CSS namespaces (If you're not scared of XML)

I've been looking at the CSS in JS debtate for a while and one of the things that still surprises me if that people go to all these lengths to create and use namespaces for their elements when there's a way to do it already baked in the platform.

If you're not allergic to XML and you've been around the web for a while you know what XML is and you may be one of those who have developed phobia to XML (you have an x-girlfriend) to not want to use it and do everything in JSON.

But there are times when XML is necessary and I believe this is one of those situations. XML Namespaces provide an easy way to disambiguate elements with the same name.

A namespace is a unique URI (Uniform Resource Locator) that qualifies and disambiguates elements. The URI must be to a valid domain but it doesn't have to point to anything concrete. In the example below we create 2 namespaces, a default one for `(x)html` and one for `svg`.

Granted, this is a contrived example because in HTML5 known foreign elements are automatically assigned to their respective namespaces. This means that HTML elements will act as though they are in the XHTML namespace (http://www.w3.org/1999/xhtml), and the `<svg>` and `<math>` elements will be assigned their proper namespaces (http://www.w3.org/2000/svg and http://www.w3.org/1998/Math/MathML).

As contrived example as this is, it illustrates how we can use Namespaces in CSS.

```
@namespace url(http://www.w3.org/1999/xhtml);
@namespace svg url(http://www.w3.org/2000/svg);

/*
  This matches all XHTML <a> elements, as XHTML
  is the default unprefixed namespace
*/
```

```
a {}

/*
  This matches all SVG <a> elements
*/
svg|a {}

/*
  This matches both XHTML and SVG <a> elements
*/
*|a {}
```

I usually create a special domain to park namespaces and other esoteric elements like applications that need their own namespace or that point to a specific DTD or XML Schema.

For my current domain the site is https://ns.rivendellweb.net. The site is a legal one and it can have as many namespaces attached to it as we need. Take a look at the example below:

```
@namespace svg url(http://www.w3.org/2000/svg);
@namespace media url(https://ns.rivendellweb.net/media/);
@namespace grid url(https://ns.rivendellweb.net/grid/);

/*
  This element is implicitly in the XHTML namespace
*/
body {
  color: blue
}

/*
  This class is in the media namespace
*/
media|div.container {
  margin: 0 auto;
}
```

```
/*
  This class is in the grid namespace

  Eventhough it has the same name as the one above
  it's treated differently because it's in a separate
  namespace.
*/
grid|div.container {
  display:grid;
}
```

Before you jump in and say that this is not going to work let's look at how it works. We'll use the same style sheet we created above and the markup shown in the example below:

```
<doctype html
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:svg="http://www.w3.org/2000/svg"
  xmlns:media="https://ns.rivendellweb.net/media/"
  xmlns:grid="https://ns.rivendellweb.net/grid/"
  xmlns:mathml="http://www.w3.org/1998/Math/MathML">
  <head>
    <title>My awesome page</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>

    <h1>The page title</h1>

    <media:div class="container">
      <video src="video/my-video.mp4">
    </div>

    <grid:div class="container">
      <p>Grid content will go here</p>
    </div>
```

```
  </body>
</html>
```

The idea is that in both markup and CSS we prefix the content with namespaces that will only match when both the namespace and the element match. That's why we can get away with having two container elements that do completely different things... they are in different namespaces so we can get away with this.

To ensure the content validates we declare the namespace prefixes in the root element of the page, in this case the `html` element.

A side benefit of using namespaces this way is that we can shorten our selectors. Rather than use classes to describe the selector as we do with BEM we can declare additional namespaces to structure our content.

The one downside I see is the verbosity we must use when creating new classes and elements in a namespace. It's not enough to write `div.container` to indicate a div with a class of container. Unless we're working in the html/xhtml namespace we must qualify the element like `media|div.container` or `grid|div.container` in css and `<media:div class="container">` or `<grid:div class="container">`. If you're totally into squeezing the most of your compressed content this may not work as it adds bytes that you may consider unnecessary.

Other developers may consider the prefix requirement to be a downside. As I mentioned earlier, I don't have an issue with using prefixes as I've worked extensively with XML so namespaces are ok with me.

The final issue to consider is that using namespaces other than a few **namespace+attribute** combinations described in the HTML5 Specification syntax section (section 8.1.2.3) requires you to write documents using XHTML syntax. I'm ok with this but may bug you if you think that tag soup documents are ok (I don't).