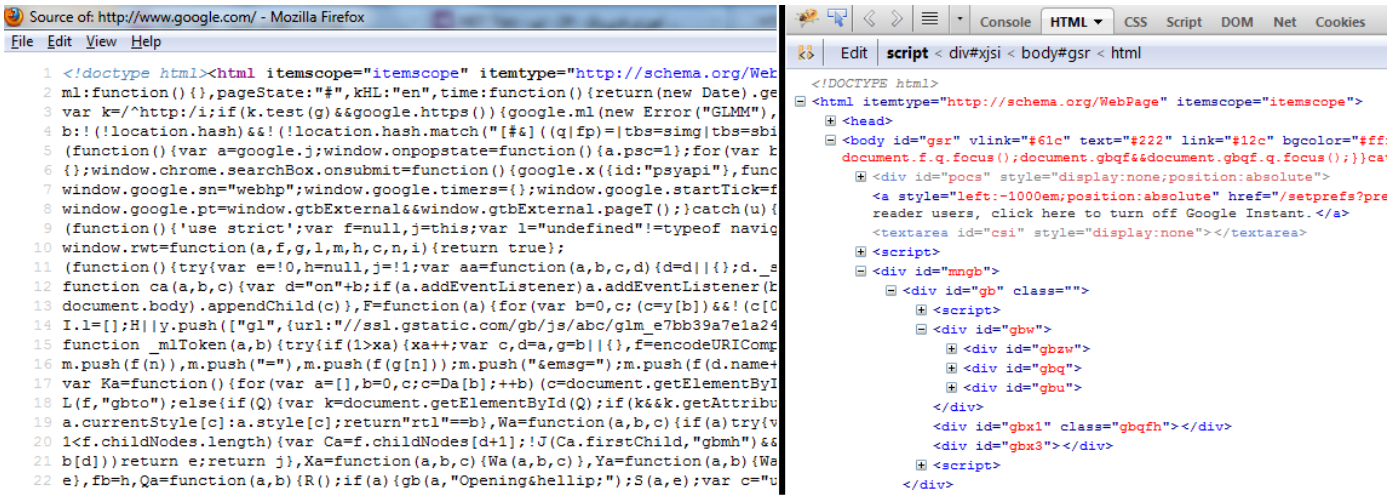
عنوان: **آموزش فایرباگ - #5 - HTML Panel**
نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۸/۲۳ ۱۸:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, FireBug, CSS, XPATH, HTML

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 و 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.

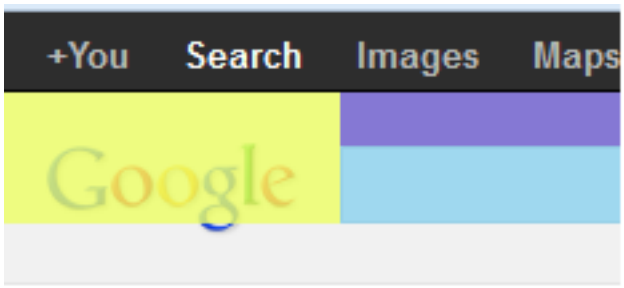
ویرایش HTML

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.

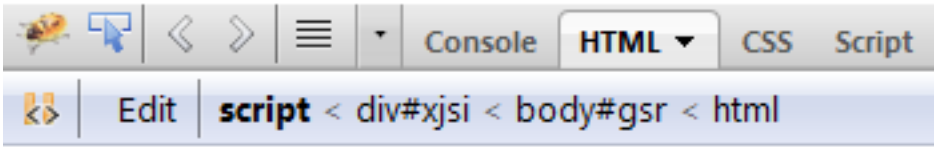
Node View

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 **Left Arrow و Right 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 بروزرسانی 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 (

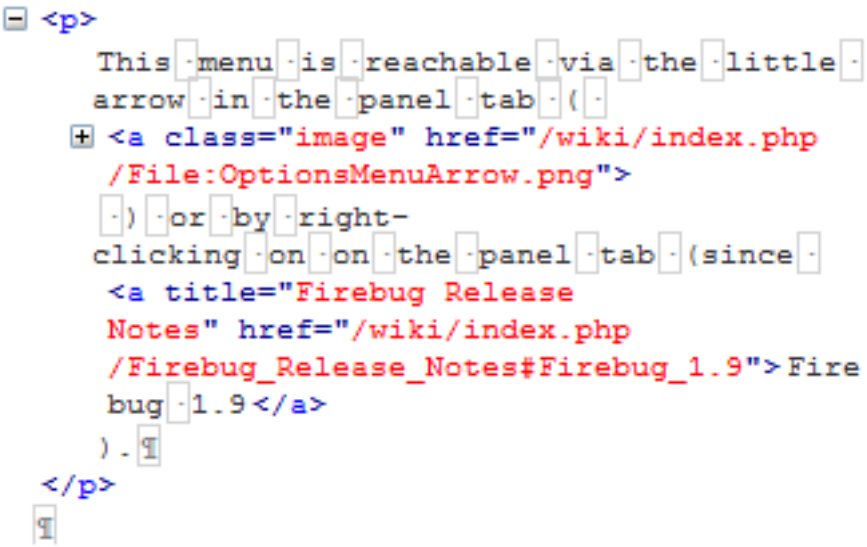
) یا راست کلیک کردن بروی تب ظاهر می‌شود.

Show Full Text

در صورت فعال بودن ، متون بصورت کامل نمایش داده می‌شوند ، در غیراینصورت بصورت خلاصه شده نمایش داده می‌شوند.

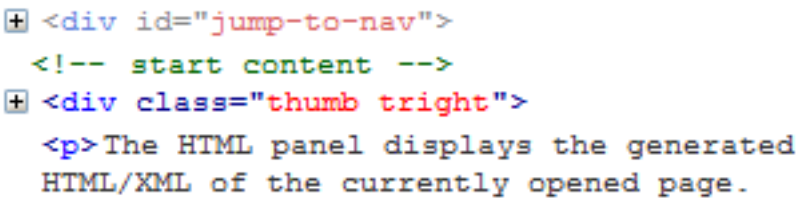
Show White Space

در صورت فعال بودن ، فضاهای خالی ، کرکترهای خط جدید و ... را نمایش می‌دهد.



Show Comments

در صورت فعال بودن ، کامنت‌ها را هم نمایش می‌دهد



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 بافته حرکت می‌کند.

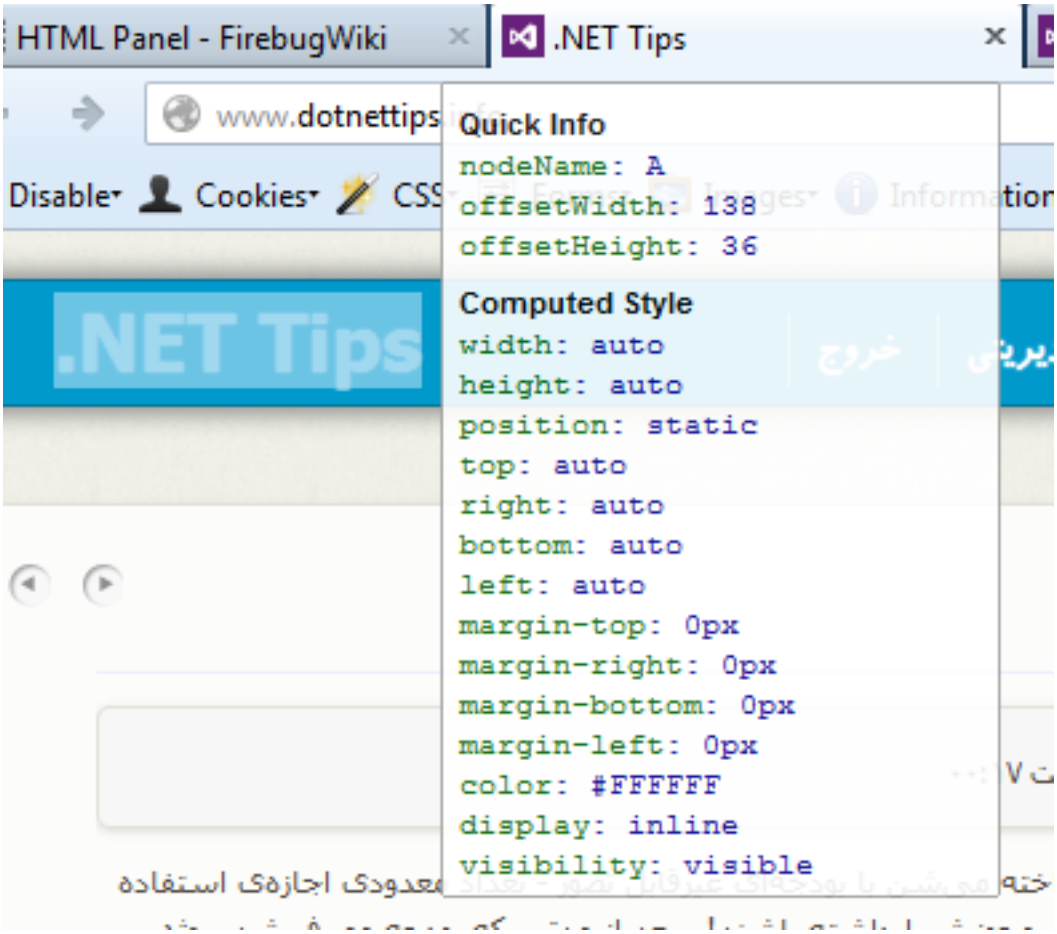
(فعال بودن این گزینه هنگام بررسی سایت هایی که اسلایدر یا سیستم هایی مشابه دارند ، باعث میشه که نتوانید بروی قسمت‌های سایت تمرکز کنید و مدام اسکرول به قسمت تغییرات منتقل می‌شود.)

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.

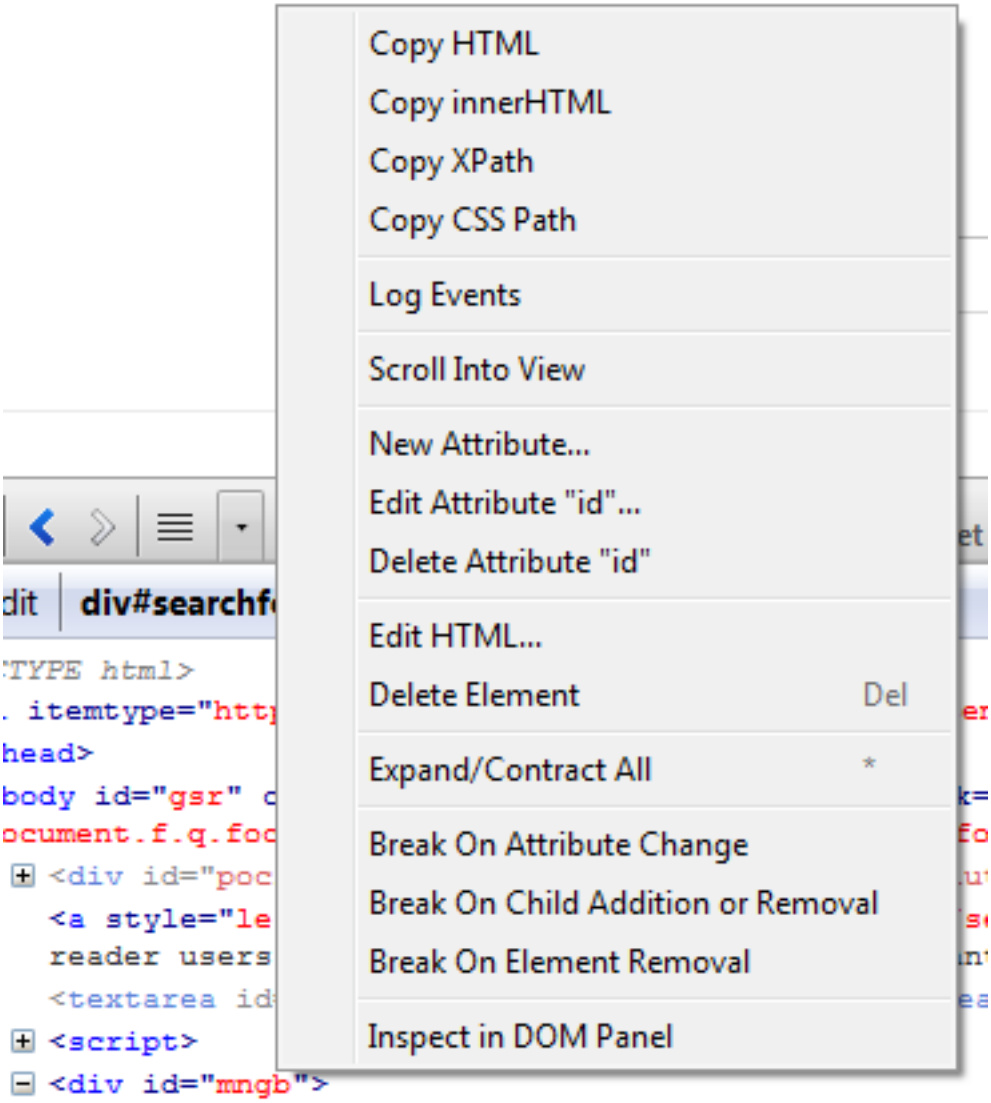
Show Quick Info Box

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.

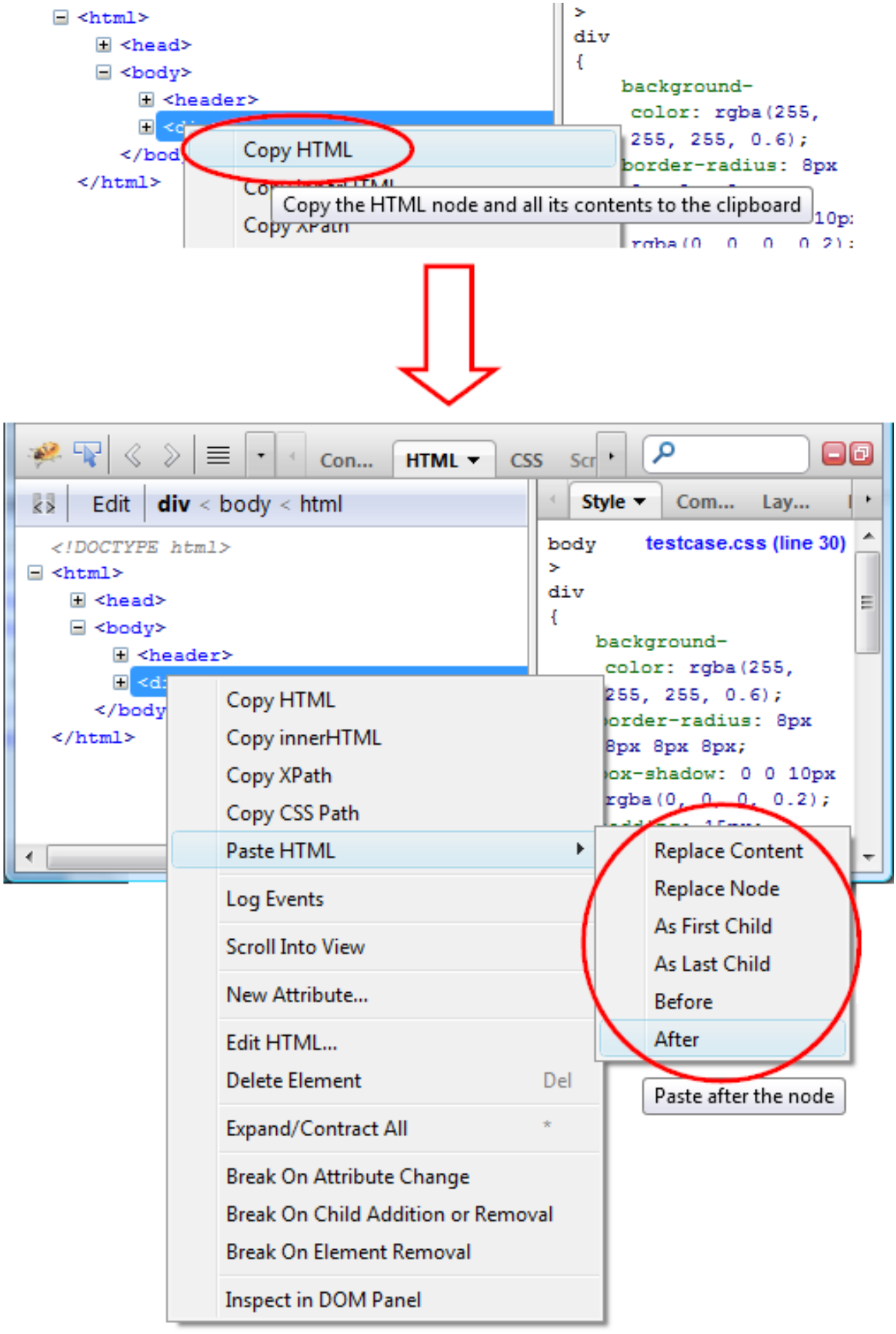
Inspect in DOM Tab
تگ فعلی را در پنل DOM ، برای بررسی باز می‌کند.

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.


<html>
<head>
<body>
<header>
</body>
</html>
Copy HTML
Copy the HTML node and all its contents to the clipboard
Copy XPath
Con...
HTML
CSS
Edit
div < body < html
<!DOCTYPE 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: **آموزش فایرباگ - #6 - HTML Panel - Side Panels**
نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۸/۲۵ ۰:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: FireBug, CSS, HTML

در قسمت قبل توضیحاتی در مورد تب HTML ارائه کردیم.
در این قسمت توضیحات کاملی در مورد پنل‌های جانبی داخل پنل HTML می‌دهیم.

Side Panels

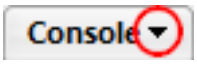
در پنل HTML درکنار ارائه امکاناتی برای مشاهده و کار با تگ‌های صفحه ، اطلاعات و امکانات دیگری هم برای تگ انتخاب شده در قسمت NodeView وجود دارد.
این امکانات در پنل هایی که سمت راست پنل اصلی قرار دارند گنجانده شده است که به ترتیب برای نمایش و ویرایش استایل‌ها ، مشاهده استایل‌های محاسبه شده ، مشاهده Layout یا آرایش و نمایش اطلاعات DOM تگ انتخاب شده در NodeView هستند.

Style - 1

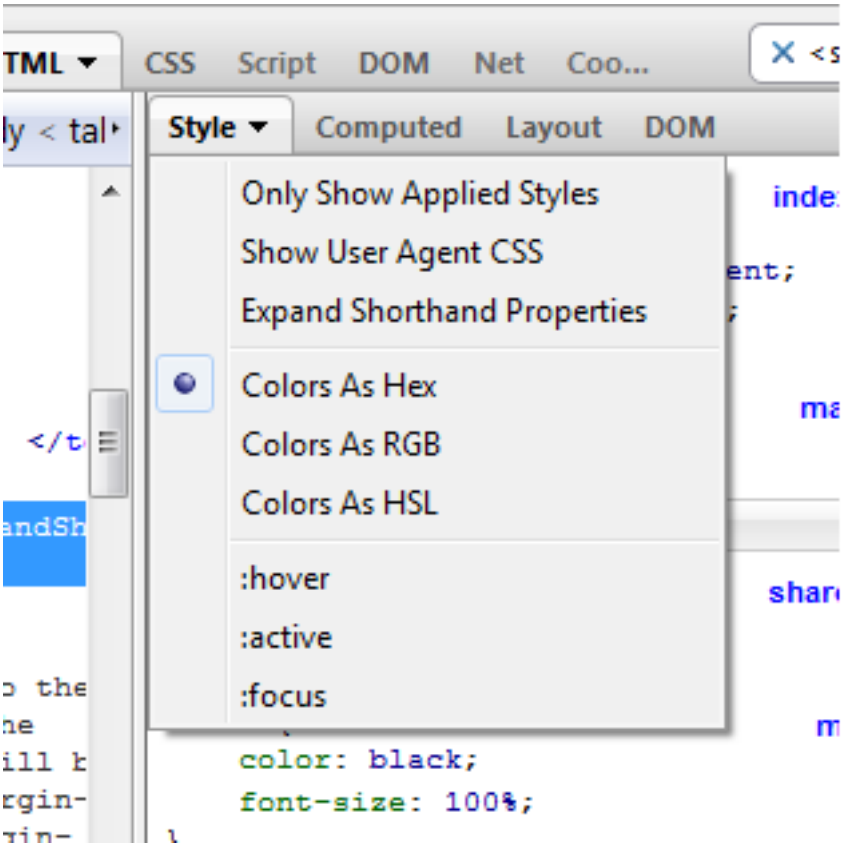
در این تب استایل هایی که در حال حاظر بروی تگ انتخاب شده اعمال شده اند ، نمایش داده می‌شود.
در صورتی که موس را بروی مقادیر استایل هایی که جلوه‌ی بصری دارند بگیرید ، یک پاپ‌آپ کوچک نمایان می‌شود که مقدار را نمایش می‌دهد.

Options Menu

هر تب یا پنل در فایرباگ دارای یک سری تنظیمات است که Options Menu نام دارد. تب Style 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.



Only Show Applied Styles

در صورت انتخاب ، فقط استایل هایی که اعمال شده اند نمایش داده می‌شوند. (استایل‌های Overwrite شده نمایش داده نمی‌شوند.)

(این گزینه قابلیت خوبی است ، اما چندبار برای بنده پیش آمده که این مورد به اشتباه استایلی که اعمال شده بود را هم Overwrite شده در نظر گرفته بود. پس در هین طراحی استایل و کار با CSS اگر احیانا یکی از استایل هایتان وجود نداشت و از وجود آن اطمینان داشتید ، غیرفعال کردن این گزینه را امتحان کنید.)

Show User Agent CSS

با فعال کردن این گزینه ، استایل هایی که توسط مرورگر اعمال شده اند هم نمایش داده می‌شوند.

Expand Shorthand Properties

با فعال کردن این گزینه ، استایل هایی که بصورت کوتاه شده تعریف شده اند را بصورت گسترده و باز شده نمایش می‌دهد.

برای مثال ، دستور margin را بصورت margin-left , margin-bottom , margin-right , margin-top نمایش می‌دهد.

سه گزینه ی Colors As RGB ، Colors As Hex و Colors As HSL تعیین کننده‌ی فرمت نمایش رنگ‌ها هستند.

سه گزینه ی :hover ، active و focus: هم برای تعیین کلاس کاذب برای تگ جاری کاربرد دارند.

برای مثال شما می‌خواهید استایلی که یک لینک زمان موس برویش قرار دارد را بررسی کنید ، لینک را در NodeView انتخاب می‌کنید و سپس از گزینه‌ی hover: را فعال می‌کنید.

Panel

Element styles

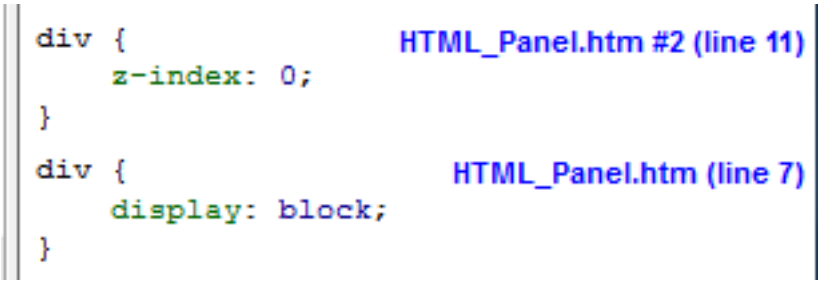
استایل هایی که بصورت inline (در خود تگ) تعریف شده اند هم در این قسمت نمایش داده می‌شود و نام rule آن element.style است.

Source Links

در بالا-راست هر بخش ، یک لینک قرار دارد که لینک فایل استایلی است که در همان قسمت وجود دارد و عددی که در پرانتز قرار دارد ، شماره خط استایل در همان فایل است.

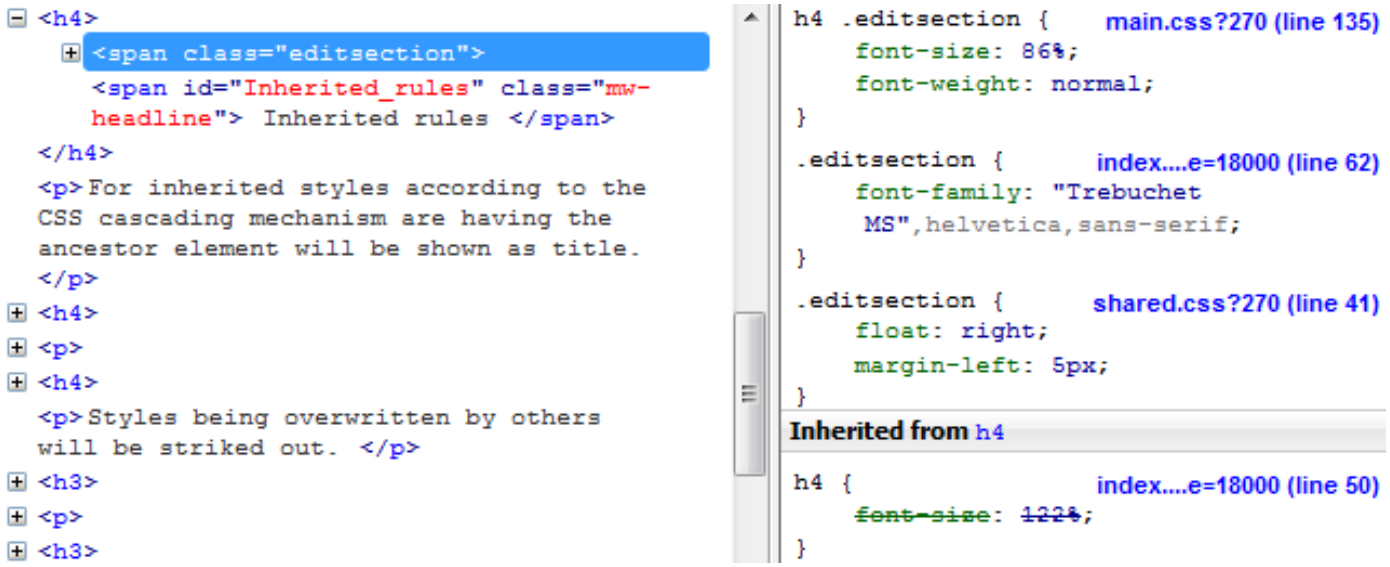
اگر نام فایل با نام صفحه‌ی جاری برابر باشد ، به معنی وجود استایل در تگ <style> در صفحه‌ی جاری است و شماره‌ی بعد از # هم ایندکس تگ <style> است.

(با کلیک بروی لینک فایل ، فایل در خط مورد نظر در پنل CSS نمایش داده می‌شود.)



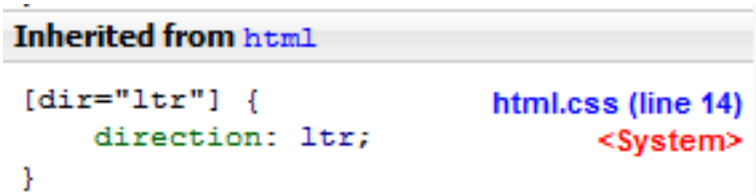
Inherited rules

ruleهای به ارث رسیده هم در قسمت‌های جداگانه به همراه استایل‌های به ارث رسیده نمایش داده می‌شود و تگی والد که استایل‌ها از آن به ارث رسیده اند هم در قسمت عنوان همان استایل‌ها نمایش قرار داده شده است. (با کلیک بروی آن ، در قسمت Nodeview انتخاب می‌شود.)



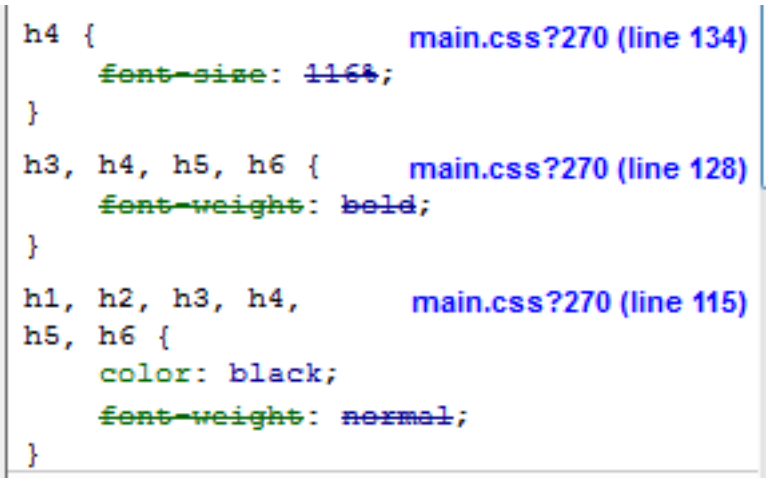
User agent rules

استایل هایی که توسط مرورگر اعمال شده اند (User agent rules) ، با عبارت <System> در زیر لینک منبع استایل ، مشخص شده اند.



Overwritten styles

استایل‌های overwrite شده ، با یک خط برویشان مشخص شده اند.



Inline editing

استایل‌های نمایش داده شده در این پنل را براحتی و با کلیک کردن بروی نام یا مقدار هر یک از دستورات می‌توانید تغییر دهید.
برای نوشتن دستورات و مقادیر آن‌ها می‌توانید از پیشنهادهای فایرباگ هم کمک بگیرید و با دکمه‌های Arrow Up و Arrow Down هم بین مقادیر مجاز حرکت کنید.
دستورات یا مقادیر نا صحیح در هین تایپ ، با رنگ قرمز و مقادیر صحیح با رنگ سبز مشخص می‌شوند.
(این امکان خیلی مفید است ، برای مثال می‌خواهید فونت‌های مختلف را برای یک استایل امتحان کنید ، دستور font-family را می‌نویسید و بعد از زدن Enter ، با دکمه های Arrow Up و Arrow Down در لحظه نتیجه‌ی اعمال فونت‌های مختلف و دردسترس را مشاهده می‌کنید و بهترین را بر می‌گزینید.
یا برای یافتن بهترین مقدار margin ، بعد از دستور margin ، زدن کلید Enter ، وارد کردن یک عدد برای شروع ، می‌توان باز هم با دکمه های Arrow Up و Arrow Down به سرعت تغییر را در صفحه مشاهده کرد.)

Rendered font highlighted

برای دستور font ، فایرباگ هوشمندانه عمل کرده و فونتی که در حال استفاده است را پررنگ می‌کند.
این امکان برای یافتن خطاهای متداول هنگام تعریف فونت‌های غیر سیستمی ، بسیار مفید است.

```
#headermenu li {                    css?v=...HIE5j41 (line
    border-left: 1px solid rgba(255, 255, 255, 0.5
    display: inline-block;
    font: bold 16px "b koodak",arial,sans-serif;
```

Context Menu

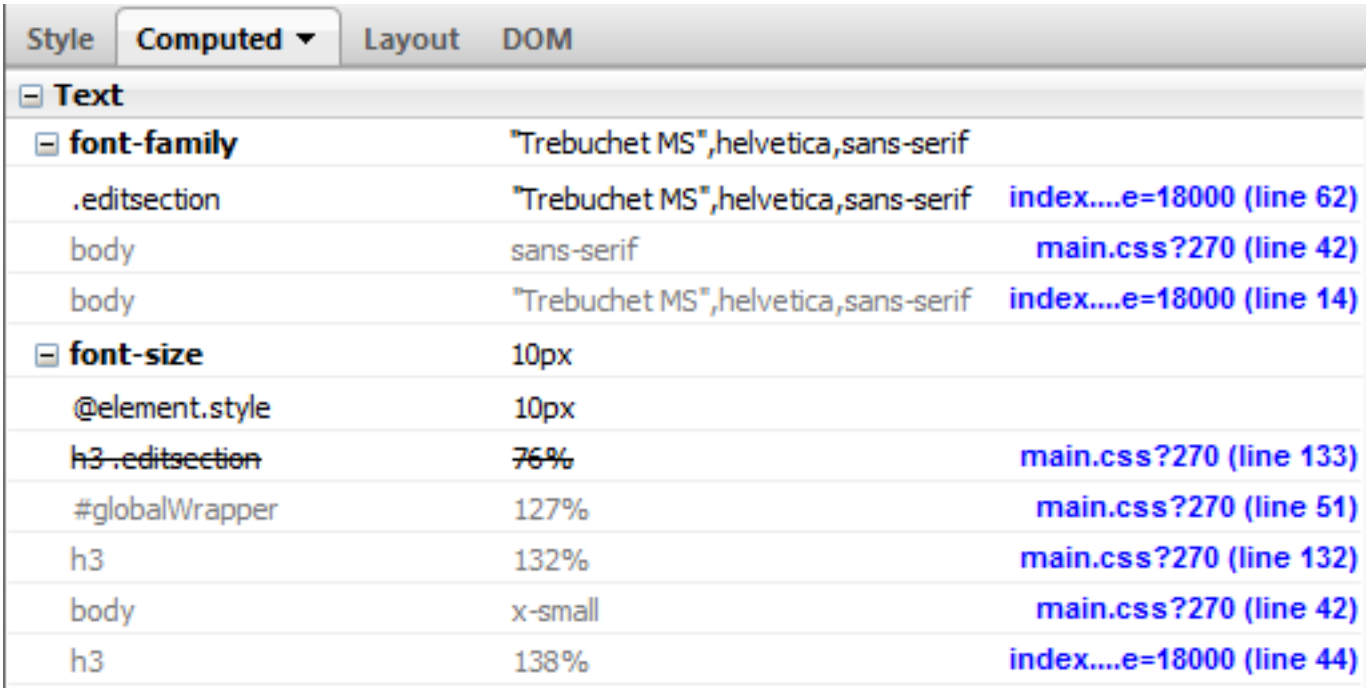
این منو زمانی که در پنل راست کلیک کنید ظاهر می‌شود و نسبت به منطقه (Context)ای که در آن راست کلیک کرده اید ، گزینه‌های متفاوتی را مشاهده خواهید کرد. در جدول زیر ، گزینه‌ها ، Contextشان و توضیح هر گزینه آمده است.

گزینه	Context	توضیحات
Copy Rule Declaration	CSS selector	CSS Rule فعلی را به همراه استایل هایش در clipboard کپی می‌کند
Copy Style Declaration	CSS selector	استایل‌های CSS Rule فعلی را در clipboard کپی می‌کند
Copy Location	source link	آدرس فایل تعریف CSS Rule را در clipboard کپی می‌کند
Open in New Tab	source link	آدرس فایل تعریف CSS Rule را در تب جدید باز می‌کند
Edit Element Style...	everywhere	امکان تعریف استایل‌های درون تگ (inline) را محیا می‌کند
Add Rule...	everywhere	یک Rule جدید ایجاد می‌کند CSS Rule هایی که در حال حاظر وجود دارد را هم پیشنهاد می‌دهد
Delete "<rule name>"	CSS selector	CSS Rule فعلی را حذف می‌کند

گزینه	Context	توضیحات
New Property...	CSS rule	یک استایل جدید به CSS Rule فعلی اضافه می‌کند
Edit "<property name>"...	CSS property	Property فعلی را به حالت ویرایش می‌برد راه دیگر ویرایش Property ، کلیک بروی آن است
Delete "<property name>"	CSS property	Property فعلی را حذف می‌کند
Disable "<property name>"	CSS property	Property فعلی را غیر فعال می‌کند را سریع‌تر ، کلیک کردن در ناحیه‌ی پشت Property ، بروی علامت قرمز رنگ است
Refresh	everywhere	محتویات پنل را بروز می‌کند
Inspect in DOM Panel	CSS rule	CSS Rule فعلی را در پنل DOM برای بررسی باز می‌کند
Inspect in CSS Panel	CSS rule	CSS Rule فعلی را در پنل CSS برای بررسی باز می‌کند
<Default Editor Name>	CSS rule	فایل تعریف استایل‌ها را در ادیتور تعریف شده باز می‌کند (این گزینه در صورت تعریف ادیتور در تنظیمات FireBug نمایش داده خواهد شد)

Computed - 2

دراین تب نتیجه‌ی پردازش استایل‌های ارائه شده توسط کاربر ، برای تگ مشخص شده در قسمت NodeView نمایش داده می‌شود. (مقادیر استایل هایی که در نهایت بروی تگ اعمال شده اند.)



Style Tracing

برای ردیابی استایل‌ها ، استایل‌ها به ترتیب اعمال شدنشان مرتب شده اند و اولین مقدار ، مقداری است که اعمال شده است.
مقادیر Overwrite بصورت خط کشیده شده و استایل‌های Overwrite شده بصورت خاکستری-کمرنگ نمایش داده می‌شوند.
هر استایل هم مانند تب Style ، یک لینک به منبع خود دارد.

Options Menu

Show User Agent CSS
در صورت انتخاب ، فقط استایل هایی که اعمال شده اند نمایش داده می‌شوند.

Sort alphabetically
در صورت انتخاب ، استایل‌ها به ترتیب الفبا ، و درصورت عدم انتخاب بصورت گروه بندی نمایش داده می‌شوند.

Show Mozilla Specific Styles
در صورت انتخاب ، استایل‌های مخصوص Mozilla را نمایش می‌دهد. (استایل هایی با پیشوند -moz-)

سه گزینه‌ی Colors As RGB ، Colors As Hex و Colors As HSL تعیین کننده‌ی فرمت نمایش رنگ‌ها هستند.

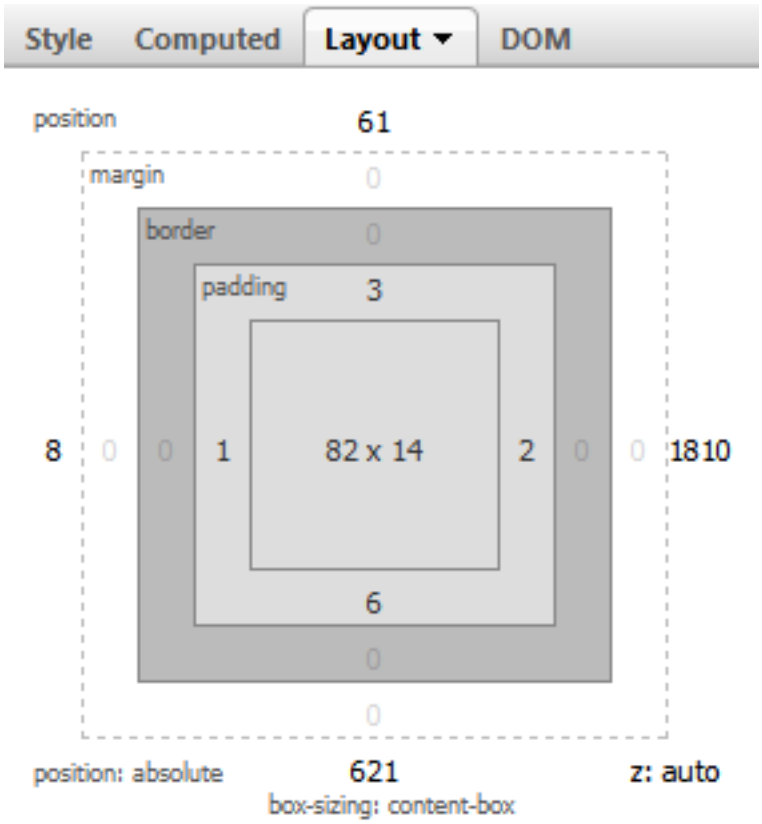
Context Menu

این منو زمانی که در پنل راست کلیک کنید ظاهر می‌شود و نسبت به منطقه (Context)ای که در آن راست کلیک کرده اید ، گزینه‌های متفاوتی را مشاهده خواهید کرد. در جدول زیر ، گزینه‌ها ، Contextشان و توضیح هر گزینه آمده است.

گزینه	Context	توضیحات
Expand All Styles	everywhere	CSS Rule فعلی را به همراه استایل هایش در clipboard کپی می‌کند
Collapse All Styles	everywhere	استایل‌های CSS Rule فعلی را در clipboard کپی می‌کند
Inspect in DOM panel	styles	آدرس فایل تعریف CSS Rule را در تب جدید باز می‌کند
Copy Location	style source link	امکان تعریف استایل‌های درون تگ (inline) را محیا می‌کند
Open in New Tab	style source link	یک Rule جدید ایجاد می‌کند CSS Rule هایی که در حال حاظر وجود دارد را هم پیشنهاد می‌دهد
Inspect in CSS panel	style source link	CSS Rule فعلی را حذف می‌کند
<Default Editor Name>	style source link	فایل تعریف استایل‌ها را در ادیتور تعریف شده باز می‌کند (این گزینه در صورت تعریف ادیتور در تنظیمات FireBug نمایش داده خواهد شد)

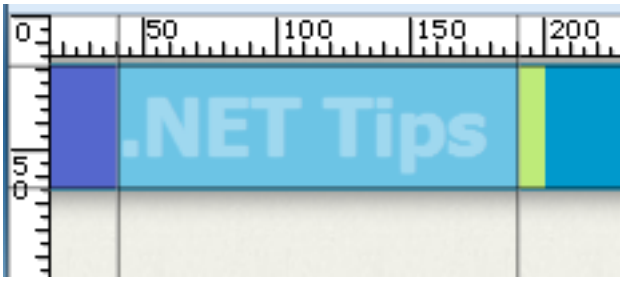
Layout - 3

در این تب ، مقادیر Box Model بصورت بصری نمایش می‌دهد. می‌توان با کلیک کردن بروی هریک از مقادیر ، آن را ویرایش کرد.
(این تغییر بصورت inline در تگ اعمال می‌شود.)
با حرکت موس بروی قسمت‌های مختلف ، می‌توان همان قسمت‌ها را در صفحه بصورت خط کشی شده مشاهده کرد.
(البته ظاهرا در ورژن 1.10.4 که بنده استفاده می‌کنم ، عملیات ویرایش مقادیر به درستی انجام نمی‌شود.)



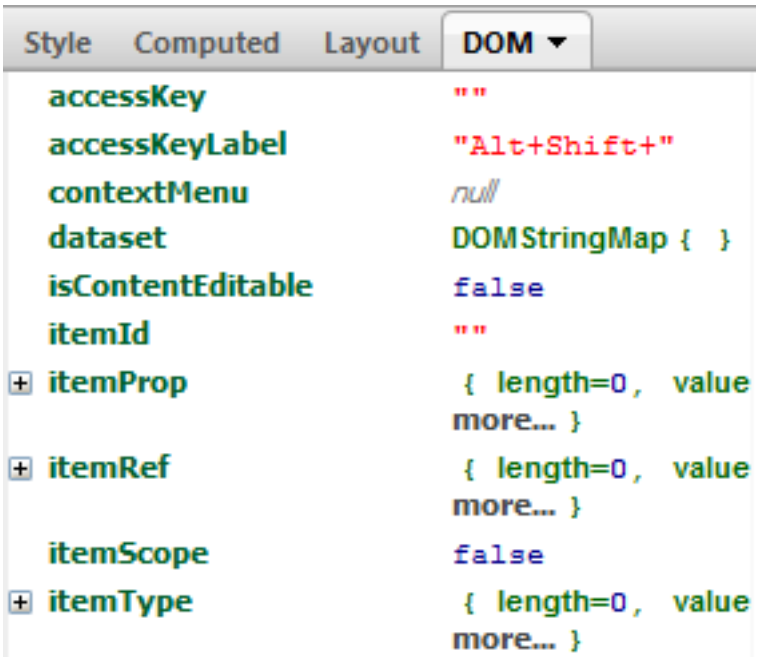
Options Menu

Show Rulers and Guides
در صورت انتخاب ، خطهای راهنما را هنگام حرکت موس بروی اجزای Box Model در صفحه نمایش می‌دهد.



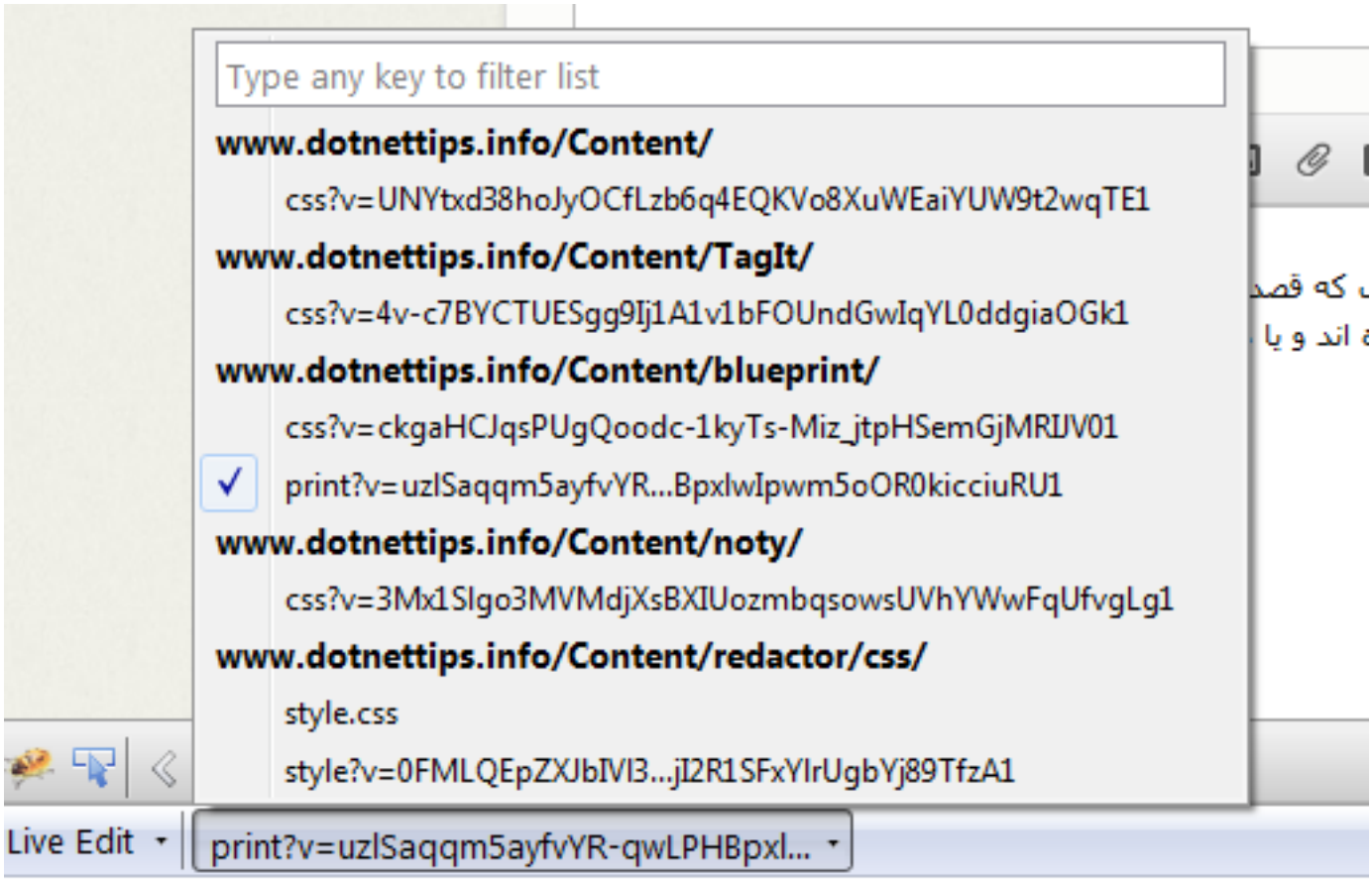
DOM - 4

این پنل اطلاعات DOM تگ جاری را نمایش می‌دهد.
این پنل تمام قابلیت‌های پنل DOM اصلی را دارا می‌باشد.
(در مقالات آینده با تب DOM آشنا خواهیم شد.)



عنوان: **آموزش فایرباگ - #7 - CSS Panel**
نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۱۲/۲۵ ۱۸:۵
آدرس: www.dotnettips.info
برچسب‌ها: FireBug, CSS, HTML

پنل CSS مانند پنل جانبیِ Style که در مقاله‌ی قبل بررسی کردیم است با این تفاوت که امکانات بیشتری برای افرادی که قصد تصریف استایل دارند محیا کرده است.
در این پنل می‌توان به اضافه ، ویرایش و حذف استایل هایی که به صفحه‌ی جاری توسط فایل‌های مختلف اضافه شده اند و یا داخل خود صفحه تعریف شده اند پرداخت.




```
body {
    background: none repeat scroll 0 center transparent;
    color: #000000;
    font-family: Tahoma;
    font-size: 10pt;
    line-height: 1.5;
}

.container {
    background: none repeat scroll 0 center transparent;
}
```

همچنین در این پنل امکان ویرایش یک فایل css بصورت کامل وجود دارد ، به این صورت که می‌توانید تمام محتویات فایل مورد نظر را در یک Text area ویرایش کنید.

مطابق با روالی که در قسمت قبل پیش گرفتیم عمل می‌کنیم و به ترتیب به تشریح ابزار پنل ، خود پنل ، منوی راست کلیک و پنل جانبیِ Elements می‌پردازیم.

Options Menu

با راست کلیک کردن بروی تب CSS و یا کلیک کردن بروی فلش کوچک روی تب CSS قابل دسترس است و امکانات زیر را محیا می‌کند:

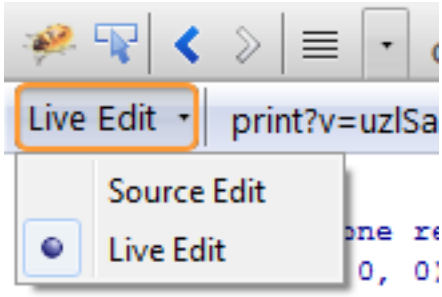
Expand Shorthand Properties : نمایش دستورات css بصورت کامل ، به این معنی که دستوراتی مانند margin که هم بصورت خلاصه و هم بصورت کامل تعریف می‌شوند را بصورت کامل نمایش می‌دهد. مثلا margin را چهار مقدار margin- margin-top , margin-right , margin-bottom , margin-left نمایش می‌دهد.

Color As Hex , Color As RGB , Color As HSL : با انتخاب یکی از سه مقدار ذکر شده ، فرمت نمایش رنگ‌ها به حالت انتخاب شده تغییر می‌کند.
مثلا دستور color : #000000 به این صورت نمایش داده می‌شود : color : rgb(0, 0, 0)

Refresh : این گزینه محتویات پنل را بروز می‌کند.

Panel Toolbar

ابزاری موجود در این پنل مشابه پنل‌های دیگر در بالای پنل و در زیر تب پنل‌ها قرار دارد و شامل ابزارهای زیر می‌شود:
Edit Button : این ابزار به دو صورت Live Edit و Source Edit در دسترس هست که با کلیک بروی فلش کوچکی که در سمت راست دکمه قرار دارد می‌توان نوع آن را تغییر داد.
انتخاب حالت Source Edit ، باعث می‌شود سورس فایل به همان صورتی که به مرورگر ارسال شده نمایش داده شود. (کامنت‌ها ، فرمت فایل و ... حفظ می‌شود.)
حالت Live Edit هم باعث نمایش محتویات فایل بصورت مرتب شده می‌شود. (کامنت‌ها و دستوراتی که توسط مرورگر تفسیر نشده اند نمایش داده نمی‌شوند.)

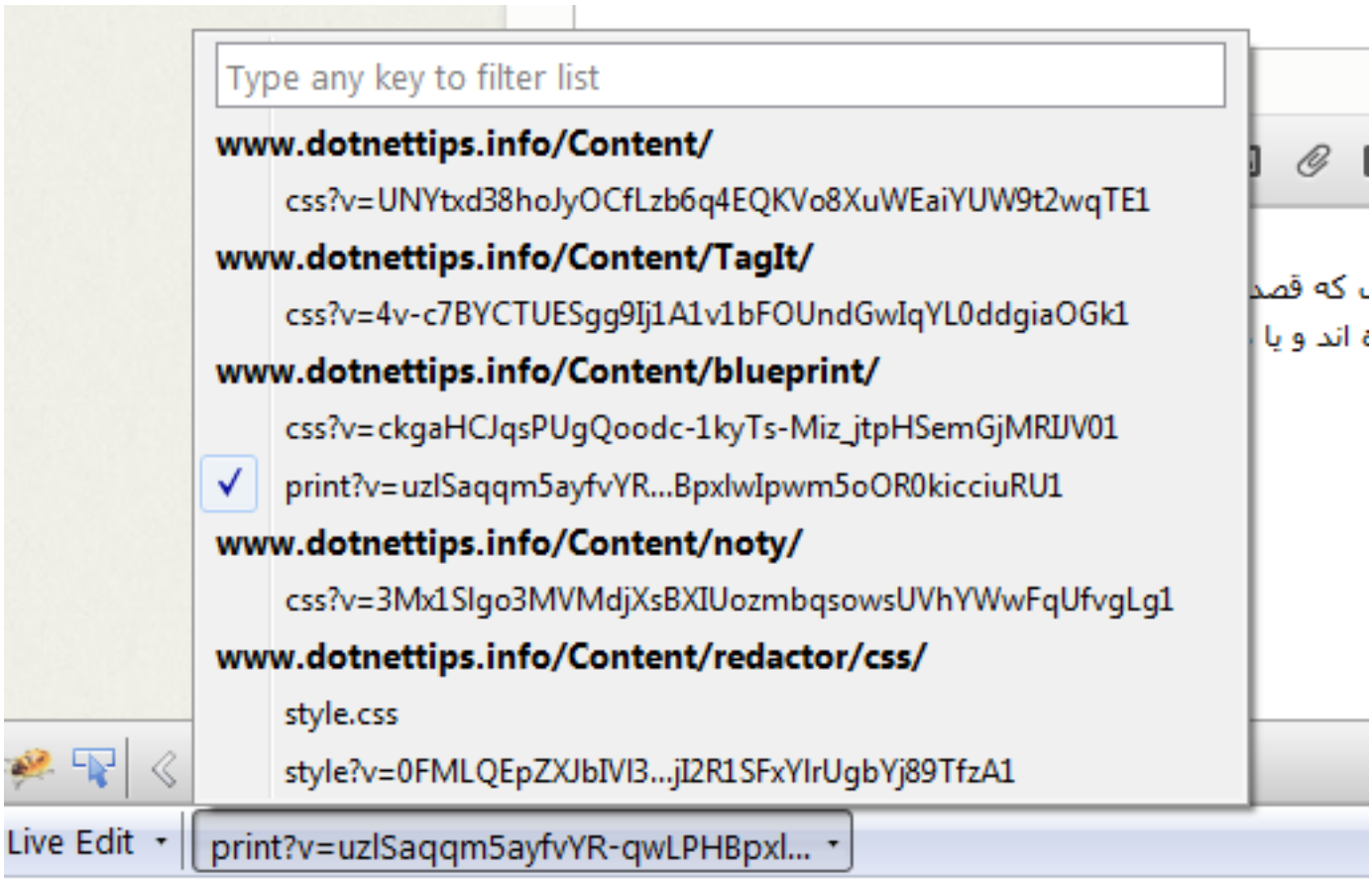


CSS Location Menu : نام فایل جاری که در پنل در حال نمایش است را نمایش می‌دهد و همچنین با کلیک بروی آن ، فایل‌های استایل دیگری که در صفحه بارگزاری شده اند را نمایش می‌دهد. با کلیک بروی هرکدام از فایل‌های نمایش داده شده ، همان فایل در پنل باز می‌شود.

استایل هایی که در خود صفحه تعریف شده اند با نام خود صفحه در این قسمت نمایش داده می‌شوند و اگر استایل‌های در چندین تگ style تعریف شده باشند ، دومین تگ با #2 ، سومین با #3 و ... مشخص می‌شوند.

هنگام باز بودن :

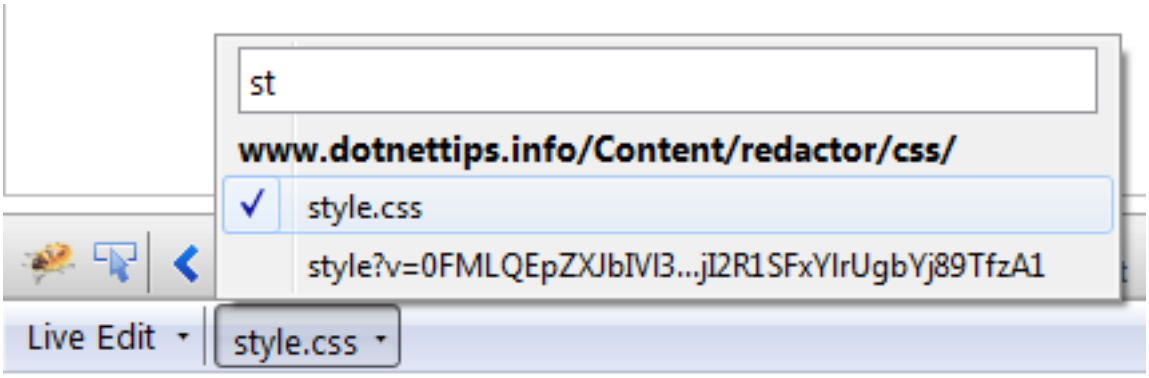
-1 فایل هایی که از پوشه‌های مختلف در صفحه بارگذاری شده اند ، با نام پوشه از هم تفکیک شده اند:




```
body {
    background: none repeat scroll 0 center transparent;
    color: #000000;
    font-family: Tahoma;
    font-size: 10pt;
    line-height: 1.5;
}

.container {
    background: none repeat scroll 0 center transparent;
}
```

-2 با تایپ کردن عبارت دلخواه می‌توان نتایج را محدود کرد:



Panel

منظور از پنل ، قسمتی هست که استایل بصورت فرمت شده و نمایش داده شده است.

Infotips

توسط این قابلیت ، زمانی که موس را بروی آدرس تصاویر ، نوع فونت و ... ببرید ، پاپ آپ کوچکی باز می‌شود و اطلاعاتی در مورد مقادیر می‌دهد. مثلا اینکه آیا تصویر مورد نظر به درستی بارگذاری شده یا نه ، نمای کوچکی از تصویر ، شکل فونت و ...

Editing rules

برای ویرایش تعاریف CSS شامل Selectorها ، دستورات و مقادیر ، این پنل ابزارهای مفیدی ارائه می‌کند.
برای ویرایش یک selector ، دستور یا مقدار آن بروی آن عبارت کلیک کنید ، در همین حال یک text box ظاهر می‌شود و می‌توانید مقدار جدید را وارد کنید. پس از انجام ویرایش مورد نظر بروی قسمتی از صفحه کلیک کنید یا کلید Tab سپس Esc را بزنید. برای ویرایش دستورات بعدی ، کلید Tab را بزنید و برای لغو تغییر جاری ، کلید Esc را بزنید.

برای ایجاد یک rule جدید ، بروی قسمتی از پنل راست کلیک کرده و سپس گزینه‌ی " New Rule... " را برگزینید.
برای ایجاد یک property هم چند راه وجود دارد:
بروی قسمت از فضای خالی تعریف یک استایل دوبار کلیک کنید.
در قسمتی از تعریف یک استایل راست کلیک کرده و گزینه‌ی " New Property... " را انتخاب کنید.

بروی مقدار آخرین property کلیک کرده و کلید Tab را بزنید.

هنگام ویرایش یا ایجاد یک Rule , Propery یا مقدار بصورت inline ، حاشیه‌ی text box متناسب با مقدار وارد شده رنگی را که نمایانگر حالت ذخیره‌ی مقدار وارد شده است نمایش می‌دهد. برای مثال اگر مقداری که برای یک selector وارد شده است نامعتبر باشد ، رنگ حاشیه‌ی text box قرمز می‌شود.
جدول زیر حالت‌های مختلف را شرح می‌دهد:

رنگ حاشیه	Selectorها	نام Properyها	مقادیر Propertyها و Ruleها
خاکستری	تغییری ایجاد نشده است	تغییری ایجاد نشده است	تغییری ایجاد نشده است
قرمز	selector نامعتبر است	نام نامعتبر است	مقدار نامعتبر است
زرد	selector صحیح است اما بروی Element فعلی تاثیر ندارد	نام صحیح است اما مقدار Property غیر مجاز است یا وارد نشده است	
سبز	selector صحیح است	نام و مقدار صحیح هستند	مقدار صحیح است

Auto-completion

زمانی که در حال ایجاد/ویراش کردن یک Rule, Propery یا مقدار آنها هستید ، میتوانید از این قابلیت استفاده کنید. مثلا با وارد کردن # تمام Selector هایی که می‌توانند با # شروع شوند در دسترس شما هستند و با دکمه‌های Up/Down می‌توانید مقادیر ممکن را مرور کنید.

اگر در حال ویرایش یک مقدار عددی هستید ، می‌توانید با دکمه‌های Up/Down مقادیر را یک واحد یک واحد افزایش دهید. با نگه داشتن کلید Shift و فشردن Up/Down می‌توانید مقادیر را 10تا 10تا تغییر دهید و با نگه داشتن Ctrl بجای Sihft ، یک دهم یک دهم.

Toggling styles

با این امکان می‌توانید یک Property را بطور موقت فعال/غیرفعال کنید. با حرکت موس از یک Property یک آیکون قرمز رنگ در کنار آن نمایش داده می‌شود که با کلیک بروی آن ، Property و مقدار آن کمرنگ شده و آیکون قرمز رنگ کنار آن ثابت می‌شود. با کلیک مجدد بروی آن ، Property فعال می‌شود.

```
table.wikitable {
    border-collapse: collapse;
}
table {
    color: black;
    font-size: 100%;
```

Context Menu

این منو زمانی که در پنل راست کلیک کنید ظاهر می‌شود و نسبت به منطقه (Context)ای که در آن راست کلیک کرده اید ، گزینه‌های متفاوتی را مشاهده خواهید کرد. در جدول زیر ، گزینه‌ها ، Contextشان و توضیح هر گزینه آمده است.

گزینه	Context	توضیحات
Copy Location	CSS Location Menu	آدرس فایل استایل را در حافظه کپی می‌کند.
Open in New Tab	CSS Location Menu	فایل استایل را در یک تب جدید باز می‌کند.
Copy Image Location	image values	آدرس تصویر را در حافظه کپی می‌کند.
Open Image in New Tab	image values	تصویر را در یک تب جدید باز می‌کند.
Copy Color	color values	مقدار رنگ را در حافظه کپی می‌کند.
Copy Rule Declaration	CSS selector	Selector و Propertyها را در حافظه کپی می‌کند.
Copy Style Declaration	CSS selector	فقط Propertyها را در حافظه کپی می‌کند.
New Rule...	همه جای پنل	یک Rule جدید بالای قسمتی که راست کلیک شده ایجاد می‌کند.
Delete "<selector>"	CSS selector	Rule را حذف می‌کند.

گزینه	Context	توضیحات
New Property...	CSS rule	یک Property جدید در Ruleی که در آن راست کلیک شده ایجاد می‌کند.
Edit "<property name>"...	CSS property	Property فعلی به حالت ویرایش درمی آید. (راه ساده‌تر ، کلیک بروی Property است.)
Delete "<property name>"...	CSS property	Property فعلی را حذف می‌کند.
Disable "<property name>"...	CSS property	Property فعلی را غیرفعال می‌کند.
Refresh	همه جای پنل	محتویات پنل را بروز رسانی می‌کند.
Inspect in DOM panel	CSS Location Menu, CSS rule	فایل استایل یا Rule را در پنل DOM باز می‌کند.

Elements Side Panel

در این پنل که سمت راست پنل CSS قرار دارد ، با وارد کردن یک CSS Selector می‌توانید Elementهایی که در صفحه با آن مطابقت دارند را مشاهده کنید.

برای وارد کردن CSS Selector هم می‌توان مقدار مورد نظر را در قسمت Try a selector... وارد کرد هم می‌توان بروی یکی از Selectorهای پنل راست کلیک کرد و گزینه‌ی Get Matching Elements را برگزید.

Context Menu این قسمت هم مشابه Context Menu پنل HTML هست و فقط در ورژن 1.11 گزینه‌ی Paste HTML اضافه شده که در این کامنت از مقاله‌ی آموزش فایرباگ - #5 - HTML Panel توضیح داده شده است.

نظرات خوانندگان

نویسنده: راضیه
تاریخ: ۱۳۹۲/۰۱/۰۸ ۲۳:۴۶

عالی بود

عنوان: **Base64 و کاربرد جالب آن**
نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: HTML, Embeded Images, Base64

Base64 یک مبنای عددی است که یکی از کاربردهای آن در نرم افزارها انتقال اطلاعات فایل باینری است. به عنوان مثال با تبدیل محتوای باینری یک تصویر به مبنای 64 میتونید اون تصویر رو در دل فایل هایی نظیر HTML و CSS و SVG قرار بدید؛ یا به اصطلاح اون تصویر رو Embeded (توکار) کنید.

نکته: در Base64 برای هر 6 بیت یک کاراکتر معادل در مبنای 64 در نظر گرفته میشه، به همین دلیل در هنگام تبدیل برای جلوگیری از Lost (گم) شدن داده‌ها عملیات مورد نظر رو بر روی سه بایت داده انجام میدیم. که با این کار برای هر 3 بایت (24 بیت) 4 کاراکتر معادل خواهیم داشت؛ به این ترتیب حجم نهایی فایل تبدیل شده در واحد بایت، برابر است با:

(حجم نهایی) = (3 / حجم کل فایل) + (حجم کل فایل)

و حالا نحوه تبدیل فایل تصویر به Base64:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Image2Base64Converter
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnOpen_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Multiselect = false;
            ofd.Filter = "Pictures |*.bmp;*.jpg;*.jpeg;*.png";

            if (ofd.ShowDialog(this) == DialogResult.OK)
            {
                txtImagePath.Text = ofd.FileName;
            }
        }

        private void btnConvert_Click(object sender, EventArgs e)
        {
            if (!File.Exists(txtImagePath.Text))
            {
                MessageBox.Show("File not found!");
                return;
            }

            btnCopy.Enabled = false;
            btnConvert.Enabled = false;
            txtOutput.Enabled = false;

            BackgroundWorker bworker = new BackgroundWorker();
            bworker.DoWork += (o, a) =>
            {
                string header = "";
                header = string.Format(header, txtImagePath.Text.GetExtension());
```

```
                StringBuilder data = new StringBuilder();

                byte[] buffer;
                BinaryReader head = new BinaryReader(
                    new FileStream(txtImagePath.Text, FileMode.Open));

                buffer = head.ReadBytes(6561);
                while (buffer.Length > 0)
                {
                    data.Append(Convert.ToBase64String(buffer));
                    buffer = head.ReadBytes(6561);
                }

                head.Close();
                head.Dispose();

                this.Invoke(new Action(() =>
                {
                    txtOutput.ResetText();
                    txtOutput.AppendText(header);
                    txtOutput.AppendText(data.ToString());

                    btnCopy.Enabled = true;
                    btnConvert.Enabled = true;
                    txtOutput.Enabled = true;
                }));
            };

            bworker.RunWorkerAsync();
        }

        private void btnCopy_Click(object sender, EventArgs e)
        {
            Clipboard.SetText(txtOutput.Text);
        }
    }

    public static class ExtensionMethods
    {
        public static string GetExtension(this string str)
        {
            string ext = string.Empty;
            for (int i = str.Length - 1; i >= 0; i--)
            {
                if (str[i] == '.')
                {
                    break;
                }

                ext = str[i] + ext;
            }

            return ext;
        }
    }
}
```

توضیح کد:

در خط 44 یک BackgroundWorker تعریف شده است تا عملیات تبدیل در یک ترد جداگانه انجام شود، که در عملیات‌های سنگین ترد اصلی برنامه آزاد باشد.

در خطوط 47 تا 64 کدهای مربوط به تبدیل قرار گرفته است:

در خط 47 و 48 ابتدا یک سرآیند برای معرفی داده‌های تبدیل شده ایجاد میکنیم؛ "" که در ان نوع فایل و پسوند آن را مشخص کرده و سپس الگوریتم به کار گرفته شده برای تبدیل آن را نیز ذکر میکنیم: "data:" + نوع فایل (image) + "/" + پسوند فایل + ";" + الگوریتم تبدیل + ",".

در انتهای کار این سرآیند تولید شده در ابتدای داده‌های تبدیل شده فایل، قرار خواهد گرفت.

در خط 50 یک StringBuilder برای نگهداری اطلاعات تبدیل شده ایجاد کرده ایم.

در خط 52 یک بافر برای خواندن اطلاعات باینری فایل ایجاد کرده ایم.

در خط 53 فایل مورد نظر را برای خواندن باز کرده ایم.

و در خطوط 56 تا 61 فایل مورد نظر را خوانده و پس از تبدیل در متغیر data ذخیره کرده ایم.

در نظر داشته باشید که برای عملکرد صحیح لازم است که عملیات تبدیل بر روی 3 بایت داده انجام شود؛ پس در هر بار خواندن

داده‌های فایل، باید مقدار داده ای که خوانده میشود ضریبی از 3 باشد.

در خطوط 63 و 64 نیز منابع اشغال شده سیستم را آزاد کرده ایم.

در انتها داده‌های مورد استفاده برای Embeded کردن تصویر برابر است با:

header + data.ToString()

embeded-images.htm

: فایل نمونه برای نحوه استفاده از تصویر توکار. Image2Base64-Converter.zip

: سورس کامل برنامه ارائه شده.

نظرات خوانندگان

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۲۰

در کد فوق یک مشکل کوچیک وجود داره که امیدوارم آقای وحید نصیری و یا دیگر دوستان نظر خودشون رو در این رابطه بیان کنن.
مشکل اینه که با اینکه عملیات تبدیل به سرعت انجام میشه، ولی در خط 70 ریختن نتیجه عملیات درون RichTextBox به کندی انجام میشه. برای رفع این مشکل چه کاری میشه انجام داد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۲۹

- مرسوم نیست این همه اطلاعات کد شده رو در یک tetxbox نمایش بدن. به چه دلیلی و چه مقصودی را برآورده می‌کند؟
- برای RichTextBox یک سری متد BeginUpdate و EndUpdate هست.
- در کد فوق می‌شود بجای GetExtension از متد توکار Path.GetExtension استفاده کرد.
- تصاویر وب عموما حجیم نیستند (خصوصا زمانیکه قرار است تبدیل به base64 شوند). یعنی حجم بالای 10 مگابایت ندارند که بخواهید با استریم‌ها کار کنید. بهتر است یک ضرب از متد File.ReadAllBytes استفاده کنید. همچنین در این حالت مشخص (با توجه به حجم پایین تصاویر مورد استفاده در وب سایت‌ها)، استفاده از تردها را هم حذف کنید.
- همیشه زمانیکه کدنویسی می‌کنید این سؤال رو از خودتون بپرسید:
آیا کاری که دارم انجام می‌دم قابلیت استفاده مجدد داره؟ میشه از قسمتی از نتیجه‌اش در یک پروژه دیگر استفاده کرد؟
مثلا در کد شما بهتر است قسمت تبدیل تصویر به معادل base64 آن تبدیل به یک متد کمکی با قابلیت استفاده مجدد شود تا اینکه منطق پیاده سازی آن در بین کدهای UI دفن شده باشد. مطالعه قسمت Refactoring در سایت در این زمینه مفید است.

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۵۳

آقای نصیری ممنون از نظرات خوبتون.
حتما تو کدنویسی‌های آینده نکات مفیدی که گفتید رو اعمال میکنم.

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۲:۴۶

ضمن تشکر از نویسنده عزیز. لازم است کمی در مورد کاربردهای آن بیشتر بحث شود. به طور مثال علت Embed کردن یک تصویر در فایل html.

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۳:۳۴

- خب البته من این رو در حین کار با فایل‌های SVG یاد گرفتم؛ برای اینکه بتونم روی فایل‌های SVG تولید شده توی برنامه ام مدیریت بهتری داشته باشم، دنبال راهی بودم که تصاویر رو درون اون Embed کنم. که با این راه آشنا شدم.

- برای فایل‌های HTML و CSS کاربرد خاصی نمیتونم مثال بزنم؛ شایدم اصلا کاربرد خاصی نداشته باشه!. ولی توی کار با فایل‌های SVG میتونه مفید واقع بشه و خیالتون رو در مورد Lost شدن تصاویر راحت کنه. هرچند که کلا این روش برای تصاویر حجیم مناسب نیست.

نویسنده: مهدی
تاریخ: ۱۳۹۲/۰۲/۰۳ ۲:۵۷

شاید یک کاربرد آن در CSS این هست که تصویر استفاده شده در فایل CSS به همراه سند CSS قابل جابجایی می‌باشد و نیاز نیست که تصویر بر روی هاست Upload شود.

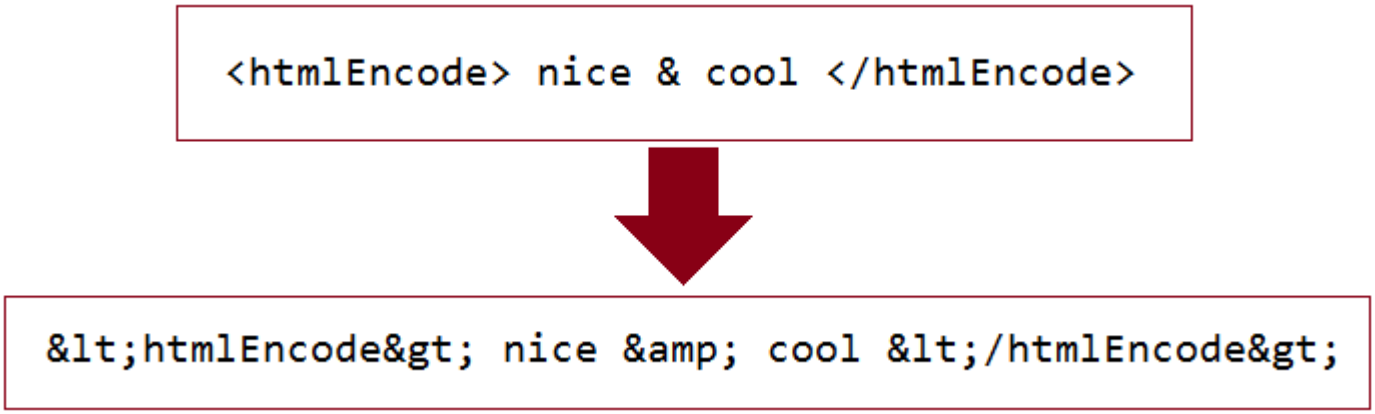
نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۳ ۸:۳۴

یک کاربرد جالب دیگه که به ذهنم رسید اینه که، مثلا فرض کنید میخواید یه برنامه برای آرشیو کردن صفحات وب روی کامپیوتر بنویسید، خب مسلما گنجوندن تمام اطلاعات صفحه وب مورد نظر توی یک فایل، مدیریت اون رو راحت‌تر میکنه.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۰۳ ۹:۱۲

تصاویر base64 از IE8 به بعد پشتیبانی میشه.

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
 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: **4 کاراکتر منحصر بفرد در CSS**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۲/۱۰/۰۴ ۲۳:۵
آدرس: www.dotnettips.info
گروه‌ها: CSS, HTML

در صورتیکه بخواهید برای نسخه‌های مختلف اینترنت اکسپلورر styleهای مختلفی را بنویسید، کافی‌است از 4 کاراکتر منحصر به فرد (\9 _ *) استفاده کنید.

IE8 and Below

برای اینکار در IE8 و پایین‌تر از آن، به انتهای هر استایل "\9" را اضافه کنید. فقط "\9" را می‌توان برای این کار استفاده کرد و تغییر دادن آن به عبارتی دیگر مثلا "\IE" اشتباه است. حتی "\8" هم برای انجام اینکار درست نیست و فقط "\9" کار می‌کند.

```
body {
 color: red; /* all browsers, of course */
 color : green\9; /* IE8 and below */
}
```

IE7 and Below برای اینکه در IE7 به طور مشخص، تغییری ایجاد کنید، کافی‌است ابتدای هر property، یک کاراکتر "*" اضافه کنید.

```
body {
 color: red; /* all browsers, of course */
 color : green\9; /* IE8 and below */
 *color : yellow; /* IE7 and below */
}
```

IE6 و برای اینکه در IE6 به طور مشخص، تغییری ایجاد کنید، کافی‌است ابتدای هر property یک کاراکتر "_" استفاده کنید.

```
body {
 color: red; /* all browsers, of course */
 color : green\9; /* IE8 and below */
 *color : yellow; /* IE7 and below */
 _color : orange; /* IE6 */
}
```