regex-selectors

> CSS3 adds complex attribute selectors that allow you to perform regular expression matches on attribute values. These types of selectors have performance implications, as regular expression matching is slower than simple class-based matching. In many cases, the attribute selectors aren't providing enough value over simply adding an additional class to the element in question. There are several types of attribute selectors to be mindful of.

The first attribute selector is actually *not* a performance issue as it simply determine if the attribute is present. For example, the following applies only when an href property is specified on an <a> element:

```
//OK
a[href] {
    color: red;
}
```

This attribute selector is okay to use and shouldn't cause any performance issues.

The second attribute selector that is okay to use applies the style only when an attribute value matches exactly. For example, the following applies only when the rel attribute of an <a> element is "external":

```
//OK
a[rel=external] {
    color: blue;
}
```

After these two, the rest of the attribute selectors cause performance issues. Each of the selectors has the same basic format, using square braces after an element name and a special character preceding the equals sign to perform a type of regular expression.

Contains

The first of the problematic selectors is the contains selector. This selector uses *= and matches an element if the attribute contains the given string. This works similar to the JavaScript indexOf() of method in that it matches anywhere in the string. For example:

```
a[href*=yahoo.com] {
    color: green;
}
```

This selector matches any <a> element whose href attribute contains the string "yahoo.com". That means it will match any of the following:

```
<a href="http://www.yahoo.com/">Yahoo!</a>

<a href="http://www.google.com/redirect=www.yahoo.com">Redirect to Yahoo!</a>

<a href="http://login.yahoo.com/">Login to Yahoo!</a>
```

Note that it doesn't matter whether or not the string has white space on either side, it's just a substring match.

Starts With

The next selector to avoid is the starts with match. This uses the ^= operator and matches only when the attribute value begins with the given string. For example:

```
a[rel^=ext] {
    color: red;
}
```

This rule will match any of the following:

```
<a href="http://www.example.com" rel="external">Example</a>

<a rel="extra">Extra! Extra!</a>

<a rel="extreme">Extreme</a>
```

All the selector cares is that the string "ext" appears at the beginning of the attribute value of rel.

Ends With

The next selector to avoid is the ends with match. This uses the $= operator and matches only when the attribute value ends with the given string. For example:

```
a[href$=.html] {
    color: blue;
}
```

This rule matches all <a> elements that have an href attribute value ending in .html. So the following all match:

```
<a href="index.html">Home</a>

<a href="http://www.example.com/example.html">Example</a>
```

Word Match

Getting even more complex is the selector that checks for a value separated by white space. The ~= operator is used to specify the attribute value must contain the given word, meaning that it must either be the entire attribute value or part of a space-separated list of values. For example:

```
a[rel~=external] {
    color: red;
}
```

This rule matches any of the following:

```
<a href="http://www.example.com" rel="external">Example</a>

<a href="http://www.example.com" rel="external me">Example</a>

<a href="http://www.example.com" rel="reference external">Example</a>

<a href="http://www.example.com" rel="friend external me">Example</a>
```

Contains With Dashes

The last problematic selector checks to see if the attribute value contains a string separated by dashes. The |= operator is used to find a substring inside of a string with the format xxx-yyy-zzz. For example:

```
a[data-info|=name] {
    color: red;
}
```

This matches all of the following:

```
<a data-info="name-address-phone">Info</a>

<a data-info="address-name-phone">Info</a>

<a data-info="address-phone-name">Info</a>
```

Performance Issues

All of these complex attribute selectors need to perform regular expression matches on attribute values over and over again to ensure the correct visual display is achieved. Doing so slows down the overall page performance as the CSS calculation takes more time.

Rule Details

Rule ID: regex-selectors

This rule is aimed at flagging selectors that have the potential to cause performance issues. As such, this rule warns when a selector is found using *=, |=, ^=, $=, or ~=.

The following patterns are considered warnings:

```
.mybox[class~=xxx]{
    color: red;
}

.mybox[class^=xxx]{
    color: red;
}

.mybox[class|=xxx]{
    color: red;
}

.mybox[class$=xxx]{
    color: red;
}

.mybox[class*=xxx]{
    color: red;
}
```

The following patterns are considered okay and do not cause warnings:

```
.mybox[class=xxx]{
    color: red;
}

.mybox[class]{
    color: red;
```

```
}
```

Further Reading

- The Skinny on CSS Attribute Selectors