عنوان: Zen Coding در Visual Studio 2012
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۵/۲۲ ۱۸:۵
آدرس: www.dotnettips.info
برچسب‌ها: Visual Studio 2012, Zen Coding

Zen Coding روشی سریع برای نوشتن کدهای HTML با استفاده از گرامر selectorهای CSS است. این روش برای اولین بار در سال 2009 توسط Sergey Chikuyonok معرفی شد و در طول این مدت، تبدیل به روشی عالی برای نوشتن کدهای HTML تکراری و یکنواخت شده است.

برای استفاده از این روش در ویژوال استادیو 2012، احتیاج به نصب افزونه‌ی محبوب و پر طرفدار Web Essentials است. این افزونه که توسط Mads Kristensen توسعه پیدا کرده است،؛ علاوه بر Zen Coding ، ویژگی‌های بسیار زیادی در زمینه‌ی توسعه وب به ویژوال استادیو اضافه می‌کند و توصیه‌ی اکید دارم که حتما این افزونه‌ی فوق العاده را نصب کنید و از امکانات دیگر آن بهره برید.

گرامر Zen Coding، یک سری عملگر دارد که مرجعی از همه‌ی آن‌ها را در زیر مشاهده می‌کنید.

ایجاد خاصیت id

. ایجاد خاصیت class

[] ایجاد خاصیت دلخواه

> ایجاد عنصر فرزند

+ ایجاد عنصر برادر یا خواهر (Sibling)

^ رجوع به والد

* تکثیر کننده‌ی عنصر است. هر عنصری را می‌تواند n بار تکرار کند

$ با شمارنده جایگزین می‌شود

$$ با شمارنده دارای padding جایگزین می‌شود

{} متن داخل آکولاد را در عنصر مورد نظر قرار می‌دهد

() امکان گروه بندی عبارات را می‌دهد.

lorem اطلاعات ساختگی برای آزمایش برنامه ایجاد می‌کند

مشخص است که بسیاری از عملگرها با selectorهای CSS یکسان اند. قبل از اینکه به بررسی دقیق‌تر هر کدام از این عملگرها بپردازیم، دو مثال را با هم بررسی می‌کنیم تا به قدرت آن پی ببریم.

پس از نصب افزونه‌ی Web Essentials ، یک صفحه‌ی html یا cshtml و یا هر قسمتی که بتوان در آن کد html نوشت را گشوده و سپس کد زیر را در آن بنویسید و در نهایت کلید Tab را فشار دهید.

```
table#tblUsers.table.table-striped.table-bordered.table-hover>thead>tr>th{row}+th{Name}+th{Last Name}+th{Operations}^^tbody>tr*6>(td{row}+td{FirstName}+td{LastName}+td>button.btn.btn-primary{Edit}+button.btn.btn-danger{Delete})
```

پس از فشردن کلید Tab، خروجی زیر را مشاهده خواهید کرد:

```
<table id="tblUsers" class="table table-striped table-bordered table-hover">
        <thead>
            <tr>
                <th>row</th>
                <th>Name</th>
                <th>Last Name</th>
                <th>Operations</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>row</td>
                <td>FirstName</td>
                <td>LastName</td>
                <td>
                    <button>Edit</button>
                    <button>Delete</button></td>
            </tr>
            <tr>
```

```
                <td>row</td>
                <td>FirstName</td>
                <td>LastName</td>
                <td>
                    <button class="btn btn-primary">Edit</button>
                    <button class="btn btn-danger">Delete</button></td>
            </tr>
            <tr>
                <td>row</td>
                <td>FirstName</td>
                <td>LastName</td>
                <td>
                    <button class="btn btn-primary">Edit</button>
                    <button class="btn btn-danger">Delete</button></td>
            </tr>
            <tr>
                <td>row</td>
                <td>FirstName</td>
                <td>LastName</td>
                <td>
                    <button class="btn btn-primary">Edit</button>
                    <button class="btn btn-danger">Delete</button></td>
            </tr>
            <tr>
                <td>row</td>
                <td>FirstName</td>
                <td>LastName</td>
                <td>
                    <button class="btn btn-primary">Edit</button>
                    <button class="btn btn-danger">Delete</button></td>
            </tr>
            <tr>
                <td>row</td>
                <td>FirstName</td>
                <td>LastName</td>
                <td>
                    <button class="btn btn-primary">Edit</button>
                    <button class="btn btn-danger">Delete</button></td>
            </tr>
        </tbody>
    </table>
```

به نظر پیچیده می‌آید و حتما با خودتان می‌گویید که همان HTML خام را بنویسم، راحت‌تر هستم. شاید الان برای شما پیچیده به نظر آید ولی اگر تا انتهای این مقاله را مطالعه کنید، این قول را به شما می‌دهم که برای شما نوشتن این گونه کدها بسیار ساده خواهد بود و بعید به نظر می‌آید که شما دیگر HTML را به شیوه‌های قدیمی بنویسید، همان طور که بنده نیز با چند بار آزمون و خطا، توانستم کد بالا را بنویسیم.

یک مثال دیگر می‌تواند به این شکل باشد:

```
ul>li[ng-repeat="user in Users"]>p[ng-bind="{{user.UserName}}"]+a{Details}
```

که خروجی زیر را در بر دارد:

```
<ul>
        <li ng-repeat="user in Users">
            <p ng-bind="{{user.UserName}}"></p>
            <a href="">Details</a>
        </li>
</ul>
```

از کدهای بالا این نتیجه را می‌توان گرفت که Zen Coding؛ بسیار انعطاف پذیر، ساده و قدرتمند است.

در ادامه نگاهی دقیق‌تر به عملگرهای استفاده شده در کدهای بالا می‌اندازیم.

خواص id (#) و class (.)

به مانند گرامر CSS، به راحتی می‌توانید id یا class خاصی را به عنصر مورد نظر نسبت دهید:

```
<!-- Type this -->
```

```
div.container

<!-- Creates this -->
<div class="container"></div>
```

```
<!-- Type this -->
ul#userList

<!-- Creates this -->
<ul id="userList"></ul>
```

```
<!-- Type this -->
div.container#wrapper

<!-- Creates this -->
<div class="container" id="wrapper"></div>
```

```
<!-- Type this -->
button.btn.btn-primary

<!-- Creates this -->
<button class="btn btn-primary"></button>
```

نکته: اگر شما عنصر مورد نظری را که می‌خواهید بر روی آن کلاس و یا آیدی را اعمال کنید، مشخص نکنید به طور پیش فرض، Zen Coding آن عنصر را div در نظر می‌گیرد.
مثال:

```
<!-- Type this -->
.container

<!-- Creates this -->
<div class="container"></div>
```

> عناصر فرزند (Child Elements)

با استفاده از عملگر (>)، عنصر مورد نظر و عناصر فرزند درون آن را می‌توانید ایجاد کنید.

```
<!-- Type this -->
div.container>div#header

<!-- Creates this -->
<div class="container">
        <div id="header"></div>
</div>
```

```
<!-- Type this -->
div.navbar>div.navbar-inner>ul.navbar

<!-- Creates this -->
<div class="navbar">
        <div class="navbar-inner">
            <ul class="navbar"></ul>
        </div>
</div>
```

[] خواص سفارشی (Custom Attributes)

با نوشتن خاصیت سفارشی خود درون براکت []، به راحت می‌توانید آن عنصر و خاصیت سفارشی مورد نظر خود را ایجاد کنید.

```
<!-- Type this -->
div[title]

<!-- Creates this -->
<div title=""></div>
```

و یا اگر بخواهید چندین خاصیت را به صورت همزمان اعمال کنید:

```
<!-- Type this -->
input[type="text" placeholder="First Name"]

<!-- Creates this -->
<input type="text" value="" placeholder="First Name" />
```

```
<!-- Type this -->
ul[ng-repeat="user in Users"]>li[ng-model="user.FirstName"]

<!-- Creates this -->
<ul ng-repeat="user in Users">
        <li ng-model="user.FirstName"></li>
</ul>
```

{} متن (Text)

این عملگر، متن مورد نظر شما را داخل عنصر قرار می‌دهد.

```
<!-- Type this -->
div>h1{My Title}

<!-- Creates this -->
<div>
      <h1>My Title</h1>
</div>
```

+ عناصر خواهر و برادر (Siblings Elements)

با قرار دادن عملگر + ، عناصر خواهر و برادر را ایجاد کنید.

```
<!-- Type this -->
form>input[type="text"]+input[type="checkbox"]

<!-- Creates this -->
<form>
        <input type="text" value="" />
        <input type="checkbox" value="" />
</form>
```

```
<!-- Type this -->
div#header>img+h1{Title}

<!-- Creates this -->
<div id="header">
        <img src="" alt="" />
        <h1>Title</h1>
</div>
```

^ بالا رفتن از عناصر (Climbing Elements)

این عملگر تقریبا برعکس عملگر > عمل می‌کند. با عملگر ^ شما می‌توانید به والد یک عنصر تا هر تعداد سطحی که بخواهید برسید. اگر شما به طور مثال بخواهید که دو سطح به والد بالاتر برگردید، باید از ^^ استفاده کنید.

```
<!-- Type this -->
table>thead>tr>th{row}^^tbody>tr>td{row1}

<!-- Creates this -->
<table>
        <thead>
```

```
            <tr>
                <th>row</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>row1</td>
            </tr>
        </tbody>
</table>
```

*** تکثیر عناصر (Elements Multiplication)**

با عملگر *، هر تعداد عنصر را که می‌خواهید تکثیر کنید.

```
<!-- Type this -->
ul>li*3>p{Hello}

<!-- Creates this -->
<ul>
        <li>
            <p>
                Hello
            </p>
        </li>
        <li>
            <p>
                Hello
            </p>
        </li>
        <li>
            <p>
                Hello
            </p>
        </li>
</ul>
```

```
<!-- Type this -->
(div.container>h1{title}+div.content{Some Text Here})*2

<!-- Creates this -->
<div class="container">
        <h1>title</h1>
        <div class="content">
            Some Text Here
        </div>
    </div>
    <div class="container">
        <h1>title</h1>
        <div class="content">
            Some Text Here
        </div>
</div>
```

$ شماره گذاری آیتم‌ها (Item Numbering)

هنگامی که شما با کمک *، از روی یک عنصر n عنصر را تکثیر می‌کنید، می‌توانید با استفاده ازعملگر $ ، به عدد متناظر با آن حلقه‌ی تکرار دست پیدا کنید.

```
<!-- Type this -->
ul>li*2>p{item $}

<!-- Creates this -->
<ul>
        <li>
            <p>
                item 1
            </p>
        </li>
        <li>
```

```
            <p>
                item 2
            </p>
        </li>
</ul>
```

نکته: استفاده از چند عملگر $، باعث قرارگرفتن همان تعداد صفر، پیش از عدد می‌شود.

```
<!-- Type this -->
ul>li*2>p{item $$$}

<!-- Creates this -->
<ul>
        <li>
            <p>
                item 001
            </p>
        </li>
        <li>
            <p>
                item 002
            </p>
        </li>
</ul>
```

() گروه بندی (Grouping)

گروه بندی یکی از امکانات فوق العاده‌ی Zen Coding است که به شما امکان نوشتن عبارات پیچیده را می‌دهد. کاربرد آن در مثال زیر کاملا واضح است، و به راحتی با یک عبارت، کل ساختار Dom را پیاده سازی شده است.

```
<!-- Type this -->
div>(header>div)+section>(ul>li*2>a)+footer>(div>span)

<!-- Creates this -->
<div>
        <header>
            <div></div>
        </header>
        <section>
            <ul>
                <li><a href=""></a></li>
                <li><a href=""></a></li>
            </ul>
            <footer>
                <div>
                    <span></span>
                </div>
            </footer>
        </section>
</div>
```

Lorem داده‌های ساختگی (Dummy Data)

حتما بارها و بارها این اتفاق افتاده است که برای تست قالب برنامه‌ی خود، آن را با داده‌های ساختگی پر کرده باشید. معمولا شما این کار را به صورت دستی انجام می‌دادید. اما این بار عبارت lorem را نوشته و کلید Tab را فشار دهید. به طور پیش فرض برای شما 30 کلمه می‌نویسد. اگر تعداد مشخصی کلمه می‌خواهید کافی است تعداد آن را جلوی کلمه‌ی lorem بنویسید. برای مثال lorem4.

حتی از lorem می‌توانید در عبارات Zen Coding استفاده کنید:

```
<!-- Type this -->
div>(h1>lorem5)+(h3>lorem3)

<!-- Creates this -->
<div>
      <h1>Lorem ipsum dolor sit amet.</h1>
      <h3>Lorem ipsum dolor.</h3>
</div>
```

نظرات خوانندگان

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۵/۲۳ ۷:۵۸

با عرض پوزش؛ گویا در انتقال کدها به سایت، کدهای html دارای خاصیت class حذف شده بودند، که الان به صورتی دستی اصلاح کردم.

نویسنده: emad
تاریخ: ۱۳۹۲/۰۵/۲۳ ۱۳:۴۵

http://visualstudiogallery.msdn.microsoft.com/6ed4c78f-a23e-49ad-b5fd-369af0c2107f

اینم لینک برا vs2010

نویسنده: R.M
تاریخ: ۱۳۹۲/۰۵/۲۳ ۱۹:۵۷

ظاهرا با Resharper سازگار نیست. درسته؟

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۵/۲۳ ۲۳:۱۴

مشکلش دقیقا چیه؟
من که با visual studio2012 update3 و resharper 8 مشکلی ندارم.
احتمالا از نسخه‌ی reshaper هست. نسخه‌ی 8 نباید مشکلی داشته باشه.