم سرویس thumbnail ارائه می‌دهد:
http://www.pageglimpse.com/features/api

عنوان: **تغییر ابعاد یک TextArea با استفاده از jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۲ ۱۹:۵۱:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

اگر به سایت stackoverflow دقت کنید، اندازه textarea ایی که کاربران امکان ارسال مطلب دارند، قابل تغییر است:

شاید برای شما جالب باشد که بدانید به چه صورتی اینکار را انجام داده‌اند. اینکار با استفاده از افزونه TextArea Resizer صورت گرفته است. (دریافت کتابخانه به همراه مثال)

البته حالت عمومی‌تری نیز توسط jQuery-UI استاندارد پشتیبانی می‌شود (امکان تغییر اندازه یک المان با قابلیت تغییر اندازه در حالت کلی). برای مثال به صفحه‌ی ساده‌ی ASP.Net زیر دقت بفرمائید:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm5.aspx.cs"
Inherits="testWebForms87.WebForm5" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
   <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.2.6/jquery.min.js"
type="text/javascript"></script>
   <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.5.3/jquery-ui.min.js"
type="text/javascript"></script>

   <script type="text/javascript">
       $(function() {
           $("#resizableArea").resizable({
               handles: "s"
           })
  .find("textarea").height("100%").width("100%");
       });
   </script>

</head>
<body>
   <form id="form1" runat="server">
   <div id="resizableArea" style="width:300px; height:200px" >
       <asp:TextBox ID="TextBox1" runat="server"  TextMode="MultiLine" ></asp:TextBox>
   </div>
   </form>
</body>
</html>
```

در اینجا با استفاده از jQuery-UI ابتدا div ایی با id مساوی resizableArea پیدا می‌شود و تابع resizable به آن اعمال می‌گردد. سپس در این div ، تمامی عناصر textarea موجود یافت شده و طول و عرض آن‌ها به اندازه‌ی جدید تغییر یافته div تنظیم خواهد شد. در مورد تنظیم‌های دیگری مانند نمایش ناحیه تغییر یافته به صورت animation می‌توان به راهنمای آن مراجعه نمود.
برای مثال با استفاده از این روش می‌توان یک GridView با قابلیت تغییر اندازه ایجاد کرد و امثال آن. یا برای نمونه شاید با مثال‌هایی که به گرید نمایش داده شده اسکرول بار اضافه می‌کنند برخورده باشید:

```
<div id="resizableArea" style="overflow:auto;height:200px;">
My Grid view ...
</div>
```

و یا استفاده از یک پنل:

```
<asp:Panel ID="pnlScroll" runat="server" Width="391px" Height="282px" ScrollBars="Vertical">
My Grid view ...
</asp:Panel>
```

با استفاده از روش عمومی فوق، می‌توان به این div امکان تغییر اندازه را نیز اضافه کرد و آن‌را از حالت غیرقابل تغییر بودن خارج نمود.

عنوان: **JQuery 1.3 ارائه شد**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۷ ۱۱:۱۹:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

روز قبل، نگارش رسمی و نهایی jQuery 1.3 ارائه شد و بلافاصله هم فایل مخصوص آن جهت بکارگیری intellisense ویژوال استودیو، توسط علاقمندان تهیه و عرضه گردید .

تازه‌های این نگارش:
Sizzle : یک موتور CSS selector قدرتمند (400 درصد بهبود سرعت نسبت به عملکرد کتابخانه قبلی در این مورد)
بازنگری در نحوه مدیریت رخدادها
موتور تزریق HTML بسیار سریع (تا 15 برابر سریعتر نسبت به کتابخانه قبلی)
موتور بسیار سریع موقعیت یابی
تشخیص نوع مرورگر در آن متوقف شده و بجای آن از تشخیص ویژگی‌ها برای ماندگاری بیشتر این کتابخانه در سال‌های آتی استفاده گردیده است. (بجای بررسی userAgent ، از باگ‌های شناخته شده یا ویژگی‌های خاص، جهت تشخیص مرورگر استفاده می‌شود تا بازه وسیعی از محصولات مشابه را بتوانند پوشش دهند)
لیست کامل این تغییرات را در این آدرس می‌توانید ملاحظه نمائید.
جهت مشاهده و آزمایش بهبود کارآیی این کتابخانه لطفا به این صفحه مراجعه کنید. (لیست کامل این آزمایشات در انتهای صفحه لیست تغییرات، قابل دسترسی است)
همچنین تیم مربوطه در پی آن است تا در طی یکی دو هفته آینده، نگارش jQuery 1.3.1 را نیز بر اساس بازخوردهای جدید جامعه برنامه نویس‌ها ارائه دهد.

مانند همیشه این کتابخانه در دو نگارش فشرده شده (جهت استفاده در سایت‌ها) و نگارش عادی و حجیم‌تر (جهت برنامه نویسی و دیباگ کردن کدها) ارائه شده است. چون کدها در Google code هاست شده احتمالا دسترسی به آن مشکل خواهد بود. این مجموعه را از این آدرس می‌توانید دریافت کنید.

توصیه می‌شود از سرورهای سریع گوگل بعنوان هاست کتابخانه جی‌کوئری سایت‌های خود استفاده کنید. برای این منظور از آدرس زیر می‌توان استفاده کرد:

```
http://ajax.googleapis.com/ajax/libs/jquery/1.3/jquery.min.js
```

نظرات خوانندگان

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۱۰/۲۷ ۱۰:۱۸:۰۰

سلام؛

این Jquery انقدر جالبه که من که برنامه نویس نیستم، علاقه مند شدم... اولین بار هم از طریق وبلاگ شما آشنا شدم. خیلی دوست دارم که در موردش بیشتر بدونم... خیلی زیاد تو سایت jquery.com گشتم و چیز های جالبی هم پیدا کردم. اگر امکانش باشه، می خواستم خواهش کنم یه سری مطالب با تمرکز بیشتر در این مورد بنویسید. مثلا این توصیه هاست روی گوگل جالب بود. حالا نمی دونم اگر روی هاست داخل ایران باشه، آیا سرعت لودشدنش در ایران بیشتر میشه یا گوگل؟! فعلا دارم دنبال پلاگین های جالب Jquery می گردم... خیلی متشکرم از مطالب خیلی جالبتون.

نویسنده: chalism
تاریخ: ۱۳۸۷/۱۰/۲۸ ۰۲:۰۴:۰۰

گوگل اجازه دانلود نمی ده.
بهتره رو یه هاست دیگه آپلودش کنید و لینک بدید.
سپاسگذارم

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۲۸ ۰۹:۱۱:۰۰

این‌کار انجام شده. لطفا به قسمت "از این آدرس می‌توانید دریافت کنید"، دقت نمائید.

نویسنده: مهدی
تاریخ: ۱۳۸۷/۱۰/۲۸ ۱۹:۵۵:۰۰

جدول مقایسه مجوزهای سورس باز رو دیدم یاد این مقایسه ی جالب فریم ورکهای جاوااسکریپت افتادم.

http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۱/۰۱ ۱۲:۱۴:۰۰

فقط mootools

نویسنده: مهدی
تاریخ: ۱۳۸۷/۱۱/۰۱ ۲۱:۴۰:۰۰

منم با dojo خیلی حال کردم ولی خوب فلن jquery رو بورسه

عنوان: **پردازش فایل‌های XML با استفاده از jQuery**
نویسنده: وحید نصیری
تاریخ: ۲۱:۰۰:۲۷ ۱۳۸۷/۱۱/۰۸
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

فرض کنید مثال دریافت اطلاعات API فیدبرنر را بخواهیم با استفاده از jQuery پیاده سازی کنیم، یعنی امکان برنامه نویسی سمت سرور را نداریم و می‌خواهیم با استفاده از جاوا اسکریپت، تعداد مشترکین فید یک سایت را نمایش دهیم.
برای مثال آدرس دریافت اطلاعات مربوط به API فیدبرنر وبلاگ جاری به صورت زیر است:
https://feedburner.google.com/api/awareness/1.0/GetFeedData?id=fhphjt61bueu08k93ehujpu234

و در حالت کلی :

```
http://api.feedburner.com/awareness/1.0/GetFeedData?uri=<feeduri>
```

که حاصل آن برای مثال یک فایل XML با فرمت زیر می‌باشد:

```
<rsp stat="ok">
 <feed id="fhphjt61bueu08k93ehujpu234" uri="vahidnasiri">
  <entry date="2009-01-23" circulation="153" hits="276" reach="10"/>
 </feed>
</rsp>
```

همانطور که مطلع هستید چند روزی است که jQuery 1.3.1 ارائه شده است. جهت استفاده از آخرین نگارش موجود آن تا این زمان، می‌توان از گوگل به عنوان هاست این کتابخانه به صورت زیر استفاده کرد:

```
<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.3.1/jquery.min.js'
type='text/javascript'></script>
```

نحوه خواندن

مقدار circulation فایل xml ذخیره شده بر روی کامپیوتر:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
     <title>FeedBurner API</title>
     <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.3.1/jquery.min.js'
type='text/javascript'>
     </script>
     <script type="text/javascript">
         function parseXml(xml){
             //find every entry and print the circulation
             $(xml).find("entry").each(function(){
                 $("#output").append($(this).attr("circulation"));
             });
         }

         $(document).ready(function(){
             $.ajax({
                 type: "GET",
                 url: "GetFeedData_local.xml",
                 dataType: "xml",
                 success: parseXml
             });
         });
     </script>
```

```
 </head>
 <body>
     <div dir="rtl" style="font-family:tahoma; font-size:12px;">
         تعداد مشترکین تغذیه خبری سایت:
         <div id="output">
         </div>
     </div>
 </body>
</html>
```

با استفاده از قابلیت Ajax کتابخانه jQuery ، اطلاعات فایل محلی GetFeedData_local.xml دریافت شده و محتوای آن به تابع parseXml پاس می‌شود (توسط قسمت success). سپس در این تابع تمام تگ‌های entry یافت شده و مقدار circulation آن‌ها به یک div با ID معادل output اضافه می‌شود.
مثال فوق در مورد خواندن اطلاعات از یک فایل xml می‌تواند برای مثال این کاربرد را در یک سایت داشته باشد:
نمایش اتفاقی سخن روز یا سخن بزرگان و امثال آن بدون برنامه نویسی سمت سرور جهت انجام این کار از یک فایل xml تهیه شده، بدون نیاز به استفاده از دیتابیس خاصی.

تا اینجای کار مشکلی نیست. اما همانطور که در مطلب مقابله با حملات CSRF نیز ذکر شد، مرورگرهای جدید امکان ارسال یا دریافت اطلاعات به صورت Ajax را بین سایت‌ها ممنوع کرده‌اند (ماجرا هم از آنجا شروع شد که یکبار جی‌میل این باگ امنیتی را داشته است). بنابراین اگر شما بجای url قسمت Ajax فوق، آدرس سایت فید برنر را قرار دهید با خطای زیر متوقف خواهید شد:

```
Access to restricted URI denied
```

تمام موارد دیگری هم که در jQuery برای دریافت اطلاعات از یک فایل یا url موجود است (مثلا تابع load یا get و امثال آن) فقط به سایت جاری و دومین جاری باید ختم شوند در غیر اینصورت توسط مرورگرهای جدید متوقف خواهند شد.

نظرات خوانندگان

نویسنده: MicroDump
تاریخ: ۱۳۸۷/۱۱/۰۹ ۱۷:۴۰:۰۰

پس نمی‌شه!

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۱۰ ۰۰:۲۳:۰۰

با استفاده از jsonp امکان cross site Ajax request در jquery مهیا است اما زیاد به صورت مساله ما نمی‌خوره!

عنوان: **سازگار کردن GridView با افزونه‌های jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۱۲ ۱۵:۳۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

افزونه‌ها/پلاگین‌های زیادی جهت کار با table استاندارد HTML وجود دارند و خروجی رندر شده‌ی یک ASP.Net GridView هم در نهایت یک جدول است. فرض کنید قصد داریم افزونه زیر را به GridView استاندارد ASP.Net اعمال کنیم.

jQuery quickSearch plug-in

ظاهرا بدون مشکل خاصی اعمال می‌گردد. برای مثال در هدر صفحه داریم: (شبیه به مثال موجود در سایت اصلی آن، جهت اعمال به GridView1)

```
<script src="jquery.min.js" type="text/javascript"></script>
<script src="jquery.quicksearch.js" type="text/javascript"></script>

<script type="text/javascript">
 $(document).ready(function() {
     //جستجو در جدول
 $('table#<%=GridView1.ClientID %> tbody tr').quicksearch({
         position: 'before',
         attached: 'span#attachSearch',
         labelText: 'جستجو',
         isFieldset: true,
         loaderText: ' ... ',
         fixWidths: true
     });
 });
</script>
```

برای اعمال بر:
(در اینجا محل قرارگیری تکست باکس مربوط به جستجو، در span ایی با id مساوی attachSearch تنظیم شده است، می‌توانید از ID خود GridView هم استفاده کنید.)

```
<span id="attachSearch"></span>
<br />
<asp:GridView ID="GridView1" runat="server"></asp:GridView>
```

نکته:
1- همانطور که در مقاله مربوط به ClientID ذکر شد، هیچ الزامی ندارد که ID مربوط به GridView شما برای مثال دقیقا همان GridView1 جهت استفاده در سمت کلاینت باشد و بسته به container آن، این نام ترکیبی از ID شیء(های) در بر گیرنده و شیء مورد نظر می‌باشد. به همین جهت از GridView1.ClientID استفاده گردید تا اسکریپت ما با آن مشکلی نداشته باشد.

2- خصوصیات ظاهری افزونه فوق از سلکتور quicksearch فایل css شما دریافت می‌شوند. برای مثال:

```
.quicksearch
{
width:190px;
}
```

مشکل!

پس از هر بار جستجو، header مربوط به GridView محو می‌شود، اما نمونه موجود در سایت اصلی افزونه، این مشکل را ندارد. چرا؟!

GridView مربوط به ASP.Net پس از رندر شدن، جدولی است که تگ‌های thead را ندارد. اگر به سورس صفحه سایت افزونه دقت نمائید، هدر جدول با تگ‌های thead محصور شده است اما GridView استاندارد ASP.Net به صورت پیش فرض این‌کار را انجام نمی‌دهد و خروجی آن چیزی شبیه به جدول زیر است: (هدر با th مشخص می‌شود و از تگ thead خبری نیست)

```
<table >
<tr >
          <th scope="col">col1</th>
             <th scope="col">col2</th>
             <th scope="col">col3</th>
</tr>
     <tr>
       <td>1</td>
            <td>2</td>
            <td>3</td>
</tr>
.
.
.
```

حداقل دو راه حل برای رفع این مشکل وجود دارد:

الف) راه حل سمت سرور

برای اضافه کردن thead باید کمی کد نویسی کرد. پس از اینکه گرید شما بایند شد، چند سطر زیر را اضافه کنید:

```
//سازگار با افزونه‌های جی کوئری
         if (GridView1.Rows.Count > 0)
         {
             //This replaces <td> with <th> and adds the scope attribute
             GridView1.UseAccessibleHeader = true;

             //This will add the <thead> and <tbody> elements
             GridView1.HeaderRow.TableSection = TableRowSection.TableHeader;

             //This adds the <tfoot> element.
             //Remove if you don't have a footer row
             //GridView1.FooterRow.TableSection = TableRowSection.TableFooter;
         }
```

ب)راه حل سمت کلاینت

سطر مربوط به جستجو را به صورت زیر هم می‌توان نوشت:

```
$('table#<%=GridView1.ClientID %> tr:has(td)').quicksearch({
```

در اینجا به دنبال هرچی tr که td دارد می‌گردیم. به صورت پیش فرض در tr مربوط به هدر گرید، ما th داریم و نه td ، بنابراین مشکل محو شدن هدر به این صورت حل خواهد شد.

نکته:

اگر می‌خواهید که ناحیه مربوط به جستجوی افزونه فوق در پرینت صفحه ظاهر نشود به css صفحه چند سطر زیر را اضافه کنید:

```
@media print
{
     .quicksearch
     {
         display: none;
     }
}
```

تمرین!

افزونه جی کوئری زیر را به یک ASP.Net GridView اعمال کنید:

table sorter

نظرات خوانندگان

نویسنده: Ameer
تاریخ: ۱۳۸۹/۰۹/۱۰ ۱۸:۳۰:۲۴

اگه گرید paging بشه دیگه کاربردی نداره!
راه حلی سراغ دارید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۰ ۱۸:۴۸:۰۲

با کد نویسی اعمال بایندینگ را پیاده سازی کنید تا بتوانید در هر بار paging مجددا thead یاد شده را اعمال و تولید کنید.

نویسنده: Ameer
تاریخ: ۱۳۸۹/۰۹/۱۰ ۱۹:۳۳:۲۴

راهشو پیدا کردم!
http://www.codeproject.com/KB/webforms/clientside_gridviewpaging.aspx

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۱۰ ۲۱:۵۴:۳۰

اون مقاله داره میگه اگر 10 هزار ردیف داشتی همه رو یکجا نمایش بده بعد اجازه بده که مثلا جاوا اسکریپت برات صفحه بندی کنه.
ولی قسمت مدیریت "GridView1_PreRender" اون جهت اضافه کردن thead احتمالا برای شما مفید بود. چون بحث اصلی مقاله جاری هم همین است.

عنوان: **افزونه جملات قصار jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۱۶ ۱۳:۲۹:۵۳
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

چندی قبل مطلبی را در مورد پردازش فایل‌های xml با استفاده از قابلیت Ajax جی کوئری نوشتم. در سایت پی‌سافت ، تعدادی فایل XML از شعرا و جملات قصار و امثال آن موجود است (با تقدیر و تشکر از زحمات این عزیزان) که امروز قصد داریم از فایل XML جملات قصار آن یک افزونه jQuery درست کنیم تا آن‌ها را به صورت اتفاقی (random) در صفحه نمایش دهد:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
       <title>XML</title>
       <script src='jquery-1.3.min.js' type='text/javascript'></script>
       <script type="text/javascript">
           var ourXml = '';
           function parseXml(xml){
               ourXml = xml; //for our timer
               var i;
               var rnd = Math.floor(Math.random() * 130) + 1; //we have 130 entries
               for (i = 1; i < 5; i++) //M1 to M4
               {
                   $(xml).find("Ghesaar" + rnd + " > M" + i).each(function(){
                      $("#output").append($(this).text() +' ');
                   });
               }
           }

           //ajax loader
           $(document).ready(function(){
               $.ajax({
                   type: "GET",
                   url: "Ghesaar.xml",
                   dataType: "xml",
                   success: parseXml
               });
           });

           //timer
           window.setInterval(function(){
               $("#output").empty().hide();
               parseXml(ourXml);
               $("#output").fadeIn("slow");
           }, 10000);
       </script>
   </head>
   <body>
           <span id="output" dir="rtl" style=" "/>
   </body>

</html>
```

توضیحات:

همه چیز از قسمت Ajax کد فوق شروع می‌شود. فایل Ghesaar.xml بارگذاری شده و به تابع parseXml ارسال می‌شود. در این تابع یک کپی از xml دریافت شده را نگهداری می‌کنیم تا در تایمری که بعدا جهت نمایش اتفاقی پیام‌ها درست خواهیم کرد، مجبور نشویم مجددا محتویات فایل xml را بارگذاری کنیم.

در تابع parseXml قصد داریم فایل xml ایی با فرمت زیر را پردازش کنیم:

```
<Ghesaar10>
   <M1>ناامیدی، آخرین نتیجه گیری    </M1>
   <M2>بی خردان است</M2>
   <M3>
   </M3>
   <M4>« ضرب المثل انگلیسی »</M4>
```

```
</Ghesaar10>
```

برای مثال در اینجا باید به دنبال Ghesaar10>M1 برای یافتن متن تگ M1 گشت و الی آخر. کلا چهار تگ داریم که در یک حلقه آن‌ها را استخراج خواهیم کرد. سپس یک عدد اتفاقی بین 1 تا 130 هم تولید کرده و بجای عدد پس از Ghesaar قرار می‌دهیم. یعنی هر بار به صورت اتفاقی یک مجموعه از جملات قصار، دریافت و پردازش خواهند شد. نهایتا این جملات استخراج شده را به یک span با id مساوی output اضافه می‌کنیم.

تا اینجا فقط یکی از جملات قصار در هنگام بارگذاری صفحه نمایش داده خواهند شد. برای تکرار نمایش، از یک تایمر می‌توان کمک گرفت که کد آن‌را در بالا ملاحظه می‌کنید.

همین!

برای تبدیل آن به یک پلاگین/افزونه جی‌کوئری ، می‌توان به صورت زیر عمل کرد:

```
$.fn.ghesaar = function(options){
   var defaults = {
       interval: 1
   };
   var options = $.extend(defaults, options);

   return this.each(function(){
       var obj = $(this);
       var ourXml = '';
       function parseXml(xml){
           ourXml = xml; //for our timer
           var i;
           var rnd = Math.floor(Math.random() * 130) + 1; //we have 130 entries
           for (i = 1; i < 5; i++) //M1 to M4
           {
               $(xml).find("Ghesaar" + rnd + " > M" + i).each(function(){
                   obj.append($(this).text() + ' ');
               });
           }
       }

       //ajax loader
       $.ajax({
           type: "GET",
           url: "Ghesaar.xml",
           dataType: "xml",
           success: parseXml
       });

       //timer
       window.setInterval(function(){
           obj.empty().hide();
           parseXml(ourXml);
           obj.fadeIn("slow");
       }, options.interval * 1000);

   });


};
```

فرمت

این فایل ساده و استاندارد است. نام فایل jquery.ghesaar.js خواهد بود.

سپس قسمت استاندارد توسعه options اضافه می‌شود تا بتوان به تابع افزونه خود مقدار interval را پاس کرد تا از این حالت خشک و جمود خارج شود.

کل وقایع افزونه درون تابع زیر رخ می‌دهد:

```
return this.each(function(){
...

});
```

اینجا همان کدهایی را که پیشتر توسعه دادیم بدون هیچ تغییری قرار می‌دهیم.
سپس در ابتدای کار شیء this که اشاره‌گری است به شیء انتخاب شده توسط جی‌کوئری را دریافت کرده و هرجایی را که قبلا $("output#") داشتیم، تبدیل به obj می‌کنیم. (یعنی این مورد هم به انتخاب کاربر خواهد شد)
جهت دریافت مقدار تنظیمی interval هم می‌توان از options.interval استفاده کرد.
به این صورت کد ما تبدیل به یک افزونه جی‌کوئری می‌شود.

اینبار نحوه‌ی استفاده از افزونه‌ی تولیدی به صورت زیر است: (عدد interval بر اساس ثانیه است)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
       <title>XML</title>
       <script src='jquery-1.3.min.js' type='text/javascript'></script>
  <script src='jquery.ghesaar.js' type='text/javascript'></script>
       <script type="text/javascript">
           $(document).ready(function(){
               $("#output").ghesaar({interval:5});
           });

       </script>
   </head>
   <body>
           <span id="output" dir="rtl" style=" "/>
   </body>

</html>
```

بهتر شد، نه؟!

به صورت ساده:
برای استفاده از آن، سه فایل jquery-1.3.min.js (یا نگارش جدیدتر آن)، jquery.ghesaar.js و Ghesaar.xml را باید آپلود کنید. چند سطری را که در قسمت head صفحه فوق مشاهده می‌نمائید باید اضافه شوند. سپس یک span یا div را جهت نمایش این جملات قصار به هر جایی از ساختار صفحه خود که علاقمند بودید اضافه کنید (id آن مهم است و در قسمت نمایش جملات قصار مورد جستجو قرار می‌گیرد).

فایل‌های این پروژه را از اینجا دریافت کنید.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۱۶ ۱۴:۵۲:۰۰

نکته‌ای در مورد بارگذاری آفلاین فایل xml و IE

http://stackoverflow.com/questions/436670/local-html-file-ajax-call-and-jquery-woes

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۱/۰۵/۰۲ ۹:۵۳

سلام استاد به نظر شما برای نمایش این جملات به صورت روزانه همون تغییر Interval کفایت می‌کنه یا راه حلی بهتری رو پیشنهاد می‌کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۰:۸

من بعید می‌دونم کسی یک روز تمام مرورگر خودش را صرفا جهت نمایش سایت شما باز نگه دارد. بنابراین تنظیم بازه‌های زمانی نمایش، کفایت می‌کند.

عنوان: **رفع تداخل jQuery با کتابخانه‌های مشابه**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۲۰ ۱۰:۳۱:۱۷
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

قبل از شروع، یک خبر!
VsDoc for jQuery 1.3.1 (جهت فعال سازی intellisense آخرین نگارش جی کوئری در VS.Net)

اگر سعی کنید jQuery را به همراه سایر کتابخانه‌های جاوا اسکریپتی دیگر به صورت همزمان استفاده کنید (مثلا mootools یا ASP.Net Ajax و امثال آن)، احتمالا قسمتی و یا تمامی کدهای جاوا اسکریپتی شما کار نخواهند کرد. برای مثال update panel شما در ASP.Net Ajax از کار می‌افتد، یا کدهای mootools شما دیگر کار نمی‌کنند. علت اینجا است که تمامی این کتابخانه‌ها از نشانه $ به عنوان متغیری عمومی که بیانگر نام مستعار کتابخانه مربوطه است استفاده می‌کنند و در نهایت تمام این‌ها با هم تداخل خواهند کرد.

خوشبختانه jQuery امکان رفع این تداخل را پیش بینی کرده است که به صورت زیر می‌باشد:

```
<script type="text/javascript" language="javascript" src="jquery.min.js"></script>
<script type="text/javascript">
   jQuery.noConflict();
   jQuery(document).ready(function($) {
       //tip-1
       $("select > option").each(function() {
           var obj = $(this);
           obj.attr("title", obj.attr("value"));
       });
       //tip-1
   });
</script>
```

کد مثال فوق، به تمامی آیتم‌های drop down list های شما در یک صفحه، بر اساس value هر آیتم موجود در آن‌ها، یک tooltip اضافه می‌کند. (با IE7 به بعد و فایرفاکس سازگار است)
در اینجا ابتدا jQuery.noConflict فراخوانی شده و سپس document ready متداول هم باید اندکی مطابق کد فوق تغییر کند.
مابقی کدهای شما از این پس نیازی به تغییر نخواهند داشت. (روش‌های دیگری هم برای تغییر نام $ وجود دارند که در مستندات مربوطه قابل مشاهده است)

عنوان: **استفاده از thickbox در بلاگر**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۳/۱۴ ۲۲:۰۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

Thickbox

یکی از پلاگین‌های jQuery است که جهت نمایش صفحات modal بکار می‌رود.

روش استفاده بسیار ساده‌ای هم دارد:

الف) jquery.js باید به صفحه معرفی شود.

ب) سپس thickbox.js باید به صفحه الحاق شود.

ج) و فایل css آن یعنی thickbox.css نیز باید به صفحه افزوده شود.

برای استفاده از آن کافی است یک لینک به صفحه اضافه کنید که ویژگی class آن مساوی thickbox باشد. مثلا:

```
<a href="#TB_inline?height=155&width=300&inlineId=hiddenModalContent&modal=true" class="thickbox">Show
hidden modal content.</a>
```

TB_inline و طول و عرض و غیره، جزو خواص استاندارد این پلاگین است. inlineId جهت معرفی id یک div در صفحه بکار می‌رود که قرار است محتویات آن زمانیکه روی این لینک کلیک شد به صورت modal نمایش داده شود.

```
<a
href="iframeModal.html?placeValuesBeforeTB_=savedValues&TB_iframe=true&height=200&width=300&modal=true"
title="add a caption to title attribute / or leave blank" class="thickbox">Open iFrame Modal</a>
```

و یا مانند مثال فوق می‌تواند یک صفحه خارجی را نیز به صورت modal بارگذاری و نمایش دهد. Title بکارگرفته شده، همان عنوان صفحه Modal ما خواهد بود.

و مثال‌های دیگری از این دست که در صفحه رسمی آن موجود است و نیازی به تکرار آن‌ها نیست.

اما مشکلی که در بلاگر وجود دارد این است که شما نمی‌توانید این ویژگی‌ها را در بسیاری از موارد به صورت دستی تنظیم کنید. یک ویجت ظاهر می‌شود که فقط به شما امکان ثبت یک عنوان و لینک مربوطه را می‌دهد و همین.

برای انتساب ویژگی‌های thickBox به این نوع لینک‌ها که کنترلی روی آن نیست می‌توان از خود jQuery کمک گرفت:

```
$(document).ready(function(){
  $("a[href='http://www.mysite.com/page.htm?TB_iframe=true&height=340&width=530']").each(function(){
      var obj = $(this);
      obj.attr("title","Contact me");
      obj.attr("class","thickbox");
     }).bind("contextmenu",function(e){
        return false;
    });
});
```

در اینجا http://www.mysite.com/page.htm?TB_iframe=true&height=340&width=530 همان صفحه‌ای که قرار است پس از کلیک بر روی لینکی منتسب به آن در صفحه به صورت modal باز شود. با استفاده از جی‌کوئری آن‌را یافته (لینکی که href آن مساوی مقدار یاد شده است) و سپس ویژگی‌های title و class آن را به صورت پویا تنظیم می‌کنیم تا توسط thickbox قابل شناسایی شود.

همچنین در اینجا کلیک راست نیز بر روی این لینک بسته شده تا کاربر فقط کلیک معمولی را انجام دهد (فقط روی همین یک لینک در صفحه اعمال می‌شود و نه کل صفحه).

عنوان: **اعمال متداول با select (یا همان DropDownList / ComboBox) توسط jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۳۰ ۱۸:۰۱:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

در این مقاله مروری خواهیم داشت بر نحوه‌ی انجام یک سری از اعمال متداول با استفاده از jQuery بر روی select استاندارد HTML. قبل از شروع به عنوان قالب کار، صفحه ساده‌ی زیر را در نظر بگیرید که از یک DropDownList استاندارد ASP.Net تشکیل شده است:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="select.aspx.cs"
Inherits="TestJQueryAjax.select" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
   <script src="js/jquery.js" type="text/javascript"></script>

   <script type="text/javascript">
   //...
   </script>

</head>
<body>
   <form id="form1" runat="server">
   <asp:DropDownList ID="ddlTest" runat="server">
       <asp:ListItem Value="1">آیتم یک</asp:ListItem>
       <asp:ListItem Value="2">آیتم دو</asp:ListItem>
       <asp:ListItem Value="3">آیتم سه</asp:ListItem>
       <asp:ListItem Value="4">آیتم چهار</asp:ListItem>
       <asp:ListItem Value="5">آیتم پنج</asp:ListItem>
   </asp:DropDownList>
   </form>
</body>
</html>
```

1) بدست آوردن value آیتم انتخاب شده

```
$("#<%=ddlTest.ClientID %>").val()
```

2) بدست آوردن text آیتم انتخاب شده

```
$("#<%=ddlTest.ClientID %> option:selected").text()
```

3) عکس العمل نشان دادن به انتخاب آیتم‌های مختلف

```
<script type="text/javascript">
       $(document).ready(function() {
           $("#<%=ddlTest.ClientID %>").change(function() {
               alert($("#<%=ddlTest.ClientID %>").val());
           });
       });
   </script>
```

4) انتخاب یک آیتم از طریق برنامه نویسی

```
$("#<%=ddlTest.ClientID %>").val(2);
```

5) افزودن یک آیتم به لیست از طریق برنامه نویسی

```
$("<option value='6'>آیتم شش</option>").appendTo("#<%=ddlTest.ClientID %>");
```

6) خالی و حذف کردن تمامی آیتم‌ها

```
$("#<%=ddlTest.ClientID %>").html("");
```

و برای تکمیل بحث می‌توان به این برگه مرجع مراجعه کرد:

jQuery – Select element cheat sheet

نظرات خوانندگان

نویسنده: Mohammad Shams Javi
تاریخ: ۱۳۸۸/۰۷/۰۱ ۰۰:۵۸:۲۲

سلام

جالب بود.
آیا افزونه مناسب تری به جای SyntaxHighlighter سراغ ندارید. هم زمان بارگذاری را خیلی افزایش می‌دهد و هم اینکه در صورت لود ناقص صفحه، باعث بروز StackOverflow در IE میشود.(بارها پیش آمد)

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۰۱ ۰۱:۴۷:۵۶

سلام،
چرا 8 مورد دیگر هم هستند:
/http://www.webdesignbooth.com/9-useful-javascript-syntax-highlighting-scripts
بعلاوه لایور رایتر مایکروسافت به همراه پلاگین copy as html برای VS.Net

البته اگر منظور سایت جاری باشد این مشکل فکر نکنم به علت این syntax HL باشد، مشکل از اتمام پهنای باند سایت گوگل پیج است (خصوصا هنگام نمایش تصاویر صفحه از آن و یک سری بررسی جاوا اسکریپتی).

نویسنده: محمد امین شریفی
تاریخ: ۱۳۸۸/۰۷/۰۳ ۱۷:۴۳:۵۹

چقدر طول میکشه که با syntax این کتابخانه آشنا شویم.
آیا همکاری با ماکروسافت و تلفیق با asp.net سینتکس آن با asp.net و MSAjax شبیه تر می شود؟(jquery فقط یک نوع syntax و روش برنامه نویسی دارد؟-چون آخه این syntax خیلی راحت بود،ولی نفهمیدم دلیل = در برنامه هاتون چی هست،مگه می خواییم مقداری را چاپ کنیم).

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۰۳ ۱۹:۳۹:۳۳

- یک تا دو روز (و مسلما برای تکمیل بحث نیاز به مطالعه مقالات و کتاب می‌باشد که زمان‌بر است)
- خیر.
- دلیل استفاده از = :
http://www.dotnettips.info/2008/11/control-id-aspnet.html

نویسنده: Amir
تاریخ: ۱۳۸۸/۰۷/۲۰ ۲۰:۵۰:۰۴

سلام. مرسی بابت آموزش. عالی بود.
چطوری میشه با radiobuttonlist این کارو انجام داد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۲۱ ۰۹:۴۳:۱۷

سلام،
می‌تونید از مقاله تعیین اعتبار یک checkBoxList با کمک jQuery کمک/ایده بگیرید:

http://www.dotnettips.info/2009/08/checkboxlist-jquery.html

کنترل radiobuttonlist جزو اختراعات مایکروسافت است! به این صورت که هنگامیکه رندر می‌شود خروجی یک جدول است حاوی input و label استاندارد html .

بر این اساس، پیدا کردن مقدار text و val آن به صورت زیر می‌تواند باشد:

```
$("table#<%=RadioButtonList1.ClientID %> input:checked").val()
$("table#<%=RadioButtonList1.ClientID %> input:checked").siblings("label").text()
```

عنوان: **نمایش خودکار مقدار یکDropDownList با کمک jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۰۳ ۱۴:۵۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

نیاز بود هنگام انتخاب یک آیتم دراپ داون لیست در کل برنامه و تمامی دراپ داون‌های آن، مقدار آن‌ها نیز به صورت یک برچسب در کنار آن نمایش داده شود.
برای مثال در لیست زیر:

```
<asp:DropDownList ID="ddlActive" runat="server">
     <asp:ListItem Value="Active">فعال</asp:ListItem>
     <asp:ListItem Value="Inactive">غیرفعال</asp:ListItem>
 </asp:DropDownList>
```

اگر آیتم فعال انتخاب شد، مقدار active نیز کنار آن نمایش داده شود و الی آخر.

راه حل اول:
در تمام صفحات به ازای تک تک دراپ داون‌ها یک label اضافه کنیم و همچنین کدهای تمام قسمت‌های برنامه را نیز اصلاح کنیم تا این مورد را لحاظ کند.

راه دوم:
یک کنترل دراپ داون سفارشی را با خاصیت مورد نظر (همراه بودن با یک لیبل) ایجاد کرده و سپس تمام فرم‌ها را باید اصلاح کرد تا از این کنترل جدید استفاده کنند.

راه سوم:
استفاده از jQuery برای اعمال این مهم به کل برنامه بدون نیاز به تغییرات اساسی در آن (و همچنین سازگاری با تمام مرورگرها):

```
//فقط در این محدوده
   $("#mainFormReq select").change(function() {
       var currentId = $(this).attr("id"); //آی دی شیء جاری
       var val = $(this).val();        //مقدار
       var text = $('#' + currentId + ' option:selected').text(); //متن
       $("#lbl" + currentId).remove();        //اگر نمونه‌ی قبلی موجود است حذف شود
       if (val && (val.length > 0) && (text != val)) {
           //اگر متن و مقدار یکی نیست نمایش داده شود
           $(this).after('<label id="lbl' + currentId + '">' + val + '</label>');
       }
   });
```

توضیحات:
در یک محدوده مشخص شده با ID مساوی mainFormReq (مثلا استفاده از master page ها و نسبت دادن این ID به content آن)، به دنبال تمام select های موجود در آن ناحیه می‌گردیم (اگر mainFormReq حذف شود، این جستجو در کل صفحه صورت خواهد گرفت) و تغییرات آن‌ها را تحت نظر قرار خواهیم داد.
سپس آی دی این کنترل انتخابی را دریافت می‌کنیم (از این ID برای تولید ID برچسب مورد نظر استفاده خواهیم کرد).
در ادامه مقدارهای text و value گزینه انتخابی دریافت می‌شوند (+).
سپس بررسی خواهیم کرد که آیا برچسبی با ID مشخص شده ما وجود دارد (در صورت انتخاب آیتم‌های دیگر، نباید برچسبی غیر منحصربفرد و تکراری در صفحه ایجاد کرد)
در ادامه اگر این مقدار null نبود و همچنین مقدار text و value هم یکی نبودند (اگر یکی بودند لزوم وجود این برچسب بی معنا است)، با استفاده از متد after کتابخانه jQuery یک برچسب را تولید و مقدار مورد نظر را پس از محل نمایش دراپ داون خود، نمایش خواهیم داد.

بهبود کد:

صورت مساله: اکنون نیاز است بجز ناحیه mainFormReq، به سه ناحیه دیگر نیز این تغییرات اعمال گردد. آیا باید همین مقدار کد را سه بار دیگر copy/paste کرد؟

روش صحیح انجام اینکار در jQuery ، نوشتن یک افزونه بر اساس کدهای فوق است که روش انجام آن به صورت زیر می‌باشد (+):

```
//<![CDATA[
(function($) {
   $.fn.dropdownlabel = function() {
       return this.change(function() {
           var obj = $(this);
           var currentId = obj.attr("id"); //آی دی شیء جاری
           var val = obj.val();        //مقدار
           var text = $('#' + currentId + ' option:selected').text(); //متن
           $("#lbl" + currentId).remove();        //اگر نمونه‌ی قبلی موجود است حذف شود
           if (val && (val.length > 0) && (text != val)) {
               //اگر متن و مقدار یکی نیست نمایش داده شود
               obj.after('<label id="lbl' + currentId + '">' + val + '</label>');
           }
       });
   };
})(jQuery);
//]]>
```

و در نهایت نحوه استفاده از آن (فایلی به نام jquery.dropdownlabel.js) به صورت زیر خواهد بود:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="TestDropdownlabel.aspx.cs"
   Inherits="testWebForms87.TestDropdownlabel" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>

   <script src="jquery.min.js" type="text/javascript"></script>
   <script src="jquery.dropdownlabel.js" type="text/javascript"></script>

   <script type="text/javascript">
       $(document).ready(function() {
           $("#mainFormReq select").dropdownlabel();
       });
   </script>

</head>
<body>
   <form id="form1" runat="server">
   <div id="mainFormReq">
       <asp:DropDownList ID="ddlActive" runat="server">
           <asp:ListItem Value=""></asp:ListItem>
           <asp:ListItem Value="Active">فعال</asp:ListItem>
           <asp:ListItem Value="Inactive">غیرفعال</asp:ListItem>
       </asp:DropDownList>
   </div>
   </form>
</body>
</html>
```

عنوان: **NiftyDotNet!**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۲۴ ۲۱:۲۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

NiftyDotNet یک کنترل ASP.Net 2.0 است که امکان ساخت قطعاتی با گوشه‌های گرد را به سهولت میسر می‌سازد. این کنترل در حقیقت محصور کننده‌ی مجموعه Nifty Corners Cube است.

صفحه‌ی اصلی در گوگل کد
یک سری مثال کاربردی
دریافت سورس، مثالها و فایل باینری آن از رپیدشیر

مثال:
گرد کردن گوشه‌های div ایی با id مساوی box1 که توسط خاصیت Selectors این کنترل مشخص شده است (اگر class این div برای مثال مساوی cls1 بود این مقدار می‌بایست مساوی div.cls1 قرار می‌گرفت؛ شبیه به روش jQuery).

```
<%@ Register Assembly="NiftyDotNet" Namespace="NiftyDotNet" TagPrefix="cc1" %>

   <div id="container">
       <cc1:Nifty ID="Nifty1" runat="server" CornerSize="Big" Selectors="div#box1">
       </cc1:Nifty>
       <div id="box1" style="background: #e6e6e6; color: Black; width: 18em;">
           <h1>
               NiftyDotNet</h1>
           <p>
               Just drag & drop to round!</p>
       </div>
   </div>
```

نظرات خوانندگان

نویسنده: MDP
تاریخ: ۱۳۸۸/۰۸/۲۴ ۲۲:۵۴:۴۸

سلام!

شبیه روش jQuery هست چون از همون روش استفاده میکنه.خود کتاب خونه منظورم نیست.روش گرد کردن منظورمه.

موفق باشید.
یادآوری خوبی بود...

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۲۵ ۰۰:۱۰:۲۷

سلام
لینک گوگل کد مشکلی نداره فقط ما رو فیلتر کردند.

نویسنده: MDP
تاریخ: ۱۳۸۸/۰۸/۲۵ ۰۲:۱۷:۰۶

بله شرمنده ، من حواسم نبود که VPN کانکت نیست.

جناب نصیری من چند وقت پیش هم دنبال یه روشی گشتم که با همین پلاگین عکس ها رو با گوشه گرد تولید کنم ، توی همه مرورگر ها غیر از IE موفق شدم.شما راهی غیر از روش من به ذهنتون نمیرسه ؟

این روش منه : http://www.irrc.ir/nifty.txt

البته این روش رو توی وبلاگ خودش گفته بود !

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۲۵ ۱۰:۳۵:۰۴

سلام
در مثال شماره 3 آن از تصویر هم استفاده کرده با یک نکته که خاصیت InnerTransparent آن‌را به true تنظیم کرده. با IE هم تست کردم مشکلی نبود.

نویسنده: MDP
تاریخ: ۱۳۸۸/۰۸/۲۵ ۱۸:۰۵:۱۵

امان از چشم کور :دی !

نویسنده: Pooya Rowshan
تاریخ: ۱۳۸۸/۰۸/۲۶ ۱۹:۰۰:۱۲

با سلام و تشکر. این کامپوننت بسیار عالی و کار آمد در زمانی که برای div ها height تعیین کنید دچار مشکل اساسی می شود. پس دقت کنید بدون div از آن استفاده کنید.

عنوان: JSLint.VS
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۲۶ ۱۹:۳۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

JSLint.VS افزونه‌ای است رایگان برای VS.Net2005/2008 جهت بررسی ساده‌تر مشکلات دستوری در فایل‌های JavaScript یک پروژه.

اکنون بجای اینکه در مرورگر به دنبال خطاهای گزارش شده بگردیم، می‌توان پیش از بررسی نهایی آن‌ها، در VS.Net مشکلات

ممکن را یافته و برطرف ساخت.
JSLint.VS به منوی کلیک راست بر روی یک فایل js اضافه می‌شود و یا در حالت انتخاب قطعه‌ای کد و سپس کلیک راست و بررسی مشکلات موجود و یا در حالت یکپارچه با امکانات build پروژه قابل استفاده است (برای یکپارچه سازی با Build باید به منوی Tools قسمت JSLint.VS options مراجعه کرد و سپس گزینه build را درصفحه ظاهر شده تیک زد).

مشاهده پروژه JSLint.VS در سایت CodePlex

پس از دریافت آن، محتویات پوشه bin آن‌را در مسیر زیر کپی نمائید:

```
%My Documents%/Visual Studio 2008/Addins
Or
%My Documents%/Visual Studio 2005/Addins
```

نظرات خوانندگان

نویسنده: Parham,پرهام
تاریخ: ۱۳۸۸/۰۹/۰۲ ۰۹:۳۹:۳۲

من پوشه Addins را در مسیری که گذاشتید ندارم! باید دستی ساخت یا کار دیگری باید کرد؟

ممنون از افزونه شما.

راستی چرا فقط اشتراک گوگل را میشه برای عنوان نظر انتخاب کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۰۲ ۰۹:۵۱:۳۵

- آیا نگارش VS.Net شما از افزونه‌ها پشتیبانی می‌کند؟
- این پوشه در سیستم من در این مسیر قرار دارد:
C:\Documents and Settings\vahid\My Documents\Visual Studio 2008\Addins

نویسنده: Parham,پرهام
تاریخ: ۱۳۸۸/۰۹/۰۳ ۱۲:۰۰:۰۲

فکر نکنم VS 2008 از افزونه‌ها پشتیبانی نکند. در لینک زیر مسیرهای Addins در سیستم من دیده میشه. نمیدانم در کدام یک باید کپی کنم. و اینکه شاید باید با دکمه Add مسیر را برایش تعریف کنم.

http://cid-7e8f37f7890fc656.skydrive.live.com/self.aspx/Public/Image.JPG

سپاس گذارم . . .

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۰۳ ۱۴:۲۴:۱۵

سلام
فقط نسخه‌ی حرفه‌ای یا Team suite و امثال آن امکان استفاده از پلاگین‌ها را دارند
http://msdn.microsoft.com/en-us/library/zcbsd3cz%28VS.80%29.aspx

+ معرفی پلاگین‌ها چندین راه دارد:
الف) مسیر My Documents\Visual Studio 2008\Addins را دستی درست کنید و فایل‌ها را در آنجا کپی کنید.
ب) یک مسیر دلخواه را ایجاد کنید (به همراه این فایل‌ها). سپس این مسیر باید در همان منویی که تصویرش را مشخص کردید به صورت صریح ذکر شود.
مثلا Resharper به روش ب عمل می‌کند.

عنوان: **چند خبر کوتاه در مورد jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۱۵ ۱۳:۴۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

نگارش جدید جی‌کوئری (jQuery 1.4 Alpha 1) منتشر شد : + و +
انتخاب jQuery به عنوان بهترین کتابخانه‌ی سورس باز سال 2009 از طرف مجله‌ی دات نت: +
بهترین افزونه‌های jQuery سال 2009 : +

نظرات خوانندگان

نویسنده: peyman naji
تاریخ: ۱۳۸۸/۰۹/۱۵ ۲۰:۰۷:۲۶

خبر بسیار عالی بود .
تشکر .

عنوان: **یک دست سازی ی و ک دریافتی در صفحات وب**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۱/۰۶ ۱۰:۱۱:۰۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript

با استفاده از jQuery ، تحت نظر قرار دادن ورودی‌های کاربران در تمام فیلدهای ورودی صفحه کار ساده‌ای است؛ اما جایگزینی مثلا ی فارسی با ی عربی و برعکس درست در لحظه‌ی تایپ آن‌ها کار ساده‌ای نیست و هر مرورگر روش خاص خودش را دارد و بعضی‌ها هم اصلا اجازه‌ی تغییر رخدادهای رسیده را نمی‌دهند.

اسکریپت زیر کار یک دست سازی ی و ک دریافتی در صفحات وب را انجام می‌دهد (برای مثال اگر کاربر ی تایپ کند به صورت خودکار به ی تبدیل می‌شود):

```
// <![CDATA[
function substituteCharInFireFox(charCode, e) {
   var keyEvt = document.createEvent("KeyboardEvent");
   keyEvt.initKeyEvent("keypress", true, true, null, false, false, false, false, 0, charCode);
   e.target.dispatchEvent(keyEvt);
   e.preventDefault();
}

function substituteCharInChrome(charCode, e) {
   //it does not work yet! /*$.browser.webkit*/
   //https://bugs.webkit.org/show_bug.cgi?id=16735
   var keyEvt = document.createEvent("KeyboardEvent");
   keyEvt.initKeyboardEvent("keypress", true, true, null, false, false, false, false, 0, charCode);
   e.target.dispatchEvent(keyEvt);
   e.preventDefault();
}

function insertAtCaret(myValue, e) {
   var obj = e.target;
   var startPos = obj.selectionStart;
   var endPos = obj.selectionEnd;
   var scrollTop = obj.scrollTop;
   obj.value = obj.value.substring(0, startPos) + myValue + obj.value.substring(endPos,
obj.value.length);
   obj.focus();
   obj.selectionStart = startPos + myValue.length;
   obj.selectionEnd = startPos + myValue.length;
   obj.scrollTop = scrollTop;
   e.preventDefault();
}

$(document).ready(function () {
   $(document).keypress(function (e) {

       var keyCode = e.keyCode ? e.keyCode : e.which;
       var arabicYeCharCode = 1610;
       var persianYeCharCode = 1740;
       var arabicKeCharCode = 1603;
       var persianKeCharCode = 1705;

       if ($.browser.msie) {
           switch (keyCode) {
               case arabicYeCharCode:
                   event.keyCode = persianYeCharCode;
                   break;
               case arabicKeCharCode:
                   event.keyCode = persianKeCharCode;
                   break;
           }
       }
       else if ($.browser.mozilla) {
           switch (keyCode) {
               case arabicYeCharCode:
                   substituteCharInFireFox(persianYeCharCode, e);
                   break;
               case arabicKeCharCode:
                   substituteCharInFireFox(persianKeCharCode, e);
                   break;
           }
       }
```

```
        else {
            switch (keyCode) {
                case arabicYeCharCode:
                    insertAtCaret(String.fromCharCode(persianYeCharCode), e);
                    break;
                case arabicKeCharCode:
                    insertAtCaret(String.fromCharCode(persianKeCharCode), e);
                    break;
            }
        }
    });
});
// ]]>
```

تابع substituteCharInChrome قرار است در نگارش‌های آتی گوگل کروم کار کند! کروم فعلا هر نوع شبیه سازی فشرده شدن کلیدهای صفحه کلید را به صفر ترجمه می‌کند. به همین جهت از روش insertAtCaret در مورد آن استفاده شد. هر دو تابع substituteChar ذکر شده در مورد فایرفاکس و کروم و یا روش ساده IE (با توجه به اینکه keyCode در IE فقط خواندنی نیست)، با اپرا کار نمی‌کنند!

دریافت این اسکریپت: (+)
نسخه‌ی فشرده شده آن: (+)
یک پروژه‌ی ساده ASP.NET نمونه در مورد استفاده از آن: (+)

این اسکریپت با IE، فایرفاکس، اپرا ، کروم گوگل و Safari شرکت اپل سازگار است و تفاوتی هم نمی‌کند که در یک html ساده استفاده شود یا در صفحات JSP ، ASP.NET ، PHP ، ASP یا هر چی!

مطالب مشابه:
مشکل ی و ک فارسی و عربی در یک دیتابیس اس کیوال سرور
یکسان سازی ی و ک دریافتی حین استفاده از NHibernate
یکسان سازی "ی" و "ک" دریافتی در حین استفاده از Entity framework
یکسان سازی "ی" و "ک" دریافتی در حین استفاده از WCF RIA Services

نظرات خوانندگان

نویسنده: شاهین کیاست
تاریخ: ۱۳۸۹/۱۱/۰۸ ۱۵:۴۱:۵۶

سلام آقای نصیری.
خسته نباشید.
پس از این که سورس شما را دیدم کلمات کلیدی آن را که متوجه نمی شدم جستجو کردم و خیلی چیزا یاد گرفتم ممنون :)
از این لینک شما به لینک * مشکل ی و ک فارسی و عربی در یک دیتابیس اس کیوال سرور رسیدم.
پس از اجرای آن script روی دیتابیس و قرار دادن این Script در نرم افزار مشکل ی و ک برای جستجو به کل حل می شود درسته؟ و نیاز به DLL هماهنگ کننده ای که معرفی کرده بودید نیست؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۱/۰۸ ۱۷:۰۱:۱۲

سلام، اون dll مخصوص NHibernate بود و باشد بهتر است. چون اسکریپت SQL را که اجرا می‌کنید در این لحظه همه چیز یکسان خواهد شد، نه چند روز دیگر با اطلاعات جدیدتر.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۱/۱۰ ۲۰:۱۶:۳۴

فایل‌های جاوا اسکریپت فوق را جهت در نظر گرفتن حالت Paste متن هم به روز کردم که از همان آدرس‌های قبلی قابل دریافت است.

عنوان: **معرفی برنامه jQueryPad**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۲/۰۷ ۰۹:۰۰:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

jQueryPad
برنامه‌ای است قابل حمل و کم حجم برای تمرین و همچنین آزمایش سریع اسکریپت‌های مبتنی بر jQuery . می‌توان در قسمت سمت چپ آن قطعه کد HTML ایی را نوشت و در TextBox سمت راست آن، قطعه کد جاوا اسکریپتی مبتنی بر jQuery . سپس با فشردن دکمه F5 بلافاصله نتیجه‌ی ترکیب این دو در پایین صفحه نمایش داده می‌شود.

اگر نیاز به استفاده از نگارش خاصی از jQuery را داشتید می‌توان فایل js موجود در پوشه‌ی template آن‌را تغییر داد.
کار انجام شده شبیه به برنامه‌ی LINQ-Pad است؛ البته برنامه‌ی LINQ-Pad در نگارش تجاری آن دارای intellisense هم هست.

نظرات خوانندگان

نویسنده: مجتبی صحرائی
تاریخ: ۱۳۸۹/۱۲/۰۸ ۱۷:۴۴:۲۰

سلام
ممنون برنامه مفیدیه

نویسنده: Anonymous
تاریخ: ۱۳۸۹/۱۲/۱۶ ۲۳:۰۴:۲۸

jsfiddle را دیده اید؟ به نظرم شاهکار است...

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۱۲/۱۶ ۲۳:۲۸:۰۴

ممنون. این هم جالب بود: (+)
فقط مثل برنامه فوق نسخه‌ی آفلاین ندارد.

عنوان: **راه حل ساده برای عکس‌هایی که از کادر بیرون می‌زنند**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۲۸ ۲۱:۱۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

ارسال تصویر در سایت‌ها، انجمن‌ها و امثال آن همیشه مشکل زا است؛ خصوصا نمایش تصاویری در قطع یک تابلوی دیواری در یک کادر کوچک.

راه ساده و دم دستی زیر، منهای استفاده از اسکریپت‌هایی که اندازه تصاویر را تشخیص داده و بر این اساس خودشان آن‌ها را بند انگشتی نمایش می‌دهند، هم برای حل این مشکل وجود دارد:

فرض کنید مطلبی که قرار است ارسال شود در یک DIV نمایش داده می‌شود و کلاس آن مثلا post-body است.

اولین کاری که می‌شود انجام داد این است:

در CSS سایت قسمت مرتبط با post-body تعریف زیر را اضافه می‌کنیم تا اضافات عکس که از کادر بیرون زده، خودبخود توسط مرورگر نمایش داده نشود:

```
overflow: hidden;
```

سپس در همین DIV که کلاس آن post-body است، با استفاده از jQuery به دنبال تصاویر گشته و موارد یافت شده را داخل یک لینک پویا قرار می‌دهیم (استفاده از متد wrap برای این محصور سازی). این لینک هم همان آدرس اصلی تصویر است. به این ترتیب کاربر با کلیک بر روی آن تصویر می‌تواند نتیجه را در صفحه‌ای دیگر مشاهده کند:

```
$(document).ready(function(){
 $('.post-body img').each(function(){
    var $img = $(this);
    var src = $img.attr("src");
    $img.attr({border:"0"}).wrap('<a target="_blank" alt="Click here to enlarge (opens new window)"
title="Click here to enlarge (opens new window)" href="'+src+'" />');
 });
});
```

عنوان: **چرا به الگوهای طراحی JavaScript نیاز داریم ؟**
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۱/۰۳/۳۰ ۲۱:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: Design patterns, JavaScript

امروزه اهمیت یادگیری JavaScript بر هیچ کس پوشیده نیست ، APIهای جدید HTML 5 و امکانات جدید وب مثل Geo Location ، Offline Web applications ، Canvas ، Offline storage ، و ... همه نیاز به دانش JavaScript دارند. در نتیجه نیاز است علاوه بر یادگیری JavaScript به تمیز کد نوشتن جهت سهولت نگهداری آگاه بود. همانطور که در کدهای سمت سرور مثل C# و یا PHP نیاز به استفاده از الگوهای طراحی (Design Patterns)است در JavaScript هم اوضاع به همین منوال است.
الگوی طراحی یک راه حل قابل استفاده مجدد است که برای حل مشکلات متداول در طراحی نرم افزار به کار می‌رود.
چرا به الگوی‌های طراحی JavaScript نیازمندیم ؟
می خواهیم کد هایی با قابلیت استفاده‌ی مجدد بنویسیم ، استفاده از عملکردهای مشابه در سطح صفحات یک Web application یا چند Web Application.
می خواهیم کد هایی با قابلیت نگهداری بنویسیم ، هر چه قدر در فاز توسعه کدهای با کیفیت بنویسیم در فاز نگهداری از آن بهره می‌بریم. باید کد هایی بنویسیم که قابل Debug و خواندن توسط دیگر افراد تیم باشند.
کدهای ما نباید با توابع و متغیرهای دیگر پلاگین‌ها تداخل نامگزاری داشته باشند. در برنامه‌های امروزی بسیار مرسوم است که از پلاگین‌های Third party استفاده شود. می‌خواهیم با رعایت Encapsulation and modularization در کدهایمان از این تداخل جلوگیری کنیم.

معمولا کدهای JavaScript که توسط اکثر ما نوشته می‌شود یک سری تابع پشت سرهم هست ، بدون هیچ کپسوله سازی :

```
function getDate() {
    var now = new Date();
    var utc = now.getTime() + (now.getTimezoneOffset() * 60000);
    var est;
    est = new Date(utc + (3600000 * -4));
    return dateFormat(est, "dddd, mmmm dS, yyyy, h:MM:ss TT") + " EST";

}
function initiate_geolocationToTextbox() {
    navigator.geolocation.getCurrentPosition(handle_geolocation_queryToTextBox);
}

function handle_geolocation_queryToTextBox(position) {
    var longitude = position.coords.longitude;
    var latitude = position.coords.latitude;
    $("#IncidentLocation").val(latitude + " " + longitude);
}
```

به این روش کدنویسی Function Spaghetti Code گفته می‌شود که معایبی دارد :
توابع و متغیرها به Global scope برنامه افزوده می‌شوند.
کد Modular نیست.
احتمال رخ دادن Conflict در اسامی متغیرها و توابع بالا می‌رود.
نگهداری کد به مرور زمان سخت می‌شود.

با شبیه سازی یک مثال مشکلات احتمالی را بررسی می‌کنیم :

```
// file1.js
function saveState(obj) {
    // write code here to saveState of some object
    alert('file1 saveState');
}
// file2.js (remote team or some third party scripts)
function saveState(obj, obj2) {
     // further code...
    alert('file2 saveState");
}
```

همانطور که می‌بینید در این مثال در 2 فایل متفاوت در برنامه مان از 2 تابع با اسامی یکسان و امضای متفاوت استفاده کرده ایم .
اگر فایل‌ها را اینگونه در برنامه آدرس دهی کنیم :

```
<script src="file1.js" type="text/javascript"></script>
<script src="file2.js" type="text/javascript"></script>
```

متد saveState در فایلی که دیرتر آدرس داده شده (file2.js) ، متد saveState در file1.js را Override می‌کند ، در نتیجه عملکردی که از متد saveState در فایل اول انتظار داریم اتفاق نمی‌افتد.
در پست بعدی به راه حل این مشکلات و کپسوله سازی خواهم پرداخت.
برای مطالعه‌ی بیشتر کتاب (Learning JavaScript Design Patterns) را از دست ندهید.

نظرات خوانندگان

نویسنده: NargesM
تاریخ: ۱۳۹۱/۰۳/۳۱ ۶:۸

سلام.. ممنون

عنوان: **30Days to Learn jQuery**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۴:۳۴
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, links

چند وقت پیش سایت معروف tutsplus.com یه دوره 30 روزه آموزش مبتدی تا حرفه ای jQuery رو به صورت رایگان برگزار کرد،در این دوره آقای Jeffrey Way (مدرس دوره) در 30 روزه jQuery رو از مبتدی تا حرفه ای و به صورت کاربردی آموزش میده، در کل مجموعه جالبه (البته خودم تا حالا وقت نشده شروع کنم برای شروع هم اگر وقت شد میخوام این مجموعه رو با کمک دوستان زیرنویس کنم اگر مایل به همکاری بودید به من ایمیل بزنید sirlafifi@gmail.com)پیشنهاد می‌کنم حتما دانلود کنید. مشاهده سرفصل‌ها و دانلود

نظرات خوانندگان

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۹:۵۶

من هم پیشنهاد میکنم این فیلمهای آموزشی رو به هیچ عنوان از دست ندین! ... از نظر من بهترین فیلمهای آموزشی در زمینه‌ی jquery هستش (با تدریس عالی Jeffrey way)

نویسنده: مهدی
تاریخ: ۱۳۹۱/۰۴/۰۱ ۷:۵۸

دست مریزاد

عالی بود

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۷:۲

سلام من حدود 100تا دوره pluralsight رو دارم جدید هم هست اگه خواستید به Sh.mjsoft[at]gmail.com ایمیل بزنید

نویسنده: orkideh
تاریخ: ۱۳۹۲/۰۵/۱۵ ۱۶:۱۵

مرسی!واقعا" عالیه.من jquery رو دارم از صفر شروع میکنم.فیلمارو هم بی صدا میبینم و بازم عالی بوده.الان جلسه دهمم و خیلی خوب یادگرفتم.ممنون

عنوان: Closure در JavaScript
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۱/۰۴/۰۴ ۷:۱۸
آدرس: www.dotnettips.info
گروه‌ها: Design patterns, JavaScript, Closure

در قسمت قبلی درباره علت نیاز به الگوهای طراحی در JavaScript و Function Spaghetti code صحبت شد. در این قسمت Closure در JavaScript مورد بررسی قرار می‌گیرد.

در JavaScript می‌توان توابع تو در تو نوشت (nested functions) ، زمانی که یک تابع درون تابع دیگر تعریف می‌شود تابع درونی به تمام متغیرها و توابع تابع بیرونی (Parent) دسترسی دارد.

Douglas Crockford

برای تعریف Closure می‌گوید :

```
an inner function always has access to the vars and parameters of its outer function, even after the
outer




function has returned
```

یک تابع درونی (nested) همیشه به متغیرها و پارامترها تابع بیرونی دسترسی دارد ، حتی اگر تابع بیرونی مقدار برگردانده باشد.

تابع زیر را در نظر بگیرید :

```
// The getDate() function returns a nested function which refers the 'date' variable defined
// by outer function getDate()
function getDate() {
   var date = new Date();    // This variable stays around even after function returns

   // nested function
   return function () {
      return date.getMilliseconds();
   }
}
```

اکنون اگر به صورت زیر تابع getDate فراخوانی شود مشاهده می‌شود که تابع درونی (با کامنت nested function مشخص شده است.) به شیء date دسترسی دارد.

```
// Once getDate() is executed the variable date should be out of scope and it is, but since
// the inner function
// referenes date, this value is available to the inner function.
var dt = getDate();

alert(dt());
alert(dt());
```

خروجی هر 2 alert یک مقدار خواهد بود.

اگر از فردی که به تازگی رو به JavaScript آورده است خواسته شود تابعی بنویسد که میلی ثانیه‌ی زمان جاری را برگداند احتمالا همچین کدی تحویل می‌دهد :

```
function myNonClosure() {
            var date = new Date();
            return date.getMilliseconds();
        }
```

در کد بالا پس از اجرای myNonClosure متغیر date از بین خواهد رفت ، این مسئله در دنیای JavaScript طبیعی هست.
این مثال را در نظر بگیرید :

```
var MyDate = function () {
    var date = new Date();
    var getMilliSeconds = function () {
        return date.getMilliseconds();
    }
}

var dt = new MyDate();

alert(dt.getMilliSeconds());  // This will throw error as getMilliSeconds is not accessible.
```

در صورت اجرای مثال بالا خطایی با این مضمون دریافت خواهد شد که getMilliSeconds دستیابی پذیر نیست. (کپسوله شده)
برای اینکه آن را دستیابی پذیر کنیم کد را به این صورت تغییر می‌دهیم :

```
// This is closure
var MyDate = function () {
    var date = new Date();   // variable stays around even after function returns
    var getMilliSeconds = function () {
        return date.getMilliseconds();
    };
    return {
        getMs : getMilliSeconds
    }
}
```

آنچه در تابع بالا انجام شده کپسوله سازی همه‌ی منطق کار (منطق کار در اینجا برگرداندن میلی ثانیه زمان جاری می‌باشد) در یک فضای نام به نام MyDate می‌باشد. همچنین فقط متدهای عمومی در اختیار استفاده کننده این تابع قرار داده شده است. برای استفاده می‌توان بدین صورت عمل کرد :

```
var dt = new MyDate();
alert(dt.getMs());  // This should work.
```

در کد بالا برای توابع و متغیرهای درونی یک container ایجاد کردیم که باعث جلوگیری از تداخل در نام متغیرها با دیگر کدها خواهد شد . (برای مشاهده‌ی تداخل‌ها به قسمت قبلی توجه کنید.)
اگر بخواهیم Closure را تشبیه کنیم ، Closure شبیه به کلاس‌ها در #C یا Java هست.
Closure یک حوزه (scope) برای متغیرها و توابع درونی خودش ایجاد می‌کند.
jQuery بهترین مثال کاربردی برای Closure می‌باشد :

```
(function($) {

    // $() is available here

})(jQuery);
```

در ادامه این مفاهیم بیشتر توضیح داده می‌شودند ، اکنون می‌خواهیم مشکلی که در قسمت قبلی مطرح کردیم به کمک Closure حل کنیم :

در آن مثال گفته شد که اگر :

```
// file1.js
function saveState(obj) {
    // write code here to saveState of some object
    alert('file1 saveState');
}
// file2.js (remote team or some third party scripts)
function saveState(obj, obj2) {
```

```
     // further code...
    alert('file2 saveState");
}
```

اگر تابعی به نام saveState در 2 فایل مختلف داشته باشیم و این 2 فایل را بدین صورت در برنامه آدرس دهیم :

```
<script src="file1.js" type="text/javascript"></script>
<script src="file2.js" type="text/javascript"></script>
```

تابع saveState در فایل دوم تابع saveState فایل اول را override می‌کند. یک از توابع بالا را به صورت زیر باز نویسی می‌کنیم و منطق کار را کپسوله می‌کنیم :

```
function App() {
    var save = function (o) {
        // write code to save state here..
        // you have acces to 'o' here...
        alert(o);
    };

    return {
        saveState: save
    };
}
```

بدون نگرانی تداخل saveState با بقیه saveStateها در هر پلاگین یا فایل دیگری می‌توان از saveState می‌توان اینگونه استفاده کرد :

```
var app = new App();

app.saveState({ name: "rajesh"});
```

برای اطلاعات بیشتر در مورد Closure ها این لینک را بررسی کنید.

نظرات خوانندگان

نویسنده: MBE
تاریخ: ۱۳۹۱/۰۴/۰۴ ۲۰:۳۱

ممنون عالی بود . اما این بحث واقعا جای کار داره . من که منتظر مقالات بیشتری در مورد Closure هستم .

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۱/۰۴/۰۴ ۲۱:۳۱

خواهش می‌کنم. در ادامه درباره الگوهای Prototype و Module مطلب می‌نویسم و سعی می کنم با یک مثال بحث را بیشتر باز کنم.

نویسنده: صالح
تاریخ: ۱۳۹۱/۰۴/۰۴ ۲۳:۴۸

نکته بسیار خوب و کاربردی در جاوا اسکریپت بود. ممون

نویسنده: پوران
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۵:۵۵

واقعا مرسی ... عالی بود ...

عنوان: **فشرده سازی فایل های CSS و JavaScript بصورت خودکار توسط MS Ajax Minifier**
نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۲۹ ۱۶:۲۹
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Ajax, MS Ajax Minifier, CSS

با توجه به افزایش کاربرد jQuery و دیگر کتابخانه‌های جاوا اسکریپت در برنامه‌های تحت وب، یکی از چالش‌های همیشگی برنامه نویسان، فشرده سازی فایل‌های دربرگیرنده کدهای جاوا اسکریپت و شیوه نامه‌ها می باشد. برای این منظور راه‌های مختلفی مانند استفاده از ابزارهای آنلاین مانند این + و این + وجود دارند. اما یک روش خودکار هم وجود دارد که در زمان Build پروژه‌های دات نت می‌توان از آن بهره گرفت.

Microsoft Ajax Minifier

یک ابزار رایگان جهت فشرده سازی فایل‌های جاوا اسکریپت و شیوه نامه‌ها است. شما می‌توانید این ابزار را از صفحه خانگی آن در سایت asp.net دریافت کنید. جهت استفاده از این ابزار می‌توان از طریق خط فرمان عمل کرد. اما روش ساده‌تر که هدف اصلی این مطلب است به شرح زیر است:

1. در VisualStudio.NET از طریق منو به مسیر **Tools, Options, Projects and Solutions** بروید و گزینه **Always show solution** را تیک بزنید.

2. از Solution Explorer بر روی عنوان پروژه کلیک راست کرده و گزینه **Unload Project** را انتخاب نمایید.

3. مجدداً روی عنوان پروژه کلیک راست کرده و گزینه **Edit** را انتخاب کنید و دستورات زیر را قبل از بسته شدن تگ Project اضافه کنید:

```
<Import Project="$(MSBuildExtensionsPath)\Microsoft\MicrosoftAjax\ajaxmin.tasks" />
<Target Name="AfterBuild">
<ItemGroup>
  <JS Include="**\*.js" Exclude="**\*.min.js;Scripts\*.js" />
</ItemGroup>
<ItemGroup>
  <CSS Include="**\*.css" Exclude="**\*.min.css" />
</ItemGroup>
<AjaxMin
    JsSourceFiles="@(JS)"  JsSourceExtensionPattern="\.js$" JsTargetExtension=".min.js"
    CssSourceFiles="@(CSS)" CssSourceExtensionPattern="\.css$" CssTargetExtension=".min.css"  />
</Target>
```

4. دوباره بر روی عنوان پروژه کلیک راست کرده و گزینه **Reload Project** را انتخاب کنید.

توجه کنید که با این کار ما یک MSBuild task با عنوان ajaxmini به پروژه اضافه کردیم. این وظیفه که در زمان Build پروژه اجرا خواهد شد فایل‌های جاوا اسکریپت را فشرده و با پسوند .min.js و همچنین فایل‌های CSS را پس از فشرده سازی با پسوند .min.css در همان مسیر فایل مادر بطور خودکار ذخیره می‌کند.

نکته:

اگر به دستورات تنظیمات فوق نگاه دقیقتری بیندازیم، متوجه عبارات **Include** و **Exclude** می شویم. توسط این دو صفت شما می‌توانید الگوهایی را جهت فشرده سازی و یا عدم فشرده سازی تعریف کنید. بدین معنا که توسط الگوی‌های ذکر شده در تنظیمات فوق از فشرده سازی فایل‌های با پسوند .min.js و .min.css خودداری می‌شود.

در این شرایط در حین توسعه برنامه، شما می‌توانید از فایل‌های با کد خوانا استفاده نمایید و زمان انتشار و Build پروژه بصورت خودکار آنها را با فایل‌های فشرده جایگزین کنید.

این ابزار تمامی فضاهای خالی، ';' و '{ }'های اضافی و توضیحات را از کدهای شما حذف می‌کند. متغیرها و توابع شما را به اسامی کوچک‌تر تغییر نام می‌دهد. و ...

همچنین شما از کتابخانه این پروژه می‌توانید در زمان اجرا و سورس برنامه خود استفاده کنید. جهت اطلاعات بیشتر می‌توانید به سایت مربوطه مراجعه نمایید.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۹ ۲۲:۴۲

ممنون. من از این کتابخانه استفاده می‌کنم:

Yahoo! UI Library: YUI Compressor for .Net

```
using System.Globalization;
using System.IO;
using System.Text;
using Yahoo.Yui.Compressor;

namespace Deploy.Core
{
    public static class CompressCssJs
    {
        public static void Compress(string file)
        {
            var ext = Path.GetExtension(file).ToLower();
            switch (ext)
            {
                case ".css":
                    compressCss(file);
                    break;
                case ".js":
                    if (!file.ToLower().EndsWith(".min.js") && !file.ToLower().EndsWith(".pack.js"))
                        compressJs(file);
                    break;
            }
        }

        static void compressCss(string file)
        {
            var css = File.ReadAllText(file);
            var compressedCss = new CssCompressor().Compress(css);
            File.WriteAllText(file, compressedCss, Encoding.UTF8);
        }

        static void compressJs(string file)
        {
            var js = File.ReadAllText(file);

            var compressedJavaScript = new JavaScriptCompressor
            {
                CompressionType = CompressionType.Standard,
                DisableOptimizations = false,
                Encoding = Encoding.UTF8,
                LineBreakPosition = -1,
                ObfuscateJavascript = true,
                PreserveAllSemicolons = false,
                ThreadCulture = CultureInfo.CurrentUICulture,
                IgnoreEval = false,
                LoggingType = LoggingType.None
            }.Compress(js);
            File.WriteAllText(file, compressedJavaScript, Encoding.UTF8);
        }
    }
}
```

نحوه استفاده از اون رو باکدنویسی در بالا ملاحظه می‌کنید (ملاحظات utf8 و زبان فارسی هم در آن لحاظ شده).
کاری که هنگام ارائه نهایی انجام می‌دم، اسکن فایل‌های نهایی و بررسی پسوندها و سپس استفاده از متد Compress فوق روی فایل‌های اسکریپت و css یافت شده است.

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۳۰ ۹:۶

ضمن تشکر از آقای نصیری بخاطر معرفی روش دیگر، توجه فرمایید من تنها یک معرفی اجمالی درباره این ابزار داشتم. این ابزار

امکانات دیگری نیز دارد که بطور خاص برخی از آنها به شرح زیر است:
آنالیز فایل‌های جاوا اسکریپت و اعلام هشدارهای مناسب به برنامه نویس. +
اگر از فایل‌های RESX جهت چند زبانه کردن سایت استفاده می‌کنید این ابزار بسیار کارآمد خواهد بود. +
همچنین این ابزار نیز از UTF-8 بخوبی پشتیبانی می‌کند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۳۰ ۹:۱۸

ممنون. ابزار یاهو هم امکان آنالیز (LoggingType) را دارد. فقط در اینجا چون مد نظر من نبوده به none تنظیم شده.

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۳۰ ۹:۳۰

به نظر میاد نسبت به هم در یک سطح باشند. سپاس از مطالب به موقع و بجای شما.

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۳۰ ۱۹:۶

فکر میکنم از این رهیافت در سلوشن‌های مرجع مایکروسافت نشه یاد کرد. در asp.net mvc 4 دو تکنیک Bundle کردن (دسته کردن فایل‌های مشابه در یک دسته) و Minify کردن (از بین بردن فواصل اضافه و کوچک کردن نام متغییرها و حذف کامنت‌ها) اضافه شده. که این کار رو به صورت توکار خود فریم‌ورک انجام میده فقط لازمه با دستورات Syte.Render و Script.Render اونارو آدرس دهی کنیم.
همچنین این قابلیت وجود دارد که در زمان debug فایل‌ها رو خانا و قابل‌فهم مشاهده کنید. در یکی از سری پست‌های آقای شهروز جعفری ^ این مهم اشاره کردن. هرچند پست ایشان اندکی قدیمی است(موقع نگارش beta مطلب منتشر شد که حالا با آمدن نگارش RC اندکی فرق کرده است)

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۳۰ ۱۹:۳۵

با تشکر از آقای رضا.ب....
مطلب آقای جعفری را مطالعه کردم. آیا امکان استفاده از فایل‌های Resource جهت Localization در این تکنیک وجود دارد؟
همچنین آیا فایل‌ها اسکریپت آنالیز می‌شوند و ما از هشدارهای مناسب مطلع می‌شویم؟

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۴/۳۰ ۲۰:۳۹

با تشکر از دوستان بابت تمامی روشهای بحث شده.
میخواستم بدونم برای فشرده سازی یک web site قبل از upload با gzip هم میشه اطلاعات مختصری بدهید.
با تشکر

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۳۰ ۲۳:۳

تا جایی که من مطلع هستم از طریق تنظیمات IIS می‌توان پاسخ به درخواست‌ها را (شامل فایل‌های استاتیک و داینامیک) به کمک gzip فشرده کرد. اما اگر اطلاعاتم صحیح باشد این موضوع کمی بار CPU را افزایش می‌دهد، هر چند گاهی تا 75% حجم اطلاعات رد و بدل شده را کاهش می‌دهد.
برای اطلاعات بیشتر در مورد تنظیمات IIS6 می‌توانید به + و + مراجعه کنید.
همچنین اگر روی IIS7 به بعد میزبانی می‌شوید می‌توانید درون فایل Web.Config و درون تگ system.webServer تنظیمات زیر را اضافه کنید:

```
<httpCompression directory="%SystemDrive%\inetpub\temp\IIS Temporary Compressed Files">
    <scheme name="gzip" dll="%Windir%\system32\inetsrv\gzip.dll" staticCompressionLevel="9"
dynamicCompressionLevel="4" />
    <scheme name="deflate" dll="%Windir%\system32\inetsrv\gzip.dll" staticCompressionLevel="9"
dynamicCompressionLevel="4" />
    <dynamicTypes>
        <add mimeType="text/*" enabled="true" />
        <add mimeType="message/*" enabled="true" />
        <add mimeType="application/x-javascript" enabled="true" />
        <add mimeType="application/atom+xml; charset=utf-8" enabled="true" />
        <add mimeType="*/*" enabled="false" />
    </dynamicTypes>
    <staticTypes>
        <add mimeType="text/*" enabled="true" />
        <add mimeType="message/*" enabled="true" />
        <add mimeType="application/javascript" enabled="true" />
        <add mimeType="*/*" enabled="false" />
    </staticTypes>
</httpCompression>
```

(برای مطالعه بیشتر این + را ببینید.)

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۴/۳۱ ۸:۱۴

ممنون از جواب خوبتون
پس بطور کلی بهتر است که اینکارو انجام ندهیم؟

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۳۱ ۹:۵۱

من پیشنهاد می‌کنم اگر شرایطش رو دارید، امتحانش کنید. حتی برای تجربه شده! و خوشحال میشم نتیجه تجربه تان را با ما به اشتراک بگذارید.

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۳۱ ۱۹:۱

تا جایی که اطلاع دارم، تکنیک B/M فقط باعث کاهش میزان درخواست‌های HTTP بر روی وب سرور میشود. (آموزش نسخه MVC 4 RC ^)

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۵/۱۶ ۱۱:۱۹

سلام
مهندس نصیری این روشی که شما فرمودین
در زمان اجرا خروجی فشرده میکنه یا اینکه نه فشرده میکنه و همونجا هم ذخیره میکنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۶ ۱۳:۵۴

هنگام ارائه نهایی (در پوشه‌ای جداگانه از فایل‌های اصلی) یکبار روی فایل‌ها اجرا می‌شود.

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۹/۰۶ ۸:۳۹

سلام

به نظر شما بهتر نیست این کدها در یک هندلر استفاده بشه که در زمان اجرا خروجی فشرده شده برای کاربر ارسال بشه با توجه به کش شدن این نوع فایل‌ها ایا سربار سیستم اضافه میشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۶ ۹:۵۶

سیستم bundling جدید MVC4 در زمان اجرا این‌کارها را انجام می‌دهد. کش شدن هم سربار سیستم رو کم می‌کنه چون درخواست جدیدی به سرور ارسال نخواهد شد.

نویسنده: سیاه
تاریخ: ۱۳۹۱/۱۱/۱۸ ۵:۲۷

سلام؛ راهی وجود داره که لینک فایلهای css و js رو در source page مرورگر غیرفعال کرد ؟
و امکان اینکه فرد با کلیک روی لینک محتوای فایلهای css و js رو نبینه .
با تشکر .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۸ ۹:۹

خیر. مرورگر باید به این لینک‌ها دسترسی داشته باشه برای رندر سایت. به همین نحو این دسترسی برای تمام دنیا وجود دارد.

نویسنده: مرتضی مختاری
تاریخ: ۱۳۹۲/۰۳/۲۳ ۲۳:۵۲

سلام؛ اگه از کش کردن اطلاعات استاتیک استفاده کنیم دیگه نیازی به minify کردن فایل‌ها داریم یا خیر؟

عنوان: آموزش فایرباگ - JavaScript Development - 3#
نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۴/۳۱ ۲:۲۱
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, xml, FireBug

توابع توسعه جاوا اسکریپت در فایرباگ به دو بخش تقسیم می‌شوند :

توابع خط فرمان - Command Line API

توابع کنسول - Console API

توابع خط فرمان توابعی هستند که فقط در خط فرمان قابل استفاده هستند و توابع کنسول هم توابعی هستند که خارج از محیط خط فرمان (، در بین کدهای جاوا اسکریپت برنامه) هم قابل استفاده هستند .
در این قسمت توابع خط فرمان را بررسی خواهیم کرد و در قسمت بعدی با توابع کنسول آشنا خواهیم شد .

توابع خط فرمان - Command Line API :

این توابع حدود 14 تا هستند که بوسیله آنها می‌توانیم در حین اجرای برنامه تست‌های مختلفی انجام داده یا اطلاعاتی از قسمت‌های مختلف بدست آوریم .
توجه : برای همراه شدن با تست‌های انجام شده در این مقاله می‌توانید کد صفحه‌ی زیر را ذخیره کنید و برای اجرای کدها ، آن‌ها را در قسمت خط فرمان (در تب کنسول) قرار بدهید و دکمه‌ی Run (یا Ctrl + Enter) را بزنید .

```
<div id="first" class="content">Content1 with css class and id</div>
<div class="content">
    Content2 with css class
    <a class="links" href="#">Link1</a>
    <a href="#">Link2</a>
</div>
<div>
    Content3 without css class and id
</div>
<input type="button" onclick="myFunc()" value="Run myFunc" />
<input type="text" id="myInput" />

<script type="text/javascript">
    function myFunc() {
        loop(1000);
        loop(50000);
    }
    function loop(number) {
        for (var i = 0; i < number; i++) { }
    }
</script>
```

قبل از توضیح این توابع ، یک مثال ساده می‌زنیم :
فرض کنید می‌خواهید همه‌ی المنت هایی که با یک selector مطابقت دارند را مشاهده کنید . (آرایه ای از المنت‌ها دریافت کنید)

```
$$("div.content");
```

نتیجه :

```
>>> $$("div.content");
[ div#first.content, div.content ]
```

می خواهید یکی از توابع جاوا اسکریپت برنامه را اجرا و آن را از لحاظ سرعت ، تعداد فراخوانی شدن و ... بررسی کنید .

```
profile("myFunc Testing");
myFunc();
profileEnd();
```

نتیجه :

```
>>> profile("myFunc Testing"); myFunc(); profileEnd();
```

myFunc Testing (0.291ms, 3 calls)

Function	Calls	Percent	Own Time	Time	Avg	Min	Max	File
loop	2	95.19%	0.277ms	0.277ms	0.139ms	0.123ms	0.154ms	1.htm (line 26)
myFunc	1	4.81%	0.014ms	0.291ms	0.291ms	0.291ms	0.291ms	1.htm (line 22)

اکنون با همه‌ی توابع خط فرمان آشنا می‌شویم :

$(id)

معادل دستور document.getElementById است که یک المنت با id داده شده بر می‌گرداند .

```
$("first");
```

نتیجه :

```
>>> $("first")
<div id="first" class="content">
```

$$(selector)

آرایه ای از المنت‌های مطابق با selector داده شده بر می‌گرداند .

```
$$("div.content")
```

نتیجه :

```
>>> $$("div.content")
[ div#first.content, div.content ]
```

به تفاوت دو دستور توجه کنید . خروجی دستور اول ، یک المنت است و خروجی دستور دوم یک آرایه از المنت که بین [و] قرار گرفته اند .

برای آشنایی بیشتر با CSS Seletorها به این لینک مراجعه کنید : http://www.w3.org/TR/css3-selectors

$x(xPathExpression)

آرایه ای از المنت هایی را بر می‌گرداند که با xPath داده شده مطابقت داشته باشند .

```
var objects = $x("html/body/div[2]/a")

for(var i = 0; i < objects.length; i++) {
   console.log(objects[i]);
}
```

نتیجه :

```
>>> var objects = $x("html/body/div[2]/a")  for(var ...cts.length; i++) {
console.log(objects[i]); }
<a class="links" href="#">
<a href="#">
```

برای آشنایی بیشتر با عبارات xPath به این لینک مراجعه کنید : http://www.w3schools.com/xpath

dir(object)

تمام خصوصیات شیء ارسال شده را لیست می‌کند .

```
var objects = $x("html/body/div[2]/a")
dir(objects);
```

نتیجه :

```
>>> var objects = $x("html/body/div[2]/a")  dir(objects);
⊞ 0                          a.links #
⊟ 1                          a #
    addEventListener         addEventListener()
    appendChild              appendChild()
    blur                     blur()
    cloneNode                cloneNode()
    compareDocumentPosition  compareDocumentPosition()
    dispatchEvent            dispatchEvent()
    focus                    focus()
```

(dirxml(node

سورس یک المنت را بصورت درختواره (tree) پرینت می‌کند . همچنین با کلیک بروی هر node ، فایرباگ آن node را در تب html نمایش می‌دهد .

```
var node = $("first");
dirxml(node);
```

نتیجه :

```
>>> var node = $("first"); dirxml(node);
<div id="first" class="content">
   Content1 with css class and id
</div>
```

توجه کنید که این دستور فقط یک node دریافت می‌کند . برای همین اگر از دستور $$("#first") استفاده می‌کنید ، چون این دستور یک آرایه بر می‌گرداند ، باید اولین عضو آرایه را دریافت و ارسال کنید .
یعنی :

```
var node = $$("#first")[0];
dirxml(node);
```

()clear

این دستور محیط console را خالی می‌کند . عملکرد این دستور معادل کلیک دکمه‌ی Clear (در بالا - چپ تب کنسول) است .

([inspect(object[,tabName

توسط این دستور می‌توانید یک شیء را در مناسب‌ترین تب فایرباگ یا یکی از تب‌های مورد نظر خود ، Inspect کنید .

```
var node = $("first");

inspect(node); // inspect in html tab
inspect(node,'dom'); // inspect in dom tab
```

(keys(object

آرایه ای از "نام" تمام خصوصیات شیء ارسال شده بر می‌گرداند .

```
var obj = $("first");
keys(obj)
```

(values(object

آرایه ای از "مقدار" تمام خصوصیات شیء ارسال شده بر می‌گرداند .

```
var obj = $("first");
values(obj)
```

(debug(fn) and undebug(fn

این متدها یک BreakPoint در ابتدای تابع مشخص شده اضافه/حذف می‌کنند . (در تب Script) . به همین ترتیب هنگامی که تابع مورد نظر فراخوانی شود ، در نقطه ای که BreakPoint قرار داده شده توقف خواهد کرد .
البته می‌شود BreakPoint را دستی هم قرار داد . در اصل این تابع ، این عملیات را ساده‌تر می‌کند .

```
debug(myFunc);
myFunc();
undebug(myFunc);
```

(monitor(fn) and unmonitor(fn

این متدها برای فعال/غیرفعال کردن Logging فراخوانی‌های یک تابع استفاده می‌شوند .
در حالت عادی برای پی بردن به اینکه یک تابع اجرا می‌شود یا نه ، در تابع مورد نظر یک alert قرار می‌دهیم و تست می‌کنیم . که این روش در برنامه برنامه‌های بزرگ صحیح نیست . زیرا در این حالت باید بین حجم زیادی کد به دنبال تابع مورد نظر بگردیم و سپس alert را قرار بدهیم و بعد اطمینان از صحت عملکرد تابع مجدد آن را حذف کرد ، که با اتلاف زمان و به خطر انداختن کدها همراه است .
اما با استفاده از این متدها ، تنها نیاز به داشتن اسم تابع داریم (و نه مکان تابع در کدهای برنامه) .
تست monitor :

```
monitor(myFunc);
// now click on "Run myFunc" button
```

تست unmonitor :

```
unmonitor(myFunc);
// now click on "Run myFunc" button
```

([monitorEvents(object[, types]) and unmonitorEvents(object[, types

این متدها عملیات Event Logging برای یک شیء را فعال/غیرفعال می‌کنند . در کنار شیء مورد نظر ، می‌توان نوع رویداد را هم به

متد ارسال کرد . در این صورت عملیات Logging فقط برای همان گروه رویداد/رویداد ، فعال/غیرفعال می‌شود .
منظور از گروه رویداد ، مجموعه رویدادهای یک شیء است . مثلا mousedown , moseover , mousemove , ... در گروه mouse قرار می‌گیرند . یعنی می‌توانید با ارسال کلمه‌ی mouse فقط رویدادهای mouse را تحت نظر بگیرید یا اینکه فقط یک رویداد را مشخص کنید ، مثل mousedown .
راه ساده‌تر فعال کردن Event Logging ، رفتن به تب Html ، راست کلیک کردن بروی المنت مورد نظر و فعال کردن گزینه‌ی Log Events می‌باشد .

```
var obj = $("myInput");
monitorEvents(obj,'keypress');
```

نتیجه پس از فشردن چند دکمه‌ی کیبورد در myInput :

keypress charCode=0, keyCode=38
keypress charCode=0, keyCode=38
keypress charCode=0, keyCode=38
keypress charCode=0, keyCode=39
keypress charCode=0, keyCode=13
keypress charCode=0, keyCode=13
keypress charCode=115, keyCode=0
keypress charCode=100, keyCode=0
keypress charCode=97, keyCode=0
keypress charCode=97, keyCode=0

توضیحات بیشتر : http://getfirebug.com/wiki/index.php/MonitorEvents

profile([title]) and profileEnd()

این متدها ، JavaScript Profiler را فعال/غیرفعال می‌کنند . هنگام فعال کردن می‌توانید یک عنوان هم برای پروفایل مشخص کنید . در قسمت قبلی مقاله در مورد این قابلیت توضیحاتی ارائه شد .
سه را برای اجرای Profiler وجود دارد :
1 - کلیک بروی دکمه‌ی Profiler در بالای تب کنسول .
2 - استفاده از کد console.profile("ProfileTitle") در کدهای جاوا اسکریپت .
3 - استفاده از متد profile("Profile Title") در خط فرمان .

```
profile("myFunc Testing");
myFunc();
profileEnd();
```

نتیجه :

```
>>> profile("myFunc Testing"); myFunc(); profileEnd();
```

myFunc Testing (0.335ms, 3 calls)

Function	Calls	Percent	Own Time	Time	Avg	Min	Max	File
loop	2	95.22%	0.319ms	5.545ms	2.773ms	2.662ms	2.883ms	1.htm (line 26)
myFunc	1	4.78%	0.016ms	5.561ms	5.561ms	5.561ms	5.561ms	1.htm (line 22)

ستون‌های Profiler :

Function : نام تابع اجرا شده .
Calls : تعداد دفعات فراخوانی تابع .
Percent : زمان اجرای تابع در زمان کل ، به درصد .
Own Time : زمان اجرای تابع به تنهایی . برای مثال در کد ما ، زمان اجرای تابع myFunc به تنهایی تقریبا صفر است زیرا عملیاتی در خود انجام نمی‌دهد و زمان صرف شده در این تابع ، برای اجرای 2 با تابع loop است . این زمان (Own Time) زمان اجرای تابع ، منهای زمان صرف شده برای فراخوانی توابع دیگر است .
Time : زمان اجرای تابع از نقطه‌ی آغاز تا پایان . مجموع زمان اجرای خود تابع به همراه زمان اجرای توابع فراخوانی شده . در کد ما ، این زمان ، مجموع زمان اجرای خود تابع به همراه دو بار فراخوانی تابع loop است .
Avg : میانگین زمان اجرای هربار تابع . فرمول : Avg = Time / Calls
Min & Max : حداقل و حداکثر زمان اجرای تابع .
File : نام فایل و شماره خطی که تابع در آن قرار دارد .

در قسمت بعد با توابع کنسول آشنا خواهیم شد .

منابع : http://michaelsync.net/2007/09/15/firebug-tutorial-commandline-api
http://michaelsync.net/2007/09/09/firebug-tutorial-logging-profiling-and-commandline-part-i
http://getfirebug.com/wiki/index.php/Console_Panel
http://getfirebug.com/wiki/index.php/MonitorEvents

Packtpub.Firebug.1.5.Editing.Debugging.and.Monitoring.Web.Pages.Apr.2010

عنوان: **نمایش خروجی RSS سایت‌های دیگر به کمک jQuery**
نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۶/۰۴ ۲۱:۵۱
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Tips

شاید خیلی از دوستان (مثل گذشته نه چندان دور خودم) خیلی بیش از اندازه به برنامه نویسی‌های سمت سرور اهمیت می دهند که این کار باعث از دست دادن ، سرعت و سادگی برنامه نویسی سمت کلاینت می‌شود.

معمولا ما برای کار با خروجی‌های XML از کدهای سمت سرور استفاده می‌کنیم ، بدون اینکه از قدرت جی کوئری در این زمینه اطلاعی داشته باشیم. البته در این مقاله خیلی به پردازش XML توسط جی کوئری نمی‌پردازیم و کار اصلی را گوگل برای ما انجام می‌دهد.
برای انجام این کار ما ابتدا توسط یکی از کتابخانه‌های گوگل ، خروجی RSS یک وب سایت رامی خوانیم و بعد به کمک jQuery آن‌ها را نمایش می‌دهیم.

```
<script src="jquery-1.8.0.min.js" type="text/javascript"></script>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">

        google.load("feeds", "1");

        function initializeda() {
            var feed = new google.feeds.Feed("http://www.drupaleasy.ir/rss.xml");
            feed.setNumEntries(5);
            feed.setResultFormat(google.feeds.Feed.JSON_FORMAT);
            feed.load(function (result) {
                if (!result.error) {
                    for (var i = 0; i < result.feed.entries.length; i++) {
                        var entry = result.feed.entries[i];
                        $('#drupaleasy ul').append('<li><a href="' + entry.link + '">' + entry.title +
'</a></li>');
                    }
                }
            });
        }

google.setOnLoadCallback(initializeda);
    </script>
```

توضیحات کد :
ابتدا نیاز داریم کتابخانه گوگل را به صفحه اضافه کنیم. شما می‌توانید مستندات کامل این کتابخانه در این لینک بخوانید.

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

در ابتدا معرفی می‌کنیم که به کدام امکان این کتابخانه نیاز داریم ، در این جا ما از ورژن "1" تابع feed استفاده می‌کنیم.بد نیست برای آشنایی بیشتر سری هم به این لینک بزنید.

```
google.load("feeds", "1");
```

ابتدا لینک مورد نظر که باید خروجی از آن گرفته شود را معرفی می‌کنیم و آن را در متغییری قرار می‌دهیم.

```
var feed = new google.feeds.Feed("http://www.drupaleasy.ir/rss.xml");
```

تنظیمات دلخواه خودمان را نظیر تعداد پست واکشی شده و نوع خروجی مشخص می‌کنیم. تجربه (شخصی)نشان داده که بهترین نوع خروجی JSON است.

```
feed.setNumEntries(5);
feed.setResultFormat(google.feeds.Feed.JSON_FORMAT);
```

و بعد هم به کمک jQuery ساختار مورد نظر خودمان را تنظیم می‌کنیم. شما می‌توانید به کمک مستندات ، و سلیقه شخصی خودتان این کار را انجام بدهید ، و در آخر هم تنظیم می‌کنید ، اجرای تابع شما در چه زمانی اتفاق بیافتید.

```
feed.load(function (result) {
                if (!result.error) {
                    for (var i = 0; i < result.feed.entries.length; i++) {
                        var entry = result.feed.entries[i];
                        $('#drupaleasy ul').append('<li><a href="' + entry.link + '">' + entry.title +
'</a></li>');
                    }
                }
            });
```

در این جا من خروجی فید را به صورت یک لیست (ul) تنظیم کردم ، به این صورت که به ازای هر سطر یک li همراه با تگ a تولید کرده ام و به ul مورد نظر append کردم.

فایل HTML :

```
<div id="drupaleasy" class="feeds">
            <span>DrupalEasy.ir</span>
                <ul>
                </ul>
                <a href="http://drupaleasy.ir">more</a>
            </div>
```

همچنین خیلی راحت می‌توانید به کمک CSS استایل‌های دلخواه خودتون رو اعمال کنید.

```
.feeds
        {
            float: right;
            background-color: rgba(234, 242, 243, 0.73);
            margin: 5px;
            border-radius: 20px;
            padding: 8px;
            width: 293px;
            height: 217px;
            border: 1px solid #293883;
        }

        #drupaleasy ul
        {
            list-style-image: url("img/drupal.png");
        }
```

این هم شد خروجی کار من

DrupalEasy.ir
طراحی کاملا استاندارد برای اوبونتو
آموزش طراحی قالب دروپال - قسمت دوم
استفاده از گرادیانت سی اس اس 3 حتی در IE6 !
نگاهی به پایگاه داده دروپال
سی اس اس 3 (CSS3) در اختیار شماست
more

نظرات خوانندگان

نویسنده: سعید
تاریخ: ۱۳۹۱/۰۶/۰۶ ۹:۳۶

جالبه که گوگل امکان cross site ajax request رو می‌ده.

نویسنده: امیرحسین
تاریخ: ۱۳۹۱/۰۶/۰۸ ۱۳:۱۹

بله این امکان رو یاهو هم می‌دهد.
ولی استفاده از گوگل خیلی خوش دست‌تر و راحت‌تر هست.

نویسنده: علیرضا صبوری
تاریخ: ۱۳۹۲/۱۲/۲۶ ۱:۱۵

سلام مطلب مفیدی بود ولی فکر میکنم بیشتر این قابلیت مربوط با کتابخانه گوگل هستش .. نه jquery . چون با جاواسکریپت هم به سادگی همین کار رو میشه کرد. موفق باشید

عنوان: **آپلود فایل ها با استفاده از PlUpload در Asp.Net Mvc**
نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۶/۰۶ ۱۳:۴۳
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.Net MVC, jQuery, Ajax

امروزه بازار برنامه‌های تماما ajax و بدون Postback شدن صفحه بسیار داغ میباشد که از این موارد میتوان به برنامه‌های تحت وب گوگل اشاره کرد. (gmail ، googlePlus ، Google Reader)
در این میان یکی از دغدغه‌های توسعه دهندگان وب ، آپلود فایل‌ها به صورت آنی (مثل attach files گوگل) میباشد. برای حل این مسئله ، ابزارها و پلاگین‌های متعددی وجود دارد که در اینجا به 10 تا از پلاگین‌های Jquery اشاره شده است.
به شخصه با پلاگین Uploadify کار کرده ام و از استفاده از آن راضی هستم ولی همین دیشب برای قسمتی از یک پروژه نیاز به ابزاری جهت آپلود فایل‌ها با امکانات مورد نظرم داشتم که به PlUpload برخورد کردم.

از امکاناتی که این ابزار در اختیار شما قرار میدهد :
- یک اینترفیس زیبا جهت آپلود و افزودن فایل ها
- پشتیبانی از زبان‌های مختلف و همین طور زبان فارسی
- امکان استفاده از قالب Jquery UI
- Drag&Drop برای مرورگرهایی که از Html5 پشتیبانی میکنند

حال که با امکانات این ابزار بیشتر آشنا شدید بریم سراغ استفاده از این ابزار در asp.net mvc :)
ابتدا پروژه را از اینجا دانلود کنید. سپس یک پروژه‌ی جدید mvc 3 بسازید (از نوع Internet Application و با نام دلخواه). سپس پوشه‌ی plupload را در قسمت سلوشن برنامه کپی کنید.
حال در فایل Views->Shared->_Layout.cshtml ، تگ head را جهت افزودن امکانات پلاگین این گونه تغییر دهید :

```
<title>@ViewBag.Title</title>

    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
    <link href="../../plupload/js/jquery.plupload.queue/css/jquery.plupload.queue.css" rel="stylesheet"
/>
    <script src="@Url.Content("~/Scripts/jquery-1.7.1.min.js")" type="text/javascript"></script>
    <script type="text/javascript" src="http://bp.yahooapis.com/2.4.21/browserplus-min.js"></script>

    <script src="../../plupload/js/plupload.full.js"></script>
    <script src="../../plupload/js/jquery.plupload.queue/jquery.plupload.queue.js"></script>
    <script src="../../plupload/js/i18n/fa.js"></script>
```

نکته : فایل fa.js که جهت استفاده از زبان فارسی در اینترفیس آپلود فایل‌ها میباشد، که وجود آن در آدرس واضح میباشد.
سپس به فایل Views->Home->Index.cshtml بروید و آن را این گونه دوباره نویسی کنید :

```
@{
    ViewBag.Title = "Uploading Files using PlUpload";
}
<h2>@ViewBag.Message</h2>

@using (Html.BeginForm("Post", "home", FormMethod.Post,
    new { enctype = "multipart/form-data" }))
{
    <div id="uploader">
        <p>You browser doesn't have Flash, Silverlight, Gears, BrowserPlus or HTML5 support.</p>
    </div>
}

<script>
    $(function () {

        $("#uploader").pluploadQueue({
            // General settings
            runtimes: 'html5,gears,flash,silverlight,browserplus,html4',
            url: '@Url.Action("Upload" , "Home")',
            max_file_size: '10mb',
            chunk_size: '1mb',
```

```
            unique_names: true,

            // Resize images on clientside if we can
            resize: { width: 320, height: 240, quality: 90 },

            // Specify what files to browse for
            filters: [
                { title: "Image files", extensions: "jpg,gif,png" },
                { title: "Zip files", extensions: "zip" }
            ],

            // Flash settings
            flash_swf_url: '/plupload/js/plupload.flash.swf',

            // Silverlight settings
            silverlight_xap_url: '/plupload/js/plupload.silverlight.xap'
        });
    });
</script>
```

توضیحات و نکات :

- جهت آپلود فایل‌ها تگ "enctype = "multipart/form-data را فراموش نکنید.

- در قسمت مقداردهی به ویژگی‌های Plupload ، قسمت runtime به صورت ترتیبی کار میکند لذا اگر اولی پشتیبانی نشود سراغ دومی میرود و اگر دومی نشود سومی و ... در صفحه‌ی اول سایت PlUpload ، موارد پشتیبانی شده توسط تکنولوژی‌ها آورده شده است لذا این ترتیب را ترتیب مناسبی میبینم و اگر اولین مورد html5 باشد امکان Drag&Drop وجود خواهد داشت.

خود سایت PlUpload داکیومنت خیلی خوبی جهت توضیح موارد مختلف دارد لذا توضیح دوباره لازم نیست.

همان طور که در ویژگی url مشاهده میکنید به کنترلر Home و اکشن متود Upload اشاره شده است که طرز کار به این گونه است که هر بار که یک فایل آپلود میشود درخواستی به این آدرس و محتوای فایل در قسمت Request.Files ارسال میشود و همین طور نام فایل که unique ارسال میشود و chunk که تیکه‌های فایل است(پست میشود).

پس اکشنی با نام Upload در کنترلر HomeController بسازید :

```
[HttpPost]
        public ActionResult Upload(int? chunk, string name)
        {
            var fileUpload = Request.Files[0];
            var uploadPath = Server.MapPath("~/App_Data");
            chunk = chunk ?? 0;
            using (var fs = new FileStream(Path.Combine(uploadPath, name), chunk == 0 ? FileMode.Create
: FileMode.Append))
            {
                if (fileUpload != null)
                {
                    var buffer = new byte[fileUpload.InputStream.Length];
                    fileUpload.InputStream.Read(buffer, 0, buffer.Length);
                    fs.Write(buffer, 0, buffer.Length);
                }
            }
            return Content("chunk uploaded", "text/plain");
        }
```

توضیحات : ابتدا فایل مورد نظر از قسمت Request.Files واکشی میشود و سپس فایل را در پوشه App_Data ذخیره میکند. (یکی از چندین روش ذخیره سازی که مطالعه در این قسمت به خواننده واگذار میشود.)

حال برنامه را اجرا کنید و از این ابزار لذت ببرید:) پروژه‌ی ضمیمه شده

نکته : قسمت فارسی ساز اونو تغییر دادم چون که ترجمه‌ی فارسی خودش یه سری نقایصی داشت که گویا از کار با google translate به وجود اومده بود!

نظرات خوانندگان

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۶/۰۶ ۱۵:۵۲

پروژه ضمیمه شد!

نویسنده: امیرحسین
تاریخ: ۱۳۹۱/۰۶/۰۶ ۱۶:۲۶

سلام
همه این ابزارها برای ارسال فایل به سرور از فلش استفاده می‌کنند.
و یه طورایی نمی‌شه گفت این کار با ای جکس انجام شده (ولی می‌شه گفت بدون پست بک)
موفق باشید.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۶/۰۶ ۱۷:۲۷

فلاش یکی از روش هاس ... ولی در کل حرف شما درسته و من به همین خاطر از واژه‌ی " آپلود فایل‌ها به صورت آنی " استفاده کردم

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۱/۰۶/۰۶ ۲۲:۹

ممنون از آموزش شما.

من خودم از این پلاگین استفاده می‌کنم.پلاگینی برای jquery هست و بدون استفاده از فلش و خیلی راحت با استفاده از ajax فایل را آپلود می‌کنه.

بازم ممنون

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۷ ۱:۴۴

تشکر بابت مطلب کاربردیتون
فقط یک مشکل داره اونم وقتی هست که فایل‌ها upload شدن . دیگه اجازه‌ی ulpoad دوباره نمیده

نویسنده: asal Mansuri
تاریخ: ۱۳۹۲/۰۱/۰۲ ۲۱:۴۶

سلام
من یک فایل آپلودر خیلی جمع و جور میخوام که فقط یک فایل رو آپلود کنه ولی ایجکسی باشه / اگه لطف کنین راهنمایی کنین که چطوری میتونم فایل آپلود رو در MVC بتونم به صورت ajax همراه با loading داشته باشم ممنون میشم .

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۳:۱۲

سلام
چطور میشه از لیست اسامی فایل‌های ارسال شده، در هنگام ثبت اطلاعات مدل در سرور استفاده کرد؟

در واقع اطلاعات فایل ها، قسمتی از اطلاعات مدل من محسوب میشه و در هنگام ثبت در دیتابیس باید این لیست رو هم ثبت کنم. ولی نمیدونم چطور در کنترلر ثبت مدل به اسامی فایلها دسترسی داشته باشم.
مثلا مدل زیر :

```
public class album
{
public string  name;
public string des;
public string[] images;
}
```

من اطلاعات مدل رو در حالت آجاکس به صورت ajax.beginform ارسال میکنم.

ممنون میشم اگر دوستان پاسخ بدن

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۳:۵۳

در متد public ActionResult Upload بالا fileUpload.FileName اسم واقعی فایل هست؛ پارامتر name یک اسم موقتی.

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۴:۱۶

خیلی ممنون از پاسختون
این قسمتی رو که گفتین درسته. من مرحله بعد از این رو مشکل دارم.
سوال اینه:
بعد از ارسال فایل‌ها به سرور، حالا نوبت ارسال اطلاعات مدل به سرور می‌رسه که این اسامی فایل‌ها هم قسمتی از اطلاعات مدل هست. چطور می‌شود در اکشن ثبت مدل، اسامی فایل‌ها اپلود شده را دریافت کرد و استفاده نمود.
یه راه حلی پیدا کردم که گفته بود بعد از اینکه کار آپلودر تموم شد اطلاعات فایل‌ها را به صورت فیلد مخفی در فرم ذخیره کن تا پست بک شود. نمیدونم آیا راه حل دیگه ای هست؟ این روش درست و اصولیه؟

```
uploader.bind('FileUploaded', function(up, file, info) {
  var obj = JSON.parse(info.response);
    $('form#quoteRequest').append('<input type="hidden" name="file_name" value="' +
obj.result.cleanFileName + '" />');
    //note obj.result.cleanFileName instead obj.cleanFileName
 });
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۴:۱۶

اینکار توصیه نمیشه. چون پس از اطمینان کامل از آپلود یک فایل باید نسبت به ذخیره اطلاعات آن اقدام کنید. الزامی نداره که ارسال فرم حتما به آپلود موفقیت آمیز فایل‌ها در این حالت خاص منجر بشه (چون مثلا از فلش یا سیلورلایت مجزای از فرم شما هم ممکن است برای ارسال داده‌های فایل‌ها استفاده کنه). یعنی آپلود فایل در اینجا بحث مجزایی از فرم شما است. ابتدا فایل‌ها را آپلود کنید؛ بعد اجازه بدید تا توضیحات مرتبط را ثبت کنند.
یا اینکه می‌شود در حین آپلود توسط افزونه هم یک سری توضیحات اضافی رو ارسال کرد:

```
// JS
up.settings.multipart_params = {
                    description: $("#imageDescription").val()
                 };
// C#
string description = context.Request.Form["description"];
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۴:۲۱

پیاده سازی دیگری از بحث PlUpload

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۴:۳۰

```
// JS
up.settings.multipart_params = {
                    description: $("#imageDescription").val(),
                    // other fields ....
                 };
// C#
string description = context.Request.Form["description"];
```

باید از multipart_params تنظیمات این افزونه برای ارسال اطلاعات اضافی استفاده کنید.

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۴:۴۳

با تشکر از دوستان .
یک مثال عملی؛ یه نرم افزار **ارسال ایمیل تحت وب مثل جیمیل** رو در نظر بگیرید. یه **فرم ارسال ایمیل** داره که می‌خواد اطلاعات رو بفرسته به سمت سرور. یه سری فایل هم قراره بصورت آجاکسی پیوست بشه. مراحل کار بدین صورته که ابتدا فایل‌ها آجاکسی به سرور فرستاده می‌شوند و یه جایی ذخیره می‌شوند. بعد فرم اطلاعات شامل گیرنده ، متن نامه و غیره بصورت آجاکس ارسال میشه. وقتی در سمت سرور می‌خوایم ایمیل رو بفرستیم، نیاز داریم که نام فایل‌های پیوست شده در مرحله قبل رو هم داشته باشیم.
همونطور که قبلا خدمت دوستان عرض کردم؛ راه حلی که من بذهنم می‌رسه اینه که بعد از اینکه فایل‌ها در مرحله اول آپلود شد، اسم فایل‌ها رو توی فرم در فیلد مخفی تزریق کنیم. بعد که فرم پست شد، طبیعتا دسترسی به نام فایل‌ها داریم. کد ذکر شده هم کار تزریق در هنگام تمام شدن ارسال فایل‌ها انجام میداد.
خواستم بدونم آیا این روش درست هست؟ و آیا روش تمیزتری وجود داره؟
باز هم تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۴ ۱۴:۵۶

همان روشی که برای دریافت نام فایل‌های آپلود شده از سرور انتخاب کردید، مناسب است:

```
$("#uploader").pluploadQueue({
      // Post init events, bound after the internal events
      init: {
               FileUploaded: function(up, file, info) {
               // Called when a file has finished uploading
               }
      }
});
```

نویسنده: فرشته محسنیان
تاریخ: ۱۳۹۲/۰۶/۱۵ ۹:۳۶

ممنون از مطالب که گفتین. فقط سایتش دیگه اجازه دانلود نمیده، آیا از جایی دیگه میتونم بگیرم؟ و دوم اینکه mvc 4 رو هم ساپورت میکنه یا نه؟ مرسی

نویسنده: محسن خان

تاریخ: ۱۳۹۲/۰۶/۱۵ ۱۰:۲۰

plupload سورس باز هست . البته با مجوز GPL. یعنی کار خودتون رو باید سورس باز کنید (یا مجوزش رو بخرید). ضمنا MVC4 با MVC3 سازگار هست. فقط یک سری افزونه بیشتر داره.

نویسنده: جواد
تاریخ: ۱۳۹۳/۰۳/۲۱ ۹:۱

باسلام. من این کارهایی که گفتید کردم. روی vs مشکلی نداشتم. ولی وقتی نرم افزارمو پابلیش کردم و روی سرور گذاشتم متأسفانه هرکار می‌کنم خطا عدم پشتیبانی مرورگر میده. با فایرباگ چک کردم دیدم به فایلهای کتابخانه اش عدم دسترسی می‌ده. تمام تنظیمات iis رو هم چک کردم. نتونستم درستش کنم.
خطایی که میده اینطوریه:

"NetworkError: 404 Not Found - http://~/Scripts/plupload/js/jquery.ui.plupload/jquery.ui.plupload.js

نویسنده: ناهید
تاریخ: ۱۳۹۳/۰۵/۰۶ ۹:۴

سلام
یه سوال؟ وقتی بخوایم اطلاعات ذخیره شده (فایلی که قبلا آپلود کرده) رو نمایش بدیم چجوری فایل رو توی plUpload نمایش بدیم؟ خود سایت FilesAdded داره که کار نمیکنه.

عنوان: ایجاد alert,confirm,prompt هایی متفاوت با jQuery Impromptu
نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۶/۱۸ ۱۸:۴۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, FireBug, jQuery

alert,confirm,prompt سه متد توکار JavaScript هستند که برای نمایش پیغام ، دریافت تایید و دریافت مقدار از کاربر هستند .

گرافیک این پیغام‌ها هم وابسته به مرورگر هستند و قابل تغییر نیستند . متن عنوان و دکمه‌ها هم با توجه به زبان سیستم عامل تعیین می‌شوند و قابل تغییر توسط برنامه نویس نیستند.

حال مواقعی پیش می‌آید که نیاز داریم پیغام هایی با گرافیک و عبارات متفاوت نمایش دهیم . برای رفع این نیاز می‌توانیم از پلاگین jQuery Impromptu استفاده کنیم . البته این پلاگین قابلیت‌های دیگری هم دارد که در این مقاله با آنها آشنا می‌شویم .

از ویژگی‌های این پلاگین می‌توان حجم کم (حدود 11 کیلوبایت) و قدرت شخصی سازی بالا اشاره کرد .

طرز استفاده به این شکل است :

```
$.prompt( msg , options )
```

msg می‌تواند یک string یا یک شئ از states باشد . string ارسال شده می‌تواند شامل کدهای html باشد .

یک شئ states هم شامل مجموعه ای از وضعیت‌های prompt است . برای مثال می‌توان یک prompt ایجاد کرد که مثل یک Wizard شامل چند مرحله باشد .

options هم تنظیماتی است که می‌توان مشخص کرد . تنظیماتی مثل : prefix,classes,persistent,timeout,...

msg :

گفتیم شئ states شامل وضعیت‌های مختلف prompt است . هر وضعیت (state) می‌تواند شامل بخش‌های زیر باشد :

html : مقدار Html وضعیت

buttons : یک شئ شامل متن و مقدار دکمه هایی که کاربر می‌تواند کلیک کند

focus : ایندکس دکمه‌ی focus شده در وضعیت

submit : تابعی که زمانی که یکی از دکمه‌های وضعیت انتخاب شود فراخوانی می‌شود .

اگر در این تابع false بازگشت داده شود یا متد preventDefault از event فراخوانی شود ، prompt باز می‌ماند . (روشی برای جلوگیری از بسته شدن prompt هنگام تغییر state یا اعتبار سنجی فرم)
همچنین شئ event شامل state (المنت state) و stateName (نام state) می‌باشد .

پیشفرض :

```
function(event, value, message, formVals){}
```

value مقدار دکمه ای است که بروی آن کلیک شده ، message مقدار html تعریف شده برای state است ، formVals هم در صورتی که در html تعریف شده برای state ، المنت‌های فرم وجود داشته باشد ، شامل نام/مقادیر آنها می‌باشد . (برای دریافت مقادیر فرم ، باید از نام المنت استفاده نمایید .)

position : مشخص کننده‌ی موقعیت state که شامل موارد زیر است :

```
position: { container: '#container', x: 0, y: 0, width: 0, arrow: 'lm' }
```

container : selector المنتی است که state باید در آن مکان قرار بگیرد .
x/y : موقعیت نسبی prompt نسبت به container
arrow : جهت نمایش فلش prompt است که می‌تواند یکی از این مقادیر باشد : tl, tc, tr, rt, tm, tb, br, bc, bl, lb, lm, lt.

نحوه تعریف یک wizard ساده با شئ states :

```
var tourSubmitFunc = function (e, v, m, f) {
    if (v === -1) {
        $.prompt.prevState();
```

```
        return false;
    } else if (v === 1) {
        $.prompt.nextState();
        return false;
    }
};

var states =
    {
        state0:
            {
                html: "State1",
                buttons: { Next: 1 },
                //position: { container: '#container', x: 10, y: 0, width: 350, arrow: 'lm' },
                submit: tourSubmitFunc
            },
        state1:
            {
                html: "State2",
                buttons: { Prev: -1, Next: 1 },
                submit: tourSubmitFunc
            },
        state2:
            {
                html: "State3",
                buttons: { Prev: -1, Done: 0 },
                submit: tourSubmitFunc
            }
    };

$.prompt(states);
```

تا به اینجا با پارامتر اول prompt آشنا شدیم و فهمیدیم که می‌توانیم یک رشته یا یک شئ states به عنوان message به prompt ارسال کنیم .

options :

اکنون با optionهای prompt (پارامتر دوم) آشنا خواهیم شد .

توجه کنید که زمانی که یک رشته به prompt ارسال کنید ، مقادیر buttons,focus,submit از این تنظیمات دریافت می‌شود .
به عبارت دیگر ، زمانی که یک شئ states به prompt ارسال کنید ، از مقادیر فوق که در تنظیمات است ، استفاده نمی‌شود .

loaded

یک تابع که زمانی که prompt کامل بارگزاری شده فراخوانی می‌شود .

```
$.prompt("Message",
    {
        loaded: function() {
            alert("Prompt Loaded !");
        }
    });
```

submit

یک تابع که زمانی که یکی از دکمه‌های state کلیک شود ، فراخوانی می‌شود .
(زمانی اتفاق میوفتد که یک رشته به عنوان متن به prompt ارسال کرده باشید و زمانی که یک شئ از states ارسال می‌کنید ، هنگام کلیک دکمه‌های آنها ، این تابع فراخوانی نمی‌شود .)
پیشفرض :

```
function(event){}
```

statechanging

یک تابع که زمانی که یک state در حال تعویض شدن هست فراخوانی می‌شود .

پیشفرض :

```
function(event, fromStateName, toStateName){}
```

برای لغو تغییر state ، مقدار return false کنید یا متد preventDefault از event را فراخوانی کنید .

statechanged

یک تابع که زمانی که یک state در حال تعویض شدن هست فراخوانی می‌شود .

پیشفرض :

```
function(event, toStateName){}
```

callback

یک تابع که زمانی که (یکی از دکمه‌های prompt کلیک شود و) prompt بسته شود ، فراخوانی می‌شود .

پیشفرض :

```
function(event[, value, message, formVals]){}
```

سه پارامتر آخر تنها زمانی که یک دکمه‌ی prompt کلیک شده باشد موجود هستند .

buttons

یک شئ شامل مجموعه ای از دکمه‌ها .

پیشفرض :

```
{ Ok : true }
```

شکل دیگر تعریف دکمه به این شکل است :

```
[
    {title: 'Hello World',value:true},
    {title: 'Good Bye',value:false}
]
```

prefix

یک پیشوند برای همه cssها و idهای المنت‌های html که توسط prompt ایجاد می‌شود .

پیشفرض : jqi

classes

یک css class که به بالاترین سطح prompt داده می‌شود .

در حالت پیشفرض مقداری ندارد .

focus

ایندکس دکمه‌ی focus شده

پیشفرض : 0

zIndex

zIndex اعمال شده بروی prompt .
پیشفرض : 999

useiframe
استفاده از یک iframe برای overlay در IE6
پیشفرض : false

top
فاصله‌ی prompt از بالای صفحه
پیشفرض : 15%

opacity
میزان شفافیت لایه‌ی ای که صفحه را پوشانده است .
پیشفرض : 0.6

overlayspeed
سرعت نمایش افکت fadeIn , fadeOut لایه‌ی پوشاننده .
مقادیر قابل قبول : "slow", "fast", number(milliseconds)
پیشفرض : "slow"

promptspeed
سرعت نمایش prompt .
مقادیر قابل قبول : "slow", "fast", number(milliseconds)
پیشفرض : "fast"

show
نام یک متد jQuery برای animate کردن نمایش prompt .
مقادیر قابل قبول : "show","fadeIn","slideDown", ...
پیشفرض : "promptDropIn"

persistent
بسته شدن prompt زمانی که بروی لایه‌ی fade کلیک شود .
false : بسته شدن
پیشفرض : true

timeout
مدت زمانی که پس از آن ، prompt بصورت خودکار بسته می‌شود . (milliseconds)
پیشفرض : 0

: **returns**

مقدار بازگشتی متد prompt ، یک شئ jQuery ، شامل همه‌ی محتویات تولید شده توسط prompt است .

: **Events using Bind**

استفاده از Eventها با بایند کردن

promptloaded

معادل loaded در optionها .

promptsubmit

هنگام submit شدن (کلیک شدن یکی از دکمه‌های) state فعال می‌شود .

promptclose

معادل callback در optionها .

promptstatechanging

معادل statechanging در optionها .

promptstatechanged

معادل statechanged در optionها .

ظرز بایند کردن یک event به شئ prompt :

```
var myPrompt = $.prompt( msg , options );
myPrompt.bind('promptloaded', function(e){});
```

Helper Functions :

jQuery.prompt.setDefaults(options)

تعیین مقادیر پیشفرض برای همه‌ی promptها .

```
jQuery.prompt.setDefaults({
prefix: 'myPrompt',
show: 'slideDown'
});
```

jQuery.prompt.setStateDefaults(options)

تعیین مقادیر پیشفرض برای stateها .

```
jQuery.prompt.setStateDefaults({
buttons: { Ok:true, Cancel:false },
focus: 1
});
```

jQuery.prompt.getCurrentState()

یک شئ jQuery از state جاری برمی‌گرداند .

jQuery.prompt.getCurrentStateName()

نام state جاری را برمی‌گرداند .

jQuery.prompt.getStateContent(stateName)

یک شئ jQuery از state مشخص شده برمی‌گرداند .

(jQuery.prompt.goToState(stateName, callback

state مشخص شده را در prompt نمایش می‌دهد .

callback تابعی است که بعد از تغییر state فراخوانی می‌شود .

(jQuery.prompt.nextState(callback

prompt را به state بعدی منتقل می‌کند .

(jQuery.prompt.prevState(callback

prompt را به state قبلی منتقل می‌کند .

()jQuery.prompt.close

prompt را می‌بندد .

حال که با این پلاگین آشنا شدیم ، سه دستور جاوا اسکریپتی که در ابتدای مقاله ذکر کردیم را با این پلاگین پیاده سازی می‌کنیم .

alert

```
$.prompt("یک پیام تستی",
    {
        prefix: 'dnt',
        buttons: { 'تایید': true }
    });
```

نتیجه :

confirm

```
$.prompt("درخواست تایید - موافقید ؟",
    {
        prefix: 'dnt',
        buttons: { 'تایید': true, 'انصراف': false }
    });
```

نتیجه :

prompt

یک فرم html مخفی برای نمایش در prompt :

```
<div class="prompt-content" style="display: none;">
    <span>نام خود را وارد نمایید : </span>
    <span>
        <input type="text" name="name" />
    </span>
</div>
```

نمایش prompt :

```
$.prompt(
    $(".prompt-content").html(),
    {
        prefix: 'dnt',
        buttons: { 'تایید': true, 'انصراف': false }
    });
```

نتیجه :

در این سه مثال آخر ، از یک css متفاوت استفاده کردیم . این پلاگین یک سری کلاس دارد که نام این کلاس‌ها از ترکیب مقدار prefix که در option مشخص کردیم حاصل می‌شود .

برای مثال اگر مقدار prefix را برابر با dnt قرار بدهیم ، برای استایل دهی متن پیام باید از کلاس div.dnt .dntmessage استفاده کنید .

همانطور که در سه مثال آخر مشاهده کردید ، تغییری در استایل prompt داشتیم که با تغییر دادن استایل‌های مورد نظر انجام شد .

برای تهیه‌ی یک قالب سفارشی باید selector المنت هایی که نیاز به تغییر دارند را از قالب پیشفرض پیدا کنید ، سپس قسمت prefix از selector را به نام قالب خودتان تغییر بدید و استایل را ویرایش کنید . سپس هنگام ایجاد prompt ، مقدار prefix را برابر نام قالب قرار دهید .

مثلا می‌خواهید یک قالب به نام dnt داشته باشید . می‌خواهید متن قالب راست به چپ باشد .

```
div.dnt .dntmessage {
     color: #444444;
     line-height: 20px;
     padding: 10px;
/*       Edited      */
     direction:rtl;
     text-align:right;
}
```

البته راه بهتری هم هست که نیاز به آشنایی با فایرباگ دارد . در این روش ابتدا کل قالب jqi (موجود در قالب پیشفرض) را در برنامه خود کپی می‌کنیم ، مقادیر jqi را با نام قالب جایگزین می‌کنیم ، مقدار prefix را در prompt برابر با نام قالب قرار می‌هیم . اکنون در FireFox یک prompt ایجاد می‌کنیم و توسط فایرباگ استایل هایی که با نام قالب بروی prompt اعمال شده‌اند را مطابق سلیقه تغییر می‌دهیم . در مرحله آخر به تب CSS در فایرباگ می‌رویم و کل استایل‌های مربوط به قالب را کپی و جایگزین استایل قبلی در برنامه می‌کنیم .

قالبی که بنده برای سه دستور فوق استفاده کردم (dnt) ، به این شکل است :

```
/*    Start : DotNetTips Theme     */

.dntfade {
     background-color: #AAAAAA;
     position: absolute;
}

div.dnt {
     background-color: #FFFFFF;
     border-radius: 10px 10px 10px 10px;
     border: 1px solid #FFFFFF;
     box-shadow: 0px 0px 10px 1px #6D6D6C;
     font-family: tahoma;
     font-size: 11px;
     padding: 7px;
     position: absolute;
     text-align: left;
     width: 400px;
}

div.dnt .dntcontainer {
     font-size: small;
}

div.dnt .dntclose {
     color: #BBBBBB;
     cursor: pointer;
     font-weight: bold;
     position: absolute;
     top: 4px;
     width: 18px;
}

div.dnt .dntmessage {
     color: #444444;
     line-height: 20px;
     padding: 10px;
}

div.dnt .dntbuttons {
```

```
     background-color: #F4F4F4;
     border: 1px solid #EEEEEE;
     padding: 5px 0px;
     text-align: right;
}

div.dnt button {
     background-color: #2F6073;
     border: 1px solid #F4F4F4;
     color: #FFFFFF;
     font-size: 12px;
     font-weight: bold;
     margin: 0px 10px;
     padding: 3px 10px;
}

div.dnt button:hover {
     background-color: #728A8C;
}

div.dnt button.dntdefaultbutton {
     background-color: #0099CC;
}

.dnt_state {
     direction: rtl;
     text-align: right;
}

.dnt_state button {
     font-family: tahoma;
}

.dntwarning .dnt .dntbuttons {
     background-color: #CCDDFF;
}

.dnt .dntarrow {
     border: 10px solid transparent;
     font-size: 0px;
     height: 0px;
     line-height: 0;
     position: absolute;
     width: 0px;
}

.dnt .dntarrowtl {
     border-bottom-color: #FFFFFF;
     left: 10px;
     top: -20px;
}

.dnt .dntarrowtc {
     border-bottom-color: #FFFFFF;
     left: 50%;
     margin-left: -10px;
     top: -20px;
}

.dnt .dntarrowtr {
     border-bottom-color: #FFFFFF;
     right: 10px;
     top: -20px;
}

.dnt .dntarrowbl {
     border-top-color: #FFFFFF;
     bottom: -20px;
     left: 10px;
}

.dnt .dntarrowbc {
     border-top-color: #FFFFFF;
     bottom: -20px;
     left: 50%;
     margin-left: -10px;
}

.dnt .dntarrowbr {
     border-top-color: #FFFFFF;
     bottom: -20px;
```

```
     right: 10px;
}

.dnt .dntarrowlt {
     border-right-color: #FFFFFF;
     left: -20px;
     top: 10px;
}

.dnt .dntarrowlm {
     border-right-color: #FFFFFF;
     left: -20px;
     margin-top: -10px;
     top: 50%;
}

.dnt .dntarrowlb {
     border-right-color: #FFFFFF;
     bottom: 10px;
     left: -20px;
}

.dnt .dntarrowrt {
     border-left-color: #FFFFFF;
     right: -20px;
     top: 10px;
}

.dnt .dntarrowrm {
     border-left-color: #FFFFFF;
     margin-top: -10px;
     right: -20px;
     top: 50%;
}

.dnt .dntarrowrb {
     border-left-color: #FFFFFF;
     bottom: 10px;
     right: -20px;
}

/*    End : DotNetTips Theme     */
```

برای مشاهده‌ی مثال‌های بیشتر به صفحه‌ی اصلی jQuery Impromptu مراجعه نمایید .

نظرات خوانندگان

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۱/۰۶/۱۹ ۱۰:۳۶

با سلام و تشکر از شما

نمونه پست ارسالی شما در این آدرس نیز موجود است لطفا تفاوت کارایی‌های این دو را بفرمائید و از طرفی اگر امکان دارد نحوه استفاده آن در دات نت را برای تائید و دریافت ورودی توضیح دهید باتشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۹ ۲۳:۵۷

- تفاوت مهم آن در حجم کمتر است. jQuery-UI نسبتا حجم بالایی دارد و اگر صرفا قرار است پیام کوتاهی به کاربری نمایش داده شود، روش فوق حداقل از لحاظ حجم (11K) مقرون به صرفه‌تر است.
- استفاده از نتایج آن هم مطابق توضیحات مفصل مطلب جاری، در قسمت callback آن باید صورت گیرد. برای مثال

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۱/۰۶/۲۰ ۱۴:۵۹

ممنون،
یه سوال بنده یه فرم ثبت نام دارم که با کلیک بر روی Botton اطلاعات در دیتابیس درج می‌شود حالا می‌خوام بعد از ثبت اطلاعات یه alert به کاربر نشون بده که اطلاعات با موفقیت ثبت شد ولی مشکل اینجاست که وقتی کاربر روی Button کلیک میکنه چون صفحه PostBack میشه دیگه پیغام به کاربر نمایش داده نمیشه،اگه امکان داره بنده رو راهنمایی بفرمائید.
دقیقا مثل همین قسمت ارسال نظر سایت.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۰ ۱۵:۵۹

این مطلب می‌تونه مفید و مرتبط باشه.

نویسنده: رضا
تاریخ: ۱۳۹۱/۰۷/۱۲ ۲:۳۲

سلام. یک linkbutton دارم تو گرید ویو که کار حذف رو انجام میده. در حالت پیش فرض با دستور

```
return confirm('')
```

و انتساب اون به خاصیت onClientClick میشد یک confirm ساده قبل حذف داشت. حالا وقتی میخام از این کنترل برای این کار استفاده کنم کار نمیکنه؟ میشه راهنمایی کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۲ ۹:۴۶

مثالی که در این کامنت معرفی شد مرتبط به GridView هم هست (انتهای مقاله).

نویسنده: omid

تاریخ: ۱۳۹۱/۱۰/۱۲ ۸:۱۹

سلام
با تشکر از راهنمایی شما برای کنترل‌ها من از این کنترل می‌خواستم استفاده کنم ولی اجرا نشد اگه راهنماییم کنید ممنون میشم .

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type" />
<title></title>

<link href="jquery-impromptu.css" media="all" rel="stylesheet" type="text/css" />
<script src="jquery-1.8.3.min.js" type="text/javascript"></script>
<script src="jquery-impromptu.js" type="text/javascript"></script>
<script type="text/javascript">

$(function(){
$show.click(function(e){
$.prompt("Hello World!");
});
});
});

</script>
</head>

<body>
<button class="show">ShowPrompt</button>
</body>
</html>
```

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۲ ۱۰:۲۴

- کد شما تعداد {} متوازنی ندارد. دو } بازه شده، سه تا بسته.
- به نظر با آخرین نگارش jQuery سازگار نیست. به صفحه خطای مرورگر مراجعه کنید تا خطای شناخته نشدن نوع مرورگر قابل مشاهده باشد. با نگارش‌های پایین‌تر مشکلی ندارد .

نویسنده: omid
تاریخ: ۱۳۹۱/۱۰/۱۲ ۱۰:۵۶

سلام
خیلی ممنون بابت راهنماییتون و آشنایی با سایت http://jsfiddle.net

نویسنده: neda
تاریخ: ۱۳۹۲/۰۱/۰۹ ۱۲:۲۶

ممنون از سایت خیلی خوب و مفیدتون
ممنون میشم اگه امکانش هست یه سمپل ساده و کوچیک از این کار هم در اختیار بگذارید
من از کدها استفاده کردم ولی موفق نشدم خروجی صحیح بگیرم
ممنونم

نویسنده: احمد احمدی
تاریخ: ۱۳۹۲/۰۱/۰۹ ۱۷:۱۷

```
<!DOCTYPE html>
<html>
```

```
<head>
<link rel="stylesheet" media="all" type="text/css" href="http://trentrichardson.com/Impromptu/jquery-impromptu.css" />
<script type="text/javascript" src="http://code.jquery.com/jquery-1.9.0.min.js"></script>
<script type="text/javascript" src="http://trentrichardson.com/Impromptu/jquery-impromptu.js"></script>
</head>
<body>
<button class="show">ShowPrompt</button>
<script type="text/javascript">
$(function(){
   $(".show").click(function(e){
      $.prompt("Hello World!");
   });
});
</script>
</body>
</html>
```

نویسنده: سمیرا
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۳:۴۴

سلام؛ وقتی با استفاده از jQuery Impromptu یک inputbox ایجاد می‌کنیم و کاربر اطلاعاتی رو توی inputbox وارد می‌کنه، چجوری می‌تونیم مقداری که کاربر وارد کرده رو برگردونیم و ازش استفاده کنیم ؟

نویسنده: مژده
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۴:۲۳

سلام
من هم همین مشکلو داشتم
از این روش استفاده کردم جواب داد :

```
var myval = m.find('#name').val();
alert(myval);
```

نویسنده: احمد احمدی
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۸:۴۴

سلام؛

طبق مطلب ارائه شده در مقاله:

تابع submit :

```
function(event, value, message, formVals){}
```

value مقدار دکمه ای است که بروی آن کلیک شده ، message مقدار html تعریف شده برای state است ، formVals هم در صورتی که در html تعریف شده برای state ، المنت‌های فرم وجود داشته باشد ، شامل نام/مقادیر آنها می‌باشد . (برای دریافت مقادیر فرم ، باید از نام المنت استفاده نمایید .)

خب حالا مشابه مثال زیر عمل کنید:

```
<div class="prompt-content" style="display: none;">
    <span>ایمیل خود را وارد نمایید : </span>
    <span>
        <input type="text" name="user_email" />
    </span>
```

```
</div>
```

```
$.prompt(
    $(".prompt-content").html(),
    {
        submit: function (e, v, m, f) {
            var userEmail = f["user_email"];
            console.log(userEmail);
        }
    });
```

نویسنده: مژده
تاریخ: ۱۳۹۲/۰۳/۰۹ ۲۱:۱۳

سلام
خسته نباشید
من توی سایتم از ajax استفاده کردم و برای حذف یه رکورد اول از کاربر تاییدیه گرفتم . اگه کاربر روی دکمه‌ی بله کلیک کنه ، عملیات شروع میشه و رکورد حذف میشه .
آخر سر توی تابعی که نتیجه برمیگرده توش (handleServerResponse)، بر اساس نتیجه‌ی برگشتی پیغام مناسبی بایدنمایش داده بشه. اما هیچ پیغامی نشون داده نمیشه
خیلی چیزا رو آزمایش کردم و آخر سر به این نتیجه رسیدم که چون دو تا پیغام میخواد پشت سر هم نمایش داده بشه ، این مشکل پیش میاد :(
حتی وقتی که پیغام تایید حذف رو برداشتم ، دیدم که پیغام دوم میاد !
واااقعا نمیدونم علتش چیه ! ممنون میشم کمکم کنید.
اینم قسمتی از کدمه :

```
function deleteFile(location, filename)//حذف فیزیکی و منطقی فایل
{
    $.prompt("آیا برای حذف فایل موجود اطمینان دارید ؟", {
        title: '<img src="Images/warning.png" />',
        buttons: { "بله": true, "خیر": false },
        focus: 2,
        submit: function (e, v, m, f) {
            if (v == true) {
                var getdate = new Date();  //Used to prevent caching during ajax call
                if (xmlhttp) {
                    $("p#vtip").fadeOut("slow").remove(); //محو شدن tooltip
                    var id = document.getElementById("id").value;
                    var i = '3';
                    params = "id=" + id + "&i=" + i + "&filename=" + filename + "&location=" +
location;
                    xmlhttp.open("POST", "FetchData/dbSearchDocument1.php", true);
                    xmlhttp.onreadystatechange = handleServerResponse4;
                    xmlhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
                    xmlhttp.send(params);
                }
            }
            else {
            }
        }
    });
}
//-----------------------------------------------------------------------
function handleServerResponse4() {
    if ((xmlhttp.readyState == 1) || (xmlhttp.readyState == 2) || (xmlhttp.readyState == 3))//loading
    {
        document.getElementById("content").innerHTML = "<img src='Images/bigloading.gif'/>";
    }
    else if (xmlhttp.readyState == 4) //ready
    {
        if (xmlhttp.status == 200) {
            if (xmlhttp.responseText == 1) {
                $.prompt(". حذف فایل با موفقیت انجام شد ", {
```

```
                    title: '<img src="Images/warning3.png" />',
                    buttons: { "بستن": true }
                });
            }
            else {
                $.prompt("! حذف فایل با خطا مواجه شده است", {
                    title: '<img src="Images/warning2.png" />',
                    buttons: { "بستن": true }
                });
            }
        }
        else {
            alert("Error during AJAX call. Please try again");
        }
    }
}
//-----------------------------------------------------------------------
```

نویسنده: مژده
تاریخ: ۱۳۹۲/۰۳/۰۹ ۲۱:۵۴

من الان یک بار دیگه این مطلبو خوندم و به نظرم رسید که باید از callback استفاده کنم.
اما حتی این توی سایتم جواب نمیده :

```
function mycallbackfunc(v,m){
      $.prompt('i clicked ' + v);
}
$.prompt('Example 8',{ callback: mycallbackfunc });

});
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۰ ۰:۱۶

زمانیکه از jQuery استفاده می‌کنید، دیگر نباید از Ajax خام استفاده کنید. خود jQuery تابع سازگار با انواع و اقسام مرورگرها رو برای کار با Ajax داره. این مثال تست شده و مشکلی نداره:

```
using System.Web.Mvc;

namespace jQueryImpromptu.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Index(string postId, string filename)
        {
            return Content("ok");
        }
    }
}
```

```
@{
    ViewBag.Title = "Index";
    var postUrl = Url.Action(actionName: "Index", controllerName: "Home");
}
<h2>
    Index</h2>
<input type="button" id="btnDelete" value="delete" />
```

```
<div id="infoDiv">
    Some Info ....
</div>
@section JavaScript
{
    <script type="text/javascript">
        $(function () {
            $('#btnDelete').click(function () {
                $.prompt("آیا برای حذف اطلاعات موجود اطمینان دارید ؟",
             {
                 title: 'Title',
                 buttons: { "بله": true, "خیر": false },
                 focus: 2,
                 submit: function (e, v, m, f) {
                     //debugger;
                     if (!v) {
                         return;
                     }
                     var postId = 10;
                     var fileName = "test.jpg"
                     $.ajax({
                         type: "POST",
                         url: '@postUrl',
                         data: JSON.stringify({ postId: postId, fileName: fileName }),
                         contentType: "application/json; charset=utf-8",
                         dataType: "json",
                         complete: function (xhr, status) {
                             var data = xhr.responseText;
                             if (status === 'error' || !data || data == "nok") {
                                 $.prompt("حذف فایل با خطا مواجه شده است !",
                                    {
                                        title: 'Title',
                                        buttons: { "بستن": true }
                                    });
                             }
                             else {
                                 $("#infoDiv").fadeOut("slow").remove();
                             }
                         }
                     });
                 }
             });
            });
        });
    </script>
}
```

نویسنده: مژده
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱۱:۳۵

سلام
ممنون که جواب دادید. خیلی لطف کردید
من تونستم با استفاده از یه js دیگه ، با callback مشکلمو حل کنم . اما حالا این js ، عنوان (title) نداره ! :D

از این سایت استفاده کردم :
/http://www.shiguenori.com/material/jquery.impromptu

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱۱:۴۴

کدی که نوشته شد بر اساس نسخه GitHub بود.

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۱۱/۰۶ ۲۱:۱۱

سلام ، اگه بخوایم مقدار ورودی‌های اکشنمون به صورت داینامیک باشه چیکار باید کنیم؟
منظورم به عنوان مثال مقادیر این خط در مثال شماست

```
var postId = 10;
var fileName = "test.jpg"
```

چنین مدلی دارم که میخوام دکمه‌ی حذفش به صورت Jquery Ajax باشه

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.IsRead)
        </td>
        <td>
            @Html.ActionLink("حذف پیام", MVC.User.ActionNames.DeleteMessageSend, MVC.User.Name, new{
id=item.Id }, new { @class="btn btn-danger" })
        </td>
    </tr>
}
```

ولی مقدار id رو باید به اکشن بفرستم ! اونم تو یه حلقه Foreach . باید کدهای جی کوئری تو حلقه باشه؟ به نظرتون راه حل چیه؟ ممنون

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۱۱/۰۶ ۲۱:۳۶

ببخشید تو این مطلب توضیح دادید.

عنوان: آموزش فایرباگ - #5 - HTML Panel
نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۸/۲۳ ۱۸:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, FireBug, CSS, XPATH, HTML

در این سری از مقالات آموزش FireBug ، به صورت ترتیبی پیش رفتیم و ابتدا توضیحات تقریبا مفصلی در مورد پنل Console دادیم.
اکنون به پرکاربردترین بخش آن ، یعنی پنل HTML می‌رسیم.

محتویاتی که در این پنل نمایش داده می‌شود ، کدهای صفحه‌ی جاری ، بصورت زنده است و با چیزی که از سمت سرور به مرورگر ارسال می‌شود متفاوت است ، تگ‌های HTML بصورت درختی مرتب شده اند و رنگی نمایش داده می‌شوند و امکان ویرایش محتوا ، ویرایش استایل ، مشاهده‌ی آرایش تگ‌ها بصورت بصری و ... وجود دارد.

نگاهی به کدهای ارسال شده به مرورگر در آدرس Google.com و نحوه‌ی نمایش آن در فایرباگ را ملاحظه بفرمایید.
(کدهای ارسال شده سمت چپ - نمایش در فایرباگ سمت راست)

این پنل به سه بخش اصلی تقسیم می‌شود :
بخش اصلی یا NodeView که محتوای صفحه را بصورت درختی و مرتب و رنگی نمایش می‌دهد.

Panel Toolbar که در بالای پنل قرار دارد.

Side Panels که شامل پنل‌های DOM , Layout , Computed , Style می‌شود.

که به ترتیب برای نمایش و ویرایش استایل‌ها ، مشاهده استایل‌های محاسبه شده ، مشاهده Layout یا آرایش و نمایش اطلاعات DOM تگ انتخاب شده در NodeView است.

در این مقاله با دو بخش NodeView و Panel Toolbar ، و در مقاله‌ی بعد با پنل‌های سمت راست یعنی Side Panels آشنا می‌شویم.

ویرایش HTML

هر تگ HTML از یک سری Attribute تشکیل می‌شود که در فایرباگ نام ویژگی بصورت آبی تیره و مقدار آن با رنگ قرمز مشخص شده است. برای ویرایش هریک از آن‌ها کافیست برویش کلیک کنید تا آن مقدار در یک باکس ویرایش به نمایش دربیاید.
با ویرایش مقدار ، این تغییر در لحظه بروی صفحه اعمال می‌شود.

برای اضافه کردن یک Attribute جدید به تگ هم بروی تگ مورد نظر راست کلیک می‌کنید و سپس گزینه‌ی New Attribute را انتخاب می‌کنید. ابتدا نام ویژگی ، یک Enter ، سپس مقدار را وارد می‌کنید. با زدن کلیدهای Enter متوالی ، می‌توانید به وارد کردن ویژگی‌ها ادامه دهید.
برای ویرایش کردن یک تگ و تگ‌های فرزندش ، بروی تگ موردنظر کلیک کنید تا به حالت انتخاب دربیاید ، سپس بروی دکمه‌ی Edit در بالای پنل کلیک کنید. در نهایت بعد از انجام ویرایش ، مجددا بروی دکمه‌ی Edit کلیک کنید.

Node View

NodeView نام بخش اصلی پنل HTML است که محتویات صفحه را بصورت مرتب شده و درختی نمایش می‌دهد.
تگ (Node) هایی که دارای فرزند می‌باشند ، یک علامت **+ یا -** در کنارشان وجود دارد که با کلیک بروی آن می‌توانید آن تگ را باز/بسته کنید. همچنین این کار با کلیدهای **+ و -** یا **Right Arrow و Left Arrow** از روی کیبورد هم قابل انجام است.
برای باز کردن یک لینک در این قسمت هم از ترکیب Ctrl و کلیک موس کمک بگیرید.
در نهایت زمانی که موس را بروی یک تگ قرار می‌دهید ، محیطی که توسط آن تگ در صفحه اشغال شده است بصورت رنگی مشخص می‌شود.
که رنگ زرد به معنی محیط margin ، رنگ آبی تیره به معنی محیط padding و رنگ آبی روشن هم به معنی محیط محتوا است.

Panel Toolbar

این قسمت در بالای پنل HTML قرار دارد که گزینه‌های زیر را دارا می‌باشد:

Break On Mutate

این دکمه امکان توقف کد JavaScript ای که سعی در ویرایش محتوای صفحه را دارد ، می‌دهد.
زمانی که FireBug تشخیص دهد که کدی سعی در ویرایش محتوا دارد ، شما را به خط مورد نظر از کد ، در پنل Script منتقل می‌کند.

Edit

این دکمه برای ویرایش مستقیم محتوای یک تگ بکار می‌رود
نکته‌ی جالب در ویرایش محتوا در فایرباگ این است که تغییرات در لحظه اعمال می‌شوند و نیاز به عمل بروزرسانیِ جداگانه نیست. برای مثال در همین قسمت Edit ، با هر ویرایش محتوا ، تغییرات در لحظه اعمال می‌شوند.

Element Path

زمانی که یک تگ را در FireBug انتخاب می‌کنید ، لیستی از تگ‌ها که از تگ جاری شروع و به تگ ریشه ختم می‌شود ، نمایش داده می‌شود که با کلیک بروی هرکدام ، همان به عنوان تگ فعلی یا انتخاب شده تعیین می‌شود.
نتیجه‌ی عملیاتی که بروی این تگ‌های انجام می‌دهید (حرکت موس و راست کلیک کردن) معادل همان عملیات بروی تگ‌های نمایش داده شده در قسمت اصلی (NodeView) است.

Options Menu

هر تب یا پنل در فایرباگ دارای یک سری تنظیمات است که Options Menu نام دارد. تب HTML هم دارای یک سری تنظیمات است که دانشتن آنها بسیار به شما کمک خواهد کرد.
این منو با کلیک کردن بروی فلش تب (

) یا راست کلیک کردن بروی تب ظاهر می‌شود.

Show Full Text

در صورت فعال بودن ، متون بصورت کامل نمایش داده می‌شوند ، در غیراینصورت بصورت خلاصه شده نمایش داده می‌شوند.

Show White Space

در صورت فعال بودن ، فضاهای خالی ، کرکترهای خط جدید و ... را نمایش می‌دهد.

```
<p>
    This menu is reachable via the little
    arrow in the panel tab (
  <a class="image" href="/wiki/index.php
    /File:OptionsMenuArrow.png">
    ) or by right-
    clicking on on the panel tab (since
    <a title="Firebug Release
    Notes" href="/wiki/index.php
    /Firebug_Release_Notes#Firebug_1.9">Fire
    bug 1.9</a>
    ).
</p>
```

Show Comments

در صورت فعال بودن ، کامنت‌ها را هم نمایش می‌دهد

```
<div id="jump-to-nav">
 <!-- start content -->
<div class="thumb tright">
  <p>The HTML panel displays the generated
  HTML/XML of the currently opened page.
```

سه گزینه ی Show Entities As Names ، Show Entities As Symbols و Show Entities As Unicode ، نوع نمایش کرکترهای ویژه را تعیین می‌کنند.

Highlight Changes

در صورت فعال بودن ، تگ تغییر یافته توسط JavaScript (یا تگ والدی که قابل مشاهده باشد) Highlight می‌شود

Expand Changes

در صورت فعال بودن ، زمان تغییر دادن یک تگ توسط JavaScript ، والدهای آن تگ باز (Expand) می‌شوند

Scroll Changes Into View

در صورت فعال بودن این گزینه ، هنگام تغییر یک تگ در صفحه توسط JavaScript ، قسمت NodeView به قسمت تغییر یافته حرکت می‌کند.

(فعال بودن این گزینه هنگام بررسی سایت هایی که اسلایدر یا سیستم هایی مشابه دارند ، باعث میشه که نتوانید بروی قسمت‌های سایت تمرکز کنید و مدام اسکرول به قسمت تغییرات منتقل می‌شود.)

Shade Box Model

در صورت فعال بودن ، فضای padding , margin و content را به شکلی که در بالا گفته شد نمایش می‌دهد ، در غیر اینصورت فقط یک خط دور تگ نشان می‌دهد.

Show Quick Info Box

در صورت فعال بودن ، یک پاپ‌آپ به همراه اطلاعات مختصری از تگ در صفحه نمایش می‌دهد.

Context Menu

اگر بروی یک تگ راست کلیک کنید ، یک منو نمایش داده می‌شود ، در این قسمت با گزینه‌های این منو آشنا می‌شویم.

Copy HTML

خود تگ و محتوایش را بصورت HTML در حافظه کپی می‌کند.

Copy innerHTML

محتوای تگ را در حافظه کپی می‌کند.

Copy XPath

آدرس XPath تگ را در حافظه کپی می‌کند.

Copy CSS Path

CSS Selector تگ را در حافظه کپی می‌کند.

Log Events

رویدادهای تگ را در پنل Console ثبت می‌کند. (کلیک موس ، فشردن کلید ، ...)
برای لغو لاگ کردن ، مجددا بروی تگ راست کلیک کرده و این گزینه را از حالت انتخاب خارج کنید.

Scroll Into View

صفحه را به جایی که تگ قابل نمایش است منتقل می‌کند.

New Attribute...
یک attribute جدید به تگ اضافه می‌کند.
برای لغو عملیات ، دکمه‌ی Esc را بزنید.

Edit Attribute "<attribute name>"...
اگر بروی یک attribute راست کلیک کرده باشید ، این گزینه و گزینه‌ی بعدی را مشاهده خواهید کرد.
معادل کلیک بروی نام attribute است ، نام را به حالت ویرایش درمی آورد.

Delete Attribute "<attribute name>"
attribute را حذف می‌کند.

Edit HTML...
تگ را به حالت ویرایش می‌برد.
معادل انتخاب تگ ، زدن کلید Edit است.

Delete Element
تگ را حذف می‌کند.
راه دیگر حذف یک تگ ، انتخاب تگ و فشردن کلید Del از کیبورد است.

Expand/Contract All
تگ و Childهایش را باز/بسته می‌کند. (بجز تگ های <link> , <style> , <script>)
می‌توان با ترکیب کلید **Shift + *** هم این کار را انجام داد که در این حالت تگ‌های فوق هم شامل باز/بسته شدن می‌شوند.

Break On Attribute Change
مانع اجرای کد JavaScript ای که attribute تگ را تغییر می‌دهد می‌شود و فایرباگ به خطی که باعث این عمل شده است در پنل Script منتقل می‌شود.
به عبارتی دیگر یک Break Point در خط JavaScript ای که باعث ویرایش attribute می‌شود قرار می‌دهد.

Break On Child Addition or Removal
مشابه توضیح قبل ، Break Point را در خطی که باعث اضافه/حذف شدن تگِ Child شده است قرار می‌دهد.

Break On Element Removal
مشابه توضیح قبل ، Break Point را در خطی که باعث حذف شدن تگ شده است قرار می‌دهد.

Inspect in DOM Tab
تگ فعلی را در پنل DOM ، برای بررسی باز می‌کند.

در قسمت بعدی با پنل‌های سمت راست (Side Panels) آشنا می‌شویم.

نظرات خوانندگان

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۱۲/۲۵ ۱۵:۰

در ورژن 1.11 از این افزونه ، امکانی برای paste کردن Html به صفحه ، به Context منوی پنل HTML اضافه شده که کار ویرایش محتویات Html را ساده‌تر کرده.
در حالی که قبلا برای این کار می‌بایست تگ را به حالت ویرایش درآورد و محتویات را اضافه کرد.

Copy HTML
Copy the HTML node and all its contents to the clipboard
Copy XPath
Con...
HTML
CSS
Edit
div < body < html
Style
Com...
Lay...
body
testcase.css (line 30)
Copy HTML
Copy innerHTML
Copy XPath
Copy CSS Path
Paste HTML
Log Events
Scroll Into View
New Attribute...
Edit HTML...
Delete Element
Del
Expand/Contract All
*
Break On Attribute Change
Break On Child Addition or Removal
Break On Element Removal
Inspect in DOM Panel
Replace Content
Replace Node
As First Child
As Last Child
Before
After
Paste after the node

عنوان: **صفحه کلید مجازی برای ورود اطلاعات**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۰۶ ۹:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, JavaScript, ASP.Net MVC, ASP.Net 4.5, Security

همانگونه که اطلاع دارید یکی از روش‌های سرقت اطلاعات استفاده از نرم افزارهای جاسوس صفحه کلید (Key Logger) است، البته ثبت کلیدهای فشرده شده می‌تواند توسط سخت افزارهایی که سر راه سوکت صفحه کلید و کیس قرار می‌گیره، انجام بشه. در صورتی که چنین سخت افزاری (مخصوصا در کافی نت‌ها) روی کامپیوتر کاربر نصب باشه، یا توسط ویروس و بدافزارها اینگونه نرم‌افزارهایی روی سیستم کاربر قرار بگیره هر کلیدی که توسط کاربر روی صفحه کلید فشرده میشه توسط اینها ثبت شده و در موقع مناسب برای فرد سازنده به طریقی (ایمیل یا ارتباط از طریق برنامه‌های مبتنی بر سوکت) حتی بسیاری از این برنامه‌ها پا را فراتر گذاشته و عنوان پنجره ای که کلیدها در آن فشرده شده نیز ثبت می‌شود (توسط توابع API ویندوز- البته اگر دوستان مایل باشن و از نظر مدیریت سایت ایرادی نداشته باشه، نحوه طراحی این نوع برنامه‌های جاسوس سخت افزار، صفحه کلید، یا ماوس آموزش می‌دم توی همین سایت)، حال برای امنیت برنامه‌های تحت وب یا ویندوز چگونه می‌توان در زمان ورود اطلاعات حساس مانند کلمه عبور یا شماره کارت اعتباری این امنیت را برای کاربر ایجاد کرد که داده‌هایش توسط این سخت افزارها یا بدافزارها جایی ثبت نشود؟

بله، حدس شما درست است استفاده از صفحه کلیدهای مجازی میشه گفت یکی از بهترین راه‌های ممکن هست، چون در این روش‌ها کلید به صورت سخت افزاری فشرده نمی‌شود (کلید فشرده شده به صف پیام‌های ویندوز نمی‌رود) در نتیجه نرم افزارها یا سخت افزارهای جاسوس نمی‌توانند این اطلاعات را ثبت کنند. و کاربر با خیال راحت می‌تواند داده‌های خود را وارد نمایند (تاکید می‌کنم این روش فقط جلو این نرم افزارها یا سخت افزارها را می‌گیرد و تضمینی برای اینکه در زمان ارسال داده‌های شما لو نرود ندارد).

خوب حال چه باید کرد؟

یک راه می‌تواند پیاده‌سازی صفحه کلید مجازی با کدهای طرف کلاینت مانند جاوا اسکریپت و وی‌بی اسکریپت است، اما گروهی پلاگینی را توسعه داده‌اند که با چند خط کدنویسی ساده به راحتی می‌توانید یک صفحه کلید مجازی چندزبانه (با هر زبانی که دلتون می‌خواد) داشته باشید و از اون در برنامه‌های خودتون استفاده کنید.

نحوه‌ی نصب:

ایتدا فایل‌های مورد نیاز را از سایت سازنده که شامل فایل جاوا اسکریپت ، فایل استایل و یک تصویر (آخرین نسخه) یا از این آدرس به صورت کامل (در حال حاضر نسخه 1.49) دریافت کرده، پس از دریافت فایل‌ها آنها را در هاست خود بارگزاری (آپلود) نمائید. سپس کدهای زیر را در صفحه‌ای که می‌خواهید صفحه کلید نمایش یابد در بین تگ <head> ... و </head> قرار دهید.

```
<script type="text/javascript" src="keyboard.js" charset="UTF-8"></script>
<link rel="stylesheet" type="text/css" href="keyboard.css">
```

حالا فقط کافی است به inputboxها و یا هر ورودی دیگر خود class=keyboardInput بدهید.

مثال:

```
<input type="text" value="" class="keyboardInput">
```

در نهایت کد صفحه شما باید اینگونه باشد:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript" src="keyboard.js" charset="UTF-8"></script>
    <link rel="stylesheet" type="text/css" href="keyboard.css"/>
</head>
<body>
    <input type="text" value="" class="keyboardInput"/>
</body>
```

```
</html>
```

با این کار پس از اجرای صفحه مورد نظر خروجی شما مانند تصویر زیر خواهد بود، جهت محدود کردن کلیدها و عملیات دلخواه و سفارشی سازی با پارامترهای دلخواه می‌تواند از دموهای موجود در سایت سازنده بهره بگیرید.

موفق ومؤید باشید.

نظرات خوانندگان

نویسنده: مجتهدی
تاریخ: ۱۳۹۱/۰۹/۰۶ ۱۸:۰

سلام
تا جایی که یادم میاد، برنامه هایی هم بودن که نقاطی که موس روشون کلیک کرده بود رو ذخیره می‌کردن (چطوریش رو نمی‌دونم!)
اگه همچین برنامه هایی وجود داشته باشن حتی اگه مثل صفحات پرداخت آنلاین بانک‌ها که جای ۱۰ کاراکتر عدد، تغییر می‌کنه، با آنالیز شماره کارت‌ها و مسائلی از قبیل اینکه هر بانک با شماره خاصی شروع می‌شه و... ، فکر کنم بدست آوردن اطلاعات حساس کار سختی نباشن...
درست می‌گم؟

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۰۷ ۱۱:۵۸

دوست من کار نشد نداره، امنیت نسبی هست
بله میشه این کار انجام داد به راحتی امکان پذیر هست، اما ذخیره نقاط چه کاربردی می‌تونی داشته باشه؟

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۹/۰۷ ۱۶:۴۶

مطلب مفیدی بود.
ممنونم

نویسنده: nojirom
تاریخ: ۱۳۹۱/۱۰/۱۹ ۱۵:۵۸

با تشکر از آقای فتح اللهی عزیز بابت این مطلب فقط زبان پیش فرض کیبورد راچگونه فارسی تنظیم کنیم

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۰/۲۰ ۷:۳۷

سلام
دوست من در فایل اسکریپت مزبوطه اوایل فایل این خط پیدا کرده

```
this.VKI_kts = this.VKI_kt = "US International";  // Default keyboard layout
```

و آن را به هر زبانی که دلخواه شماست تغییر دهید مثلا برای فارسی اینگونه است:

```
this.VKI_kts = this.VKI_kt = "\u0641\u0627\u0631\u0633\u06cc";  // Default keyboard layout
```

البته من توی همون فایل "fa" جستجو کردم و عنوان اون در این قسمت قرار دادم
موفق وموید باشید

نویسنده: حیدری
تاریخ: ۱۳۹۱/۱۲/۱۸ ۲۰:۵۶

سلام

مرسی ، دستتون درد نکنه
ایشالا همیشه موفق باشید

نویسنده: مرتضی محمودی
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۰:۱۵

درود
منو راهنمایی کنین که چطور از این کلاس برای RadTextBox یا دیگر Inputهای تلریک استفاده کنم
باتشکر....

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۰:۵۷

چه فرقی می‌کنه؟ به خاصیت یا کلاس CSS اش مقدار keyboardInput رو نسبت بده. تمام این کنترل‌های مصنوعی، نهایتا به inputهای استاندارد HTML تبدیل و رندر می‌شن. وگرنه مرورگر نمی‌تونه اون‌ها رو تشخیص بده یا پردازش کنه.

نویسنده: مرتضی محمودی
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۲:۲

آخه سمت سرور که میره دیگه کلاس میپره. من تکست هام داخل AjaxPanel هست.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۳:۱۲

دوباره باید بایندش کنی.

نویسنده: مهیار نظری پور
تاریخ: ۱۳۹۳/۰۴/۲۵ ۱۹:۱۳

بی نهایت درود؛ بنده، به یک کیبورد، به شکل کِی پَد (تنها اعداد، بصورت ماشین حسابی) نیازمندم.
برای درج در فیلد "شماره تماس" فُرم رزروشِن سایتم.
در نت خیلی گشتم، امّا چیزی نیافتم.
دقیقاً چیزی مثل کِی پَد درگاه آنلاین بانک، که برای درج شماره حساب و این‌ها بکار میره روی سایت.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۳/۰۴/۲۷ ۰:۳۹

سلام
خوب میتوانید از همان کدهای درگاه بانک الگو برداری کنید
تلاشتون بکنید اگر نتوانستید من کدش براتون بگیرم
موفق و موید باشید

عنوان: **غیر فعال کردن یک دکمه در حین انجام پردازش های سمت سرور**
نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۱/۰۹/۱۲ ۵:۴۹
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, JavaScript, Security

بعضی مواقع بهتر است یک دکمه در حال انجام پردازش‌های سمت سرور غیر فعال شود و وقتی عملیات سمت سرور به پایان رسید این دکمه دوباره فعال شود. غیر فعال کردن یک دکمه به این دلیل انجام می‌شود که از postbackهای مجدد در حین postback شدن صفحه جلوگیری شود.
فرض کنید در رویداد کلیک یک دکمه کدی نوشته اید که اطلاعات یک دانشجو را ذخیره کند. کاربر نرم افزار بعد از یک بار کلیک روی دکمه، درخواستی به سرور می‌فرستد و این باعث می‌شود کدهای درون رویداد کلیک دکمه اجرا شوند، به دلیل این که کاملا این کدها اجرا نشده اند، صفحه هم postback نشده است. کاربر به هر دلیل صبر نمی‌کند تا این پردازش تمام شود (شاید او نمی‌داند عملیاتی در سمت سرور در حال اجرا شدن هست چون پیغامی به او نشان داده نشده است)، در نتیجه باز هم روی همین دکمه کلیک می‌کند و باعث می‌شود در حین پردازش کدهای درون دکمه توسط سرور، دوباره درخواستی به سمت سرور فرستاده شود و باز هم کدهای درون همین دکمه به اجرا در آیند. به عبارت بهتر در حین postback شدن صفحه دوباره درخواست postback کردن صفحه را می‌دهد و سرور هم در حین انجام کدهای درون دکمه با درخواست قبلی، دوباره کدهای درون رویداد کلیک را اجرا می‌کند. در این مثال این کار باعث می‌شود کدهای رویداد کلیک دکمه، به تعداد کلیک‌های انجام شده اجرا شوند. در نتیجه به همین تعداد می‌تواند یک دانشجوی مستقل در دیتابیس ذخیره شود. این می‌تواند مشکلات بسیاری را همراه داشته باشد.

به همین دلیل بهتر است با کلیک روی این دکمه این کارها اتفاق بیفتند:
الف) غیرفعال کردن دکمه در حین انجام پردازش‌های سمت سرور
ب) نشان دادن یک پیغام به کاربر در حین انجام پردازش‌های سمت سرور
ج) فعال کردن دکمه بعد از انجام پردازش‌های سمت سرور

برای غیرفعال کردن دکمه در حین انجام پردازش‌های سمت سرور نمی‌توان از کدهای سمت سرور استفاده کرد. چون تا کاملا صفحه postback نشود نمی‌توان این کدها را به صفحه اعمال کرد. پس این گزینه کنار می‌رود.
راه حل بسیار خوب استفاده از جاوا اسکریپت است. مثال زیر را ببینید:

```
<asp:Button runat="server"
            ID="btnProcess"
            Text="پردازش"
            onclick="btnProcess_Click"
            OnClientClick="this.disabled = true; this.value = 'در حال پردازش اطلاعات ...';"
            UseSubmitBehavior="false"
             />
<asp:Label runat="server" ID="lblMessage" Text=""></asp:Label>
```

در رویداد OnClientClick کارهای (الف) و (ب) انجام می‌شوند و با false کردن مقدار رویداد UseSubmitBehavior کار (ج) انجام می‌شود.

و در رویداد کلیک دکمه کد زیر را بنویسید:

```
protected void btnProcess_Click(object sender, EventArgs e)
{
    // insert student in database
    System.Threading.Thread.Sleep(2000);
    lblMessage.Text = "پردازش اطلاعات به پایان رسید";
}
```

در این رویداد باید یک دانشجو اضافه شود. برای ایجاد یک پردازش سمت سروری دو ثانیه ای از متد Sleep استفاده شده است.

کد برگرفته شده از : dotnetforum.lk

نظرات خوانندگان

نویسنده: مجتبی صحرائی
تاریخ: ۱۳۹۱/۰۹/۱۲ ۶:۹

به نظر میاد وقتی تو صفحه validator داشته باشیم و validator جلوی postback شدن صفحه رو تحت شرایطی میگیره؛ دکمه در حالت غیر فعال باقی خواهد موند

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۱/۰۹/۱۲ ۷:۳۱

در صورتی که بتونیم با بررسی‌های سمت سروری جلوی postbackهای همزمان رو بگیریم امنیت بالاتری خواهیم داشت.
(منظورم استفاده از custom validator هست که در سمت سرور و بسته به شرایطی که گفتید validate می‌شود)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۱۲ ۸:۳۸

برای رفع این مشکل (غیرفعال کردن سازگار با اعتبار سنجی سمت کاربر) باید به نحو زیر عمل کرد:

```
UseSubmitBehavior="false"
OnClientClick="if ((typeof(Page_ClientValidate) == 'function') && (Page_ClientValidate()==false)) {
return false;} this.disabled=true;"
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۲۴ ۲۰:۳۶

ممنون از کار شما
من این روش تست کردم اما زمانی که با یک تاییده همراه باشه کد تقریبا از کار می‌افته، صفحه به سرور پست میشه اما عملیاتی صورت نمی‌گیره.
کدی که نوشتم:

```
$("input[type=submit]").click(function () {
                if ((typeof (Page_ClientValidate) == 'function') && (Page_ClientValidate() == false)) {
                     return false;
                }
                if (!confirm("آیا مطمئن هستید؟"))
                    return false;
                this.disabled = true;
                this.value = 'در حال پردازش اطلاعات ...';
                return true;
            });
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۴ ۲۱:۱

در jQuery برای لغو ارسال به سرور، بهتر است بجای return false از e.preventDefault استفاده شود:

```
....click(function(e) {
        e.preventDefault() // stop the automatic form submission
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۲۵ ۸:۳۰

سلام این روش استفاده کردم (با return و بدون return)
اما جواب نمی‌ده و فرم ارسال میشه، عملیات هم انجام میشه چه کاربر تایید کنه یا نکنه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۵ ۹:۴۱

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
    CodeBehind="Default.aspx.cs" Inherits="WebAppSingleSubmit._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
    <script type="text/javascript">
        $(document).ready(function () {
            $("input[type=submit]").click(function (e) {
                if ((typeof (Page_ClientValidate) == 'function') && (Page_ClientValidate() == false)) {
                    return false;
                }
                if (!confirm("آیا مطمئن هستید؟")) {
                    return false;
                }
                this.disabled = true;
                this.value = 'در حال پردازش اطلاعات ...';

                __doPostBack($(this).attr('name'), '');
            });
        });
    </script>
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <p>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="TextBox1"
            ErrorMessage="RequiredFieldValidator"></asp:RequiredFieldValidator>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
    </p>
</asp:Content>
```

نویسنده: بهاره
تاریخ: ۱۳۹۲/۱۱/۱۷ ۱۹:۴۴

سلام
آیا روشی هست که بشه جلوی کلیک کاربر روی دکمه‌ی refresh مرورگر (IE) را گرفت؟
امکانش هست که این دکمه را غیرفعال کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۷ ۱۹:۵۴

« الگوی PRG در ASP.NET MVC »

نویسنده: بهاره
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۳:۵

ممنون. ولی من از MVC استفاده نمی‌کنم. پروژه در قالب SharePoint 2007 است. Redirect به 303 در asp.net وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۳:۲۸

- مهم داشتن یک ایده اولیه برای شروع و جستجو هست.
- معادل return RedirectToAction در آن کدها Response.Redirect است در وب فرم‌ها. مابقی اصول الگوی PRG تفاوتی نمی‌کند. البته Redirect در اینجا از status code=302 استفاده می‌کند؛ ولی اکثر مرورگرها آن‌را همانند 303 پردازش خواهند کرد.

عنوان: **مدیریت کلیدهای کیبرد در جاوا اسکریپت**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۹/۱۲ ۲۳:۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Keyboard Events

با پیشرفت بسترهای موجود در زمینه شبکه و اینترنت، گرایش به استفاده از اپلیکیشنهای تحت وب روز به روز بیشتر میشود. با گسترش این برنامه‌ها نیازها و درنتیجه ابزارهای موجود توسعه پیدا می‌کنند. درحال حاضر ابزارها و نیز محیطهای توسعه مختلفی برای تولید این اپلیکیشنها وجود دارد. به دلیل نوع رابط کاربری موجود در این برنامه‌ها (اکثراً مرورگرهای وب مثل اینترنت اکسپلورر، گوگل کروم، فایرفاکس و ...) استفاده از زبانهای سمت کلاینت (مثل جاوا اسکریپت که در تمامی مرورگرهای مدرن پشتیبانی کاملی از آن میشود) جایگاه ویژه ای در این نوع برنامه‌ها دارد. درضمن وقتی صحبت از اپلیکیشن به میان می‌آید استفاده از کلیدهای میانبر کیبرد برای راحتی کار کاربران کاربرد ویژه ای دارد. اما متاسفانه زبان جاوا اسکریپت به دلیل محدودیتهایی منطقی موجود، پشتیبانی مناسبی از رویدادهای کیبرد ندارد و مشکل تفاوتها و تناقضات میان سخت افزارها، سیستم عامل‌ها و مرورگرها هم به این مسئله بیشتر دامن میزند. در مطلب جاری هدف این است تا آشنایی مقدماتی با این مبحث فراهم شود.

رویدادهای کیبرد

در جاوا اسکریپت سه رویداد زیر برای کلیدهای کیبرد وجود دارد (به ترتیب زمان رخ دادن):

keydown : زمانی که یک کلید فشرده می‌شود.

keypress : زمانی که یک کلید کاراکتری فشرده می‌شود.

keyup : زمانی که یک کلیدِ فشرده شده، رها می‌شود.

یک تفاوت اساسی میان رویدادهای keydown و keypress در جاوا اسکریپت وجود دارد: رویداد keydown پس از فشردن هر کلیدی روی کیبرد رخ میدهد و یک کد مخصوص آن کلید (scan code ^) را ارائه میدهد. اما رویداد keypress که بعد از keydown رخ میدهد کد کاراکتر آن کلید (char code) را ارائه میدهد، بنابراین تنها برای کلیدهای کاراکتری بدرستی کار میکند. برای درک بهتر کد زیر را در یک فایل html ذخیره کرده و در مرورگرهای مختلف آزمایش کنید:

```
<html>
<body>
  <div>
    Prevent default:
    <input type="checkbox" id="keydownStop" value="1" />
    keydown   
    <input type="checkbox" id="keypressStop" value="1" />
    keypress   
    <input type="checkbox" id="keyupStop" value="1" />
    keyup
  </div>
  Ignore:
  <input type="checkbox" id="keydownIgnore" value="1" />
  keydown    
  <input type="checkbox" id="keypressIgnore" value="1" />
  keypress    
  <input type="checkbox" id="keyupIgnore" value="1" />
  keyup
  <div>
    Focus on the input below and press any key.
  </div>
  <div>
    <input type="text" style=" width: 600px" id="keyInput" />
  </div>
  Log:
  <div>
    <textarea id="keyLogger" rows="18" onfocus="this.blur()" style="width: 600px; border: 1px solid
black"></textarea>
  </div>
  <input type="button" value="Clear" onclick="clearLog()" />
  <script type="text/javascript">
    document.getElementById('keyInput').onkeydown = keyHandler;
    document.getElementById('keyInput').onkeyup = keyHandler;
    document.getElementById('keyInput').onkeypress = keyHandler;
    document.getElementById('keyInput').focus();
    function keyHandler(e) {
      e = e || window.event;
      if (document.getElementById(e.type + 'Ignore').checked) return;
```

```
      var evt = e.type;
      while (evt.length < 10) evt += ' ' +
        log(evt +
          ' keyCode=' + e.keyCode +
          ' which=' + e.which +
          ' charCode=' + e.charCode +
          ' char=' + String.fromCharCode(e.keyCode || e.charCode) +
          (e.shiftKey ? ' +shift' : '') +
          (e.ctrlKey ? ' +ctrl' : '') +
          (e.altKey ? ' +alt' : '') +
          (e.metaKey ? ' +meta' : ''));
      if (document.getElementById(e.type + 'Stop').checked) {
        e.preventDefault ? e.preventDefault() : (e.returnValue = false);
      }
    }
    function clearLog() {
      document.getElementById('keyLogger').value = '';
      document.getElementById('keyInput').focus();
    }
    function log(text) {
      var area = document.getElementById('keyLogger');
      area.value += text + '\n';
      area.scrollTop = area.scrollHeight;
    }
  </script>
</body>
</html>
```

نکته: برای جلوگیری از اجرای مرورگرها در حالت Quirks حتما از تگ doctype در ابتدای فایلهای html خود استفاده کنید. درغیراینصورت رفتارهای غیرمنتظره ای (مخصوصا در IE) مشاهده خواهید کرد. برای اجرای مرورگرها در حالت استاندارد html5 (بهترین حالت در حال حاضر) میتوانید از تگ زیر استفاده کنید:

```
<!doctype html>
```

دقت کنید که قبل از این خط هیچ چیز دیگری نوشته نشود وگرنه در IE از آن صرفنظر میشود!

یا اینکه در IE با استفاده از developer tools (دکمه F12) برای Document Mode گزینه ای غیر از Quirks mode (بهتر است از حالت IE9 یا بالاتر استفاده کنید) را انتخاب کنید.

برای کسب اطلاعات بیشتر راجع به doctypeهای مختلف و نیز حالت quirks میتوانید به ^ و ^ و ^ و ^ رجوع کنید. پیشنهاد میکنم که این منابع را حتما مطالعه کنید.

نکته : در کد بالا متد preventDefault در IE 8- تعریف نشده است (درواقع در IE تنها در نسخه 9 تعریف شده است). همچنین استفاده از پراپرتی returnValue در فایرفاکس و IE9 کار نمیکند! از این خط کد برای جلوگیری از رفتار پیشفرض رویداد استفاده شده است. همانطور که در ادامه میخوانید راه حل ساده‌تری نیز برای اینکار وجود دارد.

متد String.fromCharCode برای نمایش کاراکتر کلید فشرده شده استفاده شده است. البته اگر کلید غیرکاراکتری فشرده شود ممکن است با نتایج غیرمنتظره ای روبرو شوید.

با استفاده از html تولیدی در مرورگرهای مختلف سعی کنید موارد زیر را آزمایش کنید:

کلیدهای کاراکتری چون 6 { | / a را بفشارید. در این حالت رویدادهای keydown و سپس keypress رخ خواهند داد. پس از رها کردن کلیدها نیز رویداد keyup رخ میدهد.

یکی از کلیدهای غیرکاراکتری مثل ctrl یا alt را بفشارید. در این حالت تنها رویدادهای keydown و keyup رخ خواهند داد و خبری از رویداد keypress نیست.

نکته : مرورگرهای FireFox و Opera در مورد بیشتر کلیدهای غیرکاراکتری نیز رویداد keypress را صدا خواهند زد! مرورگر IE این رفتار را تنها در مورد کلید Esc نشان میدهد. همچنین در IE و Opera کلید PrtScr هیچ رویدادی را فرا نمیخواند. ظاهرا تنها مرورگر Chrome بدرستی عمل میکند.

درحالت کلی فشردن کلیدهای غیرکاراکتری نباید رویداد keypress را فراخوانی کند.

بنابراین:

keydown و keyup برای همه کلیدها

keypress برای کلیدهای کاراکتری

پراپرتی‌های رویدادهای کیبرد

برخلاف نسخه‌های قدیمی مرورگرها که هرکدام راه و روش خودشان را برای تعامل با این رویدادها برگزیده بودند، امروزه تمامی مرورگرها تقریبا از یک روش استاندارد و مشترک برای اینکار استفاده میکنند. در تصاویر زیر تمام اجزای آبجکت KeyboardEvent با استفاده از کد زیر در مرورگرهای اپرا و کروم نشان داده شده است. استفاده از کد زیر در مرورگرهای فایرفاکس و IE نتایج جالبی مانند تصاویر زیر فراهم نمیکند!

```
document.onkeydown = function (e) {
      e = e || event;
      console.log(e);
    }
```

Elements Resources Netw

```
▼ KeyboardEvent
    altGraphKey: false
    altKey: false
    bubbles: true
    cancelBubble: false
    cancelable: true
    charCode: 0
    clipboardData: undefined
    ctrlKey: false
    currentTarget: null
    defaultPrevented: false
    detail: 0
    eventPhase: 0
    keyCode: 123
    keyIdentifier: "F12"
    keyLocation: 0
    layerX: 0
    layerY: 0
    metaKey: false
    pageX: 0
    pageY: 0
    returnValue: true
    shiftKey: false
  ▶ srcElement: <body>
  ▶ target: <body>
    timeStamp: 1354001129293
    type: "keydown"
  ▶ view: Window
    which: 123
  ▶ __proto__: KeyboardEvent
>
```

<top frame>

همانطور که مشاهده میکنید تفاوتهایی بین مرورگرها در این آبجکت به چشم میخورد. برای کسب اطلاعات بیشتر در مورد این آبجکت و اجزای استاندارد آن در DOM Level 3 به اینجا مراجعه کنید. در ادامه به بررسی پراپرتی‌های مهم آرگومان این رویدادها (همان KeyboardEvent) میپردازیم.

keyCode

همان scan code کلید فشرده شده است. برای مثال اگر کلید a فشرده شود کاراکتر تولیدی ممکن است a یا A یا 'ش' (یا کاراکتری دیگر در زبانهای مختلف) باشد اما در تمامی حالات scan code مربوطه یا همان keyCode همیشه یکسان (65 برای کلید a) خواهد بود. این کد تنها به کلید فشرده شده بستگی دارد و نه به کاراکتر حاصله! البته در IE به هنگام رخ دادن رویداد keypress کد کاراکتر (همان char code) کلید فشرده شده در این پراپرتی قرار میگیرد!

در این بین میان مرورگرهای مختلف تفاوتهایی وجود دارد که با یک جستجو در اینترنت میتوان به تمامی این کدها دسترسی پیدا کرد. خواندن مقاله کامل JavaScript Madness: Keyboard Events نیز خالی از لطف نیست.

charCode

همان کد ASCII (یا کد UTF-16 برای کاراکترهای یونیکد. اطلاعات بیشتر ^ و ^) کاراکتر کلید فشرده شده است. ممکن است با keyCode برابر باشد. این پراپرتی در IE و Opera تعریف نشده است.

در عمل ممکن است keyCode و charCode در پلتفرمهای مختلف و حتی بین سیستم عامل‌های مختلف در یک سخت افزار نتایج متفاوتی ارائه دهند. بنابراین آزمودن هر مورد مشکوک قبل از ریلیز نهایی محصول میتواند مفید باشد.

which

یک پراپرتی نسبتا غیراستاندارد! است که ترکیبی از keyCode و charCode را برمیگرداند (اطلاعات بیشتر در ^ و ^). این پراپرتی در IE تعریف نشده است.

shiftKey, ctrlKey, altKey, metaKey

پراپرتی هایی از نوع بولین که وضعیت کلیدهای Shift, Ctrl, Alt و Command (تنها در مک) را نشان میدهند.

نکته: بدستن آوردن کد درست کاراکتر فشرده شده با توجه به وضعیت کلیدها و زبان انتخابی تنها با استفاده از رویداد keypress امکان پذیر است. با توجه به تفاوتهایی که بین مرورگرهای مختلف وجود دارد برای یافتن کاراکتر فشرده شده در رویداد keypress استفاده از متد زیر توصیه میشود:

```
function getCharacter(event) {
  if (event.which == null)
     return String.fromCharCode(event.keyCode);    // IE
  else if (event.which != 0 && event.charCode != 0)
     return String.fromCharCode(event.which);  // All others
  return null; // special key
}
```

توضیحات بیشتر و کاملتر این مبحث را میتوانید از Document Object Model (DOM) Level 3 Events Specification که آخرین نسخه آن در زمان تهیه این مطلب در تاریخ June 2012 14 انتشار یافته تهیه کنید. بطور ویژه برای مبحث رویدادهای کیبرد (^) اطلاعات بسیار بیشتری در دسترس است.

نکته : با توجه به اطلاعاتی که در سند فوق وجود دارد، به دلیل ماهیت همزمانی (Sync) رویدادهای کیبرد تا زمانی که تمام عملیات موجود در متد تعیین شده برای این رویدادها انجام نشود، رویداد بعدی (با توجه به ترتیبی که در ابتدای این مطلب آورده شده است) رخ نخواهد داد. برای تست این موضوع قطعه کد زیر را آماده کردم:

```
<html>
<body>
  <input id="inputText" type="text"  />
  <script>
    document.onkeydown = keyDown;
    document.onkeypress = keyPress;
    document.onkeyup = keyUp;
    function keyDown(e) {
      var input = document.getElementById('inputText');
      input.value = 'keyDown started ...';
      input.disabled = true;
      var j = 0;
```

```
      for (var i = 0; i < 999999999; i++) {
        j = i - j;
      }
      console.log(j);
      //alert('keyDown');
      input.value = 'keyDown finished.';
      input.disabled = false;
    }
    function keyPress(e) {
      alert('keyPress');
      //console.log('keyPress');
    } function keyUp(e) {
      alert('keyUp');
      //console.log('keyUp');
    }
  </script>
</body>
</html>
```

اگر کد بالا در مرورگرهای مختلف امتحان کنید مشاهده میکنید که انجام عملیات سنگین در رویدادهای کیبرد موجب ایجاد وقفه در فراخوانی سایر رویدادها میشود.

با اجرای کد فوق در مرورگرهای مختلف نکات جالب زیر بدست آمد:

- در IE و کروم نمایش یک alert موجب از دست دادن فوکس document شده و بنابراین رویدادهای کیبرد بعد از نمایش alert کار نخواهند کرد! مثلا اگر در keydown یک alert نمایش داده شود چون رویداد keyup بر روی پنجره alert رخ میدهد بنابراین keyup فراخوانی نمیشود ولی چون رویداد keypress با keydown همزمان است این اتفاق برای keypress نمی‌افتد. این مشکل در فایرفاکس پیش نمی‌آید. در اپرا در این حالت رویداد keypress هم رخ نمیدهد! البته رفتار IE در اجرای کد فوق کمی غیرمنتظره‌تر است و ظاهرا رویداد keypress هم رخ نمیدهد.

- در اجرای کد فوق در FireFox ظاهرا alert مربوط به keyup قبل از keypress نمایش داده میشود. البته اگر به جای alert از console.log (البته نیاز به نصب Firebug است) استفاده شود این به هم خوردگی ترتیب رویدادها وجود ندارد.

- در مرورگر Opera پس از فشردن کلید enter در نوار آدرس و پس از بارگذاری صفحه، فوکس بلافاصله به عنصر document سپرده میشود طوریکه که رویداد keyup کلید enter در document بارگذاری شده فراخوانی میشود و درصورت سرعت بالای بارگذاری صفحه، کدهای مروبوط به این رویداد اجرا میشوند. در سایر مرورگرها این مورد مشاهده نشد.

- ظاهرا در تمام مرورگرها به غیر Opera کدهای جاوا اسکریپ در ثرد UI اجرا شده و موجب قفل شدن document میشود.

بنابراین وسط اجرای کدهای سنگین نمیتوان مثلا خواص عناصر UI را تغییر داد. درواقع مرورگر اپرا برخلاف سایر مرورگرها، رفتار ویژه ای در برخورد با جاوا اسکریپت و رویدادها و تغییرات عناصر DOM دارد. برای کسب اطلاعات بیشتر در این زمینه به اینجا مراجعه کنید. درضمن این مبحث کمی پیچیده‌تر از آن است که به نظر می‌آید(^). برای بررسی بیشتر میتوانید کد زیر را در مرورگرهای مختلف آزمایش کنید:

```
<html>
<head>
  <script type="text/javascript">
    function process() {
      var above = 0, below = 0;
      for (var i = 0; i < 200000; i++) {
        if (Math.random() * 2 > 1) {
          above++;
        }
        else {
          below++;
        }
      }
    }
    function test() {
      var result1 = document.getElementById('log');
      var start = new Date().getTime();
      console.log('start');
      for (var i = 0; i < 200; i++) {
        result1.value = 'time=' + (new Date().getTime() - start) + ' [i=' + i + ']';
        process();
      }
      result1.value = 'time=' + (new Date().getTime() - start) + ' [done]';
      console.log('end');
    }
    window.onload = test;
  </script>
```

```
</head>
<body>
  <input id='log' />
</body>
</html>
```

اگر کد فوق را در مرورگرهایی غیر از اپرا اجرا کنید میبینید که تنها نتیجه نهایی نمایش داده میشود و فرایندهای میانی درون حلقه نمیتوانند تغییری در محتوای UI ایجاد کنند. طبق انتظار کروم از بقیه بسیار سریعتر بوده و سپس IE و پس از آن فایرفاکس قرار دارد. اما در مورد Opera وضع کاملا فرق میکند و به دلیل به روز رسانی همزمان UI عملیات بسیار بسیار کندتر از بقیه مرورگرها به اتمام میرسد.

نکته : حالات استثنایی دیگری هم در اجرای کدهای مشابه در مرورگرهای مختلف پیش می‌آید که به دلیل پیچیده کردن بیش از حد بحث آورده نشده اند. فقط ذکر این نکته الزامی است که درحال حاضر میزان تفاوت رفتار مرورگرهای مختلف دربرخورد با کدهای یکسان قابل ملاحظه است. بنابراین در هنگام توسعه سعی کنید حداقل در این چهار مرورگر معروف آزمایشات خود را به سرانجام برسانید.

و در ادامه چند مثال ...

غیرفعال کردن ورودی کاربر

برای اینکار فقط کافی است در رویدادهای keydown یا keypress مقدار false برگشت داده شود. البته در مروگر Opera برخی از کلیدها از این رفتار پیروی نمیکنند. مثل کاراکترهای '`' و '+' و '=' که درصورت برگشت false در رویداد keydown باز هم رفتار پیش فرض را از خود نشان میدهد. اما برگرداندن مقدار false در رویداد keypress این مشکل را حل میکند(این موارد در نسخه 11.51 تست شدند). میتوانید با استفاده از کد زیر در تمام مرورگرها این موارد را آزمایش کنید:

```
<input onkeydown="return false" />
<input onkeypress="return false" />
```

تبدیل به حروف بزرگ

```
document.getElementById('myInputText').onkeypress = function (e) {
  var char = getCharacter(e || window.event);
  if (!char) return; // special key
  this.value += char.toUpperCase();
  return false; // برای اینکه کاراکتر اضافی نمایش داده نشود
}
```

در کد فوق متد getCharacter در بالا در قسمت پراپرتی‌های آرگومان رویداد نشان داده شده است. کد فوق چندان کامل نیست و همیشه کاراکتر فشرده شده را بدون توجه به موقعیت کرسر کیبرد در انتهای متن قرار میدهد.

نکته: استفاده از عبارتی چون e || window.event به این دلیل است که در مرورگر IE آرگومان رویداد (همان e که به آن implicit event object نیز میگویند) به متد مربوطه ارسال نمیشود (تا نسخه 9 که تست کردم مسئله به همین صورت است) در عوض پراپرتی‌های این آرگومان از طریق window.event (یا همان event که به آن explicit event object نیز میگویند) در دسترس هستند. این فیلد در واقع آرگومان آخرین رویداد رخ داده در پنجره جاری را در خود ذخیره میکند. اما در سایر مرورگرها به صورت استاندارد مقدار این آرگومان به عنوان پارامتر رویداد به متد مربوطه ارسال میشود. جالب است که بدانید مرورگرهای Opera و Chrome از هر دو روش پشتیبانی میکنند. مرورگر فایرفاکس تنها از ارسال آرگومان رویداد به متد مربوطه پشتیبانی میکند.

عبارت e ||window.event درواقع شکل دیگر عبارت زیر است:

```
e ? e : window.event;
```

در جاوا اسکریپت اگر عملگر مقایسه ای در عبارت مقایسه آورده نشود مقدار عبارت با false مقایسه میشود. این مقایسه از نوع abstract است (در ادامه این مطلب توضیح داده شده است). در جاوا اسکریپت 0 و رشته خالی و null و undefined و امثال اینها در مقایسه abstract برابر false درنظر گرفته میشوند. بنابراین در عبارت مقایسه بالا اگر مقدار e مثلا undefined (در IE)

باشد مقدار window.event بازگشت داده میشود و درغیراینصورت خود e برگشت داده میشود.

تنها عدد

```
document.getElementById('numberInputText').onkeypress = function (e) {
  e = e || window.event;
  var chr = getCharacter(e);
  if (!isNumeric(chr) && chr !== null) return false;
}
function isNumeric(n) {
  return !isNaN(parseFloat(n)) && isFinite(n);
}
function isNumber(val) {
  return val !== "NaN" && (+val) + '' === val + ''
}
```

در کد بالا متد isNumeric از اینجا گرفته شده است. متد isNumeric جی کوئری (که از نسخه 1.7 اضافه شده است) هم دقیقا از این روش استفاده میکند.
متد دوم یعنی isNumber (که در کد فوق از آن استفاده نشده است) روش دیگری را برای اطمینان از مقدار عددی بودن استفاده میکند. در این روش درصورتیکه val یک مقدار عددی نباشد val+ برابر NaN میشود که در نهایت عبارت مقایسه ای مقدار false را برمیگرداند. البته به غیر از خود مقدار NaN که در شرط اول مورد بررسی قرار گرفته است. برای کسب اطلاعات بیشتر در مورد رفتار نسبتا عجیب جاوا اسکریپت با NaN و متد isNaN به اینجا سر بزنید.
نکته : تفاوت == با === (یا =! و ==!) - در جاوا اسکریپت دو روش برای مقایسه مقادیر متغیرها وجود دارد. اولی (== یا =!) مقایسه را با تبدل نوع داده‌ها انجام میدهد (که اصطلاحا به آن type-converting equality comparison یا abstract comparison میگویند) و دومی (=== یا ==!) مقایسه را با مقادیر واقعی و بدون تبدیل نوع داده انجام میدهد (که به آن equality without type coercion یا strict equality comparison گفته میشود). در واقع در مقایسه strict تنها وقتی که دو متغیر از یک نوع باشند ممکن است مقدار true برگشت داده شود. برای روشنتر شدن مطلب به مثالهای زیر توجه کنید (^): 0 == false // true 0 === false // false, because they are of a different type 1 == "1" // true, auto type coercion 1 === "1" // false, because they are of a different type
توضیحات بهتری در اینجا آورده شده است.
برای کسب اطلاعات کاملتر میتوانید به ECMAScript Language Specification و قسمت مقایسه strict و abstract مراجعه کنید. (رابطه‌ها و تفاوت‌های میان ECMAScript و JavaScript و JScript و ActionScript در اینجا آورده شده است.)

کار با Scan Code و Char Code

همانطور که قبلا هم اشاره شد میان کد کاراکتر و کلید فشرده شده بر روی کیبرد تفاوت وجود دارد. برای بهره برداری از کلیدهای میانبر در صفحات وب به کد کلید فشرده شده (همان scan code) نیاز است و نه به کد کاراکتر آن. همچنین کلیدهای ویژه و غیرکاراکتری دارای کد کاراکتر نیستند. بیشتر مرورگرها رویداد Keypress را برای کلیدهای غیرکاراکتری فرا نمیخوانند. بنابراین برای این موارد رویدادهای keydown و keyup مفید هستند. امروز تمام مرورگرها از جدول کدهای یکسانی برای scan codeها استفاده میکنند که نمونه‌های آن را میتوانید در ^ و ^ و ^ مشاهده کنید. نکته ای که باید درباره پراپرتی keyCode یادآوری شود این است که جدای از وضعیت کیبرد (مثل زبان یا موقعیت کلید capsLock) در تمام حالات در ازای فشردن شده یک کلید خاص (یا ترکیبی از کلیدهای کنترلی با سایر کلیدها) همواره باید یک کد یکسان برگشت داده شود.
پس یک charCode کد یونیکد کاراکتر کلید فشرده شده است که تنها در رویداد keypress در دسترس است. یک keyCode کد خود کلید فشرده شده یا همان scan code است که در رویدادهای keydown و keyup در دسترس است.
نکته: برای تمام کلیدهای الفبایی-عددی scan code با char code یکی است. مثلا برای حروف الفبایی scan code یک کلید با کد ASCII حرف انگلیسی بزرگ آن کلید برابر است.
بنابراین برای بررسی کلید ترکیبی ctrl + a میتوان از کد زیر در رویداد keydown استفاده کرد:

```
e.ctrlKey && e.keyCode == 'A'.charCodeAt(0)
```

و فرقی نمیکند که کاراکتر حاصله 'a' یا 'A' یا 'ش' باشد.
نکته: برای تمام کلیدهای کیبرد به غیر از ';' و '=' و '-' تمام مرورگرها از کدهای یکسانی استفاده میکنند. میتوانید این کلیدها را با

استفاده از قطعه کد اول این مطلب در مرورگرهای مختلف آزمایش کنید.

نکته: با برگشت مقدار false در رویداد keydown یا keypress رفتار پیشفرض کلید یا ترکیب کلیدهای مربوطه غیرفعال میشود. البته به غیر از عملیاتهای سطح سیستم عامل (مثل alt+F4).

چند مثال دیگر در ادامه ...

جابجایی تصویر

کدهای زیر را در یک فایل html ذخیره کرده و توسط یک مرورگر فایل حاصله را باز کنید.

```
<html>
<body>
  <div id="dotnettips" style="width: 35px; height: 35px; background-image:
url(http://dotnettips.info/favicon.ico);
    position: absolute; left: 10px; top: 10px;" tabindex="0">
  </div>
  <script>
    document.getElementById('dotnettips').onkeydown = function (e) {
      e = e || event;
      switch (e.keyCode) {
        case 37: // left
          this.style.left = parseInt(this.style.left) - this.offsetWidth + 'px';
          return false;
        case 38: // up
          this.style.top = parseInt(this.style.top) - this.offsetHeight + 'px';
          return false;
        case 39: // right
          this.style.left = parseInt(this.style.left) + this.offsetWidth + 'px';
          return false;
        case 40: // down
          this.style.top = parseInt(this.style.top) + this.offsetHeight + 'px';
          return false;
      }
    }
  </script>
</body>
</html>
```

بر روی تصویر کلید کرده (یا با استفاده از Tab فوکس را روی تصویر قرار دهید) و با استفاده از کلیدهای مکان نما (Arrow keys) سعی کنید تصویر را در صفحه جابجا کنید.

در کد فوق برای جلوگیری از رفتار پیش فرض کلیدهای مکان نما (جابجایی محتوای صفحه در صورت وجود اسکرول) مقدار false برگشت داده شده است.

نکته : استفاده از خاصیت tabindex برای امکان فوکس بر روی div اجباری است.

نکته : استفاده از event به جای window.event تا زمانی که یک متغیر در اسکوپ جاری با نام event وجود نداشته باشد مشکلی ایجاد نمیکند.

Caps Lock

متاسفانه راه مستقیمی برای دریافت وضعیت کلید caps lock در جاوا اسکریپت وجود ندارد. ظاهرا تنها راه حل موجود برای این مسئله بررسی کد کاراکتر کلید فشرده شده در رویداد keypress و بررسی آن با توجه به وضعیت پراپرتی shift این رویداد است. بدین ترتیب که اگر کد کاراکتر مربوط به حروف بزرگ بوده درحالیکه کلید شیفت نگه داشته نشده است بنابراین کلید Caps Lock روشن است و بالعکس. البته با ترکیب این روش و نیز رصد scan code کلید Caps Lock (کد آن برابر 20 است) در رویداد keydown میتوان وضعیت بهتری پدید آورد. قطعه کد زیر برای اینکار است. در این کد از یک متغیر گلوبال برای نگهداری وضعیت دکمه caps lock استفاده میشود.

```
<html>
<body>
  <input id="inputText" type="text" />
  <div id="divCapsLock" style="color: Red; display: none;">Caps Lock is ON!</div>
  <script>
    var capsLock = null;
    document.onkeypress = keyPress;
    document.onkeydown = keyDown;
    function keyDown(e) {
      e = e || event;
```

```
      capsLock = (e.keyCode == 20 && capsLock !== null) ? !capsLock : capsLock;
      document.getElementById('divCapsLock').style.display = capsLock ? 'block' : 'none';
    }
    function keyPress(e) {
      if (capsLock != null) return;
      e = e || window.event;
      var charCode = e.charCode || e.keyCode;
      capsLock = (charCode >= 97 && charCode <= 122 && e.shiftKey) || (charCode >= 65 && charCode <= 90
&& !e.shiftKey);
      document.getElementById('divCapsLock').style.display = capsLock ? 'block' : 'none';
    }
  </script>
</body>
</html>
```

در متد رویداد kepress تکست باکس ابتدا بررسی میشود که آیا قبلا متغیر گلوبال capsLock مقداری غیرنال دارد. اگر مقدار غیرنال داشته باشد دیگر نیازی نیست تا کد کاراکتر کلید بررسی شود زیرا وضعیت جاری کلید معلوم است. بنابراین در ادامه کار صرفه جویی میشود. دلیل استفاده از رویدادهای سطح docment به جای خود تکست باکس این است تا از کوچکترین فرصتها! برای تعیین وضعیت جاری کلید caps lock استفاده شود. یعنی بتوان پس از فشرده شدن اولین کلید کاراکتری بدون توجه به موقعیت فوکس در صفحه، این وضعیت را از حالت نامشخص خارج کرد.

نکته : در سلسله مراتب رویدادهای اجزای یک document همیشه ابتدا رویدادهای مربوط به عناصر فرزند فراخوانده میشود و رویدادهای مربوط به عناصر document و window در پایان صدا زده میشوند. برای آزمودن این مورد قطعه کد زیر را در مرورگرهای مختلف امتحان کنید:

```
<html>
<body>
  <div id='divInput'>
    <input id="inputText" type="text" />
  </div>
  <script>
    document.getElementById('inputText').onkeydown = inputKeyDown;
    document.getElementById('divInput').onkeydown = divKeyDown;
    document.onkeydown = documentKeyDown;
    window.onkeydown = windowKeyDown;
    function divKeyDown(e) {
      console.log('divKeyDown');
    }
    function inputKeyDown(e) {
      console.log('inputKeyDown');
    }
    function documentKeyDown(e) {
      console.log('documentKeyDown');
    }
    function windowKeyDown(e) {
      console.log('windowKeyDown');
    }
  </script>
</body>
</html>
```

نکته : اگر از تگ doctype استفاده نشود (همانطور که در ابتدای این مطلب اشاره شد)، در IE رویدادهای عنصر window فراخوانی نمیشوند.

درهرصورت برای مشخص کردن وضعیت کلید caps lock نیاز است تا کاربر ابتدا کلیدی را بفشارد و در حال حاضر روش و راه حل دیگری وجود ندارد! درضمن اگر صفحه کلیدی غیر از انگلیسی استفاده شود روش فوق جواب نخواهد داد و باید بررسی‌های بیشتری انجام شود. البته در مورد زبانهایی چون فارسی که روشن یا خاموش بودن caps lock تاثیری در کاراکتر حاصله ندارد کاری نمیتوان کرد و نمیتوان از وضعیت caps lock باخبر شد! هرچند در این موارد معمولا وضعیت این دکمه مهم نیست زیرا بیشترین کاربرد این گونه هشدارها در ورودی‌های رمز عبور است در صورتیکه زبان صفحه کلید انگلیسی (یا مشابه آن) باشد. برای اینکار کد زیر به عنوان راه حل بهتر پیشنهاد میشود:

```
<html>
<body>
  <input id="inputText" type="text" />
  <div id="divCapsLock" style="color: Red; display: none;">Caps Lock is ON!</div>
  <script>
    var capsLock = null;
```

```
    var hasFocus = false;
    document.onkeyup = keyUp;
    document.onkeypress = keyPress;
    document.getElementById('inputText').onfocus = focus;
    document.getElementById('inputText').onblur = focus;
    function warnCapsLock() {
      document.getElementById('divCapsLock').style.display = (capsLock != null && capsLock && hasFocus)
? 'block' : 'none';
    }
    function focus() {
      hasFocus = !hasFocus;
      warnCapsLock();
    }
    function keyUp(e) {
      e = e || event;
      capsLock = (e.keyCode == 20 && capsLock !== null) ? !capsLock : capsLock;
      warnCapsLock();
    }
    function keyPress(e) {
      if (capsLock != null) return;
      e = e || window.event;
      var charCode = e.charCode || e.keyCode;
      capsLock = (charCode >= 97 && charCode <= 122 && e.shiftKey) || (charCode >= 65 && charCode <= 90
&& !e.shiftKey);
      warnCapsLock();
    }
  </script>
</body>
</html>
```

اگر دقت کنین میبیند که رویداد keydown در این کد نهایی با keyup جایگزین شده است. درصورت استفاده از رویداد keypress یک مشکل کوچک بوجود می‌آید و آن این است که اگر کاربر کلید caps lock را فشرده و سپس آن را در حالت فشرده نگه دارد اتفاق بدی خواهد افتاد! در این حال چون رویداد keydown مرتبا فراخوانده میشود وضعیت متغیر capsLock در این کد کاملا نامعتبر خواهد بود. بنابراین بهتر است تا از رویداد keyup که تنها یکبار فراخوانده میشود استفاده شود.

نکته : اگر کاربر پس از مشخص شدن وضعیت کلید Capa Lock در این کد، فوکس را به پنجره دیگری تغییر داده و سپس وضعیت این دکمه در آنجا تغییر دهد این کد دیگر درست کار نخواهد کرد. برای حل این مشکل هم راه حل زیر وجود دارد:

```
window.onblur = function () { capsLock = null; }
```

با این کار وضعیت متغیر capsLock ریست میشود.

نکته : در آزمایش این کد دقت کنید که زبان کیبرد حتما انگلیسی باشد.

مدیریت کلیدها

برای مدیریت کلیدهای کیبرد راههای متنوعی وجود دارد. مثلا میتوان از قطعه کد زیر استفاده کرد:

```
var Client = {};
Client.Keyboard = {};
Client.Keyboard.EnableKeyDown = true;
Client.Keyboard.EnableKeyUp = false;
Client.Keyboard.CurrentKeyEvent = null;

window.onkeydown = function (event) {
  if (!Client.Keyboard.EnableKeyDown) return true;
  return KeyboardEvents(event);
};

window.onkeyup = function (event) {
  if (!Client.Keyboard.EnableKeyUp) return true;
  return KeyboardEvents(event);
};

function Rise(event) {
  var e = Client.Keyboard.CurrentKeyEvent;
  if (event) {
    var data = { shift: e.shiftKey, ctrl: e.ctrlKey, alt: e.altKey };
    if (!event(data)) return false;
    return event(data);
  }
```

```
  return true;
}

function KeyboardEvents(e) {
  e = e || window.event;
  Client.Keyboard.CurrentKeyEvent = e;
  switch (e.keyCode) {
    case 8:
      return Rise(Client.Keyboard.Backspace);
    case 9:
      return Rise(Client.Keyboard.Tab);
    case 13:
      return Rise(Client.Keyboard.Enter);
    case 16:
      return Rise(Client.Keyboard.Shift);
    case 17:
      return Rise(Client.Keyboard.Ctrl);
    case 18:
      return Rise(Client.Keyboard.Alt);
    case 19:
      return Rise(Client.Keyboard.Pause);
    case 20:
      return Rise(Client.Keyboard.CapsLock);
    case 27:
      return Rise(Client.Keyboard.Esc);
    case 33:
      return Rise(Client.Keyboard.PageUp);
    case 34:
      return Rise(Client.Keyboard.PageDown);
    case 35:
      return Rise(Client.Keyboard.End);
    case 36:
      return Rise(Client.Keyboard.Home);
    case 37:
      return Rise(Client.Keyboard.Left);
    case 38:
      return Rise(Client.Keyboard.Up);
    case 39:
      return Rise(Client.Keyboard.Right);
    case 40:
      return Rise(Client.Keyboard.Down);
    case 44:
      return Rise(Client.Keyboard.PrtScr);
    case 45:
      return Rise(Client.Keyboard.Insert);
    case 46:
      return Rise(Client.Keyboard.Delete);

////////////////////////////////////////////////////////////////////////////////////////////////////
    case 48:
      return Rise(Client.Keyboard.Num0);
    case 49:
      return Rise(Client.Keyboard.Num1);
    case 50:
      return Rise(Client.Keyboard.Num2);
    case 51:
      return Rise(Client.Keyboard.Num3);
    case 52:
      return Rise(Client.Keyboard.Num4);
    case 53:
      return Rise(Client.Keyboard.Num5);
    case 54:
      return Rise(Client.Keyboard.Num6);
    case 55:
      return Rise(Client.Keyboard.Num7);
    case 56:
      return Rise(Client.Keyboard.Num8);
    case 57:
      return Rise(Client.Keyboard.Num9);

////////////////////////////////////////////////////////////////////////////////////////////////////
    case 65:
      return Rise(Client.Keyboard.A);
    case 66:
      return Rise(Client.Keyboard.B);
    case 67:
      return Rise(Client.Keyboard.C);
    case 68:
      return Rise(Client.Keyboard.D);
    case 69:
      return Rise(Client.Keyboard.E);
```

```
    case 70:
      return Rise(Client.Keyboard.F);
    case 71:
      return Rise(Client.Keyboard.G);
    case 72:
      return Rise(Client.Keyboard.H);
    case 73:
      return Rise(Client.Keyboard.I);
    case 74:
      return Rise(Client.Keyboard.J);
    case 75:
      return Rise(Client.Keyboard.K);
    case 76:
      return Rise(Client.Keyboard.L);
    case 77:
      return Rise(Client.Keyboard.M);
    case 78:
      return Rise(Client.Keyboard.N);
    case 79:
      return Rise(Client.Keyboard.O);
    case 80:
      return Rise(Client.Keyboard.P);
    case 81:
      return Rise(Client.Keyboard.Q);
    case 82:
      return Rise(Client.Keyboard.R);
    case 83:
      return Rise(Client.Keyboard.S);
    case 84:
      return Rise(Client.Keyboard.T);
    case 85:
      return Rise(Client.Keyboard.U);
    case 86:
      return Rise(Client.Keyboard.V);
    case 87:
      return Rise(Client.Keyboard.W);
    case 88:
      return Rise(Client.Keyboard.X);
    case 89:
      return Rise(Client.Keyboard.Y);
    case 90:
      return Rise(Client.Keyboard.Z);
      ////////////////////////////////////////////////////////////////////////
    case 91:
      //case 219: // opera
      return Rise(Client.Keyboard.LeftWindow);
    case 92:
      return Rise(Client.Keyboard.RightWindow);
    case 93:
      return Rise(Client.Keyboard.ContextMenu);
      ////////////////////////////////////////////////////////////////////////
    case 96:
      return Rise(Client.Keyboard.NumPad0);
    case 97:
      return Rise(Client.Keyboard.NumPad1);
    case 98:
      return Rise(Client.Keyboard.NumPad2);
    case 99:
      return Rise(Client.Keyboard.NumPad3);
    case 100:
      return Rise(Client.Keyboard.NumPad4);
    case 101:
      return Rise(Client.Keyboard.NumPad5);
    case 102:
      return Rise(Client.Keyboard.NumPad6);
    case 103:
      return Rise(Client.Keyboard.NumPad7);
    case 104:
      return Rise(Client.Keyboard.NumPad8);
    case 105:
      return Rise(Client.Keyboard.NumPad9);
      ////////////////////////////////////////////////////////////////////////
    case 106:
      return Rise(Client.Keyboard.Multiply);
    case 107:
      return Rise(Client.Keyboard.Add);
    case 109:
      return Rise(Client.Keyboard.Subtract);
    case 110:
      return Rise(Client.Keyboard.DecimalPoint);
    case 111:
```

```
      return Rise(Client.Keyboard.Divide);
      ////////////////////////////////////////////////////////////////////////
    case 112:
      return Rise(Client.Keyboard.F1);
    case 113:
      return Rise(Client.Keyboard.F2);
    case 114:
      return Rise(Client.Keyboard.F3);
    case 115:
      return Rise(Client.Keyboard.F4);
    case 116:
      return Rise(Client.Keyboard.F5);
    case 117:
      return Rise(Client.Keyboard.F6);
    case 118:
      return Rise(Client.Keyboard.F7);
    case 119:
      return Rise(Client.Keyboard.F8);
    case 120:
      return Rise(Client.Keyboard.F9);
    case 121:
      return Rise(Client.Keyboard.F10);
    case 122:
      return Rise(Client.Keyboard.F11);
    case 123:
      return Rise(Client.Keyboard.F12);
      ////////////////////////////////////////////////////////////////////////
    case 144:
      return Rise(Client.Keyboard.NumLock);
    case 145:
      return Rise(Client.Keyboard.ScrollLock);
    case 186:
    case 59: // opera & firefox
      return Rise(Client.Keyboard.SemiColon);
    case 187:
    case 61: // opera
      //case 107: //firefox
      return Rise(Client.Keyboard.Equal);
    case 188:
      return Rise(Client.Keyboard.Camma);
    case 189:
      return Rise(Client.Keyboard.Dash);
    case 190:
      return Rise(Client.Keyboard.Period);
    case 191:
      return Rise(Client.Keyboard.Slash);
    case 192:
      return Rise(Client.Keyboard.GraveAccent);
    case 219:
      return Rise(Client.Keyboard.OpenBracket);
    case 220:
      return Rise(Client.Keyboard.BackSlash);
    case 221:
      return Rise(Client.Keyboard.CloseBracket);
    case 222:
      return Rise(Client.Keyboard.SingleQuote);
  }
};
```

نحوه استفاده از این کد به صورت زیر است:

```
// ctrl + s
  Client.Keyboard.S = function (e) {
    if (e.ctrl) {
      // انجام عملیات موردنظر
    }
    return true;
  }
```

اگر در متد الصاق شده به پراپرتیهای Client.Keyboard هیچ مقداری برگشت داده نشود، باتوجه به کد موجود در متد Rise، عملیات پیشفرض کلید مربوطه در پنجره مرورگر غیرفعال خواهد شد. بنابراین در کد بالا بعد از شرط، مقدار true برگشت داده شده است.

هرچند سعی کردم که تفاوت میان مرورگرهای مختلف را درنظر بگیرم ولی احتمال اینکه برخی از کدها به دلیل تفاوت میان

مرورگرهای مختلف در کد بالا آورده نشده باشد وجود دارد!

نکته: کد فوق بیشتر برای روشن‌تر شدن موضوع ارائه شده است چون راه حل‌های بهتری هم برای مدیریت کلیدهای کیبرد وجود دارد. مثل استفاده از کتابخانه های Mousetrap یا HotKey یا jQuery Hotkeys یا KeyboardJS . البته در این میان بهترین کتابخانه موجود به نظر من همان Mousetrap است که تنها کتابخانه موجود است که علاوه بر پشتیبانی از ترکیب کلیدها با کلیدهای کنترلی، از توالی کلیدها (keys sequence) نیز پشتیبانی میکند که کار جالبی است.

در پایان این نکته را یادآور میشوم که امروزه توسعه اپلیکیشنهای تحت وب بدون استفاده مناسب از امکانات سمت کلاینت طرفداری ندارد. پس بهتر است هرچه بیشتر در مورد این زبان مرموز و پر از رمز و راز (JavaScript) بدانیم.

نظرات خوانندگان

نویسنده: فرهاد یزدان پناه
تاریخ: ۱۳۹۱/۱۱/۱۴ ۲۱:۲

دستتون درد نکنه
مقاله خیلی کامل و جامعی بود.
بسیار سود بردم.

نویسنده: moh
تاریخ: ۱۳۹۲/۰۳/۲۷ ۰:۷

با سلام
امکانش هست در مورد این تابع که دقیقا چه کاری انجام میدهد توضیح دهید preventDefault <.....
تفاوتش با return false چی هست

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۳/۲۷ ۱۱:۵۰

سوال خوبی است. قبل از پاسخ به سوال شما باید کمی درباره رویدادها در عناصر DOM توضیح داده بشه.
در صفحات html به صورت پیش فرض، هر رویدادی که در عناصر زیردست یا فرزند رخ میدهد به ترتیب در تمام عناصر والدش انتشار می‌یابد. به این فرایند event propagation یا event bubble up میگویند.
در حالت عادی برای تغییر رفتار پیش فرض مدیریت رویدادهای عناصر DOM در مرورگرها، با استفاده از زبان جاوا اسکریپت چند روش وجود دارد.
مثلا آرگومان رویداد (همان متغیر معروف e) در مرورگرهای مدرن دو متد برای اینکار دارد:

```
e.preventDefault()
e.stopPropagation()
```

متد preventDefault همانطور که از نامش هم پیداست از اجرای رفتار پیش فرض رویداد جاری جلوگیری میکند. مثلا با قرار دادن این متد در رویداد مربوط به کلیک یک لینک، رفتار پیش فرض کلیک روی لینک (همان انتقال به آدرس تنظیم شده در پراپرتی src لینک) اجرا نمیشود.
متد stopPropagation هم همانطور که از نامش پیداست از انتشار رویداد جاری به عناصر والدش جلوگیری میکند.
راه حل دیگری که برای تغییر این رفتارهای پیش فرض وجود دارد استفاده از return flase در انتهای رویداد است. این کار تاثیری همانند استفاده از متد preventDefault دارد. تفاوت این دو روش آن است که برگشت مقدار false تنها میتواند در انتهای یک متد استفاده شود بنابراین اگر خطایی در متد مربوطه رخ دهد احتمال اجرا نشدن این خط و درنتیجه اجرای ادامه فرایند با اجرای رفتار پیش فرض رویداد مربوطه وجود دارد. اما متد preventDefault را میتوان در ابتدای متد رویداد استفاده کرد و خیال خود را راحت کرد!
نکته: دو متد اشاره شده در نسخه‌های قدیمی مرورگرها (مثلا IE 8 و قبلتر) وجود ندارد!
برای آزمایش تمامی این موارد میتوان از کد زیر که برای اینکار آماده کردم استفاده کرد:

```
<html>
<body>
<div onclick="document.body.appendChild(outter)" style='border:solid 1px'>
    Outer Div
    <a id="inner" href="index2.htm">index2</a>
</div>
<script>
var outter = document.createElement('div');
outter.innerText = 'outter';
var inner = document.createElement('div');
inner.innerText = 'inner';

document.getElementById('inner').onclick = function(e) {
```

```
    e = e || window.event;
    document.body.appendChild(inner);
    //e.preventDefault();
    //e.stopPropagation();
    //e.returnValue = false;
    return false;
};
</script>
</body>
</html>
```

البته درصورت استفاده از jQuery به دلیل اینکه این کتابخانه از نمونه ای سفارشی برای آرگومان رویدادها استفاده میکند فرایندهای موجود کمی تفاوت دارد.

اطلاعات بیشتر: ^ و ^

عنوان: **ایجاد توابع بی‌نام خود اجرا شونده در جاوا اسکریپت**
نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۱۰/۲۰ ۲۲:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript

همان‌طور که می‌دانید توابع نیز در جاوا اسکریپت از نوع اشیا هستند بنابراین تمامی قوانین حاکم بر یک شیئ در مورد آن‌ها نیز صادق است به مثال ساده‌ی زیر توجه کنید:

```
function showMessage(msg)
{
    alert(msg);
}
```

مثال فوق تابعی ساده است که پیامی را به عنوان ورودی دریافت و سپس آن را چاپ می‌کند حال ما می‌توانیم به جای نام یک متغیر را به عنوان delegate برای این تابع ایجاد کنیم:

```
var d=function (msg)
{
    alert(msg);
}
```

در این حالت می‌توان با استفاده از این متغیر در واقع تابع را فراخوانی نمود:

```
d("Hessam");
```

حال همان‌طور که می‌توانیم برای اشیاء از عملگرها استفاده کنیم برای این متغیر نیز می‌توانیم این کار را انجام دهیم:

```
(d)("Hessam");
```

حال در صورتی‌که تابع را تنها یک بار نیاز داشته باشیم می‌توانیم بدنه آن‌را مستقیما داخل پرانتز قرار دهیم:

```
(function (msg)
{
    alert(msg);
})("Hessam");
```

تابع فوق یک تابع بی‌نام خود اجرا شونده است در مقالات بعد کاربردهای را برای این نوع توابع شرح خواهم داد

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">
           // بودن صفحه عنصر مورد نظر ایجاد می‌شود Reday در زمان
           $(function(){
              $("<p>Hi there!</p>").insertAfter("#followMe");
           });
        </script>
    </head>
<body>
     <p id="followMe">Follow me!</p>
</body>
</html>
```

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

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

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

موفق وموید باشید

عنوان: آموزش (jQuery) جی کوئری #3
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۰۸ ۱۲:۳۰
آدرس: www.dotnettips.info
گروه‌ها: 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.

1-انتخاب عناصر صفحه برای ایجاد تغییر

اولین قدم برای استفاده از هر گونه تابع jQuery، مشخص کردن و انتخاب عناصری است که می‌خواهیم تابع روی آن عناصر اعمال شود. گاهی اوقات انتخاب این مجموعه عناصر با یک توضیح ساده مشخص می‌شود، برای مثال "تمام عناصر پاراگراف موجود در صفحه". اما گاهی اوقات مشخص کردن این مجموعه نیاز به توضیح پیچیده‌تری دارد، برای مثال "تمام عناصر لیست در صفحه که دارای کلاس listElement هستند و لینکی دارند که اولین عضو آن لیست می‌باشد".
خوشبختانه jQuery یک مکانیزم بسیار قوی و قدرتمند 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 انخاب می‌کند که شناسه آن specialID باشد، به شرط آنکه این عنصر <a> باشد و دارای کلاس specialClass نیز باشد را انتخاب می‌کند.

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

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

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

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

به جز چند مورد خاص که استثنا وجود دارد، CSS3 و jQuery 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 <ul>، 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.

نظرات خوانندگان

نویسنده: هادی
تاریخ: ۱۳۹۳/۰۸/۲۶ ۱۱:۳۵

با سلام

من می‌خوام value انتخاب شده یک تگ Select رو بخونم، این کار رو با دستورات زیر در مرور گر IE جواب گرفتم:

```
$("#ddlPriortiy option:selected").val()
or
 $("#ddlPriortiy").val()
```

ولی در مرورگر فایر فاکس به من مقدار undefined رو بر میگردونه!

مشکل کجاست؟

اینم بگم این مشکل وقتی رخ میده که من این کنترل رو بصورت داینامیک درست میکنم و در صفحه میرزم.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۸/۲۶ ۱۳:۳۵

باید با jQuery live آشنا باشید. البته اسمش جدیدا شده on و live حذف شده، اما مفهومش یکی هست.

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

توضیح	فیلتر
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	:first
آخرین عنصری که با شرط ما مطابقت کند را انتخاب می‌کند. li a:last آخرین لینک از فرزندان لیست را برمی گرداند.	:last
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.	:first-child
آخرین فرزند عنصر که با شرط ما مطابقت می‌کند را انتخاب می‌کند. li a:last-child اولین عنصر لینک از هر لیست را	:last-child

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

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 انخاب 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

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

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

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: Cookie - قسمت دوم
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۲/۱۳ ۲۲:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Cookie

کوکی در جاوا اسکریپت

همانطور که در قسمت قبل اشاره کوتاهی شد، مدیریت کوکی‌های در دسترس در وضعیت جاری، در جاوا اسکریپت ازطریق پراپرتی cookie از شی document امکان‌پذیر است. این پراپرتی کاری همانند هدرهای Set-Cookie و Cookie (که در قسمت قبل درباره آن‌ها بحث شد) انجام می‌دهد. این پراپرتی یک مورد کاملا استثنایی و نسبتا عجیب در زبان جاوا اسکریپت است. در نگاه اول ظاهرا document.cookie از نوع رشته است، اما قضیه کاملا فرق می‌کند. برای روشن شدن مطلب به ادامه بحث توجه کنید.

افزودن کوکی

- برای افزودن یا ویرایش یک کوکی باید از ساختاری مانند ساختار هدر Set-Cookie که چیزی شبیه به عبارت زیر است، پیروی کرد:

```
document.cookie = "name=value; expires=date; domain=theDomain; path=thePath; secure";
```

نکته: با توجه به توضیحاتی که در قسمت قبل ارائه شد، بدیهی است که امکان ثبت یک کوکی با فلگ HttpOnly در جاوا اسکریپت وجود ندارد!

اجرای دستوری شبیه با ساختار نشان داده شده در بالا، موجب حذف کوکی‌های قبلی نمی‌شود. از این دستور برای ایجاد یک کوکی و یا ویرایش یک کوکی موجود استفاده می‌شود. کوکی‌های ایجادشده با این روش تفاوتی با کوکی‌های ایجادشده توسط هدر Set-Cookie ندارند و همانند آنها در درخواست‌های بعدی با توجه به خواص تنظیم شده، به سمت سرور ارسال خواهند شد.

همانطور که مشاهده می‌کنید خاصیت‌های کوکی به صورت جفت‌های نام-مقدار درون یک رشته به document.cookie نسبت داده می‌شوند. این خاصیت‌ها توسط یک کاراکتر ; از یکدیگر جدا می‌شوند. شرح ساختار فوق در زیر آورده شده است:

1. همیشه اولین جفتِ نام-مقدار همانند مثال بالا **باید** «عنوان و مقدار» کوکی را مشخص سازد. این قسمت تنها عضو اجباری ساختار فوق است.
2. سپس یک سمی‌کالن و یک فاصله
3. تاریخ انقضا (expires) **یا** حداکثر طول عمر کوکی (max-age)
4. سپس یک سمی‌کالن و یک فاصله
5. دمین و یا مسیر مربوط به کوکی
6. سپس یک سمی‌کالن و یک فاصله
7. سایر خواص چون Secure

نکته: این ساختار عجیب معرفی شده را **عینا** رعایت کنید. بقیه کار توسط مرورگر انجام خواهد شد.

نکته: قسمت‌های مختلف این ساختار case-sensitive **نیست،** البته به‌جز نام کوکی که کاملا case-sensitive است.

مثلا برای ثبت یک کوکی با عنوان myCookie و مقدار myValue و دمین d.com و مسیر test و طول عمر 5 روزه باید از دستور زیر استفاده کرد:

```
document.cookie = 'myCookie=myValue; max-age=432000; domain=d.com; path=/test';
```

خواندن کوکی

- برای خواندن کوکی‌ها تنها کافی است مقدار پراپرتی document.cookie بررسی شود. با اینکه از دستور نشان داده شده در بالا اینگونه برمی آید که پراپرتی document.cookie به رشته معرفی شده مقداردهی شده است، اما به محض خواندن این پراپرتی چیزی شبیه به عبارت زیر برگردانده میشود:

```
myCookie=myValue
```

از بقیه خواص اثری نیست! این رفتار به دلیل حفظ امنیت کوکی‌ها در تمام مرورگرها رعایت می‌شود.

- برای ثبت کوکی دیگری در وضعیت جاری کافی است یکبار دیگر دستور بالا را برای کوکی جدید به کار ببریم. مثلا به صورت زیر:

```
document.cookie = 'mySecondCookie=mySecondValue; path=/'
```

اینار یک کوکی سشنی بدون دمین و با مقدار / برای مسیر کوکی ثبت می‌شود! در این حالت کوکی قبلی دوباره نویسی و یا حذف نمی‌شود و تنها یک کوکی جدید به لیست کوکیهای مرورگر اضافه می‌شود! این رفتار عجیب از ویژگی‌های جالب document.cookie است.

- اگر مقدار document.cookie در این حالت خوانده شود مقدار زیر برگشت داده می‌شود:

```
myCookie=myValue; mySecondCookie=mySecondValue
```

باز هم خبری از سایر خاصیت‌ها نیست. ولی همانطور که می‌بینید کوکی دوم به لیست کوکی‌های مرورگر اضافه شده است.

نکته: عبارت برگشت داده شده از پراپرتی document.cookie همانند مقداری است که در هدر Cookie هر درخواست توسط مرورگر گنجانده می‌شود، یعنی جفت نام-مقدار کوکی‌ها به همراه یک ; و یک فاصله بین مقادیر هر کوکی. بنابراین برای بدست آوردن مقدار یک کوکی یکسری عملیات جهت Parse کردن داده‌های آن نیاز است!

متدها

امروزه کتابخانه‌های متعددی با استفاده از زبان جاوا اسکریپت برای برنامه نویسی سمت کلاینت وجود دارد که بیشتر آنها قابلیت‌هایی برای کار با کوکی‌ها نیز دارند. ازجمله می‌توان به jQuery و YUI اشاره کرد. پلاگین مخصوص کوکی‌ها در jquery در اینجا بحث شده است. برای کسب اطلاعات بیشتر درباره قابلیت‌های کار با کوکی در YUI نیز به اینجا مراجعه کنید. مطالب زیر صرفا برای روشن شدن بحث ارائه می‌شوند. بدیهی است که برای کارهای عملی بهتر است از کتابخانه‌های موجود استفاده شود.
با توجه به اطلاعات بالا از متدهای زیر می‌توان برای خواندن، افزودن و حذف کوکی‌ها استفاده کرد.

نکته: متدهای زیر از ترکیب چندین ریفرنس مختلف بدست آمده است. هرچند برای موارد خاص‌تر می‌توانند بیشتر سفارشی شوند.

افزودن و یا ویرایش کوکی

```
function setCookie(data, value) {
  if (typeof data === "string") {
    data = { name: data, value: value };
  };
  if (!data.name) throw "Cookie's name can not be null.";

  var cookie = escape(data.name) + "=" + escape(data.value);

  var expDate = null;
  if (data.expDays) {
    expDate = new Date();
    expDate.setDate(expDate.getDate() + data.expDays);
  }
  else if (data.expYear && data.expMonth && data.expDay) {
    expDate = new Date(data.expYear, data.expMonth, data.expDay);
  }
  else if (data.expires) {
    expDate = data.expires;
  }
  else if (data.maxAge) {
    expDate = new Date();
```

```
    expDate.setSeconds(expDate.getSeconds() + data.maxAge);
  }
  if (expDate != null) cookie += "; expires=" + expDate.toGMTString();

  if (data.domain)
    cookie += "; domain=" + escape(data.domain);

  if (data.path)
    cookie += "; path=" + escape(data.path);

  if (data.secure)
    cookie += "; secure";

  document.cookie = cookie;
  return document.cookie;
}
```

در کد فوق برای انکد کردن رشته‌های مورد استفاده از متد escape استفاده شده است. برای آشنایی با این متد به اینجا مراجعه کنید.

هم‌چنین کار کردن با نوع داده تاریخ در جاوا اسکریپت کمی متفاوت است. بنابراین برای آشنایی بیشتر با این نوع داده به اینجا رجوع کنید.

نکته: در متد بالا بدلیل عدم پشتیبانی از خاصیت max-age در نسخه‌های قدیمی اینترنت اکسپلورر (نسخه 8 و قبل از آن) تنها از خاصیت expires استفاده شده است.

نحوه استفاده از متد بالا به صورت زیر است:

```
setCookie('cookie1', 'Value1');
setCookie({name:'cookie1', value:'Value1'});
setCookie({name:'cookie2', value:'Value2', expDays:10});
setCookie({name:'cookie3', value:'Value3', expires:new Date()});
setCookie({name:'cookie4', value:'Value4', expYear:2013, expMonth:0, expDay:13});
setCookie({name:'cookie3', value:'Value3', maxAge:365*24*60*60});
setCookie({name:'cookie5', value:'Value5', domain:'d.net', path:'/'});
setCookie({name:'cookie6', value:'Value6', secure:true});
setCookie({name:'cookie7', value:'Value7', expDays:100, domain:'dd.com', path:'/employee',
secure:true});
```

حذف کوکی

همانطور که در قسمت قبل هم اشاره شد، برای حذف یک کوکی، کافی است تا تاریخ انقضای آن به تاریخی در گذشته مقداردهی شود. بنابراین برای اینکار می‌توان از متد زیر استفاده کرد:

```
function delCookie(data) {
  if (typeof data === "string") {
    data = { name: data };
  };
  data.expDays = -1;
  return setCookie(data);
}
```

در متد فوق از متد setCookie که در بالا معرفی شد، استفاده شده است. نحوه استفاده از این متد هم به صورت زیر است:

```
delCookie('myCookie');
delCookie({ name: 'myCookie', domain: 'd.com', path: '/test' });
```

خواندن کوکی

برای خواندن مقدار یک کوکی می‌توان از متد زیر استفاده کرد:

```
function getCookie(name) {
  var cookies = document.cookie.split(";");
  for (var i = 0; i < cookies.length; i++) {
    var cookie = cookies[i].split("=");
    if (cookie[0].trim() == escape(name)) {
      return unescape(cookie[1].trim());
    }
```

```
  }
  return null;
}
```

برای آشنایی با متد unescape که در بالا از آن استفاده شده است به اینجا مراجعه کنید. در متد فوق از متد trim زیر استفاده شده است:

```
String.prototype.trim = function () {
  return this.replace(/^\s+|\s+$/g, "");
};
```

```
String.prototype.trimStart = function () {
  return this.replace(/^\s+/, "");
};
String.prototype.trimEnd = function () {
  return this.replace(/\s+$/, "");
};
```

این متدها از اینجا گرفته شده است.

روش استفاده شده برای خواندن مقادیر کوکی‌ها در متد بالا بسیار ساده و ابتدایی است و صرفا برای آشنایی با نحوه Parse کردن رشته برگشت داده شده توسط document.cookie ارائه شده است. روش‌های مناسب‌تر و مطمئن‌تر با یک جستجوی ساده در دسترس هستند. البته همانطور که قبلا هم اشاره شد، استفاه از کتابخانه‌های موجود راه‌حل بهتری است.

هم‌چنین ازآنجاکه مقدار یک کوکی می‌تواند شامل کاراکتر = نیز باشد، بنابراین قسمت return متد فوق را می‌توان به صورت زیر تغییر داد:

```
cookie.shift(1);
return unescape(cookie.join('=').trim());
```

نکته: با توجه به مطالب ارائه شده در قسمت قبل بدست آوردن مقادیر کوکی‌ها کمی پیچیده‌تر از دیگر عملیات‌هاست. ازآنجاکه راه مستقیمی با استفاده از جاوا اسکریپت برای یافتن سایر خواص کوکی وجود ندارد، بنابراین بدست آوردن مقدار دقیق کوکی موردنظر ممکن است غیرممکن باشد! (با توجه به اینکه کوکی‌های متفاوت می‌توانند نام‌های یکسانی داشته باشند).

با توجه به نکته بالا، حال اگر با یک نام بخصوص، چندین کوکی ثبت شده باشد (با خواص متفاوت)، یکی از راه‌حل‌ها این است که آرایه‌ای از مقادیر این کوکی‌های همنام برگشت داده شود. بنابراین متد فوق را می‌توان به صورت زیر تکمیل کرد:

```
function getCookie(name) {
  var foundCookies = [];
  var cookies = document.cookie.split(";");
  for (var i = 0; i < cookies.length; i++) {
    var cookie = cookies[i].split("=");
    if (cookie[0].trim() == escape(name) && cookie.length >= 2) {
      cookie.shift(1);
      foundCookies.push(unescape(cookie.join('=').trim()));
    }
  }
  return foundCookies.length > 1
            ? foundCookies
            : foundCookies.length == 1
                ? foundCookies[0]
                : null;
}
```

خلاصه‌ای از نحوه استفاده از متدهای بالا در IE8 (برای نمایش اجرای درست در مرورگری قدیمی!) در تصویر زیر نشان داده شده است:

نکته: کار توسعه این متدها را میتوان برای پشتیبانی از SubCookieها نیز ادامه داد، اما به دلیل دورشدن از مبحث اصلی، این موضوع در این مطلب ارائه نمیشود (درباره این نوع از کوکی‌ها در قسمت قبل شرح کوتاهی داده شده است). اگر علاقه‌مند و نیازمند به این نوع کوکی‌ها هستید، کتابخانه YUI پشتیبانی کاملی از آنها ارائه میکند.

در قسمت بعدی به نکات کار با کوکی در ASP.NET میپردازیم.

منابع: http://www.w3schools.com/js/js_cookies.asp

http://www.quirksmode.org/js/cookies.html

https://developer.mozilla.org/en-US/docs/DOM/document.cookie http://www.nczonline.net/blog/2009/05/05/http-cookies-explained

عنوان: String.format در جاوا اسکریپت
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۳/۱۹ ۱۱:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, String.format

مقدمه

با اینکه زبان برنامه نویسی جاوا اسکریپت زبانی بسیار قدرتمند و با امکانات زیاد است، اما فقدان برخی متدهای کمکی پرمصرف در آن در برخی موارد باعث دردسرهایی می‌شود. امکانی برای فرمت‌بندی رشته‌ها یکی از این نیازهای نسبتا پرکاربرد است.
متدی که در این مطلب قصد توضیح پیاده‌سازی آنرا داریم، String.format نام دارد که فرایندی مشابه متد متناظر در دات نت را انجام می‌دهد. هم‌چنین سعی شده است تا نحوه پیاده‌سازی این متد کمکی از ابتدایی‌ترین نمونه‌ها تا نسخه‌های پیشرفته‌تر برای درک بهتر مطلب نشان داده شود.

پیاده‌سازی متد String.format

1. در این پیاده‌سازی از اولین فرایندی که ممکن است به ذهن یک برنامه‌نویس خطور کند استفاده شده است. این پیاده‌سازی بسیار ساده به صورت زیر است:

```
String.format = function () {
  var s = arguments[0];
  for (var i = 0; i < arguments.length - 1; i++) {
    s = s.replace("{" + i + "}", arguments[i + 1]);
  }
  return s;
};
```

.

2. پیاده‌سازی مشابهی هم با استفاده از نوع دیگری از حلقه for که تقریبا! مشابه با حلقه foreach در C# است به صورت زیر می‌توان درنظر گرفت:

```
String.format = function () {
  var s = arguments[0];
  for (var arg in arguments) {
    var i = parseInt(arg);
    s = s.replace("{" + i + "}", arguments[i + 1]);
  }
  return s;
};
```

در این متدها ابتدا فرمت واردشده توسط کاربر از لیست آرگومان‌های متد خوانده شده و در متغیر s ذخیره می‌شود. سپس درون یک حلقه به ازای هر توکن موجود در رشته فرمت، یک عملیات replace با مقدار متناظر در لیست آرگومان‌های متد انجام می‌شود. نحوه استفاده از این متد نیز به صورت زیر است:

```
console.log(String.format("{0} is nice!", "donettips.info"));
```

هر دو متد خروجی یکسانی دارند، به صورت زیر:

```
donettips.info is nice!
```

تا اینجا به نظر می‌رسد که عملیات به‌درستی پیش می‌رود. اما اولین و بزرگ‌ترین مشکل در این دو متد نحوه کارکردن متد replace در جاوا اسکریپت است. این متد با این نحوه فراخوانی تنها اولین توکن موجود را یافته و عملیات جایگزینی را برای آن انجام می‌دهد. برای روشن‌تر شدن موضوع به مثال زیر توجه کنید:

```
console.log(String.format("{0} is {1} nice! {0} is {1} nice!", "donettips.info", "very"));
```

با اجرای این مثال نتیجه زیر حاصل می‌شود:

```
donettips.info is very nice! {0} is {1} nice!
```

همان‌طور که می‌بینید عملیات replace برای سایر توکن‌ها انجام نمی‌شود.

.

3. برای حل مشکل فوق می‌توان از روش ساده زیر استفاده کرد:

```
String.format = function () {
  var original = arguments[0],
      replaced;
  for (var i = 0; i < arguments.length - 1; i++) {
    replaced = '';
    while (replaced != original) {
      original = replaced || original;
      replaced = original.replace("{" + i + "}", arguments[i + 1]);
    }
  }
  return replaced;
};
```

در این روش عملیات replace تا زمانی‌که تغییری در رشته جاری ایجاد نشود ادامه می‌یابد. با استفاده از این متد، خروجی مثال قبل درست و به صورت زیر خواهد بود:

```
donettips.info is very nice! donettips.info is very nice!
```

.

4. راه حل دیگر استفاده از امکانات شی RegExp در دستور replace است. نکته مهم استفاده از modifier کلی یا global (که با حرف g مشخص می‌شود) در شی تولیدی از RegExp است (^ و ^ و ^ ، برای جلوگیری از دورشدن از بحث اصلی، جستجو برای کسب اطلاعات بیشتر در این زمینه به خوانندگان واگذار می‌شود). برای استفاده از این شی متد ما به صورت زیر تغییر می‌کند:

```
String.format = function () {
  var s = arguments[0];
  for (var i = 0; i < arguments.length - 1; i++) {
    s = s.replace(new RegExp("\\{" + i + "\\}", "g"), arguments[i + 1]);
  }
  return s;
};
```

استفاده از این متد هم نتیجه درستی برای مثال آخر ارائه می‌دهد.

.

5. روش دیگری که کمی از دو متد قبلی سریع‌تر اجرا می‌شود (به دلیل استفاده از حلقه while) به صورت زیر است:

```
String.format = function () {
  var s = arguments[0],
      i = arguments.length - 1;
  while (i--) {
    s = s.replace(new RegExp('\\{' + i + '\\}', 'g'), arguments[i + 1]);
  }
  return s;
};
```

این متد نیز نتیجه مشابهی ارائه می‌کند. حال به مثال زیر توجه کنید:

```
console.log(String.format("{0}:0 {1}:1 {2}:2", "zero", "{2}", "two"));
```

خروجی صحیح مثال فوق باید به صورت زیر باشد:

```
zero:0 {2}:1 two:2
```

درصورتی‌که رشته‌ای که دو متد از سه متد آخر (3 و 4) به عنوان خروجی ارائه می‌دهند به‌صورت زیر است:

```
zero:0 two:1 two:2
```

برای آخرین متد که ازحلقه while (درواقع با اندیس معکوس) استفاده می‌کند (5) مثالی که خطای مورد بحث را نشان می‌دهد به صورت زیر است:

```
console.log(String.format("{0}:0 {1}:1 {2}:2", "zero", "one", "{1}"));
```

که خروجی اشتباه زیر را برمی‌گرداند:

```
zero:0 one:1 one:2
```

درصورتی‌که باید مقدار زیر را برگشت دهد:

```
zero:0 one:1 {1}:2
```

دلیل رخدادن این خطا اجرای عملیات replace به صورت جداگانه و کامل برای هر توکن، از اول تا آخر برای رشته‌های replace شده جاری است که کار را خراب می‌کند.

.

6. برای حل مشکل بالا نیز می‌توان از یکی دیگر از امکانات دستور replace استفاده کرد که به صورت زیر است:

```
String.format = function () {
  var args = arguments;
  return args[0].replace(/{(\d+)}/g, function (match, number) { return args[parseInt(number) + 1]; });
};
```

در اینجا از قابلیت سفارشی‌سازی عملیات جایگزینی در دستور replace استفاده شده است. با استفاده از این ویژگی عملیات replace برای هر توکن جداگانه انجام می‌شود و بنابراین تغییرات اعمالی در حین عملیات تاثیر مستقیمی برای ادامه روند نخواهد گذاشت.

دقت کنید که برای بکاربردن RegExp درون دستور replace به جای تولید یک نمونه از شی RegExp می‌توان عبارت مربوطه را نیز مستقیما بکار برد. در اینجا از عبارتی کلی برای دریافت تمامی توکن‌های با فرمتی به صورت {عدد} استفاده شده است.

متد سفارشی مربوطه نیز شماره ردیف توکن یافته‌شده به همراه خود عبارت یافته‌شده را به عنوان آرگومان ورودی دریافت کرده و مقدار متناظر را از لیست آرگومان‌های متد اصلی پس از تبدیل شماره ردیف توکن به یک عدد، برگشت می‌دهد (در اینجا نیز برای جلوگیری از دورشدن از بحث اصلی، جستجو برای کسب اطلاعات بیشتر در این زمینه به خوانندگان واگذار می‌شود).

برای جلوگیری از تداخل بین آرگومان‌های متد اصلی و متد تهیه‌شده برای سفارشی‌سازی عملیات جایگزینی، در ابتدای متد اصلی، لیست آرگومان‌های آن درون متغیر جداگانه‌ای (args) ذخیره شده است.

با استفاده از این متد خروجی درست نشان داده می‌شود. حال مثال زیر را درنظر بگیرید:

```
console.log(String.format("{0} is {1} nice!", "donettips.info"));
```

خروجی این مثال به‌صورت زیر است:

```
donettips.info is undefined nice!
```

پیاده‌سازی زیر برای حل این مشکل استفاده می‌شود.

.

7. برای کنترل بیشتر و رفع خطاهای احتمالی در متد بالا، می‌توان ابتدا از وجود آرگومان مربوطه در متغیر args اطمینان حاصل کرد تا از جایگزینی مقدار undefined در رشته نهایی جلوگیری کرد. مانند نمونه زیر:

```
String.format = function () {
  var s = arguments[0],
      args = arguments;
  return s.replace(/{(\d+)}/g, function (match, number) {
    var i = parseInt(number);
    return typeof args[i + 1] != 'undefined' ? args[i + 1] : match;
  });
};
```

با استفاده از این متد جدید خروجی مثال‌های قبل درست خواهد بود.
در فرمت بندی رشته‌ها برای نمایش خود کاراکتر { یا } از تکرار آن‌ها (یعنی {{ یا }}) استفاده می‌شود. اما متد ما تا این لحظه این امکان را ندارد. برای مثال:

```
console.log(String.format("{0}:0 {1}:1 {2}:2, {{0}} {{{1}}}  {{{{2}}}}   {2}", "zero", "{2}", "two"));
```

که خروجی زیر را ارائه می‌دهد:

```
zero:0 {2}:1 two:2, {zero} {{{2}}}  {{{two}}}   two
```

8. برای پیاده‌سازی امکان اشاره‌شده در بالا می‌توان از کد زیر استفاده کرد:

```
String.format = function () {
  var s = arguments[0],
      args = arguments;
  return s.replace(/\{\{|\}\}|\{(\d+)\}/g, function (match, number) {
    if (match == "{{") { return "{"; }
    if (match == "}}") { return "}"; }
    var i = parseInt(number);
    return typeof args[i + 1] != 'undefined'
                              ? args[i + 1]
                              : match;
  });
};
```

در اینجا با استفاده از یک عبارت RegExp پیچیده‌تر و کنترل تکرار کاراکترهای { و } در متد سفارشی جایگزینی در دستور replace، پیاده‌سازی اولیه این ویژگی ارائه شده است.
این متد خروجی صحیح زیر را برای مثال آخر ارائه می‌دهد:

```
zero:0 {2}:1 two:2, {0} {{2}}  {{2}}   two
```

.

پیاده‌سازی به‌صورت یک خاصیت prototype

تمامی متدهای نشان داده‌شده تا اینجا به‌صورت مستقیم از طریق String.format در دسترس خواهند بود (تعریفی شبیه به

متدهای استاتیک در دات نت). درصورتی‌که بخواهیم از این متدها به صورت یک خاصیت prototype شی string استفاده کنیم (چیزی شبیه به متدهای instance در اشیای دات نت) می‌توانیم از تعریف زیر استفاده کنیم:

```
String.prototype.format = function () {
   ...
}
```

تنها فرق مهم این پیاده‌سازی این است که رشته مربوط به فرمت وارده در این متد از طریق شی this در دسترس است و بنابراین شماره اندیس آرگومان‌های متد یکی کمتر از متدهای قبلی است که باید مدنظر قرار گیرد. مثلا برای متد آخر خواهیم داشت:

```
String.prototype.format = function () {
  var s = this.toString(),
      args = arguments;
  return s.replace(/\{\{|\}\}|\{(\d+)\}/g, function (match, number) {
    if (match == "{{") { return "{"; }
    if (match == "}}") { return "}"; }
    return typeof args[number] != 'undefined'
                              ? args[number]
                              : match;
  });
};
```

.

نکته: در تمامی خواص prototype هر شی در جاوا اسکریپت، متغیر this از نوع object است. بنابراین برای جلوگیری از وقوع هر خطا بهتر است ابتدا آن‌را به نوع مناسب تبدیل کرد. مثل استفاده از متد toString در متد فوق که موجب تبدیل آن به رشته می‌شود.

.

ازآنجاکه نیاز به تغییر اندیس در متد سفارشی عملیات replace وجود ندارد، بنابراین خط مربوط به تبدیل آرگومان number به یک مقدار عددی (با دستور parseInt) حذف شده است و از این متغیر به صورت مستقیم استفاده شده است. در این حالت عملیات تبدیل توسط خود جاوا اسکریپت مدیریت می‌شود که کار را راحت‌تر می‌سازد. بنابراین متد ما به صورت زیر قابل استفاده است:

```
console.log("{0}:0 {1}:1 {2}:2, {{0}} {{{1}}}  {{{{2}}}}   {2}".format("zero", "{2}", "two"));
```

.

پیاده‌سازی با استفاده از توکن‌های غیرعددی

برای استفاده از توکن‌های غیرعددی می‌توانیم به صورت زیر عمل کنیم:

```
String.format = function () {
  var s = arguments[0],
      args = arguments[1];
  for (var arg in args) {
    s = s.replace(new RegExp("{" + arg + "}", "g"), args[arg]);
  }
  return s;
};
```

برای حالت prototype نیز داریم:

```
String.prototype.format = function () {
  var s = this.toString(),
      args = arguments[0];
  for (var arg in args) {
    s = s.replace(new RegExp("{" + arg + "}", "g"), args[arg]);
  }
```

```
  return s;
};
```

با استفاده از این دو متد داریم:

```
console.log(String.format("{site} is {adj}! {site} is {adj}!", { site: "donettips.info", adj: "nice"
}));
console.log("{site} is {adj}! {site} is {adj}!".format({ site: "donettips.info", adj: "nice" }));
```

تا اینجا متدهایی نسبتا کامل برای نیازهای عادی برنامه‌نویسی تهیه شده است. البته کار توسعه این متد برای پشتیبانی از امکانات پیشرفته‌تر فرمت‌بندی رشته‌ها می‌تواند ادامه پیدا کند.

.

کتابخانه‌های موجود

یکی از کامل‌ترین کتابخانه‌های کار با رشته‌ها همان کتابخانه معروف Microsoft Ajax Client Libray است که بیشتر امکانات موجود کار با رشته‌ها در دات نت را در خود دارد. صرفا جهت آشنایی، پیاده‌سازی متد String.format در این کتابخانه در زیر آورده شده است:

```
String.format = function String$format(format, args) {
  /// <summary locid="M:J#String.format" />
  /// <param name="format" type="String"></param>
  /// <param name="args" parameterArray="true" mayBeNull="true"></param>
  /// <returns type="String"></returns>
//  var e = Function._validateParams(arguments, [
//    { name: "format", type: String },
//    { name: "args", mayBeNull: true, parameterArray: true }
//  ]);
//  if (e) throw e;
  return String._toFormattedString(false, arguments);
};
String._toFormattedString = function String$_toFormattedString(useLocale, args) {
  var result = '';
  var format = args[0];
  for (var i = 0; ; ) {
    var open = format.indexOf('{', i);
    var close = format.indexOf('}', i);
    if ((open < 0) && (close < 0)) {
      result += format.slice(i);
      break;
    }
    if ((close > 0) && ((close < open) || (open < 0))) {
      if (format.charAt(close + 1) !== '}') {
        throw Error.argument('format', Sys.Res.stringFormatBraceMismatch);
      }
      result += format.slice(i, close + 1);
      i = close + 2;
      continue;
    }
    result += format.slice(i, open);
    i = open + 1;
    if (format.charAt(i) === '{') {
      result += '{';
      i++;
      continue;
    }
    if (close < 0) throw Error.argument('format', Sys.Res.stringFormatBraceMismatch);
    var brace = format.substring(i, close);
    var colonIndex = brace.indexOf(':');
    var argNumber = parseInt((colonIndex < 0) ? brace : brace.substring(0, colonIndex), 10) + 1;
    if (isNaN(argNumber)) throw Error.argument('format', Sys.Res.stringFormatInvalid);
    var argFormat = (colonIndex < 0) ? '' : brace.substring(colonIndex + 1);
    var arg = args[argNumber];
    if (typeof (arg) === "undefined" || arg === null) {
      arg = '';
    }
    if (arg.toFormattedString) {
      result += arg.toFormattedString(argFormat);
    }
```

```
    else if (useLocale && arg.localeFormat) {
      result += arg.localeFormat(argFormat);
    }
    else if (arg.format) {
      result += arg.format(argFormat);
    }
    else
      result += arg.toString();
    i = close + 1;
  }
  return result;
}
```

دقت کنید قسمت ابتدایی این متد که برای بررسی اعتبار آرگومان‌های ورودی است، برای سادگی عملیات کامنت شده است. همان‌طور که می‌بینید این متد پیاده‌سازی نسبتا مفصلی دارد و امکانات بیشتری نیز در اختیار برنامه نویسان قرار می‌دهد. البته سایر متدهای مربوطه بدلیل طولانی بودن در اینجا آورده نشده است. برای مثال امکانات پیشرفته‌تری مثل زیر با استفاده از این کتابخانه در دسترس هستند:

```
console.log(String.format("{0:n}, {0:c}, {0:p}, {0:d}", 100.0001));
// result:   100.00, ¤100.00, 10,000.01 %, 100.0001

console.log(String.format("{0:d}, {0:t}", new Date(2015, 1, 1, 10, 45)));
// result:   02/01/2015, 10:45
```

آخرین نسخه این کتابخانه از اینجا قابل دریافت است (این متدها درون فایل MicrosoftAjax.debug.js قرار دارند). این کتابخانه دیگر به این صورت و با این نام توسعه داده نمیشود و چند سالی است که تصمیم به توسعه ویژگی‌های جدید آن به صورت پلاگین‌های jQuery گرفته شده است.

.

کتابخانه دیگری که می‌توان برای عملیات فرمت‌بندی رشته‌ها در جاوا اسکریپت از آن استفاده کرد، کتابخانه معروف jQuery Validation است. این کتابخانه یک متد نسبتا خوب با نام format برای فرمت کردن رشته‌ها دارد. نحوه استفاده از این متد به صورت زیر است:

```
var template = jQuery.validator.format("{0} is not a valid value");
console.log(template("abc"));
// result: 'abc is not a valid value'
```

.

کتابخانه نسبتا کامل دیگری که وجود دارد، با عنوان Stringformat از اینجا قابل دریافت است. برای استفاده از این کتابخانه باید به صورت زیر عمل کرد:

```
String.format([full format string], [arguments...]);
// or:
[date|number].format([partial format string]);
```

همان‌طور که می‌بینید این کتابخانه امکانات کامل‌تری نیز دارد. مثال‌های مربوط به این کتابخانه به صورت زیر هستند که توانایی‌های نسبتا کامل آن‌را نشان می‌دهد:

```
// Object path
String.format("Welcome back, {username}!",
{ id: 3, username: "JohnDoe" });
// Result: "Welcome back, JohnDoe!"

// Date/time formatting
String.format("The time is now {0:t}.",
new Date(2009, 5, 1, 13, 22));
// Result: "The time is now 01:22 PM."
```

```
// Date/time formatting (without using a full format string)
var d = new Date();
d.format("hh:mm:ss tt");
// Result: "02:28:06 PM"

// Custom number format string
String.format("Please call me at {0:+##0 (0) 000-00 00}.", 4601111111);
// Result: "Please call me at +46 (0) 111-11 11."

// Another custom number format string
String.format("The last year result was {0:+$#,0.00;-$#,0.00;0}.", -5543.346);
// Result: "The last year result was -$5,543.35."

// Alignment
String.format("|{0,10:PI=0.00}|", Math.PI);
// Result: "|   PI=3.14|"

// Rounding
String.format("1/3 ~ {0:0.00}", 1/3);
// Result: "1/3 ~ 0.33"

// Boolean values
String.format("{0:true;;false}", 0);
// Result: "false"

// Explicitly specified localization
// (note that you have to include the .js file for used cultures)
msf.setCulture("en-US");
String.format("{0:#,0.0}", 3641.667);
// Result: "3,641.7"

msf.setCulture("sv-SE");
String.format("{0:#,0.0}", 3641.667);
// Result: "3 641,7"
```

.

یک کتابخانه دیگر نیز از این آدرس قابل دریافت است. این کتابخانه با عنوان String.format نام‌گذاری شده است. نحوه استفاده از این کتابخانه نیز به صورت زیر است:

```
//inline arguments
String.format("some string with {0} and {1} injected using argument {{number}}", 'first value', 'second value');
//returns: 'some string with first value and second value injected argument {number}'

//single array
String.format("some string with {0} and {1} injected using array {{number}}", [ 'first value', 'second value' ]);
//returns: 'some string with first value and second value injected using array {number}'

//single object
String.format("some string with {first} and {second} value injected using {{propertyName}}",{first:'first value',second:'second value'});
//returns: 'some string with first value and second value injected using {propertyName}'
```

.

کتابخانه نسبتا معروف و کامل sprintf نیز در اینجا وجود دارد. این کتابخانه امکانات بسیاری همچون متدهای متناظر در زبان C دارد.

.

منابع http://www.w3schools.com/jsref/jsref_replace.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/replace

http://www.w3schools.com/jsref/jsref_obj_regexp.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp

http://stackoverflow.com/questions/610406/javascript-equivalent-to-printf-string-format

http://stackoverflow.com/questions/1038746/equivalent-of-string-format-in-jquery
http://stackoverflow.com/questions/2534803/string-format-in-javascript
http://jqueryvalidation.org/jQuery.validator.format
http://www.masterdata.se/r/string_format_for_javascript
https://github.com/tracker1/core-js/blob/master/js-extensions/100-String.format.js
http://www.diveintojavascript.com/projects/javascript-sprintf

نظرات خوانندگان

نویسنده: میثم هوشمند
تاریخ: ۱۳۹۲/۰۳/۲۰ ۰:۲۸

با تشکر از شما بابت ارسال این پست خیلی خیلی خوب!
از چند جهت عالی بود
اول معرفی چند کتابخانه
دوم ارائه‌ی بخشی از کد کتابخانه و توضیح پیاده سازی آن !
سوم آموزش اضافه کردن توابع جدید به صورت استاتیک و همچنین به قول خودتان instance method
عالی بود! ممنونم

عنوان: PersianDatePicker یک DatePicker شمسی به زبان JavaScript که از تاریخ سرور استفاده می‌کند
نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۳/۲۶ ۱۲:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Persian, تاریخ شمسی, DatePicker

PersianDatePicker یک DatePicker شمسی برای نمایش تاریخ شمسی در صفحات وب است.

مهمترین ویژگیها :
از تاریخ سرور برای نمایش روز جاری استفاده می‌کند.
به زبان جاوااسکریپت است و نیازی به سایر فریمورک‌ها مانند jQuery ندارد.
سبک و کم حجم است (3.5 کیلوبایت)
روزها به صورت عمودی لیست شده‌اند مانند برنامه هفتگی مدرسه که کاربران فارسی‌زبان با آن راحت‌ترند.

طرز استفاده :

```
PersianDatePicker.Show(textBox, today)
```

textBox : کنترلی است که DatePicker برای آن نمایش داده می‌شود.
today : تاریخ روز جاری است.

برای مثال :

```
<html>
<head>
    <script type="text/javascript" src="Scripts/PersianDatePicker.min.js"></script>
    <link type="text/css" rel="stylesheet" href="Content/PersianDatePicker.min.css" />
</head>
<body>
    <input type="text" onclick="PersianDatePicker.Show(this, '1392/03/22');" />
</body>
</html>
```

در ASP.NET Web Forms :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication4.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script src="Scripts/PersianDatePicker.min.js"></script>
    <link href="Content/PersianDatePicker.min.css" rel="stylesheet" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label runat="server" Text="Label" AssociatedControlID="TextBox1">تاریخ : </asp:Label>
            <asp:TextBox  runat="server" ID="TextBox1"></asp:TextBox>
        </div>
    </form>
</body>
</html>
```

```
public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        var now = PersianDateTime.Now;
        var today = now.ToString(PersianDateTimeFormat.Date);

        TextBox1.Attributes["onclick"] = "PersianDatePicker.Show(this,'" + today + "');";
    }
}
```

در مثال بالا برای گرفتن تاریخ روز جاری از PersianDateTime استفاده شده است.

در ASP.NET MVC یا PHP یا هر زبان دیگری کافیست onclick کنترل متنی مورد نظر در سمت سرور مانند مثال بالا مقدار داده شود. یا می‌توان image کوچکی کنار textbox مورد نظر قرار داد و onclick آن تنظیم شود.

برای استفاده از PersianDatePicker می‌توانید آنرا از NuGet دریافت کنید :

```
PM> Install-Package PersianDatePicker
```

همچنین یک مثال کامل ASP.NET MVC آن‌را می‌توانید از اینجا دریافت کنید:

MvcAppPersianDatePicker.zip

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۶ ۱۲:۱۴

بسیار عالی. مزیت کار شما به تقویم‌های نمونه دیگر ، سورس باز بودن آن است (نمونه غیرفشرده هم لحاظ شده). ضمنا یکی از مواردی که می‌تونه در تهیه ایده برای تولید یک محصول بهتر و پخته‌تر کمک کنه مراجعه به مشکلات و نظرات کتابخانه‌های مشابه است. مثلا همان تقویم لینک داده شده، قسمت نظراتش پر است از ایده برای کسانی که بخواهند یک کار خیلی پخته رو در این زمینه تکمیل کنند؛ بر اساس نیازهای واقعی استفاده کننده‌ها.

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۳/۲۶ ۱۲:۳۳

بله درست است و امیدوارم در آینده به کمک دیگران توسعه داده شود.

نویسنده: مرادیان
تاریخ: ۱۳۹۲/۰۳/۲۶ ۱۳:۵۰

سلام؛ میخوام بدونم چجوری یه رویداد مثل onclick یه شیء مثل textbox رو تو mvc مدیریت می‌کنن مثلا نمایش همین datepicker (ناگفته نمونه تا بخش هشتم از آموزش mvc شما رو مطالعه کردم)

بسیار ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۶ ۱۳:۵۶

با مقدار دهی HTML attributes اینکار امکان پذیر است:

```
@Html.TextBox("DateCreated", "", new { onclick="alert('yes')"} )
```

نویسنده: ح.م
تاریخ: ۱۳۹۲/۰۳/۲۶ ۱۹:۱۵

کاربرد پارامتر dayOfWeek چیست؟ مثلاً در مثال فوق که 22/03/1392 برابر روز چهارشنبه است، می‌توان dayOfWeek را برابر 2 ارسال کرد یعنی این روز دوشنبه است؟

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۳/۲۶ ۱۹:۵۴

- شماره روز جاری در هفته است . شماره سایر روزها در هفته از روی آن تعیین می‌شود.
- بله اگر آنرا برابر 2 قرار دهید یعنی روز جاری دوشنبه است.

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۳/۲۶ ۲۱:۶

بروزرسانی شد (نسخه 1.1) : پارامتر todayWeekDay حذف شد.

نویسنده: مهدی یوسفی
تاریخ: ۱۳۹۲/۰۳/۳۱ ۱۰:۵۶

یک تقویم دیگر
بر پایه دات نت بصورت کنترل در دات نت
https://zebradatepickerdotnet.codeplex.com/
سبک، ساده، شبیه تقویم ویندوز 7 برای انتخاب سال و ماه و روز

با کتابخانه فارسی تبدیل تاریخ و ....
بر پایه
http://stefangabos.ro/jquery/zebra-datepicker/#download

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۲/۰۴/۰۴ ۰:۲۸

با سلام و تشکر
برخی از امکاناتی که تقویم‌های مشابه دارند که اگر در این تقویم هم پیش بینی شود بهتر است.
تقویم با jQuery به فرموده آقای یوسفی : محدود کردن انتخاب تاریخ
یا تقویم ترجمه آقای هاشمی نژاد : انتخاب تاریخ‌های محدود بین دو تاریخ

بنده میخواستم نظرم را به عنوان بازخورد ثبت کنم ولی فکر کردم اینجا ثبت کنم بهتره
اینکه datepicker رو به datetimepicker تبدیل کنیم
بتوانیم یک نمونه یا راهنمای توسعه برای همکاران جهت ادامه راه پرارزش شما ارائه دهیم.
محصول را با برخی از ابزارهای معروف مانند KENDO UI یا Telerik که مصرف کنندگان بسیار زیادی دارند مقایسه کنیم و امکاناتشان را در کنار هم تشریح کنیم

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۴/۰۵ ۱:۳۸

- در نسخه بعدی تغییراتی مثل **محدود کردن انتخاب تاریخ** و **انتخاب ساده‌تر سال و ماه** در نظر گرفته شده است.

نویسنده: پهلوانی
تاریخ: ۱۳۹۲/۰۴/۱۳ ۲۰:۲۷

با سلام و تشکر

از این DataPicker استفاده کردم ولی:

1- به چه صورت از یک دکمه برای نمایش تقویم و نمایش تاریخ انتخابی در TexBox استفاده کنیم

2-در حالت عادی دکمه‌های برگشت ماه و سال درست نمایش داده نمی‌شوند و در حالت Compatibility View دکمه‌ها درست دیده می‌شوند

PersianDatePicker یک DatePicker شمسی به زبان JavaScript که از تاریخ سرور استفاده می‌کند

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۴/۱۴ ۱۳:۵۶

شما حتی در ویندوز XP هم مجبور نیستی از IE 8 استفاده کنی. فایرفاکس و کروم روی XP راحت نصب می‌شن

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۴/۱۵ ۲۳:۵۲

- کافیست در onclick دکمه کنار textbox، تابع PersianDatePicker.Show را صدا بزنید. البته پارامتر اول آن باید به textbox اشاره داشته باشد.
- نرم‌افزار فشرده ساز ، > را حذف کرده و به جای آن > قرار داده که در بعضی نسخه‌های IE به خاطر همسانی با کاراکتر انتهای تگ، نمایش داده نمی‌شود. در نسخه بعدی این مورد اصلاح خواهد شد (فعلا گرفتارم).

نویسنده: سیروان
تاریخ: ۱۳۹۲/۰۵/۰۲ ۱۱:۴۵

مرسی از تقویمتون ولی اگه بخوام مقدار میلادی روهم داشته باشم چیکار کنم
با تشکر

نویسنده: atrisa
تاریخ: ۱۳۹۲/۰۵/۰۵ ۱۹:۵۸

برای استفاده از این برای مقدار دهی به یک پروپرتی از یک مدل تو mvc تایپ اون فیلد رو چی باید در نظر بگیرم؟

```
public DateTime deliveryDate { get; set; }
```

الان وقتی با این دستور اسکریپت رو فراخوانی می‌کنم فقط قالب datepicker رو می‌بینم روزا رو نمی‌بینم. اگر میشه راهنماییم کنید. ممنون

```
@Html.TextBoxFor(model => model.deliveryDate,
 new { @id = "TextBox1",onclick =
"PersianDatePicker.Show(this,'PersianDateTime.Now.now.ToString(PersianDateTimeFormat.Date)')" })
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۵ ۲۱:۱۸

- این کنترل (و مشابهات آن) یک رشته تاریخ شمسی را بر می‌گردانند. اگر نیاز است مقدار آن به DateTime میلادی انتساب داده شود، باید از یک model binder سفارشی برای اینکار در ASP.NET MVC استفاده کنید. یک نمونه پیاده سازی شده آن‌را در بحث PersianDateModelBinder می‌توانید مشاهده و استفاده کنید.
- در کدی که نوشتید یک now اضافه است. همان Now اول کافی است.

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۵/۰۶ ۸:۵۹

در سمت سرور، کافیست تاریخ شمسی را تبدیل کنید .

نویسنده: مجید
تاریخ: ۱۳۹۲/۰۵/۱۳ ۱۰:۵۰

باسلام
ضمن تشکر در ie8 هنگام اجرا بر روی s.onchange در PersianDatePicker.min.js پیام خطا
Microsoft JScript runtime error: Object doesn't support this property or method
میدهد.

نویسنده: morteza
تاریخ: ۱۳۹۲/۰۵/۱۶ ۱۸:۵۸

سلام
چرا موقع نمایش برای من به جای نام ماه undefined را نمایش می‌دهد و با زدن کلید چپ و راست درست میشه ؟

نویسنده: حمید نجفی
تاریخ: ۱۳۹۲/۰۶/۲۱ ۱۴:۳۲

سلام.
اگر تاریخ انتخابی از تقویم 31 اردیبهشت، 31 تیر و یا 31 شهریور باشد، در سمت کنترلر مقدار null بر میگرداند. ممنون از زحمات شما.

نویسنده: مروراید خطیری
تاریخ: ۱۳۹۲/۰۶/۲۵ ۱۰:۷

سلام
برای من هم در قسمت نمایش ماه‌ها undefined نشون می‌ده

نویسنده: مجید_فاضلی نسب
تاریخ: ۱۳۹۲/۰۸/۰۳ ۱۹:۳

سلام و خسته نباشین
فقط فرمت PersianDateTimeFormat.Date متد ToString() کار میکنه و بقیه فرمت‌ها کار نمی‌کنن.

نویسنده: مجید
تاریخ: ۱۳۹۲/۰۹/۱۴ ۲۳:۳۸

با سلام
در هنگام نصب پکیج، چه با استفاده از کنسول و چه با استفاده از واسطِ گرافیکی، خطای زیر رخ می‌دهد:

```
Id is required
version is required
Author is required
description is required
```

نویسنده: هادی دانش پژوه
تاریخ: ۱۳۹۳/۰۲/۲۷ ۱۸:۴۴

با سلام
کنترل زیبایی درست کردیند، ممنون
ولی چرا وقتی روی تاریخی کلیک میکنم Object expected میده

```
o=i.style;o.position="absolute";o.zIndex=1e7;i.onmousedown=function()
{u=true};i.onclick=function(){s.focus()}}function f()
{o.visibility="hidden"}function l(e){s.value=e;s.focus();f();s.onchange()}
Object expected
function c(e,t){return e.substring(0,8)+(t<10?"0":"")+t}function h(e,t)
{var n=document.createElement(e);t.appendChild(n);return n}function p(e)
{s.focus();var o=w(e.substring(0,8)+"01");S(i,"");var u=h("table",i);x(u,
"datePicker");u.cellSpacing=0;var a=u.insertRow(0);x(a,"datePickerHeader");
```

و برای تغییر ماه و سال هم چند بار باید کلیک کرد.
البته این مشکل فقط در IE وجود داره ، در نسخه 8 و 11 تست کردم

نویسنده: هادی دانش پژوه
تاریخ: ۱۳۹۳/۰۲/۲۷ ۱۹:۱۷

مثل اینکه خودم تونستم درستش کنم! :)
فقط تو قسمتی که خطا میده try catch گذاشتم! الان بدرستی کار میکنه،
اینم فایل اصلاح شده: PersianDatePicker.min.js

نویسنده: سارامحمدی
تاریخ: ۱۳۹۳/۰۳/۰۹ ۱۶:۵۰

با سلام؛ من از persiandatapicker در mvc به شکل زیراستفاده می‌کنم:

```
"ReleaseDate" "" new "PersianDatePicker.Show();" } )
```

در قالب ویو هم فایل‌های جی کوئری را اضافه کردم اما بازهم موقع کلیک در تکست باکس خطا میدهد که persiandatapicker وجود ندارد. اگه میشه استفاده persiandatapicker را در mvc با مثال توضیح دهید. ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۰۹ ۱۸:۳۶

با فرض اینکه layout برنامه تعاریف لازم را دارد:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="~/Content/PersianDatePicker.css" rel="stylesheet" />
</head>
<body>
    @RenderBody()
    <script src="~/Scripts/jquery-1.10.2.min.js"></script>
    <script src="~/Scripts/PersianDatePicker.js"></script>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

فایلی را به نام PersianDatePicker.cshtml در مسیر Views\Shared\EditorTemplates ایجاد کنید؛ با این محتوا:

```
@using System.Globalization
@model DateTime?
@{
    Func<DateTime, string> toPersianDate = date =>
    {
        var dateTime = new DateTime(date.Year, date.Month, date.Day, new GregorianCalendar());
        var persianCalendar = new PersianCalendar();
        return persianCalendar.GetYear(dateTime) + "/" +
               persianCalendar.GetMonth(dateTime).ToString("00") + "/" +
               persianCalendar.GetDayOfMonth(dateTime).ToString("00");
    };
    var today = toPersianDate(DateTime.Now);
    var name = this.ViewContext.ViewData.ModelMetadata.PropertyName;
    var value = Model.HasValue ? toPersianDate(Model.Value) : string.Empty;
}
<input type="text" dir="ltr"
       name="@name" id="@name"
       value="@value"
       onclick="PersianDatePicker.Show(this,'@today');" />
```

بعد از این تنها کاری که باید انجام شود، استفاده از ویژگی UIHint است:

```
// Location: Views\Shared\EditorTemplates\PersianDatePicker.cshtml
[UIHint("PersianDatePicker")]
public DateTime AddDate { set; get; }
```

به صورت خودکار به متد EditorFor اعمال می‌شود:

```
<div>
    <label>تاریخ:</label>
    @Html.EditorFor(model => model.AddDate)
</div>
```

همچنین اگر نیاز است مقدار رشته تاریخ شمسی آن به یک DateTime میلادی انتساب داده شود (در حین ارسال اطلاعات به سرور)، باید از یک model binder سفارشی برای اینکار در ASP.NET MVC استفاده کنید. یک نمونه پیاده سازی شده آن‌را در بحث

PersianDateModelBinder می‌توانید مشاهده و استفاده کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۰۹ ۱۸:۳۸

متد setValue است که نیاز است اصلاح شود:

```
function setValue(date) {
        _textBox.value = date;
        _textBox.focus();
        hide();
        try {
            _textBox.onchange();
        }catch(ex) {}
    }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۰۹ ۱۸:۴۰

علت اینجا است که ماه را **دو** رقمی محاسبه کرده‌است.
اگر از ماه یک رقمی استفاده کنید، دراپ داون ماه‌های آن undefined نمایش داده می‌شود.

نویسنده: سارا محمدی
تاریخ: ۱۳۹۳/۰۳/۱۷ ۲۲:۳۹

باسلام وتشکر

من دقیقا همین کارها را انجام دادم اما موقع ذخیره تاریخ خطای زیر را میدهد

.The conversion of a datetime2 data type to a datetime data type resulted in an out-of-range value
The statement has been terminated

مثل اینکه فرمت تاریخ و ساعتی که خروجی persiandatapicker است را نمی‌تواند ذخیره کند

ممنون از توجهتون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۱۷ ۲۲:۵۳

با فرض فعال سازی و ثبت PersianDateModelBinder ، این خطا زمانی حاصل می‌شود که یک فیلد datetime مقدار دهی نشده را بخواهید در بانک اطلاعاتی ذخیره کنید. نوع datetime در دات نت value type است و مقدار پیش فرض آن 01-01-0001 است (DateTime.MinValue) که قابل ذخیره سازی در بانک اطلاعاتی نیست. یا فیلد را nullable تعریف کنید (هم در سمت کدها و هم در سمت بانک اطلاعاتی) و یا حتما هنگام ذخیره سازی اطلاعات، آن‌را مقدار دهی کنید تا مقدار پیش فرض خود را نداشته باشد.

نویسنده: یه برنامه نویس
تاریخ: ۱۳۹۳/۰۴/۱۲ ۱۲:۴۹

ایول.از وقتی کامپوننت شما رو دیدم datepicker تلریک و jquery از چشمم افتاده. به سادگی کار کردن با یه input میشه باهاش زندگی کرد.هم سمت کلاینت هم سمت سرور میشه راحت بهش دسترسی پیدا کرد و مقدار اولیه داد. تازه راه اندازی اش هم آسونه و عین jquery نیست که یه فایل به صفحه اضافه کنی مجبور شی همه صفحه رو بالا پایین کنی.
در کل خدا اجرتون بده.

نویسنده: نقوی
تاریخ: ۱۳۹۳/۰۹/۱۱ ۱۵:۲

با تشکر از کنترل خوبتون
من از این کنترل توی پروژه‌ی ام وی سی خودم استفاده کردم این کنترل توی مرورگرهای کروم و IE به درستی کار میکنه ولی توی **فایرفاکس دکمه هاش کار نمیکنن** و وقتی دکمه ایی فشار داده میشه کنترل بسته میشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۱ ۱۷:۵۶

- مشکلی با فایرفاکس مشاهده نشد.
- نکته‌ی UIHint ایی که کمی بالاتر توضیح داده شد ، مثال کاملش را از اینجا دریافت کنید: MvcAppPersianDatePicker.zip

نویسنده: نقوی
تاریخ: ۱۳۹۳/۰۹/۱۵ ۱۹:۵۰

بسیار ممنون از مثالی که گذاشتین. حق با شماست با فایر فاکس مشکلی نبود.
مشکل من حالت خاصیه که ازش استفاده کردم. من تو دیالوگ modal جی کوئری یو آی از این کنترل استفاده کردم و تو این حالت خاص این مشکل رو داره احتمالا جاوااسکریپتش یه جا کانفلیکت میده.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۹/۱۵ ۲۳:۴۴

اگر **z-index** یک popup بیشتر از المانی باشه، اون popup هست که لایه رویی قابل کلیک رو تشکیل می‌ده.

عنوان: آموزش فایرباگ - #8 - Script Panel
نویسنده: احمد احمدی
تاریخ: ۱۳۹۲/۰۳/۳۰ ۲۲:۰
آدرس: www.dotnettips.info
برچسب‌ها: Debugging, JavaScript, FireBug

تب Script در FireBug مخصوص دیباگ کردن کدهای JavaScript است. امکاناتی که در این قسمت گنجانده شده بسیار کاربردی بوده و همچنین در بیشتر قسمت‌ها با ابزارهای خطایابی دیگر مشابه است. برای مثال اگر با Visual Studio کار کرده باشید، با نحوه‌ی ایجاد Break Point ، قسمت‌های Watch,Stack و ... آشنا خواهید بود.

این پنل هم مشابه پنل‌های دیگر فایرباگ دارای یک بخش با عنوان Options Menu هست که با راست کلیک کردن بروی عناون یا کلیک بروی مثلث کنار عنوان پنل قابل دسترسی است. تنظیماتی که در اینجا قابل تعیین است عبارتند از:

Enabled/Disabled : برای فعال/غیرفعال کردن پنل است. فعال بودن این قسمت ممکن است موجب کاهش سرعت بارگزاری صفحات شود.
Track Throw/Catch : در صورت فعال بودن این گزینه، در صورت رویدادن خطا در بلاک try/catch ، دیباگر در آن نقطه از برنامه متوقف شده و کنترل برنامه به شما داده می‌شود. (البته بنده موفق بررسی کامل این قابلیت نشدم. ظاهرا ورژن‌های آخری باگ دارند. مثل غیرفعال شدن کامل همین پنل، "Debugger not activated"! اگر کسی موفقیتی در کار با این مورد داشت، سپاسگذار میشوم اطلاع بدهد.)
Show Break Notifications : در صورت فعال بودن این گزینه، هنگام توقف کدی در صفحه، اطلاعاتی در مورد علت توقف آن در بالای پنل ارائه خواهد شد.

توجه داشته باشید که ممکن است در ورژن‌های مختلف، تعداد این گزینه‌ها بیشتر یا کمتر باشد.

Panel Toolbar

نواری است که ابزارهای پنل بروی آن قرار دارند و به ترتیب عبارتند از:
Break On Next : این دکمه که مشابه آن در اکثر پنل‌ها وجود دارد، هنگام اجرای یک دستور JavaScript آن را متوقف کرده و شما می‌توانید به بررسی آن بپردازید.
Script Type Menu : با انتخاب یکی از چهار گزینه موجود می‌توانید نتیجه اسکریپت‌های اضافه شده به صفحه که در قسمت Script Location Menu نمایش داده شده اند را فیلتر کنید. (متاسفانه این گزینه هم مشابه گزینه Track Throw/Catch برای بنده،

ورژن 1.11.4 نتیجه ای نداشته است.)

Script Location Menu : در این قسمت اسکریپت هایی که در صفحه وجود دارند نمایش داده می‌شوند. مشابه پنل HTML می‌توانید هنگام بازبودن این لیست، با تایپ کردن نتایج را فیلتر کنید.

همچنین با راست کلیک کردن بروی این قسمت می‌توانید آیتم یا فایل انتخاب شده را در ادیتور مورد نظر باز کنید، در پنل DOM بررسی کنید، فایل را در یک تب جدید باز کنید، آدرس فایل را در حافظه موقت کپی کنید.

Execution Control Buttons : این دکمه‌ها زمانی که دیباگر به یک Break Point برسد یا به دلیل فعال بودن دکمه‌های Break On ... متوقف شود، فعال می‌شوند و با کمک آن‌ها می‌توانید عملیات خطایابی را ادامه دهید.

در جدول زیر این دکمه‌ها و توضیحات تکمیلی هریک را مشاهده می‌کنید:

نوع	دکمه	Shortcut	توضیحات
اجرای مجدد		Shift + F18	Stack فعلی را مجدد اجرا می‌کند. (Stack: لیستی از توابع به ترتیبی که با فراخوانی آنها تابع جاری فراخونی شده است. در مقاله بعدی توضیحات بیشتری ارائه خواهد شد.)
ادامه		F8	اجرای برنامه را (تا Break Point بعدی) ادامه می‌دهد.
وارد شدن		F11	در صورت رسیدن به یک تابع، با زدن این دکمه دیباگر وارد بنده‌ی آن تابع می‌شود.
رد شدن		F10	اجرای برنامه را در سطح (Scope) فعلی ادامه می‌دهد. مشابه قبلی وارد تابع نمی‌شود.
خارج شدن		Shift + F11	کنترل به تابعی که تابع جاری را فراخوانی کرده است بازمیگردد. روشی سودمند زمانی که هنگام خطایابی وارد کدهای jQuery یا مثل آن می‌شوید :)

بعد از Script Location Menu قسمتی وجود دارد که وضعیت فعلی Stack را نشان می‌دهد و با کلیک بروی هرکدام، به کد آن قسمت هدایت می‌شوید. (البته اگر فایرباگ شما در این قسمت باگ نداشته باشد!)

Context Menu

تنها قابلیت جدیدی که در این منو وجود دارد، Run To This Line است. هنگامی که پنل در حالت دیباگ است، با راست کلیک کردن بروی خط مورد نظر و انتخاب گزینه‌ی Run to This Line می‌توانید خطوط میانی را رد کرده و به خط مورد نظر بروید. البته خطوطی که رد می‌شوند ، اجرا می‌شوند.
این کار معادل این است که درخط مورد نظر یک Break Point اضافه کنید و دکمه‌ی F8 را بزنید.

Breakpoints

توسط Break Pointها می‌توانید خطوطی از برنامه را برای خطایابی مشخص کنید. زمانی که دیباگر به نقطه‌ی مورد نظر برسد متوقف شده و می‌توانید به بررسی برنامه بپردازید. می‌توانید با بردن موس بروی متغییرها مقدار آن هارا بررسی کنید یا با کمک قسمت‌های Watch و Stack در پنل جانبی اطلاعات بیشتری در مورد اجرای برنامه در نقطه‌ی جاری بدست آورید.(در مورد پنل‌های جانبی در قسمت بعدی توضیح خواهم داد.) همچنین با کمک دکمه هایی که در جدول فوق توضیح داده شده اند روند اجرای برنامه را خط به خط ادامه دهید.
برای ایجاد یک Break Point، بروی شماره‌ی خط مورد نظر کلیک کنید. نقطه ای قرمز رنگ نمایش داده خواهد شد. البته دقت کنید که همه‌ی خطوط برنامه اجرا نمی‌شوند و نمی‌توانید در آن خطوط از این امکان بهره ببرید. شماره‌ی خطوط فعال با رنگ سبز مشخص شده اند.

امکان مفید دیگری که همراه با این قابلیت ارائه شده است (که در محیط‌های دیگر هم وجود دارد)، عبارت شرطی است. به این صورت که ضمن قرار دادن Break Point در یک نقطه از برنامه، می‌توانید یک شرط هم برای توقف برنامه تعیین کنید.
فرض کنید یک حلقه دارید که 300 بار تکرار می‌شود و مثلا در اجرای 250ام آن مشکلی وجود دارد. در این حالت می‌توانید از این قابلیت استفاده کنید و شرط توقف را *i == 250* قرار بدهید.
برای تعیین یک شرط برای یک Break Point، بروی خط مورد نظر راست کلیک کنید و شرط را در قسمت مشخص شده وارد کنید.

امکان مفید دیگری که وجود دارد، Break Point خودکار است. اگر از مقالات قبلی دکمه‌ی Break On All Errors در پنل Console و Break On Mutate در پنل HTML را بخاطر داشته باشید می‌دانید که در هر یک هنگام اجرای یک رخداد مورد نظر، دیباگر در خطی که موجب آن رخداد شده است متوقف شده و می‌توانید کنترل برنامه را بدست بگیرید. در این حالت نیازی به ایجاد Break Point نیست و FireBug بصورت خودکار این کار را انجام می‌دهد.

Search

این بخش در همه پنل‌ها وجود دارد با این تفاوت که در پنل Script و CSS دو گزینه‌ی *Multiple Files* و *Use Regular Expression* وجود دارد که به ترتیب امکان جستجو در فایل‌های js و css اضافه شده به صفحه هستند. قابلیت دیگری هم که فقط در پنل Script وجود دارد، پرش به یک خط در برنامه است به این صورت که با وارد کردن # و یک عدد به عنوان شماره‌ی خط، همان خط نمایش داده می‌شود.

در قسمت بعدی پنل جانبی که شامل سه بخش Stack، Watch و Breakpoints است را بررسی خواهیم کرد.

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 بهیود دهیم.
برای درک مطالبی که قصد توضیح آنها را در این قسمت داریم، یک صفحه کارگاهی دیگر نیز در فایل قابل دانلود این کتاب موجود می‌باشد که با نام chapter2/lab.wrapped.set.html قابل دسترسی می‌باشد. نکته مهم در مورد این صفحه کارگاهی آن است که می‌بایست عبارات و دستورهای کامل را با ساختار صحیح وارد کنیم در غیر این 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.

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

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

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: آموزش فایرباگ - #9 - Script Panel - Side Panels
نویسنده: احمد احمدی
تاریخ: ۱۳۹۲/۰۴/۰۴ ۲۲:۰
آدرس: www.dotnettips.info
برچسب‌ها: Debugging, JavaScript, FireBug

سه پنل Watch، Stack و Breakpoints پنل‌های جانبی پنل Script هستند که امکانات و اطلاعات مفیدی در ارتباط با قطعه کدی که در حال دیباگ آن هستیم ارائه می‌کنند.

Watch

این پنل متغییرها و تغییرات مقادیر آن‌ها در هنگام دیباگ را نمایش می‌دهد. بصورت پیشفرض متغییرهایی که در بلاک فعلی (که دیباگر در آن توقف کرده) ایجاد شده اند در این لیست قرار دارند. اما می‌توانید متغییرها و عبارات مورد نظر را در این قسمت وارد کنید. برای مثال می‌توانید مساوی بودن یک متغییر با یک مقدار را در این لیست وارد کنید و نتیجه را در هر اجرا مشاهده کنید. یا حتی یک متد را در این لیست وارد کنید تا در هر اجرا ، اجرا شود و نتیجه نمایش داده شود.
برای اضافه کردن یک عبارت جدید به این لیست، بروی عبارت *New watch expression...* در بالای پنل کلیک کرده و عبارت مورد نظر را وارد کنید و در نهایت کلید Enter را بزنید.
برای حذف عبارتی که وارد کردید، بروی علامت X کلیک کنید.
توجه کنید که هر عبارتی که در این قسمت وارد می‌کنید در اجرای هر خط از برنامه در حالت دیباگ اجرا می‌شود و اگر عبارتی که وارد کرده اید مقادیر داخل برنامه را تغییر دهند، برنامه‌ی شما متفاوت از حالت عادی عمل خواهد کرد.

Options Menu و Context Menu در این پنل همان موارد پنل DOM هستند و از آنجایی که همه‌ی این آیتم‌ها در این پنل به درستی عمل نمی‌کنند، توضیح آن را در مقاله‌ی مربوط به پنل DOM ارائه می‌کنیم.

Stack

در این پنل، زمانی که دیباگر متوقف شده است،لیستی از توابع که با فراخوانی یکدیگر به تابع جاری رسیده اند، نمایش داده می‌شوند. برای هر تابع هم پارامترها و مقادیر آن‌ها هنگام فراخوانی قابل مشاهده است. با کلیک بروی هر کدام از آن ها، خطی که

تابع در آن قرار دارد نمایش داده می‌شود و همچنین می‌توانید مقادیر متغییرهای محلی آن تابع هنگامی که فرخوانی شده است را هم بررسی کنید. (جالب نیست؟)

با راست کلیک کردن بروی نام این پنل یا باز کردن فلش آن، می‌توانید نمایش یا عدم نمایش این لیست در Toolbar را مشخص کنید.

Breakpoints

در این پنل لیست Break Pointهای اضافه شده نمایش داده می‌شود و می‌توانید آن هارا فعال، غیرفعال یا حذف نمایید.

برای فعال، غیرفعال و حذف کردن یک یا همه‌ی Break Pointها بروی قسمت از پنل راست کلیک کرده و یکی از گزینه‌های مورد نظر را انتخاب کنید.

Watch Stack Breakpoints

warnit test.html (line 38)
console.log('This is a warning. This is a warning. This is a wa

Error Breakpoints

badStuff test.html (line 263)
/*foo*/ B3();

HTML Breakpoints

body Break On Attribute Change
<body onload="console.log('Welcome')">

h1 Break On Child Addition or Removal
<h1>log Tests</h1>

h1 Break On Element Removal
<h1>log Tests</h1>

DOM Breakpoints

1
["hello", "world"]

XHR Breakpoints

test.html

عنوان: آموزش QUnit #1
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۲/۰۴/۱۷ ۱۹:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, jQuery, QUnit, Unit testing

مقدمه: تست و آزمایش کد برنامه‌ها و وب سایت‌هایمان، بهترین راه کاهش خطا و مشکلات آنها بعد از انتشار است. از جمله روش‌های موجود، تست واحد است که ویژوال استادیو نیز از آن برای پروژه‌های دات نت پشتیبانی می‌کند. با افزایش روز افزون کتابخانه‌های جاوا اسکریپتی و جی کوئری، نیاز به تست کدهای جاواسکریپتی نیز بیشتر به نظر می‌رسد و بهتر است تست واحد و آزمایش شوند. اما برخلاف کدهای C# و ASP.NET تست کدهای جاوا اسکریپت، مخصوصا زمانی که به دستکاری عناصر DOM می‌پردازیم و یا رویدادهای درون صفحه وب را با استفاده از جی کوئری می‌نویسیم، حتی اگر در فایل جداگانه‌ای نوشته شود، این بدان معنی نیست که آماده تست واحد است و ممکن است امکان نوشتن تست وجود نداشته باشد.
بنابراین چه چیزی یک تست واحد است؟ در بهترین حالت توابعی که مقداری را برمی گردانند، بهترین حالت برای تست واحد است. اما در بیشتر موارد شما نیاز دارید تا تاثیر کد را بر روی عناصر صفحه نیز مشاهده نمایید.

ساخت تست واحد

برای تست پذیری بهتر، توابع جاوا اسکریپت و هر کد دیگری، آن را می‌بایست طوری بنویسید که مقادیر تاثیر گذار در اجرای تابع به عنوان ورودی تابع در نظر گرفته شده باشند و همیشه نتیجه به عنوان خروجی تابع برگردانده شود؛ قطعه کد زیر را در نظر بگیرید:

```
function prettyDate(time){
    var date = new Date(time || ""),
      diff = (((new Date()).getTime() - date.getTime()) / 1000),
      day_diff = Math.floor(diff / 86400);

    if ( isNaN(day_diff) || day_diff < 0 || day_diff >= 31 )
      return;

    return day_diff == 0 && (
        diff < 60 && "just now" ||
        diff < 120 && "1 minute ago" ||
        diff < 3600 && Math.floor( diff / 60 ) +
          " minutes ago" ||
        diff < 7200 && "1 hour ago" ||
        diff < 86400 && Math.floor( diff / 3600 ) +
          " hours ago") ||
      day_diff == 1 && "Yesterday" ||
      day_diff < 7 && day_diff + " days ago" ||
      day_diff < 31 && Math.ceil( day_diff / 7 ) +
        " weeks ago";
  }
```

تابع pertyDate اختلاف زمان حال را نسبت به زمان ورودی، بصورت یک رشته برمی گرداند. اما در اینجا مقدار زمان حال، در خط سوم، در خود تابع ایجاد شده است و در صورتی که بخواهیم برای چندین مقدار آن را تست کنیم زمان حال متفاوتی در نظر گرفته می‌شود و حداکثر، زمان 31 روز قبل را نمایش داده و در بقیه تاریخ ها undefined را بر می‌گرداند. برای تست واحد، چند تغییر می‌دهیم.

بهینه سازی، مرحله اول:

پارامتری به عنوان مقدار زمان جاری برای تابع در نظر می‌گیریم و تابع را جدا کرده و در یک فایل جداگانه قرار می‌دهیم. فایل prettydate.js بصورت زیر خواهد شد.

```
function prettyDate(now, time){
  var date = new Date(time || ""),
    diff = (((new Date(now)).getTime() - date.getTime()) / 1000),
    day_diff = Math.floor(diff / 86400);

  if ( isNaN(day_diff) || day_diff < 0 || day_diff >= 31 )
    return;

  return day_diff == 0 && (
      diff < 60 && "just now" ||
      diff < 120 && "1 minute ago" ||
      diff < 3600 && Math.floor( diff / 60 ) +
```

```
        " minutes ago" ||
      diff < 7200 && "1 hour ago" ||
      diff < 86400 && Math.floor( diff / 3600 ) +
        " hours ago") ||
    day_diff == 1 && "Yesterday" ||
    day_diff < 7 && day_diff + " days ago" ||
    day_diff < 31 && Math.ceil( day_diff / 7 ) +
      " weeks ago";
}
```

حال یک تابع برای تست داریم، چند تست واحد واقعی می‌نویسیم

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Refactored date examples</title>
  <script src="prettydate.js"></script>
  <script>
  function test(then, expected) {
    results.total++;
    var result = prettyDate("2013/01/28 22:25:00", then);
    if (result !== expected) {
      results.bad++;
      console.log("Expected " + expected +
        ", but was " + result);
    }
  }
  var results = {
    total: 0,
    bad: 0
  };
  test("2013/01/28 22:24:30", "just now");
  test("2013/01/28 22:23:30", "1 minute ago");
  test("2013/01/28 21:23:30", "1 hour ago");
  test("2013/01/27 22:23:30", "Yesterday");
  test("2013/01/26 22:23:30", "2 days ago");
  test("2012/01/26 22:23:30", undefined);
  console.log("Of " + results.total + " tests, " +
    results.bad + " failed, " +
    (results.total - results.bad) + " passed.");
  </script>
</head>
<body>

</body>
</html>
```

در کد بالا یک تابع بدون استفاده از Qunit برای تست واحد نوشته ایم که با آن تابع prettyDate را تست می‌کند. تابع test مقدار زمان حال و رشته خروجی را گرفته و آن را با تابع اصلی تست می‌کند در آخر تعداد تست ها، تست‌های شکست خورده و تست های پاس شده گزارش داده می‌شود.

خروجی می‌تواند مانند زیر باشد:

.Of 6 tests, 0 failed, 6 passed

.Expected 2 day ago, but was 2 days ago

.f 6 tests, 1 failed, 5 passed

عنوان: **آموزش QUnit #2**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۲/۰۴/۲۱ ۱۲:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Unit testing, jQuery, QUnit

فریم ورک تست جاوا اسکریپت QUnit: انتخاب و استفاده از یک فریم ورک برای تست کدهای جاوا اسکریپت، قطعا نتیجه بهتری را به همراه خواهد داشت. من در این جا از QUnit که یکی از بهترین‌های تست واحد است، استفاده می‌کنم. برای این کار فایل‌های qunit.js و qunit.css را دانلود و مانند زیر برای تست واحد آماده کنید:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Refactored date examples</title>

  <link rel="stylesheet" href="../qunit.css">
  <script src="../qunit.js"></script>
  <script src="prettydate.js"></script>

  <script>
  test("prettydate basics", function() {
    var now = "2013/01/28 22:25:00";
    equal(prettyDate(now, "2013/01/28 22:24:30"), "just now");
    equal(prettyDate(now, "201308/01/28 22:23:30"), "1 minute ago");
    equal(prettyDate(now, "2013/01/28 21:23:30"), "1 hour ago");
    equal(prettyDate(now, "2013/01/27 22:23:30"), "Yesterday");
    equal(prettyDate(now, "2013/01/26 22:23:30"), "2 days ago");
    equal(prettyDate(now, "2012/01/26 22:23:30"), undefined);
  });
  </script>
</head>
<body>

<div id="qunit"></div>

</body>
</html>
```

در کد بالا ابتدا فایل‌های فریم ورک و فایل prettydate.js را اضافه کردیم. برای نمایش نتیجه تست، یک تگ div با نام qunit در بین تگ body اضافه می‌کنیم.

تابع test:

این تابع برای تست توابع نوشته شده، استفاده می‌شود. ورودی‌های این تابع، یکی عنوان تست و دومی یک متود دیگر، به عنوان ورودی دریافت می‌کند که در آن بدنه تست نوشته می‌شود.

تابع equal:

اولین تابع برای سنجش تست واحد equal است و در آن، تابعی که می‌خواهیم تست کنیم با مقدار خروجی آن مقایسه می‌شود. فایل را با نام test.htm ذخیره و آن را در مرورگر خود باز نمایید. خروجی در شکل آورده شده است:

همین طور که در تصویر بزرگ می‌بینید اطلاعات مرورگر، زمان تکمیل تست و تعداد تست، تعداد تست پاس شده و تعداد تست شکست خورده، نشان داده شده است.

اگر یکی از تست‌ها با شکست روبرو شود رنگ پس زمینه قرمز و جزئیات شکست نمایش داده می‌شوند.

بهینه سازی، مرحله اول:

در حال حاضر تست ما کامل نیست زیرا امکان تست n weeks ago یا تعداد هفته پیش میسر نیست. قبل از آنکه این را به آزمون اضافه کنیم، تغییراتی در تست می‌دهیم

```
test("prettydate basics", function() {
  function date(then, expected) {
    equal(prettyDate("2013/01/28 22:25:00", then), expected);
  }
  date("2013/01/28 22:24:30", "just now");
  date("2013/01/28 22:23:30", "1 minute ago");
  date("2013/01/28 21:23:30", "1 hour ago");
  date("2013/01/27 22:23:30", "Yesterday");
  date("2013/01/26 22:23:30", "2 days ago");
  date("2012/01/26 22:23:30", undefined);
});
```

تابع prettyDate را در تابع دیگری به نام date قرار می‌دهیم. این تغییر سبب می‌شود تا امکان مقایسه زمان ورودی تست جاری با تست قبلی فراهم شود.

تست دستکاری عناصر DOM:

تا اینجا با تست توابع آشنا شدید، حالا می‌خواهیم تغییراتی در prettyDate دهیم تا امکان انتخاب عناصر DOM و به روزرسانی آن نیز وجود داشته باشد. فایل prettyDate2.js در زیر آورده شده است:

```
var prettyDate = {
  format: function(now, time){
    var date = new Date(time || ""),
      diff = (((new Date(now)).getTime() - date.getTime()) / 1000),
      day_diff = Math.floor(diff / 86400);

    if ( isNaN(day_diff) || day_diff < 0 || day_diff >= 31 )
      return;

    return day_diff === 0 && (
        diff < 60 && "just now" ||
        diff < 120 && "1 minute ago" ||
        diff < 3600 && Math.floor( diff / 60 ) +
          " minutes ago" ||
        diff < 7200 && "1 hour ago" ||
        diff < 86400 && Math.floor( diff / 3600 ) +
          " hours ago") ||
      day_diff === 1 && "Yesterday" ||
      day_diff < 7 && day_diff + " days ago" ||
      day_diff < 31 && Math.ceil( day_diff / 7 ) +
        " weeks ago";
  },

  update: function(now) {
    var links = document.getElementsByTagName("a");
    for ( var i = 0; i < links.length; i++ ) {
      if ( links[i].title ) {
        var date = prettyDate.format(now, links[i].title);
        if ( date ) {
          links[i].innerHTML = date;
        }
      }
    }
  }
};
```

prettyDate شامل دو تابع، یکی format که weeks ago به آن اضافه گردیده و تابع update که با انتخاب تگ‌ها، مقدار title را به تابع فرمت و خروجی آن را در Html هر عنصر قرار می‌دهد. حال یک تست واحد می‌نویسیم:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Refactored date examples</title>
  <link rel="stylesheet" href="../qunit.css">
  <script src="../qunit.js"></script>
  <script src="prettydate2.js"></script>
  <script>
  test("prettydate.format", function() {
    function date(then, expected) {
      equal(prettyDate.format("2013/01/28 22:25:00", then),
        expected);
    }
    date("2013/01/28 22:24:30", "just now");
    date("2013/01/28 22:23:30", "1 minute ago");
    date("2013/01/28 21:23:30", "1 hour ago");
    date("2013/01/27 22:23:30", "Yesterday");
    date("2013/01/26 22:23:30", "2 days ago");
    date("2012/01/26 22:23:30", undefined);
  });

  function domtest(name, now, first, second) {
    test(name, function() {
      var links = document.getElementById("qunit-fixture")
        .getElementsByTagName("a");
      equal(links[0].innerHTML, "January 28th, 2013");
      equal(links[2].innerHTML, "January 27th, 2013");
      prettyDate.update(now);
      equal(links[0].innerHTML, first);
      equal(links[2].innerHTML, second);
    });
```

```
  }
  domtest("prettyDate.update", "2013-01-28T22:25:00Z",
    "2 hours ago", "Yesterday");
  domtest("prettyDate.update, one day later", "2013/01/29 22:25:00",
    "Yesterday", "2 days ago");
  </script>
</head>
<body>

<div id="qunit"></div>
<div id="qunit-fixture">

<ul>
  <li id="post57">
    <p>blah blah blah...</p>
    <small>
      Posted <span>
        <a href="/2013/01/blah/57/" title="2013-01-28T20:24:17Z"
          >January 28th, 2013</a>
      </span>
      by <span><a href=""></a></span>
    </small>
  </li>
  <li id="post57">
    <p>blah blah blah...</p>
    <small>
      Posted <span>
        <a href="/2013/01/blah/57/" title="2013-01-27T22:24:17Z"
          >January 27th, 2013</a>
      </span>
      by <span><a href=""></a></span>
    </small>
  </li>
</ul>

</div>

</body>
</html>
```

همین طور که مشاهده می‌کنید در تست واحد اول خود تابع prettyDate.format را تست نموده ایم. در تست بعدی عناصر DOM نیز دستکاری و تست شده است. تابع domtest با جستجوی تگ qunit-fixture و تگ‌های a درون آن، مقدار نهایی html آن با مقدار داده شده، مقایسه شده است.

Refactored date examples ■ noglobals ■ notrycatch

☐ Hide passed tests

Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.71 Safari/537.36

Tests completed in 268 milliseconds.
12 tests of 14 passed, 2 failed.

1. prettydate.format (0, 6, 6) Rerun

2. prettyDate.update (0, 4, 4) Rerun

3. prettyDate.update, one day later (2, 2, 4) Rerun

1. okay

2. okay

3. failed
Expected: "Yesterday"
Result: "22 hours ago"
Diff: "Yesterday" "22 hours ago"
Source: at Object.<anonymous>

4. failed
Expected: "2 days ago"
Result: "Yesterday"
Diff: "2 days ago" "Yesterday"
Source: at Object.<anonymous>

در شکل بالا نتیجه تست واحد نشان داده شده است.

عنوان: آموزش QUnit #3
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۲/۰۴/۲۴ ۱۷:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Unit testing, jQuery, QUnit

در قسمت‌های قبلی با مفهوم تست واحد و کتابخانه quint آشنا شدید و مثالی را نیز با هم بررسی کردیم. در ادامه به قابلیت‌های بیشتر این کتابخانه می‌پردازیم.

توابع اعلان نتایج:

qunit سه تابع را جهت اعلان نتایج تست واحد فراهم نموده است

تابع ok:

تابع پایه‌ای تست واحد، دو پارامتر را به عنوان ورودی دریافت می‌کند و در صورتیکه بررسی نتیجه پارامتر اول برابر true باشد، تست با موفقیت روبرو شده است. پارامتر دوم برای نمایش یک پیام است. در مثال زیر حالت‌های مختلف آن بررسی شده است. مقادیر true، non-empty string به معنی موفقیت و مقادیر null،"",NaN،0،false و undefined به معنی شکست تست می‌باشد. در واقع خروجی تابع ارسالی به اعلان ok یکی از نتایج بالا می‌تواند باشد.

```
//ok( truthy [, message ] )

test( "ok test", function() {
  ok( true, "true succeeds" );
  ok( "non-empty", "non-empty string succeeds" );

  ok( false, "false fails" );
  ok( 0, "0 fails" );
  ok( NaN, "NaN fails" );
  ok( "", "empty string fails" );
  ok( null, "null fails" );
  ok( undefined, "undefined fails" );
});
```

تابع equal:

این اعلان یک مقایسه ساده بین پارامتر اول و دوم تابع می‌باشد که شرط برابری(==) را بررسی می‌نماید. وقتی مقدار اول و دوم برابر باشند، اعلان موفقیت و در غیر این صورت، تست با شکست رویرو شده و هر دو پارامتر نمایش داده می‌شوند.

```
//equal( actual, expected [, message ] )

test( "equal test", function() {
  equal( 0, 0, "Zero; equal succeeds" );
  equal( "", 0, "Empty, Zero; equal succeeds" );
  equal( "", "", "Empty, Empty; equal succeeds" );
  equal( 0, 0, "Zero, Zero; equal succeeds" );

  equal( "three", 3, "Three, 3; equal fails" );
  equal( null, false, "null, false; equal fails" );
});
```

زمانی که می‌خواهید مؤکداً شرط === را بررسی نمایید از strictEqual() استفاده کنید.

تابع deepEqual:

تکمیل شده دو تابع قبل می‌باشد و حتی امکان مقایسه دو شی را نیز با هم دارا است. علاوه بر این، امکان مقایسه NaN، تاریخ، عبارات باقاعده، آرایه‌ها و توابع نیز وجود دارند.

```
//deepEqual( actual, expected [, message ] )

test( "deepEqual test", function() {
  var obj = { foo: "bar" };
```

```
  deepEqual( obj, { foo: "bar" }, "Two objects can be the same in value" );
});
```

در صورتیکه نمی‌خواهید محتوای دو مقدار را با هم مقایسه کنید، از equal استفاده نمایید اما عموما deepEqual انتخاب بهتری می‌باشد.

تست عملیات کاربر:

گاهی لازم است رویدادهایی که از عملیات کاربران صدا زده می‌شوند تست شوند. در این موارد با صدا زدن تابع trigger جی‌کوئری، تابع مورد نظر را تست نمایید. به مثال زیر توجه نمایید:

```
function KeyLogger( target ) {
  if ( !(this instanceof KeyLogger) ) {
    return new KeyLogger( target );
  }
  this.target = target;
  this.log = [];

  var self = this;

  this.target.off( "keydown" ).on( "keydown", function( event ) {
    self.log.push( event.keyCode );
  });
}
```

این مثال یک گزارش دهنده است و در صورتیکه کاربر، کلیدی را فشار دهد، کد آن را گزارش می‌دهد و در آرایه log ذخیره می‌نماید. حال لازم است بصورت دستی این رویداد را صدا زده و تایع را تست کنیم. تست را بصورت زیر می‌نویسیم:

```
test( "keylogger api behavior", function() {

  var event,
      $doc = $( document ),
      keys = KeyLogger( $doc );

  // trigger event
  event = $.Event( "keydown" );
  event.keyCode = 9;
  $doc.trigger( event );

  // verify expected behavior
  equal( keys.log.length, 1, "a key was logged" );
  equal( keys.log[ 0 ], 9, "correct key was logged" );

});
```

برای این کار تابع KeyLogger را با شی document جی کوئری صدا زدیم و نتیجه را در متغییر keys قرار داده‌ایم. بعد رویداد keydown را با کد 9 پرکرده تایع trigger متغییر $doc را با مقدار event صدا زده‌ایم که در واقع بصورت دستی، یک رویداد اتفاق افتاده است. در آخر هم با اعلان equal تست واحد را انجام داده‌ایم.

عنوان: **Html Encoding**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۵/۰۷ ۲۲:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, HTML, Html Encoding, HtmlEncode, HtmlDecode

.

.

مقدمه

در دنیای وب دو انکدینگ معروف داریم: Url Encoding و Html Encoding. در هر کدام از این انکدینگ‌ها یک عملیات کلی صورت می‌گیرد: *تبدیل کاراکترهای غیرمجاز به عبارات معادل مجاز* .

Url Encoding همان‌طور که از نامش پیداست روشی برای کدکردن Url هاست. مثل عبارت کدشده زیر:

```
Hello%20world%20,%20hi
```

درواقع کاراکتر مشخص‌کننده رشته‌ای که Url Encoding احتمالا در آن اعمال شده است، همان کاراکتر **%** است. بحث درباره این نوع انکدینگ کمی مفصل است که خود مطلب جداگانه‌ای می‌طلبد. (اطلاعات بیشتر)

Html Encoding نیز با توجه به نامش برای انکدینگ عبارات HTML استفاده می‌شود. مثلا عبارت زیر را درنظر بگیرید:

```
<html>encoding</html>
```

این عبارت پس از اعمال عملیات Html Encoding به صورت زیر در خواهد آمد:

```
&lt;html&gt;encoding&lt;/html&gt;
```

می‌بینید که در اینجا کاراکترهای > و < به صورت عبارات **<** و **>** در آمده‌اند. شرح کاملی درباره این عبارات معادل (که اصطلاحا به آن‌ها character entity می‌گویند) در اینجا آورده شده است.

در حالت کلی Html Encoding شامل کدکردن 5 کاراکتر زیر است:

.

کاراکتر	عبارت معادل	توضیحات
>	>	
<	<	
"	"	
'	'	یا ' به غیر از IE
&	&	

نکته: در برخی استانداردها (بیشتر برای XML) برای کاراکتر ' از عبارت ' استفاده می‌شود. این عبارت جایگزین به غیر از IE در بقیه مرورگرها درست کار می‌کند.

این کاراکترها درواقع از عناصر اصلی تشکیل‌دهنده ساختار Html هستند، بنابراین وجود آن‌ها درون یک متن می‌تواند در روند رندر صفحات html اختلال ایجاد کند. بنابراین با استفاده از Html Encoding و تبدیل این کاراکترها به معادلشان (عباراتی که مرورگرها آن‌ها را می‌شناسند)، می‌توان از نمایش درست این کاراکترها مطمئن شد. البته یکی دیگر از دلایل مهم اعمال این انکدینگ، افزایش امنیت و جلوگیری از حملات XSS است.

فرمت این عبارات معادل به صورت **&entity_name;** است. به کل این عبارت اصطلاحا Character Entity گفته می‌شود. این عبارات با کاراکتر **&** شروع شده و به یک کاراکتر **;** ختم می‌شوند. کلمه میان این دو کاراکتر نیز عبارت جایگزین (یا همان entity name) هر یک از این کاراکترهاست که در لینک بالا به همراه بسیاری دیگر از کاراکترها اشاره شده است (^).
روش دیگری نیز برای کدکردن کاراکترها با فرمت **&#entity_number;** وجود دارد. این entity_number درواقع کد کاراکتر مربوطه در جدول کاراکترست جاری مرورگر است. معمولا این کدها منطبق بر جدول ASCII هستند. برای کاراکترهای خارج از جدول اسکی هم از سایر جداول (مثلا یونیکد) استفاده می‌شود. عملیات انکدینگ برای کاراکترهای با کد 160 تا 255 (براساس استاندارد ISO-8859-1) با این روش انجام می‌شود (^). اطلاعات بیشتر راجع به این کدها در اینجا آورده شده است.

خوشبختانه در سمت سرور، در دات‌نت روش‌های گوناگون و قابل اطمینانی برای اعمال این انکدینگ وجود دارد. اما متاسفانه در سمت کلاینت چنین امکاناتی اصلا فراهم نیست و برنامه نویسان خود باید دست به کار شوند. ازآنجاکه امروزه قسمت‌های بیشتری از اپلیکیشن‌های تحت وب در سمت کلاینت پیاده می‌شوند و کتابخانه‌های سمت کلاینت روز به روز پرطرفدارتر می‌شوند وجود نمونه‌های مشابه از این متدها در سمت کلاینت می‌تواند بسیار مفید باشد.
بنابراین تمرکز اصلی ادامه این مطلب بیشتر بر نحوه اعمال این انکدینگ در سمت کلاینت با استفاده از زبان جاوا اسکریپت است.
.

.

Html Encoding در دات‌نت

در دات‌نت متدهای متعددی برای اعمال Html Encoding وجود دارد. برخی از آن‌ها صرفا برای اسناد HTML طراحی شده‌اند و برخی دیگر یک پیاده‌سازی کلی دارند و بعضی نیز برای فایل‌های XML ارائه شده‌اند. این متدها عبارتند از:

متد System.Security.SecurityElement.Escape : این متد بیشتر برای اعمال این انکدینگ در XML به‌کار می‌رود. در این متد 5 کاراکتر اشاره شده در بالا به عبارات معادل انکد می‌شوند. البته برای کاراکتر ' از عبارت ' استفاده می‌شود.

متدهای موجود در System.Net.WebUtility : متدهای HtmlEncode و HtmlDecode موجود در این کلاس عملیات انکدینگ را انجام می‌دهند. این کلاس از دات‌نت 4 اضافه شده است.

متدهای کلاس System.Web.HttpUtility : در این کلاس از متدهای موجود در کلاس System.Web.Util.HttpEncoder استفاده می‌شود. در پیاده‌سازی پیش‌فرض، متدهای این کلاس از متدهای موجود در کلاس WebUtility استفاده می‌کنند. البته می‌توان با فراهم کردن یک Encoder سفارشی و تنظیم آن در فایل کانفیگ (خاصیت encoderType در قسمت HttpRuntime) این رفتار را تغییر داد. دلیل اصلی جابجایی مکان پیاده‌سازی این متدها از دات نت 4 به بعد نیز به همین دلیل است. (اطلاعات بیشتر ^ و ^).

متدهای موجود در System.Web.HttpServerUtility : متدهای HtmlEncode و HtmlDecode موجود در این کلاس مستقیما از متدهای موجود در کلاس HttpUtility استفاده می‌کنند. خاصیت Server موجود در HttpContext یا در کلاس Page از نوع این کلاس است.

متدهای موجود در کلاس System.Web.Security.AntiXss.AntiXssEncoder : این کلاس از دات نت 4.5 اضافه شده است. همانطور که از نام این کلاس بر می‌آید، از HttpEncoder مشتق شده است که در متدهای مرتبط با html encoding تغییراتی در آن اعمال شده است. متدهای این کلاس برای امنیت بیشتر به جای استفاده از Black List از یک White List استفاده می‌کنند.

درحال حاضر بهترین گزینه موجود برای عملیات انکدینگ، متدهای موجود در کلاس WebUtility هستند. ازآنجاکه این کلاس در فضای System.Net و در کتابخانه System.dll قرار دارد (کتابخانه‌ای که معمولا برای تمام برنامه‌های دات‌نتی نیاز است)، بنابراین بارگذاری آن در برنامه نیز بار اضافی بر حافظه تحمیل نمی‌کند.
پیاده‌سازی عملیات HtmlEncode کار سختی نیست. مثلا می‌توان برای سادگی از متد Replace استفاده کرد. اما برای رشته‌های طولانی این متد کارایی مناسبی ندارد. به همین دلیل در تمام پیاده‌سازی‌ها، معمولا از یک حلقه بر روی تمام کاراکترهای رشته موردنظر برای یافتن کاراکترهای غیرمجاز استفاده می‌شود. در کدهای متدهای موجود، برای افزایش سرعت حتی از اشاره‌گر و کدهای unsafe نیز استفاده شده است.
برای افزایش کارایی در تولید رشته نهایی تبدیل‌شده، بهتر است از یک StringBuilder استفاده شود. در پیاده‌سازی‌های متدهای بالا برای اینکار معمولا از یک TextWriter استفاده می‌شود. TextWriterهای موجود از کلاس StrigBuilder برای دستکاری رشته‌ها استفاده می‌کنند.

صرفا جهت آشنایی بیشتر، پیاده‌سازی خلاصه‌شده متد HtmlEncode در کلاس WebUtility در زیر آورده شده است:

```
public static unsafe void HtmlEncode(string value, TextWriter output)
{
  int index = IndexOfHtmlEncodingChars(value, 0);
  if (index == -1)
  {
    output.Write(value);
    return;
  }
  int cch = value.Length - index;
  fixed (char* str = value)
  {
    char* pch = str;
    while (index-- > 0)
    {
      output.Write(*pch++);
    }
    while (cch-- > 0)
    {
      char ch = *pch++;
      if (ch <= '>')
      {
        switch (ch)
        {
```

```
          case '<':
            output.Write("&lt;");
            break;
          case '>':
            output.Write("&gt;");
            break;
          case '"':
            output.Write("&quot;");
            break;
          case '\'':
            output.Write("&#39;");
            break;
          case '&':
            output.Write("&amp;");
            break;
          default:
            output.Write(ch);
            break;
        }
      }
      else if (ch >= 160 && ch < 256)
      {
        // The seemingly arbitrary 160 comes from RFC
        output.Write("&#");
        output.Write(((int)ch).ToString(NumberFormatInfo.InvariantInfo));
        output.Write(';');
      }
      else
      {
        output.Write(ch);
      }
    }
  }
}
private static unsafe int IndexOfHtmlEncodingChars(string s, int startPos)
{
  int cch = s.Length - startPos;
  fixed (char* str = s)
  {
    for (char* pch = &str[startPos]; cch > 0; pch++, cch--)
    {
      char ch = *pch;
      if (ch <= '>')
      {
        switch (ch)
        {
          case '<':
          case '>':
          case '"':
          case '\'':
          case '&':
            return s.Length - cch;
        }
      }
      else if (ch >= 160 && ch < 256)
      {
        return s.Length - cch;
      }
    }
  }
  return -1;
}
```

در ابتدا بررسی می‌شود که آیا اصلا متن ورودی حاوی کاراکترهای غیرمجاز است یا خیر. درصورت عدم وجود چنین کاراکترهایی، کار متد با برگشت خود متن ورودی پایان می‌یابد. درغیراینصورت عملیات انکدینگ آغاز می‌شود. همان‌طور که می‌بینید عملیات انکدینگ برای 5 کاراکتر اشاره شده به صورت جداگانه انجام می‌شود و برای کاراکترهای با کد 160 تا 255 (با توجه به توضیحات موجود در مقدمه) نیز با استاندارد &#code; عملیات تبدیل انجام می‌شود.

در سمت دیگر، پیاده‌سازی بهینه متد HtmlDecode چندان ساده نیست. چون به جای یافتن یک کاراکتر غیرمجاز باید به دنبال عبارات چند کاراکتری معادل گشت که کاری نسبتا پیچیده است.

اطلاعات و پیاده‌سازی نسبتا کاملی درباره Html Encoding در سمت سرور در اینجا قابل مشاهده است.

نکته: درصورت نیاز به کدکردن سایر کاراکترها (مثلا کاراکترهای یونیکد) پیاده‌سازی‌های موجود کارا نخواهند بود. بنابراین باید encoder سفارشی خود را تهیه کنید. مثلا می‌توانید شرط دوم در بررسی کد کاراکترها را بردارید (منظور قسمت ch < 256) که در این‌صورت متد شما محدوده وسیعی را پوشش می‌دهد. اما دقت کنید که با این تغییر متدی سفارشی برای عملیات decode نیز باید تهیه کنید!

.

.

Html Encoding در جاوا اسکریپت

برای انجام عملیات Url Encoding در جاوا اسکریپت چند متد توکار وجود دارد، که فرایند کلی عملیات همه آن‌ها تقریبا یکسان است. اما متاسفانه برای انجام عملیات Html Encoding متدی در جاوا اسکریپت وجود ندارد. بنابراین متدهای مربوطه باید توسط خود برنامه‌نویسان پیاده‌سازی شوند.

یک روش برای اینکار استفاده از لیست اشاره‌شده در بالا و انجام عملیات replace برای تمام این کاراکترهاست (5 کاراکتر اصلی و درصورت نیاز سایر کاراکترها). این کار می‌تواند کمی سخت باشد و درواقع پیاده‌سازی چنین متدی نسبتا مشکل نیز هست (مخصوصا عملیات decode).
اما خوشبختانه امکانی در اسناد html وجود دارد که این کار (مخصوصا Decode کردن) را آسان می‌کند.

این روش جالب برای انجام عملیات Html Encoding در جاوا اسکریپت، استفاده از یک قابلیت توکار در مرورگرهاست. عناصر DOM (مانند div) دو خاصیت **innerText** و **innerHTML** دارند که مرورگرها با توجه به مقادیر تنظیم‌شده برای هر یک، عملیات coding و decoding مربوطه را به صورت کاملا خودکار انجام داده و مقدار خاصیت دیگر را به‌روزرسانی می‌کنند (دقت کنید که در این دو پراپرتی، کلمه HTML کاملا با حروف بزرگ است، برخلاف Text که تنها حرف اول آن بزرگ است).

برای روشن‌تر شدن موضوع به مثال زیر برای عملیات encode توجه کنید:

```
<div id="log"></div>
<script type="text/javascript">
  var element = document.getElementById('log');
  element.innerText = '<html> encoding </html>';
  console.log(element.innerHTML);
</script>
```

که خروجی زیر را خواهد داشت:

```
&lt;html&gt; encoding &lt;/html&gt;
```

عکس این عملیات یعنی decoding نیز با استفاده از کدی مثل زیر امکان‌پذیر است:

```
<div id="log">
</div>
<script type="text/javascript">
  var element = document.getElementById('log');
  element.innerHTML = "&lt;html&gt; encoding &lt;/html&gt;";
  console.log(element.innerText);
</script>
```

خروجی کد بالا به صورت زیر است:

```
<html> encoding </html>
```

می‌بینید که با استفاده از این ویژگی جالب، می‌توان عملیات Html Encoding را انجام داد. در ادامه پیاده‌سازی مناسب این دو متد آورده شد است.

. .

.

متد htmlEncode

برای پیاده‌سازی این متد برای حالت استفاده مستقیم داریم:

```
String.htmlEncode = function (s) {
  var el = document.createElement("div");
  el.innerText = s || '';
  return el.innerHTML;
};
```

در اینجا با استفاده از متد createElement شی document یک المان DOM (در اینجا div) ایجاد شده و سپس با توجه به توضیحات بالا خاصیت innerText آن به مقدار ورودی تنظیم می‌شود. استفاده از عبارت '' || s در اینجا برای جلوگیری از برگشت عبارات ناخواسته (مثل undefined یا null) برای ورودی‌های غیرمجاز است. درنهایت خاصیت innerHTML این المان به عنوان رشته انکدشده برگشت داده می‌شود.

نحوه استفاده از این متد به صورت زیر است:

```
console.log(String.htmlEncode("<html>"));
//result:   &lt;html&gt;
```

و برای حالت استفاده از خاصیت prototype داریم:

```
String.prototype.htmlEncode = function () {
  var el = document.createElement("div");
  el.innerText = this.toString();
  return el.innerHTML;
};
```

نحوه استفاده از این متد نیز به صورت زیر است:

```
console.log("<html>".htmlEncode());
//result:    &lt;html&gt;
```

.

.

متد htmlDecode

با استفاده از مطالب اشاره‌شده در بالا، پیاده‌سازی این متد به صورت زیر است:

```
String.htmlDecode = function (s) {
  var el = document.createElement("div");
  el.innerHTML = s || '';
  return el.innerText;
};
```

و به‌صورت خاصیتی از prototype شی String داریم:

```
String.prototype.htmlDecode = function () {
  var el = document.createElement("div");
```

```
  el.innerHTML = this.toString();
  return el.innerText;
};
```

نحوه استفاده از این متدها هم به صورت زیر است:

```
console.log(String.htmlDecode("&lt;html&gt;"));
console.log("&lt;html&gt;".htmlDecode());
```

.

.

پیاده‌سازی با استفاده از jQuery

درصورت در دسترس بودن کتابخانه jQuery، کار پیاده‌سازی این متدها بسیار ساده‌تر خواهد شد. برای این‌کار می‌توان از متدهای زیر استفاده کرد:

.

- متد **htmlEncode** :

```
String.htmlEncode = function (s) {
  return $('<div/>').text(value).html();
};

String.prototype.htmlEncode = function () {
  return $('<div/>').text(this.toString()).html();
};
```

.

- متد **htmlDecode** :

```
String.htmlDecode = function (s) {
  return $('<div/>').html(s).text();
};

String.prototype.htmlDecode = function () {
  return $('<div/>').html(this.toString()).text();
};
```

.

.

نکات پایانی

1. با اینکه به نظر می‌رسد در متدهای ارائه شده در بالا، بین نسخه‌های معمولی و نسخه مخصوص jQuery تفاوتی وجود ندارد اما تست زیر نشان می‌دهد که نکات ریزی باعث به‌وجود آمدن برخی تفاوت‌ها می‌شود. رشته زیر را درنظر بگیرید:

```
var value = "a \n b";
```

با استفاده از متد htmlEncode معمولی نشان داده شده در بالا، عبارت انکدشده رشته فوق به صورت زیر خواهد بود:

"a <br> b"

می‌بینید که به صورت هوشمندانه‌ای! مقدار \n به تگ
 انکد شده است. اما اگر با استفاده از متد نوشته شده با jQuery سعی

به انکدکردن این رشته کنیم، می‌بینیم که مقدار \n بدین صورت انکد نمی‌شود! حال کدام روش درست و استاندارد است؟

در ابتدای این مطلب هم اشاره شده بود که Html Encoding برای کدکردن یکسری کاراکتر غیرمجاز در متون موجود در صفحات HTML بکار می‌رود و معمولا همان 5 کاراکتر اشاره‌شده در بالا به عنوان کاراکترهای اصلی غیرمجاز به حساب می‌آیند. کاراکتر \n از این نوع کاراکترها محسوب نمی‌شود. هم‌چنین ازآنجاکه عملیات عکس این تبدیل در Decode مربوطه صورت نمی‌گیرد، تبدیل این کاراکتر به معادلش در html اصلا کاری منطقی نیست و باعث خراب شدن متن موردنظر می‌شود.

با استفاده از متدهای HtmlEncode موجود در کلاس‌های دات نت (WebUtility و HtmlUtility که در بالا به آن‌ها اشاره شده بود) عملیات انکدینگ برای این رشته تکرار شد و نتیجه حاصله نشان داد که عبارت \n در خروجی این متدها نیز انکد نمی‌شود. بنابراین متد نوشته شده با استفاده از jQuery خروجی‌های استانداردتری ارائه می‌دهد.

با کمی تحقیق و بررسی کدهای jQuery مشخص شد که دلیل این تفاوت، در استفاده از متد createTextNode از شی document در متد text() است. بنابراین برای بهبود متد htmlEncode اولیه داریم:

```
String.htmlEncode = function (s) {
  var el = document.createElement("div");
  var txt = document.createTextNode(s);
  el.appendChild(txt);
  return el.innerHTML;
};
```

با استفاده از این متد نتایج مشابه متد نوشته شده با jQuery حاصل خواهد شد.

.

.

2. نکته مهم دیگری که باید بدان توجه داشت برقراری اصل مهم زیر در عملیات انکدینگ است:

```
String.htmlDecode(String.htmlEncode(myString)) === myString;
```

حال سعی می‌کنیم که برقراری این شرط را در یک مثال بررسی کنیم:

```
var myString = "<HTML>";
String.htmlDecode(String.htmlEncode(myString)) === myString;
// result:   true
// ---------------------------------------------------------------------------
myString = "<اچ تی ام ال>";
String.htmlDecode(String.htmlEncode(myString)) === myString;
// result:   true
```

تا اینجا همه چیز ظاهرا درست پیش رفته است. اما حالا مثال زیر را درنظر بگیرید:

```
myString = "a \r b";
String.htmlDecode(String.htmlEncode(myString)) === myString;
// result:   false
```

می‌بینید که با وارد شدن کاراکتر \r کار خراب می‌شود. این نتیجه برای تمامی متدهای جاوا اسکریپتی نشان داده شده صادق است. اما متدهای دات نتی اشاره شده در ابتدای این مطلب با این کاراکتر مشکلی ندارند و نتیجه درستی برمی‌گردانند. بنابراین یک جای کار می‌لنگد!

پس از کمی تحقیق و بررسی بیشتر مشخص شد که مرورگرها در تبدیل کاراکترها، کاراکتر carriage return (یا CR یا همان \r با کد اسکی 13 یا 0D) را تبدیل به کاراکتر line feed (یا LF یا \n با کد اسکی 10 یا 0A) می‌کنند. برای آزمایش این نکته می‌توانید از سه خط زیر استفاده کنید:

```
console.log(escape(String.htmlDecode('\r'))); // result:    %0A  :  it is url encode of character '\n'
```

```
console.log(escape(String.htmlDecode('\n'))); // result:    %0A
console.log(escape(String.htmlDecode('\r\n'))); // result:    %0A
```

با بررسی بیشتر مشخص شد که این تبدیل به محض مقداردهی به یکی از خاصیت‌های یک عنصر DOM صورت می‌گیرد. برای مثال کد زیر را در مرورگرهای مختلف امتحان کنید:

```
var el = document.createElement('div');
el.innerText = '\r';
console.log(escape(el.innerText)); // result:    %0A
el.innerHTML = '\r';
console.log(escape(el.innerHTML)); // result:    %0A
console.log(escape('\r')); // result:    %0D
```

با بررسی هایی که من کردم دلیل و یا راه‌حلی برای این مشکل پیدا نکردم!
بنابراین در استفاده از این متدها باید این نکته را مدنظر قرار داد. ازآنجاکه این مشکل حالتی به خصوص دارد نمی‌توان راه‌حلی کلی برای آن ارائه داد. پس برای موقعیت‌های گوناگون با توجه به زوایای روشن‌شده از این مشکل باید به دنبال راه‌حل مناسب بود.

البته ممکن است این اشکال درمورد کاراکترهای دیگری هم وجود داشته باشد که من به آن برخورد نکرده باشم (با درنظر گرفتن تفاوت میان مرورگرهای مختلف ممکن است پیچیده‌تر هم باشد).

نکته: ازآنجاکه برای رفع این مشکل، پیاده‌سازی متد htmlDecode به این کاملی، با عدم استفاده از ویژگی پراپرتی‌های innerHTML و innerText، کاری نسبتا سخت و پیچیده و طولانی است، بنابراین در بیشتر حالات می‌توان از این مشکل صرف‌نظر کرد! به همین دلیل در اینجا نیز متد دیگری برای رفع این مشکل ارائه نمی‌شود!

.

.

3. یک مشکل دیگر که این متدها دارند این است که متاسفانه در متد htmlEncode، از 5 کاراکتر معروف بالا، کاراکترهای ' و " در این متدها اصلا تبدیل نمی‌شوند. همچنین سایر کاراکترهای عنوان‌دار یا کاراکترهای خارج از جدول ASCII (مثلا کاراکترهای با کد 160 تا 255 یا کاراکترهای یونیکد) نیز که معمولا انکد می‌شوند در این متد تغییری نمی‌کنند و به همان صورت برگشت داده می‌شوند.
هرچند متد htmlDecode نشان داده شده در این مطلب، به‌درستی تمامی عبارات معادل (حتی عبارات معادل غیر از 5 کاراکتر نشان داده شده در بالا با هر دو استاندارد &character-entity; و &#code;) را تبدیل کرده و کاراکتر درست را برمی‌گرداند.

برای اصلاح این مشکل می‌توان متد htmlEncode را کاملا به صورت دستی و مستقیم نوشت و اعمال انکدینگ‌های موردنیاز را با استفاده یک حلقه روی تمام کاراکترها متن موردنظر انجام داد. چیزی شبیه به کد زیر:

```
String.htmlEncode = function (text) {
  text = text || '';
  var encoded = '';
  for (var i = 0; i < text.length; i++) {
    var c = text[i];
    switch (c) {
      case '<':
        encoded += '&lt;';
        break;
      case '>':
        encoded += '&gt;';
        break;
      case '&':
        encoded += '&amp;';
        break;
      case '"':
        encoded += '&quot;';
        break;
      case "'":
```

```
        encoded += '&#39;';
        break;
      default:
        // the upper limit can be removed to support more chars...
        var code = c.charCodeAt();
        if (code >= 160 & code < 256)
          encoded += '&#' + code + ';';
        else
          encoded += c;
    }
  }
  return encoded;
};
```

روش استفاده شده در متد بالا همانند متد HtmlEncode در کلاس WebUtility است.

.

.

کتابخانه‌های موجود

هرچند توضیحات ارائه شده در این مطلب کافی هستند، اما صرفا برای آشنایی با سایر کتابخانه‌های موجود، روش‌های استفاده‌شده در آن‌ها و نقایص و مزایای آن‌ها این قسمت اضافه شده است.

.

Prototype

: این کتابخانه شامل مجموعه‌ای از متدهای کمکی برای راحتی کار در سمت کلاینت است. برای عملیات html encoding دو متد escapeHTML و unescapeHTML دارد که به صورت زیر پیاده شده‌اند:

```
function escapeHTML() {
  return this.replace(/&/g, '&amp;').replace(/</g, '&lt;').replace(/>/g, '&gt;');
}

function unescapeHTML() {
  return this.stripTags().replace(/&lt;/g, '<').replace(/&gt;/g, '>').replace(/&amp;/g, '&');
}
```

همان‌طور که می‌بینید در این متدها از replace استفاده شده است که برای متن‌های طولانی کندتر از روش‌های نشان داده‌شده در این مطلب است. هم‌چنین عملیات انکد و دیکد را تنها برای 3 کاراکتر < و > و & انجام می‌دهد که نقص بزرگی محسوب می‌شود.

.

jQuery.string

: این پلاگین حاوی چند متد برای کار با رشته‌هاست که یکی از این متدها با نام htmlspecialchars مخصوص عملیات انکدینگ است. در این متد تنها همان 5 کاراکتر اصلی تبدیل می‌شوند. متاسفانه متدی برای decode در این پلاگین وجود ندارد. پیاده‌سازی خلاصه‌شده این کتابخانه تنها برای نمایش نحوه عملکرد متد فوق به صورت زیر است:

```
var andExp = /&/g,
    htmlExp = [/(<|>|")/g, /(<|>|')/g, /(<|>|'|")/g],
    htmlCharMap = { '<': '&lt;', '>': '&gt;', "'": '&#039;', '"': '&quot;' },
    htmlReplace = function (all, $1) {
  return htmlCharMap[$1];
};
$.extend({
  // convert special html characters
  htmlspecialchars: function (string, quot) {
    return string.replace(andExp, '&amp;').replace(htmlExp[quot || 0], htmlReplace);
  }
});
```

نحوه استفاده از این متد هم به صورت زیر است:

```
$.htmlspecialchars("<div>");
```

.

$.string

: پلاگین دیگری برای jQuery که عملیات مربوط به رشته‌ها را دربر دارد. در این پلاگین برای عملیات انکدینگ دو متد escapeHTML و unescapeHTML به صورت زیر تعریف شده‌اند:

```
this.escapeHTML = function (s) {
  this.str = this.s(s)
      .split('&').join('&amp;')
      .split('<').join('&lt;')
      .split('>').join('&gt;');
  return this;
};

this.unescapeHTML = function (s) {
  this.str = this.stripTags(this.s(s)).str.replace(/&amp;/g, '&').replace(/&lt;/g,
'<').replace(/&gt;/g, '>');
  return this;
};
```

همان‌طور که می‌بنید در متد encode این پلاگین از یک روش جالب اما به نسبت ناکارآمد در رشته‌های طولانی، برای استخراج کاراکترهای غیرمجاز استفاده شده است. در این متدها هم تنها 3 کاراکتر & و < و > انکد و دیکد می‌شوند.

.

encoder.js

: کتابخانه نسبتا کاملی برای عملیات انکدینگ رشته‌ها در سمت کلاینت. این کتابخانه علاوه بر encode و decode رشته‌ها متدهایی برای تبدیل html entityها به فرمت عددی‌شان و برعکس، حذف کاراکترهای یونیکد، بررسی اینکه رشته ورودی شامل کاراکترهای انکد شده است، جلوگیری از انکدینگ مجدد یک رشته و ... نیز دارد.

. .

htmlEncode

: این متد پیاده‌سازی کاملی برای اجرای عملیات Html Encode دارد و محدوده وسیعی از کاراکترها را نیز تبدیل می‌کند. مشاهده عملیات موجود در این متد برای آشنایی با مطالب ظریف‌تر پیشنهاد می‌شود.

.

.

منابع برای مطالعه بیشتر

http://en.wikipedia.org/wiki/Character_encodings_in_HTML

http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references

http://stackoverflow.com/questions/1812473/difference-between-url-encode-and-html-encode

http://stackoverflow.com/questions/1219860/javascript-jquery-html-encoding

http://d802.nexlink.net/Gabriel/archive/2008/10/15/howto-html-encode-a-string-in-javascript.aspx

http://www.jardinesoftware.com/Documents/ASPNET_HTML_Encoding.pdf

http://www.west-wind.com/weblog/posts/2009/Feb/05/Html-and-Uri-String-Encoding-without-SystemWeb

http://blog.binarymist.net/tag/infosec

http://www.w3.org/TR/REC-html40/sgml/entities.html http://www.w3schools.com/html/html_entities.asp

http://www.w3schools.com/tags/ref_entities.asp

http://www.strictly-software.com/htmlencode

عنوان: **مدیریت حافظه در JavaScript**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۰۸ ۲۲:۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Memory Manegement, GC

مدیریت حافظه در JavaScript همانند مدل مدیریت حافظه در .NET می‌باشد. حافظه وقتی مورد نیاز است تخصیص پیدا می‌کند و وقتی دیگر مورد نیاز نیست آزاد می‌شود. این پروسه در CLR به نام جمع آوری زباله یا Garbage Collector یا GC مشهور است. تفاوت عمده فی مابین مدیریت حافظه در .NET با مدیریت حافظه در JavaScript این است که مدیریت حافظه در .NET توسط CLR واحد انجام می‌شود. یعنی پیاده سازی واحدی از GC وجود دارد و شما می‌توانید از نوع فعالیت آن اطمینان حاصل نمایید ولی در JavaScript با توجه به اینکه موتورهای اجرایی مختلفی برای اجرای آن وجود دارد، در سرورها و مرورگرهای مختلف پیاده سازی‌های متفاوتی برای آن وجود دارد. اطلاع از نحوه کار GC می‌تواند به درک ما از JavaScript کمک کرده تا بتوانیم کدهای بهتری در این زبان تولید کنیم.

در این مقاله به بررسی دو الگوریتم عمده GC در JavaScript می‌پردازیم.

1. مدل Reference Counting Garbage Collector

در این مدل از جمع آوری زباله، به ازای ایجاد هر آبجکت در حافظه و یا هر تخصیصی در حافظه، شمارشگری با عنوان reference counter در نظر گرفته می‌شود. هر زمان که به این آبجکت یا حافظه تخصیصی دسترسی ایجاد شود و یا reference داده شود، یک واحد به شمارشگر آن اضافه و هر وقت که رفرنس به حافظه یا آبجکت دیگر مورد استفاده نداشت یا از دسترس خارج شد، یک واحد از شمارشگر آن کاسته می‌شود. این مدل که سریعترین، ساده‌ترین و کم سربارترین مدل GC می‌باشد، وقتی شمارشگر رفرنس حافظه به صفر رسید، حافظه ومنابع سیستم تخصیصی به آن آبجکت آزاد شده و آماده استفاده مجدد می‌شودبه عنوان نمونه به کد زیر دقت کنید:

```
var object1='GC test object 1';
function Test1(){
   var object2='GC test object 2';
   alert (object1+'-' + object2);
}
alert (object1);
```

پس از اجرای این کد، جدولی مانند زیر در GC ایجاد می‌شود که به صورت زیر مقدار طی اجرای برنامه مقدار دهی می‌شود:

Reference Counter End Program	Reference Counter Line 6	Reference Counter Line 5	Reference Counter Line 3	Reference Counter Line 1	Object
0	1	1	2	1	object1
0	0	0	1	-	object2

همانطور که در جدول فوق مشخص است، وقتی از متقیر استفاده می‌شود، reference count آن زیاد و وقتی دیگر مورد استفاده ندارد یکی کم می‌شود. وقتی مقدار reference count به صفر برسد، متقیر از حافظه حذف شده و منابع سیستمی آزاد می‌شود. این مدل که در مرورگرهای قدیمی مورد استفاده قرار گرفته است، در صورتی که دو آبجکت به یکدیگر ارجاع داشته باشند، reference counter آن صفر نشده، حافظه و منابع تخصیصی آنها آزاد نمی‌شود و احتمال ایجاد نشت حافظه زیاد می‌شود.

2. مدل Mark-and-Sweep

در این مدل از مدیریت حافظه، برای آبجکت‌های ایجادی در حافظه،GC درخت ارجاعات ایجاد کرده و دقیقا مشخص می‌کند زمانی که یک آبجکت در دسترس نباشد و یا دیگر نیازی به آن نباشد، آن را از حافظه حذف می‌کند. مانند شکل زیر:

در این صورت هنگامی که آبجکت دیگر واقعا مورد نیاز نباشد از حافظه حذف می‌شود. یعنی اگر دو آبجکت به یکدیگر نیز ارجاع داشته باشند هنگامی که دیگر مورد استفاده قرار نگیرند حذف شده و امکان ایجاد نشت حافظه به حداقل می‌رسد.
تفاوت عمده بین GC در Javascript و GC در CLR این است که در زبان‌های مبتنی بر .NET شما می‌توانید به صورت مستقیم GC را صدا زده تا عمل جمع آوری زباله انجام پذیرد ولی در JavaScript هر زمان که نیاز به حافظه بیشتر باشد (و یا در یک زمانبندی مشخص) عمل جمع آوری زباله انجام شده و از طریق کد قابل فراخوانی نمی‌باشد.

عنوان: **آموزش TypeScript #1**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۵/۲۳ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, TypeScript, Client Side Framework

با گسترش روز افزون زبان برنامه نویسی Javascript و استفاده هر چه بیشتر آن در تولید برنامه‌های تحت وب این زبان به یکی از قدرت‌های بزرگ در تولید برنامه‌های مبتنی بر وب تبدیل شده است. ترکیب این زبان با Css و Html5 تقریبا هر گونه نیاز برای تهیه و توسعه برنامه‌های وب را حل کرده است. جاوا اسکریپت در ابتدا برای اسکریپت نویسی سمت کلاینت برای صفحات وب ایجاد شد و برای سال‌ها به‌عنوان ابزاری برای مدیریت کردن رویدادهای صفحات وب محدود شده بود و در نتیجه بسیاری از امکانات لازم برای برنامه‌نویسی برنامه‌های مقیاس بزرگ را به‌همراه نداشت. امروزه به قدری Javascript توسعه داده شده است که حتی در تولید برنامه‌های Native مانند Windows Store و برنامه‌های تحت Cloud نیز استفاده می‌شود. پیشرفت‌های صورت گرفته و اشاره شده در این حوزه موجب شد تا شاهد پیداش برنامه‌های مبتنی بر جاوا اسکریپت با سایزهای بی سابقه‌ای باشیم و این بیانگر این بود که تولید برنامه‌های مبتنی بر جاوا اسکریپت در مقیاس‌های بزرگ امر دشواری است و اینک TypeScript توسط غول نرم افزاری جهان پا به عرصه گذاشته که این فرآیند را آسان‌تر نماید. به کمک TypeScript می‌توان برنامه تحت JavaScript در مقیاس بزرگ تولید کرد به طوری با هر مرورگر و سیستم عاملی سازگار باشد. TypeScript از شی گرایی نیز پشتیبانی می‌کند و خروجی آن در نهایت به JavaScript کامپایل می‌شود. خیلی‌ها عقیده دارند که هدف اصلی مایکروسافت از تولید و توسعه این زبان رقابت با CoffeeScript است. CoffeeScript یک زبان متن باز است که در سال 2009 توسط Jeremy Ashkenas ایجاد شده است و سورس آن در GitHub موجود می‌باشد. در آینده، بیشتر به مباحث مربوط به CoffeeScript و آموزش آن خواهم پرداخت.

در تصویر ذیل یک مقایسه کوتاه بین CoffeeScript و TypeScript را مشاهده می‌کنید.

Feature	CoffeeScript	TypeScript
Compiles to JavaScript	●	●
Static Type Checking		●
Interfaces		●
Visual Studio Support (Web Essentials)	●	●
Intellisense		●
Loop Comprehensions	●	
Splats/RestParameters (...)	●	●
Classes	●	●
String Interpolations	●	
Proper Variable Hoisting	●	
Prevents use of ==	●	
Operator Goodness (?, < val <, etc)	●	
Write less code	●	
Stable	●	

با TypeScript چه چیزهایی به دست خواهیم آورد؟

یک نکته مهم این است که این زبان به خوبی در Visual Studio پشتیبانی می‌شود و قابلیت Intellisense نوشتن برنامه به این زبان را دلپذیرتر خواهد کرد و از طرفی دیگر به نظر من یکی از مهم‌ترین مزیت هایی که TypeScript در اختیار ما قرار می‌دهد این است که می‌توانیم به صورت Syntax آشنای شی گرایی کد نویسی کنیم و خیلی راحت‌تر کدهای خود را سازمان دهی کرده و از نوشتن کدهای تکراری اجتناب کنیم.

یکی دیگر از مزیت‌های مهم این زبان این است که این زبان از Static Typing به خوبی پشتیبانی می‌کند. این بدین معنی است که شما ابتدا باید متغیرها را تعریف کرده و نوع آن‌ها را مشخص نمایید و هم چنین در هنگام پاس دادن مقادیر به پارامترهای توابع باید حتما به نوع داده ای آن‌ها دقت داشته باشید چون کامپایلر بین انواع داده ای در TypeScript تمایز قایل است و در صورت رعایت نکردن این مورد شما با خطا مواجه خواهید شد. این تمایز قایل شدن باعث می‌شود که برنامه هایی خواناتر داشته باشیم از طرفی باعث می‌شود که خطا یابی و نوشتن تست برای برنامه راحت‌تر و تمیزتر باشد. بر خلاف JavaScript، در TypeScript(به دلیل پشتیبانی از شی گرایی) می‌توانیم علاوه بر داشتن کلاس، اینترفیس نیز داشته باشیم و در حال حاضر مزایای استفاده از اینترفیس بر کسی پوشیده نیست.

به دلیل اینکه کدهای TypeScript ابتدا کامپایل شده و بعد تبدیل به کدهای JavaScript می‌شوند در نتیجه قبل از رسیدن به مرحله اجرای پروژه، ما از خطاهای موجود در کد خود مطلع خواهیم شد.

البته این نکته را نیز فراموش نخواهیم کرد که این زبان تازه متولد شده است(سال 2012 توسط Anders Hejlsberg) و همچنان در حال توسعه است و این در حال حاضر مهم‌ترین عیب این زبان می‌تواند باشد چون هنوز به پختگی سایر زبان‌های اسکریپتی در نیامده است.

در ذیل یک مثال کوچک به زبان TypeScript و JavaScript را برای مقایسه در خوانایی و راحتی کد نویسی قرار دادم:

TypeScript:

```
class Greeter {
    greeting: string;

    constructor (message: string) {
        this.greeting = message;
    }

    greet() {
        return "Hello, " + this.greeting;
    }
}
```

بعد از کامپایل کد بالا به کدی معادل زیر در JavaScript تبدیل خواهد شد:

```
var Greeter = (function () {
    function Greeter(message) {
        this.greeting = message;
    }
    Greeter.prototype.greet = function () {
        return "Hello, " + this.greeting;
    };
    return Greeter;
})();
```

توضیح چند واژه در TypeScript

Program : یک برنامه TypeScript مجموعه ای از یک یا چند Source File است. این Source Fileها شامل کدهای پیاده سازی برنامه هستند ولی در خیلی موارد برای خوانایی بیشتر برنامه می‌توان فقط تعاریف را در این فایل‌های سورس قرار داد.
Module : ماژول در TypeScript شبیه به مفاهیم فضای نام یا namespace در دات نت است و می‌تواند شامل چندین کلاس یا اینترفیس باشد.
Class : مشابه به مفاهیم کلاس در دات نت است و دقیقا همان مفهوم را دارد. یک کلاس می‌تواند شامل چندین تابع و متغیر با سطوح دسترسی متفاوت باشد. در TypeScript مجاز به استفاده از کلمات کلیدی public و private نیز می‌باشید. یک کلاس در Typescript می‌تواند یک کلاس دیگر را توسعه دهد(ارث بری در دات نت) و چندین اینترفیس را پیاده سازی نماید.
Interface : یک اینترفیس فقط شامل تعاریف است و پیاده سازی در آن انجام نخواهد گرفت. یک اینترفیس می‌تواند چندین اینترفیس دیگر را توسعه دهد.
Function : معادل متد در دات نت است. می‌تواند پارامتر ورودی داشته باشد و در صورت نیاز یک مقدار را برگشت دهد.
Scope : دقیقا تمام مفاهیم مربوط به محدوده فضای نام و کلاس و متد در دات نت در این جا نیز صادق است.

آماده سازی Visual Studio برای شروع به کار

در ابتدا باید Template مربوطه به TypeScript را نصب کنید تا از طریق VS.Net بتوانیم به راحتی به این زبان کد نویسی کنیم. می‌توانید فایل نصب را از اینجا دانلود کنید. بعد از نصب از قسمت Templateهای موجود گزینه Html Application With TypeScript را انتخاب کنید

یا از قسمت Add در پروژه‌های وب خود نظیر MVC گزینه TypeScript File را انتخاب نمایید.

در پست بعدی کد نویسی با این زبان را آغاز خواهیم کرد.

نظرات خوانندگان

نویسنده: کامی
تاریخ: ۱۳۹۲/۰۵/۲۳ ۱۴:۲۷

باسلام
ممنون از مطالب مفیدتون
ایا می‌تونیم مثل جاوااسکریپت داخل صفحات html با استفاده از تگ script برنامه typescript بنویسیم

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۵/۲۳ ۱۶:۱

بله امکان پذیر است. اما با توجه به این نکته که فلسفه وجودی TypeScript این است که در پروژه هایی با مقیاس بزرگ برای سازمان دهی کدهای سمت کلاینت مورد استفاده قرار گیرند و یکی از روش‌های سازمان دهی کدها این است که کدهای TypeScript در فایل هایی جداگانه با پسوند ts ذخیره شده تا کامپایل و تبدیل به کد JavaScript شوند (مهم‌ترین مزیت این روش این است که از نوشتن کدهای تکراری جلوگیری می‌شود). اما در صورتی که مایل به نوشتن کد به صورت Embed در تگ Script هستید باید از پروژه‌های متن بازی همچون TypeScript Compile یا ts-htaccess استفاده کنید.

نویسنده: مصطفی عسگری
تاریخ: ۱۳۹۲/۰۵/۲۴ ۱۸:۴۳

با نگاهی به زبان TypeScript متوجه میشویم که خیلی Syntax روان و آسانی دارد.
سوالی که همیشه من داشته ام این است .... چرا خود زبان JavaScript را تغییر نمیدهند؟
مسلما TypeScript و CoffeeScript برای برطرف کردن ضعف JavaScript بوجود آمده اند اما چرا خود مشکل را برطرف نمیکنند؟
میتوانستند همانند ارائه HTML5 و CSS3 نسخه جدیدی از JavaScript ارائه کنند که سختی‌های کار با JavaScript را برطرف کرده باشند!

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۵/۲۵ ۱:۰

یکی از دلایل محبوبیت زبان JavaScript، راحتی در نوشتن کد با این زبان است. اگر قرار باشد این زبان یک محصول همه منظوره باشد به طور قطع دچار پیچیدگی‌های پیاده سازی شده و این همه محبوبیت به دست نمی‌آورد. هدف اولیه از تولید و توسعه زبان JavaScript، استفاده از آن در پروژه‌های سمت کلاینت بود. اما با مرور زمان و محبوبیت بیش از اندازه، توسعه گران مختلف تصمیم به توسعه این زبان گرفتند که هر محصول برای یک منظور خاص به وجود آمد. برای مثال Node.Js برای پروژه‌های RealTime استفاه می‌شود و بر مبنای منطق event-driven می‌باشد که خیلی‌ها از آن به عنوان Server side JavaScript یاد می‌کنند یا به عنوان مثال دیگر Dart محصول شرکت گوگل در سال 2011 (طراحی شده بر مبنای Scratch)و TypeScript محصول شرکت مایکروسافت در سال 2012 (طراحی شده بر مبنای JavaScript)عرضه شدند که هدف اصلی از تولید این زبان‌ها پشتیبانی از مبحث static typing و مباحث OOP برای پیاده سازی پروژه‌های در سطوحی با مقیاس بزرگ بود. JavaScript به عنوان زبان پایه باقی خواهد ماند و نسخه‌های مختلف در شکل سایر زبان‌ها و فریم ورک‌های مختلف عرضه می‌شوند تا هر کدام یک نیاز را برطرف سازند. البته در پایان این نکته را هم متذکر شوم که JavaScript هم روند با توسعه ECMAScript تغییر می‌کند. برای مثال در نسخه ECMAScript 6، امکان تعریف کلاس و ماژول در JavaScript فراهم شده است.

نویسنده: سالار
تاریخ: ۱۳۹۲/۰۵/۲۵ ۱۰:۴۰

با سلام.

- برای پروژه‌های بزرگ تحت وب که کدهای سمت کلاینت زیادی دارد استفاده از typescript را پیشنهاد میکنید؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۲۵ ۱۰:۵۰

Typescript - a real world story of adoption in TFS

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۵/۲۵ ۱۳:۱۱

از آن جا که این زبان syntax نزدیکی به زبان‌های دات نتی دارد و به خوبی در Vs.Net پشتیبانی می‌شود نه تنها گزینه مناسبی برای توسعه در پروژه‌های وب است بلکه در توسعه پروژه‌های Windows Store App نیز می‌تواند یکی از بهترین انتخاب‌ها باشد. در ضمن این زبان به صورت **پیش فرض** از ECMA Script 3 هنگام تبدیل کدها به زبان Javascript استفاده می‌کند و تقریبا با تمام مرورگرهای قدیمی و جدید سازگار است البته به راحتی امکان تغییر این option برای سازگاری کامپایلر TypeScript با ECMA Script 5 نیز وجود دارد.

عنوان: آموزش TypeScript #2
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۵/۲۶ ۸:۴۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, TypeScript, Client Side Framework

در این پست قصد دارم به بررسی چند نکته که از پیش نیازهای کار با TypeScript است بپردازم. همان طور که در پست قبلی مشاهده شد بعد از دانلود و نصب افزونه TypeScript در VS.Net یک Template به نام Html Application With TypeScript به Installed Template اضافه خواهد شد. بعد از انتخاب این قسمت شما به راحتی می‌توانید در هر فایل با پسوند ts کدهای مورد نظر به زبان TypeScript را نوشته و بعد از build پروژه این کدها تبدیل به کدهای JavaScript خواهند شد. بعد کافیست فایل مورد نظر را با استفاده از تک Script در فایل خود رفرنس دهید. دقت کنید که پسوند فایل حتما باید js باشد(به دلیل اینکه بعد از build پروژه فایل‌های ts تبدیل به js می‌شوند).
برای مثال:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>TypeScript HTML App</title>
    <link rel="stylesheet" href="app.css" type="text/css" />
    <script src="app.js"></script>
</head>
<body>
    <h1>Number Type in TypeScript</h1>
    <div id="content"/>
</body>
</html>
```

اما اگر یک پروژه وب نظیر Asp.Net MVC داشته باشیم و می‌خواهیم یک یا چند فایل که حاوی کدهای typeScript هستند را به این پروژه اضافه کرده و از آن‌ها در صفحات وب خود استفاده کنیم باید به این صورت عمل نمود:
بعد از اضافه کردن فایل‌های مورد نیاز، پروژه مورد نظر را Unload کنید. بعد به صورت زیر فایل پروژه(csproj) رو با یک ویرایشگر متنی باز کنید:

در این مرحله باید دو قسمت اضافه شود. یک بخش ItemGroup است که هر فایلی که در پروژه شما دارای پسوند ts است باید در این جا تعریف شود. در واقع این قسمت فایل هایی را که باید کامپایل شده تا در نهایت تبدیل به فایل‌های JavaScript شوند را مشخص می‌کند.
بخش دوم target است که مراحل Build پروژه را برای این فایل‌های مشخص شده تعیین می‌کند. برای مثال:

```
  <ItemGroup>
    <TypeScriptCompile Include="$(ProjectDir)\**\*.ts" />
  </ItemGroup>
  <Target Name="BeforeBuild">
    <Exec Command="&quot;$(PROGRAMFILES)\Microsoft SDKs\TypeScript\tsc&quot;
@(TypeScriptCompile ->'&quot;%(fullpath)&quot;', ' ')" />
  </Target>
```

همان طور که می‌بینید در قسمت ItemGroup تمام فایل‌های با پسوند ts در پروژه include شده اند. در قسمت target دستور کامپایل این فایل‌ها تعیین شد. اما نکته مهم این است که TypeScript به صورت پیش فرض از ECMAScript 3 در هنگام کامپایل کدها استفاده می‌کند.(ECMAScript 3 در سال 1999 منتشر شد و تقریبا با تمام مرورگرها سازگاری دارد اما از امکانات جدید در Javascript پشتیبانی نمی‌کند). اگر قصد دارید که از ECMAScript 5 در هنگام کامپایل کدها استفاده نمایید کافیست دستور زیر را اضافه نمایید:

```
--target ES5
```

مثال:

```
  <ItemGroup>
    <TypeScriptCompile Include="$(ProjectDir)\**\*.ts" />
  </ItemGroup>
  <Target Name="BeforeBuild">
    <Exec Command="&quot;$(PROGRAMFILES)\Microsoft SDKs\TypeScript\tsc&quot;
--target ES5 @(TypeScriptCompile ->'&quot;%(fullpath)&quot;', ' ')" />
  </Target>
```

اما به این نکته دقت داشته باشید که ECMAScript 5 در سال 2009 منتشر شده است در نتیجه فقط با مرورگرهای جدید سازگار خواهد بود و ممکن است کدهای شما در مرورگرهای قدیمی با مشکل مواجه شود.

مرورگرهایی که از ECMAScript 5 پشتیبانی می‌کنند عبارتند از:

IE 9 و نسخه‌های بعد از آن؛

FireFox 4 و نسخه‌های بعد از آن؛

Opera 12 و نسخه‌های بعد از آن؛

Safari 5.1 و نسخه‌های بعد از آن؛

Chrome 7 و نسخه‌های بعد از آن.

ادامه دارد...

نظرات خوانندگان

نویسنده: آریو
تاریخ: ۱۳۹۲/۱۱/۱۵ ۱۷:۶

سلام من زمانی که فایل پروژه رو ویرایش کردم به این مشکل برخوردم . امکانش هست بگید مشکل از کجاست :

```
Error  1  The command ""C:\Program Files (x86)\Microsoft SDKs\TypeScript\0.8.0.0\tsc" -target ES5 "
exited with code 3.  c:\users\IT\documents\visual studio
2012\Projects\MvcApplication6\MvcApplication6\MvcApplication6.csproj  259  5  MvcApplication6
```

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۱/۱۵ ۲۱:۲۹

پاسخ مورد نظر را اینجا می‌توانید مشاهده کنید

عنوان: آموزش TypeScript #3
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۵/۲۷ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, TypeScript, Client Side Framework

در این پست به تشریح انواع داده در زبان TypeScript و ذکر مثال در این زمینه می‌پردازیم.

تعریف متغیرها و انواع داده

در TypeScript هنگام تعریف متغیرها باید نوع داده ای آن‌ها را مشخص کنیم. در TypeScript پنج نوع داده ای وجود دارد که در زیر با ذکر مثال تعریف شده اند. مفاهیم ماژول، کلاس و تابع در پست بعدی به تشریح توضیح داده خواهند شد.

number : معادل نوع داده ای number در JavaScript است. برای ذخیره سازی اعداد صحیح و اعشاری استفاده می‌شود.
یک مثال:

```
class NumberTypeOfTypeScript {
    MyFunction()
    {
        var p: number;
        p = 1;
        var q = 2;
        var r = 3.33;
        alert("Value of P=" + p + "  Value of q=" + q + " Value of r=" + r);
    }
}

window.onload = () =>{
    var value = new NumberTypeOfTypeScript();
    value.MyFunction();
}
```

حال باید یک فایل Html برای استفاده از این کلاس داشته باشیم. به صورت زیر:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>TypeScript HTML App</title>
    <link rel="stylesheet" href="app.css" type="text/css" />
    <script src="app.js"></script>
</head>
<body>
    <h1>Number Type in TypeScript</h1>
    <div id="content"/>
</body>
</html>
```

بعد از اجرای پروژه خروجی به صورت زیر خواهد بود:

string : معادل نوع داده ای رشته ای است و برای ذخیره سازی مجموعه ای از کاراکترها از نوع UTF-16 استفاده می‌شود.

یک مثال:

```
class StringTypeOfTypeScript {
    Myfunction() {
      var  s: string;
      s="TypeScript"
      var empty = "";
      var abc = "abc";
      alert("Value of s="+ s+" Empty string="+ empty+" Value of abc ="+abc) ;
    }
}
window.onload = () =>{
    var value = new StringTypeOfTypeScript();
    value.Myfunction();
}
```

کد کامپایل شده و تبدیل آن به JavaScript:

```
var StringTypeOfTypeScript = (function () {
    function StringTypeOfTypeScript() { }
    StringTypeOfTypeScript.prototype.Myfunction = function () {
        var s;
        s = "TypeScript";
        var empty = "";
        var abc = "abc";
        alert("Value of s=" + s + " Empty string=" + empty + " Value of abc =" + abc);
    };
    return StringTypeOfTypeScript;
})();
window.onload = function () {
    var value = new StringTypeOfTypeScript();
    value.Myfunction();
};
```

خروجی به صورت زیر است:

boolean : برای ذخیره سازی مقادیر true یا false می‌باشد.

مثال:

```
class booleanTypeofTypeScript {
    MyFunction() {
        var lie: bool;
        lie = false;
        var a = 12;
        if (typeof (lie) == "boolean" && typeof (a) == "boolean") {
            alert("Both is boolean type");
        }

        if (typeof (lie) == "boolean" && typeof (a) != "boolean") {
            alert("lie is boolean type and a is not!")
        }
        else {
            alert("a is boolean type and lie is not!");
        }
    }
}

window.onload =()=> {
    var access = new booleanTypeofTypeScript();
    access.MyFunction();
 }
```

کد کامپایل شده و تبدیل آن به JavaScript:

```
var booleanTypeofTypeScript = (function () {
    function booleanTypeofTypeScript() { }
    booleanTypeofTypeScript.prototype.MyFunction = function () {
        var lie;
        lie = false;
        var a = 12;
        if(typeof (lie) == "boolean" && typeof (a) == "boolean") {
            alert("Both is boolean type");
        }
        if(typeof (lie) == "boolean" && typeof (a) != "boolean") {
            alert("lie is boolean type and a is not!");
        } else {
            alert("a is boolean type and lie is not!");
        }
    };
```

```
    return booleanTypeofTypeScript;
})();
window.onload = function () {
    var access = new booleanTypeofTypeScript();
    access.MyFunction();
};
```

null : همانند دات نت هنگامی که قصد داشته باشیم مقدار یک متغیر را null اختصاص دهیم از این کلمه کلیدی استفاده می‌کنیم.
مثال:

```
class NullTypeinTypeScript {
    MyFunction() {
        var p: number = null;
        var x = null;
        if (p== null) {
            alert("p has null value!");
        }
        else { alert("p has a value"); }
    }
}
window.onload = () =>{
    var value = new NullTypeinTypeScript();
    value.MyFunction();
}
```

کد کامپایل شده و تبدیل آن به JavaScript:

```
var NullTypeinTypeScript = (function () {
    function NullTypeinTypeScript() { }
    NullTypeinTypeScript.prototype.MyFunction = function () {
        var p = null;
        var x = null;
        if(p == null) {
            alert("p has null value!");
        } else {
            alert("p has a value");
        }
    };
    return NullTypeinTypeScript;
})();
window.onload = function () {
    var value = new NullTypeinTypeScript();
    value.MyFunction();
};
```

undefined: معادل نوع undefined در Javascript است. اگر به یک متغیر مقدار اختصاص ندهید مقدار آن undefined خواهد بود.

مثال:

```
class UndefinedTypeOfTypeScript {
    Myfunction() {
        var p: number;
        var x = undefined;
        if (p == undefined && x == undefined) {
            alert("p and x is undefined");
        }
        else { alert("p and c cannot undefined"); }
    }
}
window.onload = () =>{
    var value = new UndefinedTypeOfTypeScript();
    value.Myfunction();
}
```

کد کامپایل شده و تبدیل آن به JavaScript:

```
var UndefinedTypeOfTypeScript = (function () {
    function UndefinedTypeOfTypeScript() { }
    UndefinedTypeOfTypeScript.prototype.Myfunction = function () {
        var p;
        var x = undefined;
        if(p == undefined && x == undefined) {
            alert("p and x is undefined");
        } else {
            alert("p and c cannot undefined");
        }
    };
    return UndefinedTypeOfTypeScript;
})();
window.onload = function () {
    var value = new UndefinedTypeOfTypeScript();
    value.Myfunction();
};
```

خروجی این مثال نیز به صورت زیر است:

ادامه دارد...

عنوان: آموزش TypeScript #4
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۵/۲۸ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, TypeScript, Client Side Framework

در پست‌های قبل با کلیات و primitive types در زبان TypeScript آشنا شدیم:

کلیات TypeScript

انواع داده ای اولیه در TypeScript

در این پست به مفاهیم شی گرایی در این زبان می‌پردازیم.

ماژول ها:

تعریف یک ماژول: برای تعریف یک ماژول باید از کلمه کلیدی module استفاده کنید. یک ماژول معادل یک ظرف است برای نگهداری کلاس‌ها و اینترفیس‌ها و سایر ماژول ها. کلاس‌ها و اینترفیس‌ها در TypeScript می‌توانند به صورت internal یا public باشند(به صورت پیش فرض internal است؛ یعنی فقط در همان ماژول قابل استفاده و فراخوانی است). هر چیزی که در داخل یک ماژول تعریف می‌شود محدوده آن در داخل آن ماژول خواهد بود. اگر قصد توسعه یک پروژه در مقیاس بزرگ را دارید می‌توانید همانند دات نت که در آن امکان تعریف فضای نام‌های تودرتو امکان پذیر است در TypeScript نیز، ماژول‌های تودرتو تعریف کنید. برای مثال:

```
module MyModule1 {
    module  MyModule2 {
     }
}
```

اما به صورت معمول سعی می‌شود هر ماژول در یک فایل جداگانه تعریف شود. استفاده از چند ماژول در یک فایل به مرور، درک پروژه را سخت خواهد کرد و در هنگام توسعه امکان برخورد با مشکل وجود خواهد داشت. برای مثال اگر یک فایل به نام MyModule.ts داشته باشیم که یک ماژول به این نام را شامل شود بعد از کامپایل یک فایل به نام MyModule.js ایجاد خواهد شد.

کلاس ها:

برای تعریف یک کلاس می‌توانیم همانند دات نت از کلمه کلیدی class استفاده کنیم. بعد از تعریف کلاس می‌توانیم متغیرها و توابع مورد نظر را در این کلاس قرار داده و تعریف کنیم.

```
module Utilities {
   export class Logger {
      log(message: string): void{
       if(typeofwindow.console !== 'undefined') {
           window.console.log(message);
        }
      }
   }
}
```

نکته مهم و جالب قسمت بالا کلمه export است. export معادل public در دات نت است و کلاس logger را قابل دسترس در خارج ماژول Utilities خواهد کرد. اگر از export در هنگام تعریف کلاس استفاده نکنیم این کلاس فقط در سایر کلاس‌های تعریف شده در داخل همان ماژول قابل دسترس است.

تابع log که در کلاس بالا تعریف کردیم به صورت پیش فرض public یا عمومی است و نیاز به استفاده export نیست.

برای استفاده از کلاس بالا باید این کلمه کلیدی new استفاده کنیم.

```
window.onload = function() {
  varlogger = new Utilities.Logger();
  logger.log('Logger is loaded');
};
```

برای تعریف سازنده برای کلاس بالا باید از کلمه کلیدی constructor استفاده نماییم:

```
export class Logger{
constructor(private num: number) {
}
```

با کمی دقت متوجه تعریف متغیر num به صورت private خواهید شد که برخلاف انتظار ما در زبان‌های دات نتی است. بر خلاف دات نت در زبان TypeScript، دسترسی به متغیر تعریف شده در سازنده با کمک اشاره گر this در هر جای کلاس ممکن می‌باشد. در نتیجه نیازی به تعریف متغیر جدید و پاس دادن مقادیر این متغیرها به این فیلدها نمی‌باشد.
اگر به تابع log دقت کنید خواهید دید که یک پارامتر ورودی به نام message دارد که نوع آن string است. در ضمن Typescript از پارامترهای اختیاری(پارامتر با مقدار پیش فرض) نیز پشتیبانی می‌کند. مثال:

```
pad(num: number, len: number= 2, char: string= '0')
```

استفاده از پارامترهای Rest

منظور از پارامترهای Rest یعنی در هنگام فراخوانی توابع محدودیتی برای تعداد پارامترها نیست که معادل params در دات نت است. برای تعریف این گونه پارامترهاکافیست به جای params از ... استفاده نماییم.

```
function addManyNumbers(...numbers: number[]) {
  var sum = 0;
  for(var i = 0; i < numbers.length; i++) {
    sum += numbers[i];
 }
  returnsum;
}
var result = addManyNumbers(1,2,3,5,6,7,8,9);
```

تعریف توابع خصوصی

در TypeScript امکان توابع خصوصی با کلمه کلیدی private امکان پذیر است. همانند دات نت با استفاده از کلمه کلیدی private می‌توانیم کلاسی تعریف کنیم که فقط برای همان کلاس قابل دسترس باشد(به صورت پیش فرض توابع به صورت عمومی هستند).

```
module Utilities {
    Export class Logger {
     log(message: string): void{
                if(typeofwindow.console !== 'undefined') {
                   window.console.log(this.getTimeStamp() + ' -'+ message);
                   window.console.log(this.getTimeStamp() + ' -'+ message);
                }
        }
  private getTimeStamp(): string{
      var now = newDate();
      return now.getHours() + ':'+
      now.getMinutes() + ':'+
      now.getSeconds() + ':'+
      now.getMilliseconds();
  }
 }
}
```

از آن جا که تابع getTimeStamp به صورت خصوصی تعریف شده است در نتیجه امکان استفاده از آن در خارج کلاس وجود ندارد. اگر سعی بر استفاده این تابع داشته باشیم توسط کامپایلر با یک warning مواجه خواهیم شد.

```
window.onload = function () {
    var logger = new Utilities.Logger();
    logger.getTimeStamp();
};
```

یک نکته مهم این است که کلمه private فقط برای توابع و متغیرها قابل استفاده است.

تعریف توابع static:

در TypeScript امکان تعریف توابع static وجود دارد. همانند دات نت باید از کلمه کلیدی static استفاده کنیم.

```
classFormatter {
static pad(num: number, len: number, char: string): string{
      var output = num.toString();
         while(output.length < len) {
         output = char + output;
      }
   returnoutput;
   }
  }
}
```

و استفاده از این تابع بدون وهله سازی از کلاس :

```
Formatter.pad(now.getSeconds(), 2, '0') +
```

Function Overload

همان گونه که در دات نت امکان overload کردن توابع میسر است در TypeScript هم این امکان وجود دارد.

```
static pad(num: number, len?: number, char?: string);
static pad(num: string, len?: number, char?: string);
static pad(num: any, len: number= 2, char: string= '0') {
 var output = num.toString();
 while(output.length < len) {
 output = char + output;
 }
 returnoutput;
}
```

ادامه دارد...

عنوان: آموزش TypeScript #5
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۵/۳۰ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, TypeScript, Client Side Framework

در ادامه مباحث شی گرایی در TypeScript قصد داریم به مباحث مربوط به interface و طریقه استفاده از آن بپردازیم. همانند زبان‌های دات نتی در TypeScript نیز به راحتی می‌توانید interface تعریف کنید. در یک اینترفیس اجازه پیاده سازی هیچ تابعی را ندارید و فقط باید عنوان و پارامترهای ورودی و نوع خروجی آن را تعیین کنید. برای تعریف اینترفیس از کلمه کلیدی interface به صورت زیر استفاده خواهیم کرد.

```
export interface ILogger {
  log(message: string): void;
}
```

همان طور در پست‌های قبلی مشاهده شد از کلمه کلیدی export برای عمومی کردن اینترفیس استفاده می‌کنیم. یعنی این اینترفیس از بیرون ماژول خود نیز قابل دسترسی است.
حال نیاز به کلاسی داریم که این اینترفیس را پیاده سازی کند. این پیاده سازی به صورت زیر انجام می‌گیرد:

```
export class Logger implements ILogger
{
}
```

یا:

```
export class AnnoyingLogger implements ILogger {
   log(message: string): void{
         alert(message);
     }
}
```

همانند دات نت یک کلاس می‌تواند چندین اینترفیس را پیاده سازی کند.(اصطلاحا به این روش explicit implementation یا پیاده سازی صریح می‌گویند)

```
export class MyClass implements IFirstInterface, ISecondInterface
{

}
```

*یکی از قابلیت جالب و کارآمد زبان TypeScript این است که در هنگام کار با اینترفیس‌ها حتما نیازی به پیاده سازی صریح نیست. اگر یک object تمام متغیرها و توابع مورد نیاز یک اینترفیس را پیاده سازی کند به راحتی همانند روش explicit emplementation می‌توان از آن object استفاده کرد. به این قابلیت **Duck Typing** می‌گویند. مثال:

```
IPerson {
   firstName: string;
   lastName: string;
}
class Person implements IPerson {
  constructor(public firstName: string, public lastName: string) {
  }
}

varpersonA: IPerson = newPerson('Masoud', 'Pakdel'); //expilict
varpersonB: IPerson = { firstName: 'Ashkan', lastName: 'Shahram'}; // duck typing
```

همان طور که می‌بینید object دوم به نام personB تمام متغیرها ی مورد نیاز اینترفیس IPerson را پیاده سازی کرده است در

نتییجه کامپایلر همان رفتاری را که با object اول به نام personA دارد را با آن نیز خواهد داشت.

پیاده سازی چند اینترفیس به صورت همزمان

همانند دات نت که یک کلاس فقط می‌تواند از یک کلاس ارث ببرد ولی می‌تواند n تا اینترفیس را پیاده سازی کند در TypeScript نیز چنین قوانینی وجود دارد. یعنی یک اینترفیس می‌تواند چندین اینترفیس دیگر را توسعه دهد(extend) و کلاسی که این اینترفیس را پیاده سازی می‌کند باید تمام توابع اینترفیس‌ها را پیاده سازی کند. مثال:

```
interface IMover {
 move() : void;
}

interface IShaker {
 shake() : void;
}

interface IMoverShaker extends IMover, IShaker {
}
class MoverShaker implements IMoverShaker {
 move() {
 }
 shake() {
 }
}
```

*به کلمات کلیدی extends و implements و طریقه به کار گیری آن‌ها دقت کنید.

instanceof

از instanceof زمانی استفاده می‌کنیم که قصد داشته باشیم که یک instance را با یک نوع مشخص مقایسه کنیم. اگر instance مربوطه از نوع مشخص باشد یا از این نوع ارث برده باشد مقدار true برگشت داده می‌شود در غیر این صورت مقدار false خواهد بود.

یک مثال:

```
var isLogger = logger instanceof Utilities.Logger;
var isLogger = logger instanceof Utilities.AnnoyingLogger;
var isLogger = logger instanceof Utilities.Formatter;
```

Method overriding

در TypeScript می‌توان مانند زبان‌های شی گرای دیگر Method overriding را پیاده سازی کرد. یعنی می‌توان متدهای کلاس پایه را در کلاس مشتق شده تحریف کرد. با یک مثال به شرح این مورد خواهم پرداخت.

فرض کنید یک کلاس پایه به صورت زیر داریم:

```
class BaseEmployee
{
    constructor (public fname: string,public lname: string)
    {
    }
    sayInfo()
    {
       alert('this is base class method');
    }
}
```

کلاس دیگری به نام Employee می‌سازیم که کلاس بالا را توسعه می‌دهد(یا به اصطلاح از کلاس بالا ارث می‌برد).

```
class Employee extends BaseEmployee
{
   sayInfo()
     {
        alert('this is derived class method');
     }
}
```

```
window.onload = () =>
{
    var first: BaseEmployee= new Employee();
    first.sayInfo();
    var second: BaseEmployee = new BaseEmployee();
    second.sayInfo();
}
```

نکته مهم این است که دیگر خبری از کلمه کلیدی virtual برای مشخص کردن توابعی که قصد overriding آن‌ها را داریم نیست. تمام توابع که عمومی هستند را می‌توان override کرد.

*اگر در کلاس مشتق شده قصد داشته باشیم که به توابع و فیلدهای کلاس پایه اشاره کنیم باید از کلمه کلیدی super استفاده کنیم.(معادل base در #C).

مثال:

```
class Animal {
    constructor (public name: string) {
    }
}

class Dog extends Animal {
      constructor (public name: string, public age:number)
      {
        super(name);
      }

    sayHello() {


      alert(super.name);


 }
}
```

اگر به سازنده کلاس مشتق شده دقت کنید خواهید دید که پارامتر name را به سازنده کلاس پایه پاس دادیم: کد معادل در #C به صورت زیر است:

```
public class Dog : Animal
{
      public Dog (string name, int age):base(name)
      {
      }
}
```

در تابع sayHello نیز با استفاده از کلمه کلیدی super به فیلد name در کلاس پایه دسترسی خواهیم داشت.

*دقت کنید که مباحث مربوط به interface و private modifier و Type safety که پیش‌تر در مورد آن‌ها بحث شد، فقط در فایل‌های TypeScript و در هنگام کد نویسی و طراحی معنی دار هستند، زیرا بعد از کامپایل فایل‌های ts این مفاهیم در Javascript پشتیبانی نمی‌شوند در نتیجه هیچ مورد استفاده هم نخواهد داشت.

ادامه دارد...

عنوان: آموزش Knockout.Js #1
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۰۵ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, KnockoutJS, JavaScriptFrameWork

اگر از برنامه نویس‌های پروژه‌های WPF درباره ویژگی‌های مهم الگوی MVVM بپرسید به احتمال زیاد اولین مطلبی که عنوان می‌شود این است که هنگام کار با الگوی MVVM در WPF باید از مباحث data-binding استفاده شود. به صورت خلاصه، data-binding مکانیزمی است که عناصر موجود در Xaml را به آبجکت‌های موجود در ViewModel یا سایر عناصر Xaml مقید می‌کند به طوری که با تغییر مقدار در آبجکت‌های ViewModel، عناصر View نیز خود را به روز می‌کنند یا با تغییر در مقادیر عناصر Xaml، آبجکت‌های متناظر در ViewModel نیز تغییر خواهند کرد(در صورت تنظیم Mode = TwoWay).

Knockout.Js چیست؟

در یک جمله Knockout.Js یک فریم ورک جاوا اسکریپ است که امکان پیاده سازی الگوی MVVM و مکانیزم data-binding را در پروژه‌های تحت وب به راحتی میسر می‌کند. به عبارت دیگر عناصر DOM را به data-model و آبجکت‌های data-model را به عناصر DOM مقید می‌کند، به طوری که با هر تغییر در مقدار یا وضعیت این عناصر یا آبجکت ها، تغییرات به موارد مقید شده نیز اعمال می‌گردد. به تصاویر زیر دقت کنید!

به روز رسانی data-model **بدون** استفاده از KO

به روز رسانی data-model **با** استفاده از KO

ویژگی‌های مهم KO

»ارائه یک راه حل بسیار ساده و واضح برای اتصال بخش‌های مختلف UI به data-model

»به روز رسانی خودکار عناصر و بخش‌های مختلف UI بر اساس تغییرات صورت گرفته در data-model

»به صورت کامل با کتابخانه و توابع javascript پیاده سازی شده است.

»حجم بسیار کم(سیزده کیلو بایت) بعد از فشرده سازی

»سازگار با تمام مروگرهای جدید(... IE 6+, Firefox 2+, Chrome, Safari,)

»امکان استفاده راحت بدون اعمال تغییرات اساسی در معماری پروژه هایی که در فاز توسعه هستند و بخشی از مسیر توسعه را طی کرده اند

»و...

آیا KO برای تکمیل JQuery در نظر گرفته شده است یا جایگزین؟

در اینکه JQuery بسیار محبوب است و در اکثر پروژه‌های تحت وب مورد استفاده است شکی وجود ندارد ولی این بدان معنی

نیست که با توجه به وجود JQuery و محبوبیت آن دیگر نیازی به KO احساس نمی‌شود. به عنوان یک مثال ساده : فرض کنید در یک قسمت از پروژه قصد داریم یک لیست از داده‌ها را نمایش دهیم. در پایین لیست تعداد آیتم‌های موجود در لیست مورد نظر نمایش داده میشود. یک دکمه Add داریم که امکان اضافه شدن آیتم جدید را در اختیار ما قرار می‌دهد. بعد از اضافه شدن یک مقدار، باید عددی که تعداد آیتم‌های لیست را نمایش می‌دهد به روز کنیم. خب اگر قصد داشته باشیم این کار را با JQuery انجام دهیم راه حل‌های زیر پیش رو است :

» به دست آوردن تعداد trهای جدول موجود؛

»به دست آوردن تعداد divهای موجود با استفاده از یک کلاس مشخص css؛

» یا حتی به دست آوردن تعداد آیتم‌های نمایشی در span هایی مشخص.

و البته سایر راه حل ها...

حال فرض کنید دکمه‌های دیگر نظیر Delete نیز مد نظر باشد که مراحل بالا تکرار خواهند شد. اما با استفاده از KO به راحتی می‌توانیم تعداد آیتم‌های موجود در یک آرایه را به یک عنصر مشخص bind کنیم به طور با هر تغییر در این مقدار، عنصر مورد نظر نیز به روز می‌شود یا به بیانی دیگر همواره تغییرات observe خواهند شد. برای مثال:

```
Number of items :<span data-bind="text: myList().count"></span>
```

در نتیجه برای کار با KO وابستگی مستقیم به استفاده از JQeury وجود ندارد ولی این امکان هست که بتوانیم هم از JQuery و هم از KO در کنار هم به راحتی استفاده کنیم و از قدرت‌های هر دو فریم ورک بهره ببریم و البته KO جایگزینی برای JQuery نخواهد بود.

در پست بعد، شروع به کار با KO آموزش داده خواهد شد.

ادامه دارد...

نظرات خوانندگان

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۳:۱۴

سلام و تشکر از آموزش خوبتون
KNOCKOUT در مقایسه با angular ، کدام مناسب‌تر هستند؟
آیا مقایسه این دو درست است؟
شنیدم روی ویژوال استودیو 2013 مایکروسافت پیش فرض آنگولار رو استفاده کرده. این خودش خیلی نقطه قوت هست و حتما روش فکر کرده مایکروسافت!
نظر شما چیه؟
به طور کل برای spa چه مجموعه فریم ورکی رو پیشنهاد میدید؟
مثلا ترکیب jquery + angular +requirejs چطوره؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۳:۴۲

در این مطلب فقط بوت استرپ نگارش 2 در MVC 5 پیش فرض شده. قبلا ناک آوت در MVC4 بود جزو اسکریپت‌های پیش فرض.

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۷/۰۶ ۱۴:۱۷

این سوال خوبی است. اما گمان نمی‌کنم بشود پاسخ دقیقی به بخش کدام مناسب‌تر است به طور کلی داد. شاید بتوانید بر اساس علاقه به MVC یا MVVM یکی را برگزینید. برای کسانی هم که می‌خواهند یکی را شروع کنند شاید Knockout برای شروع با توجه به داکیومنت و بخش آموزش جالب آن بهتر باشد. همچنین مقایسه هایی مانند این یا بحث هایی مانند این کمک کننده است برای انتخاب بین این فریم ورک ها. البته هر دو فریم ورک مدرن و مناسب برای بسیاری موارد هستند. نظر شخصی من این است اگر ASP.NET MVC کار می‌کنید Angular را به صورت راه حل کلی دنبال کنید چون کمی کسب مهارت و آشنایی با تمام مفاهیم آن نسبت به Knockout بیشتر طول می‌کشد. و زمانی که صرف یافتن گزینه بهتر بین این دو می‌کنید را برای مطالعه Knockout با استفاده از مقالات همین سایت یا بخش آموزش سایت رسمی آن اختصاص دهید. گمان نمی‌کنم از صرف وقت برای این دو پشیمان شوید. هر کدام شیرینی خاص خود را دارند.

عنوان: آموزش Knockout.Js #2
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۰۶ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, KnockoutJS, JavaScriptFrameWork

در پست قبلی با مفاهیم و ویژگی‌های کلی KO آشنا شدید. KO از الگوی طراحی MVVM استفاده می‌کند. از آن جا که یکی از پیش نیازهای KO آشنایی اولیه با مفاهیم View و Model است نیاز به توضیح در این موارد نیست اما اگر به هر دلیلی با این مفاهیم آشنایی ندارید می‌توانید از اینجا شروع کنید. اما درباره ViewModel که کمی مفهوم متفاوتی دارد، این نکته قابل ذکر است که KO از ViewModel برای ارتباط مستقیم بین View و Model استفاده می‌کند، چیزی شبیه به منطق MVC با این تفاوت که ViewModel به جای Controller قرار خواهد گرفت.

ابتدا باید به شرح برخی مفاهیم در KO بپردازیم:

«Observable(قابل مشاهده کردن تغییرات)

KO از Observable برای ردیابی و مشاهده تغییرات خواص ViewModel استفاده می‌کند. در واقع Observable دقیقا شبیه به متغیرها در JavaScript عمل می‌کنند با این تفاوت که به KO اجازه می‌دهند که تغییرات این خواص را پیگیری کند و این تغییرات را به بخش‌های مرتبط View اعمال نماید. اما سوال این است که KO چگونه متوجه می‌شود که این تغییرات بر کدام قسمت در View تاثیر خواهند داشت؟ جواب این سوال در مفهوم Binding است.

«Binding

برای اتصال بخش‌های مختلف View به Observableها باید از binding(مقید سازی) استفاده کنیم. بدون عملیات binding، امکان اعمال تغییرات Observableها بر روی عناصر HTML امکان پذیر نیست.

برای مثال در شکل زیر یکی از خواص ViewModel را به View متناظر مقید شده است.

با کمی دقت در شکل بالا این نکته به دست می‌آید که می‌توان در یک ViewModel، فقط خواص مورد نظر را به عناصر Html مقید کرد .

دانلود فایل‌های مورد نیاز

فایل‌های مورد نیاز برای KO رو می‌توانید از اینجا دانلود نمایید و به پروژه اضافه کنید. به صورت پیش فرض فایل‌های مورد نیاز KO، در پروژه‌های MVC 4 وجود دارد و نیاز به دانلود آن‌ها نیست و شما باید فقط مراحل BundleConfig را انجام دهید.

تعریف ViewModel

برای تعریف ViewModel و پیاده سازی مراحل Observable و binding باید به صورت زیر عمل نمایید:

```
<html lang='en'>
<head>
<title>Hello, Knockout.js</title>
<meta charset='utf-8' />
<link rel='stylesheet' href='style.css' />
</head>
<body>
<h1>Hello, Knockout.js</h1>
<script type='text/javascript' src='knockout-2.1.0.js'>
      <script type='text/javascript'>
             var personViewModel = {
                  firstName: "Masoud",
                  lastName: "Pakdel"
                };
                 ko.applyBindings(personViewModel);
       </script>
</script>
</body>
</html>
```

مشاهده می‌کنید که ابتدا یک ViewModel به نام person ایجاد کردم همراه با دو خاصیت به نام‌های firstName و lastName. تابع applyBinding برای KO بدین معنی است که این آبجکت به عنوان یک ViewModel در این صفحه مورد استفاده قرار خواهد گرفت. اما برای مشاهده تغییرات باید یک عنصر HTML را به این ViewModel مقید(bind) کنیم.

مقید سازی عناصر HTML

برای مقید سازی عناصر HTML به ViewModelها باید از data-bind attribute استفاده نماییم. برای مثال:

```
<p><span data-bind='text: firstName'></span>'s Shopping Cart</p>
```

اگر به data-bind در تگ span بالا توجه کنید خواهید دید که مقدار text در این تگ را به خاصیت firstName در viewModel این صفحه bind شده است. تا اینجا KO می‌داند که چه عنصر از DOM به کدام خاصیت از ViewModel مقید شده است اما هنوز دستور ردیابی تغییرات(Observable) را برای KO تعیین نکردیم.

چگونه خواص را Observable کنیم

در پروژه‌های WPF، فقط در صورتی تغییرات خواص یک کلاس ردیابی می‌شوند که اولا کلاس اینترفیس INotifyPropertyChanged را پیاده سازی کرده باشد ثانیا، در متد set این خواص، متد OnPropertyChanged(البته این متد می‌تواند هر نام دیگری نیز داشته باشد) صدا زده شده باشد. نکته مهم و اساسی در KO نیز همین است که برای اینکه KO بتواند تغییرات هر خاصیت را مشاهده کند حتما خواص مورد نظر باید Observable شوند. برای این کار کافیست به صورت عمل کنید:

```
var personViewModel = {
  firstName: ko.observable("Masoud"),
  lastName: ko.observable("Pakdel")
};
```

مزیت اصلی برای اینکه حتما خواص مورد نظرتان Observable شوند این است که، در صورتی که مایل نباشید تغییرات یک خاصیت بر روی View اعمال شود کافیست از دستور بالا استفاده نکنید. درست مثل اینکه هرگز مقدار آن تغییر نکرده است.

پیاده سازی متدهای get و set

همان طور که متوجه شدید، Observableها متغیر نیستند بلکه تابع هستند در نتیجه برای دستیابی به مقدار یک observable کافیست آن را بدون پارامتر ورودی صدا بزنیم و برای تغییر در مقدار آن باید همان تابع را با مقدار جدید صدا بزنیم. برای مثال:

```
personViewModel.firstName() // Get
personViewModel.firstName("Masoud") // Set
```

البته این نکته را هم متذکر شوم که در ViewModelهای خود می‌توانید توابع سفارشی مورد نیاز را بنویسید و از آن‌ها در جای مناسب استفاده نماید(شبیه به مفاهیم Commandها در WPF)

مقید سازی تعاملی

اگر با WPF آشنایی دارید می‌دانید که در این گونه پروژه‌ها می‌توان رویدادهای مورد نظر را به Commandهای خاص در ViewModel مقید کرد. در KO نیز این امر به آسانی امکان پذیر است که به آن Interactive Bindings می‌گویند. فقط کافیست در data-bind attribute از نام رویداد استفاده نماییم. مثال:

ایتدا یک ViewModel به صورت زیر خواهیم داشت:

```
function PersonViewModel() {
   this.firstName = ko.observable("Masoud");
   this.lastName = ko.observable("Pakdel");
   this.clickMe= function() {
    alert("this is test!");
  };
};
```

تنها نکته قابل ذکر تعریف تابع سفارشی به نام clickMe است که به نوعی معادل Command مورد نظر ما در WPF است. در عنصر HTML مورد نظر که در این جا button است باید data-binding به صورت زیر باشد:

```
<button data-bind='click: clickMe'>Click Me...</button>
```

در نتیجه بعد از کلیک بر روی button بالا تابع مورد نظر در viewModel اجرا خواهد شد.

پس به صورت خلاصه:

ابتدا ViewModel مورد نظر را ایجاد نمایید؛
سپس با استفاده از data-bind عملیات مقید سازی بین View و ViewModel را انجام دهید
در نهایت با استفاده از Obsevable تغییرات خواص مورد نظر را ردیابی نمایید.

ادامه دارد...

نظرات خوانندگان

نویسنده: نوید
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۳:۲۳

اگر امکان داره کدهای مثالهای مربوطه رو هم بذارید . امیدوارم این سری آموزش رو ادامه بدین . با تشکر

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۳:۴۲

در پست‌های بعدی که مفاهیم مهم و اصلی Knockout رو بررسی می‌کنیم حتما مثال‌های مربوطه قرار داده می‌شوند.

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۶:۱۱

سلام
اگر از فریم ورک‌های KnockoutJS و یا AngularJS استفاده کنم. ایا نیاز هست که JQuery را نیز ضمیمه کنم و یا دیگر به JQuery نیازی نیست؟
آیا کارهایی که JQuery انجام میدهد را این دو فریم ورک و یا کلا فریم ورک‌های دیگر می‌توانند انجام دهند؟

تشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۶:۴۳

Knockout.js جایگزین jQuery یا MooTools نیست. در این کتابخانه animation یا مدیریت عمومی رخدادها، ساده سازی Ajax و مانند آن پیاده سازی نشده‌اند (هرچند Knockout.js امکان parse اطلاعات Ajax ایی دریافتی را دارد). هدف از Knockout.js ارائه مکملی برای سایر فناوری‌های وب جهت تولید برنامه‌های غنی و *دسکتاپ مانند* وب است. پشتیبانی خوبی از آن توسط مایکروسافت صورت می‌گیره چون نویسنده‌اش عضو تیم ASP.NET MVC است.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۲/۰۶/۱۸ ۱۷:۵۲

سلام تشکر بابت مطالب ارزشمندی که گذاشتید
ایا میشه در view از چند view model استفاده کرد ؟
اگر میشه چطور باید در هنگام bind کردن به html صفحه از هم تفکیک کرد

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۱۸ ۲۲:۱۵

ممنون دوست عزیز.
بله امکان پذیر است. باید از المان‌های تودرتو استفاده کنیم. به این صورت که المان ریشه با استفاده از with به model مربوطه مقید می‌شود و المان‌های داخلی به خواص مدل bind می‌شوند. برای مثال:

```
<div data-bind="text: teacher"> </div>//مقید سازی به مدل اول
<p data-bind="with: student">//مقید سازی المان ریشه به مدل دوم
    Name: <span data-bind="text: name"> </span>,//مقید سازی خواص به المان‌های داخلی
    Family: <span data-bind="text: family"> </span>
</p>

<script type="text/javascript">
    ko.applyBindings({
        teacher: "myTeacher",
        student: {
```

```
            name:  "Masoud",
            family: "Pakdel"
        }
    });
</script>
```

عنوان: **آشنایی با JS Link در شیرپوینت 2013**
نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۲/۰۶/۲۱ ۱۷:۱۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, SharePoint 2013

شیرپوینت 2013 تغییرات محسوسی در ظاهر خود و در واسط کاربریش اعمال کرده است . یکی از این تغییرات JS Link است که به کاربر امکان مدیریت روی Render کردن موجودیت‌های روی صفحه مانند فیلدها ، آیتم‌ها و وب پارت‌ها را به کمک جاوااسکریپت می‌دهد. در این پست نحوه استفاده از این ویژگی جدید را بیان می‌کنم .

وارد سایت شده و یک لیست ایجاد کنید . (در اینجا از Custom List استفاده می‌کنیم .)

و ان را داده آمایی می‌کنیم . هدف این است که بر مبنای عدد موجود در لیست ، رنگ زمینه آن ایتم تغییر کند .

یک فایل جاوااسکریپت ایجاد کنید و کد زیر را در آن ذخیره کنید (از اینجا دانلود کنید) :

```
(function () {
    var itemCtx = {};
    itemCtx.Templates = {};
    itemCtx.Templates.Header = "<div><b title=\"اطلاعات فیلم ها\">Movie Data</b></div><ul>";
    itemCtx.Templates.Item = MyOverrideTemplate;
    itemCtx.Templates.Footer = "</ul>";
    itemCtx.BaseViewID = 1;
    itemCtx.ListTemplateType = 100;
//For Generic List (More : http://msdn.microsoft.com/en-us/library/ms462947(v=office.12).aspx)

    SPClientTemplates.TemplateManager.RegisterTemplateOverrides(itemCtx);

})();



function GT(val , index)
{
    // example of val : 60 %
    var temp = val.split(' ')[0];
    var v = Number(temp);
    return v > index;
}

function LT(val, index) {
    var temp = val.split(' ')[0];
    var v = Number(temp);
    return v < index;
}

function EQ(val, index) {
```

```
    var temp = val.split(' ')[0];
    var v = Number(temp);
    return v == index;
}



function MyOverrideTemplate(ctx) {



    if (LT(ctx.CurrentItem.PopularityPercent ,25))
    {
        return "<li title='خیلی کم بازدید' style='color:white;background-color: red;width: 300px;height:
24px;'>" +
            ctx.CurrentItem.Title + " – " + ctx.CurrentItem.PopularityPercent + "</li>";
    }
    else
    if (LT(ctx.CurrentItem.PopularityPercent ,50))
    {
        return "<li title='کم بازدید'  style='color:maroon;background-color: #ffcc00;width:
300px;height: 24px;'>" +
            ctx.CurrentItem.Title + " – " + ctx.CurrentItem.PopularityPercent + "</li>";
    }
    else
    if (LT(ctx.CurrentItem.PopularityPercent ,75))
    {
        return "<li title='بازدید معمولی'  style='color:#ffcc00;background-color: maroon;width:
300px;height: 24px;'>" +
            ctx.CurrentItem.Title + " – " + ctx.CurrentItem.PopularityPercent + "</li>";
    }
    else
    if (LT(ctx.CurrentItem.PopularityPercent ,95))
    {
        return "<li title='پر بازدید'  style='color:yellow;background-color: blue;width: 300px;height:
24px;'>" +
            ctx.CurrentItem.Title + " – " + ctx.CurrentItem.PopularityPercent + "</li>";
    }
    else
        if (EQ(ctx.CurrentItem.PopularityPercent, 100)) {
            return "<li  title='بالاترین بازدید'  style='color:black;background-color: green;width:
300px;height: 24px;'>" +
                ctx.CurrentItem.Title + " – " + ctx.CurrentItem.PopularityPercent + "</li>";
        }
        else {
            return "<li title='نامعلوم'  style='color:navy;background-color: yellow;width: 300px;height:
24px;'>" +
                ctx.CurrentItem.Title + " – " + ctx.CurrentItem.PopularityPercent + "</li>";
        }
}
```

حال وارد Site Setting شده و وارد Master Pages شوید

System Account
Shared with...
Edit page
Add a page
Add an app
Site contents
Change the look
Site settings
Site settings
Getting started

فایل جاوااسکریپت فوق را از قسمت ریبون و تب Document آپلود کنید و منتظر بمانید تا پس از بارگذاری پنجره ویژگی‌های فایل نمایش داده شود

Save Cancel Paste Cut Copy Delete Item
Commit Clipboard Actions

Hidden
Hide this Display Template option when creating new views.

Target Control Type *
View
Type of control defined in this Display Template.

Icon
Type the Web address: (Click here to test)
/Shared%20Documents/scared-2.png
Type the description:
WOW....
Icon to be displayed for this override.

Standalone *
Override
Option to include this override during view selection.

Target Scope *
/
URL of the website this override applies to.

Target List Template ID
100
ID of the list template type this override applies to.

Version: 1.0
Created at 9/12/2013 4:01 PM by System Account
Last modified at 9/12/2013 4:01 PM by System Account

Save Cancel

هنگلام پر کردن فیلدها به این نکات دقت کنید :

در قسمت Content Type گزینه جدیدی که در این نسخه از شیرپوینت اضافه شده یعنی JavaScript Display Template را انتخاب کنید

در قسمت Target Control Type یکی از سه گزینه view یا Form ویا Field باید انتخاب شوند که در اینجا View را انتخاب میکنیم standalone را روی override تنظیم می‌کنیم . همچنین گزینه Target Scope را که مسیر اعمال فایل است به رووت تنظیم می‌کنیم

در نهایت شناسه List template را به توجه به لیست مورد نظر که در اینجا Custome list است مقدار دهی می‌کنیم . (بیشتر) سپس اطلاعات را ذخیره می‌کنیم .

برای آزمایش این تغییرات بک صفحه می‌سازیم و وب پارت لیست مورد نظر را به آن اضافه می‌کنیم

سپس وارد تنظیمات وب پارت شده و وارد قسمت Miscellaneous می‌شویم

Minimize
Close
Delete
Edit Web Part
Edit Web Part

AJAX Options
Miscellaneous
Sample Data
...
XSL Link
...
Enable Data View Caching
Data View Caching Time-out (seconds)
86400
Send first row to connected Web Parts when page loads
Server Render
Disable view selector menu
Disable 'Save This View' button
Display search box
JS Link
~site/_catalogs/masterpage/N

در قسمت JS Link مسیر فایل خود را به صور نسبی وارد کنید

```
~site/_catalogs/masterpage/MyJsLinkSample.js
```

و نتیجه نهایی :

در صورت بروز Exception در فایل جاوااسکریپت ، خطا به صورت زیر نمایش داده خواهد شد :

موفق باشید

عنوان: آموزش فایرباگ - #10 - DOM Panel
نویسنده: احمد احمدی
تاریخ: ۱۳۹۲/۰۸/۱۰ ۲۲:۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, FireBug, DOM

در پنل DOM توابع و متغییرهایی که در صفحه وجود دارند بصورت درختی نمایش داده می‌شوند. objectها و arrayها قابل باز شدن هستند و بصورت درختی می‌توانید محتویات آن‌ها را مشاهده کنید. توابع هم بصورت لینک هستند که با کلیک برون آن ها، کد مربوط در پنل Script نمایش داده می‌شود. توجه کنید که محتویاتی که مشاهده می‌کنید برای همان لحظه ای است که پنل را باز کردید و برای مشاهده تغییرات ثانویه باید محتویات پنل را بروزرسانی کرد.

قبل از بررسی این پنل اجازه دهید نگاهی به تعریف DOM بیندازیم.

DOM چیست؟

مدل شیءگرای سند یا دام (DOM - Document Object Model) عنوان یکی از دو ساختوارهٔ (architecture) اصلی است (در کنار اس‌اِی‌اکس) که بر اساس آن سندهای اکس‌ام‌ال را به اشیایی که در بردارندهٔ آن است، تجزیه نموده، و آن‌ها را به‌صورت یک ساختار درختی داده‌ها در فضای حافظه اصلی پهن می‌کند. ساختوارهٔ دام، نه به زبان برنامه‌نویسی خاصّی وابستگی دارد و نه به سکّوی برنامه‌نویسی ویژه‌ای، بلکه، به منظور اجراء و پیاده‌سازی آن باید از یک زبان برنامه‌نویسی بلندتراز همچون جاوا، سی‌شارپ، جاوااسکریپت یا مشابه آن‌ها سود بجوییم. آنسوی رابط کاربر سند با مدلی شیءگرا نمایانده می‌شود.

Options Menu

این منو با راست کلیک کردن بروی نام پنل یا کلیک کردن بروی مثلثی که روی پنل قرار دارد، نمایش داده می‌شود.

Show User-defined Properties

در صورت فعال بودن، پراپرتی هایی که توسط کاربر به صفحه اضافه شده اند را نمایش می‌دهد.

Show User-defined Functions

در صورت فعال بودن، توابعی که توسط کاربر به صفحه اضافه شده اند را نمایش می‌دهد.

Show DOM Properties

در صورت فعال بودن، پراپرتی هایی که بصورت پیشفرض در DOM وجود دارند را نمایش می‌دهد.

Show DOM Functions

در صورت فعال بودن، توابعی که بصورت پیشفرض در DOM وجود دارند را نمایش می‌دهد.

Show DOM Constants

در صورت فعال بودن، const هایی که بصورت پیشفرض در DOM وجود دارند را نمایش می‌دهد.

Show Inline Event Handlers

در صورت فعال بودن، رویدادهایی که بصورت خطی در تگ‌ها تعریف شده اند را نمایش می‌دهد.

Show Closures

در صورت فعال بودن، Closure ها را نمایش می‌دهد.

Show Own Properties Only

در صورت فعال بودن، فقط پراپرتی هایی که بروی خود شئ تعریف شده اند را نمایش می‌دهد.

Show Enumerable Properties Only

در صورت فعال بودن، فقط پراپرتی‌های شمارشی را نمایش می‌دهد.

Refresh

محتویات پنل را بروزرسانی می‌کند.

Property Path

این قسمت در بالاترین بخش پنل قرار دارد و وظیفه‌ی آن نمایش مسیر شئ از خود شئ تا window است.

همچنین با راست کلیک کردن بروی این قسمت دو گزینه نمایش داده می‌شود. Refresh برای بروزسازی آدرس نمایش داده شده و Use in Command Line هم برای استفاده از شئ در خط فرمان است. پس از راست کلیک کردن بروی یک شئ و انتخاب گزینه‌ی Use in Command Line فایرباگ تمرکز برنامه را به خط فرمان منتقل می‌کند و شئ را تحت متغییری به نام $p در خط فرمان کپی می‌کند.

رنگ ها

برای مشخص کردن نوع متغییرهای این پنل، فایرباگ برای هر نوع متغییر از یک رنگ استفاده می‌کند.

اشیاء ، اشیاء DOM ، توابع Getter ، **توابع تعریفی کاربر** ، توابع DOM ، **توابع Constructor** ، پراپرتی‌های Read-only

Auto-Completion

مشابه پنل‌های HTML , CSS, Console در این پنل هم امکان اعمال تغییرات همراه با قابلیت تکمیل خودکار وجود دارد.

localStorage

در HTML5 سیستمی برای ذخیره مقادیر در سمت کاربر، به نام localStorage معرفی شد. در این پنل می‌توانید محتویات آن را بررسی/ویرایش کنید. برای کار با توابع آن هم می‌توانید از پنل Console استفاده کنید. (همچنین می‌توانید از پنل Console بصورت Popup در این پنل و پنل‌های دیگر هم استفاده کنید. به تصویر زیر توجه فرمایید.)

توجه کنید که برای مشاهده‌ی این شئ، باید گزینه‌ی Show DOM Properties فعال باشد و همچنین در Property Path، شئ window فعال باشد.

Breakpoint Column

شما می‌توانید با کلیک بروی ستون سمت چپ پراپرتی ها، آن پراپرتی را تحت نظر گرفته و در صورت تغییر یافتن مقدار آن پراپرتی، کنترل روند اجرای برنامه از همان نقطه را بدست بگیرید. به این صورت که زمانی که کدی پراپرتی موردنظر را تغییر دهد، پنل Script روند اجرای کد را در همان قسمت متوقف می‌کند.

(ممکن است فایرباگ بصورت خودکار به پنل Script سوئیچ نکند و بعد از متوقف شدن برنامه هم اگر به پنل Script سوئیچ کنید نتیجه را نبینید. پس بهتر است قبل از تغییر یافتن پراپرتی مورد نظر و بعد از قرار دادن Breakpoint به پنل Script بروید.)

Context Menu

با راست کلیک کردن در قسمت‌های مختلف پنل، منوهای متفاوتی را خواهید دید. همچنین با راست کلیک کردن بروی مقادیر پراپرتی ها، منوی متناسب با آن مقدار را خواهید دید. مثلا اگر بروی یک تگ HTML در این پنل راست کلیک کنید، منویی که خواهید دید همان منویی است که در پنل HTML مشاهده می‌کردید.

گزینه	Context	توضیحات
Copy Name	Property List	نام پراپرتی را در حافظه کپی می‌کند.
Copy Path	Property List	آدرس پراپرتی را در حافظه کپی می‌کند.
Copy Value	String and Number values	محتوای پراپرتی را در حافظه کپی می‌کند.
Edit Property...	Property List (پراپرتی و توابع کاربر)	پراپرتی را به حالت ویرایش می‌آورد.
Delete Property	Property List (پراپرتی و توابع کاربر)	پراپرتی را حذف می‌کند.
Break On Property Change	Property List (پراپرتی و توابع کاربر)	مشابه پاراگرف قبلی.
Refresh	Property List, Property Path	محتویات پنل را بروزرسانی می‌کند.

برای توابع هم دو منوی اضافی وجود دارد:

"<Log Calls to "<function name

فراخوانی‌های تابع مورد نظر را Log می‌کند. (برای توضیحات بیشتر دستور monitor که از توابع خط فرمان است را ملاحظه بفرمایید.)

Copy Function

نام و بنده‌ی تابع را در حافظه کپی می‌کند.

عنوان: **جلوگیری از درج صفحات سایت در سایتی دیگر از طریق iframeها**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۴ ۸:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, JavaScript, MVC, Security

یکی از مواردی را که به کرات در ارجاع دهنده‌های سایت مشاهده می‌کنم، درج صفحات سایت به صورت iframe داخل یک سری سایت‌های تبلیغاتی بسیار سطحی است. سؤال: چگونه می‌توان جلوی این حرکت نامطلوب را گرفت؟
برای این‌کار نیاز است اسکریپت ذیل را به ابتدای اسکریپت‌های سایت خود اضافه کنید:

```
function defrm() {
    document.write = '';
    window.top.location = window.self.location;
    setTimeout(function () {
        document.body.innerHTML = '';
    }, 0);
    window.self.onload = function (evt) {
        document.body.innerHTML = '';
    };
}
if (window.top !== window.self) {
    try {
        if (window.top.location.host)
        { /* will throw */ }
        else {
            defrm(); /* chrome */
        }
    } catch (ex) {
        defrm(); /* everyone else */
    }
}
```

اگر شیء window.top با شیء window.self یکی نبود (یعنی window جاری، window رویی نبود)، بنابراین صفحه‌ی جاری، داخل iframe قرار گرفته‌است. همچنین برای اطمینان بیشتر می‌توان window.top.location.host را نیز فراخوانی کرد. مرورگرهای جدید امکان دسترسی به دومین والد را از طریق یک iframe نمی‌دهند و همینجا یک استثناء صادر خواهد شد.
در مرحله بعد، با فراخوانی window.top.location = window.self.location و سایر کدهایی که مشاهده می‌کنید، کل صفحه را به صورت خودکار به سایت خودمان هدایت خواهیم کرد.

این مورد را نیز می‌توان به مجموعه best practices اجباری تهیه یک سایت اضافه کرد.

نظرات خوانندگان

نویسنده: حامد سبزیان
تاریخ: ۱۳۹۲/۰۹/۲۴ ۹:۴

استفاده از X-FRAME-OPTIONS در هدر درخواست با مقدار SAMEORIGIN با DENY کفایت نمیکنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۴ ۹:۲۰

خوبه ؛ فقط redirect بحث شده در مطلب را نداره

نویسنده: امید قربانی
تاریخ: ۱۳۹۲/۰۹/۲۷ ۹:۱۹

با سلام؛ اگر بخواهیم این موردی رو که اشاره کردید رو دور بزنیم چطور باید عمل کنیم منظورم اینه که اگر سایتی همچین قابلیتی رو اضافه کرده بود چطور از بین ببریمش ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۷ ۹:۲۶

حداقل با استفاده از جاوا اسکریپت، به دلایل امنیتی ، امکان دسترسی به محتویات یک iframe بارگذاری شده از سایت دیگری را ندارید؛ مگر اینکه سمت سرور، محتویات آن صفحه را دریافت و در ادامه این صفحه‌ی محلی شده را نشان دهید.

نویسنده: behrouz
تاریخ: ۱۳۹۲/۱۲/۲۸ ۲۲:۱۰

سلام
آیا سمت سرور راهی هست که بتوان تشخیص داد صفحاتی که درخواست داده می‌شوند از درون یک Iframe فرخوانی می‌شوند یا خیر(به طور خاص در ASP.NET MVC)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۸ ۲۲:۵۹

خیر. راه حل تشخیص سمت سرور، ندارد. هیچ هدر خاصی برای مشخص سازی درخواست اجرای یک صفحه از درون یک IFrame طراحی نشده‌است. حداکثر می‌توان Request.UrlReferrer را بررسی کرد (ولی اطمینانی به آن نیست. چون عوامل زیادی در تغییر آن دخیل هستند؛ از مرورگر تا فایروال و غیره)

عنوان: **آشنایی با مفاهیم شیء گرایی در جاوا اسکریپت #1**
نویسنده: مهران ر
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۴:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, OOP

با توجه به فراگیر شدن استفاده از جاوا اسکریپت و بخصوص مبحث شیء گرایی، تصمیم گرفتم طی سلسله مقالاتی با مباحث شیء گرایی در این زبان بیشتر آشنا شویم. جاوا اسکریپت یک زبان مبتنی بر شیء است و نه شیءگرا و خصوصیات زبان‌های شیء گرا، به طور کامل در آن پیاده سازی نمی‌گردد.
لازم به ذکر است که انواع داده‌ای در جاوا اسکریپت شامل 2 نوع می‌باشند:
1- نوع داده اولیه (Primitive) که شامل Number ، Boolean و Strings می‌باشند.
2- نوع داده Object که طبق تعریف هر Object مجموعه‌ای از خواص و متدها است.
نوع داده‌ای اولیه، از نوع Value Type و نوع داده ای Object، از نوع Refrence Type می‌باشد.

برای تعریف یک شیء (Object) در جاوا اسکریپت، 3 راه وجود دارد:
1 - تعریف و ایجاد یک نمونه مستقیم از یک شیء (direct instance of an object)
2 – استفاده از function برای تعریف و سپس نمونه سازی از یک شیء (Object Constructor)
3 – استفاده از متد Object.Create

روش اول :
در روش اول دو راه برای ایجاد اشیاء استفاده می‌گردد که با استفاده از دو مثال ذیل، این دو روش توضیح داده شده‌اند:

مثال اول : (استفاده از new)

```
<script type="text/javascript">
 var person = new Object();
  person.firstname = "John";
  person.lastname = "Doe";
  person.age = 50;
  person.eyecolor = "blue";
  document.write(person.firstname + " is " + person.age + " years old.");
</script>
```

```
result : John is 50 years old.
```

در این مورد، ابتدا یک شیء پایه ایجاد می‌گردد و خواص مورد نظر برایش تعریف می‌گردد و با استفاده از اسم شیء به این خواص دسترسی داریم.

مثال دوم (استفاده از literal notation)

```
<script type="text/javascript">
var obj = {
   var1: "text1",
   var2: 5,
   Method: function ()
   {
     alert(this.var1);
   }
  };
  obj.Method();
</script>
```

```
Result : text1
```

در این مورد با استفاده از کلمه کلیدی var یک شیء تعریف می‌شود و در داخل {} کلیه خواص و متدهای این شیء تعریف می‌گردد. این روش برای تعریف اشیاء در جاوا اسکریپت بسیار متداول است.
هر دو مثالهای 1 و 2 در روش اول برای ایجاد اشیاء بکار میروند. امکان گسترش دادن اشیاء در این روش و اضافه کردن خواص و متد در آینده نیز وجود دارد. بعنوان مثال می‌توان نوشت :

```
Obj.var3 = “text3”;
```

در این‌حال، خاصیت سومی به مجموع خواص شی Obj اضافه می‌گردد.
حال در این مثال اگر مقدار شی obj را برابر یک شیء دیگر قرار دهیم به نحو زیر :

```
var newobj = obj;
  newobj.var1 = "other text";
  alert(obj.var1);// other text
  alert(newobj.var1);// other text
```

و برای اینکه بتوان از امکانات زبانهای شیء گرا در این زبان استفاده کرد، بایستی الگویی را تعریف کنیم و سپس از روی این الگو، اشیا مورد نظر را پیاده سازی نمائیم.
می‌بینیم که مقدار هر دو متغیر در خروجی یکسان می‌باشد و این موضوع با ماهیت شیء گرایی که در آن همه‌ی اشیایی که از روی یک الگو نمونه سازی می‌گردند مشخصه‌هایی یکسان، ولی مقادیر متفاوتی دارند، متفاوت است. البته این موضوع از آنجا ناشی می‌گردد که اشیاء ایجاد شده در جاوا اسکریپت ذاتا type refrence هستند و به همین منظور برای پیاده سازی الگویی (کلاسی) که بتوان رفتار شیء گرایی را از آن انتظار داشت از روش زیر استفاده میکنیم. برای درک بهتر اسم این الگو را کلاس مینامیم که در روش دوم به آن اشاره میکنیم.

روش دوم :

```
<script type=”text/javascript”>
  function Person(firstname, lastname, age, eyecolor)
  {
     this.firstname = firstname;
     this.lastname = lastname;
     this.age = age;
     this.eyecolor = eyecolor;
  }

   var myFather = new Person("John", "Doe", 50, "blue");
  document.write(myFather.firstname + " is " + myFather.age + " years old.");
  result : John is 50 years old.

   var myMother=new person("Sally","Rally",48,"green");
   document.write(myMother.firstname + " is " + myFather.age + " years old.");
   result : Sally is 48 years old.
  </script>
```

یا به شکل زیر :

```
var Person = function (firstname, lastname, age, eyecolor)
  {
     this.firstname = firstname;
     this.lastname = lastname;
     this.age = age;
     this.eyecolor = eyecolor;
  }

  var myFather = new Person("John", "Doe", 50, "blue");
  document.write(myFather.firstname + " is " + myFather.age + " years old.");
   result : John is 50 years old.
```

```
   var myMother=new person("Sally","Rally",48,"green");
   document.write(myMother.firstname + " is " + myFather.age + " years old.");
   result : Sally is 48 years old.
```

به این روش Object Constructor یا سازنده اشیاء گفته می‌شود.
در اینجا با استفاده از کلمه کلیدی function و در داخل {} کلیه خواص و متدهای لازم را به شیء مورد نظر اضافه می‌کنیم.
استفاده از کلمه this در داخل function به این معنی است که هر کدام از نمونه‌های object مورد نظر، مقادیر متفاوتی خواهند داشت .

یک مثال دیگر :

```
<script type="text/javascript">
   function cat(name) {
      this.name = name;
      this.talk = function() {
      alert( this.name + " say meeow!" )
   }
 }

cat1 = new cat("felix")
cat1.talk() //alerts "felix says meeow!"
cat2 = new cat("ginger")
cat2.talk() //alerts "ginger says meeow!"
</Script>
```

در اینجا می‌بینیم که به ازای هر نمونه از اشیایی که با function می‌سازیم، خروجی متفاوتی تولید می‌گردد که همان ماهیت شیء گرایی است.

روش سوم :استفاده از متد Object.Create

```
var myObjectLiteral = {
    property1: "one",
    property2: "two",
    method1: function() {
       alert("Hello world!");
}}
var myChild = Object.create(myObjectLiteral);
myChild.method1(); // will alert "Hello world!"
```

در این روش با استفاده از متد Object.Create و استفاده از یک شیء که از قبل ایجاد شده، یک شیء جدید ایجاد می‌شود.
حال برای اضافه کردن متدها و خاصیت‌هایی به کلاس جاوا اسکریپتی مورد نظر، به طوریکه همه‌ی نمونه‌هایی که از این کلاس ایجاد می‌شوند بتوانند به این متدها و خاصیت‌ها دسترسی داشته باشند، از مفهومی به اسم prototype استفاده می‌کنیم. برای مثال کلاس زیر را در نظر بگیرید:
این کلاس یک سیستم ساده امتحانی (quiz) را پیاده می‌کند که در آن اطلاعات شخص که شامل نام و ایمیل می‌باشد گرفته شده و سه تابع، شامل ذخیره نمرات، تغییر ایمیل و نمایش اطلاعات شخص به همراه نمرات نیز به آن اضافه می‌شود.

```
function User (theName, theEmail) {
    this.name = theName;
    this.email = theEmail;
    this.quizScores = [];
    this.currentScore = 0;
}
```

حال برای اضافه نمودن متدهای مختلف به این کلاس داریم :

```
User.prototype = {
  saveScore:function (theScoreToAdd)  {
   this.quizScores.push(theScoreToAdd)
  },
  showNameAndScores:function ()  {
    var scores = this.quizScores.length > 0 ? this.quizScores.join(",") : "No Scores Yet";
    return this.name + " Scores: " + scores;
  },
  changeEmail:function (newEmail)  {
    this.email = newEmail;
    return "New Email Saved: " + this.email;
  }
}
```

و سپس برای استفاده از آن و گرفتن خروجی نمونه داریم :

```
// A User
  firstUser = new User("Richard", "Richard@examnple.com");
  firstUser.changeEmail("RichardB@examnple.com");
  firstUser.saveScore(15);
  firstUser.saveScore(10);
  document.write(firstUser.showNameAndScores()); //Richard Scores: 15,10
  document.write('<br/>');
  // Another User
  secondUser = new User("Peter", "Peter@examnple.com");
  secondUser.saveScore(18);
  document.write(secondUser.showNameAndScores()); //Peter Scores: 18
```

در نتیجه تمام نمونه‌های کلاس User می‌توانند به این متدها دسترسی داشته باشند و به این صورت مفهوم Encapsulation نیز پیاده می‌گردد.

وراثت (Inheritance) در جاوا اسکریپت :

در بسیاری از مواقع لازم است عملکردی (Functionality) که در یک کلاس تعریف می‌گردد، در کلاسهای دیگر نیز در دسترس باشد. بدین منظور از مفهوم وراثت استفاده می‌شود. در نتیجه کلاس‌ها می‌توانند از توابع خود و همچنین توابعی که کلاسهای والد در اختیار آنها می‌گذارند استفاده کنند. برای این منظور چندین راه حل توسط توسعه دهندگان ایجاد شده است که در ادامه به چند نمونه از آنها اشاره می‌کنیم.
ساده‌ترین حالت ممکن از الگویی شبیه زیر است:

```
<script type="text/javascript">
function Base()
  {
     this.color = "blue";
  }
  function Sub()
  {
  }
  Sub.prototype = new Base();
  Sub.prototype.showColor = function ()
  {
     alert(this.color);
  }
  var instance = new Sub();
  instance.showColor(); //"blue"
</Script>
```

در کد بالا ابتدا یک class (function) به نام Base که حاوی یک خصوصیت به نام color می‌باشد، تعریف شده و سپس یک کلاس دیگر بنام sub تعریف می‌کنیم که قرار است خصوصیات و متدهای کلاس Base را به ارث ببرد و سپس از طریق خصوصیت prototype کلاس Sub، که نمونه‌ای از کلاس Base را به آن نسبت می‌دهیم باعث می‌شود خواص و متدهای کلاس Base توسط کلاس Sub قابل دسترسی باشد. در ادامه متد showColor را به کلاس Sub اضافه می‌کنیم و توسط آن به خصوصیت color در این کلاس دسترسی پیدا میکنیم.

راه حل دیگری نیز برای اینکار وجود دارد که الگویی است بنام Parasitic Combination :

در این الگو براحتی و با استفاده از متد Object.create که در بالا توضیح داده شد، هر کلاسی که ایجاد میکنیم، با انتساب آن به یک شیء جدید، کلیه خواص و متدهای آن نیز توسط شیء جدید قابل استفاده میشود.

```
<script language="javascript" type="text/javascript">
if (typeof Object.create !== 'function') {
Object.create = function (o) {
//ایجاد یک کلاس خالی که قرار است خواص کلاس دریافتی توسط آرگومان کلاس پایه را به ارث ببرد
  function F() {
}
با انتساب F توسط خصوصیت Prototype باعث میشویم کلیه خواص و متدهای دریافتی توسط F با ارث برده شود
آرگومان دریافتی که یک شی است به کلاس
  F.prototype = o;
   return new F();
  };
}

var cars = {
type: "sedan",
wheels: 4
  };
  // We want to inherit from the cars object, so we do:
  var toyota = Object.create(cars);
// now toyota inherits the properties from cars
  document.write(toyota.type);
</script>
output :sedan
```

در قسمتهای دیگر به مباحثی همچون Override و CallBaseMethodها خواهیم پرداخت.

برای مطالعه بیشتر :

http://eloquentjavascript.net/chapter8.html

http://phrogz.net/JS/classes/OOPinJS2.html

عنوان: **آشنایی با مفاهیم شیء گرایی در جاوا اسکریپت #2**
نویسنده: مهران ر
تاریخ: ۱۳۹۲/۱۰/۰۸ ۱۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, OOP

از آنجا که برای کار با جاوا اسکریپت نیاز به درک کاملی درباره‌ی مفهوم حوزه کارکرد متغیرها (Scope) می‌باشد و نحوه فراخوانی توابع نیز نقش اساسی در این مورد بازی می‌کند، در این قسمت با این موارد آشنا خواهیم شد:
جاوا اسکریپت از مفهومی به نام functional scope برای تعیین حوزه متغیرها استفاده می‌کند و به این معنی است که با تعریف توابع، حوزه عملکرد متغیر مشخص می‌شود. در واقع هر متغیری که در یک تابع تعریف می‌شود در کلیه قسمتهای آن تابع، از قبیل If statement - for loops و حتی nested function نیز در دسترس میباشد.
اجازه دهید با مثالی این موضوع را بررسی نماییم.

```
function testScope() {
var myTest = true;
if (true) {
var myTest = "I am changed!"
}
alert(myTest);
}
testScope(); // will alert "I am changed!"
```

همانگونه که میبینیم با اینکه در داخل بلاک if یک متغیر جدید تعریف شده، ولی در خارج از این بلاک نیز این متغیر قابل دسترسی میباشد. البته در مثال بالا اگر بخواهیم به متغیر myTest در خارج از function دسترسی داشته باشیم، با خطای undefined مواجه خواهیم شد. یعنی برای مثال در کد زیر:

```
function testScope() {
var myTest = true;
if (true) {
var myTest = "I am changed!"
}
alert(myTest);
}
testScope(); // will alert "I am changed!"
alert(myTest); // will throw a reference error, because it doesn't exist outside of the function
```

برای حل این مشکل دو راه وجود دارد:

1 - متغیر myTest را در بیرون بلاک testScope() تعریف کنیم
2 - هنگام تعریف متغیر myTest، کلمه کلیدی var را حذف کنیم که این موضوع باعث میشود این متغیر در کل window قابل دسترس باشد و یا به عبارتی متغیر global میشود.
قبل از پرداختن به ادامه بحث خواندن مقاله مربوط به Closure در جاوااسکریپت توصیه میگردد .
در پایان بحث Scopeها با یک مثال نسبتا جامع اکثر این حالات به همراه خروجی را نشان میدهیم :

```
<script type="text/javascript">
          // a globally-scoped variable
        var a = 1;
        // global scope
        function one()
        {
            alert(a);
        }
        // local scope
        function two(a)
        {
            alert(a);
        }
        // local scope again
        function three()
        {
            var a = 3;
```

```
            alert(a);
        }
        // Intermediate: no such thing as block scope in javascript
        function four()
        {
            if (true)
            {
                var a = 4;
            }
            alert(a); // alerts '4', not the global value of '1'
        }
        // Intermediate: object properties
        function Five()
        {
            this.a = 5;
        }
        // Advanced: closure
        var six = function ()
        {
            var foo = 6;
            return function ()
            {
                // javascript "closure" means I have access to foo in here,
                // because it is defined in the function in which I was defined.
                alert(foo);
            }
        }()
        // Advanced: prototype-based scope resolution
        function Seven()
        {
            this.a = 7;
        }
        // [object].prototype.property loses to [object].property in the lookup chain
        Seven.prototype.a = -1; // won't get reached, because 'a' is set in the constructor above.
        Seven.prototype.b = 8; // Will get reached, even though 'b' is NOT set in the constructor.
        // These will print 1-8
        one();
        two(2);
        three();
        four();
        alert(new Five().a);
        six();
        alert(new Seven().a);
        alert(new Seven().b);
</Script>
```

برای مطالعه بیشتر به اینجا مراجعه نمایید.

Function Invocation Patterns In JavaScript :

از آنجا که توابع در جاوااسکریپت به منظور 1 – ساخت اشیاء و 2 – حوزه دسترسی متغیرها(Scope) نقش اساسی ایفا می‌کنند بهتر است کمی درباره استفاده و نحوه فراخوانی آنها (Function Invocation Patterns) در جاوااسکریپت بحث نماییم.

در جاوااسکریپت 4 مدل فراخوانی تابع داریم که به نامهای زیر مطرح هستند:

1. Method Invocation
2. Function Invocation
3. Constructor Invocation
4. Apply And Call Invocation

در فراخوانی توابع به هر یک از روشهای بالا باید به این نکته توجه داشت که حوزه دسترسی متغیرها در جاوااسکریپت ابتدا و انتهای توابع هستند و اگر به عنوان مثال از توابع تو در تو استفاده کردیم ،حوزه شی this برای توابع داخلی تغییر خواهد کرد .این موضوع را در طی مثالهایی نشان خواهیم داد. **Method Invocation :**

وقتی یک تابع قسمتی از یک شی باشد به آن متد میگوییم به عنوان مثال :

```
var obj = {
    value: 0,
    increment: function() {
        this.value+=1;
    }
};
obj.increment(); //Method invocation
```

در اینحالت this به شی (Object) اشاره میکند که متد در آن فراخوانی شده است و در زمان اجرا نیز به عناصر شی Bind میشود ،در مثال بالا حوزه this شی obj خواهد شد و به همین منظور به متغیر value دسترسی داریم. **Function Invocation:**
در اینحالت که از () برای فراخوانی تابع استفاده میگردد، This به شی سراسری (global object) اشاره می‌کند؛ منظور اینکه this به اجزای تابعی که فراخوانی آن انجام شده اشاره نمی‌کند. اجازه دهید با مثالی این موضوع را روشن کنیم

```
<script type="text/javascript">
var value = 500; //Global variable
var obj = {
    value: 0,
    increment: function() {
        this.value++;
        var innerFunction = function() {
            alert(this.value);
        }
        innerFunction(); //Function invocation pattern
    }
}
obj.increment(); //Method invocation pattern
<script type="text/javascript">
Result : 500
```

از آنجا که () innerFunction به شکل Function invocation pattern فراخوانی شده است به متغیر value در داخل تابع increment دسترسی نداریم و حوزه دسترسی global میشود و اگر در حوزه global نیز این متغیر تعریف نشده بود به خطای undefined میرسیدیم .
برای حل این گونه مشکلات ساختار کد نویسی ما بایستی به شکل زیر باشد :

```
<script type="text/javascript">
var value = 500; //Global variable
var obj = {
    value: 0,
    increment: function() {
        var that = this;
        that.value++;
        var innerFunction = function() {
            alert(that.value);
        }
        innerFunction(); //Function invocation pattern
    }
}
obj.increment();
<script type="text/javascript">
Result : 1
```

در واقع با تعریف یک متغیر با نام مثلا that و انتساب شی this به آن میتوان در توابع بعدی که به شکل Function invocation pattern فراخوانی میگردند به این متغیر دسترسی داشت . **Constructor Invocation :**
در این روش برای فراخوانی تابع از کلمه new استفاده میکنیم. در این حالت یک شیء مجزا ایجاد شده و به متغیر دلخواه ما اختصاص پیدا می‌کند. به عنوان مثال داریم :

```
var Dog = function(name) {
  //this == brand new object ({});
    this.name = name;
    this.age = (Math.random() * 5) + 1;
};
var myDog = new Dog('Spike');
//myDog.name == 'Spike'
//myDog.age == 2
var yourDog = new Dog('Spot');
//yourDog.name == 'Spot'
//yourDog.age == 4
```

در این مورد با استفاده از New باعث میشویم همه خواص و متدهای تابع function برای هر نمونه از آن که ساخته میشود (از طریق مفهوم Prototype که قبلا درباره آن بحث شد) بطور مجزا اختصاص یابد. در مثال بالا شی mydog چون حاوی یک نمونه از

تابع dog بصورت Constructor Invocation میباشد، در نتیجه به خواص تابع dog از قبیل name و age دسترسی داریم. در اینجا اگر کلمه new استفاده نشود به این خواص دسترسی نداریم؛ در واقع با اینکار، this به mydog اختصاص پیدا میکند.
اگر از new استفاده نشود متغیر undefined، myDog میشود.
یک مثال دیگر :

```
var createCallBack = function(init) { //First function
    return new function() { //Second function by Constructor Invocation
        var that = this;
        this.message = init;
        return function() { //Third function
            alert(that.message);
        }
    }
}
window.addEventListener('load', createCallBack("First Message"));
window.addEventListener('load', createCallBack("Second Message"));
```

در مثال بالا از مفهوم closure نیز در مثالمان استفاده کرده ایم . **Apply And Call Invocation:**

تمامی توابع جاوااسکریپت دارای دو متد توکار apply() و call() هستند که توسط این متدها میتوان این توابع را با context دلخواه فراخوانی کرد.
نحوه فراخوانی به شکل مقابل است :

```
myFunction.apply(thisContext, arrArgs);
myFunction.call(thisContext, arg1, arg2, arg3, ..., argN);
```

که thisContext به حوزه اجرایی (execution context) تابع اشاره میکند. تفاوت دو متد apply() و call() در نحوه فرستادن آرگومانها به تابع میباشد که در اولی توسط آرایه اینکار انجام میشود و در دومی همه آرگومانها را بطور صریح نوشته و با کاما از هم جدا میکنیم .

مثال :

```
var contextObject = {
testContext: 10
}
var otherContextObject = {
testContext: "Hello World!"
}
var testContext = 15; // Global variable
function testFunction() {
alert(this.testContext);
}
testFunction(); // This will alert 15
testFunction.call(contextObject); // Will alert 10
testFunction.apply(otherContextObject); // Will alert "Hello World”
```

در این مثال دو شی متفاوت با خواص همنام تعریف کرده و یک متغیر global نیز تعریف میکنیم. در انتها یک تابع تعریف میکنیم که مقدار this.testContext را نمایش میدهد. در ابتدا حوزه اجرایی تابع (this) کل window جاری میباشد و وقتی testFunction() اجرا شود مقدار متغیر global نمایش داده میشود. در اجرای دوم this به contextObject اشاره کرده و حوزه اجرایی عوض میشود و در نتیجه مقدار testContext مربوطه که در این حالت 10 میباشد نمایش داده میشود و برای فراخوانی سوم نیز به همین شکل .

یک مثال کاملتر :

```
var o = {
  i : 0,
  F : function() {
    var a = function() { this.i = 42; };
    a();
```

```
    document.write(this.i);
  }
};
o.F();
Result :0
```

خط o.f() تابع f را به شکل Method invocation اجرا میکند. در داخل تابع f یک تابع دیگر به شکل function invocation اجرا میشود که در اینحال this به global object اشاره میکند و باعث میشود مقدار i در خروجی 0 چاپ شود .

برای حل این مشکل 2 راه وجود دارد

راه اول :

```
var p = {
  i : 0,
  F : function() {
    var a = function() { this.i = 42; };
    a.apply(this);
    document.write(this.i);
  }
};
 p.F();
Result :42
```

با اینکار this را موقع اجرای تابع درونی برایش فرستاده تا حوزه اجرای تابع عوض شود و به i دسترسی پیدا کنیم .

یا اینکه همانند مثالهای قبلی :

```
var q = {
  i: 0,
  F: function F() {
    var that = this;
    var a = function () {
      that.i = 42;
    }
    a();
    document.write(this.i);
  }
}
 q.F();
```

منابع :

Javascript programmer,s refrence http://coding.abel.nu/2013/03/more-on-this-in-javascript/

http://seanmonstar.com/post/707068021/4-ways-functions-mess-with-this

عنوان: آشنایی با Promises در جاوا اسکریپت
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۱۲ ۸:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery Ajax, AngularJS

در حین انجام اعمال غیرهمزمان جاوا اسکریپتی مانند فراخوانی‌های jQuery AJAX، برای مدیریت دریافت نتایج، عموما از یک سری callback استفاده می‌شود. برای مثال:

```
$.get('http://site-url', function(data) {
//این تابع پس از پایان کار عملیات ای‌جکسی در آینده فراخوانی خواهد شد
});
```

تا اینجا مشکلی به نظر نمی‌رسد. اما مورد ذیل چطور؟

```
$.get('http://site-url/0', function(data0) {
    // callback #1
    $.get('http://site-url/1', function(data1) {
        // callback #2
        $.post('http://site-url/2', function(data2) {
            // callback #3
        });
    });
});
```

در اینجا نیاز است پس از پایان کار عملیات Ajax ایی اول، عملیات دوم و پس از آن عملیات سومی انجام شود. همانطور که مشاهده می‌کنید، این نوع کدها به سرعت از کنترل خارج می‌شوند؛ خوانایی پایینی داشته و مدیریت استثناءهای رخ داده در آن‌ها نیز در این بین مشکل است. از این جهت که خطاهای هر کدام به سطحی بالاتر منتقل نمی‌شود و باید همانجا محلی و داخل هر callback مدیریت گردد.

روش‌های زیادی برای حل این مساله ارائه شده‌است و در حال حاضر کار کردن با promiseها متداول‌ترین روش حل مدیریت فراخوانی کدهای همزمان جاوا اسکریپتی است. برای نمونه اگر از AngularJS استفاده کنید، سرویس‌های آن برای دریافت اطلاعات از سرور، از یک چنین مفهومی استفاده می‌کنند.

Promise در جاوا اسکریپت چیست؟

شیء Promise، نمایانگر قراردادی است که در آینده می‌تواند مورد قبول واقع شود، یا رد گردد. بررسی این قرارداد، تنها یکبار می‌تواند رخ دهد (پذیرش یا رد آن). هنگامیکه این بررسی صورت گرفت (رد یا پذیرش آن و نه هردو)، یک callback برای اطلاع رسانی فراخوانی می‌گردد. سپس این callback می‌تواند یک Promise دیگر را سبب شود. به این ترتیب می‌توان Promiseها را زنجیر وار به یکدیگر متصل کرد. برای نمونه jQuery به صورت توکار از promises پشتیبانی می‌کند:

```
// returns a promise
$.get('http://site-url/0')
.then(function(data) {
    // callback 1
    // returns a promise
    return $.get('http://site-url/1');
})
.then(function(data) {
    // callback 2
    // returns a promise
    return $.post('http://site-url/2');
})
.then(function(data) {
    // callback 3
});
```

متد get در jQuery یک شیء promise را بازگشت می‌دهد. در ادامه می‌توان این نتیجه را توسط متد then، زنجیروار ادامه داد. متدی که به عنوان پارامتر به then ارسال می‌شود، یک callback بوده و پس از پایان کار promise قبلی رخ می‌دهد. آرگومانی که به

این callback ارسال می‌شود، نتیجه‌ی promise قبلی است. در حین اعمال jQuery Ajax، این callback تنها زمانی فراخونی می‌شود که عملیات قبلی موفقیت آمیز بوده باشد و data ارائه شده، اطلاعاتی است که توسط response دریافتی از سرور، دریافت گردیده‌است.

در این حالت، هر callback حداقل سه کار را می‌تواند انجام دهد:

الف) یک promise دیگر را بازگشت دهد. نمونه آن‌را با return $.get در کدهای فوق ملاحظه می‌کنید.

ب) خاتمه عادی. همینجا کار promise با مقدار بازگشت داده شده، پایان می‌یابد.

ج) صدور یک استثناء. سبب برگشت خوردن و عدم پذیرش promise می‌شود.

استفاده از Promises در سایر کتابخانه‌ها

jQuery پیاده سازی توکاری از promises دارد؛ اما سایر کتابخانه‌ها، مانند AngularJS ایی که مثال زده شده چطور عمل می‌کنند؟ استانداردی به نام Promises/A+ جهت یک دست سازی پیاده سازی‌های promise در جاوا اسکریپت پیشنهاد شده‌است. jQuery نیمی از آن‌را پیاده سازی کرده‌است؛ اما کتابخانه‌ی دیگری به نام Q Library ، پیاده سازی نسبتا مفصل‌تری را از این استاندارد ارائه می‌دهد. فریم ورک AngularJS نیز در پشت صحنه از همین کتابخانه برای پیاده سازی promises استفاده می‌کند.

آشنایی با کتابخانه Q

استفاده مقدماتی از Q همانند مثالی است که از jQuery ملاحظه کردید.

```
Q.fcall(callback1)
.then(callback2);
```

اشیاء promise بازگشت داده شده توسط jQuery نیز توسط کتابخانه Q مورد پذیرش واقع می‌شوند:

```
Q.fcall(function() {
    return $.get('http://my-url');
})
.then(callback3);
```

علاوه بر این‌ها مفهومی به نام deferred objects نیز در کتابخانه‌ی Q پیاده سازی شده‌است:

```
function waitForClick() {
    var deferred = Q.defer();

$('#okButton').click(function() {
        deferred.resolve();
    });

$('#cancelButton').click(function() {
        deferred.reject();
    });

return deferred.promise;
}

Q.fcall(waitForClick)
.then(function() {
    //  ok button was clicked
}, function() {
    //  cancel button was clicked
});
```

توسط deferred objects می‌توان بررسی یک promise را به تاخیر انداخت. در مثال فوق، اولین callback فراخوانی شده به نام waitForClick، از اشیاء به تاخیر افتاده استفاده می‌کند. ابتدا توسط فراخوانی متد Q.defer، یک deferred object ایجاد می‌شود. در این بین اگر کاربر بر روی دکمه‌ی OK کلیک کرد، با فراخوانی deferred.resolve، این promise مورد پذیرش واقع خواهد شد و یا اگر کاربر بر روی دکمه‌ی cancel کلیک کند، با فراخوانی متد deferred.reject، این promise رد می‌گردد. نهایتا شیء promise

توسط deferred.promise بازگشت داده خواهد شد.

در ادامه کار، اینبار متد then، دو callback را قبول می‌کند. Callback اول پس از پذیرش قرار داد و Callback دوم پس از رد قرار داد، فراخوانی خواهد گردید.

در رنجیره تعریف شده، اگر معادلی برای reject درنظر گرفته نشده باشد، مانند مثال ذیل:

```
Q.fcall(myFunction1)
.then(success1)
.then(success2, failure1);
```

Q به دنبال نزدیک‌ترین متد callback گزارش خطای کار خواهد گشت. در این حالت متد failure1 در صورت شکست اولین promise فراخوانی خواهد شد.

همچنین اگر نتیجه‌ی success1 با شکست مواجه شود نیز failure1 فراخوانی می‌گردد. اما باید درنظر داشت که شکست success2، توسط failure1 مدیریت نمی‌شود.

Promises در AngularJS

در AngularJS امکانات کتابخانه Q توسط پارامتری به نام $q در اختیار سرویس‌های برنامه قرار می‌گیرد (تزریق می‌شود):

```
var app = angular.module("myApp", []);
app.factory('dataSvc', function($http, $q){
  var basePath="api/books";
  getAllBooks = function(){
   var deferred = $q.defer();
 $http.get(basePath).success(function(data){
    deferred.resolve(data);
   }).error(function(err){
    deferred.reject("service failed!");
   });
   return deferred.promise;
  };

  return{
   getAllBooks:getAllBooks
  };
});

app.controller('HomeController', function($scope, $window, dataSvc){
 function initialize(){
   dataSvc.getAllBooks().then(function(data){
    $scope.books = data;
   }, function(msg){
    $window.alert(msg);
   });
 }

 initialize();
});
```

در اینجا اگر دقت کنید، مباحث و عملکرد آن دقیقا مانند قبل است. ابتدا یک deferred object با فراخوانی متد q.defer ایجاد شده است. سپس با استفاده از امکانات توکار http آن (بجای استفاده از jQuery Ajax)، کار فراخوانی یک restful service صورت گرفته است (مثلا فراخوانی یک ASP.NET Web API). در صورت موفقیت کار، متد deferred.resolve و در صورت عدم موفقیت، متد deferred.reject فراخوانی شده‌است. نهایتا این سرویس، یک deferred.promise را بازگشت می‌دهد.

اکنون در کنترلری که قرار است از این سرویس استفاده کند، متد then کتابخانه Q را ملاحظه می‌کنید که دو Callback متناظر resolve و reject مدیریت promise بازگشت داده شده را به همراه دارد. اگر عملیات Ajaxایی موفقیت آمیز باشد، شیء books را مقدار دهی می‌کند و اگر خیر، پیامی را به کاربر نمایش خواهد داد.

پشتیبانی مرورگرهای جدید از استاندارد Promise

در حال حاضر کروم 32 و نگارش‌های شبانه فایرفاکس، Promise را که جزئی از استاندارد JavaScript شده‌است، به صورت توکار

و بدون نیاز به کتابخانه‌های جانبی، پشتیبانی می‌کنند.

```
if (window.Promise) { // Check if the browser supports Promises
 var promise = new Promise(function(resolve, reject) {
  //asynchronous code goes here
 });
}
```

در اینجا با فراخوانی window.Promise مشخص می‌شود که آیا مرورگر جاری از Promises پشتیبانی می‌کند یا خیر. سپس یک شیء promise ایجاد شده و این شیء توسط پارامترهای resolve و reject که هر دو تابع می‌باشند، کار مدیریت کدهای غیرهمزمان را انجام می‌دهد:

```
if (window.Promise) {
 console.log('Promise found');

 var promise = new Promise(function(resolve, reject) {
      // async
  if (result) {
   resolve(data);
  } else {
   reject('error');
  }
 });

 promise.then(function(data) {
  console.log('Promise fulfilled.');
 }, function(error) {
  console.log('Promise rejected.');
 });
} else {
 console.log('Promise not available');
}
```

در مثال فوق ابتدا یک شیء Promise ایجاد شده است. این شیء استاندارد بوده و با کروم 32 قابل آزمایش است. سپس در callback ابتدایی آن می‌توان یک عملیات AJAX ایی را انجام داد. اگر نتیجه‌ی آن موفقیت آمیز بود، تنها کافی است پارامتر اول این callback را فراخوانی کنیم و اگر خیر، پارامتر دوم آن‌را. برای استفاده از این شیء Promise ایجاد شده، می‌توان از متد then استفاده کرد. این متد نیز در اینجا دو callback پذیرش و رد promise را می‌تواند دریافت کند. برای زنجیر کردن آن کافی است متد then، یک Promise دیگر را بازگشت دهد و از نتیجه‌ی آن در then بعدی استفاده گردد.

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 'firstName the $_appellation_$' 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

عنوان: آموزش PouchDB : قسمت دوم (شروع به کار)
نویسنده: محمد رضا صفری
تاریخ: ۱۳۹۳/۰۲/۱۲ ۲۰:۱۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, NoSQL, PouchDB

در این آموزش هدف ما ایجاد برنامه‌ای بر اساس TodoMVC است که می‌تواند خودش را با یک دیتابیس آنلاین همگام سازی کند.

مطمئن باشید بیشتر از 10 دقیقه وقت شمارا نخواهد گرفت !

نصب PouchDB

فایل index.html را باز کنید و فایل‌های PouchDB را به آن اضافه کنید :

```
<script src="//cdn.jsdelivr.net/pouchdb/2.2.0/pouchdb.min.js"></script>
<script src="js/base.js"></script>
<script src="js/app.js"></script>
```

حالا PouchDB نصب شده و آماده به کار است . (در نسخه نهایی بهتر است از فایل‌های local استفاده کنید)

ایجاد بانک اطلاعاتی

بقیه کارها باید در فایل app.js انجام شود . برای شروع باید بانک اطلاعاتی جدیدی را برای درج اطلاعات خودمان ایجاد کنیم . برای ایجاد بانک اطلاعاتی خیلی ساده یک instance جدید از آبجکت PouchDB می‌سازیم .

```
var db = new PouchDB('todos');
var remoteCouch = false;
```

نیازی نیست که برای بانک خود، نما (Schema) ایجاد کنید! بعد از اینکه اسم را مشخص کنید، بانک آماده به کار است.

ثبت اطلاعات در بانک اطلاعاتی

اولین کاری که باید انجام دهیم این است که یک متد را ایجاد کنیم و توسط آن اطلاعات خودمان را در بانک اطلاعاتی ذخیره کنیم. نام متد را addTodo انتخاب می‌کنیم و کارش این است که وقتی کاربر کلید Enter را فشار داد، اطلاعات را داخل بانک اطلاعاتی ذخیره کند. متد مورد نظر به این صورت هست:

```
function addTodo(text) {
  var todo = {
    _id: new Date().toISOString(),
    title: text,
    completed: false
  };
  db.put(todo, function callback(err, result) {
    if (!err) {
      console.log('Successfully posted a todo!');
    }
  });
}
```

در PouchDB هر پرونده نیاز دارد تا یک فیلد unique با نام _id داشته باشد. اگر داده‌ای بخواهد در بانک اطلاعاتی ثبت شود و دارای _id مشابه باشد، با آن به صورت یک update رفتار خواهد شد. در اینجا ما از تاریخ با عنوان id استفاده کردیم که در این مورد خاص می‌باشد. شما می‌تواند از db.post() نیز استفاده کنید؛ اگر یک id را به صورت random می‌خواهید. تنها چیزی که اجباری است در هنگام ساختن یک پرونده، همین _id است و بقیه موارد کاملا اختیاری هستند.

نمایش اطلاعات

در اینجا ما از یک function کمکی به نام redrawTodosUI استفاده خواهیم کرد که وظیفه‌اش این است تا یک array را دریافت

کرده و آن را هر طور که مشخص کردید نمایش دهد. البته آن را به سلیقه خودتان می‌توانید آماده کنید.
تنها کاری که باید انجام دهیم این است که اطلاعات را از بانک اطلاعاتی استخراج کرده و به function مورد نظر پاس دهیم.
در اینجا می‌توانیم به سادگی از db.allDocs برای خواندن اطلاعات از بانک اطلاعاتی استفاده کنیم.
خاصیت include_docs به PouchDB این دستور را می‌دهد که ما درخواست دریافت همه اطلاعات داخل پرونده‌ها را داریم و descending هم نحوه مرتب سازی را که بر اساس _id هست، مشخص می‌کند تا بتوانیم جدیدترین اطلاعات را اول دریافت کنیم .

```
function showTodos() {
  db.allDocs({include_docs: true, descending: true}, function(err, doc) {
    redrawTodosUI(doc.rows);
  });
}
```

به روزرسانی خودکار

هر بار که ما اطلاعات جدیدی را در بانک اطلاعاتی وارد می‌کنیم، نیازمند این هستیم تا اطلاعات جدید را به صورت خودکار دریافت و نمایش دهیم. برای این منظور می‌توانیم به روش زیر عمل کنیم :

```
var remoteCouch = false;

db.info(function(err, info) {
  db.changes({
    since: info.update_seq,
    live: true
  }).on('change', showTodos);
});
```

حالا هر بار که اطلاعات جدیدی در بانک اطلاعات ثبت شود، به صورت خودکار نمایش داده خواهد شد. خاصیت live مشخص می‌کند که این کار می‌تواند بی نهایت بار انجام شود.

ویرایش اطلاعات

وقتی که کاربر یک آیتم Todo را کامل می‌کند، یک چک باکس را علامت می‌زند و اعلام می‌کند که این کار انجام شده است.

```
function checkboxChanged(todo, event) {
  todo.completed = event.target.checked;
  db.put(todo);
}
```

این بخش شبیه به قسمت ثبت اطلاعات است، با این تفاوت که باید شامل یک فیلد _rev (اضافه بر _id) نیز باشد. در غیر اینصورت تغییرات ثبت نخواهد شد. این کار برای این است که اشتباهی، اطلاعاتی در بانک ثبت نشود.

حذف اطلاعات

برای حذف اطلاعات باید از متد db.remove به همراه آبجکت مورد نظر استفاده کنید .

```
function deleteButtonPressed(todo) {
  db.remove(todo);
}
```

مانند بخش ویرایش نیز باید در اینجا هم _id و هم _rev مشخص شود.
باید توجه داشته باشید در اینجا همان آبجکتی را که از بانک فراخوانی کرده‌ایم، به این متد پاس می‌دهیم.
البته شما می‌توانید آبجکت خودتان را نیز ایجاد کنید {id: todo._id, _rev: todo._rev_} اما خوب همان روش قبلی عاقلانه‌تر است و احتمال خطای کمتری دارد .

نصب CouchDB

حالا می‌خواهیم همگام سازی را اجرا کنیم و برای این کار نیاز هست یا CouchDB را به صورت Local نصب کنیم یا از سرویس‌های آنلاین مثل IrisCouch استفاده کنید .

فعال سازی CORS

برای اینکه به صورت مستقیم با CouchDB در ارتباط باشید باید مطمئن شوید که CORS فعال هست.

در اینجا فقط نام کاربری و رمز ورود را مشخص کنید. به صورت پیش فرض CouchDB به صورت Admin Party نصب می‌شود و نیازی به User و password ندارد. مگر اینکه برایش مشخص کنید.

همچنین شما باید myname.iriscouch.com را با سرور خودتان جایگزین کنید که اگر به صورت local کار می‌کنید 127.0.0.1:5984 است.

```
$ export HOST=http://username:password@myname.iriscouch.com
$ curl -X PUT $HOST/_config/httpd/enable_cors -d '"true"'
$ curl -X PUT $HOST/_config/cors/origins -d '"*"'
$ curl -X PUT $HOST/_config/cors/credentials -d '"true"'
$ curl -X PUT $HOST/_config/cors/methods -d '"GET, PUT, POST, HEAD, DELETE"'
$ curl -X PUT $HOST/_config/cors/headers -d \
  '"accept, authorization, content-type, origin"'
```

راه اندازی ارتباط ساده دو طرفه

به فایل app.js برگردید . در اینجا باید آدرس بانک اطلاعاتی آنلاین را مشخص کنیم.

```
var db = new PouchDB('todos');
var remoteCouch = 'http://user:pass@mname.iriscouch.com/todos';
```

فراموش نکنید که نام کاربری و رمز ورود را بسته به نیاز خود تغییر دهید.

حالا می‌تونیم function که وظیفه همگام سازی اطلاعات را به عهده دارد بنویسیم :

```
function sync() {
  syncDom.setAttribute('data-sync-state', 'syncing');
  var opts = {live: true};
  db.replicate.to(remoteCouch, opts, syncError);
  db.replicate.from(remoteCouch, opts, syncError);
}
```

db.replicate() به PouchDB می‌گوید که که همه اطلاعات را "به" یا "از" remoteCouch انتقال دهد.

ما دوبار این درخواست را دادیم. در بار اول اطلاعات به داخل سرور ریموت منتقل می‌شود و در بار دوم اطلاعات local جایگزین می‌شوند.

یک callback هم وقتی که این کار به پایان برسد اجرا خواهد شد .

می‌توانید برنامه خود را در دو مرورگر مختلف اجرا کنید تا نتیجه کار را ببینید.

تبریک می‌گوییم !

شما توانستید اولین برنامه خود را توسط PouchDB ایجاد کنید. البته این یک برنامه ساده بود و در دنیای واقعی نیاز هست تا خیلی کارهای بیشتری را انجام دهید. اما باز هم اصول اولیه کار را یاد گرفتید و ادامه راه را می‌توانید تنهایی ادامه دهید .

عنوان: **نمایش تاریخ شمسی توسط JavaScript در AngularJS**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۳/۰۴/۱۱ ۱۷:۵۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Persian, AngularJS, momentjs

در برنامه‌های مبتنی بر وب رایج، معمولا تبدیل تاریخ میلادی به شمسی در سمت سرور انجام می‌گیرد و تاریخ شمسی حاصل از تبدیل، به کاربر نمایش داده می‌شود. اما در برنامه‌های Single Page و یا به اختصار SPAها که کلاینت فقط با یک سری داده به فرمت JSON درگیر است، برای نمایش تاریخ شمسی به چه طریقی باید عمل کرد؟ آیا باید تاریخ را در سمت سرور به فرمت مورد نظر تبدیل کرد و یا در سمت کلاینت؟ همه‌ی این‌ها از جمله سوالاتی هست که به هنگام توسعه‌ی SPAها با آن‌ها حتما درگیر خواهید شد.

شاید بتوان گفت که در SPA ها، هدف این است که از بار سرور تا حد ممکن کم کرد و آن را در بین کلاینت‌ها توزیع کرد. در SPAها نقش اصلی سرور تامین داده هاست و بیشتر پردازش‌ها در صورت امکان در سمت کلاینت انجام می‌شود و می‌بینید که حتی رندر کردن HTML نیز به عهده‌ی قالب‌های سمت کلاینت است. البته هنوز هم می‌توان قبل از اینکه داده را به فرمت JSON سریالایز کرد، سمت سرور بر روی آن‌ها پیمایش انجام داده و تاریخ‌های میلادی را به شمسی تبدیل کرد که هدف ما این نیست و می‌خواهیم این کار را بر عهده‌ی مرورگر کاربر قرار دهیم. **معرفی moment.js**

برای کار با داده‌هایی از جنس تاریخ در سمت کلاینت، کتابخانه‌ی جاوا اسکریپتی قدرتمندی به نام moment.js وجود دارد. این کتابخانه دارای انواع و اقسام API برای نمایش و پردازش تاریخ هست. حتی می‌تواند relative time را نیز نمایش دهد. منظور از relative time این هست که به جای نمایش تاریخ، اختلاف آن را با زمان حال نمایش دهد. برای مثال می‌نویسند فلان پست در دو ساعت پیش ارسال شده و زمان دقیق ارسال پست را نمایش نمی‌دهد.

خوشبختانه برای افزودن تاریخ شمسی به این کتاب خانه، افزونه‌ای به نام moment-jalaali برای آن تدارک دیده شده است. کار با آن نیز بسیار راحت است. کافی است در همان API هایی که برای فرمت کردن تاریخ در moment.js استفاده می‌کردید؛ یک j در ابتدای آن‌ها قرار دهید که مثال‌های کامل استفاده از آن را در مستندات آن می‌توانید مشاهده کنید.

نحوه‌ی استفاده از moment.js در AngularJS و ASP.NET

در ASP.NET فیلد هایی که از جنس DateTime هستند به شکل زیر به فرمت JSON سریالایز می‌شوند:

```
\/Date(1374222094520)\/
```

در moment.js احتیاج به کدنویسی برای parse کردن این نوع فرمت و تبدیل کردن آن به تاریخ وجود ندارد؛ چرا که moment.js به صورت تو کار از این نوع فرمت نیز پشتیبانی می‌کند و احتیاجی به کار اضافه‌تر نیست.

```
moment("/Date(1198908717056-0700)/"); // December 28 2007 10:11 PM
```

در AngularJS هر گاه قصد داشته باشیم که فرمت نمایش داده‌ها را تغییر دهیم از filterها استفاده می‌کنیم. برای مثال فیلتر uppercase داده name را با حروف بزرگ نمایش می‌دهد.

```
{{ name | uppercase }}
```

حال برای تاریخ نیز می‌خواهیم چنین کاری انجام دهیم؛ بدین صورت که یک فیلتر سفارشی به شکل زیر تعریف کرده تا تاریخ میلادی را به صورت شمسی و با فرمت دلخواهی که می‌خواهیم نمایش دهد:

```
{{post.date | jalaliDate:'jYYYY/jMM/jDD hh:mm' }}
```

تعریف فیلتر jalaliDate نیز به شکل زیر است:

```
app.filter('jalaliDate', function () {
            return function (inputDate, format) {
                var date = moment(inputDate);
                return date.fromNow() + " " + date.format(format);
```

```
        }
    });
```

خروجی این فیلتر نیز به شکل "4 ماه پیش 07/12/1392 03:10" است و مشاهده می‌کنید که به کمک filterها در AngularJS انجام این گونه از کارها بسیار ساده و لذت بخش است.

توجه کنید که این فقط یک ایده‌ی ابتدایی و ساده از پیاده سازی فیلتر فوق است. قطعا با کمک APIهای متنوع momentjs و پارامترهای ورودی فیلتر، می‌توان فیلتری بسیار پیشرفته‌تر تعریف کرد. **دریافت کدهای یک مثال پیاده سازی شده با استفاده از کدهای فوق** moment-jalali-AngularJs.rar

عنوان: **توسعه سرویس‌های Angular به روش OOP**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۶/۱۴ ۲۳:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, OOP, AngularJS

یک نکته‌ای که در توسعه سیستم‌ها و نرم افزارها تاکید فراوانی به آن می‌شود استفاده مجدد از کدهای نوشته شده قبلی است. یعنی تا جای ممکن باید ساختار پروژه به گونه‌ای نوشته شود که از تکرار کدها در جای جای پروژه جلوگیری شود. این مورد به خوبی در زبان‌های شیءگرا نظیر C# رعایت می‌شود اما در پروژه‌هایی که مبتنی بر Javascript هستند نظیر angular، باید با استفاده از خاصیت prototype جاوا اسکریپ این مورد را رعایت نمود. در مقاله Dr. Axel Rauschmayer، قدم به قدم و به خوبی روش‌های وراثت در Javascript توضیح داده شده است.

در این پست با روش‌های وراثت در کنترلرهای انگولاری آشنا شدید. این وراثت محدود به ارث بری scopeها می‌شود. اما یکی از بخش‌های بسیار مهم پروژه‌های انگولار نوشتن سرویس‌هایی با قابلیت توسعه مجدد در سایر بخش‌های پروژه می‌باشد. معادل آن، مفهوم Overriding در OOP است. با ذکر مثالی این مورد را با هم بررسی خواهیم کرد.
ابتدا یک سرویس به نام BaseService ایجاد کنید:

```
angular.module('myApp').service('BaseService', function() {

    var BaseService = function(title) {
        this.title = title;
    };

    BaseService.prototype.getMessage = function() {
        var self = this;
        return 'Hello ' + self.title;
    };

    return BaseService;
});
```

سرویس بالا دارای سازنده‌ای است که مقدار title باید در اختیار آن قرار گیرد. با استفاده از خاصیت prototype تابعی تعریف می‌کنیم که این تابع خروجی مورد نظر را برای ما تامین خواهد نمود.
حال اگر ماژول و کنترلری جهت نمایش خروجی به صورت زیر ایجاد کنیم:

```
var app= angular.module('myApp', []);


app.controller('myCtrl', function ($scope,BaseService) {

    var instance = new BaseService('Masoud');
    $scope.title = instance.getMessage();

});
```

با کدهای Html زیر:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" ng-app="myApp">
<head>
    <title></title>
</head>
    <body ng-controller="myCtrl">

        <div>
            {{title}}
        </div>

    </body>
<script src="Scripts/jquery-2.1.1.min.js"></script>
<script src="Scripts/angular.js"></script>
<script src="App/app.js"></script>
```

```
</html>
```

در نهایت خروجی به صورت زیر قابل مشاهده است:

تا اینجای کار روال معمول تعاریف سرویس در انگولار بوده است. اما قصد داریم سرویس جدیدی را ایجاد نمایم تا خروجی سرویس قبلی را اندکی تغییر دهد. به جای اینکه سرویس قبلی را تغییر دهیم یا بدتر از آن سرویس جدیدی بسازیم و کدهای قبلی را در آن کپی کنیم کافیست به صورت زیر عمل نماییم:

```
app.service('ExtService', function(BaseService) {

    var ExtService = function() {
        BaseService.apply(this, arguments);
    };

    ExtService.prototype = new BaseService();

    ExtService.prototype.getMessage = function() {
        var self = this;
        return BaseService.prototype.getMessage.apply(this, arguments) + ' From Ext Service';

    };

    return ExtService;
});
```

حال می‌توان کنترلر را به صورت زیر بازنویسی کرد.

```
app.controller('myCtrl', function ($scope,BaseService , ExtService) {

    var baseInstance = new BaseService('Masoud');
    var extInstance = new ExtService('Dotnettips');
    $scope.title = baseInstance.getMessage() + ' and ' + extInstance.getMessage();

});
```

در کنترلر بالا هر دو سرویس تزریق شده‌اند. خروجی سرویس دوم متن From Ext Service را نیز به همراه خواهد داشت. پس از اجرای برنامه خروجی زیر قابل مشاهده است:

localhost:2208/index.html
Hello Masoud and Hello Dotnettips From Ext Service

عنوان: معرفی Kendo UI
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۴ ۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

Kendo UI چیست؟

Kendo UI یک فریم ورک جاوا اسکریپتی ساخت برنامه‌های مدرن و تعاملی وب است و برای رسیدن به این مقصود، از HTML 5، CSS 3، JavaScript و jQuery کمک می‌گیرد.

امکانات فراهم شده توسط Kendo UI

1) انواع و اقسام ویجت‌ها: کنترل‌های وب تهیه شده برفراز jQuery
ویجت‌های آن در سه گروه کلی قرار می‌گیرند:
- گروه وب، مانند گرید، tree-view و غیره.
- گروه DataViz که جهت نمایش بصری اطلاعات و ترسیم انواع و اقسام نمودارها کاربرد دارد.
- گروه موبایل که با استفاده از فناوری adaptive rendering، در سیستم عامل‌های مختلف موبایل، مانند اندروید و آی او اس، ظاهری بومی و هماهنگ با آن‌ها را ارائه می‌دهد.

2) منبع داده سمت کاربر (Client side data source)
منبع داده سمت کاربر Kendo UI، از انواع و اقسام منابع داده محلی مانند آرایه‌های جاوا اسکریپتی تا منابع داده راه دور، مانند XML، JSON و JSONP، جهت نمایش اطلاعات و data binding پشتیبانی می‌کند. این منبع داده، مواردی مانند صفحه بندی، مرتب سازی اطلاعات و گروه بندی آن‌ها را نیز فراهم می‌کند. به علاوه با عملیات ثبت، ویرایش و حذف اطلاعات نیز هماهنگی کاملی را دارد.

3) به همراه یک فریم ورک MVVM توکار است
این فریم ورک MVVM مواردی مانند two way data binding و همچنین declarative binding را نیز پشتیبانی می‌کند.

4) امکان تعویض قالب
5) پویا نمایی، کشیدن و رها کردن
6) فریم ورک اعتبارسنجی

چرا Kendo UI؟

- مهم‌ترین مزیت کار با Kendo UI، فراهم آوردن تمام نیازهای توسعه‌ی یک برنامه‌ی مدرن وب، تنها در یک بسته است. به این ترتیب دیگر نیازی نیست تا گرید را از یک‌جا، tree-view را از جایی دیگر و کتابخانه‌های رسم نمودار را از منبعی ناهمگون با سایر عناصر برنامه دریافت و استفاده کنید؛ در اینجا تمام این‌ها در قالب یک بسته‌ی آماده برای شما فراهم شده‌است و همچنین با یکدیگر سازگاری کاملی دارند.
- تمام ویجت‌های آن برای نمایش سریع با کارآیی بالا طراحی شده‌اند.
- پشتیبانی خوب آن. این فریم ورک محصول شرکتی است که به صورت تخصصی کار تهیه کامپوننت‌های وب و دسکتاپ را انجام می‌دهد.

مرورگرهای پشتیبانی شده

یکی دیگر از مزایای مهم کار با Kendo UI پشتیبانی گسترده‌ی آن از اکثر مرورگرهای موجود است. این فریم ورک با مرورگرهای زیر سازگار است:
- IE 7 به بعد
- فایرفاکس 10 به بعد
- تمام نگارش‌های کروم
- اپرا 10 به بعد

- سفاری 4 به بعد

مجوز استفاده از Kendo UI

Kendo UI با سه مجوز ذیل ارائه می‌شود:

- 30 روزه آزمایشی رایگان
- تجاری
- سورس باز با مجوز Apache

پیشتر نسخه‌ی تجاری آن تحت مجوز GPL نیز در دسترس بود. اما اخیرا مجوز GPL آن حذف شده و به Apache تغییر یافته است. اما باید در نظر داشت که نسخه‌ی سورس باز آن شامل کنترل‌های مهمی مانند «گرید» نیست و این موارد تنها در نسخه‌ی تجاری آن لحاظ شده‌اند.

مثال‌های Kendo UI

پس از دریافت بسته‌ی کامل آن، پوشه‌هایی مانند styles، js و امثال آن قابل مشاهده هستند؛ به همراه پوشه‌ی examples آن که حداقل 86 پوشه‌ی دیگر در آن جهت ارائه مثال‌هایی از نحوه‌ی کاربرد المان‌های مختلف آن تدارک دیده شده‌اند.

نحوه‌ی افزودن Kendo UI به صفحه

از آنجائیکه Kendo UI یک فریم ورک جاوا اسکریپتی است، همانند سایر برنامه‌های وب، افزودن تعاریف فایل‌های js، css و تصاویر مرتبط با آن، برای شروع به کار کفایت می‌کند. برای این منظور ابتدا پوشه‌های js و styles بسته‌ی دریافتی آن‌را به برنامه‌ی خود اضافه کنید (این پوشه‌ها در فایل پیوست انتهای بحث موجود هستند).

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>

    <!--KendoUI: Web-->
    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.web.min.js" type="text/javascript"></script>

    <!--KendoUI: DataViz-->
    <link href="styles/kendo.dataviz.min.css" rel="stylesheet" type="text/css" />
    <script src="js/kendo.dataviz.min.js" type="text/javascript"></script>

    <!--KendoUI: Mobile-->
    <link href="styles/kendo.mobile.all.min.css" rel="stylesheet" type="text/css" />
    <script src="js/kendo.mobile.min.js" type="text/javascript"></script>

    <script type="text/javascript">
        $(function() {
            $("#pickDate").kendoDatePicker();
        });
    </script>
</head>
<body>
    <span>
        Pick a date: <input id="pickDate" type="text"/>
    </span>
</body>
</html>
```

در اینجا یک مثال ساده‌ی استفاده از date picker کندو یو آی را ملاحظه می‌کنید. در قسمت head صفحه، نحوه‌ی ثبت سه گروه اسکریپت و شیوه نامه، مشخص شده‌اند. اگر نیاز به کامپوننت‌های وب آن‌را دارید باید اجزایی مانند kendo.common.min.css، jquery.min.js، kendo.default.min.css و kendo.web.min.js به صفحه اضافه شوند. اگر نیاز به رسم نمودار هست، فایل‌ها kendo.dataviz.min.css و kendo.dataviz.min.js باید تعریف شوند و برای فعال سازی اجزای موبایل آن فایل‌های kendo.mobile.all.min.css و kendo.mobile.min.js نیاز است به صفحه پیوست شوند. در هر سه حالت ذکر jquery.min.js الزامی است.

دریافت سورس کامل این قسمت که حاوی فایل‌های اصلی kendoui.professional.2014.2.1008 نیز می‌باشد:

KendoUI01.7z

نظرات خوانندگان

نویسنده: محمد
تاریخ: ۱۳۹۳/۰۸/۱۴ ۱۴:۳۸

برای دریافت پکیج کامل Kendo از این آدرس استفاده کنید.

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۰۸/۱۴ ۲۰:۵

آیا امکان استفاده از Razor Wrapper هم به صورت رایگان وجود دارد یا اینکه نیازمند تهیه بسته کامل می‌باشد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۴ ۲۰:۱۸

تجاری هست و توصیه هم نمی‌شود. چون نهایتا برای بسیاری از کارها باید به پشت صحنه‌ی این ویجت‌ها و امکانات مراجعه کنید؛ یعنی نیاز است مستقیما اسکریپت نویسی کنید و با ساختار واقعی آن‌ها آشنا باشید.

نویسنده: سعیدجلالی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۸:۵۶

بله امکان استفاده از wrapper در نسخه asp.net.mvc.commercial وجود دارد
استفاده از اون هم خیلی ساده‌تر و خواناتر از جاوا اسکریپت هست. نظر آقای نصیری هم محترم است ولی در مواقع خواص میتونید همزمان هم از جاوااسکریپت استفاده کنید هم از wrapper یک نمونه رو در زیر با هم مقایسه میکنیم
با استفاده از جاوا اسکریپت

```
<input id="pickDate" type="text"/>

<script type="text/javascript">
  $(function() {
        $("#pickDate").kendoDatePicker();
  });
</script>
```

با استفاده از wrapper

```
@(Html.Kendo().DatePicker().Name("pickDate"))
```

در ضمن اینکه توی wrapper امکان استفاده از Intellisense و امکان تعریف ارتباط اغلب کامپوننت‌های وب به مدل با استفاده از forهای نمونه معادل کامپوننت فراهم شده است مانند wrapper زیر

```
@(Html.Kendo().DatePickerFor(m => m.HireDate).Name("pickDate1"))
```

نویسنده: Ara
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۹:۳۱

به دوستان پیشنهاد می‌شه این رو هم ببینید
استفاده از Kendo UI بهمراه AngularJs خیلی خوبه !
ما تو پروژه تجاری تجاری استفاده کردیم و خیلی راضی هستیم

عنوان: بررسی ساختار ویجت‌های وب Kendo UI
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۴ ۱۴:۱۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

ویجت‌های وب Kendo UI کدامند؟

ویجت‌های وب Kendo UI مجموعه‌ای از کنترل‌های سفارشی HTML 5 هستند که برفراز jQuery تهیه شده‌اند. این کنترل‌ها برای برنامه‌های وب و همچنین برنامه‌های دسکتاپ لمسی طراحی شده‌اند.
بهترین روش برای مشاهده‌ی این مجموعه، مراجعه به فایل examples\index.html پوشه‌ی اصلی Kendo UI است که لیست کاملی از این ویجت‌ها را به همراه مثال‌های مرتبط ارائه می‌دهد. تعدادی از اعضای این مجموعه شامل کنترل‌های ذیل هستند:

Window, TreeView, Tooltip, ToolBar, TimePicker, TabStrip, Splitter, Sortable, Slider, Gantt, Scheduler, ProgressBar, PanelBar, NumericTextBox, Notification, MultiSelect, Menu, MaskedTextBox, ListView, PivotGrid, Grid, Editor, DropDownList, DateTimePicker, DatePicker, ComboBox, ColorPicker, Calendar, Button, AutoComplete

نحوه‌ی استفاده کلی از ویجت‌های وب Kendo UI

با توجه به اینکه کنترل‌های Kendo UI مبتنی بر jQuery هستند، نحوه‌ی استفاده از آن‌ها، مشابه سایر افزونه‌های جی‌کوئری است.
ابتدا المانی به صفحه اضافه می‌شود:

```
<input id="pickDate" type="text"/>
```

سپس این المان را در رویداد document ready، به یکی از کنترل‌های Kendo UI مزین خواهیم کرد. برای مثال تزئین یک TextBox معمولی با یک Date Picker:

```
<script type="text/javascript">
  $(function() {
        $("#pickDate").kendoDatePicker();
  });
</script>
```

روش دیگری به نام declarative initialization نیز برای اعمال ویجت‌های وب Kendo UI قابل استفاده است که از ویژگی‌های data-* مرتبط با HTML 5 کمک می‌گیرد. برای نمونه، کدهای جاوا اسکریپتی فوق را می‌توان با ویژگی data-role ذیل جایگزین کرد:

```
<input id="dateOfBirth" type="text" data-role="datepicker" />
```

اگر در این حالت برنامه را اجرا کنید، تفاوتی را مشاهده نخواهید کرد.
برای فعال سازی حالت declarative initialization باید به دو نکته‌ی مهم دقت داشت:
الف) در مطلب معرفی Kendo UI اسکریپت‌های ذیل برای آماده سازی Kendo Ui معرفی شدند:

```
<!--KendoUI: Web-->
<link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
<link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
<script src="js/jquery.min.js" type="text/javascript"></script>
<script src="js/kendo.web.min.js" type="text/javascript"></script>

<!--KendoUI: DataViz-->
<link href="styles/kendo.dataviz.min.css" rel="stylesheet" type="text/css" />
<script src="js/kendo.dataviz.min.js" type="text/javascript"></script>

<!--KendoUI: Mobile-->
<link href="styles/kendo.mobile.all.min.css" rel="stylesheet" type="text/css" />
<script src="js/kendo.mobile.min.js" type="text/javascript"></script>
```

باید دقت داشت که در آن واحد نمی‌توان تمام این بسته‌ها را با هم بکار برد؛ چون برای مثال فایل‌های جداگانه ویجت‌های وب و موبایل با هم تداخل ایجاد می‌کنند . بجای اینکار بهتر است از فایل‌های kendo.all.min.js (که حاوی تمام اسکریپت‌های لازم است) و cssهای عنوان شده استفاده کرد:

```
<link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
<link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
<script src="js/jquery.min.js" type="text/javascript"></script>
<script src="js/kendo.all.min.js" type="text/javascript"></script>
```

ب) data-roleها توسط متد kendo.init فعال می‌شوند.

یک مثال کامل:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>

    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.all.min.js" type="text/javascript"></script>

    <script type="text/javascript">
        $(function () {
            $("#pickDate").kendoDatePicker();
        });

        $(function () {
            // initialize any widgets in the #container div
            kendo.init($("#container"));
        });
    </script>
</head>
<body>
    <span>
        Pick a date: <input id="pickDate" type="text" />
    </span>

    <div id="container">
        <input id="dateOfBirth" type="text" data-role="datepicker" />
        <div id="colors"
             data-role="colorpalette"
             data-columns="4"
             data-tile-size="{ width: 34, height: 19 }"></div>
    </div>
</body>
</html>
```

- در این مثال نحوه‌ی پیوست تمام فایل‌های لازم Kendo UI را به صورت یکجا ملاحظه می‌کنید که در ابتدای head صفحه ذکر شده‌اند.
- در اینجا pickDate به صورت معمولی فعال شده‌است.
- اما در قسمت kendo.init نام یک ناحیه یا نام یک کنترل را می‌توان ذکر کرد. برای مثال در اینجا کل ناحیه‌ی مشخص شده توسط یک div با id مساوی container به صورت یکجا با تمام کنترل‌های داخل آن فعال گردیده‌است.

بنابراین برای اعمال declarative initialization، یک ناحیه را توسط kendo.init مشخص کرده و سپس توسط data-roleها، نام ویجت وب مورد نظر را به صورت lower case مشخص می‌کنیم. همچنین فایل‌های اسکریپت مورد استفاده نیز نباید تداخلی داشته باشند.

تنظیمات ویجت‌های وب Kendo UI

تاکنون نمونه‌ی ساده‌ای از بکارگیری ویجت‌های وب Kendo UI را بررسی کردیم؛ اما این ویجت‌ها توسط تنظیمات پیش بینی شده برای آن‌ها بسیار قابل تنظیم و تغییر هستند. تنظیمات آن‌ها نیز بستگی به روش استفاده و آغاز آن‌ها دارد. برای مثال اگر این

ویجت‌ها را توسط کدهای جاوا اسکریپتی آغاز کرده‌اید، در همانجا توسط پارامترهای افزونه‌ی جی‌کوئری می‌توان تنظیمات مرتبط را اعمال کرد:

```
<script type="text/javascript">
  $(function () {
      $("#pickDate").kendoDatePicker({
              format: "yyyy/MM/dd"
        });
  });
</script>
```

که در اینجا توسط پارامتر format، نحوه‌ی دریافت تاریخ نهایی مشخص می‌شود.

در حالت declarative initialization، پارامتر format تبدیل به ویژگی data-format خواهد شد:

```
<input id="dateOfBirth" type="text"
          data-role="datepicker"
          data-format="yyyy/MM/dd" />
```

تنظیمات DataSource ویجت‌های وب

بسیاری از ویجت‌های وب Kendo UI با داده‌ها سر و کار دارند مانند Grid، Auto Complete، Combo box و غیره. این کنترل‌ها داده‌های خود را از طریق خاصیت DataSource دریافت می‌کنند. برای نمونه در اینجا یک combo box را در نظر بگیرید. در مثال اول، خاصیت dataSource کنترل ComboBox در همان افزونه‌ی جی‌کوئری تنظیم شده‌است:

```
<input id="colorPicker1" />
    <script type="text/javascript">
        $(document).ready(function () {
            $("#colorPicker1").kendoComboBox({
                dataSource: ["Blue", "Green", "Red", "Yellow"]
            });
        });
    </script>
```

و در مثال دوم، نحوه‌ی مقدار دهی ویژگی data-source را در حالت declarative initialization مشاهده می‌کنید. همانطور که عنوان شد، در این حالت ذکر متد kendo.init بر روی یک ناحیه و یا یک کنترل ویژه، جهت آغاز فعالیت آن ضروری است:

```
<input id="colorPicker2" data-role="combobox" data-source='["Blue", "Green", "Red", "Yellow"]' />
    <script type="text/javascript">
        $(document).ready(function () {
            kendo.init($("#colorPicker2"));
        });
    </script>
```

کار با رویدادهای ویجت‌های وب

نحوه‌ی کار با رویدادهای ویجت‌های وب نیز بر اساس نحوه‌ی آغاز آن‌ها متفاوت است. در مثال‌های ذیل، دو حالت متفاوت تنظیم رویداد change را توسط خواص افزونه‌ی جی‌کوئری:

```
<input id="colorPicker3" />
    <script type="text/javascript">
        function onColorChange(e) {
             alert('Color Change!');
        }

        $(document).ready(function () {
            $("#colorPicker3").kendoComboBox({
                dataSource: ["Blue", "Green", "Red", "Yellow"],
                change: onColorChange
```

```
            });
        });
    </script>
```

و همچنین توسط ویژگی data-change مشاهده می‌کنید:

```
<input id="colorPicker4" data-role="combobox"
           data-source='["Blue", "Green", "Red", "Yellow"]'
           data-change="onColorChange" />

    <script type="text/javascript">
        function onColorChange(e) {
            alert('Color Change!');
        }

        $(document).ready(function () {
            kendo.init($("#colorPicker4"));
        });
    </script>
```

در هر دو حالت، انتخاب یک گزینه‌ی جدید combo box، سبب فراخوانی متد callback ایی به نام onColorChange می‌شود.

تغییر قالب ویجت‌های وب

Kendo UI همیشه یک جفت CSS را جهت تعیین قالب‌های ویجت‌های خود، مورد استفاده قرار می‌دهد. برای نمونه در مثال‌های فوق، kendo.common.min.css حاوی اطلاعات محل قرارگیری و اندازه‌ی ویجت‌ها است. شیوه نامه‌ی دوم همیشه به شکل kendo.[skin].min.css تعریف می‌شود که دارای اطلاعات رنگ و پس زمینه‌ی ویجت‌ها خواهد بود؛ مانند kendo.black.min.css، kendo.blueopal.min.css و امثال آن که در پوشه‌ی styles قابل مشاهده هستند.
همچنین باید دقت داشت که همیشه common باید پیش از skin ذکر شود؛ زیرا در تعدادی از حالات، شیوه نامه‌ی skin، اطلاعات common را بازنویسی می‌کند.
علاوه بر skinهای پیش فرض موجود در پوشه‌ی styles، امکان استفاده از یک theme builder آنلاین نیز وجود دارد: kendo-ui-themebuilder

نظرات خوانندگان

نویسنده: انصاری
تاریخ: ۱۳۹۳/۰۹/۰۱ ۸:۵۹

امکان تبدیل تاریخ (از میلادی به شمسی) در ویجت‌های وب مثل dataPicker , Calendar , Scheduler وجود دارد؟ من میخواستم از Scheduler استفاده کنم ولی با تاریخ و فرمت (ماه و روزهای هفته) فارسی، به نظر شما امکانش هست بدون دردسر (و به شکل استاندارد) این تقویم رو شمسی کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۱ ۱۰:۵

در اینجا بحث شده.

عنوان: کار با Kendo UI DataSource
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۵ ۲۲:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, Kendo UI, json.net

Kendo UI DataSource جهت تامین داده‌های سمت کلاینت ویجت‌های مختلف KendoUI طراحی شده‌است و به عنوان یک اینترفیس استاندارد قابل استفاده توسط تمام کنترل‌های داده‌ای Kendo UI کاربرد دارد. Kendo UI DataSource امکان کار با منابع داده محلی، مانند اشیاء و آرایه‌های جاوا اسکریپتی و همچنین منابع تامین شده از راه دور، مانند JSON، JSONP و XML را دارد. به علاوه توسط آن می‌توان اعمال ثبت، ویرایش و حذف اطلاعات، به همراه صفحه بندی، گروه بندی و مرتب سازی داده‌ها را کنترل کرد.

استفاده از منابع داده محلی

در ادامه مثالی را از نحوه‌ی استفاده از یک منبع داده محلی جاوا اسکریپتی، مشاهده می‌کنید:

```
<script type="text/javascript">
        $(function () {
            var cars = [
                { "Year": 2000, "Make": "Hyundai", "Model": "Elantra" },
                { "Year": 2001, "Make": "Hyundai", "Model": "Sonata" },
                { "Year": 2002, "Make": "Toyota", "Model": "Corolla" },
                { "Year": 2003, "Make": "Toyota", "Model": "Yaris" },
                { "Year": 2004, "Make": "Honda", "Model": "CRV" },
                { "Year": 2005, "Make": "Honda", "Model": "Accord" },
                { "Year": 2000, "Make": "Honda", "Model": "Accord" },
                { "Year": 2002, "Make": "Kia", "Model": "Sedona" },
                { "Year": 2004, "Make": "Fiat", "Model": "One" },
                { "Year": 2005, "Make": "BMW", "Model": "M3" },
                { "Year": 2008, "Make": "BMW", "Model": "X5" }
            ];

            var carsDataSource = new kendo.data.DataSource({
                data: cars
            });

            carsDataSource.read();
            alert(carsDataSource.total());
        });
    </script>
```

در اینجا cars آرایه‌ای از اشیاء جاوا اسکریپتی بیانگر ساختار یک خودرو است. سپس برای معرفی آن به Kendo UI، کار با مقدار دهی خاصیت data مربوط به new kendo.data.DataSource شروع می‌شود.
ذکر new kendo.data.DataSource به تنهایی به معنای مقدار دهی اولیه است و در این حالت منبع داده مورد نظر، استفاده نخواهد شد. برای مثال اگر متد total آن‌را جهت یافتن تعداد عناصر موجود در آن فراخوانی کنید، صفر را بازگشت می‌دهد. برای شروع به کار با آن، نیاز است ابتدا متد read را بر روی این منبع داده مقدار دهی شده، فراخوانی کرد.

استفاده از منابع داده راه دور

در برنامه‌های کاربردی، عموما نیاز است تا منبع داده را از یک وب سرور تامین کرد. در اینجا نحوه‌ی خواندن اطلاعات JSON بازگشت داده شده از جستجوی توئیتر را مشاهده می‌کنید:

```
<script type="text/javascript">
        $(function () {
            var twitterDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "http://search.twitter.com/search.json",
                        dataType: "jsonp",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET',
                        data: { q: "#kendoui" }
                    },
                    schema: { data: "results" }
```

```
                },
                error: function (e) {
                    alert(e.errorThrown.stack);
                }
            });
        });
    </script>
```

در قسمت transport، جزئیات تبادل اطلاعات با سرور راه دور مشخص می‌شود؛ برای مثال url ارائه دهنده‌ی سرویس، dataType بیانگر نوع داده مورد انتظار و data کار مقدار دهی پارامتر مورد انتظار توسط سرویس توئیتر را انجام می‌دهد. در اینجا چون صرفا عملیات خواندن اطلاعات صورت می‌گیرد، خاصیت read مقدار دهی شده‌است.
در قسمت schema مشخص می‌کنیم که اطلاعات JSON بازگشت داده شده توسط توئیتر، در فیلد results آن قرار دارد.

کار با منابع داده OData

علاوه بر فرمت‌های یاد شده، Kendo UI DataSource امکان کار با اطلاعاتی از نوع OData را نیز دارا است که تنظیمات ابتدایی آن به صورت ذیل است:

```
<script type="text/javascript">
            var moviesDataSource = new kendo.data.DataSource({
                type: "odata",
                transport: {
                    read: "http://demos.kendoui.com/service/Northwind.svc/Orders"
                },
                error: function (e) {
                    alert(e.errorThrown.stack);
                }
            });
        });
    </script>
```

همانطور که ملاحظه می‌کنید، تنظیمات ابتدایی آن اندکی با حالت remote data پیشین متفاوت است. در اینجا ابتدا نوع داده‌ی بازگشتی مشخص می‌شود و در قسمت transport، خاصیت read آن، آدرس سرویس را دریافت می‌کند.

یک مثال: دریافت اطلاعات از ASP.NET Web API

یک پروژه‌ی جدید ASP.NET را آغاز کنید. تفاوتی نمی‌کند که Web forms باشد یا MVC؛ از این جهت که مباحث Web API در هر دو یکسان است.
سپس یک کنترلر جدید Web API را به نام ProductsController با محتوای زیر ایجاد کنید:

```
using System.Collections.Generic;
using System.Web.Http;

namespace KendoUI02
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
    }

    public class ProductsController : ApiController
    {
        public IEnumerable<Product> Get()
        {
            var products = new List<Product>();
            for (var i = 1; i <= 100; i++)
            {
                products.Add(new Product { Id = i, Name = "Product " + i });
            }
            return products;
        }
    }
}
```

در این مثال، هدف صرفا ارائه یک خروجی ساده JSON از طرف سرور است.

در ادامه نیاز است تعریف مسیریابی ذیل نیز به فایل Global.asax.cs برنامه اضافه شود تا بتوان به آدرس api/products در سایت، دسترسی یافت:

```
using System;
using System.Web.Http;
using System.Web.Routing;

namespace KendoUI02
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            RouteTable.Routes.MapHttpRoute(
               name: "DefaultApi",
               routeTemplate: "api/{controller}/{id}",
               defaults: new { id = RouteParameter.Optional }
               );
        }
    }
}
```

در ادامه فایلی را به نام Index.html (یا در یک View و یا یک فایل aspx دلخواه)، محتوای ذیل را اضافه کنید:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>Kendo UI: Implemeting the Grid</title>

    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.all.min.js" type="text/javascript"></script>
</head>
<body>

    <div id="report-grid"></div>
    <script type="text/javascript">
        $(function () {
            var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    }
                },
                error: function (e) {
                    alert(e.errorThrown.stack);
                },
                pageSize: 5,
                sort: { field: "Id", dir: "desc" }
            });

            $("#report-grid").kendoGrid({
                dataSource: productsDataSource,
                autoBind: true,
                scrollable: false,
                pageable: true,
                sortable: true,
                columns: [
                    { field: "Id", title: "#" },
                    { field: "Name", title: "Product" }
                ]
            });
        });
    </script>
</body>
</html>
```

- ابتدا فایل‌های اسکریپت و CSS مورد نیاز Kendo UI اضافه شده‌اند.
- گرید صفحه، در محل div ایی با id مساوی report-grid تشکیل خواهد شد.
- سپس DataSource ایی که به آدرس api/products اشاره می‌کند، تعریف شده و در آخر productsDataSource را توسط یک kendoGrid نمایش داده‌ایم.
- نحوه‌ی تعریف productsDataSource، در قسمت استفاده از منابع داده راه دور ابتدای بحث توضیح داده شد. در اینجا فقط دو خاصیت pageSize و sort نیز به آن اضافه شده‌اند. این دو خاصیت بر روی نحوه‌ی نمایش گرید نهایی تاثیر گذار هستند. pageSize تعداد رکورد هر صفحه را مشخص می‌کند و sort نحوه‌ی مرتب سازی را بر اساس فیلد Id و در حالت نزولی قرار می‌دهد.
- در ادامه، ابتدایی‌ترین حالت کار با kendoGrid را ملاحظه می‌کنید.
- تنظیم dataSource و autoBind: true (حالت پیش فرض)، سبب خواهند شد تا به صورت خودکار، اطلاعات JSON از مسیر api/products خوانده شوند.
- سه خاصیت بعدی صفحه بندی و مرتب سازی خودکار ستون‌ها را فعال می‌کنند.
- در آخر هم دو ستون گرید، بر اساس نام‌های خواص کلاس Product تعریف شده‌اند.

سورس کامل این قسمت را از اینجا می‌توانید دریافت کنید:

KendoUI02.zip

عنوان: **صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۶ ۱۶:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, DynamicLINQ, Kendo UI, json.net

پس از آشنایی مقدماتی با Kendo UI DataSource ، اکنون می‌خواهیم از آن جهت صفحه بندی، مرتب سازی و جستجوی پویای سمت سرور استفاده کنیم. در مثال قبلی، هر چند صفحه بندی فعال بود، اما پس از دریافت تمام اطلاعات، این اعمال در سمت کاربر انجام و مدیریت می‌شد.

مدل برنامه

در اینجا قصد داریم لیستی را با ساختار کلاس Product در اختیار Kendo UI گرید قرار دهیم:

```
namespace KendoUI03.Models
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public decimal Price { set; get; }
        public bool IsAvailable { set; get; }
    }
}
```

پیشنیاز تامین داده مخصوص Kendo UI Grid

برای ارائه اطلاعات مخصوص Kendo UI Grid، ابتدا باید درنظر داشت که این گرید، درخواست‌های صفحه بندی خود را با فرمت ذیل ارسال می‌کند. همانطور که مشاهده می‌کنید، صرفا یک کوئری استرینگ با فرمت JSON را دریافت خواهیم کرد:

```
/api/products?{"take":10,"skip":0,"page":1,"pageSize":10,"sort":[{"field":"Id","dir":"desc"}]}
```

سپس این گرید نیاز به سه فیلد، در خروجی JSON نهایی خواهد داشت:

```
{
"Data":
[
{"Id":1500,"Name":"1500 نام","Price":2499.0,"IsAvailable":false},
{"Id":1499,"Name":"1499 نام","Price":2498.0,"IsAvailable":true}
],
"Total":1500,
"Aggregates":null
}
```

فیلد Data که رکوردهای گرید را تامین می‌کنند. فیلد Total که بیانگر تعداد کل رکوردها است و Aggregates که برای گروه بندی بکار می‌رود.

می‌توان برای تمام این‌ها، کلاس و Parser تهیه کرد و یا ... پروژه‌ی سورس بازی به نام Kendo.DynamicLinq نیز چنین کاری را میسر می‌سازد که در ادامه از آن استفاده خواهیم کرد. برای نصب آن تنها کافی است دستور ذیل را صادر کنید:

```
PM> Install-Package Kendo.DynamicLinq
```

Kendo.DynamicLinq به صورت خودکار System.Linq.Dynamic را نیز نصب می‌کند که از آن جهت صفحه بندی پویا استفاده خواهد شد.

تامین کننده‌ی داده سمت سرور

همانند مطلب کار با Kendo UI DataSource ، یک ASP.NET Web API Controller جدید را به پروژه اضافه کنید و همچنین مسیریابی‌های مخصوص آن‌را به فایل global.asax.cs نیز اضافه نمائید.

```
using System.Linq;
using System.Net.Http;
using System.Web.Http;
using Kendo.DynamicLinq;
using KendoUI03.Models;
using Newtonsoft.Json;

namespace KendoUI03.Controllers
{
    public class ProductsController : ApiController
    {
        public DataSourceResult Get(HttpRequestMessage requestMessage)
        {
            var request = JsonConvert.DeserializeObject<DataSourceRequest>(
                requestMessage.RequestUri.ParseQueryString().GetKey(0)
            );

            var list = ProductDataSource.LatestProducts;
            return list.AsQueryable()
                       .ToDataSourceResult(request.Take, request.Skip, request.Sort, request.Filter);
        }
    }
}
```

تمام کدهای این کنترلر همین چند سطر فوق هستند. با توجه به ساختار کوئری استرینگی که در ابتدای بحث عنوان شد، نیاز است آن‌را توسط کتابخانه‌ی JSON.NET تبدیل به یک نمونه از DataSourceRequest نمائیم. این کلاس در Kendo.DynamicLinq تعریف شده‌است و حاوی اطلاعاتی مانند take و skip کوئری LINQ نهایی است.
ProductDataSource.LatestProducts صرفا یک لیست جنریک تهیه شده از کلاس Product است. در نهایت با استفاده از متد الحاقی جدید ToDataSourceResult ، به صورت خودکار مباحث صفحه بندی سمت سرور به همراه مرتب سازی اطلاعات، صورت گرفته و اطلاعات نهایی با فرمت DataSourceResult بازگشت داده می‌شود. DataSourceResult نیز در Kendo.DynamicLinq تعریف شده و سه فیلد یاد شده‌ی Data، Total و Aggregates را تولید می‌کند.

تا اینجا کارهای سمت سرور این مثال به پایان می‌رسد.

تهیه View نمایش اطلاعات ارسالی از سمت سرور

اعمال مباحث بومی سازی

```
<head>
    <meta charset="utf-8" />
    <meta http-equiv="Content-Language" content="fa" />
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <title>Kendo UI: Implemeting the Grid</title>

    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <!--شیوه نامه‌ی مخصوص راست به چپ سازی-->
    <link href="styles/kendo.rtl.min.css" rel="stylesheet" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.all.min.js" type="text/javascript"></script>

    <!--محل سفارشی سازی پیام‌ها و مسایل بومی-->
    <script src="js/cultures/kendo.culture.fa-IR.js" type="text/javascript"></script>
    <script src="js/cultures/kendo.culture.fa.js" type="text/javascript"></script>
    <script src="js/messages/kendo.messages.en-US.js" type="text/javascript"></script>

    <style type="text/css">
        body {
            font-family: tahoma;
            font-size: 9pt;
        }
    </style>

    <script type="text/javascript">
        // جهت استفاده از فایل: kendo.culture.fa-IR.js
        kendo.culture("fa-IR");
    </script>
</head>
```

- در اینجا چند فایل js و css جدید اضافه شده‌اند. فایل kendo.rtl.min.css جهت تامین مباحث RTL توکار Kendo UI کاربرد دارد.
- سپس سه فایل kendo.culture.fa.js، kendo.culture.fa-IR.js و kendo.messages.en-US.js نیز اضافه شده‌اند. فایل‌های fa و fa-Ir آن هر چند به ظاهر برای ایران طراحی شده‌اند، اما نام ماه‌های موجود در آن عربی است که نیاز به ویرایش دارد. به همین جهت به سورس این فایل‌ها، جهت ویرایش نهایی نیاز خواهد بود که در پوشه‌ی src\js\cultures مجموعه‌ی اصلی Kendo UI موجود هستند (ر.ک. فایل پیوست).
- فایل kendo.messages.en-US.js حاوی تمام پیام‌های مرتبط با Kendo UI است. برای مثال«رکوردهای 10 تا 15 از 1000 ردیف» را در اینجا می‌توانید به فارسی ترجمه کنید.
- متد kendo.culture کار مشخص سازی فرهنگ بومی برنامه را به عهده دارد. برای مثال در اینجا به fa-IR تنظیم شده‌است. این مورد سبب خواهد شد تا از فایل kendo.culture.fa-IR.js استفاده گردد. اگر مقدار آن‌را به fa تنظیم کنید، از فایل kendo.culture.fa.js کمک گرفته خواهد شد.

راست به چپ سازی گرید

تنها کاری که برای راست به چپ سازی Kendo UI Grid باید صورت گیرد، محصور سازی div آن در یک div با کلاس مساوی k-

rtl است:

```
<div class="k-rtl">
        <div id="report-grid"></div>
    </div>
```

k-rtl و تنظیمات آن در فایل kendo.rtl.min.css قرار دارند که در ابتدای head صفحه تعریف شده‌است.

تامین داده و نمایش گرید

در ادامه کدهای کامل DataSource و Kendo UI Grid را ملاحظه می‌کنید:

```
<script type="text/javascript">
        $(function () {
            var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    },
                    parameterMap: function (options) {
                        return kendo.stringify(options);
                    }
                },
                schema: {
                    data: "Data",
                    total: "Total",
                    model: {
                        fields: {
                            "Id": { type: "number" }, //تعیین نوع فیلد برای جستجوی پویا مهم است
                            "Name": { type: "string" },
                            "IsAvailable": { type: "boolean" },
                            "Price": { type: "number" }
                        }
                    }
                },
                error: function (e) {
                    alert(e.errorThrown);
                },
                pageSize: 10,
                sort: { field: "Id", dir: "desc" },
                serverPaging: true,
                serverFiltering: true,
                serverSorting: true
            });

            $("#report-grid").kendoGrid({
                dataSource: productsDataSource,
                autoBind: true,
                scrollable: false,
                pageable: true,
                sortable: true,
                filterable: true,
                reorderable: true,
                columnMenu: true,
                columns: [
                    { field: "Id", title: "شماره", width: "130px" },
                    { field: "Name", title: "نام محصول" },
                    {
                        field: "IsAvailable", title: "موجود است",
                        template: '<input type="checkbox" #= IsAvailable ? checked="checked" : "" #
disabled="disabled" ></input>'
                    },
                    { field: "Price", title: "قیمت", format: "{0:c}" }
                ]
            });
        });
    </script>
```

- با تعاریف مقدماتی Kendo UI DataSource پیشتر آشنا شده‌ایم و قسمت read آن جهت دریافت اطلاعات از سمت سرور کاربرد

دارد.

- در اینجا ذکر contentType الزامی است. زیرا ASP.NET Web API بر این اساس است که تصمیم می‌گیرد، خروجی را به صورت JSON ارائه دهد یا XML.

- با استفاده از parameterMap، سبب خواهیم شد تا پارامترهای ارسالی به سرور، با فرمت صحیحی تبدیل به JSON شده و بدون مشکل به سرور ارسال گردند.

- در قسمت schema باید نام فیلدهای موجود در DataSourceResult دقیقا مشخص شوند تا گرید بداند که data را باید از چه فیلدی استخراج کند و تعداد کل ردیف‌ها در کدام فیلد قرار گرفته‌است.

- نحوه‌ی تعریف model را نیز در اینجا ملاحظه می‌کنید. ذکر نوع فیلدها در اینجا بسیار مهم است و اگر قید نشوند، در حین جستجوی پویا به مشکل برخواهیم خورد. زیرا پیش فرض نوع تمام فیلدها string است و در این حالت نمی‌توان عدد 1 رشته‌ای را با یک فیلد از نوع int در سمت سرور مقایسه کرد.

- در اینجا serverFiltering، serverPaging و serverSorting نیز به true تنظیم شده‌اند. اگر این مقدار دهی‌ها صورت نگیرد، این اعمال در سمت کلاینت انجام خواهند شد.

پس از تعریف DataSource، تنها کافی است آن‌را به خاصیت dataSource یک kendoGrid نسبت دهیم.

- autoBind: true سبب می‌شود تا اطلاعات DataSource بدون نیاز به فراخوانی متد read آن به صورت خودکار دریافت شوند.

- با تنظیم scrollable: false، اعلام می‌کنیم که قرار است تمام رکوردها در معرض دید قرارگیرند و اسکرول پیدا نکنند.

- pageable: true صفحه بندی را فعال می‌کند. این مورد نیاز به تنظیم pageSize: 10 در قسمت DataSource نیز دارد.

- با sortable: true مرتب سازی ستون‌ها با کلیک بر روی سرستون‌ها فعال می‌گردد.

- filterable: true به معنای فعال شدن جستجوی خودکار بر روی فیلدها است. کتابخانه‌ی Kendo.DynamicLinq حاصل آن‌را در سمت سرور مدیریت می‌کند.

- reorderable: true سبب می‌شود تا کاربر بتواند محل قرارگیری ستون‌ها را تغییر دهد.

- ذکر columnMenu: true اختیاری است. اگر ذکر شود، امکان مخفی سازی انتخابی ستون‌ها نیز مسیر خواهد شد.

- در آخر ستون‌های گرید مشخص شده‌اند. با تعیین "format: "{0:c} سبب نمایش فیلدهای قیمت با سه رقم جدا کننده خواهیم شد. مقدار ریال آن از فایل فرهنگ جاری تنظیم شده دریافت می‌گردد. با استفاده از template تعریف شده نیز سبب نمایش فیلد bool به صورت یک checkbox خواهیم شد.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

KendoUI03.zip

نظرات خوانندگان

نویسنده: حمیدرضا کبیری
تاریخ: ۱۳۹۳/۰۸/۱۶ ۱۹:۳

آیا kendo UI کاملا از زبان فارسی پشتیبانی میکند ؟
برای calender آن ، به تقویم شمی گزینه ای موجود هست ؟
این گزینه با ورژن ۲۰۱۴/۱/۳۱۸ مطابقت دارد ، آیا با ورژنهای جدید مشکلی نخواهد داشت ؟

نویسنده: احمد رجبی
تاریخ: ۱۳۹۳/۰۸/۱۶ ۲۰:۱۵

میتوانید با اضافه کردن این اسکریپت تمامی قسمتهای kendo را به زبان فارسی ترجمه کنید.

نویسنده: سعیدجلالی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۸:۴۴

با تشکر از مطلب مفید شما من از wrapper mvc مجموعه kendo استفاده میکنم
توی مطالب شما در مورد استفاده از Kendo.DynamicLinq صحبت شد خواستم بدونم آیا وقتی از wrapper هم استفاده میکنیم استفاده از این پکیج لازم هست؟

```
@(Html.Kendo().Grid(Model)
      .Name("gridKendo")
      .Columns(columns =>
      {
          columns.Bound(c => c.Name).Width(50);
          columns.Bound(c => c.Family).Width(100);
          columns.Bound(c => c.Tel);
      })
      .Pageable(pager => pager
          .Input(true)
          .Numeric(true)
          .Info(true)
          .PreviousNext(true)
          .Refresh(true)
          .PageSizes(true)
      )
      )
```

چون من با استفاده از telerik profiler وقتی درخواست رو بررسی مکنم توی دستور sql چنین دستوری رو در انتها مشاهده میکنم:
صفحه اول:

```
SELECT *
FROM (
    SELECT
    FROM table a
)
WHERE ROWNUM <= :TAKE
```

صفحات بعد:

```
SELECT *
FROM (
    SELECT
        a.*,
        ROWNUM OA_ROWNUM
    FROM (
        FROM table a
```

```
    ) a
    WHERE ROWNUM <= :TAKE
)
WHERE OA_ROWNUM > :SKIP
```

پایگاه داده اوراکل است.

نویسنده: سعیدجلالی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۹:۱۲

امکان فارسی شدن تمام بخش‌ها وجود دارد.
تقویم هم فارسی شده است در این سایت برای نسخه‌های جدیدتر هم باید دوتا فایل جاوا اسکریپت all و mvc رو خودتون تغییر بدهید (با توجه به الگوی انجام شده در فایل فارسی شده فوق)
ولی برای تقویم زمانبندی scheduler من فارسی ندیده ام

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۷ ۹:۳۱

مطلب فوق نه وابستگی خاصی به وب فرم‌ها دارد و نه ASP.NET MVC. ویو آن یک فایل HTML ساده‌است و سمت سرور آن فقط یک کنترلر ASP.NET Web API که با تمام مشتقات ASP.NET سازگار است. در این حالت یک نفر می‌تواند ASP.NET نگارش خودش را خلق کند؛ بدون اینکه نگران جزئیات وب فرم‌ها باشد یا ASP.NET MVC. ضمنا دانش جاوا اسکریپتی آن هم قابل انتقال است؛ چون اساسا Kendo UI برای فناوری سمت سرور خاصی طراحی نشده‌است و حالت اصل آن با PHP، Java و امثال آن هم کار می‌کند.

نویسنده: میثم آقااحمدی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۳:۱۷

در کنترلر این خط باعث بارگذاری تمامی داده‌ها می‌شود

```
var list = ProductDataSource.LatestProducts;
```

آیا راه حلی وجود دارد که دیتای به تعداد همان pagesize از پایگاه خوانده شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۳:۲۸

- این فقط یک مثال هست و منبع داده‌ای صرفا جهت دموی ساده‌ی برنامه. فقط برای اینکه با یک کلیک بتوانید برنامه را اجرا کنید و نیازی به برپایی و تنظیم بانک اطلاعاتی و امثال آن نداشته باشد.
- شما در کدها و کوئری‌های مثلا EF در اصل با یک سری IQueryable کار می‌کنید. همینجا باید متد الحاقی ToDataSourceResult را اعمال کنید تا نتیجه‌ی نهایی در حداقل بار تعداد رفت و برگشت و با کوئری مناسبی بر اساس پارامترهای دریافتی به صورت خودکار تولید شود. در انتهای کار بجای مثلا ToList بنویسید ToDataSourceResult.

نویسنده: امین
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۴:۴۱

سلام من در ویو خودم نمیتونم اطلاعاتم رو تو kendo.grid ببینم و برای من یک لیست استرینگ در ویو نمایش داده میشه و به این شکل در کنترلر و ویو کد نویسی کردم .

```
public class EFController : Controller    {
        //
        // GET: /EF/

        public ActionResult AjaxConnected([DataSourceRequest] DataSourceRequest request )
```

```
        {
            using (var dbef=new dbTestEntities())
            {
                IQueryable<Person> persons = dbef.People;
                DataSourceResult result = persons.ToDataSourceResult(request);
                return Json(result.Data,JsonRequestBehavior.AllowGet);
            }
        }


    }
```

و ویو

```
@{    ViewBag.Title = "AjaxConnected";
}


<h2>AjaxConnected</h2>
@(Html.Kendo().Grid<TelerikMvcApp2.Models.Person>(  )
      .Name("Grid")
      .DataSource(builder => builder
          .Ajax()
          .Read(operationBuilder => operationBuilder.Action("AjaxConnected", "EF"))


      )
      .Columns(factory =>
      {
          factory.Bound(person => person.personId);
          factory.Bound(person => person.Name);
          factory.Bound(person => person.LastName);
      })
      .Pageable()
      .Sortable())
```

و یک لیست استرینگ بهم در عمل خروجی میده و از خود قالب kendogrid خبری نیست . من اطلاعات رو به طور json پاس میدم و ajaxi میگیرم.

حالا قبلش همچین خطلایی داشتم که به allowget ایراد میگرفت ولی در کل با JsonRequestBehavior.AllowGet حل شد و حالا فقط یه لسیت بهم خروجی میده! و از ظاهر گرید خبری نیست. و اگر به جای json نوشته بشه view و با ویو return کنم ظاهر kendogrid رو دارم اما خروجی دارای مقداری نیست! اینم خروجی استرینگ من :(

```
[{"personId":1,"Name":"Amin","LastName":"Saadati"},  {"personId":2,"Name":"Fariba","LastName":"Ghochani
"},{"personId":4,"Name":"Milad","LastName":"Rahman  i"},{"personId":5,"Name":"rima","LastName":"rad"},
{"personId":6,"Name":"ali","LastName":"kiva"},{"pe  rsonId":7,"Name":"sahel","LastName":"abasi"},{"per
sonId":8,"Name":"medi","LastName":"ghaem"},{"perso  nId":9,"Name":"mino","LastName":"kafash"},{"person
Id":10,"Name":"behzad","LastName":"tizro"},{"perso  nId":11,"Name":"toti","LastName":"saadati"},{"pers
onId":12,"Name":"parinaz","LastName":"karami"},{"p  ersonId":13,"Name":"sadegh","LastName":"hojati"},{
"personId":14,"Name":"milad","LastName":"ebadipor"  },{"personId":15,"Name":"farid","LastName":"riazi"
},{"personId":16,"Name":"said","LastName":"abdoli"  },{"personId":17,"Name":"behzad","LastName":"ariaf
ar"},{"personId":18,"Name":"jamshid","LastName":"k  otahi"}]
```

این سوال رو در چند سایت پرسیدم و به جوابی برایش نرسیدم. و نمیدونم ایراد کدهای نوشته شده ام کجاست!
متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۵:۲

- قصد پشتیبانی از wrapperهای آن‌را ندارم. لطفا خارج از موضوع سؤال نپرسید. اگر کسی دوست داشت در این زمینه مطلب منتشر کند، خوب. ولی من چنین قصدی ندارم.
- عرض کردم اگر از wrapperها استفاده کنید، به علت عدم درک زیر ساخت اصلی Kendo UI، قادر به دیباگ کار نخواهید بود.
- اگر متن را مطالعه کنید در قسمت «پیشنیاز تامین داده مخصوص Kendo UI Grid » دقیقا شکل نهایی خروجی JSON مورد نیاز ارائه شده‌است. این خروجی در سه فیلد data، total و aggregate قرار می‌گیرد. شما الان فقط قسمت data آن‌را بازگشت

داده‌اید؛ بجای اصل و کل آن. نام این سه فیلد هم مهم نیست؛ اما هر چیزی که تعیین می‌شوند، باید در قسمت data source در خاصیت schema آن مانند مثالی که در مطلب جاری آمده (در قسمت «تامین داده و نمایش گرید »)، دقیقا مشخص شوند، تا Kendo UI بداند که اطلاعات مختلف را باید از چه فیلدهایی از JSON خروجی دریافت کند.

عنوان: **آشنایی با Saltarelle کامپایلر قدرتمند C# به جاوااسکریپت**
نویسنده: رضا بازرگان
تاریخ: ۱۳۹۳/۰۸/۱۶ ۱۹:۵۵
آدرس: www.dotnettips.info
گروه‌ها: C#, JavaScript, Javascript extensions, Compiler, Saltarelle

شاید ساده‌ترین تعریف برای Saltarelle این باشد که «کامپایلریست که کدهای C# را به جاوا اسکریپت تبدیل می‌کند». محاسن زیادی را می‌توان برای اینگونه کامپایلرها نام برد؛ مخصوصا در پروژه‌های سازمانی که نگهداری از کدهای جاوا اسکریپت بسیار سخت و گاهی خارج از توان است و این شاید مهمترین عامل ظهور ابزارهای جدید از قبیل Typescript باشد.

در هر صورت اگر حوصله و وقت کافی برای تجهیز تیم نرم افزاری، به دانش یک زبان جدید مانند Typescript نباشد، استفاده از توان و دانش تیم تولید، از زبان C# ساده‌ترین راه حل است و اگر ابزاری مطمئن برای استفاده از حداکثر قدرت JavaScript همراه با امکانات نگهداری و توسعه کدها وجود داشته باشد، بی شک Saltarelle یکی از بهترین‌های آنهاست.

قبلا کامپایلر هایی از این دست مانند #Script وجود داشتند، اما فاقد همه امکانات C# بوده وعملا قدرت کامل C# در کد نویسی وجود نداشت. اما با توجه به ادعای توسعه دهندگان این کامپایلر سورس باز در استفاده‌ی حداکثری از کلیه ویژگی‌های C# 5 و با وجود Library های متعدد می‌توان Saltarelle را عملا یک کامپایلر موفق در این زمینه دانست.

برای استفاده از Saltarelle در یک برنامه وب ساده باید یک پروژه Console Application به Solution اضافه کرد و پکیج Saltarelle.Compiler را از nuget نصب نمایید. بعد از نصب این پکیج، کلیه Reference ها از پروژه حذف می‌شوند و هر بار Build توسط کامپایلر Saltarelle انجام می‌شود. البته با اولین Build ، مقداری Error را خواهید دید که برای از بین بردنشان نیاز است پکیج Saltarelle.Runtime را نیز در این پروژه نصب نمایید:

```
PM> Install-Package Saltarelle.Compiler
PM> Install-Package Saltarelle.Runtime
```

در صورتیکه کماکان Build نهایی با Error همرا بود، یکبار این پروژه را Unload و سپس مجددا Load نمایید

UI یک پروژه وب MVC است و Client یک Console Application که پکیج‌های مورد نیاز Saltarelle روی آن نصب شده است.

در صورتیکه پروژه را Build نماییم و نگاهی به پوشه‌ی Debug بیاندازیم، یک فایل JavaScript همنام پروژه وجود دارد:

Name	Date modified	Type	Size
mscorlib.dll	11/7/2014 13:16	Application extens...	
mscorlib	11/7/2014 13:16	XML Document	
SaltarelleSample.Client	11/7/2014 13:32	Application	
SaltarelleSample.Client.exe	11/7/2014 13:15	XML Configuratio...	
SaltarelleSample.Client	11/7/2014 13:17	JavaScript File	
SaltarelleSample.Client	11/7/2014 13:32	Program Debug D...	
SaltarelleSample.Client.vshost	11/7/2014 13:16	Application	
SaltarelleSample.Client.vshost.exe	11/7/2014 13:15	XML Configuratio...	
SaltarelleSample.Client.vshost.exe.manifest	6/18/2013 16:58	MANIFEST File	

برای اینکه بعد از هر بار Build ، فایل اسکریپت به پوشه‌ی مربوطه در پروژه UI منتقل شود کافیست کد زیر را در Post Build پروژه Client بنویسیم:

```
copy "$(TargetDir)$(TargetName).js" "$(SolutionDir)SalratelleSample.UI\Scripts"
```

اکنون پس از هر بار Build ، فایل اسکریپت مورد نظر در پوشه‌ی Scripts پروژه UI آپدیت می‌شود:

در ادامه کافیست فایل اسکریپت را به layout اضافه کنیم.

```
<script src="~/Scripts/SaltarelleSample.Client.js"></script>
```

در پوشه‌ی Saltarelle.Runtime در پکیج‌های نصب شده، یک فایل اسکریپت به نام mscorlib.min.js نیز وجود دارد که حاوی اسکریپت‌های مورد نیاز Saltarelle در هنگام اجراست. آن را به پوشه اسکریپت‌های پروژه UI کپی نمایید و سپس به Layout اضافه کنید.

```
<script src="~/Scripts/mscorlib.min.js"></script>
<script src="~/Scripts/SaltarelleSample.Client.js"></script>
```

حال نوبت به اضافه نمودن libraryهای مورد نیازمان است. برای دسترسی به آبجکت هایی از قبیل document, window, element و غیره در جاوااسکریپت می‌توان پکیج Saltarelle.Web را در پروژه‌ی Client نصب نمود و برای دسترسی به اشیاء و فرمانهای jQuery، پکیج Salratelle.jQuery را نصب نمایید.

```
> Install-Package Saltarelle.Web
> Install-Package Saltarelle.jQuery
```

به این libraryها imported library می‌گویند. در واقع، در زمان کامپایل، برای این libraryها فایل اسکریپتی تولید نمی‌شود و فقط آبجکت‌های #C هستند که که هنگام کامپایل تبدیل به کدهای ساده اسکریپت می‌شوند که اگر اسکریپت مربوط به آنها به صفحه اضافه نشده باشد، اجرای اسکریپت با خطا مواجه می‌شود.

به طور ساده‌تر وقتی از jQuery library استفاده می‌کنید هیچ فایل اسکریپت اضافه‌ای تولید نمی‌شود، اما باید اسکریپت jQuery به صفحه شما اضافه شده باشد.

```
<script src="~/Scripts/jquery-1.10.2.min.js"></script>
```

مثال ما یک اپلیکیشن ساده برای خواندن فیدهای همین سایت است. ابتدا کدهای سمت سرور را در پروژه UI می نویسیم.

کلاس‌های مورد نیاز ما برای این فید ریدر:

```
public class Feed
    {
        public string FeedId { get; set; }
        public string Title { get; set; }
        public string Address { get; set; }

    }
    public class Item
    {
        public string Title { get; set; }
        public string Link { get; set; }
        public string Description { get; set; }
    }
```

و یک کلاس برای مدیریت منطق برنامه

```
public class SiteManager
    {
        private static List<Feed> _feeds;
        public static List<Feed> Feeds
        {
            get
```

```
            {
                if (_feeds == null)
                    _feeds = CreateSites();
                return _feeds;
            }
        }
        private static List<Feed> CreateSites()
        {
            return new List<Feed>() {
                new Feed(){
                    FeedId = "1",
                    Title = "آخرین تغییرات سایت",
                    Address = "http://www.dotnettips.info/rss.xml"
                },
                 new Feed(){
                    FeedId = "2",
                    Title = "مطالب سایت",
                    Address = "http://www.dotnettips.info/feeds/posts"
                },
                 new Feed(){
                    FeedId = "3",
                    Title = "نظرات سایت",
                    Address = "http://www.dotnettips.info/feeds/comments"
                },
                 new Feed(){
                    FeedId = "4",
                    Title = "خلاصه اشتراک ها",
                    Address = "http://www.dotnettips.info/feed/news"
                },
            };
        }

        public static IEnumerable<Item> GetNews(string id)
        {
            XDocument feedXML = XDocument.Load(Feeds.Find(s=> s.FeedId == id).Address);
            var feeds = from feed in feedXML.Descendants("item")
                        select new Item
                        {
                            Title = feed.Element("title").Value,
                            Link = feed.Element("link").Value,
                            Description = feed.Element("description").Value
                        };
            return feeds;
        }

    }
```

کلاس SiteManager فقط یک لیست از فیدها دارد و متدی که با گرفتن شناسه‌ی فید ، یک لیست از آیتم‌های موجود در آن فید ایجاد می‌کند.

حال دو ApiController برای دریافت داده‌ها ایجاد می‌کنیم

```
public class FeedController : ApiController
{
        // GET api/<controller>
        public IEnumerable<Feed> Get()
        {
            return SiteManager.Feeds;
        }
    }

public class ItemsController : ApiController
    {
        // GET api/<controller>/5
        public IEnumerable<Item> Get(string id)
        {
            return SiteManager.GetNews(id);
        }
    }
```

در View پیش‌فرض که Index از کنترلر Home است، یک Html ساده برای فرم صفحه اضافه می‌کنیم

```
<div>
    <div>
        <h2>Feeds</h2>
        <ul id="Feeds">

        </ul>
    </div>
    <div>
        <h2>Items</h2>
        <p id="FeedItems">
        </p>
    </div>

</div>
```

در المنت Feeds لیست فیدها را قرار می‌دهیم و در FeedItems آیتم‌های مربوط به هر فید. حال به سراغ کدهای سمت کلاینت می‌رویم و به جای جاوا اسکریپت از Saltarelle استفاده می‌کنیم.

کلاس Program را از پروژه Client باز می‌کنیم و متد Main را به شکل زیر تغییر می‌دهیم:

```
static void Main()
        {
            jQuery.OnDocumentReady(() => {
                FillFeeds();
            });
        }
```

بعد از کامپایل شدن، کد C# شارپ بالا به صورت زیر در می‌آید:

```
$SaltarelleSample_Client_$Program.$main = function() {
$(function() {
$SaltarelleSample_Client_$Program.$fillFeeds();
});
};
$SaltarelleSample_Client_$Program.$main();
```

و این همان متد معروف jQuery است که Saltarelle.jQuery برایمان ایجاد کرده است.

متد FillFeeds را به شکل زیر پیاده سازی می‌کنیم

```
private static void FillFeeds()
        {
            jQuery.Ajax(new jQueryAjaxOptions()
            {
                Url = "/api/feed",
                Type = "GET",
                Success = (d,t,r) => {

                    // Fill
                    var ul = jQuery.Select("#Feeds");
                    jQuery.Each((List<Feed>)d, (idx,i) => {
                        var li = jQuery.Select("<li>").Text(i.Title).CSS("cursor", "pointer");
                        li.Click(eve => {
                            FillData(i.FeedId);
                        });
                        ul.Append(li);
                    });
                }
            });
        }
```

آبجکت jQuery، متدی به نام Ajax دارد که یک شی از کلاس jQueryAjaxOptions را به عنوان پارامتر می‌پذیرد. این کلاس کلیه خصوصیات متد Ajax در jQuery را پیاده سازی می‌کند. نکته شیرین آن توانایی نوشتن lambda برای Delegate هاست.

خاصیت Success یک Delegate است که 3 پارامتر ورودی را می‌پذیرد.

```
public delegate void AjaxRequestCallback(object data, string textStatus, jQueryXmlHttpRequest request);
```

data همان مقداریست که api باز می‌گرداند که یک لیست از Feed هاست. برای زیبایی کار، من یک کلاس Feed در پروژه Client اضافه می‌کنم که خصوصیاتی مشترک با کلاس اصلی سمت سرور دارد و مقدار برگشتی Ajax را به آن تبدیل می‌کنم.

کلاس Feed و Item

```
[PreserveMemberCase()]
    public class Feed
    {
        //[ScriptName("FeedId")]
        public string FeedId;

        //[ScriptName("Title")]
        public string Title;

        //[ScriptName("Address")]
        public string Address;

    }

    [PreserveMemberCase()]
    public class Item
    {
        // [ScriptName("Title")]
        public string Title;

        // [ScriptName("Link")]
        public string Link;

        // [ScriptName("Description")]
        public string Description;
    }
```

Attributeهای زیادی در Saltarelle وجود دارند و از آنجایی که کامپایلر اسم فیلدها را camelCase کامپایل می‌کند من برای جلوگیری از آن از PreserveMemberCase بر روی هر کلاس استفاده کردم. می‌توانید اسم هر فیلد را سفارشی کامپایل نمایید.

```
jQuery.Each((List<Feed>)d, (idx,i) => {
                        var li = jQuery.Select("<li>").Text(i.Title).CSS("cursor", "pointer");
                        li.Click(eve => {
                            FillData(i.FeedId);
                        });
                        ul.Append(li);
                    });
```

به ازای هر آیتمی که در شیء بازگشتی وجود دارد، با استفاد از متد each در jQuery یک li ایجاد می‌کنیم. همان طور که می‌بینید کلیه خواص، به شکل Fluent قابل اضافه شدن می‌باشد. سپس برای li یک رویداد کلیک که در صورت وقوع، متد FillData را با شناسه فید کلیک شده فراخوانی می‌کند و در آخر li را به المنت ul اضافه می‌کنیم.

برای هر کلیک هم مانند مثال بالا api را با شناسه‌ی فید مربوطه فراخوانی کرده و به ازای هر آیتم، یک سطر ایجاد می‌کنیم.

```
private static void FillData(string p)
        {
            jQuery.Ajax(new jQueryAjaxOptions()
            {
                Url = "/api/items/" + p,
                Type = "GET",
                Success = (d, t, r) => {
                    var content = jQuery.Select("#FeedItems");
                    content.Html("");
                    foreach (var item in (List<Item>)d)
                    {
```

```
                    var row = jQuery.Select("<div>").AddClass("row").CSS("direction", "rtl");
                    var link = jQuery.Select("<a>").Attribute("href", item.Link).Text(item.Title);
                    row.Append(link);
                    content.Append(row);
                }
            }
        });
    }
```

خروجی برنامه به شکل زیر است:

در این مثال ما از Saltarelle.jQuery برای استفاده از jQuery.js استفاده نمودیم. libraryهای متعددی برای Saltarelle از قبیل linq,angular,knockout,jQueryUI,nodeJs ایجاد شده و همچنین قابلیت‌های زیادی برای نوشتن imported libraryهای سفارشی نیز وجود دارد.

مطمئنا استفاده از چنین کامپایلرهایی راه حلی سریع برای رهایی از مشکلات متعدد کد نویسی با جاوا اسکریپت در نرم افزارهای بزرگ مقیاس است. اما مقایسه آنها با ابزارهایی از قبیل typescript احتیاج به زمان و تجربه کافی در این زمینه دارد.

فایل پروژه ضمیمه

نظرات خوانندگان

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۳/۰۸/۱۶ ۲۳:۴۶

با تکمیل شدن Roslyn و آسانتر شدن امور، روز به روز شاهد تعداد بیشتری از چنین مبدل هایی خواهیم بود، اما به صورت بنیادی هرگونه مبدل کد CSharp به JavaScript یا هر زبان دیگری محکوم به شکست است، و آنچه که به صورت بنیادی مشکل ندارد، تبدیل IL به سایر زبانها است، چرا که فرض کنید شما یک DLL تقریبا ساده مانند Humanizer را که برای کار با رشته‌ها و ... به کار می‌رود را در مثالتان استفاده کنید، در این صورت دیگر برنامه شما کار نخواهد کرد، حتی اگر در حد یک Pluralize کردن باشد، اما اگر تبدیل IL به JavaScript باشد، هر رفتاری را که با DLL شما داشته باشد، همان رفتار را با Humanizer خواهد داشت، برای همین است که امروزه تبدیلگرهای قدرتمند از IL استفاده می‌کنند، مانند JSIL، و SharpKit که اوایل از تبدیل CSharp به JavaScript استفاده می‌کرد و هم اکنون تازه به این نتیجه رسیده که آب در هاون می‌کوبیده و الآن با استفاده از Cecil، به تبدیل IL به JavaScript روی آورده است.

همچنین تبدیل گر مربوطه، باید یکسری کتابخانه جاوا اسکریپتی پایه که امکانات پایه و اولیه .NET را ارائه دهد داشته باشد، که باز دقیقا تبدیلگرهای حرفه ای همین رفتار را دارند، چون همه‌ی امکانات .NET در JavaScript موجود نیست که صرف تبدیل کد کافی باشد و لااقل بعضی امکانات پایه باید ارائه شوند، مثلا برای ایجاد کردن اعداد تصادفی معادل کلاس Random در .NET

نویسنده: رضا بازرگان
تاریخ: ۱۳۹۳/۰۸/۱۸ ۱۹:۳۳

با تشکر از نوشتارتون ذکر دو نکته را لازم می‌دونم.

اول اینکه هدف از این مطلب الزام به استفاده یا عدم استفاده از این نوع کامپایلرها نیست و فقط برای آشنایی با این گونه ابزارها بوده که در حال حاضر در بسیاری از نرم افزارهای اینترپراز در حال استفاده اند.

و دوم اینکه تفاوت ساختاری و ماهیتی سی شارپ و جاوا اسکریپت اونقدر واضح هست که نباید از این دو انتظار یکسان داشت. و مهمترین عامل به وجود اومدن چنین کامپایلرهایی استفاده از سینتکس سی شارپ بوده و نه قدرت دات نت فریم ورک. بنا براین فکر می‌کنم لزومی به وجود مبدل هایی از زبان میانی وجود ندارد و همچنین واضح است کلاس هایی از قبیل Random و غیره که نه توانایی زبان سی شارپ بلکه امکانات درونی دات نت فریم ورک است برای همچین ابزاری بی معناست.

و فکر می‌کنم برای چنین کامپایلری لازم نیست جاوا اسکریپت همه امکانات سی شارپ را داشته باشد. و اینکه سی شارپ بتواند قسمت زیادی از امکانات جاوا اسکریپت را در اختیار برنامه ساز قرار دهد کافیست.

باز هم تشکر می‌کنم

عنوان: استفاده از Kendo UI templates
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۸ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

در مطلب « صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid » در انتهای بحث، ستون IsAvailable به صورت زیر تعریف شد:

```
columns: [
               {
                   field: "IsAvailable", title: "موجود است",
                   template: '<input type="checkbox" #= IsAvailable ? checked="checked" : "" #
disabled="disabled" ></input>'
                }
]
```

Templates، جزو یکی از پایه‌های Kendo UI Framework هستند و توسط آن‌ها می‌توان قطعات با استفاده‌ی مجدد HTML ایی را طراحی کرد که قابلیت یکی شدن با اطلاعات جاوا اسکریپتی را دارند.
همانطور که در این مثال نیز مشاهده می‌کنید، قالب‌های Kendo UI از syntax (#) Hash استفاده می‌کنند. در اینجا قسمت‌هایی از قالب که با علامت # محصور می‌شوند، در حین اجرا، با اطلاعات فراهم شده جایگزین خواهند شد.
برای رندر مقادیر ساده می‌توان از # =# استفاده کرد. از # :# برای رندر اطلاعات HTML-encoded کمک گرفته می‌شود و # # برای رندر کدهای جاوا اسکریپتی کاربرد دارد. از حالت HTML-encoded برای نمایش امن اطلاعات دریافتی از کاربران و جلوگیری از حملات XSS استفاده می‌شود.
اگر در این بین نیاز است # به صورت معمولی رندر شود، در حالت کدهای جاوا اسکریپتی به صورت #\\ و در HTML ساده به صورت #\ باید مشخص گردد.

مثالی از نحوه‌ی تعریف یک قالب Kendo UI

```
<!--دریافت اطلاعات از منبع محلی-->
    <script id="javascriptTemplate" type="text/x-kendo-template">
        <ul>
            # for (var i = 0; i < data.length; i++) { #
            <li>#= data[i] #</li>
            # } #
        </ul>
    </script>

    <div id="container1"></div>
    <script type="text/javascript">
        $(function () {
            var data = ['User 1', 'User 2', 'User 3'];
            var template = kendo.template($("#javascriptTemplate").html());
            var result = template(data); //Execute the template
            $("#container1").html(result); //Append the result
        });
    </script>
```

این قالب ابتدا در تگ script محصور می‌شود و سپس نوع آن مساوی text/x-kendo-template قرار می‌گیرد. در ادامه توسط یک حلقه‌ی جاوا اسکریپتی، عناصر آرایه‌ی فرضی data خوانده شده و با کمک Hash syntax در محل‌های مشخص شده قرار می‌گیرند.
در ادامه باید این قالب را رندر کرد. برای این منظور یک div با id مساوی container1 را جهت تعیین محل رندر نهایی اطلاعات مشخص می‌کنیم. سپس متد kendo.template بر اساس id قالب اسکریپتی تعریف شده، یک شیء قالب را تهیه کرده و سپس با ارسال آرایه‌ای به آن، سبب اجرای آن می‌شود. خروجی نهایی، یک قطعه کد HTML است که در محل container1 درج خواهد شد.
همانطور که ملاحظه می‌کنید، متد kendo.template، نهایتا یک رشته را دریافت می‌کند. بنابراین همینجا و به صورت inline نیز می‌توان یک قالب را تعریف کرد.

کار با منابع داده راه دور

فرض کنید مدل برنامه به صورت ذیل تعریف شده‌است:

```
namespace KendoUI04.Models
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public decimal Price { set; get; }
        public bool IsAvailable { set; get; }
    }
}
```

و لیستی از آن توسط یک ASP.NET Web API کنترلر، به سمت کاربر ارسال می‌شود:

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;
using KendoUI04.Models;

namespace KendoUI04.Controllers
{
    public class ProductsController : ApiController
    {
        public IEnumerable<Product> Get()
        {
            return ProductDataSource.LatestProducts.Take(10);
        }
    }
}
```

در سمت کاربر و در View برنامه خواهیم داشت:

```
<!--دریافت اطلاعات از سرور-->
    <div>
        <div id="container2"><ul></ul></div>
    </div>

    <script id="template1" type="text/x-kendo-template">
        <li> #=Id# - #:Name# - #=kendo.toString(Price, "c")#</li>
    </script>

    <script type="text/javascript">
        $(function () {
            var producatsTemplate1 = kendo.template($("#template1").html());

            var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    }
                },
                error: function (e) {
                    alert(e.errorThrown);
                },
                change: function () {
                    $("#container2 > ul").html(kendo.render(producatsTemplate1, this.view()));
                }
            });
            productsDataSource.read();
        });
    </script>
```

ابتدا یک div با id مساوی container2 جهت تعیین محل نهایی رندر قالب template1 در صفحه تعریف می‌شود.
هرچند خروجی دریافتی از سرور نهایتا یک آرایه از اشیاء Product است، اما در template1 اثری از حلقه‌ی جاوا اسکریپتی مشاهده نمی‌شود. در اینجا چون از متد kendo.render استفاده می‌شود، نیازی به ذکر حلقه نیست و به صورت خودکار، به تعداد

عناصر آرایه دریافتی از سرور، قطعه HTML قالب را تکرار می‌کند.
در ادامه برای کار با سرور از یک Kendo UI DataSource استفاده شده‌است. قسمت transport/read آن، کار تعریف محل دریافت اطلاعات را از سرور مشخص می‌کند. رویدادگران change آن اطلاعات نهایی دریافتی را توسط متد view در اختیار متد kendo.render قرار می‌دهد. در نهایت، قطعه‌ی HTML رندر شده‌ی نهایی حاصل از اجرای قالب، در بین تگ‌های ul مربوط به container2 درج خواهد شد.
رویدادگران change زمانیکه data source، از اطلاعات راه دور و یا یک آرایه‌ی جاوا اسکریپتی پر می‌شود، فراخوانی خواهد شد. همچنین مباحث مرتب سازی اطلاعات، صفحه بندی و تغییر صفحه، افزودن، ویرایش و یا حذف اطلاعات نیز سبب فراخوانی آن می‌گردند. متد view ایی که در این مثال فراخوانی شد، صرفا در روال رویدادگردان change دارای اعتبار است و آخرین تغییرات اطلاعات و آیتم‌های موجود در data source را باز می‌گرداند.

یک نکته‌ی تکمیلی: فعال سازی intellisense کدهای جاوا اسکریپتی Kendo UI

اگر به پوشه‌ی اصلی مجموعه‌ی Kendo UI مراجعه کنید، یکی از آن‌ها vsdoc نام دارد که داخل آن فایل‌های min.intellisense.js و vsdoc.js مشهود هستند.
اگر از ویژوال استودیوهای قبل از 2012 استفاده می‌کنید، نیاز است فایل‌های vsdoc.js متناظری را به پروژه اضافه نمائید؛ دقیقا در کنار فایل‌های اصلی js موجود. اگر از ویژوال استودیوی 2012 و یا بالاتر استفاده می‌کنید باید از فایل‌های intellisense.js متناظر استفاده کنید. برای مثال اگر از kendo.all.min.js کمک می‌گیرید، فایل متناظر با آن kendo.all.min.intellisense.js خواهد بود.
بعد از اینکار نیاز است فایلی به نام _references.js را به پوشه‌ی اسکریپت‌های خود با این محتوا اضافه کنید (برای VS 2012 به بعد):

```
/// <reference path="jquery.min.js" />
/// <reference path="kendo.all.min.js" />
```

نکته‌ی مهم اینجا است که این فایل به صورت پیش فرض از مسیر /Scripts/_references.js ~ خوانده می‌شود. برای اضافه کردن مسیر دیگری مانند /js/_references.js ~ باید آن‌را به تنظیمات ذیل اضافه کنید:

```
Tools menu -> Options -> Text Editor -> JavaScript -> Intellisense -> References
```

گزینه‌ی Reference Group را به Implicit (Web) تغییر داده و سپس مسیر جدیدی را اضافه نمائید.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:
KendoUI04.zip

عنوان: فرمت کردن اطلاعات نمایش داده شده به کمک Kendo UI Grid
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۹ ۱۳:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

پیشنیازهای بحث:

- « صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid »
- « استفاده از Kendo UI templates »

صورت مساله

می‌خواهیم به یک چنین تصویری برسیم؛ که دارای گروه بندی اطلاعات است، فرمت شرطی روی ستون قیمت آن اعمال شده و تاریخ نمایش داده شده در آن نیز شمسی است. همچنین برای مثال ستون قیمت آن دارای ته جمع صفحه بوده و به علاوه یک دکمه‌ی سفارشی به نوار ابزار آن اضافه شده‌است.

مباحث قسمت سمت سرور این مثال با مطلب « صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid » دقیقا یکی است. فقط یک خاصیت AddDate نیز در اینجا اضافه شده‌است.

تغییر نحوه‌ی نمایش pager

اگر به قسمت pager تصویر فوق دقت کنید، یک دکمه‌ی refresh، تعداد موارد هر صفحه و امکان وارد کردن دستی شماره صفحه، در آن پیش بینی شده‌است. این موارد را با تنظیمات ذیل می‌توان فعال کرد:

```
$("#report-grid").kendoGrid({
                // ...
                pageable: {
                    previousNext: true, // default true
                    numeric: true, // default true
                    buttonCount: 5, // default 10
                    refresh: true, // default false
                    input: true, // default false
                    pageSizes: true // default false
                },
```

بومی سازی پیغام‌های گرید

پیغام‌های فارسی را که در تصویر فوق مشاهده می‌کنید، حاصل پیوست فایل kendo.fa-IR.js هستند:

```
<!--https://github.com/loudenvier/kendo-global/blob/master/lang/kendo.fa-IR.js-->
<script src="js/messages/kendo.fa-IR.js" type="text/javascript"></script>
```

گروه بندی اطلاعات

برای گروه بندی اطلاعات در Kendo UI Grid دو قسمت باید تغییر کنند.
ابتدا باید فیلد پیش فرض گروه بندی در قسمت data source گرید تعریف شود:

```
var productsDataSource = new kendo.data.DataSource({
                // ...
                group: { field: "IsAvailable" },
                // ...
            });
```

همین تنظیم، گروه بندی را فعال خواهد کرد. اگر علاقمند باشید که به کاربران امکان تغییر دستی گروه بندی را بدهید، خاصیت groupable را نیز true کنید.

```
$("#report-grid").kendoGrid({
// ...
groupable: true, // allows the user to alter what field the grid is grouped by
// ...
```

در این حالت با کشیدن و رها کردن یک سرستون، به نوار ابزار مرتبط با گروه بندی، گروه بندی گرید بر اساس این فیلد انتخابی به صورت خودکار انجام می‌شود.

اضافه کردن ته جمع‌های ستون‌ها

این ته جمع‌ها که aggregate نام دارند باید در دو قسمت فعال شوند:

```
var productsDataSource = new kendo.data.DataSource({
                //...
                aggregate: [
                    { field: "Name", aggregate: "count" },
                    { field: "Price", aggregate: "sum" }
                ]
                //...
            });
```

ابتدا در قسمت data source مشخص می‌کنیم که چه تابع تجمعی قرار است به ازای یک فیلد خاص استفاده شود.
سپس این متدها را می‌توان مطابق فرمت hash syntax قالب‌های Kendo UI در قسمت footerTemplate هر ستون تعریف کرد:

```
$("#report-grid").kendoGrid({
                // ...
                columns: [
                    {
                        field: "Name", title: "نام محصول",
                        footerTemplate: "تعداد: #=count#"
                    },
                    {
                        field: "Price", title: "قیمت",
                        footerTemplate: "جمع: #=kendo.toString(sum,'c0')#"
                    }
                ]
                // ...
            });
```

فرمت شرطی اطلاعات

در ستون قیمت، می‌خواهیم اگر قیمتی بیش از 2490 بود، با پس زمینه‌ی قهوه‌ای و رنگ زرد نمایش داده شود. برای این منظور می‌توان یک قالب Kendo UI سفارشی را طراحی کرد:

```
<script type="text/x-kendo-template" id="priceTemplate">
        #if( Price > 2490 ) {#
        <span style="background:brown; color:yellow;">#=kendo.toString(Price,'c0')#</span>
        #} else {#
        #= kendo.toString(Price,'c0')#
        #}#
    </script>
```

سپس نحوه‌ی استفاده‌ی از آن به صورت ذیل خواهد بود:

```
$("#report-grid").kendoGrid({
                //...
                columns: [
                    {
                        field: "Price", title: "قیمت",
                        template: kendo.template($("#priceTemplate").html()),
                        footerTemplate: "جمع: #=kendo.toString(sum,'c0')#"
                    }
                ]
                //...
            });
```

توسط متد kendo.template امکان انتساب یک قالب سفارشی به خاصیت template یک ستون وجود دارد.

فرمت تاریخ میلادی به شمسی در حین نمایش

برای تبدیل سمت کلاینت تاریخ میلادی به شمسی از کتابخانه‌ی moment-jalaali.js کمک گرفته شده‌است:

```
<!--https://github.com/moment/moment/-->
<script src="js/cultures/moment.min.js" type="text/javascript"></script>
<!--https://github.com/jalaali/moment-jalaali-->
<script src="js/cultures/moment-jalaali.js" type="text/javascript"></script>
```

پس از آن تنها کافی است متد فرمت این کتابخانه را در قسمت template ستون تاریخ و توسط hash syntax قالب‌های Kendo UI بکار برد:

```
$("#report-grid").kendoGrid({
                //...
                columns: [
                    {
                        field: "AddDate", title: "تاریخ ثبت",
                        template: "#=moment(AddDate).format('jYYYY/jMM/jDD')#"
                    }
                ]
                //...
            });
```

اضافه کردن یک دکمه به نوار ابزار گرید

نوار ابزار Kendo UI Grid را نیز می‌توان توسط یک قالب سفارشی آن مقدار دهی کرد:

```
$("#report-grid").kendoGrid({
                // ...
                toolbar: [
                    { template: kendo.template($("#toolbarTemplate").html()) }
                ]
                // ...
            });
```

برای نمونه toolbarTemplate فوق را به نحو ذیل تعریف کرده‌ایم:

```
<script>
        // این اطلاعات برای تهیه خروجی سمت سرور مناسب هستند
        function getCurrentGridFilters() {
            var dataSource = $("#report-grid").data("kendoGrid").dataSource;
            var gridState = {
                page: dataSource.page(),
                pageSize: dataSource.pageSize(),
                sort: dataSource.sort(),
                group: dataSource.group(),
                filter: dataSource.filter()
            };
            return kendo.stringify(gridState);
        }
    </script>

    <script id="toolbarTemplate" type="text/x-kendo-template">
        <a class="k-button" href="\#" onclick="alert('gridState: ' + getCurrentGridFilters());">نوار
ابزار سفارشی</a>
    </script>
```

دکمه‌ی اضافه شده، وضعیت فیلتر data source متصل به گرید را بازگشت می‌دهد. برای مثال مشخص می‌کند که در چه صفحه‌ای با چه تعداد رکورد قرار داریم و همچنین وضعیت مرتب سازی، فیلتر و غیره چیست. از این اطلاعات می‌توان در سمت سرور برای تهیه‌ی خروجی‌های PDF یا اکسل استفاده کرد. وضعیت فیلتر اطلاعات مشخص است. بر همین مبنا کوئری گرفته و سپس می‌توان نتیجه‌ی آن‌را تبدیل به منبع داده تهیه خروجی مورد نظر کرد.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

KendoUI05.zip

نظرات خوانندگان

نویسنده: سروش
تاریخ: ۱۳۹۳/۰۸/۲۰ ۸:۵۱

با سلام هنگام Bulid پروژه با خطای زیر مواجه می‌شم

Error 1 Unable to locate 'C:\Users\Administrator\Desktop\KendoUI05\KendoUI05\.nuget\NuGet.exe' KendoUI05

دلیلش چیه ؟ البته Update-Package -Reinstall را انجام داده ام
متشکرم از شما

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۰ ۹:۵

- روی solution کلیک راست کنید و گزینه‌ی Enable NuGet Package Restore را انتخاب کنید.
- یا فایل NuGet.targets پوشه‌ی nuget. را باز کرده و دریافت خودکار nuget.exe را فعال کنید:

```
تغییر از
<DownloadNuGetExe Condition=" '$(DownloadNuGetExe)' == '' ">false</DownloadNuGetExe>
به
<DownloadNuGetExe Condition=" '$(DownloadNuGetExe)' == '' ">true</DownloadNuGetExe>
```

- و یا فایل nuget.exe را از این آدرس دریافت کنید و در پوشه‌ی nuget. کپی کنید.

نویسنده: سروش
تاریخ: ۱۳۹۳/۰۸/۲۰ ۹:۱۵

مرسی. فقط برای تغییر قالب KendoUi میشه کاری کرد برای ترکیب رنگهاش ؟ مثل JqueryUi Custom

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۰ ۹:۳۲

در انتهای مطلب «بررسی ساختار ویجت‌های وب Kendo UI » قسمت «تغییر قالب ویجت‌های وب»، توضیح داده شده.

عنوان: فعال سازی عملیات CRUD در Kendo UI Grid
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۱ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, Kendo UI

پیشنیاز بحث

- « فرمت کردن اطلاعات نمایش داده شده به کمک Kendo UI Grid »

Kendo UI Grid دارای امکانات ثبت، ویرایش و حذف توکاری است که در ادامه نحوه‌ی فعال سازی آن‌ها را بررسی خواهیم کرد. مثالی که در ادامه بررسی خواهد شد، در تکمیل مطلب « فرمت کردن اطلاعات نمایش داده شده به کمک Kendo UI Grid » است.

تنظیمات Data Source سمت کاربر

برای فعال سازی صفحه بندی سمت سرور ، با قسمت read منبع داده Kendo UI پیشتر آشنا شده بودیم. جهت فعال سازی قسمت‌های ثبت اطلاعات جدید (create)، به روز رسانی رکوردهای موجود (update) و حذف ردیفی مشخص (destroy) نیاز است تعاریف قسمت‌های متناظر را که هر کدام به آدرس مشخصی در سمت سرور اشاره می‌کنند، اضافه کنیم:

```
var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    },
                    create: {
                        url: "api/products",
                        contentType: 'application/json; charset=utf-8',
                        type: "POST"
                    },
```

```
                    update: {
                        url: function (product) {
                            return "api/products/" + product.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "PUT"
                    },
                    destroy: {
                        url: function (product) {
                            return "api/products/" + product.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "DELETE"
                    },
                    //...
                },
                schema: {
                    //...
                    model: {
                        id: "Id", // define the model of the data source. Required for validation and
property types.
                        fields: {
                            "Id": { type: "number", editable: false }, //تعیین نوع فیلد برای جستجوی پویا
مهم است
                            "Name": { type: "string", validation: { required: true } },
                            "IsAvailable": { type: "boolean" },
                            "Price": { type: "number", validation: { required: true, min: 1 } },
                            "AddDate": { type: "date", validation: { required: true } }
                        }
                    }
                },
                batch: false, // enable batch editing - changes will be saved when the user clicks the
"Save changes" button
                //...
            });
```

- همانطور که ملاحظه می‌کنید، حالت‌های update و destroy بر اساس Id ردیف انتخابی کار می‌کنند. این Id را باید در قسمت model مربوط به اسکیمای تعریف شده، دقیقا مشخص کرد. عدم تعریف فیلد id، سبب خواهد شد تا عملیات update نیز در حالت create تفسیر شود.

- به علاوه در اینجا به ازای هر فیلد، مباحث اعتبارسنجی نیز اضافه شده‌اند؛ برای مثال فیلدهای اجباری با required: true مشخص گردیده‌اند.

- اگر فیلدی نباید ویرایش شود (مانند فیلد Id)، خاصیت editable آن‌را false کنید.

- در data source امکان تعریف خاصیتی به نام batch نیز وجود دارد. حالت پیش فرض آن false است. به این معنا که در حالت ویرایش، تغییرات هر ردیفی، یک درخواست مجزا را به سمت سرور سبب خواهد شد. اگر آن‌را true کنید، تغییرات تمام ردیف‌ها در طی یک درخواست به سمت سرور ارسال می‌شوند. در این حالت باید به خاطر داشت که پارامترهای سمت سرور، از حالت یک شیء مشخص باید به لیستی از آن‌ها تغییر یابند.

مدیریت سمت سرور ثبت، ویرایش و حذف اطلاعات

در حالت ثبت، متد Post، توسط آدرس مشخص شده در قسمت create منبع داده گرید، فراخوانی می‌گردد:

```
namespace KendoUI06.Controllers
{
    public class ProductsController : ApiController
    {
        public HttpResponseMessage Post(Product product)
        {
            if (!ModelState.IsValid)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            var id = 1;
            var lastItem = ProductDataSource.LatestProducts.LastOrDefault();
            if (lastItem != null)
            {
                id = lastItem.Id + 1;
            }
            product.Id = id;
            ProductDataSource.LatestProducts.Add(product);
```

```
            var response = Request.CreateResponse(HttpStatusCode.Created, product);
            response.Headers.Location = new Uri(Url.Link("DefaultApi", new { id = product.Id }));
            // گرید آی دی جدید را به این صورت دریافت می‌کند
            response.Content = new ObjectContent<DataSourceResult>(
                new DataSourceResult { Data = new[] { product } }, new JsonMediaTypeFormatter());
            return response;
        }
    }
}
```

نکته‌ی مهمی که در اینجا باید به آن دقت داشت، نحوه‌ی بازگشت Id رکورد جدید ثبت شده‌است. در این مثال، قسمت schema منبع داده سمت کاربر به نحو ذیل تعریف شده‌است:

```
var productsDataSource = new kendo.data.DataSource({
                //...
                schema: {
                    data: "Data",
                    total: "Total",
                }
                //...
            });
```

از این جهت که خروجی متد Get بازگرداننده‌ی اطلاعات صفحه بندی شده ، از نوع DataSourceResult است و این نوع، دارای خواصی مانند Data، Total و Aggergate است:

```
namespace KendoUI06.Controllers
{
    public class ProductsController : ApiController
    {
        public DataSourceResult Get(HttpRequestMessage requestMessage)
        {
            var request = JsonConvert.DeserializeObject<DataSourceRequest>(
                requestMessage.RequestUri.ParseQueryString().GetKey(0)
            );

            var list = ProductDataSource.LatestProducts;
            return list.AsQueryable()
                       .ToDataSourceResult(request.Take, request.Skip, request.Sort, request.Filter);
        }
    }
}
```

بنابراین در متد Post نیز باید بر این اساس، response.Content را از نوع لیستی از DataSourceResult تعریف کرد تا Kendo UI Grid بداند که Id رکورد جدید را باید از فیلد Data، همانند تنظیمات schema منبع داده خود، دریافت کند.

```
response.Content = new ObjectContent<DataSourceResult>(
                                new DataSourceResult { Data = new[] { product } }, new
JsonMediaTypeFormatter());
```

اگر این تنظیم صورت نگیرد، Id رکورد جدید را در گرید، مساوی صفر مشاهده خواهید کرد و عملا بدون استفاده خواهد شد؛ زیرا قابلیت ویرایش و حذف خود را از دست می‌دهد.

متدهای حذف و به روز رسانی سمت سرور نیز چنین امضایی را خواهند داشت:

```
namespace KendoUI06.Controllers
{
    public class ProductsController : ApiController
    {
        public HttpResponseMessage Delete(int id)
        {
            var item = ProductDataSource.LatestProducts.FirstOrDefault(x => x.Id == id);
            if (item == null)
                return Request.CreateResponse(HttpStatusCode.NotFound);
```

```
            ProductDataSource.LatestProducts.Remove(item);

            return Request.CreateResponse(HttpStatusCode.OK, item);
        }

        [HttpPut] // Add it to fix this error: The requested resource does not support http method
'PUT'
        public HttpResponseMessage Update(int id, Product product)
        {
            var item = ProductDataSource.LatestProducts
                                        .Select(
                                            (prod, index) =>
                                                new
                                                {
                                                    Item = prod,
                                                    Index = index
                                                })
                                        .FirstOrDefault(x => x.Item.Id == id);
            if (item == null)
                return Request.CreateResponse(HttpStatusCode.NotFound);


            if (!ModelState.IsValid || id != product.Id)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            ProductDataSource.LatestProducts[item.Index] = product;
            return Request.CreateResponse(HttpStatusCode.OK);
        }
    }
}
```

حالت Update از HTTP Verb خاصی به نام Put استفاده می‌کند و ممکن است در این بین خطای The requested resource does not support http method 'PUT' را دریافت کنید. برای رفع آن ابتدا بررسی کنید که آیا Web.config برنامه دارای تعاریف ExtensionlessUrlHandler هست یا خیر. همچنین مزین کردن این متد با ویژگی HttpPut، مشکل را برطرف می‌کند.

تنظیمات Kendo UI Grid جهت فعال سازی CRUD

در ادامه کلیه تغییرات مورد نیاز جهت فعال سازی CRUD را در Kendo UI، به همراه مباحث بومی سازی عبارات متناظر با دکمه‌ها و صفحات خودکار مرتبط، مشاهده می‌کنید:

```
$("#report-grid").kendoGrid({
                //....
                editable: {
                    confirmation: "آیا مایل به حذف ردیف انتخابی هستید؟",
                    destroy: true, // whether or not to delete item when button is clicked
                    mode: "popup", // options are "incell", "inline", and "popup"
                    //template: kendo.template($("#popupEditorTemplate").html()), // template to use
for pop-up editing
                    update: true, // switch item to edit mode when clicked?
                    window: {
                        title: "مشخصات محصول"   // Localization for Edit in the popup window
                    }
                },
                columns: [
                //....
                    {
                        command: [
                            { name: "edit", text: "ویرایش" },
                            { name: "destroy", text: "حذف" }
                        ],
                        title: " ", width: "160px"
                    }
                ],
                toolbar: [
                    { name: "create", text: "افزودن ردیف جدید" },
                    { name: "save", text: "ذخیره‌ی تمامی تغییرات" },
                    { name: "cancel", text: "لغو کلیه‌ی تغییرات" },
                    { template: kendo.template($("#toolbarTemplate").html()) }
                ],
                messages: {
                    editable: {
                        cancelDelete: "لغو",
                        confirmation: "آیا مایل به حذف این رکورد هستید؟",
```

```
                    confirmDelete: "حذف"
                },
                commands: {
                    create: "افزودن ردیف جدید",
                    cancel: "لغو کلیه‌ی تغییرات",
                    save: "ذخیره‌ی تمامی تغییرات",
                    destroy: "حذف",
                    edit: "ویرایش",
                    update: "ثبت",
                    canceledit: "لغو"
                }
            }
        });
```

- ساده‌ترین حالت CRUD در Kendo UI با مقدار دهی خاصیت editable آن به true آغاز می‌شود. در این حالت، ویرایش درون سلولی یا incell فعال خواهد شد که مباحث batching ابتدای بحث، فقط در این حالت کار می‌کند. زمانیکه incell editing فعال است، کاربر می‌تواند تمام ردیف‌ها را ویرایش کرده و در آخر کار بر روی دکمه‌ی «ذخیره‌ی تمامی تغییرات» موجود در نوار ابزار، کلیک کند. در سایر حالات، هر بار تنها یک ردیف را می‌توان ویرایش کرد.
- برای فعال سازی تولید صفحات خودکار ویرایش و افزودن ردیف‌ها، نیاز است خاصیت editable را به نحوی که ملاحظه می‌کنید، مقدار دهی کرد. خاصیت mode آن سه حالت incell (پیش فرض)، inline و popup را پشتیبانی می‌کند.
- اگر حالت‌های inline و یا popup را فعال کردید، در انتهای ستون‌های تعریف شده، نیاز است ستون ویژه‌ای به نام command را مطابق تعاریف فوق، تعریف کنید. در این حالت دو دکمه‌ی ویرایش و ثبت، فعال می‌شوند و اطلاعات خود را از تنظیمات data source گرید دریافت می‌کنند. دکمه‌ی ویرایش در حالت incell کاربردی ندارد (چون در این حالت کاربر با کلیک درون یک سلول می‌تواند آن‌را مانند برنامه‌ی اکسل ویرایش کند). اما دکمه‌ی حذف در هر سه حالت قابل استفاده است.
- به نوار ابزار گرید، سه دکمه‌ی افزودن ردیف‌های جدید، ذخیره‌ی تمامی تغییرات و لغو تغییرات صورت گرفته، اضافه شده‌اند. این دکمه‌ها استاندارد بوده و در اینجا نحوه‌ی بومی سازی پیام‌های مرتبط را نیز مشاهده می‌کنید. همانطور که عنوان شد، دکمه‌های «تمامی تغییرات» در حالت فعال سازی batching در منبع داده و استفاده از incell editing معنا پیدا می‌کند. در سایر حالات این دو دکمه کاربردی ندارند. اما دکمه‌ی افزودن ردیف‌های جدید در هر سه حالت کاربرد دارد و یکسان است.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

KendoUI06.zip

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۱ ۹:۴۵

یک نکته‌ی تکمیلی

در مثال فوق از ASP.NET Web API استفاده شده‌است. اگر علاقمند به استفاده از WCF و یا حتی فایل‌های asmx قدیمی هم باشید، اینکار میسر است. مثال‌هایی را در این زمینه، در اینجا می‌توانید مشاهده کنید.

عنوان: **اعتبار سنجی ورودی‌های کاربر در Kendo UI**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۲ ۱۵:۱۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

در مطلب « فعال سازی عملیات CRUD در Kendo UI Grid » با نحوه‌ی تعریف مقدماتی اعتبارسنجی فیلدهای تعریف شده، آشنا شدید:

```
fields: {
      "Price": { type: "number", validation: { required: true, min: 1 } }
}
```

در ادامه نگاهی خواهیم داشت به جزئیات تکمیلی امکانات اعتبارسنجی ورودی‌های کاربر در Kendo UI.

Kendo UI Validation و HTML 5

در HTML 5 امکان تعریف نوع‌های خاص کنترل‌های ورودی کاربر مانند search، date، range، number، url، email و color وجود دارد. برای مثال در اینجا اگر کاربر تاریخ غیرمعتبری را وارد کند، مرورگر پیام اعتبارسنجی متناظری را به او نمایش خواهد داد.
همچنین در HTML 5 امکان افزودن ویژگی required نیز به کنترل‌های ورودی پیش بینی شده‌است. اما باید درنظر داشت که مرورگرهای قدیمی از این امکانات پشتیبانی نمی‌کنند. در این حالت Kendo UI با تشویق استفاده از روش معرفی شده در HTML 5، با آن یکپارچه شده و همچنین این قابلیت‌های اعتبارسنجی HTML 5 را در مرورگرهای قدیمی نیز میسر می‌کند. Kendo UI Validation جزو نسخه‌ی سورس باز Kendo UI با مجوز Apache نیز می‌باشد.
نمونه‌ای از امکانات اعتبارسنجی توکار HTML 5 را در اینجا مشاهده می‌کنید:

```
<input type="text" name="firstName" required />
<input type="text" name="twitter" pattern="https?://(?:www\.)?twitter\.com/.+i" />
<input type="number" name="age" min="1" max="42" />
<input type="number" name="age" min="1" max="100" step="2" />
<input type="url" name="url" />
<input type="email" name="email" />
```

یکپارچه سازی اعتبارسنجی Kendo UI با اعتبارسنجی HTML 5

در اینجا یک فرم تشکیل شده با ساختار HTML 5 را ملاحظه می‌کنید. هر دو فیلد ورودی، با ویژگی استاندارد required مزین شده‌اند. همچنین توسط ویژگی type، ورودی دوم جهت دریافت آدرس ایمیل معرفی شده‌است.
چون فیلد دوم دارای دو اعتبارسنجی تعریف شده است، دارای دو ویژگی data-* برای تعریف پیام‌های اعتبارسنجی متناظر نیز می‌باشد. الگوی تعریف آن‌ها data-[rule]-msg است.

```
<div class="k-rtl">
        <form id="testView">
            <label for="firstName">نام</label>
            <input id="firstName"
                   name="firstName"
                   type="text"
                   class="k-textbox"
                   required
                   validationmessage="لطفا نامی را وارد کنید">
            <br>
            <label for="emailId">آدرس پست الکترونیک</label>
            <input id="emailId"
                   name="emailId"
                   type="email"
                   dir="ltr"
                   required
                   class="k-textbox"
                   data-required-msg="لطفا ایمیلی را وارد کنید."
                   data-email-msg="ایمیل وارد شده معتبر نیست.">
            <br>
```

```
            <input type="submit" class="k-button" value="ارسال">
        </form>
    </div>

    <script type="text/javascript">
        $(function () {
            $("form#testView").kendoValidator();
        });
    </script>
```

تنها کاری که جهت یکپارچه سازی امکانات اعتبارسنجی Kendo UI با اعتبارسنجی استاندارد HTML 5 باید انجام داد، فراخوانی متد kendoValidator بر روی ناحیه‌ی مشخص شده است.

تعیین محل نمایش پیام‌های اعتبارسنجی

پیام‌های اعتبارسنجی Kendo UI به صورت خودکار در کنار فیلد متناظر با آن نمایش داده می‌شوند. اما اگر نیاز به تعیین مکان دستی آن‌ها وجود داشت (جهت خوانایی بهتر) باید به نحو ذیل عمل کرد:

```
<input type="text" id="name" name="name" required>
     <span class="k-invalid-msg" data-for="name"></span>
```

در اینجا span با کلاس k-invalid-msg و ویژگی data-for که به name کنترل ورودی اشاره می‌کند، محل نمایش پیام اعتبارسنجی متناظر با فیلد name خواهد بود.

تعریف سراسری پیام‌های اعتبارسنجی

در مثال فوق، به ازای تک تک فیلدهای ورودی، پیام اعتبارسنجی متناظر با required وارد شد. می‌توان این پیام‌ها را حذف کرد و در قسمت messages متد kendoValidator قرار داد:

```
<script type="text/javascript">
        $(function () {
            $("form#testView").kendoValidator({
                messages: {
                    // {0} would be replaced with the input element's name
                    required: '{0} را تکمیل کنید.',
                    email: 'ایمیل وارد شده معتبر نیست.'
                }
            });
        });
    </script>
```

- به این صورت پیام‌های اعتبارسنجی required و email، به صورت یکسانی به تمام المان‌های دارای این ویژگی‌ها اعمال خواهند شد.
- در این پیام‌ها {0} با مقدار ویژگی name فیلد ورودی متناظر جایگزین می‌شود.
- اگر هم در markup و هم در تعاریف kendoValidator، پیام‌های اعتبارسنجی تعریف شوند، حق تقدم با تعاریف markup خواهد بود.

اعتبارسنجی سفارشی سمت کاربر

علاوه بر امکانات استاندارد HTML 5، امکان تعریف دستورهای اعتبارسنجی سفارشی نیز وجود دارد:

```
<script type="text/javascript">
        $(function () {
            $("form#testView").kendoValidator({
                rules: {
                    customRule1: function (input) {
                        if (!input.is("[id=firstName]"))
                            return true;

                        var re = /^[A-Za-z]+$/;
                        return re.test(input.val());
                    }
                   //, customRule1: ….
                },
                messages: {
                    // {0} would be replaced with the input element's name
                    required: '{0} را تکمیل کنید.',
                    email: 'ایمیل وارد شده معتبر نیست.',
                    customRule1: 'اعداد مجاز نیستند.'
                }
            });
        });
    </script>
```

- همانطور که ملاحظه می‌کنید، برای تعریف منطق اعتبارسنجی سفارشی، باید از خاصیت rules ورودی متد kendoValidator شروع کرد. در اینجا نام یک متد callback دلخواهی را وارد کرده و سپس بر اساس منطق اعتبارسنجی مورد نظر، باید true/false را بازگشت داد. برای نمونه در این مثال اگر کاربر در فیلد نام، عدد وارد کند، ورودی او مورد قبول واقع نخواهد شد.
- باید دقت داشت که اگر بررسی input.is صورت نگیرد، منطق تعریف شده به تمام کنترل‌های صفحه اعمال می‌شود.
- پیام متناظر با این دستور سفارشی جدید، در قسمت messages، دقیقا بر اساس نام callback method تعریف شده در قسمت rules باید تعریف شود.

فراخوانی دستی اعتبارسنجی یک فرم

در حالت پیش فرض، با کلیک بر روی دکمه‌ی ارسال، اعتبارسنجی کلیه عناصر فرم به صورت خودکار انجام می‌شود. اگر بخواهیم در این بین یک پیام سفارشی را نیز نمایش دهیم می‌توان به صورت زیر عمل کرد:

```
<script type="text/javascript">
        $(function () {
            $("form#testView").submit(function (event) {
                event.preventDefault();
                var validator = $("form#testView").data("kendoValidator");
                if (validator.validate()) {
                    alert("validated!");
                } else {
                    alert("There is invalid data in the form.");
                }
```

```
            });

            $("form#testView").kendoValidator();
        });
    </script>
```

در اینجا رخداد submit فرم بازنویسی شده و متد validate آن بر اساس kendoValidator تعریف شده، به صورت دستی فراخوانی می‌شود.

اعتبارسنجی سفارشی در DataSource

در تعریف فیلدهای مدل DataSource، امکان تعریف اعتبارسنجی‌های پیش فرضی مانند rquired، min، max و امثال آن وجود دارد که نمونه‌ای از آن‌را در بحث فعال سازی CRUD در Kendo UI Grid مشاهده کردید:

```
fields: {
   "serviceName": {
    type: "string",
    defaultValue: "Inspection",
    editable: true,
    nullable: false,
    validation: { /*...*/ }
   },
   // ...
}
```

برای تعریف اعتبارسنجی سفارشی در اینجا، همانند متد kendoValidator نیاز است یک یا چند callback متد سفارشی را طراحی کرد:

```
schema: {
            model: {
                        id: "ProductID",
                        fields: {
                                    ProductID: { editable: false, nullable: true },
                                    ProductName: {
                                        validation: {
                                            required: true,
                                            custom1: function (input) {
                                                if (input.is("[name='ProductName']") && input.val()
!= "") {
                                                    input.attr("data-custom1-msg", "نام محصول باید
با حرف بزرگ انگلیسی شروع شود");
                                                    return /^[A-Z]/.test(input.val());
                                                }

                                                return true;
                                            }
                                          // ,custom2: ...
                                        }
                                    },
                                    UnitPrice: { type: "number", validation: { required: true, min:
1} },
                                    Discontinued: { type: "boolean" },
                                    UnitsInStock: { type: "number", validation: { min: 0, required:
true} }
                                }
                            }
                        }
```

نام این متد که نهایتا true/false بر می‌گرداند، اختیاری است. نام کنترل جاری همان نام فیلد متناظر است (جهت محدود کردن بازه‌ی اعمال منطق اعتبارسنجی). برای مقدار دهی پیام اعتبارسنجی از متد input.attr و الگوی data-[validationRuleName]-msg استفاده می‌شود. ضمنا به هر تعداد لازم می‌توان در اینجا custom rule تعریف کرد.

متد input.val() مقدار کنترل جاری را بر می‌گرداند. برای دسترسی به مقدار سایر کنترل‌ها می‌توان از روش $("#fieldName").val() استفاده کرد.

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۰۸/۲۲ ۱۷:۱

ممنون مطلبتون.
آیا اعتبار سنجی در حالت Inline Editing درون گرید هم به همین شکل هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۲ ۱۸:۱۱

بله. اطلاعات اعتبارسنجی فیلدهای خودش را از data source دریافت می‌کند. مثال فعال سازی CRUD در Kendo UI Grid را اجرا کنید، این مورد در آن لحاظ شده.

عنوان: Kendo UI MVVM
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۴ ۱۵:۵۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVVM, ASP.NET Web API, jQuery, KnockoutJS, Kendo UI

پیشنیازها

- « استفاده از Kendo UI templates »
- « اعتبار سنجی ورودی‌های کاربر در Kendo UI »
- « فعال سازی عملیات CRUD در Kendo UI Grid » جهت آشنایی با نحوه‌ی تعریف DataSource ایی که می‌تواند اطلاعات را ثبت، حذف و یا ویرایش کند.

در این مطلب قصد داریم به یک چنین صفحه‌ای برسیم که در آن در ابتدای نمایش، لیست ثبت نام‌های موجود، از سرور دریافت و توسط یک Kendo UI template نمایش داده می‌شود. سپس امکان ویرایش و حذف هر ردیف، وجود خواهد داشت، به همراه امکان افزودن ردیف‌های جدید. در این بین مدیریت نمایش لیست ثبت نام‌ها توسط امکانات binding توکار فریم ورک MVVM مخصوص Kendo UI صورت خواهد گرفت. همچنین کلیه اعمال مرتبط با هر ردیف نیز توسط data binding دو طرفه مدیریت خواهد شد.

Kendo UI MVVM

الگوی MVVM یا Model-View-ViewModel که برای اولین بار جهت کاربردهای WPF و Silverlight معرفی شد، برای ساده سازی اتصال تغییرات کنترل‌های برنامه به خواص ViewModel یک View کاربرد دارد. برای مثال با تغییر عنصر انتخابی یک DropDownList در یک View، بلافاصله خاصیت متصل به آن که در ViewModel برنامه تعریف شده‌است، مقدار دهی و به روز خواهد شد. هدف نهایی آن نیز جدا سازی منطق کدهای UI، از کدهای جاوا اسکریپتی سمت کاربر است. برای این منظور کتابخانه‌هایی مانند Knockout.js به صورت اختصاصی برای این کار تهیه شده‌اند؛ اما Kendo UI نیز جهت یکپارچگی هرچه تمامتر اجزای آن، دارای یک فریم ورک MVVM توکار نیز می‌باشد. طراحی آن نیز بسیار شبیه به Knockout.js است؛ اما با سازگاری 100 درصد با کل مجموعه.

پیاده سازی الگوی MVVM از 4 قسمت تشکیل می‌شود:

- Model که بیانگر خواص متناظر با اشیاء رابط کاربری است.
- View همان رابط کاربری است که به کاربر نمایش داده می‌شود.
- ViewModel واسطی است بین Model و View. کار آن انتقال داده‌ها و رویدادها از View به مدل است و در حالت binding دوطرفه، عکس آن نیز صحیح می‌باشد.
- Declarative data binding جهت رهایی برنامه نویس‌ها از نوشتن کدهای هماهنگ سازی اطلاعات المان‌های View و خواص ViewModel کاربرد دارد.

در ادامه این اجزا را با پیاده سازی مثالی که در ابتدای بحث مطرح شد، دنبال می‌کنیم.

تعریف Model و ViewModel

در سمت سرور، مدل ثبت نام برنامه چنین شکلی را دارد:

```
namespace KendoUI07.Models
{
    public class Registration
    {
        public int Id { set; get; }
        public string UserName { set; get; }
        public string CourseName { set; get; }
        public int Credit { set; get; }
        public string Email { set; get; }
        public string Tel { set; get; }
    }
}
```

در سمت کاربر، این مدل را به نحو ذیل می‌توان تعریف کرد:

```
<script type="text/javascript">
        $(function () {
            var model = kendo.data.Model.define({
                id: "Id",
                fields: {
                    Id: { type: 'number' }, // leave this set to 0 or undefined, so Kendo knows it is
new.
                    UserName: { type: 'string' },
                    CourseName: { type: 'string' },
                    Credit: { type: 'number' },
                    Email: { type: 'string' },
                    Tel: { type: 'string' }
                }
            });
        });
    </script>
```

و ViewModel برنامه در ساده‌ترین شکل آن اکنون چنین تعریفی را خواهد یافت:

```
<script type="text/javascript">
        $(function () {
            var viewModel = kendo.observable({
```

```
                accepted: false,
                course: new model()
            });
        });
    </script>
```

یک viewModel در Kendo UI به صورت یک observable object تعریف می‌شود که می‌تواند دارای تعدادی خاصیت و متد دلخواه باشد. هر خاصیت آن به یک عنصر HTML متصل خواهد شد. در اینجا این اتصال دو طرفه است؛ به این معنا که تغییرات UI به خواص viewModel و برعکس منتقل و منعکس می‌شوند.

اتصال ViewModel به View برنامه

تعریف فرم ثبت نام را در اینجا ملاحظه می‌کنید. فیلدهای مختلف آن بر اساس نکات اعتبارسنجی HTML 5 با ویژگی‌های خاص آن، مزین شده‌اند. جزئیات آن‌را در مطلب « اعتبار سنجی ورودی‌های کاربر در Kendo UI » پیشتر بررسی کرده‌ایم.
اگر به تعریف هر فیلد دقت کنید، ویژگی data-bind جدیدی را هم ملاحظه خواهید کرد:

```
<div id="coursesSection" class="k-rtl k-header">
        <div class="box-col">
            <form id="myForm" data-role="validator" novalidate="novalidate">
                <h3>ثبت نام</h3>
                <ul>
                    <li>
                        <label for="Id">Id</label>
                        <span id="Id" data-bind="text:course.Id"></span>
                    </li>
                    <li>
                        <label for="UserName">نام</label>
                        <input type="text" id="UserName" name="UserName" class="k-textbox"
                               data-bind="value:course.UserName"
                               required />
                    </li>
                    <li>
                        <label for="CourseName">دوره</label>
                        <input type="text" dir="ltr" id="CourseName" name="CourseName" required
                               data-bind="value:course.CourseName" />
                        <span class="k-invalid-msg" data-for="CourseName"></span>
                    </li>
                    <li>
                        <label for="Credit">مبلغ پرداختی</label>
                        <input id="Credit" name="Credit" type="number" min="1000" max="6000"
                               required data-max-msg="6000 و 1000 بین عددی" dir="ltr"
                               data-bind="value:course.Credit"
                               class="k-textbox k-input" />
                        <span class="k-invalid-msg" data-for="Credit"></span>
                    </li>
                    <li>
                        <label for="Email">پست الکترونیک</label>
                        <input type="email" id="Email" dir="ltr" name="Email"
                               data-bind="value:course.Email"
                               required class="k-textbox" />
                    </li>
                    <li>
                        <label for="Tel">تلفن</label>
                        <input type="tel" id="Tel" name="Tel" dir="ltr" pattern="\d{8}"
                               required class="k-textbox"
                               data-bind="value:course.Tel"
                               data-pattern-msg="8 رقم" />
                    </li>
                    <li>
                        <input type="checkbox" name="Accept"
                               data-bind="checked:accepted"
                               required />
                        شرایط دوره را قبول دارم.
                        <span class="k-invalid-msg" data-for="Accept"></span>
                    </li>
                    <li>
                        <button class="k-button"
                                data-bind="enabled: accepted, click: doSave"
                                type="submit">
                            ارسال
                        </button>
                        <button class="k-button" data-bind="click: resetModel">از نو</button>
```

```
                    </li>
                </ul>
                <span id="doneMsg"></span>
            </form>
        </div>
```

برای اتصال ViewModel تعریف شده به ناحیه‌ی مشخص شده با DIV ایی با Id مساوی coursesSection، می‌توان از متد kendo.bind استفاده کرد.

```
<script type="text/javascript">
        $(function () {
            var model = kendo.data.Model.define({
            // ...
            });

            var viewModel = kendo.observable({
            // ...
            });

            kendo.bind($("#coursesSection"), viewModel);
        });
    </script>
```

به این ترتیب Kendo UI به بر اساس تعریف data-bind یک فیلد، برای مثال تغییرات خواص course.UserName را به text box نام کاربر منتقل می‌کند و همچنین اگر کاربر اطلاعاتی را در این text box وارد کند، بلافاصله این تغییرات در خاصیت course.UserName منعکس خواهند شد.

```
<input type="text" id="UserName" name="UserName" class="k-textbox"
       data-bind="value:course.UserName"
       required />
```

بنابراین تا اینجا به صورت خلاصه، مدلی را توسط متد kendo.data.Model.define، معادل مدل سمت سرور خود ایجاد کردیم. سپس وهله‌ای از این مدل را به صورت یک خاصیت جدید دلخواهی در ViewModel تعریف شده توسط متد kendo.observable در معرض دید View برنامه قرار دادیم. در ادامه اتصال ViewModel و View، با فراخوانی متد kendo.bind انجام شد. اکنون برای دریافت تغییرات کنترل‌های برنامه، تنها کافی است ویژگی‌های data-bind ایی را به آن‌ها اضافه کنیم.
در ناحیه‌ی تعریف شده توسط متد kendo.bind، کلیه خواص ViewModel در دسترس هستند. برای مثال اگر به تعریف ViewModel دقت کنید، یک خاصیت دیگر به نام accepted با مقدار false نیز در آن تعریف شده‌است (این خاصیت چون صرفا کاربرد UI داشت، در model برنامه قرار نگرفت). از آن برای اتصال checkbox تعریف شده، به button ارسال اطلاعات، استفاده کرده‌ایم:

```
<input type="checkbox" name="Accept"
       data-bind="checked:accepted"
       required />

<button class="k-button"
        data-bind="enabled: accepted, click: doSave"
        type="submit">
       ارسال
</button>
```

برای مثال اگر کاربر این checkbox را انتخاب کند، مقدار خاصیت accepted، مساوی true خواهد شد. تغییر مقدار این خاصیت، توسط ViewModel بلافاصله در کل ناحیه coursesSection منتشر می‌شود. به همین جهت ویژگی enabled: accepted که به معنای مقید بودن فعال یا غیرفعال بودن دکمه بر اساس مقدار خاصیت accepted است، دکمه را فعال می‌کند، یا برعکس و برای انجام این عملیات نیازی نیست کدنویسی خاصی را انجام داد. در اینجا بین checkbox و button یک سیم کشی برقرار است.

ارسال داده‌های تغییر کرده‌ی ViewModel به سرور

تا اینجا 4 جزء اصلی الگوی MVVM که در ابتدای بحث عنوان شد، تکمیل شده‌اند. مدل اطلاعات فرم تعریف گردید. ViewModel ایی

که این خواص را به المان‌های فرم متصل می‌کند نیز در ادامه اضافه شده‌است. توسط ویژگی‌های data-bind کار Declarative data binding انجام می‌شود.

در ادامه نیاز است تغییرات ViewModel را به سرور، جهت ثبت، به روز رسانی و حذف نهایی منتقل کرد.

```
<script type="text/javascript">
        $(function () {
            var model = kendo.data.Model.define({
                //...
            });

            var dataSource = new kendo.data.DataSource({
                type: 'json',
                transport: {
                    read: {
                        url: "api/registrations",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    },
                    create: {
                        url: "api/registrations",
                        contentType: 'application/json; charset=utf-8',
                        type: "POST"
                    },
                    update: {
                        url: function (course) {
                            return "api/registrations/" + course.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "PUT"
                    },
                    destroy: {
                        url: function (course) {
                            return "api/registrations/" + course.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "DELETE"
                    },
                    parameterMap: function (data, type) {
                        // Convert to a JSON string.  Without this step your content will be form
encoded.
                        return JSON.stringify(data);
                    }
                },
                schema: {
                    model: model
                },
                error: function (e) {
                    alert(e.errorThrown);
                },
                change: function (e) {
                    // فراخوانی در زمان دریافت اطلاعات از سرور و یا تغییرات محلی
                    viewModel.set("coursesDataSourceRows", new
kendo.data.ObservableArray(this.view()));
                }
            });

            var viewModel = kendo.observable({
                //...
            });

            kendo.bind($("#coursesSection"), viewModel);
            dataSource.read(); // دریافت لیست موجود از سرور در آغاز کار
        });
    </script>
```

در اینجا تعریف DataSource کار با منبع داده راه دور ASP.NET Web API را مشاهده می‌کنید. تعاریف اصلی آن با تعاریف مطرح شده در مطلب « فعال سازی عملیات CRUD در Kendo UI Grid » یکی هستند. هر قسمت آن مانند read، create، update و destory به یکی از متدهای کنترلر ASP.NET Web API اشاره می‌کنند. حالت‌های update و destroy بر اساس Id ردیف انتخابی کار می‌کنند. این Id را باید در قسمت model مربوط به اسکیمای تعریف شده، دقیقا مشخص کرد. عدم تعریف فیلد id، سبب خواهد شد تا عملیات update نیز در حالت create تفسیر شود.

متصل کردن DataSource به ViewModel

تا اینجا DataSource ایی جهت کار با سرور تعریف شده‌است؛ اما مشخص نیست که اگر رکوردی اضافه شد، چگونه باید اطلاعات خودش را به روز کند. برای این منظور خواهیم داشت:

```
<script type="text/javascript">
        $(function () {
            $("#coursesSection").kendoValidator({
                // ...
            });

            var model = kendo.data.Model.define({
                // ...
            });

            var dataSource = new kendo.data.DataSource({
                // ...
            });

            var viewModel = kendo.observable({
                accepted: false,
                course: new model(),
                doSave: function (e) {
                    e.preventDefault();
                    console.log("this", this.course);
                    var validator = $("#coursesSection").data("kendoValidator");
                    if (validator.validate()) {
                        if (this.course.Id == 0) {
                            dataSource.add(this.course);
                        }
                        dataSource.sync(); // push to the server
                        this.set("course", new model()); // reset controls
                    }
                },
                resetModel: function (e) {
                    e.preventDefault();
                    this.set("course", new model());
                }
             });

            kendo.bind($("#coursesSection"), viewModel);
            dataSource.read(); // دریافت لیست موجود از سرور در آغاز کار
        });
    </script>
```

همانطور که در تعاریف تکمیلی viewModel مشاهده می‌کنید، اینبار دو متد جدید دلخواه doSave و resetModel را اضافه کرده‌ایم. در متد doSave، ابتدا بررسی می‌کنیم آیا اعتبارسنجی فرم با موفقیت انجام شده‌است یا خیر. اگر بله، توسط متد add منبع داده، اطلاعات فرم جاری را توسط شیء course که هم اکنون به تمامی فیلدهای آن متصل است، اضافه می‌کنیم. در اینجا بررسی شده‌است که آیا Id این اطلاعات صفر است یا خیر. از آنجائیکه از همین متد برای به روز رسانی نیز در ادامه استفاده خواهد شد، در حالت به روز رسانی، Id شیء ثبت شده، از طرف سرور دریافت می‌گردد. بنابراین غیر صفر بودن این Id به معنای عملیات به روز رسانی است و در این حالت نیازی نیست کار بیشتری را انجام داد؛ زیرا شیء متناظر با آن پیشتر به منبع داده اضافه شده‌است.

استفاده از متد add صرفا به معنای مطلع کردن منبع داده محلی از وجود رکوردی جدید است. برای ارسال این تغییرات به سرور، از متد sync آن می‌توان استفاده کرد. متد sync بر اساس متد add یک درخواست POST، بر اساس شیءایی که Id غیر صفر دارد، یک درخواست PUT و با فراخوانی متد remove بر روی منبع داده، یک درخواست DELETE را به سمت سرور ارسال می‌کند.

متد دلخواه resetModel سبب مقدار دهی مجدد شیء course با یک وهله‌ی جدید از شیء model می‌شود. همینقدر برای پاک کردن تمامی کنترل‌های صفحه کافی است.

تا اینجا دو متد جدید را در ViewModel برنامه تعریف کرده‌ایم. در مورد نحوه‌ی اتصال آن‌ها به View، به کدهای دو دکمه‌ی موجود در فرم دقت کنید:

```
<button class="k-button"
        data-bind="enabled: accepted, click: doSave"
        type="submit">
```

```
        ارسال
</button>
<button class="k-button" data-bind="click: resetModel">از نو</button>
```

این متدها نیز توسط ویژگی‌های data-bind به هر دکمه نسبت داده شده‌اند. به این ترتیب برای مثال با کلیک کاربر بر روی دکمه‌ی submit، متد doSave موجود در ViewModel فراخوانی می‌شود.

مدیریت سمت سرور ثبت، ویرایش و حذف اطلاعات

در حالت ثبت، متد Post توسط آدرس مشخص شده در قسمت create منبع داده، فراخوانی می‌گردد. نکته‌ی مهمی که در اینجا باید به آن دقت داشت، نحوه‌ی بازگشت Id رکورد جدید ثبت شده‌است. اگر این تنظیم صورت نگیرد، Id رکورد جدید را در لیست، مساوی صفر مشاهده خواهید کرد و منبع داده این رکورد را همواره به عنوان یک رکورد جدید، مجددا به سرور ارسال می‌کند.

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using KendoUI07.Models;

namespace KendoUI07.Controllers
{
    public class RegistrationsController : ApiController
    {
        public HttpResponseMessage Delete(int id)
        {
            var item = RegistrationsDataSource.LatestRegistrations.FirstOrDefault(x => x.Id == id);
            if (item == null)
                return Request.CreateResponse(HttpStatusCode.NotFound);

            RegistrationsDataSource.LatestRegistrations.Remove(item);
            return Request.CreateResponse(HttpStatusCode.OK, item);
        }

        public IEnumerable<Registration> Get()
        {
            return RegistrationsDataSource.LatestRegistrations;
        }

        public HttpResponseMessage Post(Registration registration)
        {
            if (!ModelState.IsValid)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            var id = 1;
            var lastItem = RegistrationsDataSource.LatestRegistrations.LastOrDefault();
            if (lastItem != null)
            {
                id = lastItem.Id + 1;
            }
            registration.Id = id;
            RegistrationsDataSource.LatestRegistrations.Add(registration);

            // ارسال آی دی مهم است تا از ارسال رکوردهای تکراری جلوگیری شود
            return Request.CreateResponse(HttpStatusCode.Created, registration);
        }

        [HttpPut] // Add it to fix this error: The requested resource does not support http method
'PUT'
        public HttpResponseMessage Update(int id, Registration registration)
        {
            var item = RegistrationsDataSource.LatestRegistrations
                                        .Select(
                                            (prod, index) =>
                                                new
                                                {
                                                    Item = prod,
                                                    Index = index
                                                })
                                        .FirstOrDefault(x => x.Item.Id == id);
            if (item == null)
```

```
                return Request.CreateResponse(HttpStatusCode.NotFound);


            if (!ModelState.IsValid || id != registration.Id)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            RegistrationsDataSource.LatestRegistrations[item.Index] = registration;
            return Request.CreateResponse(HttpStatusCode.OK);
        }
    }
}
```

در اینجا بیشتر امضای این متدها مهم هستند، تا منطق پیاده سازی شده در آن‌ها. همچنین بازگشت Id رکورد جدید، توسط متد Post نیز بسیار مهم است و سبب می‌شود تا DataSource بداند با فراخوانی متد sync آن، باید عملیات Post یا create انجام شود یا Put و update.

نمایش آنی اطلاعات ثبت شده در یک لیست

ردیف‌های اضافه شده به منبع داده را می‌توان بلافاصله در همان سمت کلاینت توسط Kendo UI Template که قابلیت کار با ViewModelها را دارد، نمایش داد:

```
<div id="coursesSection" class="k-rtl k-header">
        <div class="box-col">
            <form id="myForm" data-role="validator" novalidate="novalidate">
                        <!--فرم بحث شده در ابتدای مطلب-->
            </form>
        </div>
        <div id="results">
            <table class="metrotable">
                <thead>
                    <tr>
                        <th>Id</th>
                        <th>نام</th>
                        <th>دوره</th>
                        <th>هزینه</th>
                        <th>ایمیل</th>
                        <th>تلفن</th>
                        <th></th>
                        <th></th>
                    </tr>
                </thead>
                <tbody data-template="row-template" data-bind="source: coursesDataSourceRows"></tbody>
                <tfoot data-template="footer-template" data-bind="source: this"></tfoot>
            </table>
            <script id="row-template" type="text/x-kendo-template">
                <tr>
                    <td data-bind="text: Id"></td>
                    <td data-bind="text: UserName"></td>
                    <td dir="ltr" data-bind="text: CourseName"></td>
                    <td>
                        #: kendo.toString(get("Credit"), "c0") #
                    </td>
                    <td data-bind="text: Email"></td>
                    <td data-bind="text: Tel"></td>
                    <td><button class="k-button" data-bind="click: deleteCourse">حذف</button></td>
                    <td><button class="k-button" data-bind="click: editCourse">ویرایش</button></td>
                </tr>
            </script>
            <script id="footer-template" type="text/x-kendo-template">
                <tr>
                    <td colspan="3"></td>
                    <td>
                        جمع کل: #: kendo.toString(totalPrice(), "c0") #
                    </td>
                    <td colspan="2"></td>
                    <td></td>
                    <td></td>
                </tr>
            </script>
        </div>
    </div>
```

در ناحیه‌ی coursesSection که توسط متد kendo.bind به viewModel برنامه متصل شده‌است، یک جدول را برای نمایش ردیف‌های ثبت شده توسط کاربر اضافه کرده‌ایم. thead آن بیانگر سر ستون جدول است. قسمت tbody و tfoot این جدول توسط دو Kendo UI Template مقدار دهی شده‌اند. هر کدام نیز منبع داده‌اشان را از view model دریافت می‌کنند. در row-template معادل خواص شیء course را مشاهده می‌کنید. در footer-template متد totalPrice برای نمایش جمع ستون هزینه اضافه شده‌است. بنابراین مطابق این قسمت از View، به یک خاصیت جدید coursesDataSourceRows و سه متد deleteCourse، editCourse و totalPrice نیاز است:

```
<script type="text/javascript">
        $(function () {
            // ...
            var viewModel = kendo.observable({
                accepted: false,
                course: new model(),
                coursesDataSourceRows: new kendo.data.ObservableArray([]),
                doSave: function (e) {
                       // ...
                },
                resetModel: function (e) {
                      // ...
                },
                totalPrice: function () {
                    var sum = 0;
                    $.each(this.get("coursesDataSourceRows"), function (index, item) {
                        sum += item.Credit;
                    });
                    return sum;
                },
                deleteCourse: function (e) {
                    // the current data item is passed as the "data" field of the event argument
                    var course = e.data;
                    dataSource.remove(course);
                    dataSource.sync(); // push to the server
                },
                editCourse: function(e) {
                    // the current data item is passed as the "data" field of the event argument
                    var course = e.data;
                    this.set("course", course);
                }
            });

            kendo.bind($("#coursesSection"), viewModel);
            dataSource.read(); // دریافت لیست موجود از سرور در آغاز کار
        });
    </script>
```

نحوه‌ی اتصال خاصیت جدید coursesDataSourceRows که به عنوان منبع داده ردیف‌های row-template عمل می‌کند، به این صورت است:

- ابتدا خاصیت دلخواه coursesDataSourceRows به viewModel اضافه می‌شود تا در ناحیه‌ی coursesSection در دسترس قرار گیرد.

- سپس اگر به انتهای تعریف DataSource دقت کنید، داریم:

```
<script type="text/javascript">
        $(function () {
            var dataSource = new kendo.data.DataSource({
                //...
                change: function (e) {
                    // فراخوانی در زمان دریافت اطلاعات از سرور و یا تغییرات محلی
                    viewModel.set("coursesDataSourceRows", new
kendo.data.ObservableArray(this.view()));
                }
            });
        });
    </script>
```

متد change آن، هر زمانیکه اطلاعاتی در منبع داده تغییر کنند یا اطلاعاتی به سمت سرور ارسال یا دریافت گردد، فراخوانی می‌شود. در همینجا فرصت خواهیم داشت تا خاصیت coursesDataSourceRows را جهت نمایش اطلاعات موجود در منبع داده،

مقدار دهی کنیم. همین مقدار دهی ساده سبب اجرای row-template برای تولید ردیف‌های جدول می‌شود. استفاده از new kendo.data.ObservableArray سبب خواهد شد تا اگر اطلاعاتی در فرم برنامه تغییر کند، این اطلاعات بلافاصله در لیست گزارش برنامه نیز منعکس گردد.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

KendoUI07.zip

عنوان: **ارث بری prototype ای در جاوا اسکریپت به زبان خیلی ساده**
نویسنده: فرید بکران
تاریخ: ۲۳:۰ ۱۳۹۳/۰۸/۲۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, AngularJS

انگیزه اصلی این نوشته شروع کار با AngularJs و استفاده از scope در این کتابخانه است. بیشتر دوستانی که کار با این کتابخانه را شروع می‌کنند و تجربه زیادی با جاوا اسکریپت ندارند، با مفهوم ارث بری scope مشکل پیدا می‌کنند.

ارث بری در scope های AngularJs موضوع پیچیده و عجیب و غریبی نیست. در واقع همان ارث بری prototype ای است که جاوا اسکریپت پشتیبانی می‌کند.

این روش توضیح خیلی ساده ای دارد.

در هنگام دسترسی به مقدار یک خصوصیت روی یک شی اگر آن خصوصیت در شی مورد نظر وجود نداشته باشد جاوا اسکریپت یک سطح در زنجیره‌ی prototype ها بالا رفته و به شی پدر دسترسی پیدا کرده و در آن به دنبال مقدار خصوصیت می‌گردد. این کار را آن قدر ادامه می‌دهد تا به بالاترین سطح برسد و دیگر چیزی پیدا نکند.

این بالا رفتن در زنجیره‌ی prototype ها عملا با دسترسی به خصوصیت prototype انجام می‌شود.

فرض کنید دو شی (دقت کنید که می‌گویم شی) به نام‌های employee و person داریم. این دو شی را به صورت زیر تعریف می‌کنیم.

```
var person = { type: '', name:'No Name' };
var employee = {  };
```

شی employee الان هیچ خصوصیت ای ندارد. و دسترسی به هر خصوصیت ای از آن هیچ نتیجه‌ای در بر نخواهد داشت.

```
console.log('Before Inheritance -> employee.name = ' + employee.name);
```

با مقدار دهی کردن خصوصیت prototype مربوط به employee به person این شی را از person ارث بری می‌کنیم.

```
employee.__proto__ = person;
```

بعد از اجرا شدن این خط از برنامه هنگام دسترسی پیدا کردن به مقدار name، مقدار اصلی آن که در شی person وارد شده بود را خواهیم دید.

ملاحظه کردید که وقتی خصوصیت name در شی مورد نظر وجود نداشت به شی پدر رجوع شد و مقدار خصوصیت مربوطه از آن بدست آمد.

الان فرض کنید که در قسمتی از برنامه خواستیم مقدار name در شی employee را به مقدار مشخصی تغییر دهیم. به طور مثال:

```
employee.name = 'farid';
console.log('After Assiginig -> employee.name = ' + employee.name);
console.log('After Assiginig -> person.name = ' + person.name);
```

با چاپ کردن مقادیر person.name و employee.name انتظار دارید چه نتیجه ای ببینید؟

اگر از زبان‌های شی گرایی مانند #C آمده باشید احتمالا خواهید گفت مقادیر یکسان خواهند بود. ولی در واقع این گونه نیست. مقدار person.name همان مقدار اولیه ما خواهد بود و مقدار employee.name نیز ‘farid’.

دلیل این رفتار یک نکته ساده و اساسی است.

جاوا اسکریپت فقط در زمان دسترسی به یک خصوصیت در صورت پیدا نکردن آن در شی مورد نظر ما به سطوح بالاتر prototype ای رفته و دنبال آن خصوصیت می‌گردد.

اگر ما قصد مقدار دهی به یک خصوصیت را داشته باشیم و خصوصیت مورد نظر ما در شی وجود نداشته باشد جاوا اسکریپت یک نسخه محلی از خصوصیت برای آن شی می‌سازد و مقدار ما را به آن می‌دهد.

در واقع در مثال ما هنگام مقدار دهی به employee.name آن خصوصیت در شی موجود نبود و یک نسخه محلی به نام name در شی ایجاد شد و دفعه بعدی که دسترسی به مقدار این خصوصیت اتفاق افتد این خصوصیت به صورت محلی وجود خواهد داشت و جاوا اسکریپت به سطوح بالاتر نخواهد رفت .

تمام کدهای بالا در bin زیر موجود هستند.

Prototypal Inheritance in Javascript

الان فرض کنید شیءهای ما به این صورت هستند:

```
var person = {
  info : { name: 'No Name', type: '' }
};
var employee = {};
```

به همان صورت بالا ارث بری می‌کنیم.

```
employee.__proto__ = person;
```

و سپس name را مقداردهی می‌کنیم.

```
employee.info.name = 'farid';
```

و مقادیر را چاپ می‌کنیم.

```
console.log('After Assiginig -> employee.name = ' + employee.info.name);
console.log('After Assiginig -> person.name = ' + person.info.name);
```

ملاحظه خواهید کرد که مقادیر مساوی هستند.
دلیل این امر به زبان ساده این است که وقتی اقدام به مقدار دهی name در شی employee کردیم در واقع قبل از مقدار دهی اصلی name یک دسترسی به شی info نیاز بود و دسترسی به شیء با استفاده از همان قانونی که مطرح کردیم انجام شده و شیء مربوط به person برگردانده شده است. چون name یک خصوصیت از info است نه employee.
سوالی که می‌توان مطرح کرد این است که در صورت نوشتن این خط کد چه اتفاقی خواهد افتاد؟

```
employee.info = {
  name: 'farhad'
};
```

Prototypal Inheritance with objects

با توجه به مطالب گفته شده باید قادر به حدس زدن نتیجه خواهید بود. **نکته: روش‌های کار با prototype در این نوشته فقط جنبه آموزشی و توضیحی دارد و روش درست استفاده از prototype این نیست.**

عنوان: **یکپارچه سازی سیستم اعتبارسنجی ASP.NET MVC با Kendo UI validator**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۷ ۱۱:۵۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVC, jQuery, Kendo UI

روش پیش فرض اعتبارسنجی برنامه‌های ASP.NET MVC، استفاده از دو افزونه‌ی jquery.validate و jquery.validate.unobtrusive است.

```
<script src="~/Scripts/jquery.validate.min.js" type="text/javascript"></script>
    <script src="~/Scripts/jquery.validate.unobtrusive.min.js" type="text/javascript"></script>
```

کار اصلی اعتبارسنجی، توسط افزونه‌ی jquery.validate انجام می‌شود و فایل jquery.validate.unobtrusive صرفا یک وفق دهنده و مترجم ویژگی‌های خاص ASP.NET MVC به jquery.validate است.

عدم سازگاری پیش فرض jquery.validate با بعضی از ویجت‌های Kendo UI

در حالت استفاده از Kendo UI، این سیستم هنوز هم کار می‌کند؛ اما با یک مشکل. اگر برای مثال از kendoComboBox استفاده کنید، اعتبارسنجی‌های تعریف شده در برنامه، توسط jquery.validate دیده نخواهند شد. برای مثال فرض کنید یک چنین مدلی در اختیار View برنامه قرار گرفته است:

```
public class OrderDetailViewModel
    {
        [StringLength(15)]
        [Required]
        public string Destination { get; set; }
    }
```

با این View که در آن به فیلد Destination، یک kendoComboBox متصل شده‌است:

```
@model Mvc4TestViewModel.Models.OrderDetailViewModel

@using (Ajax.BeginForm(actionName: "Index", controllerName: "Home",
                       ajaxOptions:new AjaxOptions(),
                       htmlAttributes: new { id = "Form1", name ="Form1" }, routeValues: new { }
                       ))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>OrderDetail</legend>
        <div class="editor-label">
            @Html.LabelFor(model => model.Destination)
        </div>
        <div class="editor-field">
            @Html.TextBoxFor(model => model.Destination, new { @class = "k-textbox" })
            @Html.ValidationMessageFor(model => model.Destination)
        </div>

        <p>
            <button class="k-button" type="submit" title="Sumbit">
                Sumbit
            </button>
        </p>
    </fieldset>
}

@section JavaScript
{
    <script type="text/javascript">
        $(function () {
            $("#Destination").kendoComboBox({
                dataSource: [
                    "loc 1",
                    "loc 2"
                ]
```

```
            });
    </script>
}
```

اگر برنامه را اجرا کنید و بر روی دکمه‌ی submit کلیک نمائید، ویژگی Required عمل نخواهد کرد و عملا در سمت کاربر اعتبارسنجی رخ نمی‌دهد.

```
◢ <div class="editor-field">
  ◢ <span class="k-widget k-combobox k-header k-textbox" style="">
    ◢ <span tabindex="-1" class="k-dropdown-wrap k-state-default" unselectable="on">
        <input name="Destination_input" tabindex="0" class="k-input k-textbox" role="combobox" aria-
        busy="false" aria-disabled="false" aria-expanded="false" aria-readonly="false" aria-
        activedescendant="Destination_option_selected" aria-owns="Destination_listbox" style="width: 100%;"
        aria-autocomplete="list" type="text" autocomplete="off"></input>
      ▷ <span tabindex="-1" class="k-select" unselectable="on">…</span>
      </span>
      <input name="Destination" class="k-textbox input-validation-error" id="Destination" aria-
      disabled="false" aria-invalid="true" aria-readonly="false" aria-required="true" aria-
      describedby="Destination-error" style="display: none;" type="text" data-val-required="The Destination
      field is required." data-val="true" data-val-length-max="15" data-val-length="The field Destination must
      be a string with a maximum length of 15." data-role="combobox" value=""></input>
    </span>
  ▷ <span class="field-validation-error" data-valmsg-replace="true" data-valmsg-for="Destination">…</span>
  </div>
```

همانطور که در تصویر مشاهده می‌کنید، با اتصال kendoComboBox به یک فیلد، این فیلد در حالت مخفی قرار می‌گیرد و ویجت کندو یو آی بجای آن نمایش داده خواهد شد. در این حالت چون در فایل jquery.validate.js چنین تنظیمی وجود دارد:

```
$.extend( $.validator, {
    defaults: {
     //…
     ignore: ":hidden",
```

به صورت پیش فرض از اعتبارسنجی فیلدهای مخفی صرفنظر می‌شود.
راه حل آن نیز ساده‌است. تنها باید خاصیت ignore را بازنویسی کرد و تغییر داد:

```
<script type="text/javascript">
        $(function () {
            var form = $('#Form1');
            form.data('validator').settings.ignore = ''; // default is ":hidden".
        });
    </script>
```

در اینجا صرفا خاصیت ignore فرم یک، جهت درنظر گرفتن فیلدهای مخفی تغییر کرده‌است. اگر می‌خواهید این تنظیم را به تمام فرم‌ها اعمال کنید، می‌توان از دستور ذیل استفاده کرد:

```
<script type="text/javascript">
    $.validator.setDefaults({
        ignore: ""
    });
</script>
```

یکپارچه کردن سیستم اعتبارسنجی Kendo UI با سیستم اعتبارسنجی ASP.NET MVC

در مطلب « اعتبار سنجی ورودی‌های کاربر در Kendo UI » با زیرساخت اعتبارسنجی Kendo UI آشنا شدید. برای اینکه بتوان این سیستم را با ASP.NET MVC یکپارچه کرد، نیاز است دو کار صورت گیرد:

الف) تعریف فایل kendo.aspnetmvc.js به صفحه اضافه شود:

```
<script src="~/Scripts/kendo.aspnetmvc.js" type="text/javascript"></script>
```

ب) همانند قبل، متد kendoValidator بر روی فرم فراخوانی شود تا سیستم اعتبارسنجی Kendo UI در این ناحیه فعال گردد:

```
<script type="text/javascript">
        $(function () {
            $("form").kendoValidator();
        });
    </script>
```

پس از آن خواهیم داشت:

فایل kendo.aspnetmvc.js که در بسته‌ی مخصوص Kendo UI تهیه شده برای ASP.NET MVC موجود است (در پوشه‌ی js آن)، عملکردی مشابه فایل jquery.validate.unobtrusive مایکروسافت دارد. کار آن وفق دادن و ترجمه‌ی اعتبارسنجی unobtrusive به روش Kendo UI است.

این فایل را از اینجا می‌توانید دریافت کنید:

kendo.mvc.zip

البته باید دقت داشت که در حال حاضر فقط ویژگی‌های ذیل از ASP.NET MVC توسط kendo.aspnetmvc.js پشتیبانی می‌شوند :

```
Required
```

```
StringLength
Range
RegularExpression
```

برای تکمیل آن می‌توان از یک پروژه‌ی سورس باز به نام Moon.Validation for KendoUI Validator استفاده کرد. برای مثال remote validation مخصوص Kendo UI را اضافه کرده‌است .

نظرات خوانندگان

نویسنده: امیر
تاریخ: ۱۳۹۳/۰۸/۲۸ ۱۳:۰

سئوالی برای من پیش اومد امکان استفاده از مدل mvc داخل کنترل‌های جاوااسکریپت کندو هست ؟ مثلا اگه combobox انتخاب بشه آیدیش در مدل اصلی انتخاب بشه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۸ ۱۳:۴۶

« استفاده ازExpressionها جهت ایجاد Strongly typed view در ASP.NET MVC »

عنوان: بررسی ویجت Kendo UI File Upload
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۳ ۱۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

Kendo UI به همراه یک ویجت وب مخصوص ارسال فایل‌ها به سرور نیز هست. این ویجت قابلیت ارسال چندین فایل با هم را به صورت Ajax ایی دارا است و همچنین کاربران می‌توانند فایل‌ها را با کشیدن و رها کردن بر روی آن، به لیست فایل‌های قابل ارسال اضافه کنند.
ارسال فایل Ajax ایی آن توسط HTML5 File API صورت می‌گیرد که در تمام مرورگرهای جدید پشتیبانی خوبی از آن وجود دارد. در مرورگرهای قدیمی‌تر، به صورت خودکار همان حالت متداول ارسال همزمان فایل‌ها را فعال می‌کند (یا همان post back معمولی).

فعال سازی مقدماتی kendoUpload

ابتدایی‌ترین حالت کار با kendoUpload، فعال سازی حالت post back معمولی است؛ به شرح زیر:

```
<form method="post" action="submit" enctype="multipart/form-data">
  <div>
    <input name="files" id="files" type="file" />
    <input type="submit" value="Submit" class="k-button" />
  </div>
</form>
<script>
  $(document).ready(function() {
     $("#files").kendoUpload();
  });
</script>
```

در این حالت صرفا input با نوع file، با ظاهری سازگار با سایر کنترل‌های Kendo UI به نظر می‌رسد و عملیات ارسال فایل، همانند قبل به همراه یک post back است. این روش برای حالتی مفید است که بخواهید یک فایل را به همراه سایر عناصر فرم در طی یک مرحله به سمت سرور ارسال کنید.

فعال سازی حالت ارسال فایل Ajax ایی kendoUpload

برای فعال سازی ارسال Ajax ایی فایل‌ها در Kendo UI نیاز است خاصیت async آن‌را به نحو ذیل مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: { // async configuration
                    saveUrl: "@Url.Action("Save", "Home")", // the url to save a file is '/save'
                    removeUrl: "@Url.Action("Remove", "Home")", // the url to remove a file is
'/remove'
                    autoUpload: false, // automatically upload files once selected
                    removeVerb: 'POST'
                },
                multiple: true,
                showFileList: true
            });
        });
    </script>
```

در اینجا دو آدرس ذخیره سازی فایل‌ها و همچنین حذف آن‌ها را مشاهده می‌کنید. امضای این دو اکشن متد در ASP.NET MVC به صورت ذیل هستند:

```
[HttpPost]
        public ActionResult Save(IEnumerable<HttpPostedFileBase> files)
        {
```

```
            if (files != null)
            {
                // ...
                // Process the files and save them
                // ...
            }

            // Return an empty string to signify success
            return Content("");
        }

        [HttpPost]
        public ContentResult Remove(string[] fileNames)
        {
            if (fileNames != null)
            {
                foreach (var fullName in fileNames)
                {
                    // ...
                    // delete the files
                    // ...
                }
            }

            // Return an empty string to signify success
            return Content("");
        }
```

در هر دو حالت، لیستی از فایل‌ها توسط kendoUpload به سمت سرور ارسال می‌شوند. در حالت Save، محتوای این فایل‌ها جهت ذخیره سازی بر روی سرور در دسترس خواهد بود. در حالت Remove، صرفا نام این فایل‌ها برای حذف از سرور، توسط کاربر ارسال می‌شوند.

دو دکمه‌ی حذف با کارکردهای متفاوت در ویجت kendoUpload وجود دارند. در ابتدای کار، پیش از ارسال فایل‌ها به سرور:

کلیک بر روی دکمه‌ی حذف در این حالت، صرفا فایلی را از لیست سمت کاربر حذف می‌کند.

پس از ارسال فایل‌ها به سرور:

اما پس از پایان عملیات ارسال، اگر کاربر بر روی دکمه‌ی حذف کلیک کند، توسط آدرس مشخص شده توسط خاصیت removeUrl، نام فایل‌های مورد نظر، برای حذف از سرور ارسال می‌شوند.

چند نکته‌ی تکمیلی

- تنظیم خاصیت autoUpload به true سبب می‌شود تا پس از انتخاب فایل‌ها توسط کاربر، بلافاصله و به صورت خودکار عملیات ارسال فایل‌ها به سرور آغاز شوند. اگر به false تنظیم شود، دکمه‌ی ارسال فایل‌ها در پایین لیست نمایش داده خواهد شد.
- شاید علاقمند باشید تا removeVerb را به DELETE تغییر دهید؛ بجای POST. به همین منظور می‌توان خاصیت removeVerb در اینجا مقدار دهی کرد.
- با تنظیم خاصیت multiple به true، کاربر قادر خواهد شد تا توسط صفحه‌ی دیالوگ انتخاب فایل‌ها، قابلیت انتخاب بیش از یک فایل را داشته باشد.
- showFileList نمایش لیست فایل‌ها را سبب می‌شود.

تعیین پسوند فایل‌های صفحه‌ی انتخاب فایل‌ها

هنگامیکه کاربر بر روی دکمه‌ی انتخاب فایل‌ها برای ارسال کلیک می‌کند، در صفحه‌ی دیالوگ باز شده می‌توان پسوندهای پیش فرض مجاز را نیز تعیین کرد.

برای این منظور تنها کافی است ویژگی accept را به input از نوع فایل اضافه کرد. چند مثال در این مورد:

```
<!-- Content Type with wildcard.  All Images -->
<input type="file" id="demoFile" title="Select file" accept="image/*" />

<!-- List of file extensions -->
<input type="file" id="demoFile" title="Select file" accept=".jpg,.png,.gif" />

<!-- Any combination of the above -->
<input type="file" id="demoFile" title="Select file" accept="audio/*,application/pdf,.png" />
```

نمایش متن کشیدن و رها کردن، بومی سازی برچسب‌ها و نمایش راست به چپ

همانطور که در تصاویر فوق ملاحظه می‌کنید، نمایش این ویجت راست به چپ و پیام‌های آن نیز ترجمه شده‌اند.

برای راست به چپ سازی آن مانند قبل تنها کافی است input مرتبط، در یک div با کلاس k-rtl محصور شود:

```
<div class="k-rtl k-header">
            <input name="files" id="files" type="file"  />
        </div>
```

برای بومی سازی پیام‌های آن می‌توان مانند مثال ذیل، خاصیت localization را مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                 //...
                },
                //...
                localization: {
                    select: 'انتخاب فایل‌ها برای ارسال',
                    remove: 'حذف فایل',
                    retry: 'سعی مجدد',
                    headerStatusUploading: 'در حال ارسال فایل‌ها',
                    headerStatusUploaded: 'پایان ارسال',
                    cancel: "لغو",
                    uploadSelectedFiles: "ارسال فایل‌ها",
                    dropFilesHere: "فایل‌ها را برای ارسال، کشیده و در اینجا رها کنید",
                    statusUploading: "در حال ارسال",
                    statusUploaded: "ارسال شد",
                    statusWarning: "اخطار",
                    statusFailed: "خطا در ارسال"
                }
            });
        });
    </script>
```

به علاوه متن dropFilesHere به صورت پیش فرض نامرئی است. برای نمایش آن نیاز است CSS موجود را بازنویسی کرد تا em مرتبط مرئی شود:

```
<style type="text/css">
div.k-dropzone {
    border: 1px solid #c5c5c5; /* For Default; Different for each theme */
}

div.k-dropzone em {
    visibility: visible;
}
</style>
```

تغییر قالب نمایش لیست فایل‌ها

لیست فایل‌ها در ویجت kendoUpload دارای یک قالب پیش فرض است که امکان بازنویسی کامل آن وجود دارد. ابتدا نیاز است یک kendo-template را بر این منظور تدارک دید:

```
<script id="fileListTemplate" type="text/x-kendo-template">
        <li class='k-file'>
            <span class='k-progress'></span>
            <span class='k-icon'></span>
            <span class='k-filename' title='#=name#'>#=name# (#=size# bytes)</span>
            <strong class='k-upload-status'></strong>
        </li>
    </script>
```

و سپس برای استفاده از آن خواهیم داشت:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
```

```
                name: "files",
                async: {
                // ...
                },
                // ...
                template: kendo.template($('#fileListTemplate').html()),
                // ...
            });
        });
    </script>
```

در این قالب، مقدار size هر فایل نیز در کنار نام آن نمایش داده می‌شود.

رخدادهای ارسال فایل‌ها

افزونه‌ی kendoUpload در حالت ارسال Ajax ایی فایل‌ها، رخدادهایی مانند شروع به ارسال، موفقیت، پایان، درصد ارسال فایل‌ها و امثال آن‌را نیز به همراه دارد که لیست کامل آن‌ها را در ذیل مشاهده می‌کنید:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: { // async configuration
                //...
                },
                //...
                localization: {
                },
                cancel: function () {
                    console.log('Cancel Event.');
                },
                complete: function () {
                    console.log('Complete Event.');
                },
                error: function () {
                    console.log('Error uploading file.');
                },
                progress: function (e) {
                    console.log('Uploading file ' + e.percentComplete);
                },
                remove: function () {
                    console.log('File removed.');
                },
                select: function () {
                    console.log('File selected.');
                },
                success: function () {
                    console.log('Upload successful.');
                },
                upload: function (e) {
                    console.log('Upload started.');
                }
            });
        });
    </script>
```

ارسال متادیتای اضافی به همراه فایل‌های ارسالی

فرض کنید می‌خواهید به همراه فایل‌های ارسالی به سرور، پارامتر codeId را نیز ارسال کنید. برای این منظور باید خاصیت e.data رویداد upload را به نحو ذیل مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                //...
```

```
                },
                //...
                localization: {
                },
                upload: function (e) {
                    console.log('Upload started.');
                    // Sending metadata to the save action
                    e.data = {
                        codeId: "1234567",
                        param2: 12
                        //, ...
                    };
                }
            });
        });
    </script>
```

سپس در سمت سرور، امضای متد Save بر اساس پارامترهای تعریف شده در سمت کاربر، به نحو ذیل تغییر می‌کند:

```
[HttpPost]
  public ActionResult Save(IEnumerable<HttpPostedFileBase> files, string codeId)
```

فعال سازی ارسال batch

اگر در متد Save سمت سرور یک break point قرار دهید، مشاهده خواهید کرد که به ازای هر فایل موجود در لیست در سمت کاربر، یکبار متد Save فراخوانی می‌شود و عملا متد Save، لیستی از فایل‌ها را در طی یک فراخوانی دریافت نمی‌کند. برای فعال سازی این قابلیت تنها کافی است خاصیت batch را به true تنظیم کنیم:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                    // ....
                    batch: true
                },
            });
        });
    </script>
```

به این ترتیب دیگر لیست فایل‌ها به صورت مجزا در سمت کاربر نمایش داده نمی‌شود و تمام آن‌ها با یک کاما از هم جدا خواهند شد. همچنین دیگر شاهد نمایش درصد پیشرفت تکی فایل‌ها نیز نخواهیم بود و اینبار درصد پیشرفت کل batch گزارش می‌شود.
در یک چنین حالتی باید دقت داشت که تنظیم maxRequestLength در web.config برنامه الزامی است؛ زیرا به صورت پیش فرض محدودیت 4 مگابایتی ارسال فایل‌ها توسط ASP.NET اعمال می‌شود:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
    <!-- The request length is in kilobytes, execution timeout is in seconds  -->
    <httpRuntime maxRequestLength="10240" executionTimeout="120" />
  </system.web>

  <system.webServer>
    <security>
      <requestFiltering>
        <!-- The content length is in bytes  -->
        <requestLimits maxAllowedContentLength="10485760"/>
      </requestFiltering>
    </security>
  </system.webServer>
</configuration>
```

عنوان: شروع به کار با Ember.js
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۴ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, SPA, EmberJS

Ember.js کتابخانه‌ای است جهت ساده سازی تولید برنامه‌های تک صفحه‌ای وب. برنامه‌هایی که شبیه به برنامه‌های دسکتاپ در مرورگر کاربر عمل می‌کنند. دو برنامه نویس اصلی آن Yehuda Katz که عضو اصلی تیم‌های jQuery و Ruby on Rails است و Tom Dale که ابتدا SproutCore را به وجود آورد و بعدها به Ember.js تغییر نام یافت، هستند.

منابع اصلی Ember.js

پیش از شروع به بحث نیاز است با تعدادی از سایت‌های اصلی مرتبط با Ember.js آشنا شد:
سایت اصلی: http://emberjs.com
مخزن کدهای آن: https://github.com/emberjs
انجمن اختصاصی پرسش و پاسخ: http://discuss.emberjs.com
موتور قالب‌های آن: http://handlebarsjs.com
لیست منابع مطالعاتی مرتبط مانند ویدیوهای آموزشی و لیست مقالات موجود: http://emberwatch.com
و بسته‌ی نیوگت آن: https://www.nuget.org/packages/EmberJS

مفاهیم پایه‌ای Ember.js

شیء Application

```
App = Ember.Application.create();
```

یک برنامه‌ی Ember.js با تعریف وهله‌ای از شیء Application آن آغاز می‌شود. با اینکار به صورت خودکار رویدادگردان‌هایی به صفحه اضافه می‌شوند. کامپوننت‌های پیش فرض آن ایجاد شده و همچنین قالب اصلی برنامه رندر می‌شود.

مسیر یابی

با مرور قسمت‌های مختلف برنامه توسط کاربر، نیاز است حالات برنامه را مدیریت کرد؛ اینجا است که کار قسمت مسیریابی شروع می‌شود. مسیریابی، منابع مورد نیاز جهت آدرس‌های مشخصی را تامین می‌کند.

```
App.Router.map(function() {
    this.resource('accounts'); // takes us to /accounts
    this.resource('gallery'); // takes us to /gallery
});
```

در اینجا نحوه‌ی تعریف آغازین مسیریابی Ember.js را مشاهده می‌کنید که توسط متد resource آن مسیرهای قابل ارائه توسط برنامه مشخص می‌شوند.
به این ترتیب مسیرهای accounts/ و gallery/ قابل پردازش خواهند شد.

این مسیرها، تو در تو نیز می‌توانند باشند. برای مثال:

```
App.Router.map(function() {
    this.resource('news', function() {
        this.resource('images', function () { // takes us to /news/images
            this.route('add');// takes us to /news/images/add
        });
    });
});
```

به این ترتیب نحوه‌ی تعریف مسیریابی آدرس news/images/add را مشاهده می‌کنید. همچنین در این مثال از دو متد resource و route استفاده شده‌است. از متد resource برای حالت تعریف اسامی استفاده کنید و از متد route برای تعریف افعال و تغییر دهنده‌ها. برای نمونه در اینجا فعل افزودن تصاویر با متد route مشخص شده‌است.

مدل‌ها

مدل‌ها همان اشیایی هستند که برنامه مورد استفاده قرار می‌دهد و می‌توانند یک آرایه‌ی ساده و یا اشیاء JSON دریافتی از وب سرور باشند.

حداقل به دو روش می‌توان مدل‌ها را تعریف کرد:

الف) با استفاده از افزونه‌ی Ember Data

ب) با کمک شیء Ember.Object

```
App.SiteLink = Ember.Object.extend({});
App.SiteLink.reopenClass({
    findAll: function() {
        var links = [];
        //… $.getJSON …

        return links;
    }
});
```

ابتدا یک زیرکلاس از Ember.Object به کمک متد extend ایجاد خواهد شد. سپس از متد توکار reopenClass برای توسعه‌ی API کمک خواهیم گرفت.

در ادامه متد دلخواهی را ایجاد کرده و برای مثال آرایه‌ای از اشیاء دلخواه جاوا اسکریپتی را بازگشت خواهیم داد.

پس از تعریف مدل، نیاز است آن‌را به سیستم مسیریابی معرفی کرد:

```
App.GalleryRoute = Ember.Route.extend({
    model: function() {
        return App.SiteLink.findAll();
    }
});
```

به این ترتیب زمانیکه کاربر به آدرس gallery/ مراجعه می‌کند، دسترسی به model وجود خواهد داشت. در اینجا model یک واژه‌ی کلیدی است.

کنترلرها

کنترلرها جهت ارائه‌ی اطلاعات مدل‌ها به View و قالب برنامه تعریف می‌شوند. در اینجا همیشه باید بخاطر داشت که model تامین کننده‌ی اطلاعات است. کنترلر جهت در معرض دید قرار دادن این اطلاعات، به View برنامه کاربرد دارد و مدل‌ها هیچ اطلاعی از وجود کنترلرها ندارند.

کنترلرها علاوه بر اطلاعات model، می‌توانند حاوی یک سری خواص و اشیاء صرفا نمایشی که قرار نیست در بانک اطلاعاتی ذخیره شوند نیز باشند.

در Ember.js قالب‌ها (templates) اطلاعات خود را از کنترلر دریافت می‌کنند. کنترلرها اطلاعات مدل را به همراه سایر خواص نمایشی مورد نیاز در اختیار View و قالب‌های برنامه قرار می‌دهند.

برای تعریف یک کنترلر می‌توان درون شیء مسیریابی، با تعریف متد setupController شروع کرد:

```
App.GalleryRoute = Ember.Route.extend({
    setupController: function(controller) {
        controller.set('content', ['red', 'yellow', 'blue']);
    }
});
```

در این مثال یک خاصیت دلخواه به نام content تعریف و سپس آرایه‌ای به آن انتساب داده شده‌است.

روش دوم تعریف کنترلرها با ایجاد یک زیر کلاس از شیء Ember.Controller انجام می‌شود:

```
App.GalleryController = Ember.Controller.extend({
    search: '',
    content: ['red', 'yellow', 'blue'],
    query: function() {
        var data = this.get('search');
        this.transitionToRoute('search', { query: data });
```

```
    }
});
```

قالب‌ها یا templates

قالب‌ها قسمت‌های اصلی رابط کاربری را تشکیل خواهند داد. در اینجا از کتابخانه‌ای به نام handlebars برای تهیه قالب‌های سمت کاربر کمک گرفته می‌شود.

```
<script type="text/x-handlebars" data-template-name="sayhello">
    Hello,
    <strong>{{firstName}} {{lastName}}</strong>!
</script>
```

این قالب‌ها توسط تگ اسکریپت تعریف شده و نوع آن‌ها text/x-handlebars مشخص می‌شود. به این ترتیب Ember.js، این قسمت از صفحه را یافته و عبارات داخل {{}} را با مقادیر دریافتی از کنترلر جایگزین می‌کند.

```
<script type="text/x-handlebars" data-template-name="sayhello">
    Hello,
    <strong>{{firstName}} {{lastName}}</strong>!

    {{#if person}}
    Welcome back,
    <strong>{{person.firstName}} {{person.lastName}}</strong>!
    {{/if}}

    <ul>
        {{#each friend in friends}}
        <li>
            {{friend.name}}
        </li>
        {{/each}}
    </ul>

    <img {{bindAttr src="link.url" }} />
    {{#linkTo ''about}}About{{/linkTo}}
</script>
```

در این مثال نحوه‌ی تعریف عبارات شرطی و یا یک حلقه را نیز مشاهده می‌کنید. همچنین امکان اتصال به ویژگی‌هایی مانند src یک تصویر و یا ایجاد لینک‌ها را نیز دارا است.

بهترین مرجع آشنایی با ریز جزئیات کتابخانه‌ی handlebars، مراجعه به سایت اصلی آن است.

قواعد پیش فرض نامگذاری در Ember.js

اگر به مثال‌های فوق دقت کرده باشید، خواصی مانند GalleryController و یا GalleryRoute به شیء App اضافه شده‌اند. این نوع نامگذاری‌ها در ember.js بر اساس روش convention over configuration کار می‌کنند. برای نمونه اگر مسیریابی خاصی را به نحو ذیل تعریف کردید:

```
this.resource('employees');
```

شیء مسیریابی آن App.EmployeesRoute

کنترلر آن App.EmployeesController

مدل آن App.Employee

View آن App.EmployeesView

و قالب آن employees

بهتر است تعریف شوند. به عبارتی اگر اینگونه تعریف شوند، به صورت خودکار توسط Ember.js یافت شده و هر کدام با مسئولیت‌های خاص مرتبط با آن‌ها پردازش می‌شوند و همچنین ارتباطات بین آن‌ها به صورت خودکار برقرار خواهد شد. به این ترتیب برنامه نظم بهتری خواهد یافت. با یک نگاه می‌توان قسمت‌های مختلف را تشخیص داد و همچنین کدنویسی پردازش و

اتصال قسمت‌های مختلف برنامه نیز به شدت کاهش می‌یابد.

تهیه‌ی اولین برنامه‌ی Ember.js

تا اینجا نگاهی مقدماتی داشتیم به اجزای تشکیل دهنده‌ی هسته‌ی Ember.js. در ادامه مثال ساده‌ای را جهت نمایش ساختار ابتدایی یک برنامه‌ی Ember.js، بررسی خواهیم کرد.

بسته‌ی Ember.js را همانطور که در قسمت منابع اصلی آن در ابتدای بحث عنوان شد، می‌توانید از سایت و یا مخزن کد آن دریافت کنید و یا اگر از VS.NET استفاده می‌کنید، تنها کافی است دستور ذیل را صادر نمائید:

```
PM> Install-Package EmberJS
```

پس از اضافه شدن فایل‌های js آن به پوشه‌ی Scripts برنامه، در همان پوشه، فایل جدید Scripts\app.js را نیز اضافه کنید. از آن برای افزودن تعاریف کدهای Ember.js استفاده خواهیم کرد.

در این حالت ترتیب تعریف اسکریپت‌های مورد نیاز صفحه به صورت ذیل خواهند بود:

```
<script src="Scripts/jquery-2.1.1.js" type="text/javascript"></script>
<script src="Scripts/handlebars.js" type="text/javascript"></script>
<script src="Scripts/ember.js" type="text/javascript"></script>
<script src="Scripts/app.js" type="text/javascript"></script>
```

کدهای ابتدایی فایل app.js جهت وهله سازی شیء Application و سپس تعریف مسیریابی صفحه‌ی index بر اساس روش convention over configuration به همراه تعریف یک کنترلر و افزودن متغیری به نام content به آن که با یک آرایه مقدار دهی شده‌است:

```
App = Ember.Application.create();
App.IndexRoute = Ember.Route.extend({
    setupController:function(controller) {
        controller.set('content', ['red', 'yellow', 'blue']);
    }
});
```

باید دقت داشت که تعریف مقدماتی Ember.Application.create به همراه یک سری تنظیمات پیش فرض نیز هست. برای مثال مسیریابی index به صورت خودکار به نحو ذیل توسط آن تعریف خواهد شد و نیازی به تعریف مجدد آن نیست:

```
App.Router.map(function() {
    this.resource('application');
    this.resource('index');
});
```

سپس برای اتصال این کنترلر به یک template خواهیم داشت:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/jquery-2.1.1.js" type="text/javascript"></script>
    <script src="Scripts/handlebars.js" type="text/javascript"></script>
    <script src="Scripts/ember.js" type="text/javascript"></script>
    <script src="Scripts/app.js" type="text/javascript"></script>
</head>
<body>
    <script type="text/x-handlebars" data-template-name="index">
        Hello,
        <strong>Welcome to Ember.js</strong>!
        <ul>
            {{#each item in content}}
            <li>
                {{item}}
            </li>
            {{/each}}
        </ul>
    </script>
```

```
</body>
</html>
```

توسط اسکریپتی از نوع text/x-handlebars، اطلاعات آرایه content دریافت و در طی یک حلقه در صفحه نمایش داده خواهد شد.

مقدار data-template-name در اینجا مهم است. اگر آن‌را به هر نام دیگری بجز index تنظیم کنید، منبع دریافت اطلاعات آن مشخص نخواهد بود. نام index در اینجا به معنای اتصال این قالب به اطلاعات ارائه شده توسط کنترلر index است.

تا همینجا اگر برنامه را اجرا کنید، به خوبی کار خواهد کرد. نکته‌ی دیگری که در مورد قالب‌های Ember.js قابل توجه هستند، قالب پیش فرض application است. با تعریف Ember.Application.create یک چنین قالبی نیز به ابتدای هر صفحه به صورت خودکار اضافه خواهد شد:

```
<body>
    <script type="text/x-handlebars" data-template-name="application">
        <h1>Header</h1>
        {{outlet}}
    </script>
```

outlet واژه‌ای است کلیدی که سبب رندر سایر قالب‌های تعریف شده در صفحه می‌گردد. مقدار data-template-name آن نیز به application تنظیم شده‌است (اگر این مقدار ذکر نگردد نیز به صورت خودکار از application استفاده می‌شود). برای مثال اگر بخواهید به تمام قالب‌های رندر شده در صفحات مختلف، مقدار ثابتی را اضافه کنید (مانند هدر یا منو)، می‌توان قالب application را به صورت دستی به نحو فوق اضافه کرد و آن‌را سفارشی سازی نمود.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS01.zip

نظرات خوانندگان

نویسنده: نصیری
تاریخ: ۱۳۹۳/۰۹/۱۸ ۱۷:۴۹

با سلام؛ با توجه به قدرت خیلی زیاد AngularJs و نیز اینکه پشتیبانی اون داره از طرف تیم قدرتمندی در google اداره میشه و دلیل بعدی اینکه تعداد خیلی زیادی از برنامه نویسهای دنیا به سمت این فریمورک رفتن بهتر نیست یادگرفتن و دنبال AngularJs رفتن را به EmberJs ترجیح داد
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۸ ۱۸:۱۸

- صرف نوشتن مطلبی در مورد موضوعی خاص، به معنای «بهترین بودن آن» و «برترین حالت ممکن» نیست. امروز راجع به ember.js مطلب نوشتم. شاید هفته‌های بعد در مورد meteor.js مطلب نوشتم.
- مقایسه‌ای در اینجا « AngularJS vs Ember »
- مقایسه‌ای کامل‌تر در اینجا « AngularJs vs EmberJs »
- این نظرسنجی را هم دنبال کنید: « آیا به یادگیری یا ادامه‌ی استفاده از AngularJS خواهید پرداخت؟ »

عنوان: **افزودن یک صفحه‌ی جدید و دریافت و نمایش اطلاعات از سرور به کمک Ember.js**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۵ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, SPA, EmberJS

در قسمت قبل با مقدمات برپایی یک برنامه‌ی تک صفحه‌ای وب مبتنی بر Ember.js آشنا شدیم. مثال انتهای بحث آن نیز یک لیست ساده را نمایش می‌دهد. در ادامه همین برنامه را جهت نمایش لیستی از اشیاء JSON دریافتی از سرور تغییر خواهیم داد. همچنین یک صفحه‌ی about را نیز به آن اضافه خواهیم کرد.

پیشنیازهای سمت سرور

- ابتدا یک پروژه‌ی خالی ASP.NET را ایجاد کنید. نوع آن مهم نیست که Web Forms باشد یا MVC.
- سپس قصد داریم مدل کاربران سیستم را توسط یک ASP.NET Web API Controller در اختیار Ember.js قرار دهیم. مباحث پایه‌ای Web API نیز در وب فرم‌ها و MVC یکی است.

مدل سمت سرور برنامه چنین شکلی را دارد:

```
namespace EmberJS02.Models
{
    public class User
    {
        public int Id { set; get; }
        public string UserName { set; get; }
        public string Email { set; get; }
    }
}
```

کنترلر Web API ایی که این اطلاعات را در ختیار کلاینت‌ها قرار می‌دهد، به نحو ذیل تعریف می‌شود:

```
using System.Collections.Generic;
using System.Web.Http;
using EmberJS02.Models;

namespace EmberJS02.Controllers
{
    public class UsersController : ApiController
    {
        // GET api/<controller>
        public IEnumerable<User> Get()
        {
            return UsersDataSource.UsersList;
        }
    }
}
```

در اینجا UsersDataSource.UsersList صرفا یک لیست جنریک ساده از کلاس User است و کدهای کامل آن‌را می‌توانید از فایل پیوست انتهای بحث دریافت کنید.

همچنین فرض بر این است که مسیریابی سمت سرور ذیل را نیز به فایل global.asax.cs، جهت فعال سازی دسترسی به متدهای کنترلر UsersController تعریف کرده‌اید:

```
using System;
using System.Web.Http;
using System.Web.Routing;

namespace EmberJS02
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            RouteTable.Routes.MapHttpRoute(
               name: "DefaultApi",
```

```
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
                );
            }
        }
}
```

پیشنیازهای سمت کاربر

پیشنیازهای سمت کاربر این قسمت با قسمت « تهیه‌ی اولین برنامه‌ی Ember.js » دقیقا یکی است.
ابتدا فایل‌های مورد نیاز Ember.js به برنامه اضافه شده‌اند:

```
PM> Install-Package EmberJS
```

سپس یک فایل app.js با محتوای ذیل به پوشه‌ی Scripts اضافه شده‌است:

```
App = Ember.Application.create();
App.IndexRoute = Ember.Route.extend({
    setupController:function(controller) {
        controller.set('content', ['red', 'yellow', 'blue']);
    }
});
```

و در آخر یک فایل index.html با محتوای ذیل کار برپایی اولیه‌ی یک برنامه‌ی مبتنی بر Ember.js را انجام می‌دهد:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/jquery-2.1.1.js" type="text/javascript"></script>
    <script src="Scripts/handlebars.js" type="text/javascript"></script>
    <script src="Scripts/ember.js" type="text/javascript"></script>
    <script src="Scripts/app.js" type="text/javascript"></script>
</head>
<body>
    <script type="text/x-handlebars" data-template-name="application">
        <h1>Header</h1>
        {{outlet}}
    </script>
    <script type="text/x-handlebars" data-template-name="index">
        Hello,
        <strong>Welcome to Ember.js</strong>!
        <ul>
            {{#each item in content}}
            <li>
                {{item}}
            </li>
            {{/each}}
        </ul>
    </script>
</body>
</html>
```

تا اینجا را در قسمت قبل مطالعه کرده بودید.
در ادامه قصد داریم به هدر صفحه، دو لینک Home و About را اضافه کنیم؛ به نحوی که لینک Home به مسیریابی index و لینک About به مسیریابی about که صفحه‌ی جدید «درباره‌ی برنامه» را نمایش می‌دهد، اشاره کنند.

تعریف صفحه‌ی جدید About

برنامه‌های Ember.js، برنامه‌های تک صفحه‌ای وب هستند و صفحات جدید در آن‌ها به صورت یک template جدید تعریف می‌شوند که نهایتا متناظر با یک مسیریابی مشخص خواهند بود.
به همین جهت ابتدا در فایل app.js مسیریابی about را اضافه خواهیم کرد:

```
App.Router.map(function() {
    this.resource('about');
});
```

به این ترتیب با فراخوانی آدرس /about در مرورگر توسط کاربر، منابع مرتبط با این آدرس و قالب مخصوص آن، توسط Ember.js پردازش خواهند شد.
بنابراین به صفحه‌ی index.html برنامه مراجعه کرده و صفحه‌ی about را توسط یک قالب جدید تعریف می‌کنیم:

```
<script type="text/x-handlebars" data-template-name="about">
    <h2>Our about page</h2>
</script>
```

تنها نکته‌ی مهم در اینجا مقدار data-template-name است که سبب خواهد شد تا به مسیریابی about، به صورت خودکار متصل و مرتبط شود.

در این حالت اگر برنامه را در حالت معمولی اجرا کنید، خروجی خاصی را مشاهده نخواهید کرد. بنابراین نیاز است تا لینکی را جهت اشاره به این مسیر جدید به صفحه اضافه کنیم:

```
<script type="text/x-handlebars" data-template-name="application">
    <h1>Ember Demo App</h1>
    <ul class="nav">
        <li>{{#linkTo 'index'}}Home{{/linkTo}}</li>
        <li>{{#linkTo 'about'}}About{{/linkTo}}</li>
    </ul>
    {{outlet}}
</script>
```

اگر از قسمت قبل به خاطر داشته باشید، عنوان شد که قالب ویژه‌ی application به صورت خودکار با وهله سازی Ember.Application.create به صفحه اضافه می‌شود. اگر نیاز به سفارشی سازی آن وجود داشت، خصوصا جهت تعریف عناصری که باید در تمام صفحات حضور داشته باشند (مانند منوها)، می‌توان آن‌را به نحو فوق سفارشی سازی کرد.
در اینجا با استفاده از امکان یا directive ویژه‌ای به نام linkTo، لینک‌هایی به مسیریابی‌های index و about اضافه شده‌اند. به این ترتیب اگر کاربری برای مثال بر روی لینک About کلیک کند، کتابخانه‌ی Ember.js او را به صورت خودکار به مسیریابی about و سپس نمایش قالب مرتبط با آن (قالب about ایی که پیشتر تعریف کردیم) هدایت خواهد کرد؛ مانند تصویر ذیل:

همانطور که در آدرس صفحه نیز مشخص است، هرچند صفحه‌ی about نمایش داده شده‌است، اما هنوز نیز در همان صفحه‌ی اصلی برنامه قرار داریم. به علاوه در این قسمت جدید، همچنان منوی بالای صفحه نمایان است؛ از این جهت که تعاریف آن به قالب application اضافه شده‌اند.

دریافت و نمایش اطلاعات از سرور

اکنون که با نحوه‌ی تعریف یک صفحه‌ی جدید و برپایی سیم کشی‌های مرتبط با آن آشنا شدیم، می‌خواهیم صفحه‌ی دیگری را به نام Users به برنامه اضافه کنیم و در آن لیست کاربران ارائه شده توسط کنترلر Web API سمت سرور ابتدای بحث را نمایش دهیم.
بنابراین ابتدا مسیریابی جدید users را به صفحه اضافه می‌کنیم تا لیست کاربران، در آدرس users/ قابل دسترسی شود:

```
App.Router.map(function() {
    this.resource('about');
    this.resource('users');
});
```

سپس نیاز است مدلی را توسط فراخوانی Ember.Object.extend ایجاد کرده و به کمک متد reopenClass آن‌را توسعه دهیم:

```
App.UsersLink = Ember.Object.extend({});
App.UsersLink.reopenClass({
    findAll: function () {
        var users = [];
        $.getJSON('/api/users').then(function(response) {
            response.forEach(function(item) {
                users.pushObject(App.UsersLink.create(item));
            });
        });
        return users;
    }
});
```

در اینجا متد دلخواهی را به نام findAll اضافه کرده‌ایم که توسط متد getJSON جی‌کوئری، به مسیر /api/users سمت سرور متصل شده و لیست کاربران را از سرور به صورت JSON دریافت می‌کند. در اینجا خروجی دریافتی از سرور به کمک متد pushObject به آرایه کاربران اضافه خواهد شد. همچنین نحوه‌ی فراخوانی متد create مدل UsersLink را نیز در اینجا مشاهده می‌کنید (App.UsersLink.create).

پس از اینکه نحوه‌ی دریافت اطلاعات از سرور مشخص شد، باید اطلاعات این مدل را در اختیار مسیریابی Users قرار داد:

```
App.UsersRoute = Ember.Route.extend({
    model: function() {
        return App.UsersLink.findAll();
    }
});

App.UsersController = Ember.ObjectController.extend({
    customHeader : 'Our Users List'
});
```

به این ترتیب زمانیکه کاربر به مسیر users/ مراجعه می‌کند، سیستم مسیریابی می‌داند که اطلاعات مدل خود را باید از کجا تهیه نماید.
همچنین در کنترلری که تعریف شده، صرفا یک خاصیت سفارشی و دلخواه جدید، به نام customHeader برای نمایش در ابتدای صفحه تعریف و مقدار دهی گردیده‌است.
اکنون قالبی که قرار است اطلاعات مدل را نمایش دهد، چنین شکلی را خواهد داشت:

```
<script type="text/x-handlebars" data-template-name="users">
    <h2>{{customHeader}}</h2>
    <ul>
        {{#each item in model}}
        <li>
            {{item.Id}}-{{item.UserName}} ({{item.Email}})
```

```
        </li>
        {{/each}}
    </ul>
</script>
```

با تنظیم data-template-name به users سبب خواهیم شد تا این قالب اطلاعات خودش را از مسیریابی users دریافت کند. سپس یک حلقه نوشته‌ایم تا کلیه عناصر موجود در مدل را خوانده و در صفحه نمایش دهد. همچنین در عنوان قالب نیز از خاصیت سفارشی customHeader استفاده شده‌است:

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS02.zip

عنوان: ساخت یک بلاگ ساده با Ember.js، قسمت اول
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۶ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, SPA, EmberJS

پس از آشنایی مقدماتی با اجزای مهم تشکیل دهنده‌ی Ember.js (^ و ^)، بهتر است این دانسته‌ها را جهت تکمیل یک پروژه‌ی ساده‌ی تک صفحه‌ای بلاگ، بکار بگیریم.

در این بلاگ می‌توان:

- یک مطلب جدید را ارسال کرد.
- مطالب قابل ویرایش و یا حذف هستند.
- مطالب بلاگ قسمت ارسال نظرات دارند.
- امکان گزارشگیری از آخرین نظرات ارسالی وجود دارد.
- سایت صفحات درباره و تماس با ما را نیز دارا است.

ساختار پوشه‌های برنامه

در تصویر ذیل، ساختار پوشه‌های برنامه بلاگ را ملاحظه می‌کنید. چون قسمت سمت کلاینت این برنامه کاملا جاوا اسکریپتی است، پوشه‌های Routes، Models، Libs، Controllers، App و Templates آن در پوشه‌ی Scripts تعریف شده‌اند و به این ترتیب می‌توان تفکیک بهتری را بین اجزای تشکیل دهنده‌ی یک برنامه‌ی تک صفحه‌ای وب Emeber.js پدید آورد.

Solution 'EmberJS03' (1 project)
.nuget
EmberJS03
Properties
References
App_Readme
Content
bootstrap.css
styles.css
Controllers
Models
Scripts
App
Controllers
Libs
ember-data.min.js
ember-handlebars-loader-0.0.1.js
ember.min.js
handlebars.min.js
jquery-2.1.1.min.js
jquery-2.1.1.min.map
localstorage_adapter.js
Models
Routes
Templates
Global.asax
Index.html
packages.config
Web.config

فایل CSS بوت استرپ نیز به پوشه‌ی Content اضافه شده‌است.

دریافت پیشنیازهای سمت کاربر برنامه

در ساختار پوشه‌های فوق، از پوشه‌ی Libs برای قرار دادن کتابخانه‌های پایه برنامه مانند jQuery و Ember.js استفاده خواهیم کرد. به این ترتیب:
- نیاز به آخرین نگارش‌های Ember.js و همچنین افزونه‌ی Ember-Data آن برای کار ساده‌تر با داده‌ها و سرور وجود دارد. این فایل‌ها را از آدرس ذیل می‌توانید دریافت کنید (نسخه‌های نیوگت به دلیل قدیمی بودن و به روز نشدن مداوم آن‌ها توصیه نمی‌شوند):
http://emberjs.com/builds/#/beta
برای حالت آزمایش برنامه، استفاده از فایل‌های دیباگ آن توصیه می‌شوند (فایل‌هایی با نام اصلی و بدون پسوند prod یا min). زیرا این فایل‌ها خطاها و اطلاعات بسیار مفصلی را از اشکالات رخ داده، در کنسول وب مرورگرها، فایرباگ و یا Developer tools آن‌ها نمایش می‌دهند. نسخه‌ی min برای حالت ارائه‌ی نهایی برنامه است. نسخه‌ی prod همان نسخه‌ی دیباگ است با حذف اطلاعات دیباگ (نسخه‌ی production فشرده نشده). نسخه‌ی فشرده شده‌ی prod آن، فایل min نهایی را تشکیل می‌دهد.
- دریافت جی‌کوئری
- آخرین نگارش handlebars.js را از سایت رسمی آن دریافت کنید: http://handlebarsjs.com
- Ember Handlebars Loader: https://github.com/michaelrkn/ember-handlebars-loader
- Ember Data Local Storage Adapter: https://github.com/kurko/ember-localstorage-adapter

ترتیب تعریف و قرارگیری این فایل‌ها را پس از دریافت، در فایل index.html قرار گرفته در ریشه‌ی سایت، در کدهای ذیل مشاهده می‌کنید:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Ember Blog</title>

  <link href="Content/bootstrap.css" rel="stylesheet" />
  <link href="Content/bootstrap-theme.css" rel="stylesheet" />
  <link href="Content/styles.css" rel="stylesheet" />

  <script src="Scripts/Libs/jquery-2.1.1.min.js" type="text/javascript"></script>
  <script src="Scripts/Libs/bootstrap.min.js" type="text/javascript"></script>
  <script src="Scripts/Libs/handlebars-v2.0.0.js" type="text/javascript"></script>
  <script src="Scripts/Libs/ember.js" type="text/javascript"></script>
  <script src="Scripts/Libs/ember-handlebars-loader-0.0.1.js" type="text/javascript"></script>
  <script src="Scripts/Libs/ember-data.js" type="text/javascript"></script>
  <script src="Scripts/Libs/localstorage_adapter.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

اصلاح فایل ember-handlebars-loader-0.0.1.js

اگر به فایل ember-handlebars-loader-0.0.1.js مراجعه کنید، مسیر فایل‌های قالب handlebars قسمت‌های مختلف برنامه را از پوشه‌ی templates واقع در ریشه‌ی سایت می‌خواند. با توجه به تصویر ساختار پوشه‌ی پروژه‌ی جاری، پوشه‌ی template به داخل پوشه‌ی Scripts منتقل شده‌است و نیاز به یک چنین اصلاحی دارد:

```
url: "Scripts/Templates/" + name + ".hbs",
```

کار اسکریپت ember-handlebars-loader-0.0.1.js بارگذاری خودکار فایل‌های hbs یا handlebars از پوشه‌ی قالب‌های سفارشی

برنامه است. در این حالت اگر برنامه را اجرا کنید، خطای 404 را دریافت خواهید کرد. از این جهت که mime-type پسوند hbs در IIS تعریف نشده‌است. اضافه کردن آن نیز ساده‌است. به فایل web.config برنامه مراجعه کرده و تغییر ذیل را اعمال کنید:

```
<system.webServer>
  <staticContent>
        <mimeMap fileExtension=".hbs" mimeType="text/x-handlebars-template" />
  </staticContent>
 </system.webServer>
```

مزیت استفاده از نسخه‌ی دیباگ ember.js در حین توسعه‌ی برنامه

نسخه‌ی دیباگ ember.js علاوه بر به همراه داشتن خطاهای بسیار جامع و توضیح علل مشکلات، مواردی را که در آینده منسوخ خواهند شد نیز توضیح می‌دهد:

برای مثال در اینجا عنوان شده‌است که دیگر از linkTo استفاده نکنید و آن‌را به link-to تغییر دهید.
همچنین اگر از مرورگر کروم استفاده می‌کنید، افزونه‌ی Ember Inspector را نیز می‌توانید نصب کنید تا اطلاعات بیشتری را از جزئیات مسیریابی‌های تعریف شده و قالب‌های Ember.js بتوان مشاهده کرد. این افزونه به صورت یک برگه‌ی جدید در Developer tools آن ظاهر می‌شود.

ایجاد شیء Application

همانطور که در قسمت‌های پیشین نیز عنوان شد (^ و ^)، یک برنامه‌ی Ember.js با تعریف وهله‌ای از شیء Application آن آغاز می‌شود. برای این منظور به پوشه‌ی App مراجعه کرده و فایل جدید Scripts\App\blogger.js را اضافه کنید؛ با این محتوا:

```
Blogger = Ember.Application.create();
```

مدخل این فایل را نیز پس از تعاریف وابستگی‌های اصلی برنامه، به فایل index.html اضافه خواهیم کرد:

```
<script src="Scripts/App/blogger.js" type="text/javascript"></script>
```

تا اینجا برپایی اولیه‌ی برنامه‌ی تک صفحه‌ای وب مبتنی بر Ember.js ما به پایان می‌رسد.

تعاریف مسیریابی و قالب‌ها

اکنون در ادامه قصد داریم لیستی از عناوین مطالب ارسالی را نمایش دهیم. در ابتدا این عناوین را از یک آرایه‌ی ثابت جاوا اسکریپتی دریافت خواهیم کرد و پس از آن از یک Web API کنترلر، جهت دریافت اطلاعات از سرور کمک خواهیم گرفت.

کار router در Ember.js، نگاشت آدرس درخواستی توسط کاربر، به یک route یا مسیریابی تعریف شده‌است. به این ترتیب مدل، کنترلر و قالب آن route به صورت خودکار بارگذاری و پردازش خواهند.
با مراجعه به ریشه‌ی سایت، فایل index.html بارگذاری می‌شود.

در اینجا تصویری از صفحه‌ی آغازین بلاگ را مشاهده می‌کنید. در آن صفحه‌ی posts همان ریشه‌ی سایت نیز می‌باشد. بنابراین نیاز است ابتدا مسیریابی آن‌را تعریف کرد. برای این منظور، فایل جدید Scripts\App\router.js را به پوشه‌ی App اضافه کنید؛ با این محتوا:

```
Blogger.Router.map(function () {
   this.resource('posts', { path: '/' });
});
```

علت تعریف قسمت path این است که ریشه‌ی سایت (/) نیز به مسیریابی posts ختم شود. در غیر اینصورت کاربر با مراجعه به ریشه‌ی سایت، یک صفحه‌ی خالی را مشاهده خواهد کرد؛ زیرا به صورت پیش فرض، آدرس قابل ترجمه‌ی یک صفحه، با آدرس و نام مسیریابی آن یکی است.

همچنین مدخل آن‌را نیز در فایل index.html تعریف نمائید:

```
<script src="Scripts/App/blogger.js" type="text/javascript"></script>
<script src="Scripts/App/router.js" type="text/javascript"></script>
```

در اینجا Blogger همان شیء Application برنامه است که پیشتر در فایل Scripts\App\blogger.js تعریف کردیم. سپس به کمک متد Blogger.Router.map، اولین مسیریابی برنامه را افزوده‌ایم.

افزودن مسیریابی و قالب posts

در ادامه فایل جدید Scripts\Templates\posts.hbs را اضافه کنید. به این ترتیب قالب خالی مطالب به پوشه‌ی templates اضافه می‌شود. محتوای این فایل را به نحو ذیل تنظیم کنید:

```
<div class="container">
  <h1>Emeber.js blog</h1>
  <ul>
   <li>Item 1</li>
   <li>Item 2</li>
   <li>Item 3</li>
  </ul>
</div>
```

در اینجا دیگر نیازی به ذکر تگ script از نوع text/x-handlebars نیست.

برای بارگذاری این قالب نیاز است آن‌را به template loader توضیح داده شده در ابتدای بحث، در فایل index.html اضافه کنیم:

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts'
 ]);
</script>
```

اکنون برنامه را اجرا کنید. به تصویر ذیل خواهید رسید که در آن افزونه‌ی Ember Inspector نیز نمایش داده شده‌است:

افزودن مسیریابی و قالب about

در ادامه قصد داریم صفحه‌ی about را اضافه کنیم. مجددا با افزودن مسیریابی آن به فایل Scripts\App\router.js شروع می‌کنیم:

```
Blogger.Router.map(function () {
  this.resource('posts', { path: '/' });
  this.resource('about');
});
```

سپس فایل قالب آن‌را در مسیر Scripts\Templates\about.hbs ایجاد خواهیم کرد؛ برای مثال با این محتوای فرضی:

```
<h1>About Ember Blog</h1>
<p>Bla bla bla!</p>
```

اکنون نام این فایل را به template loader فایل index.html اضافه می‌کنیم.

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about'
 ]);
</script>
```

اگر از قسمت قبل به خاطر داشته باشید، ساختار کلی قالب‌های ember.js یک چنین شکلی را دارد:

```
<script type="text/x-handlebars" data-template-name="about">

</script>
```

اسکریپت template loader، این تعاریف را به صورت خودکار اضافه می‌کند. مقدار data-template-name را نیز به نام فایل متناظر با آن تنظیم خواهد کرد و چون ember.js بر اساس ایده‌ی convention over configuration کار می‌کند، مسیریابی about با کنترلری به همین نام و قالبی هم نام پردازش خواهد شد. بنابراین نام فایل‌های قالب را باید بر اساس مسیریابی‌های متناظر با آن‌ها تعیین کرد.
برای آزمایش این مسیر و قالب جدید آن، آدرس http://localhost/#/about را بررسی کنید.

اضافه کردن منوی ثابت بالای سایت

روش اول این است که به ابتدای هر دو قالب about.hbs و posts.hbs، تعاریف منو را اضافه کنیم. مشکل این‌کار، تکرار کدها و پایین آمدن قابلیت نگهداری برنامه است. روش بهتر، افزودن کدهای مشترک بین صفحات، در قالب application برنامه است. نمونه‌ی آن‌را در مثال قسمت قبل مشاهده کرده‌اید. در اینجا چون قصد نداریم به صورت دستی قالب‌ها را به صفحه اضافه کنیم و همچنین برای ساده شدن نگهداری برنامه، قالب‌ها را در فایل‌های مجزایی قرار داده‌ایم، تنها کافی است فایل جدید Scripts\Templates\application.hbs را به پوشه‌ی قالب‌های برنامه اضافه کنیم؛ با این محتوا:

```
<div class='container'>
 <nav class='navbar navbar-default' role='navigation'>
  <ul class='nav navbar-nav'>
  <li>{{#link-to 'posts'}}Posts{{/link-to}}</li>
  <li>{{#link-to 'about'}}About{{/link-to}}</li>
  </ul>
 </nav>

  {{outlet}}
</div>
```

و سپس همانند قبل، نام فایل قالب اضافه شده را به template loader فایل index.html اضافه می‌کنیم:

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about', 'application'
 ]);
</script>
```

افزودن مسیریابی و قالب contact

برای افزودن صفحه‌ی تماس با مای سایت، ابتدا مسیریابی آن‌را در فایل Scripts\App\router.js تعریف می‌کنیم:

```
Blogger.Router.map(function () {
  this.resource('posts', { path: '/' });
  this.resource('about');
  this.resource('contact');
});
```

سپس قالب متناظر با آن‌را به نام Scripts\Templates\contact.hbs اضافه خواهیم کرد؛ فعلا با این محتوای اولیه:

```
<h1>Contact</h1>
<ul>
  <li>Phone: ...</li>
  <li>Email: ...</li>
</ul>
```

و بعد template loader فایل index.html را از وجود آن مطلع خواهیم کرد:

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about', 'application', 'contact' ]);
</script>
```

همچنین لینکی به مسیریابی آن‌را به فایل Scripts\Templates\application.hbs که منوی سایت در آن تعریف شده‌است، اضافه می‌کنیم:

```
<div class='container'>
 <nav class='navbar navbar-default' role='navigation'>
  <ul class='nav navbar-nav'>
  <li>{{#link-to 'posts'}}Posts{{/link-to}}</li>
  <li>{{#link-to 'about'}}About{{/link-to}}</li>
  <li>{{#link-to 'contact'}}Contact{{/link-to}}</li>
  </ul>
 </nav>

  {{outlet}}
</div>
```

تعریف مسیریابی تو در تو در صفحه‌ی contact

در حال حاضر صفحه‌ی Contact، ایمیل و شماره تماس را در همان بار اول نمایش می‌دهد. می‌خواهیم این دو را تبدیل به دو لینک جدید کنیم که با کلیک بر روی هر کدام، محتوای مرتبط، در قسمتی از همان صفحه بارگذاری شده و نمایش داده شود.
برای اینکار نیاز است مسیریابی را تو در تو تعریف کنیم:

```
Blogger.Router.map(function () {
  this.resource('posts', { path: '/' });
  this.resource('about');
  this.resource('contact', function () {
   this.resource('email');
   this.resource('phone');
  });
});
```

اگر مسیریابی‌های email و یا phone را به صورت مستقل مانند about و یا posts تعریف کنیم، با کلیک کاربر بر روی لینک متناظر با هر کدام، یک صفحه‌ی کاملا جدید نمایش داده می‌شود. اما در اینجا قصد داریم تنها قسمت کوچکی از همان صفحه‌ی contact را با محتوای ایمیل و یا شماره تماس جایگزین نمائیم. به همین جهت مسیریابی‌های متناظر را در اینجا به صورت تو در تو و ذیل مسیریابی contact تعریف کرده‌ایم.

پس از آن دو فایل قالب جدید Scripts\Templates\email.hbs را با محتوای:

```
<h2>Email</h2>
<p>
<span></span> Email name@site.com.
</p>
```

و فایل قالب Scripts\Templates\phone.hbs را با محتوای:

```
<h2>Phone</h2>
<p>
<span></span> Call 12345678.
</p>
```

به پوشه‌ی قالب‌ها اضافه نمائید به همراه معرفی نام آن‌ها به template loader برنامه در صفحه‌ی index.html :

```
<script>
 EmberHandlebarsLoader.loadTemplates([
 'posts', 'about', 'application', 'contact', 'email', 'phone' ]);
</script>
```

اکنون به قالب contact.hbs مجددا مراجعه کرده و تعاریف آن را به نحو ذیل تغییر دهید:

```
<h1>Contact</h1>
<div class="row">
  <div class="col-md-6">
   <p>
    Want to get in touch?
    <ul>
      <li>{{#link-to 'phone'}}Phone{{/link-to}}</li>
      <li>{{#link-to 'email'}}Email{{/link-to}}</li>
    </ul>
   </p>
  </div>
  <div class="col-md-6">
   {{outlet}}
  </div>
</div>
```

در اینجا دو لینک به مسیریابی‌های Phone و Email تعریف شده‌اند. همچنین {{outlet}} نیز قابل مشاهده است. با کلیک بر روی لینک Phone، مسیریابی آن فعال شده و سپس قالب متناظر با آن در قسمت {{outlet}} رندر می‌شود. در مورد لینک Email نیز به همین نحو رفتار خواهد شد.

در اینجا نحوه‌ی پردازش مسیریابی contact را ملاحظه می‌کنید. ابتدا درخواستی جهت مشاهده‌ی آدرس http://localhost/#/contact دریافت می‌شود. سپس router این درخواست را به مسیریابی همنامی منتقل می‌کند. این مسیریابی ابتدا قالب عمومی application را رندر کرده و سپس قالب اصلی و همنام مسیریابی جاری یا همان contact.hbs را رندر می‌کند. در این صفحه چون مسیریابی تو در تویی تعریف شده‌است، اگر درخواستی برای مشاهده‌ی http://localhost/#/contact/phone دریافت شود، محتوای آن‌را در {{outlet}} قالب contact.hbs جاری رندر می‌کند.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS03_01.zip

عنوان: ساخت یک بلاگ ساده با Ember.js، قسمت دوم
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۷ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, SPA, EmberJS

پس از تهیه ساختار اولیه‌ی بلاگی مبتنی بر ember.js در قسمت قبل ، در ادامه قصد داریم امکانات تعاملی را به آن اضافه کنیم. بنابراین کار را با تعریف کنترلرها که تعیین کننده‌ی رفتار برنامه هستند، ادامه می‌دهیم.

اضافه کردن دکمه‌ی More info به صفحه‌ی About و مدیریت کلیک بر روی آن

فایل Scripts\Templates\about.hbs را گشوده و سپس محتوای فعلی آن را به نحو ذیل تکمیل کنید:

```
<h2>About Ember Blog</h2>

<p>Bla bla bla!</p>

<button class="btn btn-primary" {{action 'showRealName' }}>more info</button>
```

در ember.js اگر قصد مدیریت عملی را که قرار است توسط کلیک بر روی المانی رخ دهد، داشته باشیم، می‌توان از handlebar helper ایی به نام action استفاده کرد. سپس برای تهیه کدهای مرتبط با آن، این اکشن را باید در کنترلر متناظر با route جاری (مسیریابی about) اضافه کنیم.
به همین جهت فایل جدید Scripts\Controllers\about.js را در پوشه‌ی کنترلرهای سمت کاربر اضافه کنید (نام آن با نام مسیریابی یکی است)؛ با این محتوا:

```
Blogger.AboutController = Ember.Controller.extend({
  actions: {
   showRealName: function () {
    alert("You clicked at showRealName of AboutController.");
   }
  }
});
```

کنترلرها به صورت یک خاصیت جدید به شیء Application برنامه اضافه می‌شوند. مطابق اصول نامگذاری ember.js، نام خاصیت کنترلر با حروف بزرگ متناظر با route آن شروع می‌شود و به نام Controller ختم خواهد شد. به این ترتیب ember.js هرگاه قصد پردازش مسیریابی about را داشته باشد، می‌داند که باید از کدام شیء جهت پردازش اعمال کاربر استفاده کند.
در ادامه این خاصیت را با تهیه یک زیرکلاس از کلاس پایه Controller تهیه شده توسط ember.js مقدار دهی می‌کنیم. به این ترتیب به کلیه امکانات این کلاس پایه دسترسی خواهیم داشت؛ به علاوه می‌توان ویژگی‌های سفارشی را نیز به آن افزود. برای مثال در اینجا در قسمت actions آن، دقیقا مطابق نام اکشنی که در فایل about.hbs تعریف کرده‌ایم، یک متد جدید اضافه شده‌است.

پس از تعریف کنترلر about.js نیاز است مدخل متناظر با آن‌را به فایل index.html برنامه نیز در انتهای تعاریف موجود، اضافه کرد:

```
<script src="Scripts/Controllers/about.js" type="text/javascript"></script>
```

اکنون یکبار برنامه را اجرا کرده و در صفحه‌ی about بر روی دکمه‌ی more info کلیک کنید.

اضافه کردن دکمه‌ی ارسال پیام خصوصی به صفحه‌ی Contact و مدیریت کلیک بر روی آن

در ادامه به قالب فعلی Scripts\Templates\contact.hbs یک دکمه را جهت ارسال پیام خصوصی اضافه می‌کنیم.

```
<h1>Contact</h1>
<div class="row">
  <div class="col-md-6">
   <p>
    Want to get in touch?
    <ul>
      <li>{{#link-to 'phone'}}Phone{{/link-to}}</li>
      <li>{{#link-to 'email'}}Email{{/link-to}}</li>
    </ul>
   </p>

   <p>
    Or, click here to send a secret message:
   </p>
   <button class="btn btn-primary" {{action 'sendMessage' }}>Send message</button>
  </div>
  <div class="col-md-6">
   {{outlet}}
  </div>
</div>
```

سپس برای مدیریت اکشن جدید sendMessage نیاز است کنترلر آن‌را نیز تعریف کنیم. با توجه به نام مسیریابی جاری، نام این کنترلر نیز contact خواهد بود. برای این منظور ابتدا فایل جدید Scripts\Controllers\contact.js را اضافه نمائید؛ با این محتوا:

```
Blogger.ContactController = Ember.Controller.extend({
  actions: {
   sendMessage: function () {
    var message = prompt('Type your message here:');
   }
  }
});
```

همچنین مدخل متناظر با فایل contact.js نیز باید به صفحه‌ی index.html اضافه شود:

```
<script src="Scripts/Controllers/contact.js" type="text/javascript"></script>
```

نمایش تصویری تعاملی در صفحه‌ی about

تا اینجا با نحوه‌ی تعریف اکشن‌ها در قالب‌ها و مدیریت آن‌ها توسط کنترلرهای متناظر آشنا شدیم. در ادامه قصد داریم با اصول binding اطلاعات در ember.js آشنا شویم. برای مثال فرض کنید می‌خواهیم دکمه‌ای را در صفحه‌ی about قرار داده و با کلیک بر روی آن، لوگوی ember.js را که به صورت یک تصویر مخفی در صفحه قرار دارد، نمایان کنیم. برای اینکار نیاز است خاصیتی را در کنترلر متناظر، تعریف کرده و سپس آن‌را به template جاری bind کرد.

برای این منظور فایل Scripts\Templates\about.hbs را گشوده و تعاریف موجود آن‌را به نحو ذیل تکمیل کنید:

```
<h2>About Ember Blog</h2>
<p>Bla bla bla!</p>
<button class="btn btn-primary" {{action 'showRealName' }}>more info</button>

{{#if isAuthorShowing}}
<button class="btn btn-warning" {{action 'hideAuthor' }}>Hide Image</button>
<p><img src="Content/images/ember-productivity-sm.png"></p>
{{else}}
<button class="btn btn-info" {{action 'showAuthor' }}>Show Image</button>
{{/if}}
```

در اینجا بر اساس مقدار خاصیت isAuthorShowing تصمیم گیری خواهد شد که آیا تصویر لوگوی ember.js نمایش داده شود یا خیر. همچنین دو اکشن نمایش و مخفی کردن تصویر نیز اضافه شده‌اند که با کلیک بر روی هر کدام، سبب تغییر وضعیت خاصیت isAuthorShowing خواهیم شد.

کنترلر about (فایل Scripts\Controllers\about.js) جهت مدیریت این خاصیت جدید، به همراه دو اکشن تعریف شده، اینبار به نحو ذیل تغییر خواهد یافت:

```
Blogger.AboutController = Ember.Controller.extend({
  isAuthorShowing: false,
  actions: {
   showRealName: function () {
    alert("You clicked at showRealName of AboutController.");
   },
   showAuthor: function () {
    this.set('isAuthorShowing', true);
   },
   hideAuthor: function () {
    this.set('isAuthorShowing', false);
   }
  }
});
```

ابتدا خاصیت isAuthorShowing به کنترلر اضافه شده‌است. از این خاصیت بار اولی که مسیر http://localhost:25918/#/about توسط کاربر درخواست می‌شود، استفاده خواهد شد.

سپس در دو متد showAuthor و hideAuthor که به اکشن‌های دو دکمه‌ی جدید تعریف شده در قالب about متصل خواهند شد، نحوه‌ی تغییر مقدار خاصیت isAuthorShowing را توسط متد set ملاحظه می‌کنید.

این قسمت مهم‌ترین تفاوت ember.js با jQuery است. در jQuery مستقیما المان‌های صفحه در همانجا تغییر داده می‌شوند. در ember.js منطق مدیریت کننده‌ی رابط کاربری و کدهای قالب متناظر با آن از هم جدا شده‌اند تا بتوان یک برنامه‌ی بزرگ را بهتر مدیریت کرد. همچنین در اینجا مشخص است که هر قسمت و هر فایل، چه ارتباطی با سایر اجزای تعریف شده دارد و چگونه به هم متصل شده‌اند و اینبار شاهد انبوهی از کدهای جاوا اسکریپتی مخلوط بین المان‌های HTML صفحه نیستیم.

نمایش پیامی به کاربر پس از ارسال پیام خصوصی در صفحه‌ی تماس با ما

قصد داریم ویژگی مشابهی را به صفحه‌ی contact نیز اضافه کنیم. اگر کاربر بر روی دکمه‌ی ارسال پیام کلیک کرد، پیام تشکری به همراه عددی ویژه به او نمایش خواهیم داد.

برای این‌کار قالب Scripts\Templates\contact.hbs را به نحو ذیل تکمیل کنید:

```
<h1>Contact</h1>
<div class="row">
  <div class="col-md-6">
   <p>
    Want to get in touch?
    <ul>
      <li>{{#link-to 'phone'}}Phone{{/link-to}}</li>
      <li>{{#link-to 'email'}}Email{{/link-to}}</li>
    </ul>
   </p>
   {{#if messageSent}}
   <p>
    Thank you. Your message has been sent.
    Your confirmation number is {{confirmationNumber}}.
   </p>
   {{else}}
   <p>
    Or, click here to send a secret message:
   </p>
   <button class="btn btn-primary" {{action 'sendMessage' }}>Send message</button>
   {{/if}}
  </div>
  <div class="col-md-6">
   {{outlet}}
  </div>
</div>
```

در آن شرط بررسی messageSent if اضافه شده‌است؛ به همراه نمایش confirmationNumber در انتهای پیام تشکر.

برای تعریف منطق مرتبط با این خواص، به کنترلر contact واقع در فایل Scripts\Controllers\contact.js مراجعه کرده و آن‌را به نحو ذیل تغییر می‌دهیم:

```
Blogger.ContactController = Ember.Controller.extend({
  messageSent: false,
  actions: {
   sendMessage: function () {
    var message = prompt('Type your message here:');
    if (message) {
      this.set('confirmationNumber', Math.round(Math.random() * 100000));
      this.set('messageSent', true);
    }
   }
  }
});
```

همانطور که مشاهده می‌کنید، مقدار اولیه خاصیت messageSent مساوی false است. بنابراین در قالب contact.hbs قسمت else شرط نمایش داده می‌شود. اگر کاربر پیامی را وارد کند، خاصیت confirmationNumber به یک عدد اتفاقی و خاصیت messageSent به true تنظیم خواهد شد. به این ترتیب اینبار به صورت خودکار پیام تشکر به همراه عددی اتفاقی، به کاربر نمایش داده می‌شود.

بنابراین به صورت خلاصه، کار کنترلر، مدیریت منطق نمایشی برنامه است و برای اینکار حداقل دو مکانیزم را ارائه می‌دهد: اکشن‌ها و خواص. اکشن‌ها بیانگر نوعی رفتار هستند؛ برای مثال نمایش یک popup و یا تغییر مقدار یک خاصیت. مقدار خواص را می‌توان مستقیما در صفحه نمایش داد و یا از آن‌ها جهت پردازش عبارات شرطی و نمایش قسمت خاصی از قالب جاری نیز می‌توان کمک گرفت.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS03_02.zip

عنوان: ساخت یک بلاگ ساده با Ember.js، قسمت سوم
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۸ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, SPA, EmberJS

پس از ایجاد کنترلرها ، در این قسمت سعی خواهیم کرد تا آرایه‌ای ثابت از مطالب و نظرات را در سایت نمایش دهیم. همچنین امکان ویرایش اطلاعات را نیز به این آرایه‌های جاوا اسکریپتی مدل، اضافه خواهیم کرد.

تعریف مدل سمت کاربر برنامه

فایل جدید Scripts\App\store.js را اضافه کرده و محتوای آن‌را به نحو ذیل تغییر دهید:

```
var posts = [
  {
      id: '1',
      title: "Getting Started with Ember.js",
      body: "Bla bla bla 1."
  },
  {
      id: '2',
      title: "Routes and Templates",
      body: "Bla bla bla 2."
  },
  {
      id: '3',
      title: "Controllers",
      body: "Bla bla bla 3."
  }
];

var comments = [
    {
        id: '1',
        postId: '3',
        text: 'Thanks!'
    },
    {
        id: '2',
        postId: '3',
        text: 'Good to know that!'
    },
    {
        id: '3',
        postId: '1',
        text: 'Great!'
    }
];
```

در اینجا دو آرایه ثابت از اشیاء مطالب و نظرات را مشاهده می‌کنید.
سپس جهت استفاده از آن، تعریف مدخل آن‌را به فایل index.html، پیش از تعاریف کنترلرها اضافه خواهیم کرد:

```
<script src="Scripts/App/store.js" type="text/javascript"></script>
```

ویرایش قالب مطالب برای نمایش لیستی از عناوین ارسالی

قالب فعلی Scripts\Templates\posts.hbs صرفا دارای یک سری عنوان درج شده به صورت مستقیم در صفحه است. اکنون قصد داریم آن‌را جهت نمایش لیستی از آرایه مطالب تغییر دهیم.

همانطور که در تصویر ملاحظه می‌کنید، با درخواست آدرس صفحه‌ی مطالب، router آن مسیریابی متناظری را یافته و سپس بر این اساس، template، کنترلر و مدلی را انتخاب می‌کند. به صورت پیش فرض، قالب و کنترلر انتخاب شده، مواردی هستند همنام با مسیریابی جاری. اما مقدار پیش فرضی برای model وجود ندارد و باید آن‌را به صورت دستی مشخص کرد.
برای این منظور فایل Scripts\Routes\posts.js را به پوشه‌ی routes با محتوای ذیل اضافه کنید:

```
Blogger.PostsRoute = Ember.Route.extend({
    controllerName: 'posts',
    renderTemplare: function () {
        this.render('posts');
    },
    model: function () {
        return posts;
    }
});
```

در اینجا صرفا جهت نمایش پیش فرض‌ها و نحوه‌ی کار یک route، دو خاصیت controllerName و renderTemplare آن نیز مقدار دهی شده‌اند. این دو خاصیت به صورت پیش فرض، همنام مسیریابی جاری مقدار دهی می‌شوند و نیازی به ذکر صریح آن‌ها نیست. اما خاصیت model یک مسیریابی است که باید دقیقا مشخص شود. در اینجا مقدار آن‌را به آرایه posts تعریف شده در فایل Scripts\App\store.js تنظیم کرده‌ایم. به این ترتیب مدل تعریف شده در اینجا، به صورت خودکار در کنترلر posts و قالب متناظر با آن، قابل استفاده خواهد بود.
همچنین اگر به خاطر داشته باشید، در پوشه‌ی کنترلرها فایل posts.js تعریف نشده‌است. اگر اینکار صورت نگیرد، ember.js به صورت خودکار کنترلر پیش فرضی را ایجاد خواهد کرد. در کل، یک قالب هیچگاه به صورت مستقیم با مدل کار نمی‌کند. این کنترلر است که مدل را در اختیار یک قالب قرار می‌دهد.
سپس مدخل تعریف این فایل را به فایل index.html، پس از تعاریف کنترلرها اضافه نمائید:

```
<script src="Scripts/Routes/posts.js" type="text/javascript"></script>
```

اکنون فایل Scripts\Templates\posts.hbs را گشوده و به نحو ذیل، جهت نمایش عناوین مطالب، ویرایش کنید:

```
<h2>Emeber.js blog</h2>
<ul>
    {{#each post in model}}
    <li>{{post.title}}</li>
    {{/each}}
</ul>
```

در این قالب، حلقه‌ای بر روی عناصر model تشکیل شده و سپس خاصیت title هر عضو نمایش داده می‌شود.

نمایش لیست آخرین نظرات ارسالی

در ادامه قصد داریم تا آرایه comments ابتدای بحث را در صفحه‌ای جدید نمایش دهیم. بنابراین نیاز است تا ابتدا مسیریابی آن تعریف شود. بنابراین فایل Scripts\App\router.js را گشوده و مسیریابی جدید recent-comments را به آن اضافه کنید:

```
Blogger.Router.map(function () {
    this.resource('posts', { path: '/' });
    this.resource('about');
    this.resource('contact', function () {
        this.resource('email');
        this.resource('phone');
    });
    this.resource('recent-comments');
});
```

سپس جهت تعیین مدل این مسیریابی جدید نیاز است تا فایل Scripts\Routes\recent-comments.js را در پوشه‌ی routes با محتوای ذیل اضافه کرد:

```
Blogger.RecentCommentsRoute = Ember.Route.extend({
    model: function () {
        return comments;
    }
});
```

در اینجا آرایه comments بازگشتی، همان آرایه‌ای است که در ابتدای بحث در فایل Scripts\App\store.js تعریف کردیم.
همچنین نیاز است تا تعریف مدخل این فایل جدید را نیز به انتهای تعاریف مداخل فایل index.html اضافه کنیم:

```
<script src="Scripts/Routes/recent-comments.js" type="text/javascript"></script>
```

اکنون قالب application واقع در فایل Scripts\Templates\application.hbs را جهت افزودن منوی مرتبط با این مسیریابی جدید، به نحو ذیل ویرایش خواهیم کرد:

```
<div class='container'>
    <nav class='navbar navbar-default' role='navigation'>
        <ul class='nav navbar-nav'>
            <li>{{#link-to 'posts'}}Posts{{/link-to}}</li>
            <li>{{#link-to 'recent-comments'}}Recent comments{{/link-to}}</li>
            <li>{{#link-to 'about'}}About{{/link-to}}</li>
            <li>{{#link-to 'contact'}}Contact{{/link-to}}</li>
        </ul>
    </nav>
    {{outlet}}
</div>
```

و در آخر قالب جدید Scripts\Templates\recent-comments.hbs را برای نمایش لیست آخرین نظرات، با محتوای زیر اضافه می‌کنیم:

```
<h1>Recent comments</h1>
<ul>
  {{#each comment in model}}
    <li>{{comment.text}}</li>
  {{/each}}
</ul>
```

برای فعال شدن آن نیاز است تا تعریف این قالب جدید را به template loader برنامه، در فایل index.html اضافه کنیم:

```
<script type="text/javascript">
    EmberHandlebarsLoader.loadTemplates([
       'posts', 'about', 'application', 'contact', 'email', 'phone',
       'recent-comments'
    ]);
</script>
```

نمایش مجزای هر مطلب در یک صفحه‌ی جدید

تا اینجا در صفحه‌ی اول سایت، لیست عناوین مطالب را نمایش دادیم. در ادامه نیاز است تا بتوان هر عنوان را به صفحه‌ی متناظر و اختصاصی آن لینک کرد؛ برای مثال لینکی مانند http://localhost:25918/#/posts/3 به سومین مطلب ارسالی اشاره می‌کند. Ember.js به عدد 3 در اینجا، یک dynamic segment می‌گوید. از این جهت که مقدار آن بر اساس شماره مطلب درخواستی، متفاوت خواهد بود. برای پردازش این نوع آدرس‌ها نیاز است مسیریابی ویژه‌ای را تعریف کرد. فایل Scripts\App\router.js را گشوده و سپس مسیریابی post را به نحو ذیل به آن اضافه نمائید:

```
Blogger.Router.map(function () {
    this.resource('posts', { path: '/' });
    this.resource('about');
    this.resource('contact', function () {
        this.resource('email');
        this.resource('phone');
    });
    this.resource('recent-comments');
    this.resource('post', { path: 'posts/:post_id' });
});
```

قسمت پویای مسیریابی با یک : مشخص می‌شود.

با توجه به اینکه این مسیریابی جدید post نام گرفت (جهت نمایش یک مطلب)، به صورت خودکار، کنترلر و قالبی به همین نام را بارگذاری می‌کند. همچنین مدل خود را نیز باید از مسیریابی خاص خود دریافت کند. بنابراین فایل جدید Scripts\Routes\post.js را در پوشه‌ی routes با محتوای ذیل اضافه کنید:

```
Blogger.PostRoute = Ember.Route.extend({
    model: function (params) {
        return posts.findBy('id', params.post_id);
    }
});
```

در اینجا مدل مسیریابی post بر اساس پارامتری به نام params تعیین می‌شود. این پارامتر حاوی مقدار متغیر پویای post_id که در مسیریابی جدید post مشخص کردیم می‌باشد. در ادامه از آرایه posts تعریف شده در ابتدای بحث، توسط متد findBy که توسط Ember.js اضافه شده‌است، عنصری را که خاصیت id آن مساوی post_id دریافتی است، انتخاب کرده و به عنوان مقدار مدل بازگشت می‌دهیم.

برای مثال، جهت آدرس http://localhost:25918/#/posts/3، مقدار post_id به صورت خودکار به عدد 3 تنظیم می‌شود.

پس از آن نیاز است مدخل این فایل جدید را در صفحه‌ی index.html نیز اضافه کنیم:

```
<script src="Scripts/Routes/post.js" type="text/javascript"></script>
```

در ادامه برای نمایش اطلاعات مدل نیاز است قالب جدید Scripts\Templates\post.hbs را با محتوای زیر اضافه کنیم:

```
<h1>{{title}}</h1>
<p>{{body}}</p>
```

و template loader صفحه‌ی index.html را نیز باید از وجود آن باخبر کرد:

```
<script type="text/javascript">
    EmberHandlebarsLoader.loadTemplates([
       'posts', 'about', 'application', 'contact', 'email', 'phone',
       'recent-comments', 'post'
    ]);
</script>
```

اکنون به قالب Scripts\Templates\posts.hbs مراجعه کرده و هر عنوان را به مطلب متناظر با آن لینک می‌کنیم:

```
<h2>Emeber.js blog</h2>
<ul>
```

```
    {{#each post in model}}
    <li>{{#link-to 'post' post.id}}{{post.title}}{{/link-to}}</li>
    {{/each}}
</ul>
```

همانطور که ملاحظه می‌کنید، link-to امکان پذیرش id یک مطلب را به صورت متغیر نیز دارا است که سبب خواهد شد تا عناوین، به مطالب متناظر لینک شوند:

همچنین با کلیک بر روی هر عنوان نیز مطلب مرتبط نمایش داده خواهد شد:

افزودن امکان ویرایش مطالب

می‌خواهیم در صفحه‌ی نمایش جزئیات یک مطلب، امکان ویرایش آن‌را نیز فراهم کنیم. بنابراین فایل Scripts\Templates\post.hbs را گشوده و محتوای آن‌را به نحو ذیل ویرایش کنید:

```
<h2>{{title}}</h2>
{{#if isEditing}}
<form>
    <div class="form-group">
        <label for="title">Title</label>
        {{input value=title id="title" class="form-control"}}
    </div>
    <div class="form-group">
        <label for="body">Body</label>
        {{textarea value=body id="body" class="form-control" rows="5"}}
    </div>
    <button class="btn btn-primary" {{action 'save' }}>Save</button>
</form>
{{else}}
<p>{{body}}</p>
<button class="btn btn-primary" {{action 'edit' }}>Edit</button>
{{/if}}
```

شبیه به این if و else را در قسمت قبل حین ایجاد صفحات about و یا contact نیز مشاهده کرده‌اید. در اینجا اگر خاصیت isEditing مساوی true باشد، فرم ویرایش اطلاعات ظاهر می‌شود و اگر خیر، محتوای مطلب جاری نمایش داده خواهد شد.
در فرم تعریف شده، المان‌های ورودی اطلاعات از handlebar helperهای ویژه‌ی input و textarea استفاده می‌کنند؛ بجای المان‌های متداول HTML. همچنین value یکی به title و دیگری به body تنظیم شده‌است (خواص مدل ارائه شده توسط کنترلر متصل به قالب). این مقادیر نیز داخل '' قرار ندارند؛ به عبارتی در یک handlebar helper به عنوان متغیر در نظر گرفته می‌شوند.
به این ترتیب اطلاعات کنترلر جاری، به این المان‌های ورودی اطلاعات به صورت خودکار bind می‌شوند و برعکس. اگر کاربر مقادیر آن‌ها را تغییر دهد، تغییرات نهایی به صورت خودکار به خواص متناظری در کنترلر جاری منعکس خواهند شد (two-way data binding).
دو دکمه نیز تعریف شده‌اند که به اکشن‌های save و edit متصل هستند.
بنابراین نیاز به یک کنترلر جدید، به نام post داریم تا بتوان رفتار قالب post را کنترل کرد. برای این منظور فایل جدید Scripts\Controllers\post.js را با محتوای ذیل ایجاد کنید:

```
Blogger.PostController = Ember.ObjectController.extend({
    isEditing: false,
    actions: {
        edit: function () {
            this.set('isEditing', true);
        },
        save: function () {
            this.set('isEditing', false);
        }
    }
});
```

همچنین مدخل تعریف آن‌را نیز به فایل index.html اضافه نمائید (پس از تعاریف کنترلرهای موجود):

```
<script src="Scripts/Controllers/post.js" type="text/javascript"></script>
```

اگر به کدهای این کنترلر دقت کرده باشید، اینبار زیرکلاسی از ObjectController ایجاد شده‌است و نه Controller، مانند مثال‌های قبل . ObjectController تغییرات رخ داده بر روی خواص مدل را که توسط کنترلر در معرض دید قالب قرار داده‌است، به صورت خودکار به مدل مرتبط نیز منعکس می‌کند (Ember.ObjectController.extend)؛ اما Controller خیر (Ember.Controller.extend). در اینجا مدل کنترلر، تنها «یک» شیء است که بر اساس id آن انتخاب شده‌است. به همین جهت از ObjectController برای ارائه two-way data binding کمک گرفته شد.
در ember.js، یک قالب تنها با کنترلر خودش دارای تبادل اطلاعات است. اگر این کنترلر از نوع ObjectController باشد، تغییرات خاصیتی در یک قالب، ابتدا به کنترلر آن منعکس می‌شود و سپس این کنترلر، در صورت یافتن معادلی از این خاصیت در مدل، آن‌را به روز خواهد کرد. در حالت استفاده از Controller معمولی، صرفا تبادل اطلاعات بین قالب و کنترلر را شاهد خواهیم بود و نه بیشتر.

در ابتدای کار مقدار خاصیت isEditing مساوی false است. این مورد سبب می‌شود تا در بار اول بارگذاری اطلاعات یک مطلب

انتخابی، صرفا عنوان و محتوای مطلب نمایش داده شوند؛ به همراه یک دکمه‌ی edit. با کلیک بر روی دکمه‌ی edit، مطابق کدهای کنترلر فوق، تنها خاصیت isEditing به true تنظیم می‌شود و در این حالت، بدنه‌ی اصلی شرط if isEditing در قالب post، رندر خواهد شد.

برای مثال در ابتدا مطلب شماره یک را انتخاب می‌کنیم:

با کلیک بر روی دکمه‌ی edit، فرم ویرایش ظاهر خواهد شد:

نکته‌ی جالب آن، مقدار دهی خودکار المان‌های ویرایش اطلاعات است. در این حالت سعی کنید، عنوان مطلب جاری را اندکی ویرایش کنید:

با ویرایش عنوان، می‌توان بلافاصله مقدار تغییر یافته را در برچسب عنوان مطلب نیز مشاهده کرد. این مورد دقیقا مفهوم two-way data binding و اتصال مقادیر value هر کدام از handlebar helperهای ویژه‌ی input و textarea را به عناصر مدل ارائه شده توسط کنترلر post، بیان می‌کند.
در این حالت در کدهای متد save، تنها کافی است که خاصیت isEditing را به false تنظیم کنیم. زیرا کلیه مقادیر ویرایش شده توسط کاربر، در همان لحظه در برنامه منتشر شده‌اند و نیاز به کار بیشتری برای اعمال تغییرات نیست.

اضافه کردن دکمه‌ی مرتب سازی بر اساس عناوین، در صفحه‌ی اول سایت

Ember.ObjectController.extend برای data bindg یک شیء کاربرد دارد. اگر قصد داشته باشیم با آرایه‌ای از اشیاء کار کنیم می‌توان از ArrayController استفاده کرد. فرض کنید در صفحه‌ی اول سایت می‌خواهیم امکان مرتب سازی مطالب را بر اساس عنوان آن‌ها اضافه کنیم. فایل Scripts\Templates\posts.hbs را گشوده و لینک Sort by title را به انتهای آن اضافه کنید:

```
<h2>Emeber.js blog</h2>
<ul>
    {{#each post in model}}
    <li>{{#link-to 'post' post.id}}{{post.title}}{{/link-to}}</li>
    {{/each}}
</ul>

<a href="#" class="btn btn-primary" {{action 'sortByTitle'}}>Sort by title</a>
```

در اینجا چون قصد تغییر رفتار قالب posts را توسط اکشن جدید sortByTitle داریم، نیاز است کنترلر متناظر با آن‌را نیز اضافه کنیم. برای این منظور فایل جدید Scripts\Controllers\posts.js را به پوشه‌ی کنترلرها اضافه کنید؛ با محتوای ذیل:

```
Blogger.PostsController = Ember.ArrayController.extend({
    sortProperties: ['id'],// مقادیر پیش فرض مرتب سازی
    sortAscending: false,
    actions: {
        sortByTitle: function () {
            this.set('sortProperties', ['title']);
            this.set('sortAscending', !this.get('sortAscending'));
        }
    }
});
```

sortProperties جزو خواص کلاس پایه ArrayController است. اگر مانند سطر اول به صورت مستقیم مقدار دهی شود، خاصیت یا خواص پیش فرض مرتب سازی را مشخص می‌کند. اگر مانند اکشن sortByTitle توسط متد set مقدار دهی شود، امکان مرتب سازی تعاملی و با فرمان کاربر را فراهم خواهد کرد.

در ادامه، تعریف مدخل این کنترلر جدید را نیز باید به فایل index.html، اضافه کرد:

```
<script src="Scripts/Controllers/posts.js" type="text/javascript"></script>
```

اگر برنامه را در این حالت اجرا کرده و بر روی دکمه‌ی Sort by title کلیک کنید، اتفاقی رخ نمی‌دهد. علت اینجا است که ArrayController خروجی تغییر یافته خودش را توسط خاصیتی به نام arrangedContent در اختیار قالب خود قرار می‌دهد. بنابراین نیاز است فایل قالب Scripts\Templates\posts.hbs را به نحو ذیل ویرایش کرد:

```
<h2>Emeber.js blog</h2>
<ul>
    {{#each post in arrangedContent}}
    <li>{{#link-to 'post' post.id}}{{post.title}}{{/link-to}}</li>
    {{/each}}
</ul>

<a href="#" class="btn btn-primary" {{action 'sortByTitle'}}>Sort by title</a>
```

اینبار کلیک بر روی دکمه‌ی مرتب سازی بر اساس عناوین، هربار لیست موجود را به صورت صعودی و یا نزولی مرتب می‌کند.

یک نکته: حلقه‌ی ویژه‌ای به نام each

اگر قالب Scripts\Templates\posts.hbs را به نحو ذیل، با یک حلقه‌ی each ساده بازنویسی کنید:

```
<h2>Ember.js blog</h2>
<ul>
    {{#each}}
    <li>{{#link-to 'post' id}}{{title}}{{/link-to}}</li>
    {{/each}}
</ul>

<a href="#" class="btn btn-primary" {{action 'sortByTitle'}}>Sort by title</a>
```

هم در حالت نمایش معمولی و هم در حالت استفاده از ArrayController برای نمایش اطلاعات مرتب شده، بدون مشکل کار می‌کند و نیازی به تغییر نخواهد داشت.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS03_03.zip

عنوان: ساخت یک بلاگ ساده با Ember.js، قسمت چهارم
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۹ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, SPA, EmberJS

در قسمت قبل ، اطلاعات نمایش داده شده، از یک سری آرایه ثابت جاوا اسکریپتی تامین شدند. در یک برنامه‌ی واقعی نیاز است داده‌ها را یا از HTML 5 local storage تامین کرد و یا از سرور به کمک Ajax. برای اینگونه اعمال، ember.js به همراه افزونه‌ای است به نام Ember Data که جزئیات کار با آن‌را در این قسمت بررسی خواهیم کرد.

استفاده از Ember Data با Local Storage

برای کار با HTML 5 local storage نیاز به Ember Data Local Storage Adapter نیز هست که در قسمت اول این سری، آدرس دریافت آن معرفی شد. این فایل‌ها نیز در پوشه‌ی Scripts\Libs برنامه کپی خواهند شد.
در ادامه به فایل Scripts\App\store.js که در قسمت قبل جهت تعریف دو آرایه ثابت مطالب و نظرات اضافه شد، مراجعه کرده و محتوای فعلی آن‌را با کدهای زیر جایگزین کنید:

```
Blogger.ApplicationSerializer = DS.LSSerializer.extend();
Blogger.ApplicationAdapter = DS.LSAdapter.extend();
```

این تعاریف سبب خواهند شد تا Ember Data از Local Storage Adapter استفاده کند.
در ادامه با توجه به حذف دو آرایه‌ی posts و comments که پیشتر در فایل store.js تعریف شده بودند، نیاز است مدل‌های متناظری را جهت تعریف خواص آن‌ها، به برنامه اضافه کنیم. این‌کار را با افزودن دو فایل جدید comment.js و post.js به پوشه‌ی Scripts\Models انجام خواهیم داد.
محتوای فایل Scripts\Models\post.js :

```
Blogger.Post = DS.Model.extend({
  title: DS.attr(),
  body: DS.attr()
});
```

محتوای فایل Scripts\Models\comment.js :

```
Blogger.Comment = DS.Model.extend({
  text: DS.attr()
});
```

سپس مداخل تعریف آن‌ها را به فایل index.html نیز اضافه خواهیم کرد:

```
<script src="Scripts/Models/post.js" type="text/javascript"></script>
<script src="Scripts/Models/comment.js" type="text/javascript"></script>
```

برای تعاریف مدل‌ها در Ember data مرسوم است که نام مدل‌ها، اسامی جمع نباشند. سپس با ایجاد وهله‌ای از DS.Model.extend یک مدل ember data را تعریف خواهیم کرد. در این مدل، خواص هر شیء را مشخص کرده و مقدار آن‌ها همیشه DS.attr() خواهد بود. این نکته را در دو مدل Post و Comment مشاهده می‌کنید. اگر دقت کنید به هر دو مدل، خاصیت id اضافه نشده‌است. این خاصیت به صورت خودکار توسط Ember data تنظیم می‌شود.

اکنون نیاز است برنامه را جهت استفاده از این مدل‌های جدید به روز کرد. برای این منظور فایل Scripts\Routes\posts.js را گشوده و مدل آن‌را به نحو ذیل ویرایش کنید:

```
Blogger.PostsRoute = Ember.Route.extend({
    //controllerName: 'posts', // مقدار پیش فرض است و نیازی به ذکر آن نیست
```

```
    //renderTemplare: function () {
    //    this.render('posts'); // مقدار پیش فرض است و نیازی به ذکر آن نیست
    //},
    model: function () {
        return this.store.find('post');
    }
});
```

در اینجا this.store معادل data store برنامه است که مطابق تنظیمات برنامه، همان ember data می‌باشد. سپس متد find را به همراه نام مدل، به صورت رشته‌ای در اینجا مشخص می‌کنیم.

به همین ترتیب فایل Scripts\Routes\recent-comments.js را نیز جهت استفاده از data store ویرایش خواهیم کرد:

```
Blogger.RecentCommentsRoute = Ember.Route.extend({
    model: function () {
        return this.store.find('comment');
    }
});
```

و فایل Scripts\Routes\post.js که در آن منطق یافتن یک مطلب بر اساس آدرس مختص به آن قرار دارد، به صورت ذیل بازنویسی می‌شود:

```
Blogger.PostRoute = Ember.Route.extend({
    model: function (params) {
        return this.store.find('post', params.post_id);
    }
});
```

اگر متد find بدون پارامتر ذکر شود، به معنای بازگشت تمامی عناصر موجود در آن مدل خواهد بود و اگر پارامتر دوم آن مانند این مثال تنظیم شود، تنها همان وهله‌ی درخواستی را بازگشت می‌دهد.

افزودن امکان ثبت یک مطلب جدید

تا اینجا اگر برنامه را اجرا کنید، برنامه بدون خطا بارگذاری خواهد شد اما فعلا رکوردی را برای نمایش ندارد. در ادامه، برنامه را جهت افزودن مطالب جدید توسعه خواهیم داد. برای اینکار ابتدا به فایل Scripts\App\router.js مراجعه کرده و سپس مسیریابی جدید new-post را تعریف خواهیم کرد:

```
Blogger.Router.map(function () {
    this.resource('posts', { path: '/' });
    this.resource('about');
    this.resource('contact', function () {
        this.resource('email');
        this.resource('phone');
    });
    this.resource('recent-comments');
    this.resource('post', { path: 'posts/:post_id' });
    this.resource('new-post');
});
```

اکنون در صفحه‌ی اول سایت، توسط قالب Scripts\Templates\posts.hbs، دکمه‌ای را جهت ایجاد یک مطلب جدید اضافه خواهیم کرد:

```
<h2>Ember.js blog</h2>
<ul>
    {{#each post in arrangedContent}}
    <li>{{#link-to 'post' post.id}}{{post.title}}{{/link-to}}</li>
    {{/each}}
</ul>

<a href="#" class="btn btn-primary" {{action 'sortByTitle' }}>Sort by title</a>
{{#link-to 'new-post' classNames="btn btn-success"}}New Post{{/link-to}}
```

در اینجا دکمه‌ی New Post به مسیریابی جدید new-post اشاره می‌کند.
برای تعریف عناصر نمایشی این مسیریابی، فایل جدید قالب Scripts\Templates\new-post.hbs را با محتوای زیر اضافه کنید:

```
<h1>New post</h1>
<form>
  <div class="form-group">
    <label for="title">Title</label>
    {{input value=title id="title" class="form-control"}}
  </div>

  <div class="form-group">
    <label for="body">Body</label>
    {{textarea value=body id="body" class="form-control" rows="5"}}
  </div>


  <button class="btn btn-primary" {{action 'save'}}>Save</button>
</form>
```

با نمونه‌ی این فرم در قسمت قبل در حین ویرایش یک مطلب، آشنا شدیم. دو المان دریافت اطلاعات در آن قرار دارند که هر کدام به خواص مدل برنامه bind شده‌اند. همچنین یک دکمه‌ی save، با اکشنی به همین نام در اینجا تعریف شده‌است.
پس از آن نیاز است نام فایل قالب new-post را به template loader برنامه در فایل index.html اضافه کرد:

```
<script type="text/javascript">
    EmberHandlebarsLoader.loadTemplates([
       'posts', 'about', 'application', 'contact', 'email', 'phone',
       'recent-comments', 'post', 'new-post'
    ]);
</script>
```

برای مدیریت دکمه‌ی save این قالب جدید نیاز است کنترلر جدیدی را در فایل جدید Scripts\Controllers\new-post.js تعریف کنیم؛ با این محتوا:

```
Blogger.NewPostController = Ember.Controller.extend({
    actions: {
        save: function () {
            var newPost = this.store.createRecord('post', {
                title: this.get('title'),
                body: this.get('body')
            });
            newPost.save();
            this.transitionToRoute('posts');
        }
    }
});
```

به همراه افزودن مدخلی از آن به فایل index.html برنامه:

```
<script src="Scripts/Controllers/new-post.js" type="text/javascript"></script>
```

در اینجا کنترلر جدید NewPostController را مشاهده می‌کنید. از این جهت که برای دسترسی به خواص مدل تغییر کرده، از متد this.get استفاده شده‌است، نیازی نیست حتما از یک ObjectController مانند قسمت قبل استفاده کرد و Controller معمولی نیز برای اینکار کافی است.
آرگومان اول this.store.createRecord نام مدل است و آرگومان دوم آن، وهله‌ای که قرار است به آن اضافه شود. همچنین باید دقت داشت که برای تنظیم یک خاصیت، از متد this.set و برای دریافت مقدار یک خاصیت تغییر کرده از this.get به همراه نام خاصیت مورد نظر استفاده می‌شود و نباید مستقیما برای مثال از this.title استفاده کرد.
this.store.createRecord صرفا یک شیء جدید (ember data object) را ایجاد می‌کند. برای ذخیره سازی نهایی آن باید متد save آن‌را فراخوانی کرد (پیاده سازی الگوی active record است). به این ترتیب این شیء در local storage ذخیره خواهد شد.
پس از ذخیره‌ی مطلب جدید، از متد this.transitionToRoute استفاده شده‌است. این متد، برنامه را به صورت خودکار به

صفحه‌ی متناظر با مسیریابی posts هدایت می‌کند.

اکنون برنامه را اجرا کنید. بر روی دکمه‌ی سبز رنگ new post در صفحه‌ی اول کلیک کرده و یک مطلب جدید را تعریف کنید. بلافاصله عنوان و لینک متناظر با این مطلب را در صفحه‌ی اول سایت مشاهده خواهید کرد.
همچنین اگر برنامه را مجددا بارگذاری کنید، این مطالب هنوز قابل مشاهده هستند؛ زیرا در local storage مرورگر ذخیره شده‌اند.

در اینجا اگر به لینک‌های تولید شده دقت کنید، id آن‌ها عددی نیست. این روشی است که local storage با آن کار می‌کند.

افزودن امکان حذف یک مطلب به سایت

برای حذف یک مطلب، دکمه‌ی حذف را به انتهای قالب Scripts\Templates\post.hbs اضافه خواهیم کرد:

```
<h2>{{title}}</h2>
{{#if isEditing}}
<form>
    <div class="form-group">
        <label for="title">Title</label>
        {{input value=title id="title" class="form-control"}}
    </div>
    <div class="form-group">
        <label for="body">Body</label>
        {{textarea value=body id="body" class="form-control" rows="5"}}
    </div>
    <button class="btn btn-primary" {{action 'save' }}>Save</button>
</form>
{{else}}
<p>{{body}}</p>
<button class="btn btn-primary" {{action 'edit' }}>Edit</button>
<button class="btn btn-danger" {{action 'delete' }}>Delete</button>
{{/if}}
```

سپس کنترلر Scripts\Controllers\post.js را جهت مدیریت اکشن جدید delete به نحو ذیل تکمیل می‌کنیم:

```
Blogger.PostController = Ember.ObjectController.extend({
    isEditing: false,
    actions: {
        edit: function () {
            this.set('isEditing', true);
        },
        save: function () {
            var post = this.get('model');
            post.save();

            this.set('isEditing', false);
        },
        delete: function () {
            if (confirm('Do you want to delete this post?')) {
                this.get('model').destroyRecord();
                this.transitionToRoute('posts');
            }
        }
    }
});
```

متد destroyRecord، مدل انتخابی را هم از حافظه و هم از data store حذف می‌کند. سپس کاربر را به صفحه‌ی اصلی سایت هدایت خواهیم کرد.

متد save نیز در اینجا بهبود یافته‌است. ابتدا مدل جاری دریافت شده و سپس متد save بر روی آن فراخوانی می‌شود. به این ترتیب اطلاعات از حافظه به local storage نیز منتقل خواهند شد.

ثبت و نمایش نظرات به همراه تنظیمات روابط اشیاء در Ember Data

در ادامه قصد داریم امکان افزودن نظرات را به مطالب، به همراه نمایش آن‌ها در ذیل هر مطلب، پیاده سازی کنیم. برای اینکار نیاز است رابطه‌ی بین یک مطلب و نظرات مرتبط با آن‌را در مدل ember data مشخص کنیم. به همین جهت فایل Scripts\Models\post.js را گشوده و تغییرات ذیل را به آن اعمال کنید:

```
Blogger.Post = DS.Model.extend({
  title: DS.attr(),
  body: DS.attr(),
  comments: DS.hasMany('comment', { async: true })
});
```

در اینجا خاصیت جدیدی به نام comments به مدل مطلب اضافه شده‌است و توسط آن می‌توان به تمامی نظرات یک مطلب دسترسی یافت؛ تعریف رابطه‌ی یک به چند، به کمک متد DS.hasMany که پارامتر اول آن نام مدل مرتبط است. تعریف async: true

برای کار با local storage اجباری است و در نگارش‌های آتی ember data حالت پیش فرض خواهد بود.
همچنین نیاز است یک سر دیگر رابطه را نیز مشخص کرد. برای این منظور فایل Scripts\Models\comment.js را گشوده و به نحو ذیل تکمیل کنید:

```
Blogger.Comment = DS.Model.extend({
    text: DS.attr(),
    post: DS.belongsTo('post', { async: true })
});
```

در اینجا خاصیت جدید post به مدل نظر اضافه شده‌است و مقدار آن از طریق متد DS.belongsTo که مدل post را به یک نظر، مرتبط می‌کند، تامین خواهد شد. بنابراین در این حالت اگر به شیء comment مراجعه کنیم، خاصیت جدید post.id آن، به id مطلب متناظر اشاره می‌کند.

در ادامه نیاز است بتوان تعدادی نظر را ثبت کرد. به همین جهت با تعریف مسیریابی آن شروع می‌کنیم. این مسیریابی تعریف شده در فایل Scripts\App\router.js نیز باید تو در تو باشد؛ زیرا قسمت ثبت نظر (new-comment) دقیقا داخل همان صفحه‌ی نمایش یک مطلب ظاهر می‌شود:

```
Blogger.Router.map(function () {
    this.resource('posts', { path: '/' });
    this.resource('about');
    this.resource('contact', function () {
        this.resource('email');
        this.resource('phone');
    });
    this.resource('recent-comments');
    this.resource('post', { path: 'posts/:post_id' }, function () {
        this.resource('new-comment');
    });
    this.resource('new-post');
});
```

لینک آن‌را نیز به انتهای فایل Scripts\Templates\post.hbs اضافه می‌کنیم. از این جهت که این لینک به مدل جاری اشاره می‌کند، با استفاده از متغیر this، مدل جاری را به عنوان مدل مورد استفاده مشخص خواهیم کرد:

```
<h2>{{title}}</h2>
{{#if isEditing}}
<form>
    <div class="form-group">
        <label for="title">Title</label>
        {{input value=title id="title" class="form-control"}}
    </div>
    <div class="form-group">
        <label for="body">Body</label>
        {{textarea value=body id="body" class="form-control" rows="5"}}
    </div>
    <button class="btn btn-primary" {{action 'save' }}>Save</button>
</form>
{{else}}
<p>{{body}}</p>
<button class="btn btn-primary" {{action 'edit' }}>Edit</button>
<button class="btn btn-danger" {{action 'delete' }}>Delete</button>
{{/if}}

<h2>Comments</h2>
{{#each comment in comments}}
<p>
    {{comment.text}}
</p>
{{/each}}

<p>{{#link-to 'new-comment' this class="btn btn-success"}}New comment{{/link-to}}</p>
{{outlet}}
```

پس از تکمیل روابط مدل‌ها، قالب Scripts\Templates\post.hbs را جهت استفاده از این خواص به روز خواهیم کرد. در تغییرات جدید، قسمت <h2>Comments</h2> به انتهای صفحه اضافه شده‌است. سپس حلقه‌ای بر روی خاصیت جدید comments تشکیل

شده و مقدار خاصیت text هر آیتم نمایش داده می‌شود.

در انتهای قالب نیز یک {{outlet}} اضافه شده‌است. کار آن نمایش قالب ارسال یک نظر جدید، پس از کلیک بر روی لینک New Comment می‌باشد. این قالب را با افزودن فایل Scripts\Templates\new-comment.hbs با محتوای ذیل ایجاد خواهیم کرد:

```
<h2>New comment</h2>

<form>
  <div class="form-group">
    <label for="text">Your thoughts:</label>
    {{textarea value=text id="text" class="form-control" rows="5"}}
  </div>

  <button class="btn btn-primary" {{action "save"}}>Add your comment</button>
</form>
```

سپس نام این قالب را به template loader فایل index.html نیز اضافه می‌کنیم؛ تا در ابتدای بارگذاری برنامه شناسایی شده و استفاده شود:

```
<script type="text/javascript">
    EmberHandlebarsLoader.loadTemplates([
       'posts', 'about', 'application', 'contact', 'email', 'phone',
       'recent-comments', 'post', 'new-post', 'new-comment'
    ]);
</script>
```

این قالب به خاصیت text یک comment متصل بوده و همچنین اکشن جدیدی به نام save دارد. بنابراین برای مدیریت اکشن save، نیاز به کنترلری متناظر خواهد بود. به همین جهت فایل جدید Scripts\Controllers\new-comment.js را با محتوای ذیل ایجاد کنید:

```
Blogger.NewCommentController = Ember.ObjectController.extend({
    needs: ['post'],
    actions: {
        save: function () {
            var comment = this.store.createRecord('comment', {
                text: this.get('text')
            });
            comment.save();

            var post = this.get('controllers.post.model');
            post.get('comments').pushObject(comment);
            post.save();

            this.transitionToRoute('post', post.id);
        }
    }
});
```

و مدخل تعریف آن‌را نیز به صفحه‌ی index.html اضافه می‌کنیم:

```
<script src="Scripts/Controllers/new-comment.js" type="text/javascript"></script>
```

قسمت ذخیره سازی comment جدید با ذخیره سازی یک post جدید که پیشتر بررسی کردیم، تفاوتی ندارد. از متد this.store.createRecord جهت معرفی وهله‌ای جدید از comment استفاده و سپس متد save آن، برای ثبت نهایی فراخوانی شده‌است.

در ادامه باید این نظر جدید را به post متناظر با آن مرتبط کنیم. برای اینکار نیاز است تا به مدل کنترلر post دسترسی داشته باشیم. به همین جهت خاصیت needs را به تعاریف کنترلر جاری به همراه نام کنترلر مورد نیاز، اضافه کرده‌ایم. به این ترتیب می‌توان توسط متد this.get و پارامتر controllers.post.model در کنترلر NewComment به اطلاعات کنترلر post دسترسی یافت. سپس خاصیت comments شیء post جاری را یافته و مقدار آن‌را به comment جدیدی که ثبت کردیم، تنظیم می‌کنیم. در ادامه با فراخوانی متد save، کار تنظیم ارتباطات یک مطلب و نظرهای جدید آن به پایان می‌رسد.

در آخر با فراخوانی متد transitionToRoute به مطلبی که نظر جدیدی برای آن ارسال شده‌است باز می‌گردیم.

همانطور که در این تصویر نیز مشاهده می‌کنید، اطلاعات ذخیره شده در local storage را توسط افزونه‌ی Ember Inspector نیز می‌توان مشاهده کرد.

افزودن دکمه‌ی حذف به لیست نظرات ارسالی

برای افزودن دکمه‌ی حذف، به قالب Scripts\Templates\post.hbs مراجعه کرده و قسمتی را که لیست نظرات را نمایش می‌دهد، به نحو ذیل تکمیل می‌کنیم:

```
{{#each comment in comments}}
```

```
<p>
    {{comment.text}}
    <button class="btn btn-xs btn-danger" {{action 'delete' }}>delete</button>
</p>
{{/each}}
```

همچنین برای مدیریت اکشن جدید delete، کنترلر جدید comment را در فایل Scripts\Controllers\comment.js اضافه خواهیم کرد.

```
Blogger.CommentController = Ember.ObjectController.extend({
    needs: ['post'],
    actions: {
        delete: function () {
            if (confirm('Do you want to delete this comment?')) {
                var comment = this.get('model');
                comment.deleteRecord();
                comment.save();

                var post = this.get('controllers.post.model');
                post.get('comments').removeObject(comment);
                post.save();
            }
        }
    }
});
```

به همراه تعریف مدخل آن در فایل index.html :

```
<script src="Scripts/Controllers/comment.js" type="text/javascript"></script>
```

در این حالت اگر برنامه را اجرا کنید، پیام «Do you want to delete this **post**» را مشاهده خواهید کرد بجای پیام «Do you want to delete this **comment**». علت اینجا است که قالب post به صورت پیش فرض به کنترلر post متصل است و نه کنترلر comment. برای رفع این مشکل تنها کافی است از itemController به نحو ذیل استفاده کنیم:

```
{{#each comment in comments  itemController="comment"}}
<p>
    {{comment.text}}
    <button class="btn btn-xs btn-danger" {{action 'delete' }}>delete</button>
</p>
{{/each}}
```

به این ترتیب اکشن delete به کنترلر comment ارسال خواهد شد و نه کنترلر پیش فرض post جاری.
در کنترلر Comment روش دیگری را برای حذف یک رکورد مشاهده می‌کنید. می‌توان ابتدا متد deleteRecord را بر روی مدل فراخوانی کرد و سپس آن‌را save نمود تا نهایی شود. همچنین در اینجا نیاز است نظر حذف شده را از سر دیگر رابطه نیز حذف کرد. روش دسترسی به post جاری در این حالت، همانند توضیحات NewCommentController است که پیشتر بحث شد.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS03_04.zip

عنوان: ساخت یک بلاگ ساده با Ember.js، قسمت پنجم
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۰ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, SPA, EmberJS

مقدمات ساخت بلاگ مبتنی بر ember.js در قسمت قبل به پایان رسید. در این قسمت صرفا قصد داریم بجای استفاده از HTML 5 local storage از یک REST web service مانند یک ASP.NET Web API Controller و یا یک ASP.NET MVC Controller استفاده کنیم و اطلاعات نهایی را به سرور ارسال و یا از آن دریافت کنیم.

تنظیم Ember data برای کار با سرور

Ember data به صورت پیش فرض و در پشت صحنه با استفاده از Ajax برای کار با یک REST Web Service طراحی شده‌است و کلیه تبادلات آن نیز با فرمت JSON انجام می‌شود. بنابراین تمام کدهای سمت کاربر قسمت قبل نیز در این حالت کار خواهند کرد. تنها کاری که باید انجام شود، حذف تنظیمات ابتدایی آن برای کار با HTML 5 local storage است.
برای این منظور ابتدا فایل index.html را گشوده و سپس مدخل localstorage_adapter.js را از آن حذف کنید:

```
<!--<script src="Scripts/Libs/localstorage_adapter.js" type="text/javascript"></script>-->
```

همچنین دیگر نیازی به store.js نیز نمی‌باشد:

```
<!--<script src="Scripts/App/store.js" type="text/javascript"></script>-->
```

اکنون برنامه را اجرا کنید، چنین پیام خطایی را مشاهده خواهید کرد:

همانطور که عنوان شد، ember data به صورت پیش فرض با سرور کار می‌کند و در اینجا به صورت خودکار، یک درخواست Get را به آدرس http://localhost:25918/posts جهت دریافت آخرین مطالب ثبت شده، ارسال کرده‌است و چون هنوز وب سرویسی در برنامه تعریف نشده، با خطای 404 و یا یافت نشد، مواجه شده‌است.
این درخواست نیز بر اساس تعاریف موجود در فایل Scripts\Routes\posts.js، به سرور ارسال شده‌است:

```
Blogger.PostsRoute = Ember.Route.extend({
    model: function () {
        return this.store.find('post');
    }
```

```
});
```

Ember data شبیه به یک ORM عمل می‌کند. تنظیمات ابتدایی آن‌را تغییر دهید، بدون نیازی به تغییر در کدهای اصلی برنامه، می‌تواند با یک منبع داده جدید کار کند.

تغییر تنظیمات پیش فرض آغازین Ember data

آدرس درخواستی http://localhost:25918/posts به این معنا است که کلیه درخواست‌ها، به همان آدرس و پورت ریشه‌ی اصلی سایت ارسال می‌شوند. اما اگر یک ASP.NET Web API Controller را تعریف کنیم، نیاز است این درخواست‌ها، برای مثال به آدرس api/posts ارسال شوند؛ بجای /posts.
برای این منظور پوشه‌ی جدید Scripts\Adapters را ایجاد کرده و فایل web_api_adapter.js را با این محتوا به آن اضافه کنید:

```
DS.RESTAdapter.reopen({
      namespace: 'api'
});
```

سپس تعریف مدخل آن‌را نیز به فایل index.html اضافه نمائید:

```
<script src="Scripts/Adapters/web_api_adapter.js" type="text/javascript"></script>
```

تعریف فضای نام در اینجا سبب خواهد شد تا درخواست‌های جدید به آدرس api/posts ارسال شوند.

تغییر تنظیمات پیش فرض ASP.NET Web API

در سمت سرور، بنابر اصول نامگذاری خواص، نام‌ها با حروف بزرگ شروع می‌شوند:

```
namespace EmberJS03.Models
{
    public class Post
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }
    }
}
```

اما در سمت کاربر و کدهای اسکریپتی، عکس آن صادق است. به همین جهت نیاز است که
CamelCasePropertyNamesContractResolver را در JSON.NET تنظیم کرد تا به صورت خودکار اطلاعات ارسالی به کلاینت‌ها را به صورت camel case تولید کند:

```
using System;
using System.Web.Http;
using System.Web.Routing;
using Newtonsoft.Json.Serialization;

namespace EmberJS03
{
    public class Global : System.Web.HttpApplication
    {

        protected void Application_Start(object sender, EventArgs e)
        {
            RouteTable.Routes.MapHttpRoute(
               name: "DefaultApi",
               routeTemplate: "api/{controller}/{id}",
               defaults: new { id = RouteParameter.Optional }
               );

            var settings =
GlobalConfiguration.Configuration.Formatters.JsonFormatter.SerializerSettings;
            settings.ContractResolver = new CamelCasePropertyNamesContractResolver();
```

```
        }
    }
}
```

نحوه‌ی صحیح بازگشت اطلاعات از یک ASP.NET Web API جهت استفاده در Ember data

با تنظیمات فوق، اگر کنترلر جدیدی را به صورت ذیل جهت بازگشت لیست مطالب تهیه کنیم:

```
namespace EmberJS03.Controllers
{
    public class PostsController : ApiController
    {
        public IEnumerable<Post> Get()
        {
            return DataSource.PostsList;
        }
    }
}
```

با یک چنین خطایی در سمت کاربر مواجه خواهیم شد:

```
WARNING: Encountered "0" in payload, but no model was found for model name "0" (resolved model name
using DS.RESTSerializer.typeForRoot("0"))
```

این خطا از آنجا ناشی می‌شود که Ember data، اطلاعات دریافتی از سرور را بر اساس قرارداد JSON API دریافت می‌کند. برای حل این مشکل راه‌حل‌های زیادی مطرح شده‌اند که تعدادی از آن‌ها را در لینک‌های زیر می‌توانید مطالعه کنید:

http://jsonapi.codeplex.com

https://github.com/xqiu/MVCSPAWithEmberjs

https://github.com/rmichela/EmberDataAdapter

https://github.com/MilkyWayJoe/Ember-WebAPI-Adapter

http://blog.yodersolutions.com/using-ember-data-with-asp-net-web-api

http://emadibrahim.com/2014/04/09/emberjs-and-asp-net-web-api-and-json-serialization

و خلاصه‌ی آن‌ها به این صورت است:

خروجی JSON تولیدی توسط ASP.NET Web API چنین شکلی را دارد:

```
[
  {
    Id: 1,
    Title: 'First Post'
  }, {
    Id: 2,
    Title: 'Second Post'
  }
]
```

اما Ember data نیاز به یک چنین خروجی دارد:

```
{
  posts: [{
    id: 1,
    title: 'First Post'
  }, {
    id: 2,
    title: 'Second Post'
  }]
}
```

به عبارتی آرایه‌ی مطالب را از ریشه‌ی posts باید دریافت کند (مطابق فرمت JSON API). برای انجام اینکار یا از لینک‌های معرفی شده استفاده کنید و یا راه حل ساده‌ی ذیل هم پاسخگو است:

```
using System.Web.Http;
using EmberJS03.Models;

namespace EmberJS03.Controllers
{
    public class PostsController : ApiController
    {
        public object Get()
        {
            return new { posts = DataSource.PostsList };
        }
    }
}
```

در اینجا ریشه‌ی posts را توسط یک anonymous object ایجاد کرده‌ایم.
اکنون اگر برنامه را اجرا کنید، در صفحه‌ی اول آن، لیست عناوین مطالب را مشاهده خواهید کرد.

تاثیر قرارداد JSON API در حین ارسال اطلاعات به سرور توسط Ember data

در تکمیل کنترلرهای Web API مورد نیاز (کنترلرهای مطالب و نظرات)، نیاز به متدهای Post، Update و Delete هم خواهد بود.
دقیقا فرامین ارسالی توسط Ember data توسط همین HTTP Verbs به سمت سرور ارسال می‌شوند. در این حالت اگر متد Post کنترلر نظرات را به این شکل طراحی کنیم:

```
public HttpResponseMessage Post(Comment comment)
```

کار نخواهد کرد؛ چون مطابق فرمت JSON API ارسالی توسط Ember data، یک چنین شیء JSON ایی را دریافت خواهیم کرد:

```
{"comment":{"text":"data...","post":"3"}}
```

بنابراین Ember data چه در حین دریافت اطلاعات از سرور و چه در زمان ارسال اطلاعات به آن، اشیاء جاوا اسکریپتی را در یک ریشه‌ی هم نام آن شیء قرار می‌دهد.
برای پردازش آن، یا باید از راه حل‌های ثالث مطرح شده در ابتدای بحث استفاده کنید و یا می‌توان مطابق کدهای ذیل، کل اطلاعات JSON ارسالی را توسط کتابخانه‌ی JSON.NET نیز پردازش کرد:

```
namespace EmberJS03.Controllers
{
    public class CommentsController : ApiController
    {
        public HttpResponseMessage Post(HttpRequestMessage requestMessage)
        {
            var jsonContent = requestMessage.Content.ReadAsStringAsync().Result;
            // {"comment":{"text":"data...","post":"3"}}
            var jObj = JObject.Parse(jsonContent);
            var comment = jObj.SelectToken("comment", false).ToObject<Comment>();


            var id = 1;
            var lastItem = DataSource.CommentsList.LastOrDefault();
            if (lastItem != null)
            {
                id = lastItem.Id + 1;
            }
            comment.Id = id;
            DataSource.CommentsList.Add(comment);

            // ارسال آی دی با فرمت خاص مهم است
            return Request.CreateResponse(HttpStatusCode.Created, new { comment = comment });
        }
    }
```

```
}
```

در اینجا توسط requestMessage به محتوای ارسال شده‌ی به سرور که همان شیء JSON ارسالی است، دسترسی خواهیم داشت. سپس متد JObject.Parse، آن‌را به صورت عمومی تبدیل به یک شیء JSON می‌کند و نهایتا با استفاده از متد SelectToken آن می‌توان ریشه‌ی comment و یا در کنترلر مطالب، ریشه‌ی post را انتخاب و سپس تبدیل به شیء Comment و یا Post کرد.
همچنین فرمت return نهایی هم مهم است. در این حالت خروجی ارسالی به سمت کاربر، باید مجددا با فرمت JSON API باشد؛ یعنی باید comment اصلاح شده را به همراه ریشه‌ی comment ارسال کرد. در اینجا نیز anonymous object تهیه شده، چنین کاری را انجام می‌دهد.

Lazy loading در Ember data

تا اینجا اگر برنامه را اجرا کنید، لیست مطالب صفحه‌ی اول را مشاهده خواهید کرد، اما لیست نظرات آن‌ها را خیر؛ از این جهت که ضرورتی نداشت تا در بار اول ارسال لیست مطالب به سمت کاربر، تمام نظرات متناظر با آن‌ها را هم ارسال کرد. بهتر است زمانیکه کاربر یک مطلب خاص را مشاهده می‌کند، نظرات خاص آن‌را به سمت کاربر ارسال کنیم.
در تعاریف سمت کاربر Ember data، پارامتر دوم رابطه‌ی hasMany که با async:true مشخص شده‌است، دقیقا معنای lazy loading را دارد.

```
Blogger.Post = DS.Model.extend({
   title: DS.attr(),
   body: DS.attr(),
   comments: DS.hasMany('comment', { async: true } /* lazy loading */)
});
```

در سمت سرور، دو راه برای فعال سازی این lazy loading تعریف شده در سمت کاربر وجود دارد:
الف) Idهای نظرات هر مطلب را به صورت یک آرایه، در بار اول ارسال لیست نظرات به سمت کاربر، تهیه و ارسال کنیم:

```
namespace EmberJS03.Models
{
    public class Post
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }

        // lazy loading via an array of IDs
        public int[] Comments { set; get; }
    }
}
```

در اینجا خاصیت Comments، تنها کافی است لیستی از Idهای نظرات مرتبط با مطلب جاری باشد. در این حالت در سمت کاربر اگر مطلب خاصی جهت مشاهده‌ی جزئیات آن انتخاب شود، به ازای هر Id ذکر شده، یکبار دستور Get صادر خواهد شد.
ب) این روش به علت تعداد رفت و برگشت بیش از حد به سرور، کارآیی آنچنانی ندارد. بهتر است جهت مشاهده‌ی جزئیات یک مطلب، تنها یکبار درخواست Get کلیه نظرات آن صادر شود.
برای اینکار باید مدل برنامه را به شکل زیر تغییر دهیم:

```
namespace EmberJS03.Models
{
    public class Post
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }

        // load related models via URLs instead of an array of IDs
        // ref. https://github.com/emberjs/data/pull/1371
        public object Links { set; get; }

        public Post()
        {
```

```
            Links = new { comments = "comments" }; // api/posts/id/comments
        }
    }
}
```

در اینجا یک خاصیت جدید به نام Links ارائه شده‌است. نام Links در Ember data استاندارد است و از آن برای دریافت کلیه اطلاعات لینک شده‌ی به یک مطلب استفاده می‌شود. با تعریف این خاصیت به نحوی که ملاحظه می‌کنید، اینبار Ember data تنها یکبار درخواست ویژه‌ای را با فرمت api/posts/id/comments، به سمت سرور ارسال می‌کند. برای مدیریت آن، قالب مسیریابی پیش فرض api/{controller}/{id} را می‌توان به صورت api/{controller}/{id}/{name} اصلاح کرد:

```
namespace EmberJS03
{
    public class Global : System.Web.HttpApplication
    {

        protected void Application_Start(object sender, EventArgs e)
        {
            RouteTable.Routes.MapHttpRoute(
               name: "DefaultApi",
               routeTemplate: "api/{controller}/{id}/{name}",
               defaults: new { id = RouteParameter.Optional, name = RouteParameter.Optional }
               );

            var settings =
GlobalConfiguration.Configuration.Formatters.JsonFormatter.SerializerSettings;
            settings.ContractResolver = new CamelCasePropertyNamesContractResolver();
        }
    }
}
```

اکنون دیگر درخواست جدید api/posts/3/comments با پیام 404 یا یافت نشد مواجه نمی‌شود.
در این حالت در طی یک درخواست می‌توان کلیه نظرات را به سمت کاربر ارسال کرد. در اینجا نیز ذکر ریشه‌ی comments همانند ریشه posts، الزامی است:

```
namespace EmberJS03.Controllers
{
    public class PostsController : ApiController
    {
        //GET api/posts/id
        public object Get(int id)
        {
            return
                new
                {
                    posts = DataSource.PostsList.FirstOrDefault(post => post.Id == id),
                    comments = DataSource.CommentsList.Where(comment => comment.Post == id).ToList()
                };
        }
    }
}
```

پردازش‌های async و متد transitionToRoute در Ember.js

اگر متد حذف مطالب را نیز به کنترلر Posts اضافه کنیم:

```
namespace EmberJS03.Controllers
{
    public class PostsController : ApiController
    {
        public HttpResponseMessage Delete(int id)
        {
            var item = DataSource.PostsList.FirstOrDefault(x => x.Id == id);
            if (item == null)
                return Request.CreateResponse(HttpStatusCode.NotFound);
```

```
            DataSource.PostsList.Remove(item);

            //حذف کامنت‌های مرتبط
            var relatedComments = DataSource.CommentsList.Where(comment => comment.Post ==
id).ToList();
            relatedComments.ForEach(comment => DataSource.CommentsList.Remove(comment));

            return Request.CreateResponse(HttpStatusCode.OK, new { post = item });
        }
    }
}
```

قسمت سمت سرور کار تکمیل شده‌است. اما در سمت کاربر، چنین خطایی را دریافت خواهیم کرد:

```
Attempted to handle event `pushedData` on  while in state root.deleted.inFlight.
```

منظور از حالت inFlight در اینجا این است که هنوز کار حذف سمت سرور تمام نشده‌است که متد transitionToRoute را صادر کرده‌اید. برای اصلاح آن، فایل Scripts\Controllers\post.js را باز کرده و پس از متد destroyRecord، متد then را قرار دهید:

```
Blogger.PostController = Ember.ObjectController.extend({
    isEditing: false,
    actions: {
        edit: function () {
            this.set('isEditing', true);
        },
        save: function () {
            var post = this.get('model');
            post.save();

            this.set('isEditing', false);
        },
        delete: function () {
            if (confirm('Do you want to delete this post?')) {
                var thisController = this;
                var post = this.get('model');
                post.destroyRecord().then(function () {
                    thisController.transitionToRoute('posts');
                });
            }
        }
    }
});
```

به این ترتیب پس از پایان عملیات حذف سمت سرور، قسمت then اجرا خواهد شد . همچنین باید دقت داشت که this اشاره کننده به کنترلر جاری را باید پیش از فراخوانی then ذخیره و استفاده کرد.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

EmberJS03_05.zip