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:

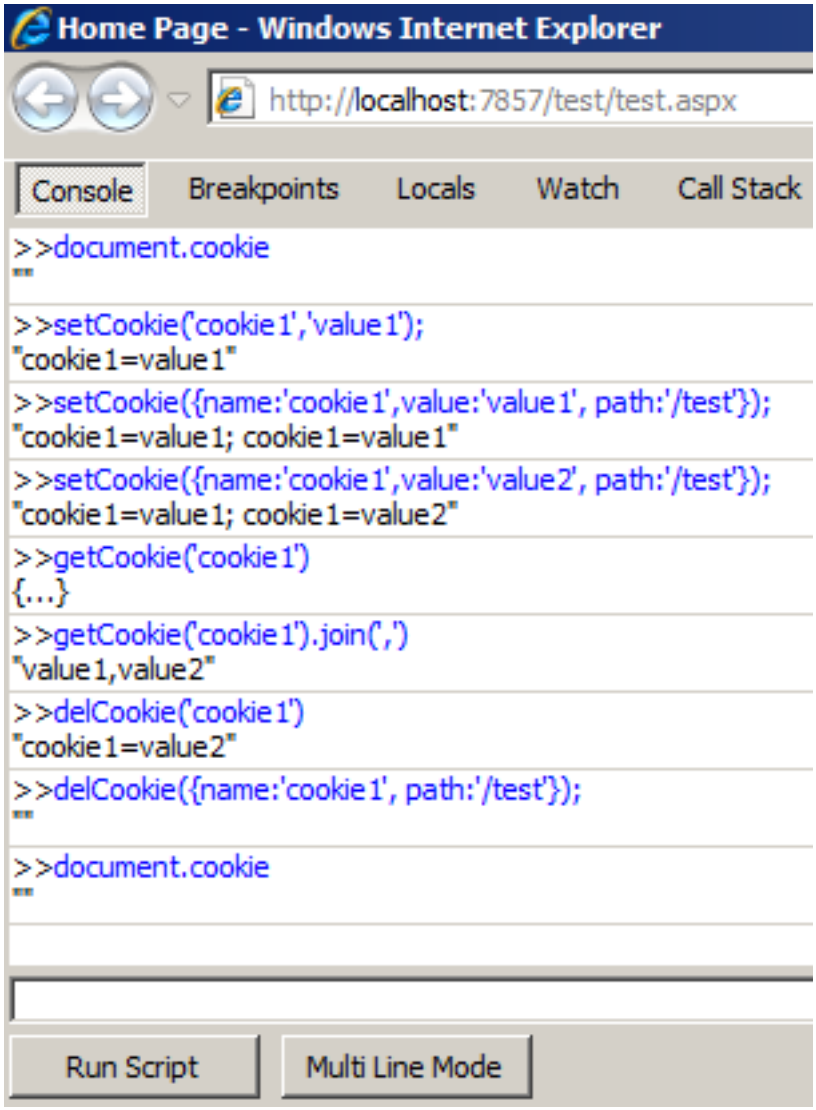
```
cookie.shift(1);
return unescape(cookie.join('=').trim());
```

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